7.5

IBM Message Service Client for .NET

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen” auf Seite 265 gelesen werden.

Diese Ausgabe bezieht sich auf Version 7 Release 5 von IBM® WebSphere MQ und auf alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Wenn Sie Informationen an IBMsenden, erteilen Sie IBM ein nicht ausschließliches Recht, die Informationen in beliebiger Weise zu verwenden oder zu verteilen, ohne dass eine Verpflichtung für Sie entsteht.

Inhaltsverzeichnis

Einführung in Message Service Client for .NET

Message Service Client for .NET stellt eine Anwendungsprogrammierschnittstelle (API) mit dem Namen XMS bereit, die dieselbe Gruppe von Schnittstellen wie Java Message Service (JMS) hat. API. Message Service Client for .NET enthält eine vollständig verwaltete Implementierung von XMS, die von jeder .NET-kompatiblen Sprache verwendet werden kann.

XMS unterstützt Folgendes:

- Punkt-zu-Punkt (Point-to-point)-Messaging
- Publish/Subscribe-Messaging
- Synchrone Nachrichtenübermittlung
- Asynchrone Nachrichtenübermittlung

Die Eine XMS -Anwendung kann Nachrichten mit den folgenden Anwendungstypen austauschen:

- Eine XMS -Anwendung
- IBM WebSphere MQ -Klassen für JMS-Anwendung
- Eine native IBM WebSphere MQ -Anwendung
- JMS-Anwendung, die den Standard-Messaging-Provider von WebSphere verwendet

Eine XMS-Anwendung kann eine Verbindung zu folgenden Messaging-Servern herstellen und deren Ressourcen verwenden:

IBM WebSphere MQ-Warteschlangenmanager
Die Anwendung kann die Verbindung entweder im Bindungsmodus oder im Clientmodus herstellen.

WebSphere Application Server Service Integration Bus
Die Anwendung kann eine direkte TCP/IP-Verbindung oder HTTP über TCP/IP verwenden.

WebSphere Event Broker oder WebSphere Message Broker
Der Transport der Nachrichten zwischen der Anwendung und dem Broker erfolgt mithilfe von WebSphere MQ Real-Time Transport. Die Übermittlung der Nachrichten zur Anwendung kann mit WebSphere MQ Multicast Transport erfolgen.

Wenn Sie eine Verbindung zu einem IBM WebSphere MQ -Warteschlangenmanager herstellen, kann die Eine XMS -Anwendung IBM WebSphere MQ Enterprise-Transport für die Kommunikation mit WebSphere Event Broker oder WebSphere Message Brokerverwenden. Alternativ kann die Eine XMS -Anwendung veröffentlichen und subskribieren, indem sie eine Verbindung zu IBM WebSphere MQherstellt.

Zugehörige Konzepte

„Messaging-Stile” auf Seite 7

„Das XMS-Objektmodell” auf Seite 7
Die XMS-API ist eine objektorientierte Schnittstelle. Das XMS-Objektmodell basiert auf dem Objektmodell von JMS 1.1.

„Das XMS-Nachrichtenmodell” auf Seite 10
Das XMS-Nachrichtenmodell ist identisch mit dem Nachrichtenmodell von IBM WebSphere MQ -Klassen für JMS.

Einführung in Message Service Client for .NET

Message Service Client for .NET stellt eine Anwendungsprogrammierschnittstelle (API) mit dem Namen XMS bereit, die dieselbe Gruppe von Schnittstellen wie Java Message Service (JMS) hat. API. Message Service Client for .NET enthält eine vollständig verwaltete Implementierung von XMS, die von jeder .NET-kompatiblen Sprache verwendet werden kann.

XMS unterstützt Folgendes:

- Punkt-zu-Punkt (Point-to-point)-Messaging

- Publish/Subscribe-Messaging
- Synchrone Nachrichtenübermittlung
- Asynchrone Nachrichtenübermittlung

Die Eine XMS -Anwendung kann Nachrichten mit den folgenden Anwendungstypen austauschen:

- Eine XMS -Anwendung
- IBM WebSphere MQ -Klassen für JMS-Anwendung
- Eine native IBM WebSphere MQ -Anwendung
- JMS-Anwendung, die den Standard-Messaging-Provider von WebSphere verwendet

Eine XMS-Anwendung kann eine Verbindung zu folgenden Messaging-Servern herstellen und deren Ressourcen verwenden:

IBM WebSphere MQ-Warteschlangenmanager
Die Anwendung kann die Verbindung entweder im Bindungsmodus oder im Clientmodus herstellen.

WebSphere Application Server Service Integration Bus
Die Anwendung kann eine direkte TCP/IP-Verbindung oder HTTP über TCP/IP verwenden.

WebSphere Event Broker oder WebSphere Message Broker
Der Transport der Nachrichten zwischen der Anwendung und dem Broker erfolgt mithilfe von WebSphere MQ Real-Time Transport. Die Übermittlung der Nachrichten zur Anwendung kann mit WebSphere MQ Multicast Transport erfolgen.

Wenn Sie eine Verbindung zu einem IBM WebSphere MQ -Warteschlangenmanager herstellen, kann die Eine XMS -Anwendung IBM WebSphere MQ Enterprise-Transport für die Kommunikation mit WebSphere Event Broker oder WebSphere Message Brokerverwenden. Alternativ kann die Eine XMS -Anwendung veröffentlichen und subskribieren, indem sie eine Verbindung zu IBM WebSphere MQherstellt.

Zugehörige Konzepte

„Messaging-Stile” auf Seite 7

„Das XMS-Objektmodell” auf Seite 7
Die XMS-API ist eine objektorientierte Schnittstelle. Das XMS-Objektmodell basiert auf dem Objektmodell von JMS 1.1.

„Das XMS-Nachrichtenmodell” auf Seite 10
Das XMS-Nachrichtenmodell ist identisch mit dem Nachrichtenmodell von IBM WebSphere MQ -Klassen für JMS.

Neuerungen in diesem Release

In diesem Release von Message Service Client for .NETgibt es eine Reihe von Erweiterungen.

„Über eine Anwendung für einen Message Service Client for .NET einen Nachrichtendeskriptor lesen und schreiben” auf Seite 89
Mit Ausnahme von `StrucId` und `Version`können Sie auf alle Nachrichtendeskriptorfelder in einer IBM WebSphere MQ -Nachricht zugreifen. Das Feld `BackoutCount` kann gelesen, aber nicht beschrieben werden. Zugriff auf die Felder ist nur verfügbar, wenn eine Verbindung zu einem Warteschlangenmanager der IBM WebSphere MQ Version 6 und höher hergestellt wird. Der Zugriff wird durch die später beschriebenen Zieleigenschaften gesteuert.

„Beispielanwendungen” auf Seite 19
Die XMS -Beispielanwendungen bieten einen Überblick über die allgemeinen Funktionen der einzelnen APIs. Sie können sie verwenden, um Ihre Installation und Messaging-Server-Konfiguration und Ihre eigenen Anwendungen zu überprüfen.

Leistungsverbesserungen
Die Leistung von XMS .NET wurde verbessert.

„Automatische IBM WebSphere MQ-Clientverbindungswiederholung über XMS” auf Seite 48

Sie können einen WebSphere MQ V7.1 XMS IBM WebSphere MQ -Client so konfigurieren, dass die Verbindung nach einem Netz-, Warteschlangenmanager-oder Serverausfall automatisch wiederhergestellt wird.

„Verwaltete IBM WebSphere MQ-XA-Transaktionen über XMS" auf Seite 43
Verwaltete WebSphere MQ XA-Transaktionen können über XMSverwendet werden.

GMO-KONVERTIERUNG
Die Angabe eines Werts für `GMO_CONVERT` in einer Nachricht ist optional. Wenn ein Wert für `GMO_CONVERT` angegeben wird, erfolgt die Konvertierung gemäß dem angegebenen Wert.

Messaging-Stile

XMS unterstützt die Messaging-Stile Punkt-zu-Punkt (Point-to-point) und Publish/Subscribe.

Messaging-Stile werden auch als Messagingdomänen bezeichnet.

Punkt-zu-Punkt (Point-to-point)-Messaging

Eine gängige Form des Punkt-zu-Punkt (Point-to-point)-Messagings arbeitet mit Warteschlangensteuerung. Im einfachsten Fall sendet eine Anwendung eine Nachricht an eine andere Anwendung gibt dabei implizit oder explizit eine Zielwarteschlange an. Das zugrunde liegende Messaging- und Warteschlangensystem empfängt die Nachricht von der sendenden Anwendung und leitet sie an die Zielwarteschlange weiter. Die empfangende Anwendung kann die Nachricht dann aus der Warteschlange abrufen.

Wenn das zugrunde liegende Messaging- und Warteschlangensystem WebSphere Message Broker enthält, kann WebSphere Message Broker die Nachricht replizieren und Kopien der Nachricht an unterschiedliche Warteschlangen senden. Auf diese Weise kann mehr als eine Anwendung die Nachricht empfangen. WebSphere Message Broker kann eine Nachricht auch umwandeln und ihr weitere Daten hinzufügen.

Ein Schlüsselmerkmal des Punkt-zu-Punkt (Point-to-point)-Messagings ist, dass eine Anwendung eine Nachricht in eine lokale Warteschlange stellt, wenn sie eine Nachricht sendet. Das zugrunde liegende Messaging- und Warteschlangensystem bestimmt dann, an welche Zielwarteschlange die Nachricht gesendet wird. Die empfangende Anwendung ruft schließlich die Nachricht aus der Zielwarteschlange ab.

Publish/Subscribe-Messaging

Im Publish/Subscribe-Messaging gibt es zwei Anwendungstypen: Publisher und Subskribent.

Ein *Publisher* stellt Informationen in Form einer Veröffentlichungsnachricht bereit. Bei der Veröffentlichung einer Nachricht gibt der Publisher ein Thema an, auf das sich die Informationen in der Nachricht beziehen.

Ein *Subskribent* konsumiert die bereitgestellten Informationen. Ein Subskribent gibt die Themen an, zu denen er Nachrichten empfangen möchte, indem er Subskriptionen (Abonnements) erstellt.

Das Publish/Subscribe-System erhält Veröffentlichungen von Publishern und Subskriptionen von Subskribenten. Das System leitet die Veröffentlichungen an die Subskribenten weiter. Ein Subskribent erhält nur Veröffentlichungen zu den Themen, die er abonniert hat.

Ein Schlüsselmerkmal des Publish/Subscribe-Messagings ist, dass ein Publisher ein Thema angibt, wenn er eine Nachricht veröffentlicht. Er gibt jedoch nicht die Subskribenten an. Wenn eine Nachricht zu einem Thema veröffentlicht wird, für das es keine Subskribenten gibt, empfängt keine Anwendung diese Nachricht.

Eine Anwendung kann sowohl als Veröffentlichungsanwendung als auch als Subskribent fungieren.

Das XMS-Objektmodell

Die XMS-API ist eine objektorientierte Schnittstelle. Das XMS-Objektmodell basiert auf dem Objektmodell von JMS 1.1.

Die folgende Liste fasst die wichtigsten XMS-Klassen beziehungsweise -Objekttypen zusammen:

ConnectionFactory
Ein ConnectionFactory-Objekt enthält eine Gruppe von Parametern für eine Verbindung. Eine Anwendung verwendet ein ConnectionFactory-Objekt zum Erstellen einer Verbindung. Eine Anwendung kann die Parameter während der Laufzeit bereitstellen und basierend darauf ein ConnectionFactory-Objekt erstellen. Alternativ dazu können die Verbindungsparameter auch in einem Repository für verwaltete Objekte gespeichert werden. Eine Anwendung kann ein Objekt aus dem Repository abrufen und basierend darauf ein ConnectionFactory-Objekt erstellen.

Connection
Ein Connection-Objekt enthält eine aktive Verbindung von einer Anwendung zu einem Messaging-Server. Eine Anwendung verwendet eine Verbindung, um Sitzungen zu erstellen.

Destination
Eine Anwendung sendet oder empfängt Nachrichten mithilfe eines Destination-Objekts. In der Publish/Subscribe-Domäne kapselt ein Destination-Objekt ein Thema und in der Punkt-zu-Punkt (Point-to-point)-Domäne kapselt ein Destination-Objekt eine Warteschlange. Eine Anwendung kann mit den Parametern während der Laufzeit ein Destination-Objekt erstellen. Alternativ dazu kann die Anwendung ein Destination-Objekt auch basierend auf einer Objektdefinition erstellen, die in einem Repository für verwaltete Objekte gespeichert ist.

Session
Ein Session-Objekt ist ein Einzelthreadkontext zum Senden und Empfangen von Nachrichten. Eine Anwendung verwendet ein Session-Objekt zum Erstellen von Message-, MessageProducer- und MessageConsumer-Objekten.

Message
Ein Message-Objekt kapselt das Message-Objekt, das eine Anwendung mithilfe eines MessageProducer-Objekts sendet oder eines MessageConsumer-Objekts empfängt.

MessageProducer
Eine Anwendung verwendet ein MessageProducer-Objekt zum Senden von Nachrichten an ein Ziel.

MessageConsumer
Eine Anwendung verwendet ein MessageConsumer-Objekt zum Empfangen von Nachrichten, die an ein Ziel gesendet wurden.

In Abbildung 1 auf Seite 8 sind diese Objekte und ihre Beziehungen dargestellt.

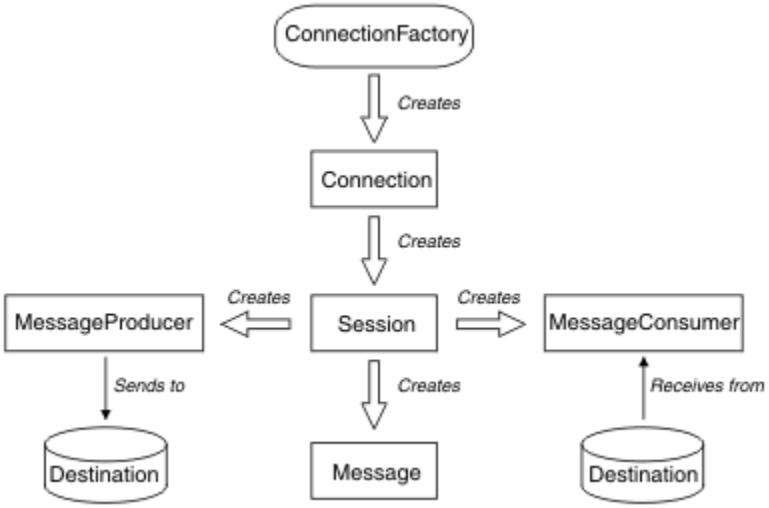

Abbildung 1. XMS-Objekte und ihre Beziehungen

In .NET werden die XMS-Klassen als eine Gruppe von .NET-Schnittstellen definiert. Wenn Sie XMS .NET-Anwendungen codieren, benötigen Sie nur die deklarierten Schnittstellen.

Das XMS-Objektmodell basiert auf den domänenunabhängigen Schnittstellen, die in *Java Message Service Specification Version 1.1* beschrieben werden. Domänenspezifische Klassen wie Topic, TopicPublisher und TopicSubscriber werden nicht bereitgestellt.

Attribute und Eigenschaften von Objekten

Ein XMS-Objekt kann Attribute und Eigenschaften besitzen, bei denen es sich um Merkmale des Objekts handelt, die auf unterschiedliche Weise implementiert werden.

Attribute
Ein Objektmerkmal, das immer vorhanden ist und Speicherplatz belegt, selbst wenn dem Attribut kein Wert zugeordnet ist. In dieser Hinsicht ähnelt ein Attribut einem Feld in einer Datenstruktur mit fester Länge. Das Unterscheidungskriterium von Attributen ist, dass es für jedes Attribut spezifische Methoden gibt, mit denen sein Wert festgelegt und abgerufen wird.

Eigenschaften
Eine Objekteigenschaft ist erst dann vorhanden und belegt Speicherplatz, nachdem ein Wert für sie angegeben wurde. Nachdem ein Wert für eine Eigenschaft festgelegt wurde, kann sie nicht mehr gelöscht oder der ihr zugeordnete Speicher wiederhergestellt werden. Sie können jedoch den Wert einer Eigenschaft ändern. XMS stellte eine Gruppe generischer Methoden zum Festlegen und Abrufen von Eigenschaftswerten bereit.

Zugehörige Konzepte

Primitive XMS-Datentypen
XMS stellt Datentypen bereit, die den acht primitiven Java-Datentypen entsprechen: 'byte', 'short', 'int', 'long', 'float', 'double', 'char' und 'boolean'. Dies ermöglicht den Austausch von Nachrichten zwischen XMS und JMS, ohne dass Daten verloren gehen oder beschädigt werden.

Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp
Wenn eine Anwendung den Wert einer Eigenschaft abruft, kann der Wert von XMS in einen anderen Datentyp konvertiert werden. Es gibt viele Regeln, mit denen gesteuert wird, welche Konvertierungen unterstützt werden und wie XMS die Konvertierungen ausführt.

Zugehörige Verweise

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Verwaltete Objekte

Mithilfe von verwalteten Objekten können Sie die Verbindungseinstellungen von Clientanwendungen verwalten, die über ein zentrales Repository verwaltet werden sollen. Dabei ruft eine Anwendung Objektdefinitionen aus dem zentralen Repsitory ab und erstellt basierend darauf `ConnectionFactory`- und `Destination`-Objekte. Verwaltete Objekte ermöglichen eine Entkopplung der Anwendungen von den Ressourcen, die sie während der Laufzeit verwenden.

XMS-Anwendungen können beispielsweise mit verwalteten Objekten geschrieben und getestet werden, die auf eine Gruppe von Verbindungen und Zielen in einer Testumgebung verweisen. Wenn diese Anwendungen dann in der Produktionsumgebung bereitgestellt werden, kann die Konfiguration der verwalteten Objekte so geändert werden, dass sie nun auf Verbindungen und Ziele in der Produktionsumgebung verweisen.

XMS unterstützt zwei Arten von verwalteten Objekten:

- Ein `ConnectionFactory`-Objekt, mit dem die Anwendungen die einleitende Verbindung zum Server herstellen.
- Ein `Destination`-Objekt, mit dem die Anwendungen das Ziel für gesendete Nachrichten und die Quelle von empfangenen Nachrichten angeben. Ein Ziel kann entweder ein Thema oder eine Warteschlange auf dem Server sein, zu dem die Anwendung eine Verbindung herstellt.

Das Verwaltungstool **`JMSAdmin`** gehört zum Lieferumfang von IBM WebSphere MQ. Sie wird verwendet, um verwaltete Objekte für in einem zentralen Repository verwalteter Objekte zu erstellen und zu verwalten.

Die verwalteten Objekte im Repository können von IBM WebSphere MQ -Klassen für JMS- und XMS-Anwendungen verwendet werden. XMS -Anwendungen können die Objekte `ConnectionFactory` und

`Destination` verwenden, um eine Verbindung zu einem IBM WebSphere MQ Queue Managerherzustellen. Ein Administrator kann die im Repository befindlichen Objektdefinitionen ohne Auswirkung auf den Anwendungscode ändern.

Das folgende Diagramm veranschaulicht, wie verwaltete Objekte normalerweise von einer XMS-Anwendung verwendet werden.

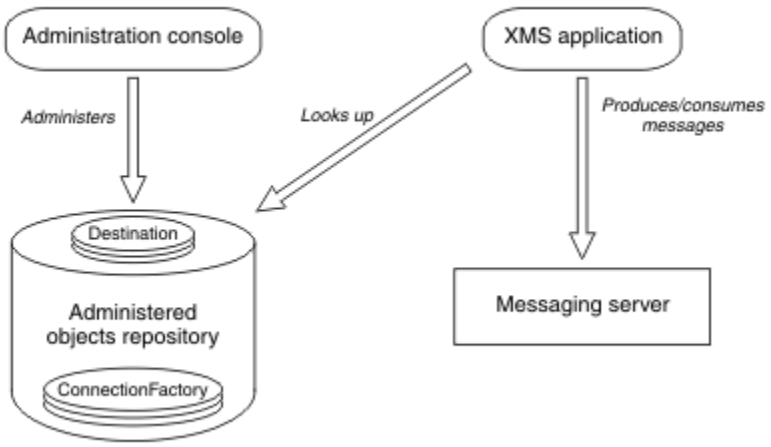

Abbildung 2. Typische Verwendung von verwalteten Objekten durch eine XMS-Anwendung

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Das XMS-Nachrichtenmodell

Das XMS-Nachrichtenmodell ist identisch mit dem Nachrichtenmodell von IBM WebSphere MQ -Klassen für JMS.

XMS implementiert insbesondere auch dieselben Nachrichtenheaderfelder und Nachrichteneigenschaften wie IBM WebSphere MQ -Klassen für JMS:

- JMS-Header-Felder. Die Namen dieser Felder beginnen mit dem Präfix 'JMS'.
- JMS-definierte Eigenschaften. Diese Felder haben Eigenschaften, deren Namen mit dem Präfix 'JMSX' beginnen.
- Von IBM definierte Eigenschaften Diese Felder haben Eigenschaften, deren Namen mit dem Präfix 'JMS_IBM' beginnen.

Daher können XMS-Anwendungen Nachrichten mit IBM WebSphere MQ -Klassen für JMS-Anwendungen austauschen. In jeder Nachricht werden einige Headerfelder und Eigenschaften durch die Anwendung und andere durch XMS oder IBM WebSphere MQ -Klassen für JMS festgelegt. Einige der von XMS oder IBM WebSphere MQ -Klassen für JMS definierten Felder werden beim Senden der Nachricht und andere bei deren Empfang festgelegt. Headerfelder und Eigenschaften werden gegebenenfalls mithilfe

einer Nachricht über einen Messaging-Server weitergegeben. Sie werden dann für Anwendungen, die die Nachricht erhalten, verfügbar gemacht.

Anwendungen an der Verwendung einer neueren XMS-Version hindern

Wenn eine neuere Version von XMS installiert wird, wechseln die Anwendungen, die die vorherige Version verwenden, standardmäßig automatisch zur neueren Version, ohne dass sie erneut kompiliert werden müssen.

Informationen zu diesem Vorgang

Die Funktion für die Koexistenz mehrerer Versionen stellt sicher, dass die vorherige XMS-Version bei der Installation einer neueren XMS-Version nicht überschrieben wird. Stattdessen können mehrere Instanzen ähnlicher XMS .NET-Assemblys (jedoch mit unterschiedlichen Versionsnummern) gleichzeitig im Global Assembly Cache (GAC) koexistieren. Intern verwendet der GAC eine Richtliniendatei, um die Anwendungsaufrufe an die neuste Version von XMS weiterzuleiten. In diesem Fall können die Anwendung ohne Neukompilierung ausgeführt werden und die neuen Funktionen verwenden, die in der neueren XMS .NET-Version verfügbar sind.

Wenn eine Anwendung die ältere XMS -Version verwenden muss, setzen Sie das Attribut `publisherpolicy` in der Anwendungskonfigurationsdatei auf no .

Anmerkung: Der Name einer Anwendungskonfigurationsdatei besteht aus dem Namen des ausführbaren Programms, auf das sich die Datei bezieht, und dem Suffix '.config'. Demnach hätte beispielsweise die Anwendungskonfigurationsdatei für das ausführbare Programm 'text.exe' den Namen 'text.exe.config'.

Alle Anwendungen eines Systems verwenden jedoch stets dieselbe Version von XMS .NET.

Messaging-Serverumgebung einrichten

In diesem AbschnittKapitel wird beschrieben, wie die Messaging-Server-Umgebung so eingerichtet wird, dass XMS-Anwendungen eine Verbindung zu einem Server herstellen können.

Für Anwendungen, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellen, ist der IBM WebSphere MQ-Client (oder Warteschlangenmanager für den Bindungsmodus) erforderlich.

Aktuell gibt es keine Voraussetzungen für Anwendungen, die eine Echtzeitverbindung zu einem Broker verwenden.

Sie müssen die Messaging-Serverumgebung einrichten, bevor Sie jegliche XMS-Anwendungen verwenden, einschließlich der mit XMS bereitgestellten Beispielanwendungen.

Dieser AbschnittKapitel enthält folgende ThemenAbschnitte:

- „Warteschlangenmanager und Broker für eine Anwendung konfigurieren, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellt” auf Seite 12
- „Broker für Anwendungen mit Echtzeitverbindung konfigurieren” auf Seite 14
- „Service Integration Bus für eine Anwendung konfigurieren, die eine Verbindung zu einem WebSphere Service Integration Bus herstellt” auf Seite 15

Zugehörige Tasks

Message Service Client for .NET mit dem Installationsassistenten installieren
Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

Warteschlangenmanager und Broker für eine Anwendung konfigurieren, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellt

In diesem Abschnitt wird vorausgesetzt, dass Sie IBM WebSphere MQ Version 7.0verwenden. Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem IBM WebSphere MQ-Queue Manager herstellt, müssen Sie den Queue Manager konfigurieren. Für eine Publish/Subscribe-Anwendung ist eine zusätzliche Konfiguration erforderlich, wenn Sie die Publish/Subscribe-Schnittstelle in der Warteschlange verwenden.

Vorbereitende Schritte

XMS funktioniert mit WebSphere Message Broker Version 6.1 oder früher.

Bevor Sie mit dieser Task beginnen, müssen Sie folgende Schritte ausführen:

- Stellen Sie sicher, dass Ihre Anwendung Zugriff auf einen aktiven Queue Manager hat.
- Wenn Ihre Anwendung eine Publish/Subscribe-Anwendung ist und die Publish/Subscribe-Schnittstelle in der Warteschlange verwendet, stellen Sie sicher, dass auf dem Warteschlangenmanager das Attribut "PSMODE" auf "ENABLED" gesetzt ist.
- Stellen Sie sicher, dass Ihre Anwendung eine Verbindungsfactory verwendet, deren Eigenschaften korrekt für eine Verbindung zu einem Warteschlangenmanager festgelegt sind. Wenn Ihre Anwendung eine Publish/Subscribe-Anwendung ist, stellen Sie sicher, dass die entsprechenden Eigenschaften der Verbindungsfactory für die Verwendung des Brokers festgelegt sind. Weitere Informationen zu den Eigenschaften einer Verbindungsfactory finden Sie im Abschnitt „Eigenschaften von ConnectionFactory” auf Seite 190.

Informationen zu diesem Vorgang

Sie konfigurieren den Queue Manager und Broker für die Ausführung von XMS-Anwendungen auf dieselbe Weise wie den Warteschlangenmanager und die Publish/Subscribe-Schnittstelle in der Warteschlange für die Ausführung von WebSphere MQ-JMS-Anwendungen. In der folgenden Vorgehensweise sind die erforderlichen Schritte zusammengefasst.

Vorgehensweise

1. Erstellen Sie auf dem Queue Manager die Warteschlangen, die Ihre Anwendung benötigt.

   Informationen zur Erstellung von Warteschlangen finden Sie im Thema *Warteschlangen definieren* in der *WebSphere MQ -Produktdokumentation*.

   Wenn Ihre Anwendung eine Publish/Subscribe-Anwendung ist und die Publish/Subscribe-Schnittstelle in der Warteschlange verwendet, die Zugriff auf IBM WebSphere MQ -Klassen für JMS-Systemwarteschlangen benötigt, dann warten Sie bis Schritt 4a, bevor Sie die Warteschlangen erstellen.
2. Erteilen Sie der Benutzer-ID, die Ihrer Anwendung zugeordnet ist, die Berechtigung zum Herstellen einer Verbindung zum Queue Manager und die geeignete Berechtigung für den Zugriff auf die Warteschlangen.

   Informationen zur Autorisierung finden Sie im Abschnitt *Sicherheit* der Produktdokumentation zu *IBM WebSphere MQ*. Wenn Ihre Anwendung eine Verbindung zu Queue Manager im Clientmodus herstellt, lesen Sie auch den Artikel *Clients* in der *IBM WebSphere MQ Produktdokumentation*.
3. Wenn Ihre Anwendung eine Verbindung zum Warteschlangenmanager im Clientmodus herstellt, stellen Sie sicher, dass auf dem Queue Manager ein Serververbindungskanal definiert ist und ein Listener gestartet wird.

   Weitere Informationen hierzu finden Sie im Abschnitt *Clients* in der *Produktdokumentation zuIBM WebSphere MQ*.

Sie müssen diesen Schritt nicht für jede Anwendung ausführen, die eine Verbindung zum Queue Manager herstellt. Für die Unterstützung aller Anwendungen mit Verbindungen im Clientmodus ist nur eine Serververbindungskanaldefinition und ein Listener erforderlich.

4. Wenn Ihre Anwendung eine Publish/Subscribe-Anwendung ist und die Publish/Subscribe-Schnittstelle in der Warteschlange verwendet, führen Sie die folgenden Schritte aus.
   a) Erstellen Sie auf dem Queue Manager die IBM WebSphere MQ -Klassen für JMS-Systemwarteschlangen, indem Sie das Script mit MQSC-Befehlen ausführen, die mit IBM WebSphere MQ bereitgestellt werden. Stellen Sie sicher, dass die Benutzer-ID, die dem WebSphere Message Broker zugeordnet ist, über Zugriffsberechtigung für die Warteschlangen verfügt.

      Informationen dazu, wo Sie das Script finden und wie es ausgeführt wird, enthält das Thema *Using Java™* in der *WebSphere MQ -Produktdokumentation*.

      Führen Sie diesen Schritt nur einmal für den Queue Manager aus. Dieselbe Gruppe von IBM WebSphere MQ -Klassen für JMS-Systemwarteschlangen kann alle XMS- und IBM WebSphere MQ -Klassen für JMS-Anwendungen unterstützen, die eine Verbindung zum Queue Manager herstellen.

   b) Erteilen Sie der Benutzer-ID, die Ihrer Anwendung zugeordnet ist, die Berechtigung für den Zugriff auf die Systemwarteschlangen für die IBM WebSphere MQ -Klassen für JMS.

      Informationen zu den Berechtigungen, die die Benutzer-ID benötigt, finden Sie im Thema *Java verwenden* in der *IBM WebSphere MQ -Produktdokumentation*.

   c) Wenn Sie einen Broker für WebSphere Event Broker oder WebSphere Message Broker verwenden, erstellen und implementieren Sie einen Nachrichtenfluss für die Warteschlangen, über den die Anwendungen Nachrichten senden können, die veröffentlicht werden sollen.

      Der grundlegende Nachrichtenfluss umfasst den Nachrichtenverarbeitungsknoten 'MQInput' zum Lesen der veröffentlichten Nachrichten sowie den Nachrichtenverarbeitungsknoten 'Publication' zum Veröffentlichen der Nachrichten.

      Informationen zum Erstellen und Implementieren eines Nachrichtenflusses finden Sie in der Produktdokumentation zu WebSphere Event Broker oder WebSphere Message Broker .

      Wenn bereits zuvor ein entsprechender Nachrichtenfluss für den Broker vorhanden ist, können Sie diesen Schritt überspringen.

Ergebnisse

Sie können Ihre Anwendung nun starten.

Zugehörige Tasks

Broker für Anwendungen mit Echtzeitverbindung konfigurieren
Bevor Sie eine Anwendung ausführen können, die eine Echtzeitverbindung zu einem Broker verwendet, müssen Sie den entsprechenden Broker konfigurieren.

Service Integration Bus für eine Anwendung konfigurieren, die eine Verbindung zu einem WebSphere Service Integration Bus herstellt
Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem WebSphere Service Integration Busherstellt, müssen Sie Service Integration Bus auf dieselbe Weise konfigurieren wie Service Integration Bus für die Ausführung von JMS-Anwendungen, die den Standard-Messaging-Provider verwenden.

Message Service Client for .NET mit dem Installationsassistenten installieren
Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

Zugehörige Verweise

Voraussetzungen für XMS -Anwendungen, die eine Verbindung zu IBM WebSphere MQ herstellen
Es gelten einige Voraussetzungen, wenn eine XMS-Anwendung eine Verbindung zu IBM WebSphere MQ herstellt.

Broker für Anwendungen mit Echtzeitverbindung konfigurieren

Bevor Sie eine Anwendung ausführen können, die eine Echtzeitverbindung zu einem Broker verwendet, müssen Sie den entsprechenden Broker konfigurieren.

Vorbereitende Schritte

XMS funktioniert mit WebSphere Message Broker Version 6.1 oder früher.

Bevor Sie mit dieser Task beginnen, müssen Sie folgende Schritte ausführen:

- Stellen Sie sicher, dass Ihre Anwendung Zugriff auf einen aktiven Broker hat.
- Stellen Sie sicher, dass Ihre Anwendung eine Verbindungsfactory verwendet, deren Eigenschaften korrekt für eine Echtzeitverbindung zu einem Broker festgelegt sind. Weitere Informationen zu den Eigenschaften einer Verbindungsfactory finden Sie im Abschnitt „Eigenschaften von ConnectionFactory" auf Seite 190.

Informationen zu diesem Vorgang

Bei der Konfiguration eines Brokers zum Ausführen von XMS-Anwendungen ist die Vorgehensweise dieselbe wie bei der Konfiguration eines Brokers für das Ausführen von Anwendungen für die IBM WebSphere MQ -Klassen für JMS. Die folgenden Schritte fassen zusammen, was Sie tun müssen, aber weitere Details finden Sie in der Produktdokumentation zu WebSphere Event Brokeroder WebSphere Message Broker :

Vorgehensweise

1. Erstellen und implementieren Sie einen Nachrichtenfluss, um die Nachrichten an dem TCP/IP-Port, der von einem Broker überwacht wird, zu lesen und zu veröffentlichen.

   Führen Sie dazu eine der folgenden Vorgehensweisen aus:

   - Erstellen Sie einen Nachrichtenfluss, der den Nachrichtenverarbeitungsknoten **Real-timeOptimizedFlow** enthält.
   - Erstellen Sie einen Nachrichtenfluss, der die beiden Nachrichtenverarbeitungsknoten **Real-time-Input** und 'Publication' enthält.

   Konfigurieren Sie den Knoten **Real-timeOptimizedFlow** bzw. **Real-timeInput** für die Überwachung des Ports, der für die Echtzeitverbindung verwendet wird. In XMS wird standardmäßig Portnummer 1506 für Echtzeitverbindungen verwendet.

   Wenn bereits zuvor ein entsprechender Nachrichtenfluss für den Broker vorhanden ist, können Sie diesen Schritt überspringen.
2. Wenn es erforderlich ist, dass Ihrer Anwendung Nachrichten mithilfe der WebSphere MQ Multicast Transport übermittelt werden, konfigurieren Sie die Multicastunterstützung für den Broker. Konfigurieren Sie die Themen, die multicastfähig sein müssen, und geben Sie dabei für die Themen, die eine zuverlässige Multicastkommunikation erfordern, eine zuverlässige Servicequalität an.
3. Wenn Ihre Anwendung beim Herstellen der Verbindung zum Broker einen Benutzernamen und ein Kennwort angibt, die der Broker zum Authentifizieren der Anwendung nutzen soll, konfigurieren Sie für den Benutzernamensserver und den Broker eine einfache Kennwortauthentifizierung, wie sie in ähnlicher Weise auch bei Telnet verwendet wird.

Ergebnisse

Sie können Ihre Anwendung nun starten.

Zugehörige Tasks

Warteschlangenmanager und Broker für eine Anwendung konfigurieren, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellt

In diesem Abschnitt wird vorausgesetzt, dass Sie IBM WebSphere MQ Version 7.0verwenden. Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem IBM WebSphere MQ-Queue Manager herstellt, müssen Sie den Queue Manager konfigurieren. Für eine Publish/Subscribe-Anwendung ist eine

zusätzliche Konfiguration erforderlich, wenn Sie die Publish/Subscribe-Schnittstelle in der Warteschlange verwenden.

Service Integration Bus für eine Anwendung konfigurieren, die eine Verbindung zu einem WebSphere Service Integration Bus herstellt
Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem WebSphere Service Integration Busherstellt, müssen Sie Service Integration Bus auf dieselbe Weise konfigurieren wie Service Integration Bus für die Ausführung von JMS-Anwendungen, die den Standard-Messaging-Provider verwenden.

Message Service Client for .NET mit dem Installationsassistenten installieren
Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

Zugehörige Verweise

Voraussetzungen für XMS -Anwendungen, die eine Verbindung zu IBM WebSphere MQ herstellen
Es gelten einige Voraussetzungen, wenn eine XMS-Anwendung eine Verbindung zu IBM WebSphere MQ herstellt.

Service Integration Bus für eine Anwendung konfigurieren, die eine Verbindung zu einem WebSphere Service Integration Bus herstellt

Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem WebSphere Service Integration Busherstellt, müssen Sie Service Integration Bus auf dieselbe Weise konfigurieren wie Service Integration Bus für die Ausführung von JMS-Anwendungen, die den Standard-Messaging-Provider verwenden.

Vorbereitende Schritte

Bevor Sie mit dieser Task beginnen, müssen Sie folgende Schritte ausführen:

- Stellen Sie sicher, dass ein Messaging-Bus erstellt und Ihr Server diesem Bus als Busmember hinzugefügt wird.
- Stellen Sie sicher, dass Ihre Anwendung Zugriff auf einen Service Integration Bus hat, der mindestens eine aktive Messaging-Engine enthält.
- Wenn HTTP-Operationen erforderlich sind, muss eine HTTP-Messaging-Engine mit eingehendem Transportkanal definiert werden. Während der Serverinstallation werden standardmäßig SSL- und TCP-Kanäle definiert.
- Stellen Sie sicher, dass Ihre Anwendung eine Verbindungsfactory verwendet, deren Eigenschaften für eine Verbindung zum Service Integration Bus über einen Bootstrap-Server festgelegt sind. Folgende Informationen sind mindestens erforderlich:
  - Der Providerendpunkt, der die Position und das Protokoll angibt, das bei der (über den Bootstrap-Server ausgeführten) Verhandlung einer Verbindung zum Messaging-Server verwendet werden soll. In seiner einfachsten Form, d. h. für einen Server, der mit Standardeinstellungen installiert wurde, kann der Providerendpunkt auf den Hostnamen des Servers gesetzt werden.
  - Der Name des Bus, über den die Nachrichten gesendet werden.

  Weitere Informationen zu den Eigenschaften einer Verbindungsfactory finden Sie im Abschnitt „Eigenschaften von ConnectionFactory” auf Seite 190.

Informationen zu diesem Vorgang

Sie müssen alle Warteschlangen- oder Themenbereiche definieren, die Sie benötigen. Während der Serverinstallation wird standardmäßig der Themenbereich 'Default.Topic.Space' definiert, doch wenn Sie weitere Themenbereiche benötigen, müssen Sie diese selbst erstellen. Innerhalb eines Themenbereichs brauchen Sie vorab keine individuellen Themen zu definieren, denn der Server instanziiert einzelne Themen dynamisch nach Bedarf.

In der folgenden Vorgehensweise sind die erforderlichen Schritte zusammengefasst.

Vorgehensweise

1. Erstellen Sie die Warteschlangen, die Ihre Anwendung für das Punkt-zu-Punkt (Point-to-point)-Messaging benötigt.
2. Erstellen Sie zusätzliche Topic-Bereiche, die Ihre Anwendung für das Publish/Subscribe-Messaging benötigt.

Ergebnisse

Sie können Ihre Anwendung nun starten.

Zugehörige Tasks

Warteschlangenmanager und Broker für eine Anwendung konfigurieren, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellt
In diesem Abschnitt wird vorausgesetzt, dass Sie IBM WebSphere MQ Version 7.0verwenden. Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem IBM WebSphere MQ-Queue Manager herstellt, müssen Sie den Queue Manager konfigurieren. Für eine Publish/Subscribe-Anwendung ist eine zusätzliche Konfiguration erforderlich, wenn Sie die Publish/Subscribe-Schnittstelle in der Warteschlange verwenden.

Broker für Anwendungen mit Echtzeitverbindung konfigurieren
Bevor Sie eine Anwendung ausführen können, die eine Echtzeitverbindung zu einem Broker verwendet, müssen Sie den entsprechenden Broker konfigurieren.

Message Service Client for .NET mit dem Installationsassistenten installieren
Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

Zugehörige Verweise

Voraussetzungen für XMS -Anwendungen, die eine Verbindung zu IBM WebSphere MQ herstellen
Es gelten einige Voraussetzungen, wenn eine XMS-Anwendung eine Verbindung zu IBM WebSphere MQ herstellt.

Message Service Client for .NET mit dem Installationsassistenten installieren

Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

Informationen zu diesem Vorgang

Gehen Sie wie folgt vor, um Message Service Client for .NET unter Windows zu installieren:

Vorgehensweise

1. Wenn Sie die Installation aus einem SupportPac durchführen, führen Sie die folgenden Schritte aus; fahren Sie andernfalls direkt mit Schritt „2” auf Seite 16 fort.
   a) Melden Sie sich unter Windows als Administrator an.
   b) Führen Sie das Installationsprogramm dotNETClientsetup.exe aus.
2. Warten Sie, bis der Installationsassistent geöffnet wird und folgende Nachricht anzeigt:

   ```
   Welcome to IBM Message Service Client for .NET installation wizard
   ```

   Klicken Sie auf **Weiter**.

   Der Assistent fordert Sie möglicherweise auf, die Lizenzvereinbarung zu lesen.

3. Wenn Sie aufgefordert werden die Lizenzvereinbarung zu lesen und Sie die Bedingungen der Lizenzvereinbarung akzeptieren, klicken Sie auf **I accept the terms in the license agreement** (Ich akzeptiere die Bedingungen in der Lizenzvereinbarung) und dann auf **Next** (Weiter).

   Der Installationsassistent fordert Sie auf, den Installationstyp auszuwählen, der Ihren Anforderungen am besten entspricht.
4. Wählen Sie den gewünschten Installationstyp aus:
   - Wenn alle Programmfeatures im Standardinstallationsverzeichnis installiert werden sollen, klicken Sie auf **Complete** (Vollständig).
   - Wenn nur ausgewählte Features installiert werden sollen und Sie ein Installationsverzeichnis angeben möchten, klicken Sie auf **Custom** (Angepasst).
5. Klicken Sie auf **Weiter**.

   Wenn Sie sich für die vollständige Installation entscheiden, zeigt der Installationsassistent eine Nachricht an, dass er zur Installation bereit ist, so wie in Schritt „8" auf Seite 17 beschrieben. Wenn Sie sich für die angepasste Installation entscheiden, fordert der Installationsassistent Sie auf, die Features auszuwählen, die installiert werden sollen, und Sie müssen Schritt „6" auf Seite 17 und Schritt „7" auf Seite 17 ausführen, bevor Sie zu Schritt „8" auf Seite 17 gehen.
6. Klicken Sie nur bei einer angepassten Installation auf ein Symbol in der Liste der Features, um anzugeben, wie die Message Service Client for .NET-Features installiert werden sollen. Wenn Message Service Client for .NET nicht im vorgeschlagenen Verzeichnis installiert werden soll, wählen Sie ein anderes Verzeichnis aus.

   Wenn das für die Installation von Message Service Client for .NET ausgewählte Verzeichnis nicht vorhanden ist, wird es automatisch vom Installationsassistenten erstellt.

   Wenn Sie XMS-Anwendungen entwickeln möchten, stellen Sie sicher, dass das Feature **Development Tools and Samples** (Entwicklungstools und -beispiele) ausgewählt ist. Dieses Feature stellt die Beispielanwendungen sowie die Bibliotheken und sonstigen Dateien bereit, die zur Kompilierung von .NET-Anwendungen erforderlich sind. Wenn Sie dieses Feature nicht auswählen, werden nur die Dateien installiert, die zur Ausführung von XMS-Anwendungen erforderlich sind.
7. Wenn Sie die angepasste Installation durchführen, klicken Sie auf **Next** (Weiter), nachdem Sie die gewünschten Optionen ausgewählt haben, so wie in Schritt „6" auf Seite 17 beschrieben.

   Der Installationsassistent zeigt eine Nachricht an, dass er zur Installation bereit ist.
8. Klicken Sie auf **Install** (Installieren), um die Installation zu starten.

   Der Installationsassistent zeigt durch einen Balken den Fortschritt der Installation an. Warten Sie, bis die Installation beendet ist. Nach erfolgreicher Installation wird im Fenster folgende Nachricht angezeigt:

   ```
   The installation wizard has successfully installed IBM Message Service Client
   for .NET. Click Finish to exit the wizard.
   ```
9. Klicken Sie auf **Finish** (Fertig stellen), um den Installationsassistenten zu schließen.

Ergebnisse

Sie haben Message Service Client for .NET erfolgreich installiert, das sofort einsatzfähig ist.

Nächste Schritte

Bevor Sie XMS-Anwendungen, einschließlich der mit XMS bereitgestellten Beispielanwendungen, ausführen, müssen Sie die Messaging-Server-Umgebung einrichten; Details hierzu finden Sie im Abschnitt „Messaging-Serverumgebung einrichten" auf Seite 11.

Zugehörige Konzepte

JNDI-Suche, Web-Service

Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser

Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Messaging-Serverumgebung einrichten
In diesem AbschnittKapitel wird beschrieben, wie die Messaging-Server-Umgebung so eingerichtet wird, dass XMS-Anwendungen eine Verbindung zu einem Server herstellen können.

XMS-Beispielanwendungen verwenden
Verwenden Sie die mit XMS bereitgestellten Beispielanwendungen, um Ihre Installation und die Messaging-Server-Konfiguration zu überprüfen und Sie bei der Erstellung eigener Anwendungen zu unterstützen. Die Beispiele geben einen Überblick über die allgemeinen Funktionen der jeweiligen API.

Zugehörige Tasks

Warteschlangenmanager und Broker für eine Anwendung konfigurieren, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellt
In diesem Abschnitt wird vorausgesetzt, dass Sie IBM WebSphere MQ Version 7.0verwenden. Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem IBM WebSphere MQ-Queue Manager herstellt, müssen Sie den Queue Manager konfigurieren. Für eine Publish/Subscribe-Anwendung ist eine zusätzliche Konfiguration erforderlich, wenn Sie die Publish/Subscribe-Schnittstelle in der Warteschlange verwenden.

Broker für Anwendungen mit Echtzeitverbindung konfigurieren
Bevor Sie eine Anwendung ausführen können, die eine Echtzeitverbindung zu einem Broker verwendet, müssen Sie den entsprechenden Broker konfigurieren.

Service Integration Bus für eine Anwendung konfigurieren, die eine Verbindung zu einem WebSphere Service Integration Bus herstellt
Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem WebSphere Service Integration Busherstellt, müssen Sie Service Integration Bus auf dieselbe Weise konfigurieren wie Service Integration Bus für die Ausführung von JMS-Anwendungen, die den Standard-Messaging-Provider verwenden.

Zugehörige Verweise

Voraussetzungen für XMS -Anwendungen, die eine Verbindung zu IBM WebSphere MQ herstellen
Es gelten einige Voraussetzungen, wenn eine XMS-Anwendung eine Verbindung zu IBM WebSphere MQ herstellt.

Voraussetzungen für XMS -Anwendungen, die eine Verbindung zu IBM WebSphere MQ herstellen

Es gelten einige Voraussetzungen, wenn eine XMS-Anwendung eine Verbindung zu IBM WebSphere MQ herstellt.

Für Anwendungen, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellen, müssen Sie die entsprechenden IBM WebSphere MQ-Clientbibliotheken auf der Maschine installieren, auf der Sie die XMS-Anwendung ausführen. Diese Bibliotheken sind auf Maschinen mit einem lokalen Warteschlangenmanager vorinstalliert.

Verwenden Sie für XMS-Client für .NET die Clientbibliotheken, die mit IBM WebSphere MQ Version 7.0.1.0 oder höher geliefert werden. Dies sind die *IBM WebSphere MQ-Klassen für .NET*. Sie ermöglichen Clientmodusverbindungen zu Warteschlangenmanagern von IBM WebSphere MQ Version 7.0, IBM WebSphere MQ Version 6.0 und IBM WebSphere MQ Version 5.3 sowie Bindungsmodusverbindungen zu einem lokalen Warteschlangenmanager, wenn es sich auch um Version 7.0.1.0 oder höher handelt.

Microsoft .NET Framework Version 2.0 Redistributable Package muss auf dem Computer installiert werden, auf dem XMS installiert werden soll. Wenn dieses Paket nicht verfügbar ist, schlägt die XMS-Installation fehl. Anschließend müssen Sie die Installationsprozedur beenden, Microsoft .NET Framework Version 2.0 Redistributable Package auf Ihrem Computer installieren und das Installationsverfahren erneut ausführen.

Auf der Microsoft-Download-Site müssen Sie nach dotnetfx.exe für Microsoft .NET Framework Version 2.0 Redistributable Package (x86) und NetFx64.exe für Microsoft .NET Framework Version 2.0 Redistributable Package (x64) suchen, sofern zutreffend.

Zugehörige Konzepte

„Messaging-Serverumgebung einrichten” auf Seite 11
In diesem AbschnittKapitel wird beschrieben, wie die Messaging-Server-Umgebung so eingerichtet wird, dass XMS-Anwendungen eine Verbindung zu einem Server herstellen können.

Zugehörige Tasks

Warteschlangenmanager und Broker für eine Anwendung konfigurieren, die eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellt
In diesem Abschnitt wird vorausgesetzt, dass Sie IBM WebSphere MQ Version 7.0verwenden. Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem IBM WebSphere MQ-Queue Manager herstellt, müssen Sie den Queue Manager konfigurieren. Für eine Publish/Subscribe-Anwendung ist eine zusätzliche Konfiguration erforderlich, wenn Sie die Publish/Subscribe-Schnittstelle in der Warteschlange verwenden.

Broker für Anwendungen mit Echtzeitverbindung konfigurieren
Bevor Sie eine Anwendung ausführen können, die eine Echtzeitverbindung zu einem Broker verwendet, müssen Sie den entsprechenden Broker konfigurieren.

Service Integration Bus für eine Anwendung konfigurieren, die eine Verbindung zu einem WebSphere Service Integration Bus herstellt
Bevor Sie eine Anwendung ausführen können, die eine Verbindung zu einem WebSphere Service Integration Busherstellt, müssen Sie Service Integration Bus auf dieselbe Weise konfigurieren wie Service Integration Bus für die Ausführung von JMS-Anwendungen, die den Standard-Messaging-Provider verwenden.

Message Service Client for .NET mit dem Installationsassistenten installieren
Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

XMS-Beispielanwendungen verwenden

Verwenden Sie die mit XMS bereitgestellten Beispielanwendungen, um Ihre Installation und die Messaging-Server-Konfiguration zu überprüfen und Sie bei der Erstellung eigener Anwendungen zu unterstützen. Die Beispiele geben einen Überblick über die allgemeinen Funktionen der jeweiligen API.

Zugehörige Konzepte

„Beispielanwendungen” auf Seite 19
Die Beispielanwendungen geben einen Überblick über die allgemeinen Funktionen der jeweiligen API. Mithilfe der Beispielanwendungen können Sie Ihre Installation und Messaging-Server-Konfiguration überprüfen sowie Ihre eigenen Anwendungen erstellen.

Zugehörige Tasks

Message Service Client for .NET mit dem Installationsassistenten installieren
Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

„Beispielanwendungen ausführen” auf Seite 21
Sie können die .NET-Beispielanwendungen interaktiv im einfachen oder erweiterten Modus oder nicht interaktiv mithilfe von automatisch generierten oder angepassten Antwortdateien ausführen.

„.NET-Beispielanwendungen erstellen” auf Seite 22
Wenn Sie eine .NET-Beispielanwendung erstellen, wird eine ausführbare Version Ihres gewählten Beispiels erstellt.

Beispielanwendungen

Die Beispielanwendungen geben einen Überblick über die allgemeinen Funktionen der jeweiligen API. Mithilfe der Beispielanwendungen können Sie Ihre Installation und Messaging-Server-Konfiguration überprüfen sowie Ihre eigenen Anwendungen erstellen.

Wenn Sie beim Erstellen Ihrer eigenen Anwendungen Unterstützung benötigen, können Sie die Beispielanwendungen als Ausgangspunkt dafür verwenden. Für jede Anwendung wird eine Quellenversion und

ein kompilierte Version bereitgestellt. Prüfen Sie den Quellencode einer Beispielanwendung, um die wichtigsten Schritte zum Erstellen aller für Ihre Anwendung erforderlichen Objekte (ein ConnectionFactory-, Connection-, Session-, Destination- und ein Producer- und/oder ein Consumer-Objekt) zu ermitteln, und um spezielle Eigenschaften festzulegen, mit denen die Funktionsweise Ihrer Anwendung gesteuert wird. Weitere Informationen finden Sie unter „XMS-Anwendungen schreiben” auf Seite 23. Die Beispielanwendungen können sich in zukünftigen XMS-Releases ändern.

Die folgende Tabelle zeigt die drei Gruppen von Beispielanwendungen (eine für jede API), die mit XMSbereitgestellt werden.

Name der Beispielanwendung	Beschreibung
SampleConsumerCS	Eine Nachrichtenkonsumentenanwendung, die Nachrichten aus einer Warteschlange abruft oder ein Thema abonniert.
SampleProducerCS	Eine Nachrichtenproduzentenanwendung, die Nachrichten in einer Warteschlange oder in einem Thema erstellt.
SampleConfigCS	Eine Konfigurationsanwendung, mit dem Sie ein dateibasiertes Repository für verwaltete Objekte erstellen können. Die Anwendung enthält eine Verbindungsfactory und ein Ziel für Ihre speziellen Verbindungseinstellungen. Dieses Repository für verwaltete Objekte kann dann zusammen mit jeder als Beispiel bereitgestellten Konsumenten- und Produzentenanwendung verwendet werden.

Die Beispielanwendungen, die dieselben Funktionen in verschiedenen APIs unterstützen, haben eine unterschiedliche Syntax.

- Die Nachrichtenkonsumenten- und Nachrichtenproduzenten-Beispielanwendungen unterstützen beide die folgenden Funktionen:
  - Verbindungen zu IBM WebSphere MQ, WebSphere Event Broker, WebSphere Message Broker (unter Verwendung einer Echtzeitverbindung zu einem Broker) und einem WebSphere Service Integration Bus
  - Suchvorgänge in einem Repository für verwaltete Objekte mithilfe der Ausgangskontextschnittstelle
  - Verbindungen zu Warteschlangen (IBM WebSphere MQ und WebSphere Service Integration Bus) und Themen (IBM WebSphere MQ, Echtzeitverbindung zu einem Broker und WebSphere Service Integration Bus)
  - Basis-, Byte-, Zuordnungs-, Objekt-, Datenstrom- und Textnachrichten
- Die Nachrichtenkonsumenten-Beispielanwendung unterstützt den synchronen und asynchronen Empfangsmodus sowie SQL-Selektoranweisungen.
- Die Nachrichtenproduzenten-Beispielanwendung unterstützt den persistenten und nicht persistenten Übermittlungsmodus.

Betriebsmodi

Die Beispielanwendungen können in einem der folgenden beiden Modi ausgeführt werden:

Einfacher Modus
Sie können die Beispielanwendungen mit minimalen Benutzereingaben ausführen.

Erweiterter Modus
In diesem Fall können Sie die Funktionsweise der Beispielanwendungen exakter für Ihren Bedarf anpassen.

Alle Beispielanwendungen sind untereinander kompatibel und können deshalb über verschiedene Programmiersprachen hinweg ausgeführt werden.

Wo Sie die Beispiele finden

Informationen dazu, wo die Beispielanwendungen für Message Service Client for .NET installiert sind, finden Sie unter *Verzeichnisse, die unter Windows (.NET) installiert sind* in der Online-Produktdokumentation zu IBM IBM WebSphere MQ .

Zugehörige Konzepte

„Eigene Anwendung erstellen” auf Seite 48
Sie können Ihre eigenen Anwendungen auf die gleiche Weise erstellen, wie Sie die Beispielanwendungen erstellen.

Zugehörige Tasks

Beispielanwendungen ausführen
Sie können die .NET-Beispielanwendungen interaktiv im einfachen oder erweiterten Modus oder nicht interaktiv mithilfe von automatisch generierten oder angepassten Antwortdateien ausführen.

.NET-Beispielanwendungen erstellen
Wenn Sie eine .NET-Beispielanwendung erstellen, wird eine ausführbare Version Ihres gewählten Beispiels erstellt.

„Beispielanwendungen ausführen” auf Seite 21
Sie können die .NET-Beispielanwendungen interaktiv im einfachen oder erweiterten Modus oder nicht interaktiv mithilfe von automatisch generierten oder angepassten Antwortdateien ausführen.

„.NET-Beispielanwendungen erstellen” auf Seite 22
Wenn Sie eine .NET-Beispielanwendung erstellen, wird eine ausführbare Version Ihres gewählten Beispiels erstellt.

Beispielanwendungen ausführen

Sie können die .NET-Beispielanwendungen interaktiv im einfachen oder erweiterten Modus oder nicht interaktiv mithilfe von automatisch generierten oder angepassten Antwortdateien ausführen.

Vorbereitende Schritte

Bevor Sie die bereitgestellten Beispielanwendungen ausführen können, müssen Sie zunächst die Messaging-Serverumgebung einrichten, damit die Anwendungen eine Verbindung zu einem Server herstellen können. Weitere Informationen hierzu finden Sie unter „Messaging-Serverumgebung einrichten” auf Seite 11.

Vorgehensweise

Führen Sie folgende Schritte aus, um eine .NET-Beispielanwendung auszuführen:

Tipp: Wenn Sie eine Beispielanwendung ausführen, geben Sie Folgendes ein: jederzeit, um Hilfe zu erhalten, was als Nächstes zu tun ist.

1. Wählen Sie den Modus aus, in dem Sie die Beispielanwendung ausführen möchten.

   Geben Sie entweder `Advanced` (Erweitert) oder `Simple` (Einfach) ein.
2. Beantworten Sie die Fragen.

   Drücken Sie die Eingabetaste, um den Standardwert auszuwählen, der in eckigen Klammern am Ende der Frage angezeigt wird. Wenn Sie einen anderen Wert auswählen möchten, geben Sie den entsprechenden Wert ein und drücken Sie dann die Eingabetaste.

   Es folgt eine Beispielfrage:

   ```
   Enter connection type [wpm]:
   ```

   In diesem Fall lautet der Standardwert `wpm` (die entspricht einer Verbindung zu einem WebSphere Service Integration Bus).

Ergebnisse

Wenn Sie die Beispielanwendungen ausführen, werden automatisch Antwortdateien im aktuellen Arbeitsverzeichnis generiert. Antwortdateinamen besitzen das Format *connection_type-sample_type*.rsp; z. B. wpm-producer.rsp. Bei Bedarf können Sie die generierte Antwortdatei nutzen, um die Beispielanwendung erneut mit denselben Optionen auszuführen, damit Sie die zuvor gewählten Einstellungen nicht erneut eingeben müssen.

Zugehörige Konzepte

Beispielanwendungen
Die Beispielanwendungen geben einen Überblick über die allgemeinen Funktionen der jeweiligen API. Mithilfe der Beispielanwendungen können Sie Ihre Installation und Messaging-Server-Konfiguration überprüfen sowie Ihre eigenen Anwendungen erstellen.

„Beispielanwendungen” auf Seite 19
Die Beispielanwendungen geben einen Überblick über die allgemeinen Funktionen der jeweiligen API. Mithilfe der Beispielanwendungen können Sie Ihre Installation und Messaging-Server-Konfiguration überprüfen sowie Ihre eigenen Anwendungen erstellen.

Zugehörige Tasks

.NET-Beispielanwendungen erstellen
Wenn Sie eine .NET-Beispielanwendung erstellen, wird eine ausführbare Version Ihres gewählten Beispiels erstellt.

„.NET-Beispielanwendungen erstellen” auf Seite 22
Wenn Sie eine .NET-Beispielanwendung erstellen, wird eine ausführbare Version Ihres gewählten Beispiels erstellt.

.NET-Beispielanwendungen erstellen

Wenn Sie eine .NET-Beispielanwendung erstellen, wird eine ausführbare Version Ihres gewählten Beispiels erstellt.

Vorbereitende Schritte

Installieren Sie den entsprechenden Compiler. Weitere Informationen finden Sie unter *Message Service Client for .NET* in der Online-Produktdokumentation zu IBM IBM WebSphere MQ . Diese Task setzt voraus, dass Visual Studio 2005 installiert ist und Sie mit der Verwendung von Visual Studio 2005 vertraut sind.

Vorgehensweise

Führen Sie folgende Schritte aus, um eine .NET-Beispielanwendung auszuführen:

1. Klicken Sie auf die Lösungsdatei Samples.sln, die mit den .NET-Beispielen bereitgestellt wird.
2. Klicken Sie mit der rechten Maustaste im 'Solution Explorer' auf die Samples (Beispiele) und wählen Sie **Build Solution** (Lösung erstellen) aus.

Ergebnisse

Im entsprechenden Unterordner des Beispiels, der je nach der von Ihnen gewählten Konfiguration entweder bin/Debug oder bin/Release heißt, wird ein ausführbares Programm erstellt. Dieses Programm hat denselben Namen wie der Ordner und das Suffix CS. Wenn Sie beispielsweise die C#-Version der Beispielanwendung für einen Nachrichtenproduzenten erstellen, wird im Ordner SampleProducer das ausführbare Programm SampleProducerCS.exe erstellt.

Zugehörige Konzepte

Beispielanwendungen
Die Beispielanwendungen geben einen Überblick über die allgemeinen Funktionen der jeweiligen API. Mithilfe der Beispielanwendungen können Sie Ihre Installation und Messaging-Server-Konfiguration überprüfen sowie Ihre eigenen Anwendungen erstellen.

„Beispielanwendungen” auf Seite 19

Die Beispielanwendungen geben einen Überblick über die allgemeinen Funktionen der jeweiligen API. Mithilfe der Beispielanwendungen können Sie Ihre Installation und Messaging-Server-Konfiguration überprüfen sowie Ihre eigenen Anwendungen erstellen.

„Eigene Anwendung erstellen” auf Seite 48
Sie können Ihre eigenen Anwendungen auf die gleiche Weise erstellen, wie Sie die Beispielanwendungen erstellen.

Zugehörige Tasks

Beispielanwendungen ausführen
Sie können die .NET-Beispielanwendungen interaktiv im einfachen oder erweiterten Modus oder nicht interaktiv mithilfe von automatisch generierten oder angepassten Antwortdateien ausführen.

„Beispielanwendungen ausführen” auf Seite 21
Sie können die .NET-Beispielanwendungen interaktiv im einfachen oder erweiterten Modus oder nicht interaktiv mithilfe von automatisch generierten oder angepassten Antwortdateien ausführen.

XMS-Anwendungen entwickeln

Dieser AbschnittKapitel enthält Informationen, die beim Schreiben von XMS-Anwendungen nützlich sein können.

Die Informationen in diesem AbschnittKapitel gelten für .NET-Anwendungen.

Informationen zum Schreiben von XMS-Anwendungen finden Sie in folgenden Abschnitten:

XMS-Anwendungen schreiben

Dieser AbschnittKapitel enthält Informationen, die Ihnen beim Schreiben von XMS-Anwendungen helfen.

Dieser AbschnittKapitel enthält allgemeine Konzepte zum Schreiben von XMS-Anwendungen. Im Abschnitt „XMS .NET-Anwendungen schreiben” auf Seite 49 finden Sie spezifische Informationen zum Erstellen von .NET-Anwendungen.

Dieser AbschnittKapitel enthält folgende ThemenAbschnitte:

Zugehörige Verweise

.NET-Schnittstellen
In diesem Thema werden die Schnittstellen der .NET-Klasse und die zugehörigen Eigenschaften und Methoden dokumentiert.

Threading-Modell

Für die Verwendung von XMS-Objekten in einer Multithread-Anwendung gibt es allgemeine Regeln.

- Nur folgende Objekte können gleichzeitig in unterschiedlichen Threads verwendet werden:
  - ConnectionFactory
  - Verbindung
  - ConnectionMetaData
  - Ziel
- Ein Session-Objekt kann zu jedem Zeitpunkt nur in einem einzigen Thread verwendet werden.

Ausnahmen zu diesen Regeln werden durch Einträge mit der Bezeichnung "Threadkontext" in den Schnittstellendefinitionen der Methoden in den API-Referenzkapiteln„Message Service Clients for .NET - Referenz” auf Seite 96 angezeigt.

Fehlerbedingungen, die zur Laufzeit behandelbar sind

Rückgabecodes von API-Aufrufen sind Fehlerbedingungen, die zur Laufzeit behandelt werden können. Die Art und Weise, wie Sie mit diesem Fehlertyp umgehen, hängt davon ab, ob Sie die C- oder C++-API verwenden.

Fehlererkennung zur Laufzeit

Wenn eine Anwendung eine C-API-Funktion aufruft und der Aufruf fehlschlägt, wird eine Antwort mit einem anderen Rückgabecode als XMS_OK zusammen mit einem XMS-Fehlerblock zurückgegeben, der weitere Informationen zur Ursache des Fehlers enthält.

Die C++-API löst eine Ausnahme aus, wenn eine Methode verwendet wird.

Eine Anwendung wird von einem Listener für Ausnahmebedingungen asynchron über ein Problem mit einer Verbindung benachrichtigt. Der Listener für Ausnahmebedingungen wird über die API XMS C oder C++ bereitgestellt und initialisiert.

Fehlerbehandlung zur Laufzeit

Einige Fehlerbedingungen weisen darauf hin, dass bestimmte Ressourcen nicht verfügbar sind. Welche Aktion eine Anwendung in einem solchen Fall ausführen kann, hängt von der XMS-Funktion ab, die von der Anwendung aufgerufen wird. Wenn beispielsweise die Herstellung einer Verbindung zum Server fehlschlägt, kann die Anwendung es in regelmäßigen Abständen erneut versuchen, bis eine Verbindung hergestellt wird. Ein XMS-Fehlerblock oder eine Ausnahme enthält möglicherweise nicht genug Informationen, um zu bestimmen, welche Aktion ausgeführt werden soll. In diesen Situationen gibt es häufig einen Link zu einem Fehlerblock oder einer Ausnahme mit detaillierteren Diagnoseinformationen.

Führen Sie in der C-API immer Prüfungen auf eine Antwort mit einem anderen Rückgabecode als XMS_OK durch und übergeben Sie immer einen Fehlerblock im API-Aufruf. Welche Aktion ausgeführt wird, hängt normalerweise davon ab, welche API-Funktion die Anwendung verwendet.

Schließen Sie in der C++-API Aufrufe an Methoden immer in einen Try-Block ein und geben Sie die Exception-Klasse in der Catch-Anweisung an, um alle XMS-Ausnahmetypen abzufangen.

Der Listener für Ausnahmebedingungen ist ein asynchroner Fehlerbedingungspfad, der jederzeit gestartet werden kann. Wenn die Listenerfunktion in einem eigenen Thread gestartet wird, ist dies normalerweise ein Hinweis auf einen schwerwiegenderen Fehler als eine normale XMS-API-Fehlerbedingung. Es kann jede geeignete Aktion ausgeführt werden, aber Sie müssen dabei die Regeln für das XMS-Threading-Modell, die im Abschnitt „Threading-Modell” auf Seite 24 beschrieben werden, genau einhalten.

ConnectionFactory- und Connection-Objekte

Ein Verbindungsfactoryobjekt (ConnectionFactory) stellt eine Vorlage bereit, die von einer Anwendung zum Erstellen eines Verbindungsobjekts (Connection) verwendet wird. Das Connection-Objekt verwendet die Anwendung wiederum zum Erstellen eines Sitzungsobjekts (Session).

Für .NET verwendet eine Eine XMS-Anwendung zunächst ein XMSFactoryFactory-Objekt, um einen Verweis auf ein ConnectionFactory-Objekt abzurufen, das für den erforderlichen Protokolltyp geeignet ist. Mit diesem ConnectionFactory-Objekt können dann nur Verbindungen für diesen Protokolltyp erstellt werden.

Eine XMS-Anwendung kann mehrere Verbindungen erstellen und eine Multithread-Anwendung kann ein einzelnes Connection-Objekt gleichzeitig in mehreren Threads verwenden. Ein Connection-Objekt enthält die Kommunikationsverbindung zwischen einer Anwendung und einem Messaging-Server.

Eine Verbindung dient mehreren Zwecken:

- Wenn eine Anwendung eine Verbindung erstellt, kann die Anwendung authentifiziert werden.
- Eine Anwendung kann einer Verbindung eine eindeutige Client-ID zuordnen. Die Client-ID wird verwendet, um permanente Subskriptionen in der Publish/Subscribe-Domäne zu unterstützen. Die Client-ID kann auf zwei Arten festgelegt werden:

  Die bevorzugte Art, um einer Verbindung eine Client-ID zuzuweisen, besteht darin, mithilfe von Eigenschaften ein clientspezifisches ConnectionFactory-Objekt zu konfigurieren und es der Verbindung, die mit diesem Objekt erstellt wird, transparent zuzuweisen.

  Als Alternative dazu kann beim Zuweisen einer Client-ID auch ein im Connection-Objekt festgelegter providerspezifischer Wert verwendet werden. Die administrativ konfigurierte ID wird durch diesen Wert jedoch nicht überschrieben. Der Wert wird für den Fall bereitgestellt, dass keine adminstrativ erstellte ID vorhanden ist. Bei dem Versuch, eine administrativ erstellte ID durch einen providerspezifischen Wert zu überschreiben, wird eine Ausnahmebedingung ausgelöst. Wenn eine Anwendung explizit eine ID festlegt, muss sie dies unmittelbar nach dem Erstellen der Verbindung tun und bevor eine andere Aktion über die Verbindung ausgeführt wird, denn andernfalls wird eine Ausnahmebedingung ausgelöst.

Eine XMS-Anwendung erstellt normalerweise eine Verbindung, eine oder mehrere Sitzungen sowie eine Reihe von Nachrichtenproduzenten und Nachrichtenkonsumenten.

Das Erstellen einer Kommunikationsverbindung ist ein relativ kostenintensiver Vorgang, denn er benötigt eine große Menge an Systemressourcen und kann darüber hinaus eventuell die Authentifizierung der Anwendung erfordern.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Zugehörige Verweise

IConnectionFactory (für die .NET-Schnittstelle)
Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Eigenschaften von ConnectionFactory
Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Gestarteter und gestoppter Verbindungsmodus

Eine Verbindung kann sowohl im gestarteten als auch im gestoppten Modus betrieben werden.

Wenn eine Anwendung eine Verbindung erstellt, befindet sich die Verbindung im gestoppten Modus. Im gestoppten Verbindungsmodus kann die Anwendung Sitzungen initialisieren sowie Nachrichten synchron oder asynchron senden, jedoch nicht empfangen.

Eine Anwendung kann eine Verbindung durch Aufrufen der Methode `Start Connection` starten. Im gestarteten Verbindungsmodus kann die Anwendung Nachrichten senden und empfangen. Außerdem kann die Anwendung die Verbindung stoppen und erneut starten, indem sie die Methoden 'Stop Connection' und `Start Connection` aufruft.

Zugehörige Konzepte

Verbindung schließen
Zum Schließen einer Verbindung ruft die Anwendung die Methode 'Close Connection' (Verbindung schließen) auf.

Ausnahmebehandlung
Wenn eine Anwendung Nachrichten von einer Verbindung ausschließlich asynchron verarbeitet, erfährt sie nur dann von einem Problem mit der Verbindung, wenn sie einen Listener für Ausnahmebedingungen verwendet.

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Verbindung schließen

Zum Schließen einer Verbindung ruft die Anwendung die Methode 'Close Connection' (Verbindung schließen) auf.

Wenn eine Anwendung einer Verbindung schließt, führt XMS folgende Aktionen aus:

- Es schließt alle Sitzungen, die dieser Verbindung zugeordnet sind, und löscht bestimmte Objekte, die diesen Sitzungen zugeordnet sind. Weitere Informationen dazu, welche Objekte gelöscht werden, finden Sie im Abschnitt „Objektlöschung” auf Seite 42. Gleichzeitig macht XMS alle Transaktionen rückgängig, die in der Sitzung aktuell in Bearbeitung sind.
- Es beendet die Kommunikationsverbindung zum Messaging-Server.
- Es gibt den Speicher und andere interne Ressourcen frei, die von der Verbindung verwendet werden.

Für Nachrichten, deren Empfang während einer Sitzung nicht bestätigt wurde, gibt XMS auch vor dem Schließen der Verbindung keine Empfangsbestätigung aus. Weitere Informationen zu Empfangsbestätigungen für Nachrichten finden Sie im Abschnitt „Nachrichtenbestätigung” auf Seite 29.

Zugehörige Konzepte

Gestarteter und gestoppter Verbindungsmodus
Eine Verbindung kann sowohl im gestarteten als auch im gestoppten Modus betrieben werden.

Ausnahmebehandlung
Wenn eine Anwendung Nachrichten von einer Verbindung ausschließlich asynchron verarbeitet, erfährt sie nur dann von einem Problem mit der Verbindung, wenn sie einen Listener für Ausnahmebedingungen verwendet.

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Ausnahmebehandlung

Wenn eine Anwendung Nachrichten von einer Verbindung ausschließlich asynchron verarbeitet, erfährt sie nur dann von einem Problem mit der Verbindung, wenn sie einen Listener für Ausnahmebedingungen verwendet.

XMS .NET-Ausnahmen werden alle von System.Exception abgeleitet. Weitere Informationen finden Sie unter „Fehlerbehandlung in .NET” auf Seite 53.

Zugehörige Konzepte

Gestarteter und gestoppter Verbindungsmodus
Eine Verbindung kann sowohl im gestarteten als auch im gestoppten Modus betrieben werden.

Verbindung schließen

Zum Schließen einer Verbindung ruft die Anwendung die Methode 'Close Connection' (Verbindung schließen) auf.

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Verbindung zu einem WebSphere Service Integration Bus

Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Das HTTP-Protokoll kann in Situationen verwendet werden, in denen eine direkte TCP/IP-Verbindung nicht möglich ist. Dies kommt häufig vor, wenn die Kommunikation über eine Firewall ausgeführt wird, beispielsweise beim Nachrichtenaustausch zwischen zwei Unternehmen. Die Kommunikation durch eine Firewall über HTTP wird häufig als *HTTP-Tunnelung* bezeichnet. Das HTTP-Tunneling ist jedoch wesentlich langsamer als eine direkte TCP/IP-Verbindung, weil sich die Menge der zu übertragenden Daten durch die HTTP-Header erheblich vergrößert und weil das HTTP-Protokoll mehr Kommunikationsflüsse benötigt als TCP/IP.

Zum Erstellen einer TCP/IP-Verbindung kann eine Anwendung eine Verbindungsfactory verwenden, deren Eigenschaft XMSC_WPM_TARGET_TRANSPORT_CHAIN auf den Wert 'XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC' gesetzt ist. Dies ist der Standardwert der Eigenschaft. Wenn die Verbindung erfolgreich erstellt wurde, wird die Eigenschaft XMSC_WPM_CONNECTION_PROTOCOL auf den Wert 'XMSC_WPM_CP_TCP' gesetzt.

Um eine HTTP-Verbindung zu erstellen, muss eine Anwendung eine Verbindungsfactory verwenden, deren Eigenschaft 'XMSC_WPM_TARGET_TRANSPORT_CHAIN' auf den Namen einer eingehenden Transportkette gesetzt ist, die für die Verwendung eines HTTP-Transportkanals konfiguriert ist. Wenn die Verbindung erfolgreich erstellt wurde, wird die Eigenschaft 'XMSC_WPM_CONNECTION_PROTOCOL' auf den Wert 'XMSC_WPM_CP_HTTP' gesetzt. Informationen zum Konfigurieren von Transportketten finden Sie unter Transportketten in der WebSphere Application Server -Dokumentation.

Beim Herstellen einer Verbindung zu einem Bootstrap-Server steht einer Anwendung eine ähnlich Auswahl an Kommunikationsprotokollen zur Verfügung. Die Eigenschaft XMSC_WPM_PROVIDER_ENDPOINTS einer Verbindungsfactory enthält eine Sequenz von einer oder mehreren Endpunktadressen von Bootstrap-Servern. Die Bootstrap-Transportkettenkomponente einer jeden Endpunktadresse kann einen der beiden folgenden Werte annehmen: 'XMSC_WPM_BOOTSTRAP_TCP' für eine TCP/IP-Verbindung zu einem Bootstrap-Server oder 'XMSC_WPM_BOOTSTRAP_HTTP' für eine HTTP-Verbindung.

Zugehörige Konzepte

Gestarteter und gestoppter Verbindungsmodus
Eine Verbindung kann sowohl im gestarteten als auch im gestoppten Modus betrieben werden.

Verbindung schließen
Zum Schließen einer Verbindung ruft die Anwendung die Methode 'Close Connection' (Verbindung schließen) auf.

Ausnahmebehandlung
Wenn eine Anwendung Nachrichten von einer Verbindung ausschließlich asynchron verarbeitet, erfährt sie nur dann von einem Problem mit der Verbindung, wenn sie einen Listener für Ausnahmebedingungen verwendet.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Zugehörige Verweise

IConnectionFactory (für die .NET-Schnittstelle)
Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Eigenschaften von ConnectionFactory

Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Sitzungen

Sitzung - Ein Session-Objekt ist ein Einzelthreadkontext zum Senden und Empfangen von Nachrichten.

Eine Anwendung kann mithilfe einer Sitzung Nachrichten, Nachrichtenproduzenten, Nachrichtenkonsumenten, Warteschlangenbrowser und temporäre Ziele erstellen. Außerdem kann eine Anwendung eine Sitzung zum Ausführen lokaler Transaktionen verwenden.

Eine Anwendung kann mehrere Sitzungen erstellen, wobei jede Sitzung unabhängig von den anderen Sitzungen Nachrichten produziert und konsumiert. Wenn zwei Nachrichtenkonsumenten in verschiedenen Sitzungen (oder auch in derselben Sitzung) dasselbe Thema subskribieren, empfängt jeder eine eigene Kopie aller Nachrichten, die in diesem Thema veröffentlicht werden.

Anders als ein Connection-Objekt kann ein Session-Objekt nicht gleichzeitig in verschiedenen Threads verwendet werden. Nur die Methode 'Close Session' eines Session-Objekts kann von einem anderen Thread aus aufgerufen werden als der Thread, den das Session-Objekt zu diesem Zeitpunkt verwendet. Mit der Methode 'Close Session' wird eine Sitzung beendet und alle der Sitzung zugeordneten Systemressourcen werden freigeben.

Wenn es erforderlich ist, dass eine Anwendung Nachrichten in mehreren Threads gleichzeitig verarbeitet, dann muss die Anwendung in jedem dieser Threads eine Sitzung erstellen und diese Sitzung dann für die Sende- und Empfangsoperationen in dem jeweiligen Thread verwenden.

Durchgeführte Sitzungen

XMS-Anwendungen können lokale Transaktionen ausführen. Eine *lokale Transaktion* ist eine Transaktion, die Änderungen umfasst, welche nur die Ressourcen des Warteschlangenmanagers oder Service Integration Bus betreffen, mit dem die Anwendung verbunden ist.

Die Informationen in diesem Thema sind nur relevant, wenn eine Anwendung eine Verbindung zu einem IBM WebSphere MQ -Warteschlangenmanager oder einem WebSphere Service Integration Bus herstellt. Für Echtzeitverbindungen zu einem Broker sind die Informationen in diesem Abschnitt irrelevant.

Um lokale Transaktionen ausführen zu können, muss eine Anwendung zunächst eine transaktionsbasierte Sitzung erstellen, indem sie die Methode 'Create Session' eines Connection-Objekts aufruft und als Parameter angibt, dass die Sitzung transaktionsbasiert ist. Danach werden alle innerhalb der Sitzung gesendeten und empfangenen Nachrichten in einer Folge von Transaktionen gruppiert. Eine Transaktion endet, wenn die Anwendung die Nachrichten, die sie seit Beginn der Transaktion gesendet oder empfangen hat, festschreibt oder rückgängig macht.

Um eine Transaktion festzuschreiben, ruft eine Anwendung die Methode 'Commit' des Session-Objekts auf. Wenn eine Transaktion festgeschrieben wird, werden alle Nachrichten, die innerhalb der Transaktion gesendet wurden, für die Zustellung an andere Anwendungen verfügbar. Außerdem werden sämtliche Nachrichten, die innerhalb der Transaktion empfangen wurden, bestätigt, damit der Messaging-Server nicht erneut versucht, diese an die Anwendung zuzustellen. In der Punkt-zu-Punkt (Point-to-point)-Domäne entfernt der Messaging-Server zudem die empfangenen Nachrichten aus ihren Warteschlangen.

Um eine Transaktion rückgängig zu machen, ruft eine Anwendung die Methode 'Rollback' des Session-Objekts auf. Wenn eine Transaktion rückgängig gemacht wird, werden alle Nachrichten, die innerhalb der Transaktion gesendet wurden, vom Messaging-Server gelöscht und alle Nachrichten, die innerhalb der Transaktion empfangen wurden, sind erneut für die Zustellung verfügbar. In der Punkt-zu-Punkt (Point-to-point)-Domäne werden die empfangenen Nachrichten in ihre Warteschlangen zurückgestellt und für andere Anwendungen wieder sichtbar.

Wenn eine Anwendung eine transaktionsbasierte Sitzung erstellt oder die Methode 'Commit' bzw. 'Rollback' aufruft, wird automatisch eine neue Transaktion gestartet. Daher verfügt eine transaktionsbasierte Sitzung immer über eine aktive Transaktion.

Wenn eine Anwendung eine transaktionsbasierte Sitzung schließt, erfolgt ein impliziter Rollback. Sobald eine Anwendung eine Verbindung schließt, wird für alle transaktionsbasierten Sitzungen der Verbindung ein impliziter Rollback durchgeführt.

Eine Transaktion ist voll und ganz in einer transaktionsbasierten Sitzung enthalten. Eine Transaktion kann sich nicht über verschiedene Sitzungen erstrecken. Dies bedeutet, dass es einer Anwendung nicht möglich ist, Nachrichten in zwei oder mehr transaktionsbasierten Sitzungen zu senden und zu empfangen und alle diese Aktionen danach als einzelne Transaktion festzuschreiben oder rückgängig zu machen.

Zugehörige Konzepte

Nachrichtenbestätigung
Jede Sitzung ohne Transaktionsunterstützung verfügt über einen Bestätigungsmodus, der bestimmt, wie die von der Anwendung empfangenen Nachrichten bestätigt werden. Es sind drei Bestätigungsmodi verfügbar und die Auswahl des Bestätigungsmodus wirkt sich auf das Design der Anwendung aus.

Asynchrone Nachrichtenübermittlung
XMS führt alle asynchronen Nachrichtenübermittlungen für eine Sitzung über einen einzigen Thread aus. Dies bedeutet, dass immer nur eine Nachrichtenlistenerfunktion oder eine `onMessage()`-Methode nach der anderen ausgeführt werden kann.

Synchrone Nachrichtenübermittlung
Nachrichten werden synchron an eine Anwendung übermittelt, wenn die Anwendung für den Empfang die Receive-Methoden der MessageConsumer-Objekte verwendet.

Nachrichtenübermittlungsmodus
XMS unterstützt zwei Nachrichtenübermittlungsmodi.

Nachrichtenbestätigung

Jede Sitzung ohne Transaktionsunterstützung verfügt über einen Bestätigungsmodus, der bestimmt, wie die von der Anwendung empfangenen Nachrichten bestätigt werden. Es sind drei Bestätigungsmodi verfügbar und die Auswahl des Bestätigungsmodus wirkt sich auf das Design der Anwendung aus.

Die Informationen in dieser Thema -Datei sind nur relevant, wenn eine Anwendung eine Verbindung zu einem IBM WebSphere MQ -Warteschlangenmanager oder zu einem WebSphere Service Integration Busherstellt. Für Echtzeitverbindungen zu einem Broker sind die Informationen in diesem Abschnitt irrelevant.

XMS verwendet zur Bestätigung des Empfangs von Nachrichten denselben Mechanismus wie JMS.

Bei einer Sitzung ohne Transaktionsunterstützung wird die Art und Weise der Bestätigung von Nachrichten, die von der Anwendung empfangen werden, durch den Bestätigungsmodus der Sitzung bestimmt. Die drei verfügbaren Bestätigungsmodi werden im Folgenden beschrieben:

XMSC_AUTO_ACKNOWLEDGE
Die Sitzung bestätigt automatisch jede Nachricht, die von der Anwendung empfangen wird.

Wenn Nachrichten synchron an die Anwendung übermittelt werden, bestätigt die Sitzung den Empfang einer Nachricht jedes Mal, wenn ein Receive-Aufruf erfolgreich abgeschlossen wurde.

Wenn die Anwendung eine Nachricht erfolgreich empfängt, aber ein Fehler eine ordnungsgemäße Bestätigung verhindert, wird die Nachricht wieder für die Zustellung verfügbar. Daher muss die Anwendung in der Lage sein, eine erneut übermittelte Nachricht zu verarbeiten.

XMSC_DUPS_OK_ACKNOWLEDGE
Die Sitzung bestätigt den Empfang von Nachrichten durch die Anwendung zu bestimmten Zeiten, die sie auswählt.

Dieser Bestätigungsmodus verringert den von der Sitzung zu leistenden Arbeitsaufwand, aber ein Fehler, der eine Nachrichtenbestätigung verhindert, kann dazu führen, dass mehrere Nachrichten für eine erneute Zustellung verfügbar werden. Daher muss die Anwendung in der Lage sein, eine erneut übermittelte Nachricht zu verarbeiten.

XMSC_CLIENT_ACKNOWLEDGE
Die Anwendung bestätigt die empfangenen Nachrichten, indem sie die Acknowledge-Methode (Bestätigungsmethode) der Message-Klasse (Nachrichtenklasse) aufruft.

Die Anwendung kann den Empfang jeder Nachricht einzeln bestätigen oder einen Nachrichtenstapel empfangen und die Acknowledge-Methode nur für die zuletzt empfangene Nachricht aufrufen. Wenn die Acknowledge-Methode aufgerufen wird, werden alle Nachrichten bestätigt, die seit dem letzten Aufruf der Methode empfangen wurden.

In Verbindung mit jedem dieser Bestätigungsmodi kann eine Anwendung die Zustellung von Nachrichten in einer Sitzung stoppen und erneut starten, indem sie die Recover-Methode der Session-Klasse aufruft. Nachrichten, deren Empfang zuvor nicht bestätigt wurde, werden erneut übermittelt. Sie werden jedoch möglicherweise nicht in der Reihenfolge übermittelt, in der sie zuvor übermittelt wurden. Es kann vorkommen, dass zwischenzeitlich Nachrichten mit einer höheren Priorität eingetroffen und manche der ursprünglichen Nachrichten abgelaufen sind. In der Punkt-zu-Punkt-Domäne wurden einige der ursprünglichen Nachrichten möglicherweise bereits von einer anderen Anwendung verarbeitet.

Eine Anwendung kann feststellen, ob eine Nachricht erneut übermittelt wird, indem sie den Inhalt des Headerfelds 'JMSRedelivered' der Nachricht überprüft. Hierfür ruft die Anwendung die Methode 'Get JMSRedelivered' der Message-Klasse auf.

Zugehörige Konzepte

Durchgeführte Sitzungen
XMS-Anwendungen können lokale Transaktionen ausführen. Eine *lokale Transaktion* ist eine Transaktion, die Änderungen umfasst, welche nur die Ressourcen des Warteschlangenmanagers oder Service Integration Bus betreffen, mit dem die Anwendung verbunden ist.

Asynchrone Nachrichtenübermittlung
XMS führt alle asynchronen Nachrichtenübermittlungen für eine Sitzung über einen einzigen Thread aus. Dies bedeutet, dass immer nur eine Nachrichtenlistenerfunktion oder eine `onMessage()`-Methode nach der anderen ausgeführt werden kann.

Synchrone Nachrichtenübermittlung
Nachrichten werden synchron an eine Anwendung übermittelt, wenn die Anwendung für den Empfang die Receive-Methoden der MessageConsumer-Objekte verwendet.

Nachrichtenübermittlungsmodus
XMS unterstützt zwei Nachrichtenübermittlungsmodi.

Asynchrone Nachrichtenübermittlung

XMS führt alle asynchronen Nachrichtenübermittlungen für eine Sitzung über einen einzigen Thread aus. Dies bedeutet, dass immer nur eine Nachrichtenlistenerfunktion oder eine `onMessage()`-Methode nach der anderen ausgeführt werden kann.

Wenn in einer Sitzung mehr als ein Nachrichtenkonsument Nachrichten asynchron empfängt und eine Nachrichtenlistenerfunktion oder `onMessage()`-Methode gerade eine Nachricht an einen Nachrichtenkonsumenten übermittelt, dann müssen alle anderen Nachrichtenkonsumenten, denen dieselbe Nachricht übermittelt werden soll, warten, bis dieser eine Übermittlungsvorgang abgeschlossen ist. Auch andere Nachrichten, die an diese Sitzung übermittelt werden sollen, müssen ebenso lange warten.

Wenn für eine Anwendung eine gleichzeitige Nachrichtenübermittlung erforderlich ist, sollten Sie mehr als eine Sitzung erstellen, damit XMS mehrere Threads gleichzeitig für die asynchrone Nachrichtenübermittlung nutzen kann. Dies bedeutet, dass dann auch mehrere Nachrichtenlistenerfunktionen oder `onMessage()`-Methoden gleichzeitig ausgeführt werden können.

Eine Sitzung wird nicht in dem Moment asynchron, in dem einem Nachrichtenlistener ein Konsument zugeordnet wird. Eine Sitzung wird erst dann asynchron, wenn die Methode `Connection.Start` aufgerufen wird. Bis zum Aufruf der Methode `Connection.Start` sind alle synchronen Aufrufe zulässig. Die Nachrichtenübermittlung an Konsumenten beginnt mit dem Aufruf von `Connection.Start` .

Wenn in einer asynchronen Sitzung synchrone Aufrufe erforderlich sind, z. B. zum Erstellen eines Nachrichtenkonsumenten oder -produzenten, muss zunächst die Methode `Connection.Stop` aufgerufen werden. Durch erneuten Aufruf von `Connection.Start` wird die Sitzung fortgesetzt und die Nachrich-

tenübermittlung erneut gestartet. Die einzige Ausnahme dazu ist der Thread zur Übermittlung von Sitzungsnachrichten, also der Thread, der Nachrichten an die Callback-Funktion übermittelt. Dieser Thread kann in der Callback-Funktion für Nachrichten jeden beliebigen Aufruf für eine Sitzung ausführen (mit Ausnahme des Aufrufs 'Close').

Anmerkung: Im nicht verwalteten Modus wird der Aufruf MQDISC innerhalb einer Callback-Funktion nicht vom WMQ.NET-Client unterstützt. Eine Clientanwendung kann also innerhalb der Callback-Funktion 'MessageListener' im asynchronen Empfangsmodus keine Sitzungen erstellen (Create) oder schließen (Close). Sie müssen die Sitzung außerhalb der `MessageListener`-Methode erstellen und verwerfen.

Zugehörige Konzepte

Durchgeführte Sitzungen
XMS-Anwendungen können lokale Transaktionen ausführen. Eine *lokale Transaktion* ist eine Transaktion, die Änderungen umfasst, welche nur die Ressourcen des Warteschlangenmanagers oder Service Integration Bus betreffen, mit dem die Anwendung verbunden ist.

Nachrichtenbestätigung
Jede Sitzung ohne Transaktionsunterstützung verfügt über einen Bestätigungsmodus, der bestimmt, wie die von der Anwendung empfangenen Nachrichten bestätigt werden. Es sind drei Bestätigungsmodi verfügbar und die Auswahl des Bestätigungsmodus wirkt sich auf das Design der Anwendung aus.

Synchrone Nachrichtenübermittlung
Nachrichten werden synchron an eine Anwendung übermittelt, wenn die Anwendung für den Empfang die Receive-Methoden der MessageConsumer-Objekte verwendet.

Nachrichtenübermittlungsmodus
XMS unterstützt zwei Nachrichtenübermittlungsmodi.

Synchrone Nachrichtenübermittlung

Nachrichten werden synchron an eine Anwendung übermittelt, wenn die Anwendung für den Empfang die Receive-Methoden der MessageConsumer-Objekte verwendet.

Bei Verwendung der Receive-Methoden kann für eine Anwendung festgelegt werden, dass sie eine bestimmte Zeit lang oder ohne zeitliche Begrenzung auf eine Nachricht warten soll. Wenn die Anwendung hingegen überhaupt nicht auf eine Nachricht warten soll, kann die Receive-Methode ohne jegliche Wartezeit ('No Wait') festgelegt werden.

Zugehörige Konzepte

Durchgeführte Sitzungen
XMS-Anwendungen können lokale Transaktionen ausführen. Eine *lokale Transaktion* ist eine Transaktion, die Änderungen umfasst, welche nur die Ressourcen des Warteschlangenmanagers oder Service Integration Bus betreffen, mit dem die Anwendung verbunden ist.

Nachrichtenbestätigung
Jede Sitzung ohne Transaktionsunterstützung verfügt über einen Bestätigungsmodus, der bestimmt, wie die von der Anwendung empfangenen Nachrichten bestätigt werden. Es sind drei Bestätigungsmodi verfügbar und die Auswahl des Bestätigungsmodus wirkt sich auf das Design der Anwendung aus.

Asynchrone Nachrichtenübermittlung
XMS führt alle asynchronen Nachrichtenübermittlungen für eine Sitzung über einen einzigen Thread aus. Dies bedeutet, dass immer nur eine Nachrichtenlistenerfunktion oder eine `onMessage()`-Methode nach der anderen ausgeführt werden kann.

Nachrichtenübermittlungsmodus
XMS unterstützt zwei Nachrichtenübermittlungsmodi.

Nachrichtenübermittlungsmodus

XMS unterstützt zwei Nachrichtenübermittlungsmodi.

- *Persistente* Nachrichten werden einmal übermittelt. Dabei übernimmt ein Messaging-Server spezielle Vorsichtsmaßnahmen wie beispielsweise die Protokollierung der Nachrichten, um sicherzustellen, dass persistente Nachrichten, selbst im Falle eines Fehlers, während der Übermittlung, nicht verloren gehen.

- *Nicht persistente* Nachrichten werden nicht mehr als einmal übermittelt. Da nicht persistente Nachrichten im Falle eines Fehlers während der Übermittlung verloren gehen können, sind sie nicht so zuverlässig wie persistente Nachrichten.

Die Wahl des geeigneten Übermittlungsmodus ist ein Kompromiss zwischen Zuverlässigkeit und Leistung. Nicht persistente Nachrichten werden normalerweise schneller übermittelt als persistente Nachrichten.

Zugehörige Konzepte

Durchgeführte Sitzungen
XMS-Anwendungen können lokale Transaktionen ausführen. Eine *lokale Transaktion* ist eine Transaktion, die Änderungen umfasst, welche nur die Ressourcen des Warteschlangenmanagers oder Service Integration Bus betreffen, mit dem die Anwendung verbunden ist.

Nachrichtenbestätigung
Jede Sitzung ohne Transaktionsunterstützung verfügt über einen Bestätigungsmodus, der bestimmt, wie die von der Anwendung empfangenen Nachrichten bestätigt werden. Es sind drei Bestätigungsmodi verfügbar und die Auswahl des Bestätigungsmodus wirkt sich auf das Design der Anwendung aus.

Asynchrone Nachrichtenübermittlung
XMS führt alle asynchronen Nachrichtenübermittlungen für eine Sitzung über einen einzigen Thread aus. Dies bedeutet, dass immer nur eine Nachrichtenlistenerfunktion oder eine onMessage()-Methode nach der anderen ausgeführt werden kann.

Synchrone Nachrichtenübermittlung
Nachrichten werden synchron an eine Anwendung übermittelt, wenn die Anwendung für den Empfang die Receive-Methoden der MessageConsumer-Objekte verwendet.

Ziele

XMS-Anwendungen geben mit einem Destination-Objekt das Ziel für gesendete Nachrichten und die Quelle von empfangenen Nachrichten an.

Eine XMS-Anwendung kann entweder ein Destination-Objekt während der Laufzeit erstellen oder ein vordefiniertes Ziel aus dem Repository für verwaltete Objekte abrufen.

Als flexibelste Möglichkeit zum Angeben eines Ziels kann eine XMS-Anwendung das Ziel, ähnlich wie bei einem ConnectionFactory-Objekt, als verwaltetes Objekt definieren. Bei diesem Ansatz können Anwendungen, die in C, C++, .NET-Sprachen und Java geschrieben sind, die Definitionen des Ziels gemeinsam nutzen. Die Eigenschaften von verwalteten Destination-Objekten können geändert werden, ohne Änderungen am Code vorzunehmen.

Um Ziele für .NET-Anwendungen zu erstellen, verwenden Sie die Methode 'CreateTopic' oder 'CreateQueue'. Beide Methoden sind in der .NET-API sowohl im ISession- als auch im XMSFactoryFactory-Objekt verfügbar. Weitere Informationen finden Sie in den Abschnitten „Ziele in .NET” auf Seite 51 und „IDestination” auf Seite 116.

Zugehörige Verweise

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Themen-URIs

Der Uniform Resource Identifier (URI) eines Themas gibt den Namen des Themas und ggf. auch eine oder mehrere seiner Eigenschaften an.

Ein Themen-URI beginnt mit der Sequenz 'topic://' gefolgt von dem Namen des Themas. Darüber hinaus kann er optional eine Liste der Name/Wert-Paare enthalten, mit denen die anderen Themeneigenschaften festgelegt werden. Ein Themenname darf nicht leer sein.

Es folgte ein Beispiel für ein Fragment aus einem .NET-Code:

```
topic = session.CreateTopic("topic://Sport/Football/Results?multicast=7");
```

Weitere Informationen zu den Eigenschaften eines Themas, einschließlich des Namens und der gültigen Werte, die Sie in einem URI verwenden können, finden Sie im Abschnitt „Eigenschaften von Destination" auf Seite 198.

Beim Angeben eines Themen-URI, der in einer Subskription verwendet werden soll, können auch Platzhalter verwendet werden. Die Syntax für die Verwendung der Platzhalter hängt vom Verbindungstyp und der Brokerversion ab. Folgende Optionen sind verfügbar:

- IBM WebSphere MQ V7.0-Queue Manager mit Platzhalterzeichenformat auf Zeichenebene
- IBM WebSphere MQ V7.0-Queue Manager mit Platzhalterzeichenformat auf Themenebene
- IBM WebSphere MQ V6.0 Queue Manager mit Broker V1 (IBM WebSphere MQ V6.0 Publish/Subscribe)
- IBM WebSphere MQ V6.0 mit oder Echtzeitverbindung zum Broker V2 (WebSphere Event Broker oder WebSphere Message Broker)
- WebSphere Service Integration Bus

IBM WebSphere MQ V7.0-Queue Manager mit Platzhalterzeichenformat auf Zeichenebene

IBM WebSphere MQ V7.0 Queue Manager mit Platzhalterformat auf Zeichenebene verwendet die folgenden Platzhalterzeichen:

* für 0 oder mehr Zeichen
? für 1 Zeichen
% für ein Escapezeichen

In Tabelle 1 auf Seite 33 sind einige Beispiele für die Verwendung dieses Platzhalterformats aufgeführt.

Tabelle 1. Beispiel-URIs mit Platzhalterzeichen auf Zeichenebene für Warteschlangenmanager von IBM WebSphere MQ Version 7.0

Uniform Resource Identifier (URI)	**Entsprechungen**	**Beispiele**
"topic://Sport*Results"	Alle Themen, die mit "Sport" beginnen und mit "Results" enden	"topic://SportsResults" und "topic://Sport/Hockey/National/Div3/Results"
"topic://Sport?Results"	Alle Themen, die mit "Sport" beginnen, gefolgt einem einzelnen Zeichen, gefolgt von "Results"	"topic://SportsResults" und "topic://SportXResults"
"topic://Sport/*ball*/Div?/Results/*/???"	Themen	"topic://Sport/Football/Div1/Results/2002/Nov" und "topic://Sport/Netball/National/Div3/Results/02/Jan"

IBM WebSphere MQ V7.0-Queue Manager mit Platzhalterzeichenformat auf Themenebene

IBM WebSphere MQ V7.0 Queue Manager mit Platzhalterformat auf Themenebene verwendet die folgenden Platzhalter:

für mehrere Ebenen
+ für eine einzige Ebene

In Tabelle 2 auf Seite 34 sind einige Beispiele für die Verwendung dieses Platzhalterformats aufgeführt.

Tabelle 2. Beispiel-URIs mit Platzhalterzeichen auf Themenebene für Warteschlangenmanager von IBM WebSphere MQ Version 7.0		
Uniform Resource Identifier (URI)	**Entsprechungen**	**Beispiele**
"topic://Sport/+/Results"	Alle Themen mit einem einzigen Gliederungsebenennamen zwischen den Ebenen "Sport" und "Results"	"topic://Sport/Football/Results" und "topic://Sport/Ju-Jitsu/Results"
"topic://Sport/#/Results"	Alle Themen die mit "Sport/" beginnen und mit "/Results" enden	"topic://Sport/Football/Results" und "topic://Sport/Hockey/National/Div3/Results"
"topic://Sport/Football/#"	Alle Themen, die mit "Sport/Football/" beginnen	"topic://Sport/Football/Results" und "topic://Sport/Football/TeamNews/Signings/Managerial"

IBM WebSphere MQ V6.0 Queue Manager mit Broker V1

IBM WebSphere MQ V6.0 Queue Manager mit Broker V1 verwendet die folgenden Platzhalterzeichen:

* für 0 oder mehr Zeichen
? für 1 Zeichen
% für ein Escapezeichen

In Tabelle 1 auf Seite 33 sind einige Beispiele für die Verwendung dieses Platzhalterformats aufgeführt.

IBM WebSphere MQ V6.0 mit oder Echtzeitverbindung zu einem Broker V2

IBM WebSphere MQ V6.0 mit Echtzeitverbindung zu einem Broker V2 verwendet die folgenden Platzhalterzeichen:

für mehrere Ebenen
+ für eine einzige Ebene

In Tabelle 2 auf Seite 34 sind einige Beispiele für die Verwendung dieses Platzhalterformats aufgeführt.

WebSphere Service Integration Bus

WebSphere Service Integration Bus verwendet folgende Platzhalterzeichen:

* für eine beliebige Anzahl von Zeichen auf einer Gliederungsebene
// für 0 oder mehr Ebenen
//. für 0 oder mehr Ebenen am Ende eines Themenausdrucks

In Tabelle 3 auf Seite 34 sind einige Beispiele für die Verwendung dieses Platzhalterformats aufgeführt.

Tabelle 3. Beispiel-URIs mit Platzhalterschema für WebSphere Service Integration Bus		
Uniform Resource Identifier (URI)	**Entsprechungen**	**Beispiele**
"topic://Sport/*ball/Results"	Alle Themen mit einem einzigen Gliederungsebenennamen, der mit "ball" endet, zwischen den Ebenen "Sport" und "Results"	"topic://Sport/Football/Results" und "topic://Sport/Netball/Results"
"topic://Sport//Results"	Alle Themen die mit "Sport/" beginnen und mit "/Results" enden	"topic://Sport/Football/Results" und "topic://Sport/Hockey/National/Div3/Results"

Tabelle 3. Beispiel-URIs mit Platzhalterschema für WebSphere Service Integration Bus (Forts.)		
Uniform Resource Identifier (URI)	**Entsprechungen**	**Beispiele**
"topic://Sport/Football//."	Alle Themen, die mit "Sport/Football/" beginnen	"topic://Sport/Football/Results" und "topic://Sport/Football/TeamNews/Signings/Managerial"
"topic://Sport/*ball//Results//."	Themen	"topic://Sport/Football/Results" und "topic://Sport/Netball/National/Div3/Results/2002/November"

Zugehörige Konzepte

Warteschlangen-URIs
Der Uniform Resource Identifier (URI) einer Warteschlange gibt den Namen der Warteschlange und ggf. auch eine oder mehrere ihrer Eigenschaften an.

Temporäre Ziele
XMS-Anwendungen können temporäre Ziele erstellen und verwenden.

Zielplatzhalterzeichen
XMS bietet Unterstützung für Zielplatzhalterzeichen, um sicherzustellen, dass Platzhalterzeichen an die Stelle übergeben werden können, an der sie für den Abgleich benötigt werden. Es gibt für jeden Servertyp, mit dem XMS zusammenarbeiten kann, ein anderes Platzhalterschema.

Zugehörige Verweise

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Warteschlangen-URIs

Der Uniform Resource Identifier (URI) einer Warteschlange gibt den Namen der Warteschlange und ggf. auch eine oder mehrere ihrer Eigenschaften an.

Der URI für eine Warteschlange beginnt mit der Zeichenfolge queue://, gefolgt vom Namen der Warteschlange; er kann auch eine Liste von Name/Wert-Paaren enthalten, die die übrigen Warteschlangeneigenschaften festlegen.

Bei IBM WebSphere MQ-Warteschlangen (jedoch nicht bei Warteschlangen des Standard-Messaging-Providers für WebSphere Application Server) kann der Warteschlangenmanager, in dem sich die Warteschlange befindet, vor der Warteschlange angegeben werden. In diesem Fall wird der Warteschlangenmanagername durch einen Schrägstrich (/) vom Warteschlangennamen getrennt.

Falls ein Warteschlangenmanager angegeben wird, muss es derjenige sein, mit dem XMS für die Verbindung, die diese Warteschlange verwendet, direkt verbunden ist, oder er muss über diese Warteschlange zugänglich sein. Ferne Warteschlangenmanager werden nur für das Abrufen von Nachrichten aus Warteschlangen, nicht jedoch für das Einreihen von Nachrichten in Warteschlangen unterstützt. Weitere Informationen finden Sie in der Dokumentation zu IBM WebSphere MQ-Warteschlangenmanagern.

Falls kein Warteschlangenmanager angegeben wird, ist es irrelevant, ob das zusätzliche Trennzeichen '/' eingefügt oder weggelassen wird, da es keinen Einfluss auf die Definition der Warteschlange hat.

Alle folgenden Warteschlangendefinitionen geben auf unterschiedliche Weise dieselbe Warteschlange an: eine IBM WebSphere MQ-Warteschlange mit dem Namen 'QB' in einem Warteschlangenmanager mit dem Namen 'QM_A', mit dem XMS direkt verbunden ist:

```
queue://QB
queue:///QB
queue://QM_A/QB
```

Zugehörige Konzepte

Themen-URIs
Der Uniform Resource Identifier (URI) eines Themas gibt den Namen des Themas und ggf. auch eine oder mehrere seiner Eigenschaften an.

Temporäre Ziele
XMS-Anwendungen können temporäre Ziele erstellen und verwenden.

Zielplatzhalterzeichen
XMS bietet Unterstützung für Zielplatzhalterzeichen, um sicherzustellen, dass Platzhalterzeichen an die Stelle übergeben werden können, an der sie für den Abgleich benötigt werden. Es gibt für jeden Servertyp, mit dem XMS zusammenarbeiten kann, ein anderes Platzhalterschema.

Zugehörige Verweise

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Temporäre Ziele

XMS-Anwendungen können temporäre Ziele erstellen und verwenden.

Eine Anwendung verwendet ein temporäres Ziel normalerweise, um Antworten auf Anforderungsnachrichten zu empfangen. Um das Ziel anzugeben, an das die Antwort auf eine Anforderungsnachricht gesendet werden soll, ruft eine Anwendung die Methode 'Set JMSReplyTo' des Nachrichtenobjekts (Message) auf, das die Anforderungsnachricht darstellt. Das in dem Aufruf angegebene Ziel kann ein temporäres Ziel sein.

Obwohl zum Erstellen eines temporären Ziels eine Sitzung verwendet wird, ist der Geltungsbereich eines temporären Ziels eigentlich die Verbindung, die zum Erstellen der Sitzung verwendet wurde. Alle Sitzungen der Verbindung können Nachrichtenproduzenten und Nachrichtenkonsumenten für das temporäre Ziel erstellen. Das temporäre Ziel bleibt bestehen, bis es explizit gelöscht wird oder die Verbindung endet, je nachdem, welches Ereignis zuerst eintritt.

Wenn eine Anwendung eine temporäre Warteschlange erstellt, wird die Warteschlange auf dem Messaging-Server erstellt, mit dem die Anwendung verbunden ist. Wenn die Anwendung mit einem Warteschlangenmanager verbunden ist, wird eine dynamische Warteschlange aus der Modellwarteschlange erstellt, deren Name durch die Eigenschaft XMSC_WMQ_TEMPORARY_MODEL angegeben wird. Das Präfix, das zur Bildung des Namens der dynamischen Warteschlange verwendet wird, wird durch die Eigenschaft XMSC_WMQ_TEMP_Q_PREFIX angegeben. Wenn die Anwendung mit einem Service Integration Bus verbunden ist, wird eine temporäre Warteschlange im Bus erstellt und das Präfix, das zur Bildung des Namens der temporären Warteschlange verwendet wird, wird durch die Eigenschaft XMSC_WPM_TEMP_Q_PREFIX angegeben.

Wenn eine Anwendung, die mit einem Service Integration Bus verbunden ist, ein temporäres Topic erstellt, wird das Präfix, das zur Bildung des Namens des temporären Topic verwendet wird, durch die Eigenschaft XMSC_WPM_TEMP_TOPIC_PREFIX angegeben.

Zugehörige Konzepte

Themen-URIs
Der Uniform Resource Identifier (URI) eines Themas gibt den Namen des Themas und ggf. auch eine oder mehrere seiner Eigenschaften an.

Warteschlangen-URIs
Der Uniform Resource Identifier (URI) einer Warteschlange gibt den Namen der Warteschlange und ggf. auch eine oder mehrere ihrer Eigenschaften an.

Zielplatzhalterzeichen
XMS bietet Unterstützung für Zielplatzhalterzeichen, um sicherzustellen, dass Platzhalterzeichen an die Stelle übergeben werden können, an der sie für den Abgleich benötigt werden. Es gibt für jeden Servertyp, mit dem XMS zusammenarbeiten kann, ein anderes Platzhalterschema.

Zugehörige Verweise

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Zielplatzhalterzeichen

XMS bietet Unterstützung für Zielplatzhalterzeichen, um sicherzustellen, dass Platzhalterzeichen an die Stelle übergeben werden können, an der sie für den Abgleich benötigt werden. Es gibt für jeden Servertyp, mit dem XMS zusammenarbeiten kann, ein anderes Platzhalterschema.

Verfügbare Schemas:

Verbindungstyp	**Platzhalter-schema**	**Beschreibung**
IBM WebSphere MQ-Warte-schlangenmanager	*	0 oder mehr Zeichen
	?	1 Zeichen
	%	Escapezeichen
Echtzeitverbindung zu einem Broker	#	Abgleich mehrerer Ebenen
	+	Abgleich einer einzelnen Ebene
WebSphere Service Integration Bus	*	Abgleich aller Zeichen auf einer einzigen Ebene in der Hierarchie
	//	Abgleich 0 oder mehrerer Ebenen
	//.	Abgleich 0 oder mehrerer Ebenen (am Ende eines Topic-Ausdrucks)

Siehe auch Topicnamen und Verwendung von Platzhalterzeichen in Topicausdrücken in der Dokumentation zu WebSphere Application Server .

Zugehörige Konzepte

Themen-URIs
Der Uniform Resource Identifier (URI) eines Themas gibt den Namen des Themas und ggf. auch eine oder mehrere seiner Eigenschaften an.

Warteschlangen-URIs
Der Uniform Resource Identifier (URI) einer Warteschlange gibt den Namen der Warteschlange und ggf. auch eine oder mehrere ihrer Eigenschaften an.

Temporäre Ziele
XMS-Anwendungen können temporäre Ziele erstellen und verwenden.

Zugehörige Verweise

IDestination (für die .NET-Schnittstelle)

Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Nachrichtenproduzenten

In XMS kann einem Nachrichtenproduzenten bei dessen Erstellung entweder genau ein gültiges Ziel oder gar kein Ziel zugeordnet werden. Wenn ein Nachrichtenproduzent erstellt wird, ohne dass ihm ein Ziel zugeordnet wird, muss stattdessen beim Senden einer Nachricht ein gültiges Ziel angegeben werden.

Nachrichtenproduzent ohne zugeordnetes Ziel

In XMS .NET kann ein Nachrichtenproduzent mit einem NULL-Ziel erstellt werden.

Um bei Verwendung der .NET-API einen Nachrichtenproduzenten ohne zugeordnetes Ziel zu erstellen, muss NULL als Parameter an die Methode `CreateProducer()` des ISession-Objekts übergeben werden (z. B. `session.CreateProducer(null)`). Beim Senden der Nachricht muss jedoch ein gültiges Ziel angegeben werden.

Nachrichtenproduzenten mit zugeordnetem Ziel

In diesem Szenario wird ein Nachrichtenproduzent erstellt, dem ein gültiges Ziel zugeordnet wird. In diesem Fall muss bei der Sendeoperation kein Ziel angegeben werden.

Nachrichtenkonsumenten

Bei den Nachrichtenkonsumenten wird zwischen permanenten und nicht permanenten Subskribenten sowie zwischen synchronen und asynchronen Nachrichtenkonsumenten unterschieden.

Permanente Subskribenten

Ein permanenter Subskribent ist ein Nachrichtenkonsument, der alle in einem Thema veröffentlichten Nachrichten empfängt, einschließlich der Nachrichten, die veröffentlicht werden, während der Subskribent inaktiv ist.

Die Informationen in diesem Thema sind nur relevant, wenn eine Anwendung eine Verbindung zu einem IBM WebSphere MQ -Warteschlangenmanager oder einemWebSphere Service Integration Bus herstellt. Für Echtzeitverbindungen zu einem Broker sind die Informationen in diesem Abschnitt irrelevant.

Um einen permanenten Subskribenten für ein Thema zu erstellen, ruft eine Anwendung die Methode 'Create Durable Subscriber' eines Session-Objekts auf und gibt als Parameter einen Namen zum Angeben der permanenten Subskription an sowie ein Destination-Objekt, das das Thema angibt. Die Anwendung kann einen permanenten Subskribenten mit oder ohne Nachrichtenselektor erstellen und angeben, ob der permanente Subskribent Nachrichten empfangen soll, die über seine eigene Verbindung veröffentlicht werden.

Der Sitzung, die zum Erstellen eines permanenten Subskribenten verwendet wird, muss eine Client-ID zugeordnet sein. Die Client-ID der Sitzung ist mit der Client-ID der Verbindung identisch, die zum Erstellen der Sitzung verwendet wird. Weitere Informationen zum Angeben der Client-ID finden Sie im Abschnitt „ConnectionFactory- und Connection-Objekte” auf Seite 24.

Der Name, mit dem die permanente Subskription angegeben wird, muss innerhalb der Client-ID eindeutig sein. Daher ist die Client-ID Bestandteil der vollständigen, eindeutigen ID der permanenten Subskription. Der Messaging-Server zeichnet die permanente Subskription in einem Datensatz auf und stellt auf diese Weise sicher, dass alle in einem Thema veröffentlichten Nachrichten aufbewahrt werden, bis sie vom permanenten Subskribenten bestätigt wurden oder bis sie ablaufen.

Der Messaging-Server setzt die Aufzeichnung der permanenten Subskription selbst dann fort, nachdem der permanente Subskribenten geschlossen wurde. Damit eine zuvor erstellte permanente Subskription weiterverwendet werden kann, muss eine Anwendung einen permanenten Subskribenten erstellen und dabei denselben Subskriptionsnamen und eine Sitzung mit derselben Client-ID verwenden, die der ur-

sprünglichen permanenten Sitzung zugeordnet waren. Zu jedem Zeitpunkt kann immer nur eine Sitzung einen permanenten Subskribenten für eine bestimmte permanente Subskription haben.

Der Geltungsbereich einer permanenten Subskription ist der Messaging-Server, der die Subskription aufzeichnet. Wenn zwei Anwendungen, die mit unterschiedlichen Messaging-Servern verbunden sind, jeweils einen permanenten Subskribenten mit demselben Subskriptionsnamen und derselben Client-ID erstellen, entstehen dabei zwei völlig voneinander unabhängige permanente Subskriptionen.

Um eine permanente Subskription zu löschen, ruft eine Anwendung die Methode 'Unsubscribe' eines Session-Objekts auf und gibt als Parameter den Namen der permanenten Subskription an. Die Client-ID, die der Sitzung zugeordnet ist, muss dabei dieselbe sein, die auch der permanenten Subskription zugeordnet ist. Der Messaging-Server löscht den aufgezeichneten Datensatz der permanenten Subskription und sendet danach keine weiteren Nachrichten an den permanenten Subskribenten.

Um eine vorhandene Subskription zu ändern, kann eine Anwendung einen permanenten Subskribenten erstellen und dabei denselben Subskriptionsnamen und dieselbe Client-ID, jedoch ein anderes Thema und/oder einen anderen Nachrichtenselektor angeben. Eine permanente Subskription zu ändern hat dieselben Auswirkungen wie die vorhandene Subskription zu löschen und eine neue zu erstellen.

Für eine Anwendung, die eine Verbindung zu einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 herstellt, verwaltet XMS die Subskribentenwarteschlangen. Daher ist es nicht erforderlich, dass die Anwendung einen Subskribentennamen angibt. Falls dennoch eine Subskribentenwarteschlange angegeben wird, ignoriert XMS dies.

Für eine Anwendung, die eine Verbindung zu einem Warteschlangenmanager von IBM WebSphere MQ Version 6.0 herstellt, muss jedoch jedem permanenten Subskribenten eine eigene Subskribentenwarteschlange zugewiesen sein. Geben Sie den Namen der Subskribentenwarteschlange an, indem Sie die Eigenschaft XMSC_WMQ_DUR_SUBQ des Destination-Objekts, das das Thema darstellt, festlegen. Der Standardname der Subskribentenwarteschlange lautet SYSTEM.JMS.D.SUBSCRIBER.QUEUE.

Permanente Subskribenten, die Verbindungen zu Warteschlangenmanagern von IBM WebSphere MQ Version 6.0 herstellen, können eine einzelne Subskribentenwarteschlange gemeinsam nutzen oder jeder permanente Subskribent kann seine Nachrichten aus seiner eigenen exklusiven Subskribentenwarteschlange abrufen. Hinweise dazu, welcher Ansatz am besten für Ihre Anwendung geeignet ist, finden Sie im Handbuch *IBM WebSphere MQ Using Java*.

Beachten Sie, dass Sie die Subskribentenwarteschlange einer permanenten Subskription nicht ändern können. Die einzige Möglichkeit, die Subskribentenwarteschlange zu ändern, ist die Subskription zu löschen und eine neue zu erstellen.

Wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt, muss jeder permanente Subskribent eine festgelegte Ausgangsposition für permanente Subskriptionen haben. Um die Ausgangsposition für permanente Subskriptionen für alle permanenten Subskribenten anzugeben, die dieselbe Verbindung verwenden, legen Sie die Eigenschaft XMSC_WPM_DUR_SUB_HOME des ConnectionFactory-Objekts, das zum Erstellen der Verbindung verwendet wird, entsprechend fest. Um die Ausgangsposition für permanente Subskriptionen für ein bestimmtes Thema anzugeben, legen Sie die Eigenschaft 'XMSC_WPM_DUR_SUB_HOME' des Destination-Objekts, das das Thema angibt, entsprechend fest. Die Ausgangsposition für permanente Subskriptionen muss für eine Verbindung angegeben werden, bevor eine Anwendung einen permanenten Subskribenten erstellen kann, der diese Verbindung verwendet. Ein Wert, der für ein Ziel angegeben wird, überschreibt in jedem Fall den für eine Verbindung angegebenen Wert.

Nicht permanente Subskribenten

Ein nicht permanenter Subskribent ist ein Nachrichtenkonsument, der nur die Nachrichten empfängt, die veröffentlicht werden, während der Subskribent aktiv ist. Nachrichten, die veröffentlicht werden, während der Subskribent inaktiv ist, gehen hingegen verloren.

Die Informationen in diesem Thema sind nur relevant, wenn Sie Publish/Subscribe -Messaging über IBM WebSphere MQ V6.0 -Warteschlangenmanager verwenden.

Wenn Konsumentenobjekte nicht vor oder während dem Schließen der Verbindung gelöscht werden, können Nachrichten für Subskribenten, die nicht mehr aktiv sind, in den Brokerwarteschlangen verbleiben.

In dieser Situation können die Warteschlangen von diesen Nachrichten bereinigt werden. Verwenden Sie dazu das Bereinigungsdienstprogramm, das von den IBM WebSphere MQ bereitgestellt wird. Details zur Nutzung dieses Dienstprogramms finden Sie im Handbuch *IBM WebSphere MQ Using Java.* Möglicherweise müssen Sie auch die Warteschlangenlänge der Subskribentenwarteschlange erhöhen, wenn viele Nachrichten in dieser Warteschlange verbleiben.

Synchrone Nachrichtenkonsumenten

Der synchrone Nachrichtenkonsument empfängt Nachrichten aus einer Warteschlange synchron.

Ein synchroner Nachrichtenkonsument empfängt zu jedem Zeitpunkt immer nur eine Nachricht. Wenn die Methode `Receive(wait interval)` verwendet wird, wartet der Aufruf nur eine in Millisekunden angegebene Zeit lang auf eine Nachricht oder bis der Nachrichtenkonsument geschlossen wird.

Wenn die Methode 'ReceiveNoWait()' verwendet wird, empfängt der synchrone Nachrichtenkonsument Nachrichten ohne jede Verzögerung. Sobald die nächste Nachricht verfügbar ist, wird sie sofort empfangen, andernfalls wird ein Verweis auf ein Message-Objekt mit dem Wert 'null' zurückgegeben.

Asynchrone Nachrichtenkonsumenten

Der asynchrone Nachrichtenkonsument empfängt Nachrichten aus einer Warteschlange asynchron. Der von der Anwendung registrierte Nachrichtenlistener wird aufgerufen, sobald eine neue Nachricht in der Warteschlange verfügbar ist.

Nicht verarbeitbare Nachrichten

Eine nicht verarbeitbare Nachricht ist eine Nachricht, die nicht von einer empfangenden MDB-Anwendung verarbeitet werden kann. Wenn eine nicht verarbeitbare Nachricht auftritt, kann das XMS -Objekt MessageConsumer sie gemäß den beiden Warteschlangeneigenschaften BOQNAME und BOTHRESH erneut in die Warteschlange stellen.

Unter bestimmten Umständen kann eine an eine MDB zugestellte Nachricht in einer IBM WebSphere MQ-Warteschlange rückgängig gemacht werden. Dies kann beispielsweise der Fall sein, wenn eine Nachricht innerhalb einer Arbeitseinheit zugestellt wird, die dann rückgängig gemacht wird. Eine Nachricht, für die ein Rollback erfolgt ist, wird normalerweise erneut zugestellt, allerdings kann eine falsch formatierte Nachricht dazu führen, dass eine MDB wiederholt fehlschlägt und daher keine Zustellung möglich ist. Eine solche Nachricht wird als nicht verarbeitbare Nachricht bezeichnet. Sie können in der Konfiguration von IBM WebSphere MQ festlegen, dass eine nicht verarbeitbare Nachricht zur weiteren Untersuchung automatisch an eine andere Warteschlange übertragen oder gelöscht wird. Weitere Informationen zur Konfiguration von IBM WebSphere MQ auf diese Weise finden Sie im Abschnitt Handhabung von nicht verarbeitbaren Nachrichten in ASF.

Manchmal kommt eine fehlerhaft formatierte Nachricht in einer Warteschlange an. In diesem Kontext bedeutet 'fehlerhaft formatiert', dass die empfangende Anwendung die Nachricht nicht ordnungsgemäß verarbeiten kann. Eine solche Nachricht kann dazu führen, dass die empfangende Anwendung fehlschlägt und diese fehlerhaft formatierte Nachricht zurücksetzt. Die Nachricht kann dann mehrfach an die Eingabewarteschlange übermittelt und mehrfach von der Anwendung zurückgesetzt werden. Diese Nachrichten werden als nicht verarbeitbare Nachrichten bezeichnet. Das XMS-Objekt 'MessageConsumer' erkennt nicht verarbeitbare Nachrichten und leitet diese an ein alternatives Ziel um.

Der IBM WebSphere MQ-Warteschlangenmanager zeichnet die Häufigkeit auf, mit der die einzelnen Nachrichten zurückgesetzt wurden. Sobald diese Anzahl einen konfigurierbaren Schwellenwert erreicht, reiht der Nachrichtenkonsument die Nachricht in eine namentlich genannte Rücksetzwarteschlange ein. Wenn diese Neueinreihung aus irgendeinem Grund fehlschlägt, wird die Nachricht aus der Eingabewarteschlange entfernt und entweder neu in die Warteschlange für nicht zustellbare Nachrichten eingereiht oder gelöscht.

XMS-Objekte des Typs 'ConnectionConsumer' handhaben nicht verarbeitbare Nachrichten auf dieselbe Weise und mit denselben Warteschlangeneigenschaften. Wenn mehrere Verbindungskonsumenten dieselbe Warteschlange überwachen, kann es vorkommen, dass die nicht verarbeitbare Nachricht häufiger an eine Anwendung übermittelt wird, als durch den Schwellenwert festgelegt, bevor die Neueinreihung durchgeführt wird. Dieses Verhalten ist auf die Art und Weise zurückzuführen, in der einzelne Verbin-

dungskonsumenten Warteschlangen überwachen und nicht verarbeitbare Nachrichten erneut in die Warteschlange stellen.

Der Schwellenwert und der Name der Rücksetzwarteschlange sind Attribute einer IBM WebSphere MQ-Warteschlange. Diese Attribute haben die Namen 'BackoutThreshold' und 'BackoutRequeueQName'. Sie werden auf folgende Warteschlangen angewendet:

- Beim Punkt-zu-Punkt-Messaging handelt es sich um die zugrunde liegende lokale Warteschlange. Dies ist wichtig, wenn Nachrichtenkonsumenten und Verbindungskonsumenten Aliasnamen einer Warteschlange verwenden.
- Beim Publish/Subscribe-Messaging im normalen Modus des IBM WebSphere MQ-Messaging-Providers wird die verwaltete Warteschlange des Themas auf Basis der Modellwarteschlange erstellt.
- Beim Publish/Subscribe-Messaging im Migrationsmodus des IBM WebSphere MQ-Messaging-Providers gelten sie für die CCSUB-Warteschlange, die im TopicConnectionFactory-Objekt definiert ist, oder für die CCDSUB-Warteschlange, die im Topic-Objekt definiert ist.

Geben Sie folgenden MQSC-Befehl aus, um die Attribute BackoutThreshold und BackoutRequeueQName festzulegen:

```
ALTER QLOCAL(your.queue.name) BOTHRESH(threshold value)
BOQNAME(your.backout.queue.name)
```

Wenn Ihr System beim Publish/Subscribe-Messaging eine dynamische Warteschlange für jede Subskription erstellt, werden diese Attributwerte aus der Modellwarteschlange SYSTEM.JMS.MODEL.QUEUEder IBM WebSphere MQ -Klassen für JMS abgerufen. Um diese Einstellungen zu ändern, verwenden Sie Folgendes:

```
ALTER QMODEL(SYSTEM.JMS.MODEL.QUEUE) BOTHRESH(threshold value)
BOQNAME(your.backout.queue.name)
```

Wenn der Rücksetzschwellenwert null ist, ist die Behandlung nicht verarbeitbarer Nachrichten inaktiviert und nicht verarbeitbare Nachrichten bleiben in der Eingabewarteschlange. Andernfalls wird die Nachricht an die namentlich genannte Rücksetzwarteschlange gesendet, sobald der Rücksetzungszähler den Schwellenwert erreicht. Wenn der Rücksetzungszähler den Schwellenwert erreicht, die Nachricht aber nicht in die Rücksetzwarteschlange eingereiht werden kann, wird die Nachricht an die Warteschlange für nicht zustellbare Nachrichten gesendet oder gelöscht. Diese Situation tritt ein, wenn die Rücksetzwarteschlange nicht definiert ist oder wenn das MessageConsumer-Objekt die Nachricht nicht an die Rücksetzwarteschlange senden kann.

Behandlung nicht verarbeitbarer Nachrichten in ASF
Wenn Sie Anwendungsserverfunktionen (Application Server Facilities, ASF) verwenden, werden nicht verarbeitbare Nachrichten nicht vom MessageConsumer, sondern vom ConnectionConsumer verarbeitet. Der ConnectionConsumer reiht Nachrichten in Übereinstimmung mit den Eigenschaften 'BackoutThreshold' und 'BackoutRequeueQName' der Warteschlange neu ein.

Wenn eine Anwendung ConnectionConsumers verwendet, hängen die Umstände, unter denen eine Nachricht zurückgesetzt wird, von der Sitzung ab, die der Anwendungsserver bereitstellt:

- Wenn die Sitzung nicht transaktionsbasiert und AUTO_ACKNOWLEDGE oder DUPS_OK_ACKNOWLEDGE festgelegt ist, wird eine Nachricht nur nach einem Systemfehler oder einer unerwarteten Beendigung der Anwendung zurückgesetzt.
- Wenn die Sitzung nicht transaktionsbasiert und CLIENT_ACKNOWLEDGE festgelegt ist, können unbestätigte Nachrichten vom Anwendungsserver mit dem Aufruf von `Session.recover()` zurückgesetzt werden.

  Normalerweise ruft die Clientimplementierung von MessageListener oder der Anwendungsserver `Message.acknowledge()` auf. `Message.acknowledge()` bestätigt alle Nachrichten, die bisher in der Sitzung übermittelt wurden.
- Wenn die Sitzung transaktionsbasiert ist, können unbestätigte Nachrichten vom Anwendungsserver mit dem Aufruf von `Session.rollback()` zurückgesetzt werden.

Warteschlangenbrowser

Eine Anwendung verwendet einen Warteschlangenbrowser, um die Nachrichten in einer Warteschlange zu durchsuchen, ohne sie zu entfernen.

Um einen Warteschlangenbrowser zu erstellen, ruft eine Anwendung die Methode 'Create Queue Browser' eines ISession-Objekts auf und gibt als Parameter ein Destination-Objekt an, das die zu durchsuchende Warteschlange angibt. Die Anwendung kann einen Warteschlangenbrowser mit oder ohne Nachrichtenselektor erstellen.

Nach dem Erstellen eines Warteschlangenbrowsers kann die Anwendung die Methode 'GetEnumerator' des IQueueBrowser-Objekts aufrufen, um eine Liste der Nachrichten in der Warteschlange abzurufen. Die Methode gibt einen Aufzählungsausdruck zurück, der eine Liste der Nachrichtenobjekte ('Message') enthält. Die Reihenfolge der Message-Objekte in der Liste ist identisch mit der Reihenfolge, in der die Nachrichten aus der Warteschlange abgerufen würden. Die Anwendung kann dann den Aufzählungsausdruck dazu verwenden, alle Nachrichten einzeln nacheinander zu durchsuchen.

Der Aufzählungsausdruck wird dynamisch aktualisiert, wenn Nachrichten in die Warteschlange eingereiht bzw. daraus entfernt werden. Jedes Mal, wenn die Anwendung 'IEnumerator.MoveNext()' aufruft, um die nächste Nachricht in der Warteschlange zu durchsuchen, gibt die Nachricht den aktuellen Inhalt der Warteschlange wieder.

Eine Anwendung kann die GetEnumerator-Methode für einen bestimmten Warteschlangenbrowser mehrmals aufrufen. Bei jedem Aufruf wird ein neuer Aufzählungsausdruck zurückgegeben. Daher kann eine Anwendung mehrere Aufzählungsausdrücke zum Durchsuchen einer Warteschlange verwenden und dadurch mehrere Positionen in der Warteschlange kennzeichnen.

Mithilfe eines Warteschlangenbrowsers kann eine Anwendung nach einer bestimmten Nachricht suchen, die aus einer Warteschlange entfernt werden soll, und dann einen Nachrichtenkonsumenten mit einem Nachrichtenselektor dazu verwenden, die Nachricht zu löschen. Der Nachrichtenselektor kann die Nachricht mithilfe des Werts im Headerfeld 'JMSMessageID' auswählen. Weitere Informationen zu diesem anderen JMS-Nachrichtenheaderfeldern finden Sie im Abschnitt „Headerfelder in Eine XMS -Nachricht” auf Seite 76.

Anforderer

Eine Anwendung verwendet einen Anforderer, um eine Anforderungsnachricht zu senden und dann auf eine Antwort zu warten und diese zu empfangen.

Viele Messaging-Anwendungen basieren auf Algorithmen, die eine Anforderungsnachricht senden und dann auf eine Antwort warten. Zur Unterstützung der Entwicklung dieser Art von Anwendung stellt XMS die Klasse 'Requestor' bereit.

Um einen Anforderer zu erstellen, ruft eine Anwendung den Konstruktor 'Create Requestor' oder die Klasse 'Requestor' auf und gibt als Parameter ein Session-Objekt an sowie ein Destination-Objekt, das angibt, wohin die Anforderungsnachricht gesendet werden soll. Die Sitzung darf keine Sitzung mit Transaktionsunterstützung sein und nicht den Bestätigungsmodus XMSC_CLIENT_ACKNOWLEDGE aufweisen. Der Konstruktor erstellt automatisch eine temporäre Warteschlange oder ein temporäres Thema, an die oder das die Antwortnachrichten gesendet werden sollen.

Nach dem Erstellen eines Anforderers kann die Anwendung die Request-Methode des Requestor-Objekts aufrufen, um eine Anforderungsnachricht zu senden und dann eine Antwort von der Anwendung, die die Anforderungsnachricht empfangen hat, zu erwarten und zu empfangen. Der Aufruf wartet, bis die Antwort empfangen wurde oder bis die Sitzung endet, je nachdem, welches Ereignis zuerst eintritt. Für den Anforderer ist nur eine Antwort für jede Anforderungsnachricht erforderlich.

Wenn die Anwendung den Anforderer schließt, wird die temporäre Warteschlange bzw. das temporäre Thema gelöscht. Die zugeordnete Sitzung wird jedoch nicht geschlossen.

Objektlöschung

Wenn eine Anwendung ein von ihr erstelltes XMS-Objekt löscht, gibt XMS die internen Ressourcen frei, die dem Objekt zugeordnet waren.

Wenn eine Anwendung ein Eine XMS -Objekt erstellt, ordnet XMS dem Objekt Speicher und weitere interne Ressourcen zu. XMS behält diese internen Ressourcen so lange bei, bis die Anwendung das Objekt explizit löscht, indem sie die Methode zum Schließen oder Löschen des Objekts aufruft. Erst dann gibt XMS die internen Ressourcen frei. Wenn eine Anwendung versucht, ein Objekt zu löschen, das bereits gelöscht ist, wird der entsprechende Aufruf ignoriert.

Wenn ein Connection- oder Session-Objekt durch eine Anwendung gelöscht wird, löscht XMS bestimmte zugeordnete Objekte automatisch und gibt deren interne Ressourcen frei. Diese Objekte wurden vom jeweiligen Connection- oder Session-Objekt erstellt und haben keine von diesem Objekt unabhängige Funktion. Diese Objekte sind in Tabelle 4 auf Seite 43 aufgeführt.

Anmerkung: Wenn eine Anwendung eine Verbindung mit abhängigen Sitzungen schließt, werden auch alle von diesen Sitzungen abhängenden Objekte gelöscht. Nur Connection- oder Session-Objekte können abhängige Objekte haben.

Tabelle 4. Automatisch gelöschte Objekte

Gelöschtes Objekt	Methode	Abhängige Objekte, die automatisch gelöscht werden
Verbindung	Close Connection (Verbindung schließen)	ConnectionMetaData- und Session-Objekte
Sitzungen	Close Session (Sitzung schließen)	MessageConsumer-, MessageProducer-, QueueBrowser- und Requestor-Objekte

Verwaltete IBM WebSphere MQ-XA-Transaktionen über XMS

Verwaltete IBM WebSphere MQXA-Transaktionen können über XMS verwendet werden.

Um XMS-Transaktionen über XMS verwenden zu können, muss eine Sitzung mit Transaktionsunterstützung erstellt werden. Wenn eine XA-Transaktion im Gebrauch ist, erfolgt die Transaktionssteuerung über globale Transaktionen von Distributed Transaction Coordinator (DTC) und nicht über XMS-Sitzungen. Werden XA-Transaktionen verwendet, können `Session.commit` und `Session.rollback` nicht in der XMS-Sitzung ausgegeben werden. Verwenden Sie stattdessen die DTC-Methoden `Transscope.Commit` und `Transscope.Rollback`, um die Transaktionen festzuschreiben oder rückgängig zu machen. Wenn eine Sitzung für XA-Transaktionen verwendet wird, müssen der Nachrichtenproduzent und -konsument, die mithilfe der Sitzung erstellt werden, Teil der XA-Transaktion sein. Ihre Verwendung außerhalb des Geltungsbereichs der XA-Transaktion ist nicht möglich. Das heißt, sie können für Operationen wie `Producer.send` oder `Consumer.receive` außerhalb der XA-Transaktion nicht verwendet werden.

Es wird ein `IllegalStateException`-Ausnahmeobjekt ausgelöst, wenn

- Eine Sitzung mit XA-Transaktionsunterstützung wird für `Session.commit` oder `Session.rollback` verwendet.
- Produzenten- oder Konsumentenobjekte, die einmal in einer XA-transaktionsbasierten Sitzung verwendet wurden, werden danach außerhalb des Geltungsbereichs der XA-Transaktion verwendet.

XA-Transaktionen werden für asynchrone Nachrichtenkonsumenten nicht unterstützt.

Anmerkung:

1. Das `Producer`-, `Consumer`-, `Session`- oder `Connection`-Objekt wird möglicherweise vor der Festschreibung der XA-Transaktion geschlossen. In diesem Fall werden die Nachrichten in der Transaktion durch ein Rollback rückgängig gemacht. Auch bei einer Unterbrechung der Verbindung vor der Festschreibung der XA-Transaktion werden alle Nachrichten in der Transaktion rückgängig gemacht. Bei einem `Producer` -Objekt bedeutet ein Rollback, dass die Nachrichten nicht in die Warteschlange eingereiht werden. Bei einem `Consumer` -Objekt bedeutet ein Rollback, dass die Nachrichten in der Warteschlange verbleiben.
2. Wenn ein `Producer`-Objekt eine Nachricht mit der Option `TimeToLive` in `TransactionScope` einreiht und ein `commit`-Befehl ausgegeben wird, nachdem das Zeitlimit abgelaufen ist, kann die

Nachricht ablaufen, bevor der `commit`-Befehl ausgegeben wird. In diesem wird die Nachricht nicht für die `Consumer`-Objekte verfügbar gemacht.

3. `Session`-Objekte werden nicht über mehrere Threads hinweg unterstützt. Das bedeutet, dass Transaktionen mit `Session`-Objekten, die von mehreren Threads gemeinsam genutzt werden, nicht unterstützt werden.

Primitive XMS-Datentypen

XMS stellt Datentypen bereit, die den acht primitiven Java-Datentypen entsprechen: 'byte', 'short', 'int', 'long', 'float', 'double', 'char' und 'boolean'. Dies ermöglicht den Austausch von Nachrichten zwischen XMS und JMS, ohne dass Daten verloren gehen oder beschädigt werden.

In Tabelle 5 auf Seite 44 sind die den Java-Datentypen entsprechenden primitiven XMS-Datentypen einschließlich Größe, Mindest- und Höchstwert aufgelistet.

Tabelle 5. XMS-Datentypen und deren Java-Entsprechungen

XMS-Datentyp	Kompatibler Java-Datentyp	Größe	Mindestwert:	Höchstwert
System.Boolean	boolean	32 Bit	false	true
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	float	32 Bit	-3.402823E-38 (mit 7-stelliger Genauigkeit)	3.402823E+38 (mit 7-stelliger Genauigkeit)
System.Double	double	64 Bit	-1.79769313486231E-308 (mit 15-stelliger Genauigkeit)	1.79769313486231E+308 (mit 15-stelliger Genauigkeit)

Zugehörige Konzepte

Attribute und Eigenschaften von Objekten
Ein XMS-Objekt kann Attribute und Eigenschaften besitzen, bei denen es sich um Merkmale des Objekts handelt, die auf unterschiedliche Weise implementiert werden.

Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp
Wenn eine Anwendung den Wert einer Eigenschaft abruft, kann der Wert von XMS in einen anderen Datentyp konvertiert werden. Es gibt viele Regeln, mit denen gesteuert wird, welche Konvertierungen unterstützt werden und wie XMS die Konvertierungen ausführt.

Zugehörige Verweise

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp

Wenn eine Anwendung den Wert einer Eigenschaft abruft, kann der Wert von XMS in einen anderen Datentyp konvertiert werden. Es gibt viele Regeln, mit denen gesteuert wird, welche Konvertierungen unterstützt werden und wie XMS die Konvertierungen ausführt.

Eine Eigenschaft eines Objekts hat einen Namen und einen Wert. Dem Wert ist ein Datentyp zugeordnet, wobei der Wert einer Eigenschaft auch als *Eigenschaftstyp* bezeichnet wird.

Eine Anwendung verwendet die Methoden der Klasse 'PropertyContext', um Eigenschaften von Objekten abzurufen und festzulegen. Um den Wert einer Eigenschaft abzurufen, ruft eine Anwendung die für den Eigenschaftstyp geeignete Methode auf. Um beispielsweise den Wert einer ganzzahligen Eigenschaft (Integer) abzurufen, ruft eine Anwendung normalerweise die Methode 'GetIntProperty' auf.

Wenn eine Anwendung den Wert einer Eigenschaft abruft, kann der Wert jedoch von XMS in einen anderen Datentyp konvertiert werden. Wenn beispielsweise der Wert einer ganzzahligen Eigenschaft (Integer) abgerufen werden soll, kann eine Anwendung die Methode 'GetStringProperty' aufrufen, sodass der Eigenschaftswert als Zeichenfolge zurückgegeben wird. Die von XMS unterstützten Konvertierungen sind in Tabelle 6 auf Seite 45 dargestellt.

Tabelle 6. Unterstützte Konvertierungen von einem Eigenschaftstyp in andere Datentypen

Merkmaltyp	**Unterstützte Zieldatentypen**
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 array	System.String
System.Int16	System.String, System.Int32, System.Int64

Folgende allgemeine Regeln steuern die unterstützten Konvertierungen:

- Numerische Eigenschaftswerte können von einem Datentyp in einen anderen Datentyp konvertiert werden, vorausgesetzt, bei der Konvertierung gehen keine Daten verloren. So kann beispielsweise ein Eigenschaftswert mit dem Datentyp 'System.Int32' in einen Wert mit dem Datentyp 'System.Int64' konvertiert werden, es ist jedoch nicht möglich, ihn in einen Wert mit dem Datentyp 'System.Int16' zu konvertieren.
- Ein Eigenschaftswert eines Datentyps kann in eine Zeichenfolge umgewandelt werden.
- Ein Zeichenfolgen-Eigenschaftswert kann in jeden anderen Datentyp konvertiert werden, vorausgesetzt, die Zeichenfolge wird für die Konvertierung korrekt formatiert. Wenn eine Anwendung versucht, eine Zeichenfolge zu konvertieren, die nicht das richtige Format aufweist, gibt XMS möglicherweise Fehler zurück.
- Wenn eine Anwendung eine nicht unterstützte Konvertierung versucht, gibt XMS möglicherweise einen Fehler zurück.

Folgende Regeln gelten für die Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen:

- Bei der Konvertierung eines booleschen Eigenschaftswerts in eine Zeichenfolge wird der Wert 'true' in die Zeichenfolge "true" und der Wert 'false' in die Zeichenfolge "false" konvertiert.
- Beim Konvertieren eines booleschen Eigenschaftswerts in einen numerischen Datentyp einschließlich 'System.SByte' wird der Wert 'true' in den Wert '1' und der Wert 'false' in den Wert '0' konvertiert.
- Beim Konvertieren eines Zeichenfolgeeigenschaftswerts in einen booleschen Wert wird die Zeichenfolge "true" (ohne Beachtung der Groß-/Kleinschreibung) oder der Wert "1" in den booleschen Wert 'true' und die Zeichenfolge "false" (ohne Beachtung der Groß-/Kleinschreibung) oder der Wert "0" in den booleschen Wert 'false' konvertiert. Alle anderen Zeichenfolgen können nicht konvertiert werden.
- Beim Konvertieren eines Zeichenfolgeeigenschaftswerts in einen Wert des Datentyps 'System.Int32', 'System.Int64', 'System.SByte' oder 'System.Int16' muss die Zeichenfolge das folgende Format haben:

  [*blanks*][*sign*] *digits*

  Die Zeichenfolgenkomponenten sind wie folgt definiert:

  blanks
  Optionale führende Leerzeichen.

  Vorzeichen
  Ein optionales Pluszeichen (+) oder Minuszeichen (-).

  Ziffern
  Eine zusammenhängende Folge von Ziffern (0-9). Es muss mindestens ein Ziffernzeichen vorhanden sein.

  Auf die Ziffernfolge können weitere Zeichen folgen, bei denen es sich nicht um Ziffern handelt, doch wird die Konvertierung beendet, sowie das erste dieser Zeichen erreicht wird. Es wird vorausgesetzt, dass die Zeichenfolge eine Ganzzahl im Dezimalformat darstellt.

  Wenn die Zeichenfolge nicht korrekt formatiert ist, gibt XMS möglicherweise einen Fehler zurück.
- Beim Konvertieren eines Zeichenfolgeeigenschaftswerts in einen Wert des Datentyps 'System.Double' oder 'System.Float' muss die Zeichenfolge das folgende Format haben:

  [*blanks*][*sign*][*digits*][*point*[*d_digits*]][*e_char*[*e_sign*]*e_digits*]

  Die Zeichenfolgenkomponenten sind wie folgt definiert:

  blanks
  Optionale führende Leerzeichen.

  Vorzeichen
  Optionales Pluszeichen (+) oder Minuszeichen (-).

  Ziffern
  Eine zusammenhängende Folge von Ziffern (0-9). In einer der Komponenten *digits* oder *d_digits* muss mindestens ein Ziffernzeichen vorhanden sein.

  point
  (Optional) Dezimalzeichen (.).

  d_digits
  Eine zusammenhängende Folge von Ziffern (0-9). In einer der Komponenten *digits* oder *d_digits* muss mindestens ein Ziffernzeichen vorhanden sein.

  e_Zeich
  Ein Exponentenzeichen, entweder *E* oder *e*.

  e_Vorz
  Optionales Pluszeichen (+) oder Minuszeichen (-) für den Exponenten.

  e_Ziffern
  Eine zusammenhängende Folge von Ziffern (0-9) für den Exponenten. Mindestens eine Ziffer muss vorhanden sein, wenn die Zeichenfolge ein Exponentenzeichen enthält.

  Auf die Ziffernfolge bzw. die optionalen, einen Exponenten darstellenden Zeichen, können weitere Zeichen folgen, bei denen es sich nicht um Ziffern handelt, doch wird die Konvertierung beendet,

sowie das erste dieser Zeichen erreicht wird. Es wird davon ausgegangen, dass die Zeichenfolge eine Gleitkommazahl mit einem Exponenten der Potenz 10 darstellt.

Wenn die Zeichenfolge nicht korrekt formatiert ist, gibt XMS möglicherweise einen Fehler zurück.

- Beim Konvertieren eines numerischen Eigenschaftswerts einschließlich Werte des Datentyps 'System.SByte' in eine Zeichenfolge wird der Wert in die Zeichenfolgedarstellung des Werts als Dezimalzahl konvertiert, nicht in die Zeichenfolge, die den ASCII-Zeichen für diesen Wert enthält. Die Ganzzahl 65 wird beispielsweise in die Zeichenfolge "65" konvertiert, nicht in die Zeichenfolge "A".
- Beim Konvertieren eines Byte-Array-Eigenschaftswerts in eine Zeichenfolge wird jedes Byte in die zwei Hexadezimalzeichen konvertiert, die das Byte darstellen. Beispiel: Das Byte-Array {0xF1, 0x12, 0x00, 0xFF} wird in die Zeichenfolge "F11200FF" konvertiert.

Konvertierungen von einem Eigenschaftstyp in andere Datentypen werden von den Methoden sowohl der Property- als auch der PropertyContext-Klassen unterstützt.

Zugehörige Konzepte

Attribute und Eigenschaften von Objekten
Ein XMS-Objekt kann Attribute und Eigenschaften besitzen, bei denen es sich um Merkmale des Objekts handelt, die auf unterschiedliche Weise implementiert werden.

Primitive XMS-Datentypen
XMS stellt Datentypen bereit, die den acht primitiven Java-Datentypen entsprechen: 'byte', 'short', 'int', 'long', 'float', 'double', 'char' und 'boolean'. Dies ermöglicht den Austausch von Nachrichten zwischen XMS und JMS, ohne dass Daten verloren gehen oder beschädigt werden.

Zugehörige Verweise

Zuordnungsnachrichten
Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Datenstromnachrichten
Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Iteratoren

Ein Iterator enthält eine Liste mit Objekten sowie einen Cursor, der die aktuelle Position in der Liste kennzeichnet. Das Konzept eines Iterators, wie er in Message Service Client for C/C++ verfügbar ist, wird über die IEnumerator-Schnittstelle in Message Service Client for .NET implementiert.

Beim Erstellen eines Iterators wird der Cursor vor dem ersten Objekt positioniert. Eine Anwendung verwendet einen Iterator, um alle Objekte einzeln der Reihe nach abzurufen.

Die Iterator-Klasse von Message Service Client for C/C++ entspricht funktional der Enumerator-Klasse in Java.XMS .NET ist Java ähnlich und verwendet eine IEnumerator-Schnittstelle.

Eine Anwendung kann mit einem IEnumerator-Element folgende Tasks ausführen:

- Eigenschaften einer Nachricht abrufen
- Name/Wert-Paare im Hauptteil einer Zuordnungsnachricht abrufen
- Nachrichten in einer Warteschlange durchsuchen
- Namen der JMS-definierten Nachrichteneigenschaften abrufen, die von einer Verbindung unterstützt werden

IDs des codierten Zeichensatzes

In XMS .NET werden alle Zeichenfolgen mit der nativen .NET-Zeichenfolge übergeben. Da diese Zeichenfolge eine feste Codierung hat, sind zur Interpretation keine weiteren Informationen erforderlich. Daher ist die Eigenschaft XMSC_CLIENT_CCSID für XMS .NET-Anwendungen nicht erforderlich.

XMS-Fehlercodes und -Ausnahmecodes

XMS umfasst zahlreiche Fehlercodes, um auf Fehler hinzuweisen. Die Fehlercodes werden in dieser Dokumentation nicht explizit aufgeführt, weil sie je nach Release unterschiedlich sein können. Nur die XMS-Ausnahmecodes (im Format 'XMS_X_...') sind dokumentiert, weil in allen XMS-Releases identisch sind.

Eigene Anwendung erstellen

Sie können Ihre eigenen Anwendungen auf die gleiche Weise erstellen, wie Sie die Beispielanwendungen erstellen.

Erstellen Sie Ihre .NET -Anwendung gemäß der Beschreibung im Artikel „.NET-Beispielanwendungen erstellen” auf Seite 22 Thema, in dem auch die Voraussetzungen für die Erstellung eigener .NET -Anwendungen aufgeführt sind. Weitere Anleitungen zum Erstellen Ihrer eigenen Anwendungen finden Sie in den Makefiles, die für die jeweilige Beispielanwendung bereitgestellt werden.

Tipp: Als Unterstützung bei der Problemdiagnose im Falle eines Fehlers kann es hilfreich sein, die Anwendungen mit eingegebenen Symbolen zu kompilieren.

Zugehörige Verweise

.NET-Schnittstellen
In diesem Thema werden die Schnittstellen der .NET-Klasse und die zugehörigen Eigenschaften und Methoden dokumentiert.

Eigenschaften von XMS-Objekten
In diesem AbschnittKapitel werden die mit XMS definierten Objekteigenschaften dokumentiert.

Automatische IBM WebSphere MQ-Clientverbindungswiederholung über XMS

Konfigurieren Sie Ihren XMS -Client so, dass die Verbindung nach einem Netz-, Warteschlangenmanager- oder Serverausfall automatisch wiederhergestellt wird, wenn IBM WebSphere MQ V7.1 Client oder höher als Nachrichtenprovider verwendet wird.

Mit den Eigenschaften `WMQ_CONNECTION_NAME_LIST` und `WMQ_CLIENT_RECONNECT_OPTIONS` der Klasse `MQConnectionFactory` können Sie die automatische Wiederherstellung einer Clientverbindung konfigurieren. Bei Aktivierung der automatischen Clientverbindungswiederholung wird eine Clientverbindung nach einem Verbindungsfehler oder optional nach dem Stoppen des Warteschlangenmanagers automatisch wiederhergestellt. Einige Clientanwendungen sind aufgrund ihres Designs für die automatische Verbindungswiederherstellung ungeeignet.

Automatisch wiederverbindungsfähige Clientverbindungen werden unmittelbar nach dem Einrichten der Verbindung wiederverbindungsfähig.

Anmerkung: Die Eigenschaften `Client Reconnect Options` (Clientwiederverbindungsoptionen), `Client Reconnect Timeout` (Zeitlimit für Clientverbindungswiederholung) und `Connection Name-list` (Verbindungsnamensliste) können auch über eine CCDT (Definitionstabelle für Clientkanal) oder durch Aktivieren der Clientverbindungswiederholung in der Datei `mqclient.ini` festgelegt werden.

Anmerkung: Wenn sowohl im Objekt `ConnectionFactory` als auch in der CCDT Verbindungswiederholungseigenschaften festgelegt sind, gilt folgende Vorrangregel. Wenn die Eigenschaft der Verbindungsnamensliste im Objekt `ConnectionFactory` auf den Standardwert festgelegt ist, hat die CCDT Vorrang. Wenn die Verbindungsnamensliste hingegen nicht auf den Standardwert festgelegt ist, haben die im Objekt `ConnectionFactory` festgelegten Eigenschaftswerte Vorrang. Der Standardwert der Verbindungsnamensliste lautet `localhost(1414)`.

XMS .NET-Anwendungen schreiben

Dieser AbschnittKapitel enthält Informationen, die Ihnen beim Schreiben von XMS.NET-Anwendungen helfen.

Dieser AbschnittKapitel enthält Informationen, die sich speziell auf das Schreiben von XMS .NET-Anwendungen beziehen. Allgemeine Informationen zum Schreiben von XMS-Anwendungen finden Sie im Abschnitt „XMS-Anwendungen schreiben” auf Seite 23.

Der AbschnittKapitel enthält folgende ThemenAbschnitte:

Zugehörige Verweise

.NET-Schnittstellen
In diesem Thema werden die Schnittstellen der .NET-Klasse und die zugehörigen Eigenschaften und Methoden dokumentiert.

Datentypen für .NET

XMS .NET unterstützt die Datentypen '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' und 'System.Object'. Datentypen für XMS .NET unterscheiden sich von Datentypen für XMS C++. Mithilfe der Informationen in diesem AbschnittKapitel können Sie die entsprechenden Datentypen identifizieren.

In der folgenden Tabelle werden die entsprechenden XMS .NET- und XMS C++-Datentypen aufgeführt und kurz beschrieben.

Tabelle 7. Datentypen für XMS .NET und XMS C++

XMS .NET-Datentyp	XMS C++-Datentyp	Beschreibung
System.SByte	xmsSBYTE xmsINT8	8-Bit-Wert mit Vorzeichen
System.Byte	xmsBYTE xmsUINT8	8-Bit-Wert ohne Vorzeichen
System.Int16	xmsINT16 xmsSHORT	16-Bit-Wert mit Vorzeichen
System.UInt16	xmsUINT16 xmsUSHORT	16-Bit-Wert ohne Vorzeichen
System.Int32	xmsINT32 xmsINT	32-Bit-Wert mit Vorzeichen
System.UInt32	xmsUINT32 xmsUINT	32-Bit-Wert ohne Vorzeichen

Tabelle 7. Datentypen für XMS .NET und XMS C++ (Forts.)		
XMS .NET-Datentyp	**XMS C++-Datentyp**	**Beschreibung**
System.Int64	xmsLONG xmsINT64	64-Bit-Wert mit Vorzeichen
System.UInt64	xmsULONG xmsUINT64	64-Bit-Wert ohne Vorzeichen
System.Char	xmsCHAR16	16-Bit-Zeichen ohne Vorzeichen (Unicode für .NET)
System.Single	xmsFLOAT	32-Bit-IEEE-Gleitkommazahl
System.Double	xmsDOUBLE	64-Bit-IEEE-Gleitkommazahl
System.Boolean	xmsBOOL	True/False-Wert (Wahr/Falsch)
Nicht zutreffend	xmsCHAR	8-Bit-Wert mit oder ohne Vorzeichen (mit/ohne Vorzeichen hängt von der Plattform ab)
System.Decimal	Nicht zutreffend	96-Bit-Ganzzahl multipliziert mit 10^0 bis 10^{28}
System.Object	Nicht zutreffend	Basis aller Typen
System.String	Nicht zutreffend	Zeichenfolgedatentyp

Verwaltete und nicht verwaltete Operationen in .NET

Verwalteter Code wird ausschließlich in der Umgebung der .NET Common Language Runtime ausgeführt und ist vollständig von den in dieser Laufzeit bereitgestellten Services abhängig. Eine Anwendung wird als 'nicht verwaltet' klassifiziert, wenn ein Teil der Anwendung oder Services, die von der Anwendung aufgerufen werden, außerhalb der Umgebung der .NET Common Language Runtime ausgeführt werden.

Bestimmte erweiterte Funktionen können in der verwalteten .NET-Umgebung aktuell nicht unterstützt werden.

Wenn Ihre Anwendung Funktionen benötigt, die in der vollständig verwalteten Umgebung aktuell nicht unterstützt werden, können Sie Ihre Anwendung für die Verwendung der nicht verwalteten Umgebung anpassen, ohne wesentliche Änderungen an der Anwendung vornehmen zu müssen. Wenn Sie sich für diese Vorgehensweise entscheiden, müssen Sie jedoch beachten, dass der XMS-Stack nicht verwalteten Code verwendet.

Verbindungen zu einem IBM WebSphere MQ-Queue Manager

Verwaltete Verbindungen zu WMQ_CM_CLIENT unterstützen keine SSL-Verbindungen, keine TCP-Kommunikation und keine Kanalkomprimierung. Diese Verbindungen werden jedoch möglicherweise von eine nicht verwalteten Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt. Weitere Informationen finden Sie im Abschnitt .NET-Anwendungen entwickeln.

Wenn Sie eine Verbindungsfactory basierend auf einem verwalteten Objekt in einer nicht verwalteten Umgebung erstellen, müssen Sie den Wert für den Verbindungsmodus manuell in 'XMSC_WMQ_CM_CLIENT_UNMANAGED' ändern.

Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine

Verbindungen zur Messaging-Engine eines WebSphere Service Integration Bus, für die das SSL-Protokoll (einschließlich HTTPS) verwendet werden muss, werden aktuell nicht als verwalteter Code unterstützt.

Zugehörige Verweise
XMSC_WMQ_CONNECTION_MODE

Ziele in .NET

In .NET werden Ziele entsprechend dem jeweiligen Protokolltyp erstellt und können dann auch nur mit dem Protokolltyp verwendet werden, mit dem sie erstellt wurden.

Es werden zwei Funktionen zum Erstellen von Zielen bereitgestellt, eine für Themen und eine für Warteschlangen:

- `IDestination CreateTopic(String topic);`
- `IDestination CreateQueue(String queue);`

Diese Funktionen sind in der API in den folgenden zwei Objekten verfügbar:

- ISession
- XMSFactoryFactory

Bei beiden Methoden können Zeichenfolgen im URI-Stil verwendet werden, die auch Parameter enthalten können und folgendem Format entsprechen:

```
"topic://some/topic/name?priority=5"
```

Alternativ dazu kann in diesen Methoden auch ein reiner Zielname angegeben werden, d. h. ein Name ohne den Präfix 'topic://' oder 'queue://' und ohne Parameter.

Beispiel für eine Zeichenfolge im URI-Stil:

```
CreateTopic("topic://some/topic/name");
```

Mit dieser Zeichenfolge im URI-Stil wird dasselbe Ergebnis erzielt wie mit dem folgenden Zielnamen. Beispiel für einen reinen Zielnamen:

```
CreateTopic("some/topic/name");
```

Wie bei WebSphere Service Integration Bus-JMS können Themen auch in einer Kurzform angegeben werden, die sowohl den *topicname* (Themaname) als auch den *topicspace* (Themabereich) einschließt, aber keine Parameter enthalten kann:

```
CreateTopic("topicspace:topicname");
```

Eigenschaften in .NET

Eine .NET-Anwendung verwendet die Methoden in der Schnittstelle 'PropertyContext', um Eigenschaften von Objekten abzurufen und festzulegen.

Die Schnittstelle PropertyContext enthält Methoden zum Abrufen und Festlegen von Eigenschaften. Diese Methoden werden direkt oder indirekt von folgenden Klassen übernommen:

- BytesMessage
- Verbindung
- ConnectionFactory
- ConnectionMetaData
- Zieladresse
- MapMessage
- Nachricht
- MessageConsumer
- MessageProducer

- ObjectMessage
- QueueBrowser
- Sitzungen
- StreamMessage
- TextMessage

Wenn eine Anwendung den Wert einer Eigenschaft festlegt, ersetzt der neue Wert immer den vorherigen Wert, der der Eigenschaft zugewiesen war.

Weitere Informationen zu XMS-Eigenschaften finden Sie im Abschnitt „Eigenschaften von XMS-Objekten” auf Seite 189.

Um die Verwendung zu vereinfachen, sind XMS-Eigenschaftsnamen und -werte in .NET als öffentliche Konstanten in einem 'XMSC' genannten Konstrukt vordefiniert. Die Namen dieser Konstanten haben das Format XMSC.*< Konstante>*. Beispiel: XMSC.USERID (eine Eigenschaftsnamenskonstante) und XMSC.DELIVERY_AS_APP (eine Wertkonstante)

Außerdem haben Sie mithilfe des Konstrukts 'IBM.XMS.MQC' Zugriff auf IBM WebSphere MQ-Konstanten. Bei IBM.XMS -Namensbereich wurde bereits importiert. Sie können auf die Werte für diese Eigenschaften im Format MQC.*< Konstante>*zugreifen. Beispiel: MQC.MQRO_COA_WITH_FULL_DATA

Wenn Sie darüber hinaus eine Hybridanwendung nutzen, die sowohl XMS .NET- als auch IBM WebSphere MQ-Klassen für .NET verwendet und die beide Namensbereiche, also 'IBM.XMS' und 'IBM.WMQ' importiert, dann müssen Sie den Namensbereich des MQC-Konstrukts vollständig qualifiziert angeben, um sicherzustellen, dass jedes Element eindeutig ist.

Einige erweiterte Funktionen werden in der verwalteten .NET-Umgebung aktuell nicht unterstützt. Weitere Informationen hierzu finden Sie im Abschnitt „Verwaltete und nicht verwaltete Operationen in .NET” auf Seite 50.

Behandlung nicht vorhandener Eigenschaften in .NET

Die Handhabung nicht vorhandener Eigenschaften in XMS .NET ist im Wesentlichen mit der JMS-Spezifikation sowie mit den C- und C++-Implementierungen für XMS konsistent.

In JMS kann der Zugriff auf eine nicht vorhandene Eigenschaft zu einer Java-Systemausnahme führen, wenn eine Methode versucht, den nicht vorhandenen Wert (Null-Wert) in den erforderlichen Datentyp umzuwandeln. Wenn eine Eigenschaft nicht vorhanden ist, treten folgende Ausnahmebedingungen auf:

- 'getStringProperty' und 'getObjectProperty' geben 'null' zurück
- 'getBooleanProperty' gibt 'false' zurück, weil 'Boolean.valueOf(null)' den Wert 'false' zurückgibt
- 'getIntProperty' usw. lösen die Ausnahme 'java.lang.NumberFormatException' aus, weil 'Integer.valueOf(null)' diese Ausnahme auslöst

Wenn eine Eigenschaft in XMS .NET nicht vorhanden ist, treten folgende Ausnahmebedingungen auf:

- 'GetStringProperty' und 'GetObjectProperty' (sowie 'GetBytesProperty') geben 'null' zurück (dieses Verhalten ist identisch mit Java)
- 'GetBooleanProperty' löst die Ausnahme 'System.NullReferenceException' aus
- 'GetIntProperty' usw. lösen die Ausnahme 'System.NullReferenceException' aus

Diese Implemntierung unterscheidet sich zwar von Java, ist jedoch im Wesentlichen mit der JMS-Spezifikation sowie mit den C- und C++-Schnittstellen für XMS konsitent. Wie die Java-Implementierung gibt XMS .NET alle Ausnahmen von Aufruf System.Convert an das aufrufende Programm weiter. Anders als Java löst XMS jedoch explizit die Ausnahme 'NullReferenceException' aus, anstatt lediglich das native .NET-Framework zu verwenden und den Wert 'null' an die Konvertierungsroutinen zu übergeben. Wenn Ihre Anwendung eine Eigenschaft auf eine Zeichenfolge wie beispielsweise "abc" setzt und die Methode 'GetIntProperty' aufruft, wird die von 'Convert.ToInt32("abc")' ausgelöste Ausnahme 'System.FormatException' an das aufrufende Programm weitergegeben, wobei dieses Verhalten mit Java konsistent ist. Die Ausnahme 'MessageFormatException' wird nur ausgelöst, wenn die für 'setProperty' und 'getProperty' verwendeten Datentypen inkompatibel sind. Auch dieses Verhalten ist mit Java konsistent.

Fehlerbehandlung in .NET

XMS .NET-Ausnahmen werden alle von System.Exception abgeleitet. XMS-Methodenaufrufe können spezielle XMS-Ausnahmebedingungen auslösen, wie z. B. 'MessageFormatException', allgemeine Ausnahmen des Typs 'XMSException' oder Systemausnahmen wie 'NullReferenceException'.

Sie können Anwendungen schreiben, mit denen diese Fehler abgefangen werden, entweder in spezifischen Catch-Blocks oder in allgemeinen `System.Exception`-Catch-Blocks, je nach den Anforderungen der Anwendungen.

Listener für Nachrichten und Ausnahmebedingungen in .NET

Eine .NET-Anwendung verwendet einen Nachrichtenlistener, um Nachrichten asynchron zu empfangen, und einen Ausnahmelistener, um asynchron über Probleme mit der Verbindung benachrichtigt zu werden.

Die Nachrichten- und Ausnahmelistener für .NET haben dieselbe Funktionalität wie die Listener für C++. Es gibt jedoch einige geringfügige Implementierungsunterschiede.

Nachrichtenlistener in .NET

Wenn Nachrichten asynchron empfangen werden sollen, müssen Sie folgende Schritte ausführen:

1. Definieren Sie eine Methode, die mit der Signatur des Nachrichtenlistenerdelegierten übereinstimmt. Die von Ihnen definierte Methode kann statisch oder eine Instanzdefinitionsmethode sein und in jeder beliebigen, zugänglichen Klasse definiert werden. Die Delegiertensignatur lautet wie folgt:

   ```
   public delegate void MessageListener(IMessage msg);
   ```

   Daher können Sie die Methode wie folgt definieren:

   ```
   void SomeMethodName(IMessage msg);
   ```

2. Instanziieren Sie diese Methode mit einer ähnlichen Anweisung wie der folgenden als ein Delegat:

   ```
   MessageListener OnMsgMethod = new MessageListener(SomeMethodName)
   ```

3. Registrieren Sie den Delegierten bei mindestens einem Konsumenten, indem Sie ihn in der Eigenschaft 'MessageListener' des Konsumenten angeben:

   ```
   consumer.MessageListener = OnMsgMethod;
   ```

   Sie können den Delegierten entfernen, indem Sie die Eigenschaft 'MessageListener' wieder auf den Wert 'null' setzen:

   ```
   consumer.MessageListener = null;
   ```

Listener für Ausnahmebedingungen in .NET

Der Ausnahmelistener funktioniert in ähnlicher Weise wie der Nachrichtenlistener, hat jedoch eine andere Delegiertendefinition und wird nicht dem Nachrichtenkonsumenten, sondern der Verbindung zugeordnet. Dies ist mit C++ identisch.

1. Definieren Sie die Methode. Die Delegiertensignatur lautet wie folgt:

   ```
   public delegate void ExceptionListener(Exception ex);
   ```

   Daher können Sie die Methode wie folgt definieren:

   ```
   void SomeMethodName(Exception ex);
   ```

2. Instanziieren Sie diese Methode mit einer ähnlichen Anweisung wie der folgenden als ein Delegat:

   ```
   ExceptionListener OnExMethod = new ExceptionListener(SomeMethodName)
   ```

3. Registrieren Sie den Delegierten bei der Verbindung, indem Sie deren Eigenschaft 'ExceptionListener' wie folgt festlegen:

   ```
   connection.ExceptionListener = OnExMethod ;
   ```

   Sie können den Delegierten entfernen, indem Sie die Eigenschaft 'ExceptionListener' wieder auf den Wert 'null' setzen:

   ```
   null:  connection.ExceptionListener = null;
   ```

Wenn keine Verweise mehr übrig sind, werden Ausnahmen oder Nachrichten automatisch vom Garbage-Collector des Systems gelöscht.

Es folgt ein Beispielcode:

```
using System;
using System.Threading;
using IBM.XMS;

public class Sample
{
    public static void Main()
    {
        XMSFactoryFactory factoryFactory = XMSFactoryFactory.GetInstance(XMSC.CT_RTT);

        IConnectionFactory connectionFactory = factoryFactory.CreateConnectionFactory();
        connectionFactory.SetStringProperty(XMSC.RTT_HOST_NAME, "localhost");
        connectionFactory.SetStringProperty(XMSC.RTT_PORT, "1506");

        //
        // Create the connection and register an exception listener
        //

        IConnection connection = connectionFactory.CreateConnection();
        connection.ExceptionListener = new ExceptionListener(Sample.OnException);

        ISession session = connection.CreateSession(false, AcknowledgeMode.AutoAcknowledge);
        IDestination topic = session.CreateTopic("topic://xms/sample");

        //
        // Create the consumer and register an async message listener
        //

        IMessageConsumer consumer = session.CreateConsumer(topic);
        consumer.MessageListener = new MessageListener(Sample.OnMessage);

        connection.Start();

        while (true)
        {
            Console.WriteLine("Waiting for messages....");
            Thread.Sleep(1000);
        }
    }

    static void OnMessage(IMessage msg)
    {
        Console.WriteLine(msg);
    }

    static void OnException(Exception ex)
    {
        Console.WriteLine(ex);
    }
}
```

Mit verwalteten Objekten arbeiten

Dieser AbschnittKapitel enthält Informationen zu verwalteten Objekten. XMS-Anwendungen können Objektdefinitionen aus einem zentralen Repository für verwaltete Objekte abrufen und zum Erstellen von Verbindungsfactorys und Zielen verwenden.

Die Informationen in diesem AbschnittKapitel sollen Ihnen helfen, verwaltete Objekte zu erstellen und zu steuern, indem die von XMS unterstützten Typen des Repositorys für verwaltete Objekte beschrieben

werden. Außerdem wird in diesem AbschnittKapitel erläutert, wie eine XMS-Anwendung eine Verbindung zu einem Repository für verwaltete Objekte herstellt, um die erforderlichen verwalteten Objekte abzurufen.

Der AbschnittKapitel enthält folgende ThemenAbschnitte:

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Unterstützte Repositorytypen für verwaltete Objekte

XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Dateisystemobjektverzeichnisse haben die Form serialisierter JNDI-Objekte (Java and Naming Directory Interface). LDAP-Objektverzeichnisse sind Verzeichnisse, die JNDI-Objekte enthalten. Dateisystem-und LDAP-Objektverzeichnisse können entweder mit dem JMSAdmin-Tool verwaltet werden, das mit IBM WebSphere MQ v6.0bereitgestellt wird, oder mit dem WebSphere MQ Explorer, der mit WebSphere MQ v7.0 und höher bereitgestellt wird. Das Dateisystem und die LDAP-Objektverzeichnisse können verwendet werden, um Clientverbindungen zu verwalten, indem IBM WebSphere MQ -Verbindungsfactorys und -Ziele zentralisiert werden. Der Netzadministrator kann mehrere Anwendungen bereitstellen, die auf dasselbe zentrale Repository verweisen und automatisch aktualisiert werden, wenn im zentralen Repository Änderungen an den Verbindungseinstellungen vorgenommen werden.

Ein COS-Benennungsverzeichnis (CORBA Object Services Naming Directory) enthält WebSphere Service Integration Bus-Verbindungsfactorys und -Ziele und kann über die Administrationskonsole für WebSphere Application Server verwaltet werden. Damit eine XMS-Anwendung Objekte aus dem CORBA Object Services Naming Directory abrufen kann, muss ein Web-Service für die JNDI-Suchfunktion bereitgestellt werden. Dieser Web-Service ist nicht in allen WebSphere Serviceintegrationstechnologien verfügbar. Detaillierte Informationen finden Sie in der entsprechenden Produktdokumentation.

Anmerkung: Damit Änderungen am Objektverzeichnis wirksam werden, müssen die Anwendungsverbindungen anschließend neu gestartet werden.

Zugehörige Konzepte

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen

Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Verwaltete Objekte
Mithilfe von verwalteten Objekten können Sie die Verbindungseinstellungen von Clientanwendungen verwalten, die über ein zentrales Repository verwaltet werden sollen. Dabei ruft eine Anwendung Objektdefinitionen aus dem zentralen Repsitory ab und erstellt basierend darauf `ConnectionFactory`- und `Destination`-Objekte. Verwaltete Objekte ermöglichen eine Entkopplung der Anwendungen von den Ressourcen, die sie während der Laufzeit verwenden.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

Eigenschaftszuordnung für verwaltete Objekte

Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

Um beispielsweise eine XMS-Verbindungsfactory mit Eigenschaften zu erstellen, die aus einer IBM WebSphere MQ JMS-Verbindungsfactory abgerufen werden, müssen die Eigenschaften zwischen beiden zugeordnet werden.

Alle Eigenschaftszuordnungen werden automatisch ausgeführt.

Die folgende Tabelle veranschaulicht die Zuordnungen einiger der gängigsten Eigenschaften von Verbindungsfactorys und Zielen. Die Eigenschaften in dieser Tabelle sind nur eine kleine Auswahl von Beispielen und nicht alle der aufgeführten Eigenschaften sind für alle Verbindungstypen und Server relevant.

Tabelle 8. Beispiele für die Namenszuordnung von Eigenschaften für Verbindungsfactorys und Ziele

IBM WebSphere MQ JMS-Eigenschaftsname	**XMS-Eigenschaftsname**
PERSISTENCE (PER)	XMSC_DELIVERY_MODE
EXPIRY (EXP)	XMSC_TIME_TO_LIVE
PRIORITY (PRI)	XMSC_PRIORITY

Tabelle 9. Beispiele für die Namenszuordnung von Eigenschaften für Verbindungsfactorys und Ziele		
IBM WebSphere MQ JMS-Eigenschaftsname	**XMS-Eigenschaftsname**	**WebSphere Service Integration Bus-Eigenschaftsname**
PERSISTENCE (PER)	XMSC_DELIVERY_MODE	
EXPIRY (EXP)	XMSC_TIME_TO_LIVE	
PRIORITY (PRI)	XMSC_PRIORITY	
	XMSC_WPM_HOST_NAME	serverName
	XMSC_WPM_BUS_NAME	busName
	XMSC_WPM_TOPIC_SPACE	topicName

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

IDestination (für die .NET-Schnittstelle)

Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

IConnectionFactory (für die .NET-Schnittstelle)
Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Eigenschaften von ConnectionFactory
Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte

Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Die in der folgenden Tabelle aufgeführten Eigenschaften müssen von einer Anwendung mindestens festgelegt werden, um eine Verbindung zu einem Messaging-Server zu erstellen. Wenn Sie anpassen möchten, wie eine Verbindung erstellt wird, kann Ihre Anwendung beliebige Eigenschaften des ConnectionFactory-Objekts zusätzlich festlegen. Weitere Informationen finden Sie im Abschnitt „Eigenschaften von ConnectionFactory” auf Seite 190. Dort finden Sie auch eine vollständige Liste der verfügbaren Eigenschaften.

Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager

Tabelle 10. Eigenschafteneinstellungen für verwaltete ConnectionFactory-Objekte für Verbindungen zu einem IBM WebSphere MQ-Warteschlangenmanager

Erforderliche XMS-Eigenschaft	Entsprechende IBM WebSphere MQ-JMS-Eigenschaft erforderlich
XMSC_CONNECTION_TYPE	XMS leitet dies anhand des Klassennamens der Verbindungsfactory und der Eigenschaft 'TRANSPORT' (TRAN) ab.
XMSC_WMQ_HOST_NAME	HOSTNAME (HOST)
XMSC_WMQ_PORT	PORT
XMSC_WMQ_QUEUE_MANAGER	Name des Warteschlangenmanagers

Echtzeitverbindung zu einem Broker

Tabelle 11. Eigenschafteneinstellungen für verwaltete ConnectionFactory-Objekte für Echtzeitverbindungen zu einem Broker

Erforderliche XMS-Eigenschaft	Entsprechende IBM WebSphere MQ-JMS-Eigenschaft erforderlich
XMSC_CONNECTION_TYPE	XMS leitet dies anhand des Klassennamens der Verbindungsfactory und der Eigenschaft 'TRANSPORT' (TRAN) ab.
XMSC_RTT_HOST_NAME	HOSTNAME (HOST)
XMSC_RTT_PORT	PORT

Verbindung zu einem WebSphere Service Integration Bus

Tabelle 12. Eigenschafteneinstellungen für verwaltete ConnectionFactory-Objekte für Verbindungen zu einem WebSphere Service Integration Bus

XMS-Eigenschaft	Beschreibung
XMSC_CONNECTION_TYPE	Der Typ des Messaging-Servers, zu dem eine Anwendung eine Verbindung herstellt.. Dieser Wert wird anhand des Klassennamens für die Verbindungsfactory bestimmt.
XMSC_WPM_BUS_NAME	Für eine Verbindungsfactory der Name des Service Integration Bus, zu dem die Anwendung eine Verbindung herstellt, bzw. für ein Ziel der Name des Service Integration Bus, in dem sich das Ziel befindet.

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager
Damit eine XMS .NET-Anwendung sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

Sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine
Damit eine XMS .NET-Anwendung sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

IConnectionFactory (für die .NET-Schnittstelle)
Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Eigenschaften von ConnectionFactory
Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

Erforderliche Eigenschaften für verwaltete Destination-Objekte

Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

Tabelle 13. IBM WebSphere MQ-JMS-Eigenschafteneinstellungen für verwaltete Destination-Objekte

Verbindungstyp	Eigenschaft	Beschreibung
IBM WebSphere MQ Queue Manager	QUEUE (QU) TOPIC (TOP)	Die Warteschlange, zu der Sie eine Verbindung herstellen möchten Das Thema, das die Anwendung als Ziel verwendet
Echtzeitverbindung zu einem Broker	TOPIC (TOP)	Das Thema, das die Anwendung als Ziel verwendet

Tabelle 14. Eigenschafteneinstellungen für verwaltete Destination-Objekte

Verbindungstyp	Eigenschaft	Beschreibung
IBM WebSphere MQ Queue Manager	QUEUE (QU) TOPIC (TOP)	Die Warteschlange, zu der Sie eine Verbindung herstellen möchten Das Thema, das die Anwendung als Ziel verwendet
Echtzeitverbindung zu einem Broker	TOPIC (TOP)	Das Thema, das die Anwendung als Ziel verwendet
WebSphere Service Integration Bus	topicName queueName	Falls Ihre Anwendung eine Verbindung zu einem Thema herstellen soll Falls Ihre Anwendung eine Verbindung zu einer Warteschlange herstellen soll

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte

Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Verwaltete Objekte erstellen

Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Vorbereitende Schritte

Weitere Informationen zu den verschiedenen Repositorytypen für verwaltete Objekte, die XMS unterstützt, finden Sie im Abschnitt „Unterstützte Repositorytypen für verwaltete Objekte” auf Seite 55.

Informationen zu diesem Vorgang

Erstellen Sie die verwalteten Objekte für IBM WebSphere MQ mit dem IBM WebSphere MQ Explorer oder IBM WebSphere MQ-JMS-Verwaltungstool (JMSAdmin).

Verwenden Sie zum Erstellen der verwalteten Objekte für IBM WebSphere MQ, WebSphere Event Brokeroder WebSphere Message Brokerdas IBM WebSphere MQ JMS-Verwaltungstool (JMSAdmin).

Verwenden Sie zum Erstellen von verwalteten Objekten für den WebSphere Service Integration Bus die Administrationskonsole für WebSphere Application Server.

In der folgenden Vorgehensweise sind die Schritte zusammengefasst, die Sie zum Erstellen von verwalteten Objekten ausgeführen müssen.

Vorgehensweise

1. Erstellen Sie eine Verbindungsfactory und definieren Sie die erforderlichen Eigenschaften, damit eine Verbindung von Ihrer Anwendung zu Ihrem ausgewählten Server hergestellt werden kann.

   Welche Eigenschaften XMS mindestens zum Herstellen einer Verbindung benötigt, ist im Abschnitt „Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte” auf Seite 58 beschrieben.
2. Erstellen Sie das erforderliche Ziel auf dem Messaging-Server, zu dem Ihre Anwendung eine Verbindung herstellen soll:

   - Erstellen Sie für eine Verbindung zu einem IBM WebSphere MQ-Queue Manager eine Warteschlange oder ein Thema.
   - Für eine Echtzeitverbindung zu einem Broker, erstellen Sie ein Thema.
   - Erstellen Sie für eine Verbindung zu einem WebSphere Service Integration Bus eine Warteschlange oder ein Thema.

   Welche Eigenschaften XMS mindestens zum Herstellen einer Verbindung benötigt, ist im Abschnitt „Erforderliche Eigenschaften für verwaltete Destination-Objekte” auf Seite 60 beschrieben.

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen

Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Verwaltete Objekte
Mithilfe von verwalteten Objekten können Sie die Verbindungseinstellungen von Clientanwendungen verwalten, die über ein zentrales Repository verwaltet werden sollen. Dabei ruft eine Anwendung Objektdefinitionen aus dem zentralen Repsitory ab und erstellt basierend darauf `ConnectionFactory`- und `Destination`-Objekte. Verwaltete Objekte ermöglichen eine Entkopplung der Anwendungen von den Ressourcen, die sie während der Laufzeit verwenden.

Mit verwalteten Objekten arbeiten
Dieser AbschnittKapitel enthält Informationen zu verwalteten Objekten. XMS-Anwendungen können Objektdefinitionen aus einem zentralen Repository für verwaltete Objekte abrufen und zum Erstellen von Verbindungsfactorys und Zielen verwenden.

ConnectionFactory- und Connection-Objekte
Ein Verbindungsfactoryobjekt (ConnectionFactory) stellt eine Vorlage bereit, die von einer Anwendung zum Erstellen eines Verbindungsobjekts (Connection) verwendet wird. Das Connection-Objekt verwendet die Anwendung wiederum zum Erstellen eines Sitzungsobjekts (Session).

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Zugehörige Tasks

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

IConnectionFactory (für die .NET-Schnittstelle)
Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Eigenschaften von ConnectionFactory
Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

IDestination (für die .NET-Schnittstelle)
Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Eigenschaften von Destination
Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

InitialContext-Objekte

Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Informationen zu diesem Vorgang

Ein InitialContext-Objekt enthält eine Verbindung zum Repository. Die XMS-API stellt Methoden zum Ausführen folgender Tasks bereit:

- InitialContext-Objekt erstellen
- Verwaltetes Objekt im Repository für verwaltete Objekte suchen

Weitere Informationen zum Erstellen eines InitialContext-Objekts finden Sie in den Abschnitten „Initial-Context” auf Seite 118 für .NET und „Eigenschaften von InitialContext” auf Seite 200.

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

InitialContext (für die .NET-Schnittstelle)
Eine Anwendung verwendet ein InitialContext-Objekt, um Objekte aus Objektdefinitionen zu erstellen, die aus einem Repository mit verwalteten Objekten abgerufen werden.

Eigenschaften von InitialContext
Eine Übersicht über die Eigenschaften des Objekts InitialContext mit Links zu detaillierteren Referenzinformationen.

InitialContext-Eigenschaften

Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

In JNDI und in der .NET-Implementierung von XMS werden die zusätzlichen Informationen in einer Umgebungs-Hashtabelle für den Konstruktor bereitgestellt.

Die Position des Repositorys für verwaltete Objekte ist in der Eigenschaft XMSC_IC_URL definiert. Diese Eigenschaft wird normalerweise im Aufruf 'Create' übergeben, kann jedoch so geändert werden, dass vor der Suche eine Verbindung zu einem anderen Benennungsverzeichnis hergestellt wird. Im Dateisystem- oder LDAP-Kontext gibt diese Eigenschaft die Verzeichnisadresse an. Bei COS-Benennungen ist dies die Adresse des Web-Services, der mit diesen Eigenschaften eine Verbindung zum JNDI-Verzeichnis herstellt.

Die folgenden Eigenschaften werden unverändert an den Web-Service übergeben, der sie zum Herstellen einer Verbindung zum JNDI-Verzeichnis verwendet.

- XMSC_IC_PROVIDER_URL
- XMSC_IC_SECURITY_CREDENTIALS
- XMSC_IC_SECURITY_AUTHENTICATION
- XMSC_IC_SECURITY_PRINCIPAL
- XMSC_IC_SECURITY_PROTOCOL

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

InitialContext (für die .NET-Schnittstelle)
Eine Anwendung verwendet ein InitialContext-Objekt, um Objekte aus Objektdefinitionen zu erstellen, die aus einem Repository mit verwalteten Objekten abgerufen werden.

Eigenschaften von InitialContext
Eine Übersicht über die Eigenschaften des Objekts InitialContext mit Links zu detaillierteren Referenzinformationen.

URI-Format für XMS-Ausgangskontexte

Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

Dateisystemkontext

Im Dateisystemkontext gibt die URL die Position eines Verzeichnisses im Dateisystem an. Die Struktur der URL ist in RFC 1738, *Uniform Resource Locators (URL)*, wie folgt definiert: Die URL hat den Präfix `file://` gefolgt von der gültigen Definition einer Datei, die auf dem System geöffnet werden kann, auf dem XMS ausgeführt wird.

Diese Syntax ist plattformspezifisch und als Trennzeichen kann entweder '/' oder '\' verwendet werden. Wenn Sie '\' als Trennzeichen verwenden möchten, muss ihm jedes Mal ein zusätzliches '\' als Escapezeichen vorangestellt werden. Dadurch wird verhindert, dass das .NET Framework versucht, das Trennzeichen als Escapezeichen für das nachfolgende Zeichen zu interpretieren.

Die folgenden Beispiele veranschaulichen diese Syntax:

```
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-Kontext

Für den LDAP-Kontext ist die Grundstruktur der URL im RFC 2255, *The LDAP URL Format* definiert, und die Syntax beginnt mit dem von Groß-/Kleinschreibung unabhängigen Präfix `ldap://`.

Die exakte Syntax ist in folgendem Beispiel dargestellt:

```
LDAP://[Hostname][:Port]["/"[DistinguishedName]]
```

Diese Syntax ist zwar in dem genannten RFC definiert, jedoch ohne Unterstützung für jegliche Attribute, Gültigkeitsbereiche, Filter oder Erweiterungen.

Folgende Beispiele veranschaulichen diese Syntax:

```
ldap://madison:389/cn=JMSData,dc=IBM,dc=UK
ldap://madison/cn=JMSData,dc=IBM,dc=UK
LDAP:///cn=JMSData,dc=IBM,dc=UK
```

WSS-Kontext

Für den WSS-Kontext hat die URL das Format eines Web-Service-Endpunkts mit dem Präfix `http://`.

Alternativ können Sie das Präfix `cosnaming://` oder `wsvc://` verwenden.

Die Bedeutung dieser beiden Präfixe wird so interpretiert, dass Sie einen WSS-Kontext verwenden, wobei der Zugriff auf die URL über das Hypertext Transfer Protocol (HTTP) erfolgt, sodass der ursprüngliche Kontexttyp einfach direkt aus der URL abgeleitet werden kann.

Folgende Beispiele veranschaulichen diese Syntax:

```
http://madison.ibm.com:9080/xmsjndi/services/JndiLookup
cosnaming://madison/jndilookup
```

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

JNDI-Suche, Web-Service
Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

InitialContext (für die .NET-Schnittstelle)

Eine Anwendung verwendet ein InitialContext-Objekt, um Objekte aus Objektdefinitionen zu erstellen, die aus einem Repository mit verwalteten Objekten abgerufen werden.

Eigenschaften von InitialContext
Eine Übersicht über die Eigenschaften des Objekts InitialContext mit Links zu detaillierteren Referenzinformationen.

JNDI-Suche, Web-Service

Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Der Web-Service wird in der EAR-Datei SIBXJndiLookupEAR.ear bereitgestellt, die sich im Installationsverzeichnis befindet. Für das aktuelle Release von Message Service Client for .NETfinden Sie SIBXJndiLookupEAR.ear im Verzeichnis `<install_dir>\java\lib` . Dieser kann in einem WebSphere -SIB-Server über die Administrationskonsole oder mit dem Scripting-Tool "wsaadmin" installiert werden. Weitere Informationen zur Bereitstellung von Web-Service-Anwendungen finden Sie in der Produktdokumentation.

Um den Web-Service in XMS-Anwendungen zu definieren, geben Sie im InitialContext-Objekt für die Eigenschaft XMSC_IC_URL die URL des Web-Service-Endpunkts an. Wird der Web-Service beispielsweise auf einem Server-Host namens MyHost bereitgestellt, lautet die Endpunkt-URL des Web-Service wie folgt:

```
wsvc://MyHost:9080/SIBXJndiLookup/services/JndiLookup
```

Dank der Festlegung der XMSC_IC_URL-Eigenschaft kann bei einer InitialContext-Suche der Web-Service am definierten Endpunkt aufgerufen werden, der dann wiederum im CORBA Object Services Naming Directory nach dem gewünschten verwalteten Objekt sucht.

.NET-Anwendungen können den Web-Service verwenden. Die serverseitige Bereitstellung ist für XMS C, /C++ und XMS .NET identisch.XMS.NET ruft den Web-Service direkt über das Microsoft .NET -Framework auf.

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Message Service Client for .NET mit dem Installationsassistenten installieren
Für die Installation wird ein InstallShield X/Windows MSI-Installationsprogramm verwendet. Es sind zwei Installationsoptionen verfügbar: Sie können zwischen einer vollständigen und einer angepassten Installation wählen.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

Verwaltete Objekte abrufen

Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Objekte, die abgerufen werden sollen, können folgende Arten von Namen haben:

- Einen einfachen Namen, der das Destination-Objekt beschreibt, beispielsweise das Warteschlangenziel 'SalesOrders'
- Einen zusammengesetzten Namen, der aus mehreren durch das Trennzeichen '/' getrennten Subkontexten bestehen kann und mit dem Objektnamen enden muss. Ein Beispiel für einen zusammengesetzten Namen ist 'Warehouse/PickLists/DispatchQueue2', wobei 'Warehouse' und 'Picklists' Subkontexte im Benennungsverzeichnis sind und 'DispatchQueue2' der Name eines Destination-Objekts ist.

Zugehörige Konzepte

Unterstützte Repositorytypen für verwaltete Objekte
XMS unterstützt drei Typen von Verzeichnissen für verwaltete Objekte: Dateisystem, Lightweight Directory Access Protocol (LDAP) und COS-Benennung. Mit verwalteten Dateisystemobjekten und LDAP-Objekten können Verbindungen zu IBM WebSphere MQ und WebSphere Application Server hergestellt werden, während COS-Benennungen nur Verbindungen zu WebSphere Application Server ermöglichen.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

JNDI-Suche, Web-Service

Um aus XMS auf ein CORBA Object Services Naming Directory zugreifen zu können, muss ein Web-Service für eine JNDI-Suche auf einem WebSphere Service Integration Bus-Server bereitgestellt werden. Dieser Web-Service setzt die Java-Informationen aus dem COS-Namensservice in ein Format um, das XMS -Anwendungen lesen können.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

InitialContext (für die .NET-Schnittstelle)
Eine Anwendung verwendet ein InitialContext-Objekt, um Objekte aus Objektdefinitionen zu erstellen, die aus einem Repository mit verwalteten Objekten abgerufen werden.

Eigenschaften von InitialContext
Eine Übersicht über die Eigenschaften des Objekts InitialContext mit Links zu detaillierteren Referenzinformationen.

Kommunikation für XMS-Anwendungen sichern

Dieser AbschnittKapitel enthält Informationen zur Einrichtung einer sicheren Kommunikation, damit XMS-Anwendungen über Secure Sockets Layer (SSL) eine Verbindung zu einer WebSphere Service Integration Bus-Messaging-Engine oder einem IBM WebSphere MQ-Queue Manager herstellen können.

Dieser AbschnittKapitel enthält Informationen zur Konfiguration von XMS-ConnectionFactory-Eigenschaften, um Anwendungen die Herstellung sicherer Verbindungen zu ermöglichen.

Der AbschnittKapitel enthält folgende ThemenAbschnitte:

Sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager

Damit eine XMS .NET-Anwendung sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

Ob bei der Verschlüsselungsverhandlung das SSL-Protokoll (Secure Sockets Layer) oder TLS-Protokoll (Transport Layer Security) verwendet wird, hängt davon ab, welche Cipher-Suite Sie im ConnectionFactory-Objekt angeben.

Wenn Sie die Clientbibliotheken von IBM WebSphere MQ Version 7.0.0.1 und höher verwenden und eine Verbindung zu einem Warteschlangenmanager von IBM WebSphere MQ Version 7 herstellen, können Sie mehrere Verbindungen zu demselben Warteschlangenmanager in der Anwendung XMS herstellen. Mehrere Verbindungen zu unterschiedlichen Warteschlangenmanagern sind jedoch nicht zulässig. Wenn Sie dies versuchen, wird der `MQRC_SSL_ALREADY_INITIALIZED` ausgelöst.

Wenn Sie die Clientbibliotheken von IBM WebSphere MQ Version 6 und höher verwenden, können Sie eine SSL-Verbindung nur erstellen, wenn Sie zuvor eine SSL-Verbindung geschlossen haben. Mehrere gleichzeitig bestehende SSL-Verbindungen von demselben Prozess zu demselben Warteschlangenmanager oder zu mehreren unterschiedlichen Warteschlangenmanagern sind nicht zulässig. Wenn Sie versu-

chen, mehrere Anforderungen zu stellen, wird die Warnung MQRC_SSL_ALREADY_INITIALIZED ausgegeben, was bedeuten kann, dass einige für die SSL-Verbindung angeforderte Parameter ignoriert wurden.

Die folgende Tabelle enthält ConnectionFactory-Eigenschaften für Verbindungen über SSL zu einem IBM WebSphere MQ-Manager, mit einer Kurzbeschreibung:

Tabelle 15. Eigenschaften von ConnectionFactory für Verbindungen zu einem IBM WebSphere MQ-Queue Manager über SSL

Name der Eigenschaft	Beschreibung
XMSC_WMQ_SSL_CERT_STORES	Die Positionen der Server, auf denen sich die Zertifikatswiderrufslisten (CRLs) befinden, die für eine SSL-Verbindung zu einem Warteschlangenmanager verwendet werden sollen.
XMSC_WMQ_SSL_CIPHER_SPEC	Der Name der CipherSpec, die für eine sichere Verbindung zu einem Warteschlangenmanager verwendet werden soll.
XMSC_WMQ_SSL_CIPHER_SUITE	Der Name der CipherSuite , die für eine SSL-oder TLS-Verbindung zu einem Warteschlangenmanager verwendet werden soll. Welches Protokoll bei der Vereinbarung der sicheren Verbindung verwendet wird, ist von der angegebenen Cipher-Suite abhängig.
XMSC_WMQ_SSL_CRYPTO_HW	Konfigurationsdetails für die Verschlüsselungshardware, die mit dem Clientsystem verbunden ist.
XMSC_WMQ_SSL_FIPS_REQUIRED	Der Wert dieser Eigenschaft legt fest, ob eine Anwendung nicht FIPS-konforme Cipher-Suites verwenden kann oder nicht. Wenn diese Eigenschaft auf 'true' gesetzt ist, werden nur FIPS-Algorithmen für die Client-Server-Verbindung verwendet.
XMSC_WMQ_SSL_KEY_REPOSITORY	Gibt die Position der Schlüsseldatenbankdatei an, in der Schlüssel und Zertifikate gespeichert werden.
XMSC_WMQ_SSL_KEY_RESETCOUNT	Der Wert von KeyResetCount gibt die Gesamtzahl unverschlüsselter Bytes an, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL-Dialog gesendet oder empfangen werden.
XMSC_WMQ_SSL_PEER_NAME	Der Peername, der für eine SSL-Verbindung zu einem Warteschlangenmanager verwendet werden soll.

Zugehörige Verweise

IConnectionFactory (für die .NET-Schnittstelle)
Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Eigenschaften von ConnectionFactory
Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Namenszuordnung zwischen Cipher-Suites und CipherSpecs für Verbindungen zu einem IBM WebSphere MQQueue Manager

Das InitialContext-Objekt übernimmt die Umsetzung zwischen der Eigenschaft 'SSLCIPHERSUITE' der JMSAdmin-Verbindungsfactory und der fast äquivalenten XMS-Eigenschaft 'XMSC_WMQ_SSL_CIPHER_SPEC'. Eine ähnliche Umsetzung ist erforderlich, wenn Sie einen Wert für 'XMSC_WMQ_SSL_CIPHER_SUITE' angeben, aber einen Wert für 'XMSC_WMQ_SSL_CIPHER_SPEC' weglassen.

In Tabelle 16 auf Seite 72 sind die verfügbaren CipherSpecs und ihre JSSE-Entsprechungen CipherSuite aufgelistet.

Tabelle 16. Verfügbare CipherSpecs und deren entsprechende JSSE-Cipher-Suites

CipherSpec	**Entsprechende JSSE-Cipher-Suite**
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA
NULL_MD5	SSL_RSA_WITH_NULL_MD5
NULL_SHA	SSL_RSA_WITH_NULL_SHA
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA
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
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA

Anmerkung: Eine Eins-zu-eins-Zuordnung für den Cipher-Suite-Namen SSL_RSA_WITH_3DES_EDE_CBC_SHA oder SSL_RSA_WITH_DES_CBC_SHA muss die Einstellung der Eigenschaft XMSC_WMQ_SSL_FIPSREQUIRED berücksichtigen und eine Heuristik anwenden.

Wenn Sie SSL_RSA_WITH_3DES_EDE_CBC_SHA oder SSL_RSA_WITH_DES_CBC_SHA für die Eigenschaft XMSC_WMQ_SSL_CIPHER_SUITEangeben, und kein Wert für XMSC_WMQ_SSL_CIPHER_SPECvorhanden ist, wird ein Wert für XMSC_WMQ_SSL_CIPHER_SPEC gemäß den folgenden Tabellen ausgewählt.

Die Werte, die für XMSC_WMQ_SSL_CIPHER_SPEC verwendet werden, wenn Sie SSL_RSA_WITH_3DES_EDE_CBC_SHA für die Eigenschaft XMSC_WMQ_SSL_CIPHER_SUITE angeben, werden in der folgenden Tabelle gezeigt:

Tabelle 17. Verwendete Werte für XMSC_WMQ_SSL_CIPHER_SPEC bei Angabe von SSL_RSA_WITH_3DES_EDE_CBC_SHA für die Eigenschaft XMSC_WMQ_SSL_CIPHER_SUITE

Eingabe: XMSC_WMQ_SSL_FIPSREQUIRED-Wert	**Ausgabe: XMSC_WMQ_SSL_CIPHER_SPEC ausgewählt**
false (d. h. MQSSL_FIPS_NO)	TRIPLE_DES_SHA_US
true (d. h. MQSSL_FIPS_YES)	TLS_RSA_WITH_3DES_EDE_CBC_SHA

Anmerkung:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA ist veraltet. Nach wie vor sind mit dieser CipherSpec jedoch noch Datenübertragungen bis zu 32 GB möglich, bevor die Verbindung mit Fehler AMQ9288 beendet wird. Zur Vermeidung dieses Fehlers sollten Sie entweder auf Triple DES verzichten oder, wenn Sie diese CipherSpec verwenden möchten, die Zurücksetzung von geheimen Schlüsseln aktivieren.

Die Werte, die für XMSC_WMQ_SSL_CIPHER_SPEC verwendet werden, wenn Sie SSL_RSA_WITH_DES_CBC_SHA für die Eigenschaft XMSC_WMQ_SSL_CIPHER_SUITE angeben, werden in der folgenden Tabelle gezeigt:

Tabelle 18. Verwendete Werte für XMSC_WMQ_SSL_CIPHER_SPEC bei Angabe von SSL_RSA_WITH_DES_CBC_SHA für die Eigenschaft XMSC_WMQ_SSL_CIPHER_SUITE

Eingabe: XMSC_WMQ_SSL_FIPSREQUIRED-Wert	**Ausgabe: XMSC_WMQ_SSL_CIPHER_SPEC ausgewählt**
false (d. h. MQSSL_FIPS_NO)	DES_SHA_EXPORT
true (d. h. MQSSL_FIPS_YES)	TLS_RSA_WITH_DES_CBC_SHA

Sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine

Damit eine XMS .NET-Anwendung sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

XMS stellt SSL- und HTTPS-Unterstützung für Verbindungen zu einem WebSphere Service Integration Bus bereit. Die Protokolle SSL und HTTPS ermöglichen sichere Verbindungen zur Authentifizierung und für Vertraulichkeit.

Wie die WebSphere-Sicherheit wird die XMS-Sicherheit unter Beachtung von JSSE-Sicherheitsstandards und -Namenskonventionen konfiguriert, was die Verwendung von Cipher-Suites zur Angabe der Algorithmen, die bei der Vereinbarung einer sicheren Verbindung verwendet werden, einschließt. Ob bei der Verschlüsselungsverhandlung das SSL-Protokoll oder TLS-Protokoll verwendet wird, hängt davon ab, welche Cipher-Suite Sie im ConnectionFactory-Objekt angeben.

Anmerkung: Für eine .NET-Anwendung werden die Sicherheitsfunktionen durch Microsoft Secure Channel (SChannel) bereitgestellt.

In Tabelle 19 auf Seite 73 sind die Eigenschaften aufgeführt, die im ConnectionFactory-Objekt definiert sein müssen.

Tabelle 19. Eigenschaften von ConnectionFactory für sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine

Name der Eigenschaft	**Beschreibung**
XMSC_WPM_SSL_CIPHER_SUITE	Der Name der Cipher-Suite, die in einer SSL- oder TLS-Verbindung zu einer Messaging-Engine für WebSphere Service Integration Bus verwendet werden soll. Welches Protokoll bei der Vereinbarung der sicheren Verbindung verwendet wird, ist von der angegebenen Cipher-Suite abhängig. **Anmerkung:** Diese Eigenschaft wird in einer .NET-Anwendung unterstützt.
XMSC_WPM_SSL_KEYRING_LABEL	Das Zertifikat, das bei der Authentifizierung beim Server verwendet werden soll. **Anmerkung:** Diese Eigenschaft wird in einer .NET-Anwendung unterstützt.

Es folgt ein Beispiel für ConnectionFactory-Eigenschaften für sichere Verbindungen zu einer WebSphere Integration-Messaging-Engine:

```
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);
```

Dabei sollte die Variable 'chain_name' (Kettenname) entweder auf 'BootstrapTunneledSecureMessaging' oder 'BootstrapSecureMessaging' und die Variable 'port_number' (Portnummer) auf die Nummer des Ports festgelegt sein, den der Bootstrap-Server auf eingehende Anforderungen überwacht.

Es folgt ein Beispiel für ConnectionFactory-Eigenschaften für sichere Verbindungen zu einer WebSphere Integration-Messaging-Engine mit eingefügten Beispielwerten:

```
  /* 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");
```

Zugehörige Verweise

IConnectionFactory (für die .NET-Schnittstelle)
Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Eigenschaften von ConnectionFactory
Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Cipher-Suite- und CipherSpec-Namenszuordnungen für Verbindungen zu einem WebSphere Service Integration Bus

Da das GSKit (Global Security Kit) CipherSpecs anstelle von Cipher-Suites verwendet, müssen die in der Eigenschaft 'XMSC_WPM_SSL_CIPHER_SUITE' im JSSE-Format angegebenen Cipher-Suite-Namen den im GSKit-Format angegebenen CipherSpec-Namen zugeordnet werden.

In Tabelle 20 auf Seite 74 sind die entsprechenden CipherSpecs für die jeweiligen erkannten Cipher-Suites aufgeführt.

Tabelle 20. Verfügbare Cipher-Suites und deren entsprechende CipherSpecs

CipherSuite	**Entsprechende CipherSpec**
SSL_RSA_WITH_NULL_MD5	NULL_MD5
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	RC2_MD5_EXPORT
SSL_RSA_EXPORT_WITH_RC4_40_MD5	RC4_MD5_EXPORT
SSL_RSA_WITH_RC4_128_MD5	RC4_MD5_US
SSL_RSA_WITH_NULL_SHA	NULL_SHA
SSL_RSA_EXPORT1024_WITH_RC4_56_SHA	RC4_56_SHA_EXPORT1024
SSL_RSA_WITH_RC4_128_SHA	RC4_SHA_US
SSL_RSA_WITH_DES_CBC_SHA	DES_SHA_EXPORT
SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA	DES_SHA_EXPORT1024
SSL_RSA_FIPS_WITH_DES_CBC_SHA	FIPS_WITH_DES_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TRIPLE_DES_SHA_US
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	FIPS_WITH_3DES_EDE_CBC_SHA

Tabelle 20. Verfügbare Cipher-Suites und deren entsprechende CipherSpecs (Forts.)	
CipherSuite	**Entsprechende CipherSpec**
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

Anmerkung:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA ist veraltet. Nach wie vor sind mit dieser CipherSpec jedoch noch Datenübertragungen bis zu 32 GB möglich, bevor die Verbindung mit Fehler AMQ9288 beendet wird. Zur Vermeidung dieses Fehlers sollten Sie entweder auf Triple DES verzichten oder, wenn Sie diese CipherSpec verwenden möchten, die Zurücksetzung von geheimen Schlüsseln aktivieren.

XMS-Nachrichten

In diesem AbschnittKapitel werden die Struktur und der Inhalt von XMS-Nachrichten beschrieben und es wird erläutert, wie Anwendungen XMS-Nachrichten verarbeiten.

Dieser AbschnittKapitel enthält folgende ThemenAbschnitte:

Zugehörige Verweise

IMessage (für die .NET-Schnittstelle)
Ein Nachrichtenobjekt stellt eine Nachricht dar, die von einer Anwendung gesendet oder empfangen wird. IMessage ist eine Superklasse für die Nachrichtenklassen, wie beispielsweise IMapMessage.

Teile der Eine XMS -Nachricht

Eine XMS-Nachricht besteht aus einem Header, einer Gruppe von Eigenschaften und einem Hauptteil.

Kopfzeile
Der Header einer Nachricht enthält Felder, wobei die Gruppe der Headerfelder für alle Nachrichten identisch ist. Mithilfe der Werte der Headerfelder können XMS-Anwendungen Nachrichten ermitteln und weiterleiten. Weitere Informationen zu Headerfeldern finden Sie im Abschnitt „Headerfelder in Eine XMS -Nachricht” auf Seite 76.

Gruppe der Eigenschaften
Die Eigenschaften einer Nachricht enthalten zusätzliche Informationen zu dieser Nachricht. Obwohl die Gruppe der Headerfelder für alle Nachrichten identisch ist, können alle Nachrichten unterschiedliche Gruppen von Eigenschaften haben. Weitere Informationen finden Sie unter „Eigenschaften der Eine XMS -Nachricht” auf Seite 77.

Hauptteil
Der Hauptteil einer Nachricht enthält Anwendungsdaten. Weitere Informationen finden Sie unter „Hauptteil der Eine XMS -Nachricht” auf Seite 80.

Eine Anwendung kann auswählen, welche Nachrichten sie empfangen möchte. Dies geschieht mithilfe von Nachrichtenselektoren, die die Auswahlkriterien angeben. Die Kriterien können auf den Werten bestimmter Headerfelder und den Werten aller Eigenschaften einer Nachricht basieren. Weitere Informationen zu Nachrichtenselektoren finden Sie unter „Nachrichtenselektoren” auf Seite 87.

Zugehörige Verweise

Headerfelder in Eine XMS -Nachricht
Damit die Eine XMS -Anwendung Nachrichten mit einer WebSphere JMS -Anwendung austauschen kann, enthält der Header der Eine XMS -Nachricht die JMS-Nachrichtenheaderfelder.

Eigenschaften der Eine XMS -Nachricht
XMS unterstützt drei Arten von Nachrichteneigenschaften: JMS-definierte Eigenschaften, von IBM definierte Eigenschaften und anwendungsdefinierte Eigenschaften.

Hauptteil der Eine XMS -Nachricht
Der Hauptteil einer Nachricht enthält Anwendungsdaten. Eine Nachricht kann jedoch durchaus keinen Rumpf bzw. Hauptteil haben und nur die Headerfelder und Eigenschaften enthalten.

Nachrichtenselektoren
Eine XMS-Anwendung verwendet Nachrichtenselektoren, um die Nachrichten auszuwählen, die sie empfangen möchte.

Zuordnung von XMS-Nachrichten zu IBM WebSphere MQ-Nachrichten
Die JMS-Header-Felder und -Eigenschaften von Eine XMS-Nachrichten werden Feldern in den Headerstrukturen einer IBM WebSphere MQ-Nachricht zugeordnet.

Headerfelder in Eine XMS -Nachricht

Damit die Eine XMS -Anwendung Nachrichten mit einer WebSphere JMS -Anwendung austauschen kann, enthält der Header der Eine XMS -Nachricht die JMS-Nachrichtenheaderfelder.

Die Namen dieser Headerfelder beginnen mit dem Präfix JMS. Eine Beschreibung der JMS-Nachrichtenheaderfelder finden Sie im Abschnitt *Java Message Service Specification Version 1.1*.

XMS implementiert die JMS-Nachrichtenheaderfelder als Attribute eines Message-Objekts. Für jedes Headerfeld gibt es spezifische Methoden, mit denen sein Wert festgelegt und abgerufen wird. Eine Beschreibung dieser Methoden finden Sie im Abschnitt „IMessage” auf Seite 131. Ein Headerfeld ist immer lesbar und änderbar (ohne Schreib- oder Leseschutz).

In Tabelle 21 auf Seite 76 werden die JMS-Nachrichtenheaderfelder aufgelistet und es wird gezeigt, wie der Wert jedes einzelnen Felds für eine übertragene Nachricht festgelegt wird. Für einige Felder wird der Wert automatisch durch XMS festgelegt, wenn eine Anwendung eine Nachricht sendet oder, im Fall von 'JMSRedelivered', wenn eine Anwendung eine Nachricht empfängt.

Tabelle 21. JMS-Nachrichtenheaderfelder

Name des JMS-Nachrichtenheaderfelds	**Vorgang zum Festlegen des Werts für eine übertragene Nachricht (Format: *Methode* [*Klasse*])**
JMSCorrelationID	Set JMSCorrelationID [Message]
JMSDeliveryMode	Send [MessageProducer]
JMSDestination	Send [MessageProducer]
JMSExpiration	Send [MessageProducer]
JMSMessageID	Send [MessageProducer]
JMSPriority	Send [MessageProducer]
JMSRedelivered	Receive [MessageConsumer]
JMSReplyTo	Set JMSReplyTo [Message]
JMSTimestamp	Send [MessageProducer]
JMSType	Set JMSType [Message]

Zugehörige Verweise

Teile der Eine XMS -Nachricht

Eine XMS-Nachricht besteht aus einem Header, einer Gruppe von Eigenschaften und einem Hauptteil.

Eigenschaften der Eine XMS -Nachricht
XMS unterstützt drei Arten von Nachrichteneigenschaften: JMS-definierte Eigenschaften, von IBM definierte Eigenschaften und anwendungsdefinierte Eigenschaften.

Hauptteil der Eine XMS -Nachricht
Der Hauptteil einer Nachricht enthält Anwendungsdaten. Eine Nachricht kann jedoch durchaus keinen Rumpf bzw. Hauptteil haben und nur die Headerfelder und Eigenschaften enthalten.

Nachrichtenselektoren
Eine XMS-Anwendung verwendet Nachrichtenselektoren, um die Nachrichten auszuwählen, die sie empfangen möchte.

Zuordnung von XMS-Nachrichten zu IBM WebSphere MQ-Nachrichten
Die JMS-Header-Felder und -Eigenschaften von Eine XMS-Nachrichten werden Feldern in den Headerstrukturen einer IBM WebSphere MQ-Nachricht zugeordnet.

Eigenschaften der Eine XMS -Nachricht

XMS unterstützt drei Arten von Nachrichteneigenschaften: JMS-definierte Eigenschaften, von IBM definierte Eigenschaften und anwendungsdefinierte Eigenschaften.

Eine XMS-Anwendung kann Nachrichten mit einer WebSphere JMS-Anwendung austauschen, weil XMS die folgenden vordefinierten Eigenschaften eines Message-Objekts unterstützt:

- Dieselben JMS-definierten Eigenschaften, die WebSphere JMS unterstützt. Die Namen dieser Eigenschaften beginnen mit dem Präfix 'JMSX'.
- Dieselben von IBM definierten Eigenschaften, die WebSphere JMS unterstützt. Die Namen dieser Eigenschaften beginnen mit dem Präfix 'JMS_IBM_'.

Jede vordefinierte Eigenschaft hat zwei Namen:

- Ein JMS-Name für eine JMS-definierte Eigenschaft oder ein WebSphere JMS -Name für eine IBM-definierte Eigenschaft.

  Dies ist der Name, unter dem die Eigenschaft in JMS oder WebSphere JMS bekannt ist, und es ist außerdem der Name, der mit einer Nachricht übertragen wird, die diese Eigenschaft aufweist. Eine XMS-Anwendung identifiziert anhand dieses Namens die Eigenschaft in einem Nachrichtenselektorausdruck.
- Eine XMS , um die Eigenschaft in allen Situationen außer in einem Nachrichtenselektorausdruck zu identifizieren. Jeder XMS-Name ist als benannte Konstante in der Klasse `IBM.XMS.XMSC` definiert. Der Wert der benannten Konstante ist der entsprechende JMS-oder WebSphere JMS-Name.

Zusätzlich zu den vordefinierten Eigenschaften kann Eine XMS-Anwendung eine eigene Gruppe von Nachrichteneigenschaften erstellen und verwenden. Dies sind die so genannten *anwendungsdefinierten Eigenschaften*.

Nachdem eine Anwendung eine Nachricht erstellt hat, sind die Eigenschaften der Nachricht sowohl lesbar als auch änderbar, also weder lese- noch schreibgeschützt. Auch wenn die Anwendung die Nachricht gesendet hat, bleiben die Eigenschaften lesbar und änderbar. Wenn eine Anwendung eine Nachricht empfängt, sind die Eigenschaften der Nachricht zunächst schreibgeschützt. Wenn eine Anwendung die Methode `Clear Properties` der Nachrichtenklasse aufruft, wenn die Eigenschaften einer Nachricht schreibgeschützt sind, werden die Eigenschaften lesbar und beschreibbar. Außerdem löscht diese Methode die Eigenschaften.

Wenn die empfangene Nachricht nach dem Löschen der Nachrichteneigenschaften weitergeleitet wird, ist das weitere Verhalten der Nachricht konsistent mit dem Weiterleitungsverhalten eines standardmäßigen BytesMessage-Objekts für WMQ XMS for .NET, dessen Eigenschaften gelöscht wurden.

Dies wird jedoch nicht empfohlen, weil dabei folgende Eigenschaften verloren gehen:

- Der Eigenschaftswert 'JMS_IBM_Encoding', der angibt, dass die Nachrichtendaten nicht aussagekräftig entschlüsselt werden können.

- Der Eigenschaftswert 'JMS_IBM_Format', der angibt, dass die Headerverkettung zwischen dem MQMD- bzw. dem neuen MQRFH2-Nachrichtenheader und den vorhandenen Headern unterbrochen wird.

Zugehörige Verweise

Teile der Eine XMS -Nachricht
Eine XMS-Nachricht besteht aus einem Header, einer Gruppe von Eigenschaften und einem Hauptteil.

Headerfelder in Eine XMS -Nachricht
Damit die Eine XMS -Anwendung Nachrichten mit einer WebSphere JMS -Anwendung austauschen kann, enthält der Header der Eine XMS -Nachricht die JMS-Nachrichtenheaderfelder.

Hauptteil der Eine XMS -Nachricht
Der Hauptteil einer Nachricht enthält Anwendungsdaten. Eine Nachricht kann jedoch durchaus keinen Rumpf bzw. Hauptteil haben und nur die Headerfelder und Eigenschaften enthalten.

Nachrichtenselektoren
Eine XMS-Anwendung verwendet Nachrichtenselektoren, um die Nachrichten auszuwählen, die sie empfangen möchte.

Zuordnung von XMS-Nachrichten zu IBM WebSphere MQ-Nachrichten
Die JMS-Header-Felder und -Eigenschaften von Eine XMS-Nachrichten werden Feldern in den Headerstrukturen einer IBM WebSphere MQ-Nachricht zugeordnet.

JMS-definierte Eigenschaften einer Nachricht

Es werden mehrere JMS-definierte Eigenschaften einer Nachricht sowohl von XMS als auch von WebSphere JMS unterstützt.

In Tabelle 22 auf Seite 78 werden die JMS-definierten Eigenschaften einer Nachricht aufgelistet, die sowohl von XMS als auch von WebSphere JMS unterstützt werden. Eine Beschreibung der JMS-definierten Eigenschaften finden Sie im Abschnitt *Java Message Service Specification Version 1.1*. Für eine Echtzeitverbindung zu einem Broker sind die JMS-definierten Eigenschaften nicht gültig.

In der Tabelle ist für alle Eigenschaften der Datentyp angegeben und wie der jeweilige Eigenschaftswert für eine übertragene Nachricht festgelegt wird. Für einige Eigenschaften wird der Wert automatisch durch XMS festgelegt, wenn eine Anwendung eine Nachricht sendet oder, im Fall von 'JMSXDeliveryCount', wenn eine Anwendung eine Nachricht empfängt.

Tabelle 22. JMS-definierte Eigenschaften einer Nachricht

XMS-Name der JMS-definierten Eigenschaft	**JMS-Name**	**Datentyp**	**Wie der Wert für eine übertragene Nachricht festgelegt wird (im Format *Methode* [*Klasse*])**
JMSX_APPID	JMSXAppID	System.String	Send [MessageProducer]
JMSX_DELIVERY_COUNT	JMSXDeliveryCount	System.Int32	Receive [MessageConsumer]
JMSX_GROUPID	JMSXGroupID	System.String	Set String Property [PropertyContext]
JMSX_GROUPSEQ	JMSXGroupSeq	System.Int32	Set Integer Property [PropertyContext]
JMSX_USERID	JMSXUserID	System.String	Send [MessageProducer]

IBM-definierte Eigenschaften einer Nachricht

Es werden mehrere von IBM definierte Eigenschaften einer Nachricht XMS und WebSphere JMS unterstützt.

In Tabelle 23 auf Seite 79 werden die von IBM definierten Eigenschaften einer Nachricht aufgelistet, die sowohl von XMS als auch von WebSphere JMS unterstützt werden. Weitere Informationen zu den

von IBM definierten Eigenschaften finden Sie im Handbuch *IBM WebSphere MQ Using Java* oder in der Produktdokumentation zu WebSphere Application Server.

In der Tabelle ist für alle Eigenschaften der Datentyp angegeben und wie der jeweilige Eigenschaftswert für eine übertragene Nachricht festgelegt wird. Für einige Eigenschaften wird der Wert automatisch durch XMS festgelegt, wenn eine Anwendung eine Nachricht sendet.

Tabelle 23. IBM-definierte Eigenschaften einer Nachricht

XMS-Name der IBM-definierten Eigenschaft	**WebSphere JMS-Name**	**Datentyp**	**Wie der Wert für eine übertragene Nachricht festgelegt wird (im Format *Methode* [*Klasse*])**
JMS_IBM_CHARACTER_SET	JMS_IBM_Character_Set	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_ENCODING	JMS_IBM_Encoding	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_EXCEPTION-MESSAGE	JMS_IBM_ExceptionMessage	System.String	Receive [MessageConsumer]
JMS_IBM_EXCEPTION-REASON	JMS_IBM_ExceptionReason	System.Int32	Receive [MessageConsumer]
JMS_IBM_EXCEPTIONTIMESTAMP	JMS_IBM_ExceptionTimestamp	System.Int64	Receive [MessageConsumer]
JMS_IBM_EXCEPTION-PROBLEM DESTINATION	JMS_IBM_ExceptionProblemDestination	System.String	Receive [MessageConsumer]
JMS_IBM_FEEDBACK	JMS_IBM_Feedback	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_FORMAT	JMS_IBM_Format	System.String	Set String Property [PropertyContext]
JMS_IBM_LAST_MSG_IN_GROUP	JMS_IBM_Last_Msg_In_Group	System.Boolean	Set Integer Property [PropertyContext]
JMS_IBM_MSGTYPE	JMS_IBM_MsgType	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_PUTAPPLTYPE	JMS_IBM_PutApplType	System.Int32	Send [MessageProducer]
JMS_IBM_PUTDATE	JMS_IBM_PutDate	System.String	Send [MessageProducer]
JMS_IBM_PUTTIME	JMS_IBM_PutTime	System.String	Send [MessageProducer]
JMS_IBM_REPORT_COA	JMS_IBM_Report_COA	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_COD	JMS_IBM_Report_COD	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_DISCARD_MSG	JMS_IBM_Report_Discard_Msg	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_EXCEPTION	JMS_IBM_Report_Exception	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_EXPIRATION	JMS_IBM_Report_Expiration	System.Int32	Set Integer Property [PropertyContext]

Tabelle 23. IBM-definierte Eigenschaften einer Nachricht (Forts.)			
XMS-Name der IBM-definierten Eigenschaft	**WebSphere JMS-Name**	**Datentyp**	**Wie der Wert für eine übertragene Nachricht festgelegt wird (im Format *Methode* [*Klasse*])**
JMS_IBM_REPORT_NAN	JMS_IBM_Report_NAN	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_PAN	JMS_IBM_Report_PAN	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_PASS_CORREL_ID	JMS_IBM_Report_Pass_Correl_ID	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_PASS_MSG_ID	JMS_IBM_Report_Pass_Msg_ID	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_SYSTEM_MESSAGEID	JMS_IBM_System_MessageID	System.String	Send [MessageProducer]

Anwendungsdefinierte Eigenschaften einer Nachricht

Eine XMS-Anwendung kann ihre eigenen Nachrichteneigenschaften erstellen und verwenden. Wenn eine Anwendung einer Nachricht sendet, werden diese Eigenschaften ebenfalls mit der Nachricht übertragen. Eine empfangende Anwendung kann dann mithilfe von Nachrichtenselektoren anhand der Werte dieser Eigenschaften die Nachrichten auswählen, die sie empfangen möchte.

Damit eine WebSphere JMS -Anwendung Nachrichten auswählen und verarbeiten kann, die von einer XMS -Anwendung gesendet werden, muss der Name einer anwendungsdefinierten Eigenschaft den Regeln für die Bildung von Kennungen in Nachrichtenselektorausdrücken entsprechen (siehe *IBM WebSphere MQ Javaverwenden*). Der Wert einer anwendungsdefinierten Eigenschaft muss einen der folgenden Datentypen haben: System.Boolean, System.SByte, System.Int16, System.Int32, System.Int64, System.Float, System.Double oder System.String.

Hauptteil der Eine XMS -Nachricht

Der Hauptteil einer Nachricht enthält Anwendungsdaten. Eine Nachricht kann jedoch durchaus keinen Rumpf bzw. Hauptteil haben und nur die Headerfelder und Eigenschaften enthalten.

XMS unterstützt fünf Nachrichtenhauptteiltypen:

Bytes
Der Hauptteil enthält einen Bytestrom. Eine Nachricht mit diesem Hauptteiltyp wird als *Bytenachricht* bezeichnet. Die Schnittstelle 'IBytesMessage' enthält die Methoden zur Verarbeitung des Hauptteils einer Bytenachricht. Weitere Informationen finden Sie im Abschnitt „Bytenachrichten” auf Seite 82.

Zuordnung
Der Hauptteil enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist. Eine Nachricht mit diesem Hauptteiltyp wird als *Zuordnungsnachricht* bezeichnet. Die Schnittstelle 'IMapMessage' enthält die Methoden zur Verarbeitung des Hauptteils einer Zuordnungsnachricht. Weitere Informationen finden Sie im Abschnitt „Zuordnungsnachrichten” auf Seite 83.

Objekt
Der Hauptteil enthält ein serialisiertes Java- oder .NET-Objekt. Eine Nachricht mit diesem Hauptteiltyp wird als *Objektnachricht* bezeichnet. Die Schnittstelle 'IObjectMessage' enthält die Methoden zur Verarbeitung des Hauptteils einer Objektnachricht. Weitere Informationen finden Sie im Abschnitt „Objektnachrichten” auf Seite 84.

Datenstrom
Der Hauptteil enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist. Eine Nachricht mit diesem Hauptteiltyp wird als *Datenstromnachricht* bezeichnet. Die Schnittstelle 'IStreamMessage' enthält die Methoden zur Verarbeitung des Hauptteils einer Datenstromnachricht. Weitere Informationen finden Sie im Abschnitt „Datenstromnachrichten” auf Seite 85.

Text
Der Hauptteil enthält eine Zeichenfolge. Eine Nachricht mit diesem Hauptteiltyp wird als *Textnachricht* bezeichnet. Die Schnittstelle 'ITextMessage' enthält die Methoden zur Verarbeitung des Hauptteils einer Textnachricht. Weitere Informationen finden Sie im Abschnitt „Textnachrichten” auf Seite 86.

Die Schnittstelle 'IMessage' ist das übergeordnete Element aller Nachrichtenobjekte und kann in Messaging-Funktionen zur Darstellung eines beliebigen XMS-Nachrichtentyps verwendet werden.

Informationen zur Größe sowie zu den Höchst- und Mindestwerten eines jeden Datentyps finden Sie im Abschnitt Tabelle 5 auf Seite 44.

Weitere Informationen zu den erforderlichen Datentypen für Elemente von Anwendungsdaten, die in den Hauptteil einer Nachricht geschrieben wurden, und zu den fünf Hauptteiltypen einer Nachricht finden Sie in den Unterabschnitten.

Zugehörige Verweise
Teile der Eine XMS -Nachricht
Eine XMS-Nachricht besteht aus einem Header, einer Gruppe von Eigenschaften und einem Hauptteil.

Headerfelder in Eine XMS -Nachricht
Damit die Eine XMS -Anwendung Nachrichten mit einer WebSphere JMS -Anwendung austauschen kann, enthält der Header der Eine XMS -Nachricht die JMS-Nachrichtenheaderfelder.

Eigenschaften der Eine XMS -Nachricht
XMS unterstützt drei Arten von Nachrichteneigenschaften: JMS-definierte Eigenschaften, von IBM definierte Eigenschaften und anwendungsdefinierte Eigenschaften.

Nachrichtenselektoren
Eine XMS-Anwendung verwendet Nachrichtenselektoren, um die Nachrichten auszuwählen, die sie empfangen möchte.

Zuordnung von XMS-Nachrichten zu IBM WebSphere MQ-Nachrichten
Die JMS-Header-Felder und -Eigenschaften von Eine XMS-Nachrichten werden Feldern in den Headerstrukturen einer IBM WebSphere MQ-Nachricht zugeordnet.

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Aus diesem Grund muss jedes Element der Anwendungsdaten, das von der Eine XMS -Anwendung in den Hauptteil einer Nachricht geschrieben wird, über einen der in Tabelle 24 auf Seite 81aufgelisteten Datentypen verfügen. Für jeden Datentyp ist in der Tabelle der kompatible Java-Datentyp angegeben. Mit den von XMS bereitgestellten Methoden können ausschließlich Anwendungsdatenelemente mit diesen Datentypen geschrieben werden.

Tabelle 24. XMS-Datentypen, die mit Java-Datentypen kompatibel sind

XMS-Datentyp	Beschreibung	Kompatibler Java-Datentyp
System.Boolean	Boolescher Wert 'true' oder 'false' (Wahr oder Falsch)	boolean
System.Char16	Doppelbyte-Zeichen	char
System.SByte	8-Bit-Ganzzahl mit Vorzeichen	Byte
System.Int16	16-Bit-Ganzzahl mit Vorzeichen	short

Tabelle 24. XMS-Datentypen, die mit Java-Datentypen kompatibel sind (Forts.)		
XMS-Datentyp	**Beschreibung**	**Kompatibler Java-Datentyp**
System.Int32	32-Bit-Ganzzahl mit Vorzeichen	Int
System.Int64	64-Bit-Ganzzahl mit Vorzeichen	long
System.Float	Gleitkommazahl mit Vorzeichen	float
System.Double	Gleitkommazahl mit doppelter Genauigkeit und Vorzeichen	double
System.String	Zeichenfolge	Zeichenfolge

Informationen zur Größe sowie zu den Höchst- und Mindestwerten eines jeden Datentyps finden Sie im Abschnitt „Primitive XMS-Datentypen” auf Seite 44.

Zugehörige Konzepte

Attribute und Eigenschaften von Objekten
Ein XMS-Objekt kann Attribute und Eigenschaften besitzen, bei denen es sich um Merkmale des Objekts handelt, die auf unterschiedliche Weise implementiert werden.

Primitive XMS-Datentypen
XMS stellt Datentypen bereit, die den acht primitiven Java-Datentypen entsprechen: 'byte', 'short', 'int', 'long', 'float', 'double', 'char' und 'boolean'. Dies ermöglicht den Austausch von Nachrichten zwischen XMS und JMS, ohne dass Daten verloren gehen oder beschädigt werden.

Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp
Wenn eine Anwendung den Wert einer Eigenschaft abruft, kann der Wert von XMS in einen anderen Datentyp konvertiert werden. Es gibt viele Regeln, mit denen gesteuert wird, welche Konvertierungen unterstützt werden und wie XMS die Konvertierungen ausführt.

Zugehörige Verweise

Bytenachrichten
Der Hauptteil einer Bytenachricht enthält einen Bytestrom. Der Hauptteil enthält nur die tatsächlichen Daten, wobei die sendende und empfangende Anwendung dafür zuständig sind, diese Daten zu interpretieren.

Zuordnungsnachrichten
Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Objektnachrichten
Der Hauptteil einer Objektnachricht enthält ein serialisiertes Java- oder .NET-Objekt.

Datenstromnachrichten
Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Textnachrichten
Der Hauptteil einer Textnachricht enthält eine Zeichenfolge.

Bytenachrichten

Der Hauptteil einer Bytenachricht enthält einen Bytestrom. Der Hauptteil enthält nur die tatsächlichen Daten, wobei die sendende und empfangende Anwendung dafür zuständig sind, diese Daten zu interpretieren.

Bytenachrichten sind nützlich, wenn Eine XMS-Anwendung Nachrichten mit Anwendungen austauschen muss, die nicht die XMS- oder JMS-Anwendungsprogrammierschnittstelle verwenden.

Nachdem eine Anwendung eine Bytenachricht erstellt hat, ist der Hauptteil der Nachricht zunächst lesegeschützt. Die Anwendung stellt die Anwendungsdaten im Hauptteil zusammen, indem Sie die entsprechenden Schreibmethoden der IBytesMessage-Schnittstelle für .NET aufruft. Jedes Mal, wenn die Anwendung einen weiteren Wert in den Datenstrom der Bytenachricht schreibt, wird der Wert unmittelbar

nach dem zuletzt von der Anwendung geschriebenen Wert angefügt. XMS verwaltet einen internen Cursor, mit dessen Hilfe die Position des zuletzt geschriebenen Bytewerts gespeichert wird.

Wenn die Anwendung die Nachricht sendet, wird der Hauptteil der Nachricht nun schreibgeschützt. In diesem Modus kann die Anwendung die Nachricht wiederholt senden.

Wenn eine Anwendung eine Bytenachricht empfängt, ist der Hauptteil der Nachricht schreibgeschützt. Mit den entsprechenden Lesemethoden der IBytesMessage-Schnittstelle kann die Anwendung den Inhalt des Datentstroms in der Bytenachricht lesen. Die Anwendung liest die Bytewerte der Reihenfolge entsprechend nacheinander, und XMS verwaltet einen internen Cursor, mit dessen Hilfe die Position des zuletzt gelesenen Bytewerts gespeichert wird.

Wenn eine Anwendung die Methode 'Reset' der IBytesMessage-Schnittstelle aufruft, während der Hauptteil einer Bytenachricht änderbar ist, wird der Hauptteil dadurch schreibgeschützt. Außerdem wird der Cursor durch diese Methode zurück an den Anfang des Bytenachrichten-Datenstroms gesetzt.

Wenn eine Anwendung die Methode `Clear Body` der IMessage-Schnittstelle für .NET aufruft, während der Hauptteil einer Bytenachricht schreibgeschützt ist, wird der Hauptteil dadurch änderbar. Außerdem löscht diese Methode den Inhalt des Hauptteils.

Zugehörige Verweise

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Zuordnungsnachrichten
Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Objektnachrichten
Der Hauptteil einer Objektnachricht enthält ein serialisiertes Java- oder .NET-Objekt.

Datenstromnachrichten
Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Textnachrichten
Der Hauptteil einer Textnachricht enthält eine Zeichenfolge.

IBytesMessage (für die .NET-Schnittstelle)
Eine Bytenachricht ist eine Nachricht, deren Hauptteil aus einem Bytestrom besteht.

Zuordnungsnachrichten

Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

In jedem Name/Wert-Paar ist der Name eine Zeichenfolge, die angibt, um welchen Wert es sich handelt, und der Wert ist ein Element der Anwendungsdaten, dem einer der XMS-Datentypen zugeordnet ist, die in Tabelle 24 auf Seite 81 aufgelistet sind. Die Reihenfolge der Name/Wert-Paare ist nicht definiert. Die Klasse 'MapMessage' enthält die Methoden zum Festlegen und Abrufen von Name/Wert-Paaren.

Eine Anwendung kann durch Angabe des Namens beliebig auf ein Name/Wert-Paar zugreifen.

Eine .NET-Anwendung kann mithilfe der Eigenschaft 'MapNames' eine Auflistung der Namen im Hauptteil einer Zuordnungsnachricht abrufen.

Wenn eine Anwendung den Wert eines Name/Wert-Paars abruft, kann der Wert durch XMS in einen anderen Datentyp konvertiert werden. Wenn beispielsweise ein Ganzzahl-Wert (Integer) aus dem Hauptteil einer Zuordnungsnachricht abgerufen werden soll, kann eine Anwendung die GetString-Methode der MapMessage-Klasse aufrufen, sodass die Ganzzahl als Zeichenfolge zurückgegeben wird. Dabei unterstützt XMS dieselben Konvertierungen wie bei der Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen. Weitere Informationen zu den unterstützen Konvertierungen finden Sie im Abschnitt „Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp” auf Seite 45.

Nachdem eine Anwendung eine Zuordnungsnachricht erstellt hat, ist der Hauptteil sowohl lesbar als auch änderbar, also weder lese- noch schreibgeschützt. Auch wenn die Anwendung die Nachricht gesendet hat, bleibt der Hauptteil lesbar und änderbar. Wenn eine Anwendung eine Zuordnungsnachricht empfängt, ist der Hauptteil der Nachricht zunächst schreibgeschützt. Wenn eine Anwendung die Methode 'Clear Body' aufruft, während der Hauptteil einer Zuordnungsnachricht schreibgeschützt ist, wird der Hauptteil dadurch wieder lesbar und änderbar. Außerdem löscht diese Methode den Inhalt des Hauptteils.

Zugehörige Konzepte

Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp
Wenn eine Anwendung den Wert einer Eigenschaft abruft, kann der Wert von XMS in einen anderen Datentyp konvertiert werden. Es gibt viele Regeln, mit denen gesteuert wird, welche Konvertierungen unterstützt werden und wie XMS die Konvertierungen ausführt.

Zugehörige Verweise

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Bytenachrichten
Der Hauptteil einer Bytenachricht enthält einen Bytestrom. Der Hauptteil enthält nur die tatsächlichen Daten, wobei die sendende und empfangende Anwendung dafür zuständig sind, diese Daten zu interpretieren.

Objektnachrichten
Der Hauptteil einer Objektnachricht enthält ein serialisiertes Java- oder .NET-Objekt.

Datenstromnachrichten
Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Textnachrichten
Der Hauptteil einer Textnachricht enthält eine Zeichenfolge.

IMapMessage (für die .NET-Schnittstelle)
Eine Zuordnungsnachricht ist eine Nachricht, deren Hauptteil aus einer Gruppe von Name/Wert-Paaren besteht, wobei jedem Wert ein Datentyp zugeordnet ist.

Objektnachrichten

Der Hauptteil einer Objektnachricht enthält ein serialisiertes Java- oder .NET-Objekt.

Eine XMS-Anwendung kann eine Objektnachricht empfangen, ihre Headerfelder und Eigenschaften ändern und sie anschließend an ein anderes Ziel senden. Außerdem kann eine Anwendung den Hauptteil einer Objektnachricht kopieren und zum Erstellen einer anderen Objektnachricht verwenden. XMS behandelt den Hauptteil einer Objektnachricht wie ein Byte-Array.

Nachdem eine Anwendung eine Objektnachricht erstellt hat, ist der Hauptteil der Nachricht sowohl lesbar als auch änderbar, also weder lese- noch schreibgeschützt. Auch wenn die Anwendung die Nachricht gesendet hat, bleibt der Hauptteil lesbar und änderbar. Wenn eine Anwendung eine Objektnachricht empfängt, ist der Hauptteil der Nachricht zunächst schreibgeschützt. Wenn eine Anwendung die Methode `Clear Body` der IMessage-Schnittstelle für .NET aufruft, wenn der Hauptteil einer Objektnachricht schreibgeschützt ist, wird der Hauptteil lesbar und beschreibbar. Außerdem löscht diese Methode den Inhalt des Hauptteils.

Zugehörige Verweise

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Bytenachrichten

Der Hauptteil einer Bytenachricht enthält einen Bytestrom. Der Hauptteil enthält nur die tatsächlichen Daten, wobei die sendende und empfangende Anwendung dafür zuständig sind, diese Daten zu interpretieren.

Zuordnungsnachrichten
Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Datenstromnachrichten
Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Textnachrichten
Der Hauptteil einer Textnachricht enthält eine Zeichenfolge.

IObjectMessage (für die .NET-Schnittstelle)
Eine Objektnachricht ist eine Nachricht, deren Hauptteil ein serialisiertes Java -oder .NET-Objekt umfasst.

Datenstromnachrichten

Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Der Datentyp eines Werts ist einer der XMS-Datentypen, die in Tabelle 24 auf Seite 81 aufgelistet sind.

Nachdem eine Anwendung eine Datenstromnachricht erstellt hat, ist der Hauptteil der Nachricht zunächst änderbar. Die Anwendung stellt die Anwendungsdaten im Hauptteil zusammen, indem Sie die entsprechenden Schreibmethoden der IStreamMessage-Schnittstelle für .NET aufruft. Jedes Mal, wenn die Anwendung einen weiteren Wert in den Nachrichtendatenstrom schreibt, wird der Wert und dessen Datentyp unmittelbar nach dem zuletzt von der Anwendung geschriebenen Wert angefügt. XMS verwaltet einen internen Cursor, mit dessen Hilfe die Position des zuletzt geschriebenen Werts gespeichert wird.

Wenn die Anwendung die Nachricht sendet, wird der Hauptteil der Nachricht nun schreibgeschützt. In diesem Modus kann die Anwendung die Nachricht mehrmals senden.

Wenn eine Anwendung eine Datenstromnachricht empfängt, ist der Hauptteil der Nachricht zunächst schreibgeschützt. Mit den entsprechenden Lesemethoden der IStreamMessage-Schnittstelle für .NET kann die Anwendung den Inhalt Nachrichtendatenstroms lesen. Die Anwendung liest die Werte der Reihenfolge entsprechend nacheinander, und XMS verwaltet einen internen Cursor, mit dessen Hilfe die Position des zuletzt gelesenen Werts gespeichert wird.

Wenn eine Anwendung den Wert eines Nachrichtendatenstroms abruft, kann der Wert durch XMS in einen anderen Datentyp konvertiert werden. Wenn beispielsweise ein Ganzzahl-Wert (Integer) aus dem Nachrichtendatenstrom abgerufen werden soll, kann eine Anwendung die ReadString-Methode aufrufen, sodass die Ganzzahl als Zeichenfolge zurückgegeben wird. Dabei unterstützt XMS dieselben Konvertierungen wie bei der Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen. Weitere Informationen zu den unterstützen Konvertierungen finden Sie im Abschnitt „Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp” auf Seite 45.

Wenn ein Fehler auftritt, während eine Anwendung einen Wert im Nachrichtendatenstrom zu lesen versucht, wird die Cursorposition nicht aktualisiert. Die Anwendung kann einen erneuten Fehler vermeiden, indem sie versucht, den Wert als einen anderen Datentyp zu lesen.

Wenn eine Anwendung die Methode 'Reset' der IStreamMessage-Schnittstelle für .NET aufruft, während der Hauptteil einer Datenstromnachricht lesegeschützt ist, wird der Hauptteil dadurch schreibgeschützt. Außerdem wird der Cursor durch diese Methode zurück an den Anfang des Nachrichtendatenstroms gesetzt.

Wenn eine Anwendung die Methode `Clear Body` der IMessage-Schnittstelle für .NET aufruft, wenn der Hauptteil einer Datenstromnachricht schreibgeschützt ist, wird der Hauptteil schreibgeschützt. Außerdem löscht diese Methode den Inhalt des Hauptteils.

Zugehörige Konzepte
Implizite Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen Datentyp

Wenn eine Anwendung den Wert einer Eigenschaft abruft, kann der Wert von XMS in einen anderen Datentyp konvertiert werden. Es gibt viele Regeln, mit denen gesteuert wird, welche Konvertierungen unterstützt werden und wie XMS die Konvertierungen ausführt.

Zugehörige Verweise

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Bytenachrichten
Der Hauptteil einer Bytenachricht enthält einen Bytestrom. Der Hauptteil enthält nur die tatsächlichen Daten, wobei die sendende und empfangende Anwendung dafür zuständig sind, diese Daten zu interpretieren.

Zuordnungsnachrichten
Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Objektnachrichten
Der Hauptteil einer Objektnachricht enthält ein serialisiertes Java- oder .NET-Objekt.

Textnachrichten
Der Hauptteil einer Textnachricht enthält eine Zeichenfolge.

IStreamMessage (für die .NET-Schnittstelle)
Eine Datenstromnachricht ist eine Nachricht, deren Hauptteil aus einem Strom von Werten besteht, wobei jedem Wert ein Datentyp zugeordnet ist. Der Inhalt des Hauptteils wird nacheinander geschrieben und gelesen.

Textnachrichten

Der Hauptteil einer Textnachricht enthält eine Zeichenfolge.

Nachdem eine Anwendung eine Textnachricht erstellt hat, ist der Hauptteil der Nachricht sowohl lesbar als auch änderbar, also weder lese- noch schreibgeschützt. Auch wenn die Anwendung die Nachricht gesendet hat, bleibt der Hauptteil lesbar und änderbar. Wenn eine Anwendung eine Textnachricht empfängt, ist der Hauptteil der Nachricht zunächst schreibgeschützt. Wenn eine Anwendung die Methode 'Clear Body' der IMessage-Schnittstelle für .NET aufruft, während der Hauptteil einer Textnachricht schreibgeschützt ist, wird der Hauptteil dadurch wieder lesbar und änderbar. Außerdem löscht diese Methode den Inhalt des Hauptteils.

Zugehörige Verweise

Datentypen für Anwendungsdatenelemente
Um sicherzustellen, dass eine XMS-Anwendung Nachrichten mit einer Anwendung für die IBM WebSphere MQ -Klassen für JMS austauschen kann, müssen beide Anwendungen die Anwendungsdaten im Hauptteil einer Nachricht auf gleiche Weise interpretieren können.

Bytenachrichten
Der Hauptteil einer Bytenachricht enthält einen Bytestrom. Der Hauptteil enthält nur die tatsächlichen Daten, wobei die sendende und empfangende Anwendung dafür zuständig sind, diese Daten zu interpretieren.

Zuordnungsnachrichten
Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Objektnachrichten
Der Hauptteil einer Objektnachricht enthält ein serialisiertes Java- oder .NET-Objekt.

Datenstromnachrichten
Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

ITextMessage (für die .NET-Schnittstelle)
Eine Textnachricht ist eine Nachricht, deren Hauptteil aus einer Zeichenfolge besteht.

Nachrichtenselektoren

Eine XMS-Anwendung verwendet Nachrichtenselektoren, um die Nachrichten auszuwählen, die sie empfangen möchte.

Nachdem eine Anwendung einen Nachrichtenkonsumenten erstellt hat, kann sie ihm einen Nachrichtenselektorausdruck zuordnen. Der Nachrichtenselektorausdruck gibt die Auswahlkriterien an.

Wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 herstellt, erfolgt die Nachrichtenauswahl auf Seiten des Warteschlangenmanagers. In diesem Fall nimmt XMS keine Auswahl vor und übermittelt lediglich die vom Warteschlangenmanager empfangene Nachricht. Auf diese Weise wird die Leistung verbessert.

Wenn eine Anwendung jedoch eine Verbindung zu IBM WebSphere MQ V6.0 und niedriger WebSphere Ereignisbrokeroder WebSphere Nachrichtenbrokerherstellt, bestimmt WebSphere Service Integration Bus XMS , ob jede eingehende Nachricht die Auswahlkriterien erfüllt. Wenn eine Nachricht die Auswahlkriterien erfüllt, stellt XMS die Nachricht dem Nachrichtenkonsumenten zu. Wenn eine Nachricht die Auswahlkriterien nicht erfüllt, stellt XMS die Nachricht nicht zu und bleibt in der Punkt-zu-Punkt-Domäne in der Warteschlange.

Eine Anwendung kann mehrere Nachrichtenkonsumenten erstellen und jedem einen eigenen Nachrichtenselektorausdruck zuordnen. Wenn eine eingehende Nachricht die Auswahlkriterien mehrerer Nachrichtenkonsumenten erfüllt, übermittelt XMS die Nachricht an jeden dieser Konsumenten.

Ein Nachrichtenselektorausdruck kann folgende Nachrichteneigenschaften referenzieren:

- JMS-definierte Eigenschaften
- IBM-definierte Eigenschaften
- Anwendungsdefinierte Eigenschaften

Darüber hinaus kann er folgende Nachrichtenheaderfelder referenzieren:

- JMSCorrelationID
- JMSDeliveryMode
- JMSMessageID
- JMSPriority
- JMSTimestamp
- JMSType

Ein Nachrichtenselektorausdruck kann jedoch keine Daten im Hauptteil einer Nachricht referenzieren.

Es folgt ein Beispiel für einen Nachrichtenselektorausdruck:

```
JMSPriority > 3 AND manufacturer = 'Jaguar' AND model in ('xj6','xj12')
```

XMS stellt eine Nachricht einem Nachrichtenkonsumenten mit diesem Nachrichtenselektorausdruck nur dann zu, wenn die Nachricht eine höhere Priorität als 3hat; eine anwendungsdefinierte Eigenschaft (Hersteller) mit dem Wert `Jaguar;` und eine andere anwendungsdefinierte Eigenschaft (Modell) mit dem Wert `xj6` oder `xj12.`

Für die Erstellung eines Nachrichtenselektorausdrucks in XMS gelten dieselben Syntaxregeln wie in den IBM WebSphere MQ -Klassen für JMS. Informationen zum Erstellen eines Nachrichtenselektorausdrucks finden Sie im Handbuch *WebSphere MQ Using Java*. Beachten Sie, dass in einem Nachrichtenselektorausdruck die Namen von JMS-definierten Eigenschaften die JMS-Namen sein müssen und die Namen von von IBM definierten Eigenschaften die IBM WebSphere MQ -Klassen für JMS-Namen sein müssen. Hingegen ist die Verwendung der XMS-Namen der Eigenschaften in einem Nachrichtenselektorausdruck nicht zulässig.

Zugehörige Verweise

Teile der Eine XMS -Nachricht
Eine XMS-Nachricht besteht aus einem Header, einer Gruppe von Eigenschaften und einem Hauptteil.

Headerfelder in Eine XMS -Nachricht
Damit die Eine XMS -Anwendung Nachrichten mit einer WebSphere JMS -Anwendung austauschen kann, enthält der Header der Eine XMS -Nachricht die JMS-Nachrichtenheaderfelder.

Eigenschaften der Eine XMS -Nachricht
XMS unterstützt drei Arten von Nachrichteneigenschaften: JMS-definierte Eigenschaften, von IBM definierte Eigenschaften und anwendungsdefinierte Eigenschaften.

Hauptteil der Eine XMS -Nachricht
Der Hauptteil einer Nachricht enthält Anwendungsdaten. Eine Nachricht kann jedoch durchaus keinen Rumpf bzw. Hauptteil haben und nur die Headerfelder und Eigenschaften enthalten.

Zuordnung von XMS-Nachrichten zu IBM WebSphere MQ-Nachrichten
Die JMS-Header-Felder und -Eigenschaften von Eine XMS-Nachrichten werden Feldern in den Headerstrukturen einer IBM WebSphere MQ-Nachricht zugeordnet.

Zuordnung von XMS-Nachrichten zu IBM WebSphere MQ-Nachrichten

Die JMS-Header-Felder und -Eigenschaften von Eine XMS-Nachrichten werden Feldern in den Headerstrukturen einer IBM WebSphere MQ-Nachricht zugeordnet.

Wenn Eine XMS-Anwendung mit einem IBM WebSphere MQ-Warteschlangenmanager verbunden ist, werden Nachrichten, die an den Warteschlangenmanager gesendet werden, auf dieselbe Weise IBM WebSphere MQ-Nachrichten zugeordnet, wie IBM WebSphere MQ -Klassen für JMS-Nachrichten unter ähnlichen Umständen IBM WebSphere MQ-Nachrichten zugeordnet werden.

Wenn die Eigenschaft XMSC_WMQ_TARGET_CLIENT eines Destination-Objekts auf den Wert 'XMSC_WMQ_TARGET_DEST_JMS' festgelegt wird, werden die JMS-Headerfelder und -Eigenschaften einer an dieses Ziel gesendeten Nachricht den Feldern in den MQMD- und MQRFH2-Headerstrukturen der IBM WebSphere MQ-Nachricht zugeordnet. Bei dieser Einstellung der Eigenschaft 'XMSC_WMQ_TARGET_CLIENT' wird davon ausgegangen, dass die Anwendung, die die Nachricht empfängt, einen MQRFH2-Header korrekt verarbeiten kann. Die empfangende Anwendung kann deshalb eine weitere XMS-Anwendung, eine IBM WebSphere MQ -Klassen für JMS-Anwendung oder eine native IBM WebSphere MQ-Anwendung sein, die so konzipiert ist, dass sie einen MQRFH2-Header verarbeiten kann.

Wenn die Eigenschaft 'XMSC_WMQ_TARGET_CLIENT' eines Destination-Objekts stattdessen auf 'XMSC_WMQ_TARGET_DEST_MQ' festgelegt wird, werden die JMS-Headerfelder und -Eigenschaften einer an dieses Ziel gesendeten Nachricht den Feldern in der MQMD-Headerstruktur der IBM WebSphere MQ-Nachricht zugeordnet. Die Nachricht enthält in diesem Fall keinen MQRFH2-Header, und alle JMS-Headerfelder und -Eigenschaften, die den Feldern in der MQMD-Headerstruktur nicht zugeordnet werden können, werden ignoriert. Die Anwendung, die die Nachricht empfängt, kann demnach eine native IBM WebSphere MQ-Nachricht sein, die nicht für die Verarbeitung eines MQRFH2-Headers ausgelegt ist.

Die von einem Warteschlangenmanager empfangenen IBM WebSphere MQ-Nachrichten werden XMS-Nachrichten in derselben Weise zugeordnet wie IBM WebSphere MQ-Nachrichten unter ähnlichen Umständen den Nachrichten für die IBM WebSphere MQ -Klassen für JMS zugeordnet werden.

Wenn eine eingehende IBM WebSphere MQ-Nachricht einen MQRFH2-Header aufweist, verfügt die sich ergebende XMS-Nachricht über einen Hauptteil, dessen Typ durch den Wert der Eigenschaft **Msd** bestimmt wird, die im Ordner `mcd` des MQRFH2-Headers enthalten ist. Wenn die Eigenschaft **Msd** nicht im MQRFH2-Header enthalten ist oder die IBM WebSphere MQ-Nachricht keinen MQRFH2-Header aufweist, verfügt die sich ergebende XMS-Nachricht über einen Hauptteil, dessen Typ durch den Wert des Feldes *`Format`* im MQMD-Header bestimmt wird. Wenn das Feld *`Format`* den Wert 'MQFMT_STRING' hat, ist die XMS-Nachricht eine Textnachricht. Andernfalls handelt es sich bei der XMS-Nachricht um eine Bytenachricht. Wenn die IBM WebSphere MQ-Nachricht keinen MQRFH2-Header hat, werden nur die JMS-Headerfelder und -Eigenschaften festgelegt, die aus den Feldern im MQMD-Header abgeleitet werden können.

Weitere Informationen zur Zuordnung von IBM WebSphere MQ -Klassen für JMS -Nachrichten zu IBM WebSphere MQ -Nachrichten enthält der Abschnitt *IBM WebSphere MQ Java*.

Zugehörige Verweise
Teile der Eine XMS -Nachricht

Eine XMS-Nachricht besteht aus einem Header, einer Gruppe von Eigenschaften und einem Hauptteil.

Headerfelder in Eine XMS -Nachricht
Damit die Eine XMS -Anwendung Nachrichten mit einer WebSphere JMS -Anwendung austauschen kann, enthält der Header der Eine XMS -Nachricht die JMS-Nachrichtenheaderfelder.

Eigenschaften der Eine XMS -Nachricht
XMS unterstützt drei Arten von Nachrichteneigenschaften: JMS-definierte Eigenschaften, von IBM definierte Eigenschaften und anwendungsdefinierte Eigenschaften.

Hauptteil der Eine XMS -Nachricht
Der Hauptteil einer Nachricht enthält Anwendungsdaten. Eine Nachricht kann jedoch durchaus keinen Rumpf bzw. Hauptteil haben und nur die Headerfelder und Eigenschaften enthalten.

Nachrichtenselektoren
Eine XMS-Anwendung verwendet Nachrichtenselektoren, um die Nachrichten auszuwählen, die sie empfangen möchte.

Über eine Anwendung für einen Message Service Client for .NET einen Nachrichtendeskriptor lesen und schreiben

Sie können auf alle Felder des Nachrichtendeskriptors (MQMD) einer IBM WebSphere MQ-Nachricht zugreifen, außer auf StrucId und Version; BackoutCount kann gelesen, aber nicht geschrieben werden. Diese Funktion ist nur verfügbar, wenn eine Verbindung zu einem IBM WebSphere MQ Warteschlangenmanager Version 6 und höher hergestellt wird. Sie wird durch die später beschriebenen Zieleigenschaften gesteuert.

Mithilfe der vom Message Service Client for .NET bereitgestellten Nachrichtenattribute können XMS-Anwendungen MQMD-Felder festlegen und auch IBM WebSphere MQ-Anwendungen steuern.

Bei Verwendung des Publish/Subscribe-Messagings gelten einige Einschränkungen. Beispielsweise werden MQMD-Felder wie 'MsgID' und 'CorrelId' ignoriert, wenn für sie ein Wert festgelegt wurde.

Die in diesem Abschnitt beschriebene Funktion ist für das Publish/Subscribe-Messaging nicht verfügbar, wenn eine Verbindung zu einem Warteschlangenmanager von IBM WebSphere MQ Version 6 hergestellt wird. Sie ist auch nicht verfügbar, wenn die Eigenschaft **PROVIDERVERSION** auf 6 gesetzt ist.

Zugriff auf IBM WebSphere MQ-Nachrichtendaten aus einer Message Service Client for .NET-Anwendung

Sie können auf die vollständigen IBM WebSphere MQ-Nachrichtendaten, einschließlich des MQRFH2-Headers (falls vorhanden) und aller anderen IBM WebSphere MQ-Header (falls vorhanden), innerhalb einer Message Service Client for .NET-Anwendung als dem Hauptteil einer JMSBytesMessage zugreifen.

Die in diesem Abschnitt beschriebene Funktion ist nur verfügbar, wenn eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager mit Version 7 oder höher hergestellt wird und sich der IBM WebSphere MQ-Messaging-Provider im normalen Modus befindet.

Zielobjekteigenschaften bestimmen, wie die XMS-Anwendung auf die gesamte IBM WebSphere MQ-Nachricht (einschließlich MQRFH2-Header, falls vorhanden) als dem Hauptteil einer JMSBytesMessage zugreift.

Fehlerbehebung

Dieser AbschnittKapitel enthält Informationen, die Ihnen bei der Erkennung und Behebung von Problemen bei der Verwendung von Message Service Client for .NET helfen sollen.

Dieser AbschnittKapitel enthält Informationen, die Ihnen bei der Problembestimmung für XMS-Anwendungen helfen, und er beschreibt, wie First Failure Data Capture (FFDC) und die Tracefunktion für .NET-Anwendungen konfiguriert werden.

Dieser AbschnittKapitel enthält folgende ThemenAbschnitte:

- „Tracekonfiguration für .NET-Anwendungen” auf Seite 90
- „FFDC-Konfiguration für .NET-Anwendungen” auf Seite 94

- „Tipps zur Fehlerbehebung” auf Seite 94

Tracekonfiguration für .NET-Anwendungen

Für XMS .NET-Anwendungen können Sie einen Trace aus einer Anwendungskonfigurationsdatei sowie aus den XMS-Umgebungsvariablen konfigurieren. Sie können die Komponenten auswählen, für die ein Trace erstellt werden soll. Traces werden normalerweise unter Anleitung des IBM Support durchgeführt.

Die Tracefunktion für XMS .NET basiert auf der Standardtraceinfrastruktur von .NET.

Standardmäßig ist sämtliche Traceerstellung bis auf die Fehlertraceerstellung inaktiviert. Sie können die Traceerstellung aktivieren und die Traceeinstellungen auf eine der folgenden Arten konfigurieren:

- Durch Verwendung einer Anwendungskonfigurationsdatei mit einem Namen, der aus dem Namen des ausführbaren Programms besteht, auf das sich die Datei bezieht, mit dem Suffix `.config`. Demnach hätte beispielsweise die Anwendungskonfigurationsdatei für das ausführbare Programm 'text.exe' den Namen 'text.exe.config'. Die Verwendung einer Anwendungskonfigurationsdatei ist die bevorzugte Methode für die Aktivierung der Tracefunktion für XMS .NET-Anwendungen. Weitere Informationen dazu finden Sie unter „Tracekonfiguration mit einer Anwendungskonfigurationsdatei” auf Seite 91.
- Durch Verwendung von XMS-Umgebungsvariablen wie für XMS C- oder C++-Anwendungen. Weitere Informationen dazu finden Sie unter „Tracekonfiguration mit XMS-Umgebungsvariablen” auf Seite 93.

Die aktive Tracedatei hat einen Namen im Format xms_trace <*PID*> .log, wobei <*PID*> für die Prozess-ID der Anwendung steht. Die Größe der aktiven Tracedatei ist standardmäßig auf 20 MB begrenzt. Wenn dieser Grenzwert erreicht ist, wird die Datei umbenannt und archiviert. Archivierte Dateien haben Namen im Format xms_trace <*PID*> *_YY.MM.DD_HH.MM.SS*.log

Standardmäßig werden vier Tracedateien aufbewahrt, d. h., eine aktive Datei und drei archivierte Dateien. Diese vier Dateien werden so lange als Rollpuffer verwendet, bis die Anwendung gestoppt wird, wobei die älteste Datei entfernt und durch die neueste Datei ersetzt wird. Sie können die Anzahl der Tracedateien ändern, indem Sie in der Anwendungskonfigurationsdatei eine andere Zahl angeben. Es müssen jedoch mindestens zwei Dateien vorhanden sein (eine aktive Datei und eine archivierte Datei).

Es sind zwei Tracedateiformate verfügbar:

- Tracedateien im Format 'basic' sind in einem WebSphere Application Server-Format lesbar. Dieses Format ist das Standardformat für Tracedateien. Das Format 'basic' ist nicht mit Traceanalysetool kompatibel.
- Tracedateien im Format 'advanced' sind mit Traceanalysetools kompatibel. Sie müssen in der Anwendungskonfigurationsdatei angeben, dass Sie Tracedateien im Format 'advanced' erstellen möchten.

Traceeinträge enthalten die folgenden Informationen:

- Datum und Zeitpunkt, an dem der Trace protokolliert wurde
- Der Klassenname
- Der Tracetyp
- Die Tracenachricht

Das folgende Beispiel zeigt einen Auszug aus einem Trace:

```
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    >    Allocate    Entry
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    >    Initialize    Entry
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    <    Initialize    Exit
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    <    Allocate    Exit
```

Im vorstehenden Beispiel lautet das Format wie folgt:

```
[Date Time:Microsecs]         Thread-id    Classname        Trace-type    Methodname    Entry
or Exit
```

Dabei steht `Trace-type` für:

> für Eintritt

< für Austritt
d für Debuginformationen

Tracekonfiguration mit einer Anwendungskonfigurationsdatei

Die bevorzugte Methode für die Konfiguration der Traceerstellung für XMS .NET-Anwendungen ist die Anwendungskonfigurationsdatei. Der Abschnitt 'Trace' dieser Datei enthält Parameter, die Folgendes definieren: wofür ein Trace erstellt werden soll, die Position und die maximal zulässige Größe der Tracedatei, die Anzahl verwendeter Tracedateien und das Format der Tracedatei.

Sie aktivieren den Trace mithilfe der Anwendungskonfigurationsdatei, indem Sie die Datei in dasselbe Verzeichnis stellen wie die ausführbare Datei für Ihre Anwendung.

Die Traceerstellung kann sowohl von der Komponente als auch vom Tracetyp aktiviert werden. Es ist auch möglich, die Traceerstellung für eine gesamte Tracegruppe zu aktivieren. Sie können die Traceerstellung für Komponenten in einer Hierarchie entweder einzeln oder insgesamt aktivieren. Es sind folgende Tracetypen verfügbar:

- Debug-Trace
- Ausnahmebedingungstrace
- Warnungen, Informationsnachrichten und Fehlernachrichten
- Methodeneintritts- und Methodenaustrittstrace

Das folgende Beispiel zeigt die Traceeinstellungen, die im Abschnitt 'Trace' einer Anwendungskonfigurationsdatei definiert sind:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <configSections>
        <sectionGroup name="IBM.XMS">
            <section name="Trace"
                     type="System.Configuration.SingleTagSectionHandler" />
        </sectionGroup>
    </configSections>

    <IBM.XMS>
        <Trace traceSpecification="*=all=enabled" traceFilePath=""
               traceFileSize="20000000" traceFileNumber="3"
               traceFormat="advanced" />
    </IBM.XMS>
 </configuration>
```

In Tabelle 25 auf Seite 92 sind die Parametereinstellungen im Detail beschrieben.

Tabelle 25. Traceparametereinstellungen der Anwendungskonfigurationsdatei

Parameter	Beschreibung
`traceSpecification=<ComponentName>=<type>=<state>`	`<ComponentName>` ist der Name der Klasse, für die ein Trace erstellt werden soll. In diesem Namen können Sie ein Platzhalterzeichen (*) verwenden. Beispiel: `*=all=enabled` gibt an, dass Sie für alle Klassen einen Trace erstellen möchten, und `IBM.XMS.impl.*=all=enabled` gibt an, dass Sie nur einen API-Trace anfordern. `<type>` kann einer der folgenden Tracetypen sein: • alle • Debug • Ereignis • EntryExit `<state>` kann aktiviert oder inaktiviert sein. Sie können mehrere Traceelemente verbinden, indem Sie einen Begrenzer ':' (Doppelpunkt) verwenden.
`traceFilePath="<filename>"`	Wenn Sie die Variable 'traceFilePath' nicht angeben oder wenn 'traceFilePath' vorhanden ist, aber eine leere Zeichenfolge enthält, wird die Tracedatei im aktuellen Verzeichnis gespeichert. Zum Speichern der Tracedatei in einem benannten Verzeichnis geben Sie den Verzeichnisnamen in 'traceFilePath' an; Beispiel: `traceFilePath="c:\somepath"`
`traceFileSize="<size>"`	Die maximal zulässige Größe der Tracedatei. Wenn eine Datei diese Größe erreicht, wird sie archiviert und umbenannt. Das Standardmaximum ist 20 KB, das als `traceFileSize="20000000"` angegeben wird.
`traceFileNumber="<number>"`	Die Anzahl der Tracedateien, die aufbewahrt werden sollen. Der Standardwert ist 4 (eine aktive Datei und 3 Archivdateien). Die zulässige Mindestanzahl ist 2.
`traceFormat="<format>"`	Das Standard-Traceformat ist 'basic'. Tracedateien werden in diesem Format erzeugt, wenn Sie `traceFormat="basic"` angeben oder wenn Sie kein traceFormat angeben oder wenn traceFormat vorhanden ist, aber eine leere Zeichenfolge enthält. Wenn Sie einen Trace benötigen, der mit Traceanalysetools kompatibel ist, müssen Sie `traceFormat="advanced"` angeben.

Die Traceeinstellungen in der Anwendungskonfigurationsdatei sind dynamisch und werden jedes Mal erneut gelesen, wenn die Datei gespeichert oder ersetzt wird. Wenn in der Datei nach der Bearbeitung Fehler gefunden werden, werden die Tracedateieinstellungen auf ihre Standardwerte zurückgesetzt.

Zugehörige Konzepte

Tracekonfiguration mit XMS-Umgebungsvariablen

Alternativ zur Verwendung einer Anwendungskonfigurationsdatei können Sie die Traceerstellung mithilfe von XMS-Umgebungsvariablen aktivieren. Diese Umgebungsvariablen werden nur verwendet, wenn in der Anwendungskonfigurationsdatei keine Tracespezifikation enthalten ist.

Tracekonfiguration mit XMS-Umgebungsvariablen

Alternativ zur Verwendung einer Anwendungskonfigurationsdatei können Sie die Traceerstellung mithilfe von XMS-Umgebungsvariablen aktivieren. Diese Umgebungsvariablen werden nur verwendet, wenn in der Anwendungskonfigurationsdatei keine Tracespezifikation enthalten ist.

Um die Traceerstellung für eine XMS .NET-Anwendung zu konfigurieren, legen Sie die folgenden Umgebungsvariablen fest, bevor Sie die Anwendung ausführen:

Tabelle 26. Umgebungsvariableneinstellungen für .NET-Trace

Umgebungsvariablen	**Standard**	**Einstellung**	**Bedeutet**
XMS_TRACE_ON	Nicht zutreffend	Nicht zutreffend: Der Wert dieser Variablen wird ignoriert	Wenn XMS_TRACE_ON festgelegt ist, ist standardmäßig 'all trace' aktiviert.
XMS_TRACE_FILE_PATH	Aktuelles Arbeitsverzeichnis	/dirpath/	Der Verzeichnispfad, in den Trace- und FFDC-Datensätze geschrieben werden. XMS erstellt FFDC- und Tracedateien im aktuellen Arbeitsverzeichnis, es sei denn, Sie geben eine alternative Position an. Sie können eine alternative Position angeben, indem Sie die Umgebungsvariable 'XMS_TRACE_FILE_PATH' auf den vollständig qualifizierten Pfadnamen des Verzeichnisses setzen, in dem XMS die FFDC- und Tracedateien erstellen soll. Sie müssen diese Umgebungsvariable festlegen, bevor Sie die Anwendung starten, für die Sie einen Trace erstellen möchten. Sie müssen sicherstellen, dass die Benutzer-ID, unter der die Anwendung ausgeführt wird, über die Berechtigung zum Schreiben in das Verzeichnis verfügt, in dem XMS die FFDC- und Tracedateien erstellt.

Tabelle 26. Umgebungsvariableneinstellungen für .NET-Trace (Forts.)			
Umgebungsvariablen	**Standard**	**Einstellung**	**Bedeutet**
XMS_TRACE_FORMAT	BASIC	BASIC, ADVANCED	Gibt das erforderliche Traceformat an, das entweder BASIC oder ADVANCED sein kann. Das Standardformat ist BASIC. Das ADVANCED-Format ist mit Traceanalysetools kompatibel.
XMS_TRACE_SPECIFICATION	Nicht zutreffend	Siehe „Tracekonfiguration mit einer Anwendungskonfigurationsdatei" auf Seite 91	Überschreibt die Tracespezifikation, die das in „Tracekonfiguration mit einer Anwendungskonfigurationsdatei" auf Seite 91 angegebene Format einhält.

Zugehörige Konzepte

Tracekonfiguration mit einer Anwendungskonfigurationsdatei

Die bevorzugte Methode für die Konfiguration der Traceerstellung für XMS .NET-Anwendungen ist die Anwendungskonfigurationsdatei. Der Abschnitt 'Trace' dieser Datei enthält Parameter, die Folgendes definieren: wofür ein Trace erstellt werden soll, die Position und die maximal zulässige Größe der Tracedatei, die Anzahl verwendeter Tracedateien und das Format der Tracedatei.

FFDC-Konfiguration für .NET-Anwendungen

Für die .NET-Implementierung von XMS wird für jede FFDC eine FFDC-Datei erstellt.

First-Failure Data Capture-Dateien (FFDC-Dateien) werden in lesbaren Textdateien gespeichert. Das Namensformat für diese Dateien ist `xmsffdc<processID>_<Date>T<Timestamp>.txt`. Ein Beispiel für einen Dateinamen ist `xmsffdc264_2006.01.06T13.18.52.990955.txt`. Die Zeitmarke enthält Mikrosekundenauflösung.

Die Dateien beginnen mit dem Datum und dem Zeitpunkt, an dem die Ausnahmebedingung auftrat, gefolgt vom Ausnahmetyp. Die Dateien enthalten eine eindeutige kurze Prüf-ID, die verwendet werden kann, um die Position zu finden, an der die FFDC auftrat.

Sie müssen keine Konfiguration durchführen, um FFDC einschalten zu können. Standardmäßig werden alle FFDC-Dateien in das aktuelle Verzeichnis geschrieben. Falls erforderlich, können Sie jedoch ein anderes Verzeichnis angeben, indem Sie ffdcDirectory im Abschnitt 'Trace' in der Anwendungskonfigurationsdatei ändern. Im folgenden Beispiel werden alle Tracedateien im Verzeichnis c:client\ffdc protokolliert:

```
<IBM.XMS>
       <Trace ffdc=true ffdcDirectory="c:\client\ffdc"/>
</IBM.XMS>
```

Sie können die Tracefunktion inaktivieren, indem Sie FFDC im Abschnitt 'Trace' der Anwendungskonfigurationsdatei auf 'false' setzen.

Wenn Sie keine Anwendungskonfigurationsdatei verwenden, ist FFDC aktiviert und die Tracefunktion inaktiviert.

Tipps zur Fehlerbehebung

Verwenden Sie diese Tipps, um Fehler bei der Verwendung von XMS zu beheben.

Eine XMS-Anwendung kann keine Verbindung zu einem Warteschlangenmanager herstellen (MQRC_NOT_AUTHORIZED)

Das Verhalten des XMS .NET-Clients unterscheidet sich vom Verhalten des IBM WebSphere MQ JMS-Clients. Deshalb ist es möglich, dass Ihre XMS-Anwendung keine Verbindung zu Ihrem Warteschlangenmanager herstellen kann, obwohl Ihre JMS-Anwendung dies kann.

- Eine einfache Lösung für dieses Problem besteht in der Verwendung einer Benutzer-ID, die nicht mehr als 12 Zeichen lang ist und vollständig in der Berechtigungsliste des Warteschlangenmanagers autorisiert ist. Wenn diese Lösung nicht ideal ist, wäre ein anderer, aber komplexerer Ansatz die Verwendung von Sicherheitsexits. Wenn Sie weitere Hilfe zu diesem Problem benötigen, wenden Sie sich zur Unterstützung an den IBM Support.
- Wenn Sie die Eigenschaft XMSC_USERID der Verbindungsfactory festlegen, muss sie mit der Benutzer-ID und dem Kennwort des angemeldeten Benutzers übereinstimmen. Wenn Sie diese Eigenschaft nicht festlegen, verwendet der Warteschlangenmanager standardmäßig die Benutzer-ID des angemeldeten Benutzers.
- Die Benutzerauthentifizierung für IBM WebSphere MQ wird ausgeführt, indem die Details des derzeit angemeldeten Benutzers verwendet werden und nicht die in den Feldern XMSC.USERID und XMSC.PASSWORD angegebenen Informationen. Dies ist darauf ausgelegt, die Konsistenz mit IBM WebSphere MQ beizubehalten. Weitere Informationen zur Authentifizierung finden Sie im Abschnitt *Authentifizierungsinformationen* in der Online-Produktdokumentation zu IBM IBM WebSphere MQ.

Verbindung wird zur Messaging-Engine umgeleitet

Wenn Sie eine Verbindung zu einem Service Integration Bus von WebSphere Application Server Version 6.0.2 herstellen, können alle Verbindungen vom ursprünglichen Providerendpunkt an die Messaging-Engine umgeleitet werden, die der Bus für diese Clientverbindung auswählt. Dabei wird die Verbindung immer an einen Host-Server umgeleitet, der durch den Hostnamen statt durch eine IP-Adresse angegeben wird. Deshalb können Verbindungsprobleme auftreten, wenn der Hostname nicht aufgelöst werden kann.

Um erfolgreich eine Verbindung zum Service Integration Bus von WebSphere Application Server Version 6.0.2 herzustellen, müssen Sie eine Zuordnung zwischen den Hostnamen und IP-Adressen auf Ihrer Client-Host-Maschine bereitstellen. Sie können die Zuordnung beispielsweise in einer lokalen Hosttabelle auf Ihrer Client-Hostmaschine angeben.

Eine XMS -Anwendung, die einen größeren JVM-Heapspeicher verwendet

Die XMS .NET-Anwendung, die Nachrichten über die WebSphere Application Server -Messaging-Engines sendet, muss normalerweise einen größeren JVM-Heapspeicher als den angegebenen Standard verwenden. Informationen zum Ändern der Konfigurationseinstellungen für den Heapspeicher finden Sie unter Messaging-Leistung mit Serviceintegrationstechnologien optimieren in der Produktdokumentation zu WebSphere Application Server Version 7.

Unterstützung für Telnet-ähnliche Kennwortauthentifizierung

Das XMS .NET Real Time Transport-Protokoll unterstützt nur eine einfache Telnet-ähnliche Kennwortauthentifizierung. Das XMS .NET Real Time Transport-Protokoll unterstützt nicht das Datenschutzniveau.

Werte für Eigenschaftstyp 'double' festlegen

Auf einer Windows-64-Bit-Plattform funktionieren die Methoden SetDoubleProperty () oder GetDoubleProperty () möglicherweise nicht ordnungsgemäß, wenn Werte für den Eigenschaftstyp double festgelegt oder abgerufen werden, wenn die Werte kleiner als Double.Epsilonsind.

Wenn Sie z. B. den Wert "4.9E-324" für eine Eigenschaft vom Typ "double" festlegen, wird er von Windows-64-Bit-Plattformen als "0.0" behandelt. Wenn also JMS oder eine andere Anwendung in einer verteilten Messaging-Umgebung den Wert der Eigenschaft vom Typ "double" auf einem UNIX- oder Windows-32-Bit-System mit 4.9E-324 festlegt und XMS .NET auf einem 64-Bit-System ausgeführt wird,

gibt "GetDoubleProperty()" den Wert "0.0" zurück. Dies ist ein bekanntes Problem in Microsoft .NET 2.0 Framework.

Message Service Clients for .NET - Referenz

Dieser Referenzabschnitt enthält Informationen, die Ihnen bei der Verwendung von Message Service Client for .NET helfen sollen. Die Informationen unterstützen Sie bei der Ausführung der Aufgaben in Verbindung mit der Programmierung mit XMS.

.NET-Schnittstellen

In diesem Thema werden die Schnittstellen der .NET-Klasse und die zugehörigen Eigenschaften und Methoden dokumentiert.

In der folgenden Tabelle werden alle Schnittstellen zusammengefasst, die im Namespace `IBM.XMS` definiert sind.

Tabelle 27. Zusammenfassung der Schnittstellen der .NET-Klasse

Schnittstelle	Beschreibung
„IBytesMessage” auf Seite 99	Eine Bytenachricht ist eine Nachricht, deren Hauptteil aus einem Bytestrom besteht.
„IConnection” auf Seite 109	Ein Connection-Objekt stellt die aktive Verbindung der Anwendung zu einem Messaging-Server dar.
„IConnectionFactory” auf Seite 113	Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.
„IConnectionMetaData” auf Seite 115	Ein ConnectionMetaData-Objekt stellt Informationen zu einer Verbindung bereit.
„IDestination” auf Seite 116	Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.
„ExceptionListener” auf Seite 117	Eine Anwendung verwendet einen Listener für Ausnahmebedingungen, um asynchron über ein Problem mit einer Verbindung benachrichtigt zu werden.
„IllegalStateException” auf Seite 118	XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung eine Methode zu einem falschen oder ungeeigneten Zeitpunkt aufruft oder wenn XMS kein geeigneter Status für die angeforderte Operation ist.
„InitialContext” auf Seite 118	Eine Anwendung verwendet ein InitialContext-Objekt, um Objekte aus Objektdefinitionen zu erstellen, die aus einem Repository mit verwalteten Objekten abgerufen werden.
„InvalidClientIDException” auf Seite 121	XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung eine Client-ID für eine Verbindung festlegen möchte, die Client-ID aber nicht gültig ist oder bereits verwendet wird.
„InvalidDestinationException” auf Seite 121	XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung ein Ziel angibt, das nicht gültig ist.

Tabelle 27. Zusammenfassung der Schnittstellen der .NET-Klasse (Forts.)	
Schnittstelle	**Beschreibung**
„InvalidSelectorException” auf Seite 121	XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung einen Ausdruck für einen Nachrichtenselektor bereitstellt, dessen Syntax nicht gültig ist.
„IMapMessage” auf Seite 122	Eine Zuordnungsnachricht ist eine Nachricht, deren Hauptteil aus einer Gruppe von Name/Wert-Paaren besteht, wobei jedem Wert ein Datentyp zugeordnet ist.
„IMessage” auf Seite 131	Ein Nachrichtenobjekt stellt eine Nachricht dar, die von einer Anwendung gesendet oder empfangen wird. IMessage ist eine Superklasse für die Nachrichtenklassen, wie beispielsweise IMapMessage.
„IMessageConsumer” auf Seite 138	Eine Anwendung verwendet einen Nachrichtenkonsumenten (Message Consumer), um Nachrichten von einem Ziel zu empfangen.
„MessageEOFException” auf Seite 141	XMS gibt diese Ausnahmebedingung aus, wenn XMS beim Lesen des Hauptteils der Bytenachrichten durch die Anwendung das Ende eines Bytenachrichtendatenstroms ermittelt.
„MessageFormatException” auf Seite 141	XMS gibt diese Ausnahmebedingung aus, wenn XMS eine Nachricht mit einem ungültigen Format ermittelt.
„IMessageListener (Delegat)” auf Seite 141	Eine Anwendung verwendet einen Nachrichtenlistener, um Nachrichten asynchron zu empfangen.
„MessageNotReadableException” auf Seite 142	XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung den Hauptteil einer Nachricht lesen will, die lesegeschützt ist.
„MessageNotWritableException” auf Seite 142	XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung in den Hauptteil einer Nachricht schreiben will, die schreibgeschützt ist.
„IMessageProducer” auf Seite 142	Eine Anwendung verwendet einen Nachrichtenproduzenten (Message Producer), um Nachrichten an ein Ziel zu senden.
„IObjectMessage” auf Seite 149	Eine Objektnachricht ist eine Nachricht, deren Hauptteil ein serialisiertes Java -oder .NET-Objekt umfasst.
„IPropertyContext” auf Seite 150	IPropertyContext ist eine abstrakte Superklasse, die Methoden enthält, die Eigenschaften abrufen und festlegen. Diese Methoden werden von anderen Klassen geerbt.
„IQueueBrowser” auf Seite 159	Eine Anwendung verwendet einen Warteschlangenbrowser, um Nachrichten in einer Warteschlange anzuzeigen, ohne sie zu entfernen.

Tabelle 27. Zusammenfassung der Schnittstellen der .NET-Klasse (Forts.)	
Schnittstelle	**Beschreibung**
„Requestor” auf Seite 161	Eine Anwendung verwendet einen Requestor (Anforderer), um eine Anforderungsnachricht zu senden und dann auf die Antwort zu warten und sie zu empfangen.
„ResourceAllocationException” auf Seite 163	XMS gibt diese Ausnahmebedingung aus, wenn XMS die für eine Methode erforderlichen Ressourcen nicht zuordnen kann.
„SecurityException” auf Seite 163	XMS löst diese Ausnahme aus, wenn die Benutzer-ID und das Kennwort, die zur Authentifizierung einer Anwendung übergeben wurden, zurückgewiesen werden. XMS löst diese Ausnahme auch aus, wenn eine Berechtigungsprüfung fehlschlägt und verhindert, dass eine Methode abgeschlossen wird.
„ISession” auf Seite 164	Eine Sitzung (Session) ist ein Einzelthreadkontext zum Senden und Empfangen von Nachrichten.
„IStreamMessage” auf Seite 175	Eine Datenstromnachricht ist eine Nachricht, deren Hauptteil aus einem Strom von Werten besteht, wobei jedem Wert ein Datentyp zugeordnet ist.
„ITextMessage” auf Seite 184	Eine Textnachricht ist eine Nachricht, deren Hauptteil aus einer Zeichenfolge besteht.
„TransactionInProgressException” auf Seite 185	XMS löst diese Ausnahme aus, wenn eine Anwendung eine Operation anfordert, die nicht gültig ist, weil eine Transaktion in Bearbeitung ist.
„TransactionRolledBackException” auf Seite 186	XMS löst diese Ausnahme aus, wenn eine Anwendung die Methode Session.commit() aufruft, um die aktuelle Transaktion festzuschreiben, die Transaktion dann aber rückgängig gemacht wird.
XMSC	Für .NET werden Eigenschaftsnamen und -werte von XMS in dieser Klasse als öffentliche Konstanten definiert. Weitere Informationen finden Sie im Abschnitt „Eigenschaften von XMS-Objekten” auf Seite 189.
„XMSException” auf Seite 186	Wenn XMS beim Verarbeiten eines Aufrufs an eine .NET-Methode einen Fehler erkennt, löst XMS eine Ausnahmebedingung aus. Eine Ausnahme ist ein Objekt, das Informationen über den Fehler enthält. Es gibt verschiedene Typen von XMS-Ausnahmebedingungen, und bei einem XMSException-Objekt handelt es sich um einen dieser Typen. Allerdings ist die Klasse XMSException eine Superklasse für die anderen XMS-Ausnahmeklassen. XMS löst ein XMSException-Objekt in Situationen aus, in denen keiner der anderen Ausnahmetypen geeignet ist.

Tabelle 27. Zusammenfassung der Schnittstellen der .NET-Klasse (Forts.)	
Schnittstelle	**Beschreibung**
„XMSFactoryFactory” auf Seite 187	Wenn eine Anwendung keine verwalteten Objekte verwendet, können Sie mithilfe dieser Klasse Verbindungsfactorys, Warteschlangen und Themen erstellen.

In der Definition jeder Methode werden die Ausnahmecodes aufgelistet, die XMS gegebenenfalls zurückgibt, wenn es bei der Verarbeitung eines Aufrufs der Methode einen Fehler erkennt. Jeder Ausnahmecode wird durch seine benannte Konstante dargestellt, der eine entsprechende Ausnahme zugehörig ist.

Zugehörige Konzepte

Eigene Anwendung erstellen
Sie können Ihre eigenen Anwendungen auf die gleiche Weise erstellen, wie Sie die Beispielanwendungen erstellen.

XMS-Anwendungen schreiben
Dieser AbschnittKapitel enthält Informationen, die Ihnen beim Schreiben von XMS-Anwendungen helfen.

XMS .NET-Anwendungen schreiben
Dieser AbschnittKapitel enthält Informationen, die Ihnen beim Schreiben von XMS.NET-Anwendungen helfen.

Zugehörige Verweise

Eigenschaften von XMS-Objekten
In diesem AbschnittKapitel werden die mit XMS definierten Objekteigenschaften dokumentiert.

IBytesMessage

Eine Bytenachricht ist eine Nachricht, deren Hauptteil aus einem Bytestrom besteht.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IBytesMessage
```

Zugehörige Verweise

Bytenachrichten
Der Hauptteil einer Bytenachricht enthält einen Bytestrom. Der Hauptteil enthält nur die tatsächlichen Daten, wobei die sendende und empfangende Anwendung dafür zuständig sind, diese Daten zu interpretieren.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
BodyLength	Abrufen der Länge des Nachrichtenhauptteils in Bytes, wenn der Nachrichtenhauptteil schreibgeschützt ist.

BodyLength - Hauptteillänge abrufen

Schnittstelle:

```
Int64 BodyLength
    {
      get;
    }
```

Abrufen der Länge des Nachrichtenhauptteils in Bytes, wenn der Nachrichtenhauptteil schreibgeschützt ist.

Der zurückgegebene Wert ist die Länge des gesamten Hauptteils, unabhängig davon, an welcher Stelle der Cursor zum Lesen der Nachricht aktuell positioniert ist.

Exceptions:

- XMSException
- MessageNotReadableException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
ReadBoolean	Lesen eines booleschen Werts aus dem Datenstrom der Bytenachricht.
ReadSignedByte	Lesen des nächsten Bytes aus dem Datenstrom der Bytenachricht als eine 8-Bit-Ganzzahl mit Vorzeichen.
ReadBytes	Lesen eines Byte-Arrays aus dem Datenstrom der Bytenachricht ab der aktuellen Position des Cursors.
ReadChar	Lesen der nächsten 2 Bytes aus dem Datenstrom der Bytenachricht als ein Zeichen.
ReadDouble	Lesen der nächsten 8 Bytes aus dem Datenstrom der Bytenachricht als eine Gleitkommazahl mit doppelter Genauigkeit.
ReadFloat	Lesen der nächsten 4 Bytes aus dem Datenstrom der Bytenachricht als eine Gleitkommazahl.
ReadInt	Lesen der nächsten 4 Bytes aus dem Datenstrom der Bytenachricht als eine 32-Bit-Ganzzahl mit Vorzeichen.
ReadLong	Lesen der nächsten 8 Bytes aus dem Datenstrom der Bytenachricht als eine 64-Bit-Ganzzahl mit Vorzeichen.
ReadShort	Lesen der nächsten 2 Bytes aus dem Datenstrom der Bytenachricht als eine 16-Bit-Ganzzahl mit Vorzeichen.
ReadByte	Lesen des nächsten Bytes aus dem Datenstrom der Bytenachricht als eine 8-Bit-Ganzzahl ohne Vorzeichen.
ReadUnsignedShort	Lesen der nächsten 2 Bytes aus dem Datenstrom der Bytenachricht als eine 16-Bit-Ganzzahl ohne Vorzeichen.
ReadUTF	Lesen einer Zeichenfolge, die in UTF-8 codiert ist, aus dem Datenstrom der Bytenachricht.
Reset	Zurücksetzen des Hauptteils der Nachricht in den Lesezugriffsmodus und erneutes Positionieren des Cursors auf den Anfang des Datenstroms der Bytenachricht.
WriteBoolean	Schreiben eines booleschen Werts in den Datenstrom der Bytenachricht.
WriteByte	Schreiben eines Bytes in den Datenstrom der Bytenachricht.
WriteBytes	Schreiben eines Byte-Arrays in den Datenstrom der Bytenachricht.
WriteBytes	Schreiben eines partiellen Byte-Arrays in den Datenstrom der Bytenachricht, so wie durch die angegebene Länge definiert.
WriteChar	Schreiben eines Zeichens in den Datenstrom der Bytenachricht als 2 Bytes, das höchstwertige Byte zuerst.

Methode	Beschreibung
WriteDouble	Konvertieren einer Gleitkommazahl mit doppelter Genauigkeit in eine lange Ganzzahl und Schreiben der langen Ganzzahl in den Datenstrom der Bytenachricht als 8 Bytes, das höchstwertige Byte zuerst.
WriteFloat	Konvertieren einer Gleitkommazahl in eine Ganzzahl und Schreiben der Ganzzahl in den Datenstrom der Bytenachricht als 4 Bytes, das höchstwertige Byte zuerst.
WriteInt	Schreiben einer Ganzzahl in den Datenstrom der Bytenachricht als 4 Bytes, das höchstwertige Byte zuerst.
WriteLong	Schreiben einer langen Ganzzahl in den Datenstrom der Bytenachricht als 8 Bytes, das höchstwertige Byte zuerst.
WriteObject	Schreiben des angegebenen Objekts in den Datenstrom der Bytenachricht.
WriteShort	Schreiben einer kurzen Ganzzahl in den Datenstrom der Bytenachricht als 2 Bytes, das höchstwertige Byte zuerst.
WriteUTF	Schreiben einer Zeichenfolge, die in UTF-8 codiert ist, in den Datenstrom der Bytenachricht.

ReadBoolean - Booleschen Wert lesen

Schnittstelle:

```
Boolean ReadBoolean();
```

Lesen eines booleschen Werts aus dem Datenstrom der Bytenachricht.

Parameter:
--

Rückgabe:
Der boolesche Wert, der gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadSignedByte - Byte lesen

Schnittstelle:

```
Int16   ReadSignedByte();
```

Lesen des nächsten Bytes aus dem Datenstrom der Bytenachricht als eine 8-Bit-Ganzzahl mit Vorzeichen.

Parameter:
--

Rückgabe:
Das Byte, das gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Bytes lesen

Schnittstelle:

```
Int32   ReadBytes(Byte[] array);
Int32   ReadBytes(Byte[] array, Int32 length);
```

Lesen eines Byte-Arrays aus dem Datenstrom der Bytenachricht ab der aktuellen Position des Cursors.

Parameter:

array (Ausgabe)
Der Puffer, in dem die gelesenen Bytes abgelegt werden sollen. Wenn die Anzahl der verbleibenden Bytes, die noch vor dem Aufruf aus dem Datenstrom gelesen werden müssen, größer-gleich der Länge des Puffers ist, wird der Puffer gefüllt. Andernfalls wird der Puffer teilweise mit allen verbleibenden Bytes gefüllt.

Wenn Sie bei der Eingabe einen Nullzeiger angeben, überspringt die Methode die Bytes, ohne sie zu lesen. Wenn die Anzahl der verbleibenden Bytes, die noch vor dem Aufruf aus dem Datenstrom gelesen werden müssen, größer-gleich der Länge des Puffers ist, entspricht die Anzahl der übersprungenen Bytes der Länge des Puffers. Andernfalls werden alle verbleibenden Bytes übersprungen. Der Cursor bleibt an der nächsten Position, um im Bytenachrichtendatenstrom zu lesen.

length (Eingabe)
Die Länge des Puffers in Bytes.

Rückgabe:
Die Anzahl Bytes, die in den Puffer gelesen werden. Wenn der Puffer teilweise gefüllt ist, ist der Wert kleiner als die Länge des Puffers, was bedeutet, dass es keine weiteren Bytes gibt, die noch gelesen werden müssen. Wenn vor dem Aufruf keine weiteren Bytes mehr aus dem Datenstrom gelesen werden müssen, lautet der Wert `XMSC_END_OF_STREAM`.

Wenn Sie bei der Eingabe einen Nullzeiger angeben, gibt die Methode keinen Wert zurück.

Exceptions:

- XMSException
- MessageNotReadableException

ReadChar - Zeichen lesen

Schnittstelle:

```
Char    ReadChar();
```

Lesen der nächsten 2 Bytes aus dem Datenstrom der Bytenachricht als ein Zeichen.

Parameter:
--

Rückgabe:
Das Zeichen, das gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Gleitkommazahl mit doppelter Genauigkeit lesen

Schnittstelle:

```
Double  ReadDouble();
```

Lesen der nächsten 8 Bytes aus dem Datenstrom der Bytenachricht als eine Gleitkommazahl mit doppelter Genauigkeit.

Parameter:
--

Rückgabe:
Die Gleitkommazahl mit doppelter Genauigkeit, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Gleitkommazahl lesen

Schnittstelle:

```
Single  ReadFloat();
```

Lesen der nächsten 4 Bytes aus dem Datenstrom der Bytenachricht als eine Gleitkommazahl.

Parameter:
--

Rückgabe:
Die Gleitkommazahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Ganzzahl lesen

Schnittstelle:

```
Int32   ReadInt();
```

Lesen der nächsten 4 Bytes aus dem Datenstrom der Bytenachricht als eine 32-Bit-Ganzzahl mit Vorzeichen.

Parameter:
--

Rückgabe:
Die Ganzzahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Lange Ganzzahl lesen

Schnittstelle:

```
Int64   ReadLong();
```

Lesen der nächsten 8 Bytes aus dem Datenstrom der Bytenachricht als eine 64-Bit-Ganzzahl mit Vorzeichen.

Parameter:
--

Rückgabe:
Die lange Ganzzahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadShort - Kurze Ganzzahl lesen

Schnittstelle:

```
Int16   ReadShort();
```

Lesen der nächsten 2 Bytes aus dem Datenstrom der Bytenachricht als eine 16-Bit-Ganzzahl mit Vorzeichen.

Parameter:
--

Rückgabe:
Die kurze Ganzzahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - Byte ohne Vorzeichen lesen

Schnittstelle:

```
Byte    ReadByte();
```

Lesen des nächsten Bytes aus dem Datenstrom der Bytenachricht als eine 8-Bit-Ganzzahl ohne Vorzeichen.

Parameter:
--

Rückgabe:
Das Byte, das gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadUnsignedShort - Kurze Ganzzahl ohne Vorzeichen lesen

Schnittstelle:

```
Int32   ReadUnsignedShort();
```

Lesen der nächsten 2 Bytes aus dem Datenstrom der Bytenachricht als eine 16-Bit-Ganzzahl ohne Vorzeichen.

Parameter:
--

Rückgabe:
Die kurze Ganzzahl ohne Vorzeichen, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadUTF - UTF-Zeichenfolge lesen

Schnittstelle:

```
String  ReadUTF();
```

Lesen einer Zeichenfolge, die in UTF-8 codiert ist, aus dem Datenstrom der Bytenachricht.

Anmerkung: Stellen Sie vor dem Aufruf von ReadUTF () sicher, dass der Cursor des Puffers auf den Anfang des Datenstroms der Bytenachricht zeigt.

Parameter:
--

Rückgabe:
Ein Zeichenfolgeobjekt, das die gelesene Zeichenfolge enthält.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - Zurücksetzen

Schnittstelle:

```
void Reset();
```

Zurücksetzen des Hauptteils der Nachricht in den Lesezugriffsmodus und erneutes Positionieren des Cursors auf den Anfang des Datenstroms der Bytenachricht.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotReadableException

WriteBoolean - Booleschen Wert schreiben

Schnittstelle:

```
void  WriteBoolean(Boolean value);
```

Schreiben eines booleschen Werts in den Datenstrom der Bytenachricht.

Parameter:

value (Eingabe)
Der boolesche Wert, der geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteByte - Byte schreiben

Schnittstelle:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Schreiben eines Bytes in den Datenstrom der Bytenachricht.

Parameter:

value (Eingabe)
Das Byte, das geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteBytes - Bytes schreiben

Schnittstelle:

```
void  WriteBytes(Byte[] value);
```

Schreiben eines Byte-Arrays in den Datenstrom der Bytenachricht.

Parameter:

value (Eingabe)
Das Byte-Array, das geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteBytes - Partielles Byte-Array schreiben

Schnittstelle:

```
void  WriteBytes(Byte[] value, int offset, int length);
```

Schreiben eines partiellen Byte-Arrays in den Datenstrom der Bytenachricht, so wie durch die angegebene Länge definiert.

Parameter:

value (Eingabe)
Das Byte-Array, das geschrieben werden soll.

offset (Eingabe)
Die Anfangsposition für das Byte-Array, das geschrieben werden soll.

length (Eingabe)
Die Anzahl der zu schreibenden Bytes.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteChar - Zeichen schreiben

Schnittstelle:

```
void  WriteChar(Char value);
```

Schreiben eines Zeichens in den Datenstrom der Bytenachricht als 2 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Das Zeichen, das geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteDouble - Gleitkommazahl mit doppelter Genauigkeit schreiben

Schnittstelle:

```
void  WriteDouble(Double value);
```

Konvertieren einer Gleitkommazahl mit doppelter Genauigkeit in eine lange Ganzzahl und Schreiben der langen Ganzzahl in den Datenstrom der Bytenachricht als 8 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die Gleitkommazahl mit doppelter Genauigkeit, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteFloat - Gleitkommazahl schreiben

Schnittstelle:

```
void  WriteFloat(Single value);
```

Konvertieren einer Gleitkommazahl in eine Ganzzahl und Schreiben der Ganzzahl in den Datenstrom der Bytenachricht als 4 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die Gleitkommazahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteInt - Ganzzahl schreiben

Schnittstelle:

```
void  WriteInt(Int32 value);
```

Schreiben einer Ganzzahl in den Datenstrom der Bytenachricht als 4 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die Ganzzahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteLong - Lange Ganzzahl schreiben

Schnittstelle:

```
void  WriteLong(Int64 value);
```

Schreiben einer langen Ganzzahl in den Datenstrom der Bytenachricht als 8 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die lange Ganzzahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteObject - Objekt schreiben

Schnittstelle:

```
void  WriteObject(Object value);
```

Schreiben des angegebenen Objekts in den Datenstrom der Bytenachricht.

Parameter:

value (Eingabe)
Das zu schreibende Objekt, bei dem es sich um eine Referenz auf einen primitiven Datentyp handeln muss.

Rückgabe:
Void

Exceptions:

- XMSException

- MessageNotWritableException

WriteShort - Kurze Ganzzahl schreiben

Schnittstelle:

```
void  WriteShort(Int16 value);
```

Schreiben einer kurzen Ganzzahl in den Datenstrom der Bytenachricht als 2 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die kurze Ganzzahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteUTF - UTF-Zeichenfolge schreiben

Schnittstelle:

```
void  WriteUTF(String value);
```

Schreiben einer Zeichenfolge, die in UTF-8 codiert ist, in den Datenstrom der Bytenachricht.

Parameter:

value (Eingabe)
Ein Zeichenfolgeobjekt, das die zu schreibende Zeichenfolge enthält.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden von der Schnittstelle IMessage übernommen:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Die folgenden Methoden werden aus der Schnittstelle IMessage übernommen:

clearBody, clearProperties, PropertyExists

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnection

Ein Connection-Objekt stellt die aktive Verbindung der Anwendung zu einem Messaging-Server dar.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnection
```

Eine Liste der XMS-definierten Eigenschaften eines Connection-Objekts finden Sie im Abschnitt „Eigenschaften von Connection” auf Seite 190.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
ClientID	Abrufen und Festlegen der Client-ID für die Verbindung.
ExceptionListener	Abrufen des Listeners für Ausnahmebedingungen, der bei der Verbindung registriert ist, und Registrieren eines Listeners für Ausnahmebedingungen bei der Verbindung.
MetaData	Abrufen der Metadaten für die Verbindung.

ClientID - Client-ID abrufen und festlegen

Schnittstelle:

```
String ClientID
      {
        get;
        set;
      }
```

Abrufen und Festlegen der Client-ID für die Verbindung.

Die Client-ID kann entweder vom Administrator in einer ConnectionFactory vorkonfiguriert werden oder durch Festlegung der Eigenschaft ClientID zugewiesen werden.

Eine Client-ID wird nur zur Unterstützung permanenter Subskriptionen in der Publish/Subscribe-Domäne verwendet; in der Punkt-zu-Punkt-Domäne wird sie ignoriert.

Wenn eine Anwendung eine Client-ID für eine Verbindung festlegt, muss sie dies unmittelbar nach dem Erstellen der Verbindung tun und bevor sie eine andere Operation für die Verbindung ausführt. Wenn die Anwendung versucht, nach diesem Zeitpunkt eine Client-ID festzulegen, löst der Aufruf die Ausnahme IllegalStateException aus.

Diese Eigenschaft ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

Exceptions:

- XMSException
- IllegalStateException
- InvalidClientIDException

ExceptionListener - Listener für Ausnahmebedingungen abrufen und festlegen

Schnittstelle:

```
    ExceptionListener ExceptionListener
        {
          get;
          set;
        }
```

Abrufen des Listeners für Ausnahmebedingungen, der bei der Verbindung registriert ist, und Registrieren eines Listeners für Ausnahmebedingungen bei der Verbindung.

Wenn kein Listener für Ausnahmebedingungen bei der Verbindung registriert ist, gibt die Methode null zurück. Wenn bereits ein Listener für Ausnahmebedingungen bei der Verbindung registriert ist, können Sie die Registrierung abbrechen, indem Sie statt des Listeners für Ausnahmebedingungen eine Null angeben.

Weitere Informationen zur Verwendung von Listener für Ausnahmebedingungen finden Sie im Abschnitt „Listener für Nachrichten und Ausnahmebedingungen in .NET" auf Seite 53.

Exceptions:

- XMSException

Metadaten - Metadaten abrufen

Schnittstelle:

```
IConnectionMetaData MetaData
    {
      get;
    }
```

Abrufen der Metadaten für die Verbindung.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
Close	Schließen der Verbindung.
CreateSession	Erstellen einer Sitzung.
Start	Starten oder erneutes Starten der Übermittlung eingehender Nachrichten für die Verbindung.
Stop	Stoppen der Übermittlung eingehender Nachrichten für die Verbindung.

Close - Verbindung schließen

Schnittstelle:

```
void  Close();
```

Schließen der Verbindung.

Wenn eine Anwendung versucht, eine Verbindung zu schließen, die bereits geschlossen ist, wird der Aufruf ignoriert.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

CreateSession - Sitzung erstellen

Schnittstelle:

```
    ISession CreateSession(Boolean transacted,
                           AcknowledgeMode acknowledgeMode);
```

Erstellen einer Sitzung.

Parameter:

transacted (Eingabe)
Der Wert `True` bedeutet, dass es sich um eine Sitzung mit Transaktionsunterstützung handelt. Der Wert `False` bedeutet, dass es sich um eine Sitzung ohne Transaktionsunterstützung handelt.

Für eine Echtzeitverbindung zu einem Broker muss der Wert `False` angegeben werden.

acknowledgeMode (Eingabe)
Gibt an, wie Nachrichten, die von einer Anwendung empfangen werden, bestätigt werden. Der Wert muss einer der folgenden aus dem AcknowledgeMode-Aufzählungsausdruck sein:

```
AcknowledgeMode.AutoAcknowledge
AcknowledgeMode.ClientAcknowledge
AcknowledgeMode.DupsOkAcknowledge
```

Für eine Echtzeitverbindung zu einem Broker muss der Wert `AcknowledgeMode.AutoAcknowledge` oder `AcknowledgeMode.DupsOkAcknowledge` sein.

Dieser Parameter wird ignoriert, wenn es sich um eine Sitzung mit Transaktionsunterstützung handelt. Weitere Informationen zu den Bestätigungsmodi finden Sie im Abschnitt „Nachrichtenbestätigung” auf Seite 29.

Rückgabe:
Das Session-Objekt

Exceptions:

- XMSException

Start - Verbindung starten

Schnittstelle:

```
void  Start();
```

Starten oder erneutes Starten der Übermittlung eingehender Nachrichten für die Verbindung. Der Aufruf wird ignoriert, wenn die Verbindung bereits gestartet wurde.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

Stop - Verbindung stoppen

Schnittstelle:

```
void  Stop();
```

Stoppen der Übermittlung eingehender Nachrichten für die Verbindung. Der Aufruf wird ignoriert, wenn die Verbindung bereits gestoppt wurde.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionFactory

Eine Anwendung verwendet eine Verbindungsfactory, um eine Verbindung zu erstellen.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionFactory
```

Eine Liste der XMS-definierten Eigenschaften eines ConnectionFactory-Objekts finden Sie im Abschnitt „Eigenschaften von ConnectionFactory" auf Seite 190.

Zugehörige Konzepte

ConnectionFactory- und Connection-Objekte
Ein Verbindungsfactoryobjekt (ConnectionFactory) stellt eine Vorlage bereit, die von einer Anwendung zum Erstellen eines Verbindungsobjekts (Connection) verwendet wird. Das Connection-Objekt verwendet die Anwendung wiederum zum Erstellen eines Sitzungsobjekts (Session).

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager
Damit eine XMS .NET-Anwendung sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

Sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine
Damit eine XMS .NET-Anwendung sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
CreateConnection	Erstellen einer Verbindungsfactory mit den Standardeigenschaften.
CreateConnection	Erstellen einer Verbindung unter Verwendung einer angegebenen Benutzeridentität.

CreateConnection - Verbindungsfactory erstellen (mit der Standardbenutzeridentität)

Schnittstelle:

```
IConnection CreateConnection();
```

Erstellen einer Verbindungsfactory mit den Standardeigenschaften.

Wenn Sie eine Verbindung zu IBM WebSphere MQ herstellen und die Eigenschaft XMSC_USERID der Verbindungsfactory festlegen, muss sie mit der **userid** des angemeldeten Benutzers übereinstimmen. Wenn Sie diese Eigenschaft nicht festlegen, verwendet der Warteschlangenmanager standardmäßig die **userid** des angemeldeten Benutzers. Wenn Sie eine weitere Authentifizierung einzelner Benutzer auf Verbindungsebene benötigen, können Sie einen Clientauthentifizierungsexit schreiben, der in IBM WebSphere MQkonfiguriert ist.

Parameter:

--

Exceptions:

- XMSException

CreateConnection - Verbindung erstellen (mit einer angegebenen Benutzeridentität)

Schnittstelle:

```
IConnection CreateConnection(String userId, String password);
```

Erstellen einer Verbindung unter Verwendung einer angegebenen Benutzeridentität.

Wenn Sie eine Verbindung zu IBM WebSphere MQ herstellen und die Eigenschaft XMSC_USERID der Verbindungsfactory festlegen, muss sie mit der **userid** des angemeldeten Benutzers übereinstimmen. Wenn Sie diese Eigenschaft nicht festlegen, verwendet der Warteschlangenmanager standardmäßig die **userid** des angemeldeten Benutzers. Wenn Sie eine weitere Authentifizierung einzelner Benutzer auf Verbindungsebene benötigen, können Sie einen Clientauthentifizierungsexit schreiben, der in IBM WebSphere MQkonfiguriert ist.

Die Verbindung wird im gestoppten Modus erstellt. Nachrichten werden erst übermittelt, wenn die Anwendung **Connection.start()** aufruft.

Parameter:

userID (Eingabe)
Ein Zeichenfolgeobjekt, das die Benutzer-ID enthält, die für die Authentifizierung der Anwendung verwendet werden soll. Wenn Sie eine Null angeben, wird versucht, die Verbindung ohne Authentifizierung herzustellen.

password (Eingabe)
Ein Zeichenfolgeobjekt, das das Kennwort enthält, das für die Authentifizierung der Anwendung verwendet werden soll. Wenn Sie eine Null angeben, wird versucht, die Verbindung ohne Authentifizierung herzustellen.

Rückgabe:

Das Connection-Objekt.

Exceptions:

- XMSException
- XMS_X_SECURITY_EXCEPTION

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionMetaData

Ein ConnectionMetaData-Objekt stellt Informationen zu einer Verbindung bereit.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionMetaData
```

Eine Liste der XMS-definierten Eigenschaften eines ConnectionMetaData-Objekts finden Sie im Abschnitt „Eigenschaften von ConnectionMetaData” auf Seite 197.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften:

Methode	Beschreibung
JMSXPropertyNames	Gibt eine Auflistung der Namen der von JMS definierten Nachrichteneigenschaften zurück, die von der Verbindung unterstützt werden.

JMSXPropertyNames - JMS-definierte Nachrichteneigenschaften abrufen

Schnittstelle:

```
System.Collections.IEnumerator JMSXPropertyNames
    {
      get;
    }
```

Gibt eine Auflistung der Namen der von JMS definierten Nachrichteneigenschaften zurück, die von der Verbindung unterstützt werden.

JMS-definierte Nachrichteneigenschaften werden von einer Echtzeitverbindung zu einem Broker nicht unterstützt.

Exceptions:

- XMSException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IDestination

Ein Ziel ist die Adresse, an die eine Anwendung Nachrichten sendet, und/oder es ist eine Quelle, von der eine Anwendung Nachrichten empfängt.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IDestination
```

Eine Liste der XMS-definierten Eigenschaften eines Destination-Objekts finden Sie im Abschnitt „Eigenschaften von Destination" auf Seite 198.

Zugehörige Konzepte

ConnectionFactory- und Connection-Objekte
Ein Verbindungsfactoryobjekt (ConnectionFactory) stellt eine Vorlage bereit, die von einer Anwendung zum Erstellen eines Verbindungsobjekts (Connection) verwendet wird. Das Connection-Objekt verwendet die Anwendung wiederum zum Erstellen eines Sitzungsobjekts (Session).

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Ziele
XMS-Anwendungen geben mit einem Destination-Objekt das Ziel für gesendete Nachrichten und die Quelle von empfangenen Nachrichten an.

Zielplatzhalterzeichen
XMS bietet Unterstützung für Zielplatzhalterzeichen, um sicherzustellen, dass Platzhalterzeichen an die Stelle übergeben werden können, an der sie für den Abgleich benötigt werden. Es gibt für jeden Servertyp, mit dem XMS zusammenarbeiten kann, ein anderes Platzhalterschema.

Themen-URIs
Der Uniform Resource Identifier (URI) eines Themas gibt den Namen des Themas und ggf. auch eine oder mehrere seiner Eigenschaften an.

Warteschlangen-URIs
Der Uniform Resource Identifier (URI) einer Warteschlange gibt den Namen der Warteschlange und ggf. auch eine oder mehrere ihrer Eigenschaften an.

Temporäre Ziele
XMS-Anwendungen können temporäre Ziele erstellen und verwenden.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

.NET-Eigenschaften

Zusammenfassung der Methoden:

Methode	Beschreibung
Name	Abrufen des Namens des Ziels.
TypeId	Abrufen des Typs des Ziels.

Name - Zielname abrufen

Schnittstelle:

```
String Name
{
  get;
}
```

Abrufen des Namens des Ziels. Der Name ist eine Zeichenfolge, die entweder den Namen einer Warteschlange oder den Namen eines Themas angibt.

Exceptions:

- XMSException

TypeId - Zieltyp abrufen

Schnittstelle:

```
DestinationType TypeId
{
  get;
}
```

Abrufen des Typs des Ziels. Der Typ des Ziels ist einer der folgenden Werte:

```
DestinationType.Queue
DestinationType.Topic
```

Exceptions:

- XMSException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ExceptionListener

Vererbungshierarchie:

--

Eine Anwendung verwendet einen Listener für Ausnahmebedingungen, um asynchron über ein Problem mit einer Verbindung benachrichtigt zu werden.

Wenn eine Anwendung eine Verbindung nur zum asynchronen Konsumieren von Nachrichten und zu keinem anderen Zweck verwendet, kann die Anwendung nur dann Kenntnis von einem Problem mit der Verbindung erhalten, wenn ein Listener für Ausnahmebedingungen verwendet wird. In anderen Situationen kann ein Listener für Ausnahmebedingungen eine direktere Möglichkeit darstellen, von einem Problem mit einer Verbindung zu erfahren, als bis zum nächsten synchronen Aufruf von XMS zu warten.

Delegat

Zusammenfassung Delegat:

Delegat	Beschreibung
ExceptionListener	Benachrichtigen der Anwendung über ein Problem mit einer Verbindung.

ExceptionListener - Listener für Ausnahmebedingungen

Schnittstelle:

```
public delegate void ExceptionListener(Exception ex)
```

Benachrichtigen der Anwendung über ein Problem mit einer Verbindung.

Methoden, die dieses Delegat implementieren, können für die Verbindung registriert werden.

Weitere Informationen zur Verwendung von Listener für Ausnahmebedingungen finden Sie im Abschnitt „Listener für Nachrichten und Ausnahmebedingungen in .NET” auf Seite 53.

Parameter:

exception (Eingabe)
Ein Zeiger auf eine Ausnahme, die von XMSerstellt wurde.

Rückgabe:
Void

IllegalStateException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.Exception
           |
           +----IBM.XMS.IllegalStateException
```

XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung eine Methode zu einem falschen oder ungeeigneten Zeitpunkt aufruft oder wenn XMS kein geeigneter Status für die angeforderte Operation ist.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

InitialContext

Eine Anwendung verwendet ein InitialContext-Objekt, um Objekte aus Objektdefinitionen zu erstellen, die aus einem Repository mit verwalteten Objekten abgerufen werden.

Vererbungshierarchie:
--

Zugehörige Konzepte
InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
Environment	Abrufen der Umgebung.

Environment - Umgebung abrufen

Schnittstelle:

```
Hashtable Environment
{
     get;
}
```

Abrufen der Umgebung.

Exceptions:

- Ausnahmen sind spezifisch für den verwendeten Verzeichnisservice.

Konstruktoren

Zusammenfassung der Konstruktoren:

Konstruktor	Beschreibung
InitialContext	Erstellen eines InitialContext-Objekts.

InitialContext - Ausgangskontext erstellen

Schnittstelle:

```
InitialContext(Hashtable env);
```

Erstellen eines InitialContext-Objekts.

Parameter:

Die zum Einrichten einer Verbindung zum Repository mit verwalteten Objekten erforderlichen Informationen werden dem Konstruktor in einer Umgebungshashtabelle bereitgestellt.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
AddToEnvironment	Hinzufügen einer neuen Eigenschaft zur Umgebung.
Close	Schließen des Kontextes.

Methode	Beschreibung
Lookup	Erstellen eines Objekts aus einer Objektdefinition, die aus dem Repository mit verwalteten Objekten abgerufen wird.
RemoveFromEnvironment	Entfernen einer Eigenschaft aus der Umgebung.

AddToEnvironment - Neue Eigenschaft zur Umgebung hinzufügen

Schnittstelle:

```
Object AddToEnvironment(String propName, Object propVal);
```

Hinzufügen einer neuen Eigenschaft zur Umgebung.

Parameter:

propName (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der hinzuzufügenden Eigenschaft angibt.

propVal (Eingabe)
Der Wert der hinzuzufügenden Eigenschaft.

Rückgabe:
Der alte Wert der Eigenschaft.

Exceptions:

- Ausnahmen sind spezifisch für den verwendeten Verzeichnisservice.

Close - Kontext schließen

Schnittstelle:

```
void Close()
```

Schließen des Kontextes.

Parameter:
--

Rückgabe:
--

Exceptions:

- Ausnahmen sind spezifisch für den verwendeten Verzeichnisservice.

Lookup - Objekt im Ausgangskontext suchen

Schnittstelle:

```
Object Lookup(String name);
```

Erstellen eines Objekts aus einer Objektdefinition, die aus dem Repository mit verwalteten Objekten abgerufen wird.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des verwalteten Objekts angibt, das abgerufen werden soll. Dabei kann es sich um einen einfachen oder komplexen Namen handeln. Weitere Informationen dazu finden Sie unter „Verwaltete Objekte abrufen” auf Seite 69.

Rückgabe:
Entweder ein IConnectionFactory oder ein IDestination, abhängig vom Typ des abgerufenen Objekts. Wenn die Funktion auf das Verzeichnis zugreifen, aber das erforderliche Objekt nicht finden kann, wird eine Null zurückgegeben.

Exceptions:

- Ausnahmen sind spezifisch für den verwendeten Verzeichnisservice.

RemoveFromEnvironment - Eigenschaft aus der Umgebung entfernen

Schnittstelle:

```
Object RemoveFromEnvironment(String propName);
```

Entfernen einer Eigenschaft aus der Umgebung.

Parameter:

propName (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der zu entfernenden Eigenschaft angibt.

Rückgabe:
Das Objekt, das entfernt wurde.

Exceptions:

- Ausnahmen sind spezifisch für den verwendeten Verzeichnisservice.

InvalidClientIDException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidClientIDException
```

XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung eine Client-ID für eine Verbindung festlegen möchte, die Client-ID aber nicht gültig ist oder bereits verwendet wird.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

InvalidDestinationException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidDestinationException
```

XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung ein Ziel angibt, das nicht gültig ist.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

InvalidSelectorException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
```

```
          |
          +----IBM.XMS.InvalidSelectorException
```

XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung einen Ausdruck für einen Nachrichtenselektor bereitstellt, dessen Syntax nicht gültig ist.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

IMapMessage

Eine Zuordnungsnachricht ist eine Nachricht, deren Hauptteil aus einer Gruppe von Name/Wert-Paaren besteht, wobei jedem Wert ein Datentyp zugeordnet ist.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IMapMessage
```

Wenn eine Anwendung den Wert eines Name/Wert-Paars abruft, kann der Wert von XMS in einen anderen Datentyp konvertiert werden. Weitere Informationen zu dieser Form impliziter Konvertierungen finden Sie im Abschnitt „Zuordnungsnachrichten” auf Seite 83.

Zugehörige Verweise

Zuordnungsnachrichten
Der Hauptteil einer Zuordnungsnachricht enthält eine Gruppe von Name/Wert-Paaren, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
MapNames	Abrufen einer Auflistung der Namen im Hauptteil der Zuordnungsnachricht.

MapNames - Zuordnungsnamen abrufen

Schnittstelle:

```
System.Collections.IEnumerator MapNames
{
  get;
}
```

Abrufen einer Auflistung der Namen im Hauptteil der Zuordnungsnachricht.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
GetBoolean	Abrufen des booleschen Werts mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Methode	Beschreibung
GetByte	Abrufen des Bytes mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetBytes	Abrufen des Byte-Arrays mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetChar	Abrufen des Zeichens mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetDouble	Abrufen der Gleitkommazahl mit doppelter Genauigkeit mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetFloat	Abrufen der Gleitkommazahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetInt	Abrufen der Ganzzahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetLong	Abrufen der langen Ganzzahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetObject	Abrufen eines Verweises auf den Wert eines Name/Wert-Paars aus dem Hauptteil der Zuordnungsnachricht.
GetShort	Abrufen der kurzen Ganzzahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
GetString	Abrufen der Zeichenfolge mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.
ItemExists	Prüfen, ob der Hauptteil der Zuordnungsnachricht ein Name/Wert-Paar mit dem angegebenen Namen enthält.
SetBoolean	Festlegen eines booleschen Werts im Hauptteil der Zuordnungsnachricht.
SetByte	Festlegen eines Bytes im Hauptteil der Zuordnungsnachricht.
SetBytes	Festlegen eines Byte-Arrays im Hauptteil der Zuordnungsnachricht.
SetChar	Festlegen eines 2-Byte-Zeichens im Hauptteil der Zuordnungsnachricht.
SetDouble	Festlegen einer Gleitkommazahl mit doppelter Genauigkeit im Hauptteil der Zuordnungsnachricht.
SetFloat	Festlegen einer Gleitkommazahl im Hauptteil der Zuordnungsnachricht.
SetInt	Festlegen einer Ganzzahl im Hauptteil der Zuordnungsnachricht.
SetLong	Festlegen einer langen Ganzzahl im Hauptteil der Zuordnungsnachricht.
SetObject	Festlegen eines Werts, bei dem es sich um einen primitiven XMS-Datentyp handeln muss, im Hauptteil der Zuordnungsnachricht.
SetShort	Festlegen einer kurzen Ganzzahl im Hauptteil der Zuordnungsnachricht.
SetString	Festlegen einer Zeichenfolge im Hauptteil der Zuordnungsnachricht.

GetBoolean - Booleschen Wert abrufen

Schnittstelle:

```
Boolean GetBoolean(String name);
```

Abrufen des booleschen Werts mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des booleschen Werts angibt.

Rückgabe:
Der boolesche Wert, der aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde.

Exceptions:

- XMSException

GetByte - Byte abrufen

Schnittstelle:

```
Byte    GetByte(String name);
    Int16   GetSignedByte(String name);
```

Abrufen des Bytes mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Bytes angibt.

Rückgabe:
Das Byte, das aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde. Es wird keine Datenkonvertierung für das Byte durchgeführt.

Exceptions:

- XMSException

GetBytes - Bytes abrufen

Schnittstelle:

```
Byte[]  GetBytes(String name);
```

Abrufen des Byte-Arrays mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Byte-Arrays angibt.

Rückgabe:
Die Anzahl der Bytes im Array.

Exceptions:

- XMSException

GetChar - Zeichen abrufen

Schnittstelle:

```
Char    GetChar(String name);
```

Abrufen des Zeichens mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Zeichens angibt.

Rückgabe:
Das Zeichen, das aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde.

Exceptions:

- XMSException

GetDouble - Gleitkommazahl mit doppelter Genauigkeit abrufen

Schnittstelle:

```
Double  GetDouble(String name);
```

Abrufen der Gleitkommazahl mit doppelter Genauigkeit mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Gleitkommazahl mit doppelter Genauigkeit angibt.

Rückgabe:
Die Gleitkommazahl mit doppelter Genauigkeit, die aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde.

Exceptions:

- XMSException

GetFloat - Gleitkommazahl abrufen

Schnittstelle:

```
Single  GetFloat(String name);
```

Abrufen der Gleitkommazahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Gleitkommazahl angibt.

Rückgabe:
Die Gleitkommazahl, die aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde.

Exceptions:

- XMSException

GetInt - Ganzzahl abrufen

Schnittstelle:

```
Int32   GetInt(String name);
```

Abrufen der Ganzzahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Ganzzahl angibt.

Rückgabe:
Die Ganzzahl, die aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde

Exceptions:

- XMSException

GetLong - Lange Ganzzahl abrufen

Schnittstelle:

```
Int64   GetLong(String name);
```

Abrufen der langen Ganzzahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der langen Ganzzahl angibt.

Rückgabe:
Die lange Ganzzahl, die aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde.

Exceptions:

- XMSException

GetObject - Objekt abrufen

Schnittstelle:

```
Object  GetObject(String name);
```

Abrufen eines Verweises auf den Wert eines Name/Wert-Paars aus dem Hauptteil der Zuordnungsnachricht. Das Name/Wert-Paar wird durch den Namen angegeben.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Name/Wert-Paars angibt.

Rückgabe:
Der Wert, bei dem es sich um einen der folgenden Objektypen handelt:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Exceptions:
XMSException

GetShort - Kurze Ganzzahl abrufen

Schnittstelle:

```
Int16   GetShort(String name);
```

Abrufen der kurzen Ganzzahl mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der kurzen Ganzzahl angibt.

Rückgabe:
Die kurze Ganzzahl, die aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde.

Exceptions:

- XMSException

GetString - Zeichenfolge abrufen

Schnittstelle:

```
String  GetString(String name);
```

Abrufen der Zeichenfolge mit dem angegebenen Namen aus dem Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Zeichenfolge im Hauptteil der Zuordnungsnachricht angibt.

Rückgabe:
Ein Zeichenfolgeobjekt, das die Zeichenfolge enthält, die aus dem Hauptteil der Zuordnungsnachricht abgerufen wurde. Wenn eine Datenkonvertierung erforderlich ist, ist dieser Wert die Zeichenfolge nach der Konvertierung.

Exceptions:

- XMSException

ItemExists - Vorhandensein eines Name/Wert-Paars prüfen

Schnittstelle:

```
Boolean ItemExists(String name);
```

Prüfen, ob der Hauptteil der Zuordnungsnachricht ein Name/Wert-Paar mit dem angegebenen Namen enthält.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Name/Wert-Paars angibt.

Rückgabe:

- `True`, wenn der Hauptteil der Zuordnungsnachricht ein Name/Wert-Paar mit dem angegebenen Namen enthält.
- `False`, wenn der Hauptteil der Zuordnungsnachricht kein Name/Wert-Paar mit dem angegebenen Namen enthält.

Exceptions:

- XMSException

SetBoolean - Booleschen Wert festlegen

Schnittstelle:

```
void  SetBoolean(String name, Boolean value);
```

Festlegen eines booleschen Werts im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des booleschen Werts im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Der boolesche Wert, der festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetByte - Byte festlegen

Schnittstelle:

```
void  SetByte(String name, Byte value);
void  SetSignedByte(String name, Int16 value);
```

Festlegen eines Bytes im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Bytes im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Das Byte, das festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetBytes - Bytes festlegen

Schnittstelle:

```
void  SetBytes(String name, Byte[] value);
```

Festlegen eines Byte-Arrays im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Byte-Arrays im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Das Byte-Array, das festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetChar - Zeichen festlegen

Schnittstelle:

```
void  SetChar(String name, Char value);
```

Festlegen eines 2-Byte-Zeichens im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Zeichens im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Das Zeichen, das festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetDouble - Gleitkommazahl mit doppelter Genauigkeit festlegen

Schnittstelle:

```
void  SetDouble(String name, Double value);
```

Festlegen einer Gleitkommazahl mit doppelter Genauigkeit im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Gleitkommazahl mit doppelter Genauigkeit im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Die Gleitkommazahl mit doppelter Genauigkeit, die festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetFloat - Gleitkommazahl festlegen

Schnittstelle:

```
void  SetFloat(String name, Single value);
```

Festlegen einer Gleitkommazahl im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Gleitkommazahl im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Die Gleitkommazahl, die festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetInt - Ganzzahl festlegen

Schnittstelle:

```
void  SetInt(String name, Int32 value);
```

Festlegen einer Ganzzahl im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Ganzzahl im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Die Ganzzahl, die festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetLong - Lange Ganzzahl festlegen

Schnittstelle:

```
void  SetLong(String name, Int64 value);
```

Festlegen einer langen Ganzzahl im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der langen Ganzzahl im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Die lange Ganzzahl, die festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetObject - Objekt festlegen

Schnittstelle:

```
void  SetObject(String name, Object value);
```

Festlegen eines Werts, bei dem es sich um einen primitiven XMS-Datentyp handeln muss, im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Werts im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Ein Byte-Array, das den festzulegenden Wert enthält.

Rückgabe:
Void

Exceptions:

- XMSException

SetShort - Kurze Ganzzahl festlegen

Schnittstelle:

```
void  SetShort(String name, Int16 value);
```

Festlegen einer kurzen Ganzzahl im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der kurzen Ganzzahl im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Die kurze Ganzzahl, die festgelegt werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

SetString - Zeichenfolge festlegen

Schnittstelle:

```
void  SetString(String name, String value);
```

Festlegen einer Zeichenfolge im Hauptteil der Zuordnungsnachricht.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Zeichenfolge im Hauptteil der Zuordnungsnachricht angibt.

value (Eingabe)
Ein Zeichenfolgeobjekt, das die festzulegende Zeichenfolge enthält.

Rückgabe:
Void

Exceptions:

- XMSException

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden von der Schnittstelle IMessage übernommen:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Die folgenden Methoden werden aus der Schnittstelle IMessage übernommen:

clearBody, clearProperties, PropertyExists

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessage

Ein Nachrichtenobjekt stellt eine Nachricht dar, die von einer Anwendung gesendet oder empfangen wird. IMessage ist eine Superklasse für die Nachrichtenklassen, wie beispielsweise IMapMessage.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
```

Eine Liste der JMS-Nachrichtenheaderfelder in einem Nachrichtenobjekt finden Sie im Abschnitt „Headerfelder in Eine XMS -Nachricht” auf Seite 76. Eine Liste der mit JMS definierten Eigenschaften eines Nachrichtenobjekts finden Sie im Abschnitt „JMS-definierte Eigenschaften einer Nachricht” auf Seite 78. Eine Liste der mit IBM definierten Eigenschaften eines Nachrichtenobjekts finden Sie im Abschnitt „IBM-definierte Eigenschaften einer Nachricht” auf Seite 78. Eine Liste der JMS_IBM_MQMD*-Eigenschaften für das Nachrichtenobjekt finden Sie im Abschnitt „JMS_IBM_MQMD*-Eigenschaften” auf Seite 203

Nachrichten werden vom Garbage-Collector gelöscht. Beim Löschen einer Nachricht werden die von ihr belegten Ressourcen freigegeben.

Zugehörige Verweise

XMS-Nachrichten
In diesem AbschnittKapitel werden die Struktur und der Inhalt von XMS-Nachrichten beschrieben und es wird erläutert, wie Anwendungen XMS-Nachrichten verarbeiten.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
JMSCorrelationID	Abrufen und Festlegen der Korrelations-ID der Nachricht als ein Zeichenfolgeobjekt.
JMSDeliveryMode	Abrufen und Festlegen des Zustellmodus der Nachricht.
JMSDestination	Abrufen und Festlegen des Ziels der Nachricht.
JMSExpiration	Abrufen und Festlegen der Ablaufzeit der Nachricht.
JMSMessageID	Abrufen und Festlegen der Nachrichten-ID der Nachricht als ein Zeichenfolgeobjekt, das die Nachrichten-ID enthält.
JMSPriority	Abrufen und Festlegen der Priorität der Nachricht.
JMSRedelivered	Abrufen einer Angabe, ob die Nachricht erneut zugestellt wird, und Angeben, ob die Nachricht erneut zugestellt wird.
JMSReplyTo	Abrufen und Festlegen des Ziels, an dem eine Antwort auf die Nachricht gesendet werden soll.
JMSTimestamp	Abrufen und Festlegen der Zeit, zu der die Nachricht gesendet wurde.
JMSType	Abrufen und Festlegen des Typs der Nachricht.
PropertyNames	Abrufen einer Auflistung der Namenseigenschaften der Nachricht.

GetJMSCorrelationID - JMSCorrelationID abrufen und festlegen

Schnittstelle:

```
String JMSCorrelationID
{
  get;
  set;
}
```

Abrufen und Festlegen der Korrelations-ID der Nachricht als ein Zeichenfolgeobjekt.

Exceptions:

- XMSException

JMSDeliveryMode - JMSDeliveryMode abrufen und festlegen

Schnittstelle:

```
DeliveryMode JMSDeliveryMode
{
  get;
  set;
}
```

Abrufen und Festlegen des Zustellmodus der Nachricht.

Der Zustellmodus der Nachricht ist einer der folgenden Werte:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Eine neu erstellte Nachricht, die nicht gesendet wurde, hat den Zustellmodus `DeliveryMode.Persistent`, außer bei einer Echtzeitverbindung zu einem Broker, für die der Zustellmodus `DeliveryMode.NonPersistent` ist. Für eine Nachricht, die empfangen wird, gibt die Methode den Zustellmodus zurück, der beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt wurde, es sei denn, die empfangende Anwendung ändert den Zustellmodus, indem sie die Eigenschaft JMSDeliveryMode festlegt.

Exceptions:

- XMSException

JMSDestination - JMSDestination abrufen und festlegen

Schnittstelle:

```
IDestination JMSDestination
{
  get;
  set;
}
```

Abrufen und Festlegen des Ziels der Nachricht.

Das Ziel wird beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt. Der Wert von JMSDestination wird ignoriert. Sie haben jedoch die Möglichkeit, das Ziel einer Nachricht, die empfangen wurde, mithilfe von JMSDestination zu ändern.

Für eine neu erstellte Nachricht, die nicht gesendet wurde, gibt die Methode ein Destination-Objekt mit einem Nullwert zurück, außer wenn die sendende Anwendung durch Einstellung von JMSDestination ein Ziel festlegt. Für eine Nachricht, die empfangen wurde, gibt die Methode ein Destination-Objekt für das Ziel zurück, das beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt wurde, es sei denn, die empfangende Anwendung ändert das Ziel, indem sie die Eigenschaft JMSDestination festlegt.

Exceptions:

- XMSException

JMSExpiration - JMSExpiration abrufen und festlegen

Schnittstelle:

```
Int64 JMSExpiration
{
  get;
  set;
}
```

Abrufen und Festlegen der Ablaufzeit der Nachricht.

Die Ablaufzeit wird beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt. Der Wert wird berechnet, indem die Lebensdauer, so wie von der sendenden Anwendung angegeben, zu der Zeit hinzugefügt wird, zu der die Nachricht gesendet wird. Die Ablaufzeit wird in Millisekunden seit 00:00:00 GMT am 1. Januar 1970 ausgedrückt.

Für eine neu erstellte Nachricht, die nicht gesendet wurde, gilt die Ablaufzeit 0, es sei denn, die sendende Anwendung legt durch Einstellung von JMSExpiration eine andere Ablaufzeit fest. Für eine Nachricht, die empfangen wurde, gibt die Methode die Ablaufzeit zurück, die beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt wurde, es sei denn, die empfangende Anwendung ändert die Ablaufzeit, indem sie die Eigenschaft JMSExpiration festlegt.

Wenn die Lebensdauer 0 ist, legt der Aufruf IMessageProducer.send() die Ablaufzeit auf 0 fest, um anzugeben, dass die Nachricht nicht abläuft.

XMS löscht abgelaufene Nachrichten und übermittelt sie nicht an Anwendungen.

Exceptions:

- XMSException

JMSMessageID - JMSMessageID abrufen und festlegen

Schnittstelle:

```
String JMSMessageID
{
  get;
  set;
}
```

Abrufen und Festlegen der Nachrichten-ID der Nachricht als ein Zeichenfolgeobjekt, das die Nachrichten-ID enthält.

Die Nachrichten-ID wird beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt. Für eine Nachricht, die empfangen wurde, gibt die Methode die Nachrichten-ID zurück, die beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt wurde, es sei denn, die empfangende Anwendung ändert die Nachrichten-ID, indem sie die Eigenschaft JMSMessageID festlegt.

Wenn die Nachricht keine Nachrichten-ID hat, gibt die Methode eine Null zurück.

Exceptions:

- XMSException

JMSPriority - JMSPriority abrufen und festlegen

Schnittstelle:

```
Int32 JMSPriority
{
  get;
  set;
}
```

Abrufen und Festlegen der Priorität der Nachricht.

Die Priorität wird beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt. Der Wert ist eine Ganzzahl im Bereich von 0 (niedrigste Priorität) bis 9 (höchste Priorität).

Eine neu erstellte Nachricht, die nicht gesendet wurde, hat die Priorität 4, es sei denn, die sendende Anwendung legt durch Einstellung von JMSPriority eine andere Priorität fest. Für eine Nachricht, die empfangen wurde, gibt die Methode die Priorität zurück, die beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt wurde, es sei denn, die empfangende Anwendung ändert die Priorität, indem sie die Eigenschaft JMSPriority festlegt.

Exceptions:

- XMSException

JMSRedelivered - JMSRedelivered abrufen und festlegen

Schnittstelle:

```
Boolean JMSRedelivered
{
  get;
  set;
}
```

Abrufen einer Angabe, ob die Nachricht erneut zugestellt wird, und Angeben, ob die Nachricht erneut zugestellt wird. Die Angabe wird beim Empfangen der Nachricht durch den Aufruf IMessageConsumer.receive() festgelegt.

Diese Eigenschaft hat folgende Werte:

- `True`, wenn die Nachricht erneut zugestellt wird.
- `False`, wenn die Nachricht nicht erneut zugestellt wird.

Bei einer Echtzeitverbindung zu einem Broker ist der Wert immer `False`.

Eine Angabe durch JMSRedelivered vor dem Senden der Nachricht, dass eine erneute Zustellung erfolgt, wird beim Senden der Nachricht vom Aufruf IMessageProducer.send() ignoriert und beim Empfangen der Nachricht vom Aufruf IMessageConsumer.receive() ignoriert und ersetzt. Sie haben jedoch die Möglichkeit, die Angabe für eine Nachricht, die empfangen wurde, mithilfe von JMSRedelivered zu ändern.

Exceptions:

- XMSException

JMSReplyTo - JMSReplyTo abrufen und festlegen

Schnittstelle:

```
IDestination JMSReplyTo
{
  get;
  set;
}
```

Abrufen und Festlegen des Ziels, an dem eine Antwort auf die Nachricht gesendet werden soll.

Der Wert dieser Eigenschaft ist ein Destination-Objekt für das Ziel, an dem eine Antwort auf die Nachricht gesendet werden soll. Ein Destination-Objekt mit einem Nullwert bedeutet, dass keine Antwort erwartet wird.

Exceptions:

- XMSException

JMSTimestamp - JMSTimestamp abrufen und festlegen

Schnittstelle:

```
Int64 JMSTimestamp
{
  get;
  set;
}
```

Abrufen und Festlegen der Zeit, zu der die Nachricht gesendet wurde.

Die Zeitmarke wird beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt und in Millisekunden seit 00:00:00 GMT am 1. Januar 1970 ausgedrückt.

Eine neu erstellte Nachricht, die nicht gesendet wurde, hat die Zeitmarke 0, es sei denn, die sendende Anwendung legt durch Einstellung von JMSTimestamp eine andere Zeitmarke fest. Für eine Nachricht, die empfangen wurde, gibt die Methode die Zeitmarke zurück, die beim Senden der Nachricht durch den Aufruf IMessageProducer.send() festgelegt wurde, es sei denn, die empfangende Anwendung ändert die Zeitmarke, indem sie die Eigenschaft JMSTimestamp festlegt.

Exceptions:

- XMSException

Anmerkungen:

1. Wenn die Zeitmarke nicht definiert ist, gibt die Methode 0 zurück, löst jedoch keine Ausnahme aus.

JMSType - JMSType abrufen und festlegen

Schnittstelle:

```
String JMSType
{
  get;
```

```
  set;
}
```

Abrufen und Festlegen des Typs der Nachricht.

Der Wert von JMSType ist eine Zeichenfolge, die den Typ der Nachricht enthält. Wenn eine Datenkonvertierung erforderlich ist, gibt dieser Wert den Typ nach der Konvertierung an.

Exceptions:

- XMSException

PropertyNames - Eigenschaften abrufen

Schnittstelle:

```
System.Collections.IEnumerator PropertyNames
{
  get;
}
```

Abrufen einer Auflistung der Namenseigenschaften der Nachricht.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
Acknowledge	Bestätigen dieser Nachricht und aller zuvor nicht bestätigten Nachrichten, die von der Sitzung empfangen wurden.
ClearBody	Löschen des Inhalts des Hauptteils der Nachricht.
ClearProperties	Löschen der Eigenschaften der Nachricht.
PropertyExists	Überprüfen, ob die Nachricht eine Eigenschaft mit dem angegebenen Namen hat.

Acknowledge - Bestätigen

Schnittstelle:

```
void  Acknowledge();
```

Bestätigen dieser Nachricht und aller zuvor nicht bestätigten Nachrichten, die von der Sitzung empfangen wurden.

Eine Anwendung kann diese Methode aufrufen, wenn der Bestätigungsmodus der Sitzung 'AcknowledgeMode.ClientAcknowledge' lautet. Aufrufe an die Methode werden ignoriert, wenn die Sitzung einen anderen Bestätigungsmodus hat oder es sich um eine transaktionsbasierte Sitzung handelt.

Nachrichten, die empfangen, aber nicht bestätigt wurden, werden möglicherweise erneut zugestellt.

Weitere Informationen zur Bestätigung von Nachrichten finden Sie im Abschnitt „Nachrichtenbestätigung” auf Seite 29.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

- IllegalStateException

ClearBody - Inhalt des Hauptteils löschen

Schnittstelle:

```
void  ClearBody();
```

Löschen des Inhalts des Hauptteils der Nachricht. Die Headerfelder und Nachrichteneigenschaften werden nicht gelöscht.

Wenn eine Anwendung einen Nachrichtenhauptteil löscht, verbleibt der Hauptteil in demselben Status wie ein leerer Hauptteil in einer neu erstellten Nachricht. Der Zustand eines leeren Hauptteils in einer neu erstellten Nachricht hängt vom Typ des Nachrichtenhauptteils ab. Weitere Informationen finden Sie unter „Hauptteil der Eine XMS -Nachricht" auf Seite 80.

Eine Anwendung kann den Inhalt eines Nachrichtenhauptteils jederzeit löschen, unabhängig davon, in welchem Zustand sich der Hauptteil befindet. Wenn ein Nachrichtenhauptteil schreibgeschützt ist, hat eine Anwendung, die in den Hauptteil schreiben will, nur eine einzige Möglichkeit: Sie muss zuerst den Inhalt des Hauptteils löschen.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

ClearProperties - Eigenschaften löschen

Schnittstelle:

```
void  ClearProperties();
```

Löschen der Eigenschaften der Nachricht. Die Headerfelder und der Inhalt des Nachrichtenhauptteils werden nicht gelöscht.

Wenn eine Anwendung die Eigenschaften einer Nachricht löscht, werden die Eigenschaften lesbar und beschreibbar.

Eine Anwendung kann die Eigenschaften einer Nachricht jederzeit löschen, unabhängig davon, in welchem Zustand sich die Eigenschaften befinden. Wenn die Eigenschaften einer Nachricht schreibgeschützt sind, hat die Anwendung nur eine einzige Möglichkeit, die Eigenschaften beschreibbar zu machen: Sie muss zuerst die Eigenschaften löschen.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

PropertyExists - Vorhandensein einer Eigenschaft prüfen

Schnittstelle:

```
Boolean PropertyExists(String propertyName);
```

Überprüfen, ob die Nachricht eine Eigenschaft mit dem angegebenen Namen hat.

Parameter:

propertyName (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:

- `True`, wenn die Nachricht eine Eigenschaft mit dem angegebenen Namen hat.
- `False`, wenn die Nachricht keine Eigenschaft mit dem angegebenen Namen hat.

Exceptions:

- XMSException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessageConsumer

Eine Anwendung verwendet einen Nachrichtenkonsumenten (Message Consumer), um Nachrichten von einem Ziel zu empfangen.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageConsumer
```

Eine Liste der XMS-definierten Eigenschaften eines MessageConsumer-Objekts finden Sie im Abschnitt „Eigenschaften von MessageConsumer” auf Seite 207.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
MessageListener	Abrufen des Nachrichtenlisteners, der beim Nachrichtenkonsumenten registriert ist, und Registrieren eines Nachrichtenlisteners beim Nachrichtenkonsumenten.
MessageSelector	Abrufen des Nachrichtenselektors für den Nachrichtenkonsumenten.

MessageListener - Nachrichtenlistener abrufen und festlegen

Schnittstelle:

```
MessageListener MessageListener
{
  get;
  set;
}
```

Abrufen des Nachrichtenlisteners, der beim Nachrichtenkonsumenten registriert ist, und Registrieren eines Nachrichtenlisteners beim Nachrichtenkonsumenten.

Wenn kein Nachrichtenlistener beim Nachrichtenkonsumenten registriert ist, hat MessageListener den Wert null. Wenn bereits ein Nachrichtenlistener beim Nachrichtenkonsumenten registriert ist, können Sie die Registrierung abbrechen, indem Sie stattdessen eine Null angeben.

Weitere Informationen zur Verwendung von Nachrichtenlisteners finden Sie im Abschnitt „Listener für Nachrichten und Ausnahmebedingungen in .NET” auf Seite 53.

Exceptions:

- XMSException

MessageSelector - Nachrichtenselektor abrufen

Schnittstelle:

```
String MessageSelector
{
  get;
}
```

Abrufen des Nachrichtenselektors für den Nachrichtenkonsumenten. Der Rückgabewert ist ein Zeichenfolgeobjekt, das den Nachrichtenselektorausdruck enthält. Wenn eine Datenkonvertierung erforderlich ist, gibt dieser Wert den Nachrichtenselektorausdruck nach der Konvertierung an. Wenn der Nachrichtenkonsument keinen Nachrichtenselektor besitzt, ist der Wert von MessageSelector ein Zeichenfolgeobjekt mit einem Nullwert.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
Close	Schließen des Nachrichtenkonsumenten.
Receive	Empfangen Sie die nächste Nachricht für den Nachrichtenkonsumenten. Der Aufruf wartet auf unbestimmte Zeit auf eine Nachricht oder bis der Nachrichtenkonsument geschlossen wird.
Receive	Empfangen Sie die nächste Nachricht für den Nachrichtenkonsumenten. Der Aufruf wartet nur eine angegebene Zeit lang auf eine Nachricht oder bis der Nachrichtenkonsument geschlossen wird.
ReceiveNoWait	Empfangen der nächsten Nachricht für den Nachrichtenkonsumenten, falls sofort eine Nachricht verfügbar ist.

Close - Nachrichtenkonsumenten schließen

Schnittstelle:

```
void  Close();
```

Schließen des Nachrichtenkonsumenten.

Wenn eine Anwendung versucht, einen Nachrichtenkonsumenten zu schließen, der bereits geschlossen ist, wird der Aufruf ignoriert.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

Receive - Empfangen

Schnittstelle:

```
IMessage Receive();
```

Empfangen Sie die nächste Nachricht für den Nachrichtenkonsumenten. Der Aufruf wartet auf unbestimmte Zeit auf eine Nachricht oder bis der Nachrichtenkonsument geschlossen wird.

Parameter:

--

Rückgabe:

Ein Zeiger auf das Message-Objekt. Wenn der Nachrichtenkonsument geschlossen wird, während der Aufruf auf eine Nachricht wartet, gibt die Methode einen Zeiger auf ein Message-Objekt mit einem Nullwert zurück.

Exceptions:

- XMSException

Receive - Empfangen (mit einem Warteintervall)

Schnittstelle:

```
IMessage Receive(Int64 delay);
```

Empfangen Sie die nächste Nachricht für den Nachrichtenkonsumenten. Der Aufruf wartet nur eine angegebene Zeit lang auf eine Nachricht oder bis der Nachrichtenkonsument geschlossen wird.

Parameter:

delay (Eingabe)

Die Zeit in Millisekunden, die der Aufruf auf eine Nachricht wartet. Wenn Sie ein Warteintervall von 0 angeben, wartet der Aufruf auf unbestimmte Zeit auf eine Nachricht.

Rückgabe:

Ein Zeiger auf das Message-Objekt. Wenn während des Warteintervalls keine Nachricht eingeht oder der Nachrichtenkonsument geschlossen wird, während der Aufruf auf eine Nachricht wartet, gibt die Methode einen Zeiger auf ein Message-Objekt mit einem Nullwert zurück, löst aber keine Ausnahme aus.

Exceptions:

- XMSException

ReceiveNoWait - Empfangen ohne Wartezeit

Schnittstelle:

```
IMessage ReceiveNoWait();
```

Empfangen der nächsten Nachricht für den Nachrichtenkonsumenten, falls sofort eine Nachricht verfügbar ist.

Parameter:

--

Rückgabe:

Ein Zeiger auf ein Message-Objekt. Wenn nicht sofort eine Nachricht verfügbar ist, gibt die Methode einen Zeiger auf ein Message-Objekt mit einem Nullwert zurück.

Exceptions:

- XMSException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

MessageEOFException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageEOFException
```

XMS gibt diese Ausnahmebedingung aus, wenn XMS beim Lesen des Hauptteils der Bytenachrichten durch die Anwendung das Ende eines Bytenachrichtendatenstroms ermittelt.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

MessageFormatException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageFormatException
```

XMS gibt diese Ausnahmebedingung aus, wenn XMS eine Nachricht mit einem ungültigen Format ermittelt.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

IMessageListener (Delegat)

Vererbungshierarchie:

--

Eine Anwendung verwendet einen Nachrichtenlistener, um Nachrichten asynchron zu empfangen.

Delegat

Zusammenfassung der Methoden:

Methode	Beschreibung
MessageListener	Asynchrones Zustellen einer Nachricht an den Nachrichtenkonsumenten.

MessageListener - Nachrichtenlistener

Schnittstelle:

```
public delegate void MessageListener(IMessage msg);
```

Asynchrones Zustellen einer Nachricht an den Nachrichtenkonsumenten.

Methoden, die dieses Delegat implementieren, können für die Verbindung registriert werden.

Weitere Informationen zur Verwendung von Nachrichtenlisteners finden Sie im Abschnitt „Listener für Nachrichten und Ausnahmebedingungen in .NET" auf Seite 53.

Parameter:

mesg (Eingabe)
Das Message-Objekt.

Rückgabe:
Void

MessageNotReadableException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotReadableException
```

XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung den Hauptteil einer Nachricht lesen will, die lesegeschützt ist.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

MessageNotWritableException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotWritableException
```

XMS gibt diese Ausnahmebedingung aus, wenn eine Anwendung in den Hauptteil einer Nachricht schreiben will, die schreibgeschützt ist.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

IMessageProducer

Eine Anwendung verwendet einen Nachrichtenproduzenten (Message Producer), um Nachrichten an ein Ziel zu senden.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageProducer
```

Eine Liste der XMS-definierten Eigenschaften eines MessageProducer-Objekts finden Sie im Abschnitt „Eigenschaften von MessageProducer” auf Seite 207.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaften	Beschreibung
DeliveryMode	Abrufen und Einstellen des Standardzustellmodus für Nachrichten, die vom Nachrichtenproduzenten gesendet werden.
Destination	Abrufen des Ziels für den Nachrichtenproduzenten.
DisableMsgID	Abrufen einer Angabe, ob für eine empfangende Anwendung Nachrichten-IDs in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen, und Angeben, ob für eine empfangende Anwendung Nachrichten-IDs in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.
DisableMsgTS	Abrufen einer Angabe, ob für eine empfangende Anwendung Zeitmarken in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen, und Angeben, ob für eine empfangende Anwendung Zeitmarken in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.
Priority	Abrufen und Festlegen der Standardpriorität für Nachrichten, die vom Nachrichtenproduzenten gesendet werden.
TimeToLive	Abrufen und Festlegen der Standarddauer der Existenz einer Nachricht, bevor sie verfällt.

DeliveryMode - Standardzustellmodus abrufen und einstellen

Schnittstelle:

```
DeliveryMode DeliveryMode
{
  get;
  set;
}
```

Abrufen und Einstellen des Standardzustellmodus für Nachrichten, die vom Nachrichtenproduzenten gesendet werden.

Der Standardzustellmodus hat einen der folgenden Werte:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Für eine Echtzeitverbindung zu einem Broker muss der Wert `DeliveryMode.NonPersistent` angegeben werden.

Der Standardwert ist `DeliveryMode.Persistent`, außer für eine Echtzeitverbindung zu einem Broker, für die `DeliveryMode.NonPersistent` der Standardwert ist.

Exceptions:

- XMSException

Destination - Ziel abrufen

Schnittstelle:

```
IDestination Destination
{
  get;
}
```

Abrufen des Ziels für den Nachrichtenproduzenten.

Parameter:
--

Rückgabe:
Das Zielobjekt (Destination). Verfügt der Nachrichtenproduzent nicht über ein Ziel, gibt die Methode ein Zielobjekt mit einem Nullwert zurück.

Exceptions:

- XMSException

DisableMsgID - Flag für Nachrichten-ID-Inaktivierung abrufen und setzen

Schnittstelle:

```
Boolean DisableMessageID
{
  get;
  set;
}
```

Abrufen einer Angabe, ob für eine empfangende Anwendung Nachrichten-IDs in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen, und Angeben, ob für eine empfangende Anwendung Nachrichten-IDs in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.

Bei einer Verbindung zu einem Warteschlangenmanager oder bei einer Echtzeitverbindung zu einem Broker wird dieses Flag ignoriert. Bei einer Verbindung zu einem Service Integration Bus wird das Flag berücksichtigt.

DisabledMsgID hat folgende Werte:

- `True`, wenn für eine empfangende Anwendung keine Nachrichten-IDs in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.
- `False`, wenn für eine empfangende Anwendung Nachrichten-IDs in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.

Exceptions:

- XMSException

DisableMsgTS - Flag für Zeitmarkeninaktivierung abrufen und setzen

Schnittstelle:

```
Boolean DisableMessageTimestamp
{
  get;
  set;
}
```

Abrufen einer Angabe, ob für eine empfangende Anwendung Zeitmarken in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen, und Angeben, ob für eine empfangende Anwendung Zeitmarken in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.

Bei einer Echtzeitverbindung zu einem Broker wird dieses Flag ignoriert. Bei einer Verbindung zu einem Warteschlangenmanager oder bei einer Verbindung zu einem Service Integration Bus wird das Flag berücksichtigt.

DisableMsgTS hat folgende Werte:

- `True`, wenn für eine empfangende Anwendung keine Zeitmarken in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.
- `False`, wenn für eine empfangende Anwendung Zeitmarken in Nachrichten, die vom Nachrichtenproduzenten gesendet werden, enthalten sein müssen.

Rückgabe:

Exceptions:

- XMSException

Priority - Standardpriorität abrufen und festlegen

Schnittstelle:

```
Int32 Priority
{
  get;
  set;
}
```

Abrufen und Festlegen der Standardpriorität für Nachrichten, die vom Nachrichtenproduzenten gesendet werden.

Der Wert der Standardnachrichtenpriorität ist eine Ganzzahl im Bereich von 0 (niedrigste Priorität) bis 9 (höchste Priorität).

Bei einer Echtzeitverbindung zu einem Broker wird die Priorität einer Nachricht ignoriert.

Exceptions:

- XMSException

TimeToLive - Standardlebensdauer abrufen und festlegen

Schnittstelle:

```
Int64 TimeToLive
{
  get;
  set;
}
```

Abrufen und Festlegen der Standarddauer der Existenz einer Nachricht, bevor sie verfällt.

Die Zeit wird ab dem Zeitpunkt gemessen, an dem der Nachrichtenproduzent die Nachricht sendet, und gibt die Standardlebensdauer in Millisekunden an. Der Wert 0 bedeutet, dass eine Nachricht nie verfällt.

Bei einer Echtzeitverbindung zu einem Broker ist dieser Wert immer 0.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
Close	Schließen des Nachrichtenproduzenten.

Methode	Beschreibung
Send	Senden Sie eine Nachricht an das Ziel, das angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des Standardzustellmodus und der standardmäßigen Priorität und Lebensdauer des Nachrichtenproduzenten gesendet.
Send	Senden Sie eine Nachricht an das Ziel, das angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des angegebenen Zustellmodus und der angegebenen Priorität und Lebensdauer gesendet.
Send	Senden Sie eine Nachricht an ein angegebenes Ziel, wenn Sie einen Nachrichtenproduzenten verwenden, für den kein Ziel angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des Standardzustellmodus und der standardmäßigen Priorität und Lebensdauer des Nachrichtenproduzenten gesendet.
Send	Senden Sie eine Nachricht an ein angegebenes Ziel, wenn Sie einen Nachrichtenproduzenten verwenden, für den kein Ziel angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des angegebenen Zustellmodus und der angegebenen Priorität und Lebensdauer gesendet.

Close - Nachrichtenproduzent schließen

Schnittstelle:

```
void  Close();
```

Schließen des Nachrichtenproduzenten.

Wenn eine Anwendung versucht, einen Nachrichtenproduzenten zu schließen, der bereits geschlossen ist, wird der Aufruf ignoriert.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

Send - Senden

Schnittstelle:

```
void Send(IMessage msg) ;
```

Senden Sie eine Nachricht an das Ziel, das angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des Standardzustellmodus und der standardmäßigen Priorität und Lebensdauer des Nachrichtenproduzenten gesendet.

Parameter:

msg (Eingabe)
Das Message-Objekt.

Rückgabe:
Void

Exceptions:

- XMSException

- MessageFormatException
- InvalidDestinationException

Send - Senden (unter Angabe von Zustellmodus, Priorität und Lebensdauer)

Schnittstelle:

```
void Send(IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive);
```

Senden Sie eine Nachricht an das Ziel, das angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des angegebenen Zustellmodus und der angegebenen Priorität und Lebensdauer gesendet.

Parameter:

msg (Eingabe)
Das Message-Objekt.

deliveryMode (Eingabe)
Der Zustellmodus für die Nachricht, wobei es sich um einen der folgenden Werte handeln muss:

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

Für eine Echtzeitverbindung zu einem Broker muss der Wert `DeliveryMode.NonPersistent` angegeben werden.

priority (Eingabe)
Die Priorität der Nachricht. Der Wert kann eine Ganzzahl im Bereich von 0 (niedrigste Priorität) bis 9 (höchste Priorität) sein. Bei einer Echtzeitverbindung zu einem Broker wird der Wert ignoriert.

timeToLive (Eingabe)
Die Lebensdauer der Nachricht in Millisekunden. Der Wert 0 bedeutet, dass die Nachricht nie verfällt. Bei einer Echtzeitverbindung zu einem Broker muss der Wert 0 sein.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Send - Senden (an ein angegebenes Ziel)

Schnittstelle:

```
void Send(IDestination dest, IMessage msg) ;
```

Senden Sie eine Nachricht an ein angegebenes Ziel, wenn Sie einen Nachrichtenproduzenten verwenden, für den kein Ziel angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des Standardzustellmodus und der standardmäßigen Priorität und Lebensdauer des Nachrichtenproduzenten gesendet.

Normalerweise geben Sie beim Erstellen eines Nachrichtenproduzenten ein Ziel an, aber falls nicht, müssen Sie jedes Mal, wenn Sie eine Nachricht senden, ein Ziel angeben.

Parameter:

dest (Eingabe)
Das Zielobjekt (Destination).

msg (Eingabe)
Das Message-Objekt.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageFormatException
- InvalidDestinationException

Send - Senden (an ein angegebenes Ziel; unter Angabe eines Zustellmodus, einer Priorität und einer Lebensdauer)

Schnittstelle:

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

Senden Sie eine Nachricht an ein angegebenes Ziel, wenn Sie einen Nachrichtenproduzenten verwenden, für den kein Ziel angegeben wurde, als der Nachrichtenproduzent erstellt wurde. Die Nachricht wird unter Verwendung des angegebenen Zustellmodus und der angegebenen Priorität und Lebensdauer gesendet.

Normalerweise geben Sie beim Erstellen eines Nachrichtenproduzenten ein Ziel an, aber falls nicht, müssen Sie jedes Mal, wenn Sie eine Nachricht senden, ein Ziel angeben.

Parameter:

dest (Eingabe)
Das Zielobjekt (Destination).

msg (Eingabe)
Das Message-Objekt.

deliveryMode (Eingabe)
Der Zustellmodus für die Nachricht, wobei es sich um einen der folgenden Werte handeln muss:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Für eine Echtzeitverbindung zu einem Broker muss der Wert `DeliveryMode.NonPersistent` angegeben werden.

priority (Eingabe)
Die Priorität der Nachricht. Der Wert kann eine Ganzzahl im Bereich von 0 (niedrigste Priorität) bis 9 (höchste Priorität) sein. Bei einer Echtzeitverbindung zu einem Broker wird der Wert ignoriert.

timeToLive (Eingabe)
Die Lebensdauer der Nachricht in Millisekunden. Der Wert 0 bedeutet, dass die Nachricht nie verfällt. Bei einer Echtzeitverbindung zu einem Broker muss der Wert 0 sein.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IObjectMessage

Eine Objektnachricht ist eine Nachricht, deren Hauptteil ein serialisiertes Java -oder .NET-Objekt umfasst.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IObjectMessage
```

Zugehörige Verweise

Objektnachrichten

Der Hauptteil einer Objektnachricht enthält ein serialisiertes Java- oder .NET-Objekt.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
Object	Abrufen und Festlegen des Objekts, das den Hauptteil der Objektnachricht bildet.

Object - Objekt als Bytes abrufen und festlegen

Schnittstelle:

```
System.Object Object
    {
      get;
      set;
    }

    Byte[] GetObject();
```

Abrufen und Festlegen des Objekts, das den Hauptteil der Objektnachricht bildet.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException
- MessageNotWritableException

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden von der Schnittstelle IMessage übernommen:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Die folgenden Methoden werden aus der Schnittstelle IMessage übernommen:

clearBody, clearProperties, PropertyExists

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IPropertyContext

IPropertyContext ist eine abstrakte Superklasse, die Methoden enthält, die Eigenschaften abrufen und festlegen. Diese Methoden werden von anderen Klassen geerbt.

Vererbungshierarchie:

--

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
GetBooleanProperty	Abrufen des Werts der booleschen Eigenschaft mit dem angegebenen Namen.
GetByteProperty	Abrufen des Werts der Byteeigenschaft mit dem angegebenen Namen.
GetBytesProperty	Abrufen des Werts der Byte-Array-Eigenschaft mit dem angegebenen Namen.
GetCharProperty	Abrufen des Werts der 2-Byte-Zeicheneigenschaft mit dem angegebenen Namen.
GetDoubleProperty	Abrufen des Werts der Gleitkommaeigenschaft mit doppelter Genauigkeit mit dem angegebenen Namen.
GetFloatProperty	Abrufen des Werts der Gleitkommaeigenschaft mit dem angegebenen Namen.
GetIntProperty	Abrufen des Werts der ganzzahligen Eigenschaft mit dem angegebenen Namen.
GetLongProperty	Abrufen des Werts der Long-Integer-Eigenschaft (erweiterte Ganzahl) mit dem angegebenen Namen.
GetObjectProperty	Abrufen des Werts und Datentyps der Eigenschaft mit dem angegebenen Namen.
GetShortProperty	Abrufen des Werts der Short-Integer-Eigenschaft (kurze Ganzahl) mit dem angegebenen Namen.
GetStringProperty	Abrufen des Werts der Zeichenfolgeeigenschaft mit dem angegebenen Namen.
SetBooleanProperty	Festlegen des Werts der booleschen Eigenschaft mit dem angegebenen Namen.
SetByteProperty	Festlegen des Werts der Byteeigenschaft mit dem angegebenen Namen.
SetBytesProperty	Festlegen des Werts der Byte-Array-Eigenschaft mit dem angegebenen Namen.
SetCharProperty	Festlegen des Werts der 2-Byte-Zeicheneigenschaft mit dem angegebenen Namen.
SetDoubleProperty	Festlegen des Werts der Gleitkommaeigenschaft mit doppelter Genauigkeit mit dem angegebenen Namen.

Methode	Beschreibung
SetFloatProperty	Festlegen des Werts der Gleitkommaeigenschaft mit dem angegebenen Namen.
SetIntProperty	Festlegen des Werts der ganzzahligen Eigenschaft mit dem angegebenen Namen.
SetLongProperty	Festlegen des Werts der Long-Integer-Eigenschaft (erweiterte Ganzahl) mit dem angegebenen Namen.
SetObjectProperty	Festlegen des Werts und Datentyps einer Eigenschaft mit dem angegebenen Namen.
SetShortProperty	Festlegen des Werts der Short-Integer-Eigenschaft (kurze Ganzahl) mit dem angegebenen Namen.
SetStringProperty	Festlegen des Werts der Zeichenfolgeeigenschaft mit dem angegebenen Namen.

GetBooleanProperty - Boolesche Eigenschaft abrufen

Schnittstelle:

```
Boolean GetBooleanProperty(String property_name);
```

Abrufen des Werts der booleschen Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetByteProperty - Byteeigenschaft abrufen

Schnittstelle:

```
Byte    GetByteProperty(String property_name) ;
Int16   GetSignedByteProperty(String property_name) ;
```

Abrufen des Werts der Byteeigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetBytesProperty - Array-Eigenschaft abrufen

Schnittstelle:

```
Byte[]  GetBytesProperty(String property_name) ;
```

Abrufen des Werts der Byte-Array-Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Die Anzahl der Bytes im Array.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetCharProperty - Zeicheneigenschaft abrufen

Schnittstelle:

```
Char    GetCharProperty(String property_name) ;
```

Abrufen des Werts der 2-Byte-Zeicheneigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetDoubleProperty - Gleitkommaeigenschaft mit doppelter Genauigkeit abrufen

Schnittstelle:

```
Double  GetDoubleProperty(String property_name) ;
```

Abrufen des Werts der Gleitkommaeigenschaft mit doppelter Genauigkeit mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetFloatProperty - Gleitkommaeigenschaft abrufen

Schnittstelle:

```
Single  GetFloatProperty(String property_name) ;
```

Abrufen des Werts der Gleitkommaeigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetIntProperty - Ganzzahlige Eigenschaft abrufen

Schnittstelle:

```
Int32   GetIntProperty(String property_name) ;
```

Abrufen des Werts der ganzzahligen Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetLongProperty - Long-Integer-Eigenschaft abrufen

Schnittstelle:

```
Int64   GetLongProperty(String property_name) ;
```

Abrufen des Werts der Long-Integer-Eigenschaft (erweiterte Ganzahl) mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetObjectProperty - Objekteigenschaft abrufen

Schnittstelle:

```
Object  GetObjectProperty( String property_name) ;
```

Abrufen des Werts und Datentyps der Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft, wobei es sich um einen der folgenden Objekttypen handelt:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetShortProperty - Short-Integer-Eigenschaft abrufen

Schnittstelle:

```
Int16   GetShortProperty(String property_name) ;
```

Abrufen des Werts der Short-Integer-Eigenschaft (kurze Ganzahl) mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Der Wert der Eigenschaft.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

GetStringProperty - Zeichenfolgeeigenschaft abrufen

Schnittstelle:

```
String  GetStringProperty(String property_name) ;
```

Abrufen des Werts der Zeichenfolgeeigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

Rückgabe:
Ein Zeichenfolgeobjekt, das die Zeichenfolge enthält, die dem Wert der Eigenschaft entspricht. Wenn eine Datenkonvertierung erforderlich ist, ist dieser Wert die Zeichenfolge nach der Konvertierung.

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException

SetBooleanProperty - Boolesche Eigenschaft festlegen

Schnittstelle:

```
void SetBooleanProperty( String property_name,  Boolean value) ;
```

Festlegen des Werts der booleschen Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetByteProperty - Byteeigenschaft festlegen

Schnittstelle:

```
void SetByteProperty( String property_name,  Byte value) ;
      void SetSignedByteProperty( String property_name,  Int16 value) ;
```

Festlegen des Werts der Byteeigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetBytesProperty - Array-Eigenschaft festlegen

Schnittstelle:

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

Festlegen des Werts der Byte-Array-Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft, wobei es sich um ein Byte-Array handelt.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetCharProperty - Zeicheneigenschaft festlegen

Schnittstelle:

```
void SetCharProperty( String property_name,  Char value) ;
```

Festlegen des Werts der 2-Byte-Zeicheneigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetDoubleProperty - Gleitkommaeigenschaft mit doppelter Genauigkeit festlegen

Schnittstelle:

```
void SetDoubleProperty( String property_name,  Double value) ;
```

Festlegen des Werts der Gleitkommaeigenschaft mit doppelter Genauigkeit mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetFloatProperty - Gleitkommaeigenschaft festlegen

Schnittstelle:

```
void SetFloatProperty( String property_name,  Single value) ;
```

Festlegen des Werts der Gleitkommaeigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetIntProperty - Ganzzahlige Eigenschaft festlegen

Schnittstelle:

```
void SetIntProperty( String property_name,  Int32 value) ;
```

Festlegen des Werts der ganzzahligen Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetLongProperty - Long-Integer-Eigenschaft festlegen

Schnittstelle:

```
void SetLongProperty( String property_name,  Int64 value) ;
```

Festlegen des Werts der Long-Integer-Eigenschaft (erweiterte Ganzahl) mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetObjectProperty - Objekteigenschaft festlegen

Schnittstelle:

```
void SetObjectProperty( String property_name,  Object value) ;
```

Festlegen des Werts und Datentyps einer Eigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

objectType (Eingabe)
Der Wert der Eigenschaft, wobei es sich um einen der folgenden Objekttypen handeln muss:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value (Eingabe)
Der Wert der Eigenschaft als ein Byte-Array.

length (Eingabe)
Die Anzahl der Bytes im Array.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetShortProperty - Short-Integer-Eigenschaft festlegen

Schnittstelle:

```
void SetShortProperty( String property_name,  Int16 value) ;
```

Festlegen des Werts der Short-Integer-Eigenschaft (kurze Ganzahl) mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Der Wert der Eigenschaft.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

SetStringProperty - Zeichenfolgeeigenschaft festlegen

Schnittstelle:

```
void SetStringProperty( String property_name,  String value);
```

Festlegen des Werts der Zeichenfolgeeigenschaft mit dem angegebenen Namen.

Parameter:

property_name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Eigenschaft angibt.

value (Eingabe)
Ein Zeichenfolgeobjekt, das die Zeichenfolge enthält, die dem Wert der Eigenschaft entspricht.

Rückgabe:
Void

Threadkontext:
Wird durch die Unterklasse bestimmt.

Exceptions:

- XMSException
- MessageNotWritableException

IQueueBrowser

Eine Anwendung verwendet einen Warteschlangenbrowser, um Nachrichten in einer Warteschlange anzuzeigen, ohne sie zu entfernen.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
```

```
   |
   +----IBM.XMS.IQueueBrowser
```

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
MessageSelector	Abrufen des Nachrichtenselektors für den Warteschlangenbrowser.
Queue	Abrufen der Warteschlange, die dem Warteschlangenbrowser zugeordnet ist, als Zielobjekt, das die Warteschlange darstellt.

MessageSelector - Nachrichtenselektor abrufen

Schnittstelle:

```
String MessageSelector
{
  get;
}
```

Abrufen des Nachrichtenselektors für den Warteschlangenbrowser.

Der Nachrichtenselektor ist ein Zeichenfolgeobjekt, das den Nachrichtenselektorausdruck enthält. Wenn eine Datenkonvertierung erforderlich ist, gibt dieser Wert den Nachrichtenselektorausdruck nach der Konvertierung an. Wenn der Warteschlangenbrowser keinen Nachrichtenselektor besitzt, gibt die Methode ein Zeichenfolgeobjekt mit einem Nullwert.

Exceptions:

- XMSException

Queue - Warteschlange abrufen

Schnittstelle:

```
IDestination Queue
{
  get;
}
```

Abrufen der Warteschlange, die dem Warteschlangenbrowser zugeordnet ist, als Zielobjekt, das die Warteschlange darstellt.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
Close	Schließen des Warteschlangenbrowsers.
GetEnumerator	Abrufen einer Liste der Nachrichten in der Warteschlange.

Close - Warteschlangenbrowser schließen

Schnittstelle:

```
void  Close();
```

Schließen des Warteschlangenbrowsers.

Wenn eine Anwendung versucht, einen Warteschlangenbrowser zu schließen, der bereits geschlossen ist, wird der Aufruf ignoriert.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException

GetEnumerator - Nachrichten abrufen

Schnittstelle:

```
IEnumerator GetEnumerator();
```

Abrufen einer Liste der Nachrichten in der Warteschlange.

Die Methode gibt einen Aufzählungsausdruck zurück, der eine Liste der Nachrichtenobjekte ('Message') enthält. Die Reihenfolge der Message-Objekte ist mit der Reihenfolge identisch, in der die Nachrichten aus der Warteschlange abgerufen würden. Die Anwendung kann dann den Aufzählungsausdruck dazu verwenden, alle Nachrichten einzeln nacheinander zu durchsuchen.

Der Aufzählungsausdruck wird dynamisch aktualisiert, wenn Nachrichten in die Warteschlange eingereiht bzw. daraus entfernt werden. Jedes Mal, wenn die Anwendung die Methode 'IEnumerator.MoveNext()' zum Durchsuchen der nächsten Nachricht in der Warteschlange aufruft, spiegelt die Nachricht den aktuellen Inhalt der Warteschlange wider.

Wenn eine Anwendung diese Methode mehrmals für einen Warteschlangenbrowser aufruft, wird bei jedem Aufruf ein neuer Aufzählungsausdruck zurückgegeben. Daher kann eine Anwendung mehrere Aufzählungsausdrücke zum Durchsuchen einer Warteschlange verwenden und dadurch mehrere Positionen in der Warteschlange kennzeichnen.

Parameter:
--

Rückgabe:
Das Iterator-Objekt.

Exceptions:

- XMSException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

Requestor

Eine Anwendung verwendet einen Requestor (Anforderer), um eine Anforderungsnachricht zu senden und dann auf die Antwort zu warten und sie zu empfangen.

Vererbungshierarchie:
--

Konstruktoren

Zusammenfassung der Konstruktoren:

Konstruktor	Beschreibung
Requestor	Erstellen eines Requestors.

Requestor - Requestor erstellen

Schnittstelle:

```
Requestor(ISession sess, IDestination dest);
```

Erstellen eines Requestors.

Parameter:

sess (Eingabe)
Ein Sitzungsobjekt. Die Sitzung darf keine Sitzung mit Transaktionsunterstützung sein und muss einen der folgenden Bestätigungsmodi haben:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

dest (Eingabe)
Ein Zielobjekt, das das Ziel angibt, an das die Anwendung Anforderungsnachrichten senden kann.

Threadkontext:
Die Sitzung, die dem Requestor zugeordnet ist.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
Close	Schließen des Requestors.
Request	Senden einer Anforderungsnachricht und dann Warten auf eine Antwort von der Anwendung, die die Anforderungsnachricht erhält, und Empfangen der Antwort.

Close - Requestor schließen

Schnittstelle:

```
void  Close();
```

Schließen des Requestors.

Wenn eine Anwendung versucht, einen Requestor zu schließen, der bereits geschlossen ist, wird der Aufruf ignoriert.

Anmerkung: Wenn eine Anwendung einen Requestor schließt, wird die zugehörige Sitzung nicht ebenfalls geschlossen. In dieser Hinsicht verhält sich XMS anders als JMS.

Parameter:
--

Rückgabe:
Void

Threadkontext:
Alle

Exceptions:

- XMSException

Request - Antwort anfordern

Schnittstelle:

```
IMessage Request(IMessage requestMessage);
```

Senden einer Anforderungsnachricht und dann Warten auf eine Antwort von der Anwendung, die die Anforderungsnachricht erhält, und Empfangen der Antwort.

Ein Aufruf dieser Methode bewirkt eine Blockierung, bis eine Antwort empfangen oder die Sitzung beendet wird, je nachdem, was früher eintritt.

Parameter:

requestMessage (Eingabe)
Das Nachrichtenobjekt, das die Anforderungsnachricht enthält.

Rückgabe:
Ein Zeiger auf das Nachrichtenobjekt, das die Antwortnachricht enthält.

Threadkontext:
Die Sitzung, die dem Requestor zugeordnet ist.

Exceptions:

- XMSException

ResourceAllocationException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.ResourceAllocationException
```

XMS gibt diese Ausnahmebedingung aus, wenn XMS die für eine Methode erforderlichen Ressourcen nicht zuordnen kann.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

SecurityException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.SecurityException
```

XMS löst diese Ausnahme aus, wenn die Benutzer-ID und das Kennwort, die zur Authentifizierung einer Anwendung übergeben wurden, zurückgewiesen werden. XMS löst diese Ausnahme auch aus, wenn eine Berechtigungsprüfung fehlschlägt und verhindert, dass eine Methode abgeschlossen wird.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

ISession

Eine Sitzung (Session) ist ein Einzelthreadkontext zum Senden und Empfangen von Nachrichten.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.ISession
```

Eine Liste der XMS-definierten Eigenschaften eines Session-Objekts finden Sie im Abschnitt „Eigenschaften von Session" auf Seite 207.

.NET-Eigenschaften

Übersicht .NET-Eigenschaften:

.NET-Eigenschaft	Beschreibung
AcknowledgeMode	Abrufen des Bestätigungsmodus für die Sitzung.
Transacted	Bestimmen, ob es sich um eine Sitzung mit Transaktionsunterstützung handelt.

AcknowledgeMode - Bestätigungsmodus abrufen

Schnittstelle:

```
AcknowledgeMode AcknowledgeMode
    {
      get;
    }
```

Abrufen des Bestätigungsmodus für die Sitzung.

Der Bestätigungsmodus wird beim Erstellen der Sitzung angegeben.

Sofern die Sitzung keine Sitzung mit Transaktionsunterstützung ist, hat der Bestätigungsmodus einen der folgenden Werte:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Weitere Informationen zu den Bestätigungsmodi finden Sie im Abschnitt „Nachrichtenbestätigung" auf Seite 29.

Eine Sitzung mit Transaktionsunterstützung hat keinen Bestätigungsmodus. Wenn die Sitzung transaktionsbasiert ist, gibt die Methode stattdessen `AcknowledgeMode.SessionTransacted` zurück.

Exceptions:

- XMSException

Transacted - Bestimmen, ob Transaktionen unterstützt werden

Schnittstelle:

```
Boolean Transacted
    {
      get;
    }
```

Bestimmen, ob es sich um eine Sitzung mit Transaktionsunterstützung handelt.

Folgende Werte werden zurückgegeben:

- `True` für eine Sitzung mit Transaktionsunterstützung.
- `False` für eine Sitzung ohne Transaktionsunterstützung.

Bei einer Echtzeitverbindung zu einem Broker gibt die Methode immer `False` zurück.

Exceptions:

- XMSException

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
Close	Schließen der Sitzung.
Commit	Festschreiben aller Nachrichten, die in der aktuellen Transaktion verarbeitet werden.
CreateBrowser	Erstellen eines Warteschlangenbrowsers für die angegebene Warteschlange.
CreateBrowser	Erstellen eines Warteschlangenbrowsers für die angegebene Warteschlange unter Verwendung eines Nachrichtenselektors.
CreateBytesMessage	Erstellen einer Bytesnachricht.
CreateConsumer	Erstellen eines Nachrichtenkonsumenten für das angegebene Ziel.
CreateConsumer	Erstellen eines Nachrichtenkonsumenten für das angegebene Ziel unter Verwendung eines Nachrichtenselektors.
CreateConsumer	Erstellen eines Nachrichtenkonsumenten für das angegebene Ziel unter Verwendung eines Nachrichtenselektors und, wenn das Ziel ein Thema ist, mit der Angabe, ob der Nachrichtenkonsument die von seiner eigenen Verbindung veröffentlichten Nachrichten empfängt.
CreateDurableSubscriber	Erstellen eines permanenten Subskribenten für das angegebene Thema.
CreateDurableSubscriber	Erstellen eines permanenten Subskribenten für das angegebene Thema unter Verwendung eines Nachrichtenselektors und mit der Angabe, ob der permanente Subskribent die von seiner eigenen Verbindung veröffentlichten Nachrichten empfängt.
CreateMapMessage	Erstellen einer Zuordnungsnachricht.
CreateMessage	Erstellen einer Nachricht, die keinen Hauptteil hat.
CreateObjectMessage	Erstellen einer Objektnachricht.
CreateProducer	Erstellen eines Nachrichtenproduzenten, um Nachrichten an das angegebene Ziel zu senden.
CreateQueue	Erstellen eines Zielobjekt, das eine Warteschlange im Messaging-Server darstellt.
CreateStreamMessage	Erstellen einer Datenstromnachricht.
CreateTemporaryQueue	Erstellen einer temporären Warteschlange.
CreateTemporaryTopic	Erstellen eines temporären Themas.
CreateTextMessage	Erstellen einer Textnachricht mit einem leeren Hauptteil.
CreateTextMessage	Erstellen einer Textnachricht, deren Hauptteil mit dem angegebenen Text initialisiert wird.
CreateTopic	Erstellen eines Zielobjekts, um ein Thema darzustellen.
Recover	Wiederherstellen der Sitzung.

Methode	Beschreibung
Rollback	Rückgängigmachen aller Nachrichten, die in der aktuellen Transaktion verarbeitet wurden.
Unsubscribe	Löschen einer permanenten Subskription.

Close - Sitzung schließen

Schnittstelle:

```
void Close();
```

Schließen der Sitzung. Bei einer Sitzung mit Transaktionsunterstützung werden alle in Bearbeitung befindlichen Transaktionen rückgängig gemacht.

Wenn eine Anwendung versucht, eine Sitzung zu schließen, der bereits geschlossen ist, wird der Aufruf ignoriert.

Parameter:
--

Rückgabe:
Void

Threadkontext:
Alle

Exceptions:

- XMSException

Commit - Festschreiben

Schnittstelle:

```
void Commit();
```

Festschreiben aller Nachrichten, die in der aktuellen Transaktion verarbeitet werden.

Es muss sich um eine Sitzung mit Transaktionsunterstützung handeln.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException
- IllegalStateException
- TransactionRolledBackException

CreateBrowser - Warteschlangenbrowser erstellen

Schnittstelle:

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

Erstellen eines Warteschlangenbrowsers für die angegebene Warteschlange.

Parameter:

queue (Eingabe)
Ein Zielobjekt, das die Warteschlange angibt.

Rückgabe:
Das QueueBrowser-Objekt.

Exceptions:

- XMSException
- InvalidDestinationException

CreateBrowser - Warteschlangenbrowser erstellen (mit Nachrichtenselektor)

Schnittstelle:

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

Erstellen eines Warteschlangenbrowsers für die angegebene Warteschlange unter Verwendung eines Nachrichtenselektors.

Parameter:

queue (Eingabe)
Ein Zielobjekt, das die Warteschlange angibt.

selector (Eingabe)
Ein Zeichenfolgeobjekt, das einen Nachrichtenselektorausdruck enthält. Es werden nur die Nachrichten mit Eigenschaften, die mit dem Nachrichtenselektorausdruck übereinstimmen, an den Warteschlangenbrowser übergeben.

Ein leeres Zeichenfolgeobjekt bedeutet, dass es keinen Nachrichtenselektor für den Warteschlangenbrowser gibt.

Rückgabe:
Das QueueBrowser-Objekt.

Exceptions:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateBytesMessage - Bytesnachricht erstellen

Schnittstelle:

```
IBytesMessage CreateBytesMessage();
```

Erstellen einer Bytesnachricht.

Parameter:
--

Rückgabe:
Das BytesMessage-Objekt.

Exceptions:

- XMSException
- IllegalStateException (Sitzung ist geschlossen)

CreateConsumer - Konsumenten erstellen

Schnittstelle:

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

Erstellen eines Nachrichtenkonsumenten für das angegebene Ziel.

Parameter:

dest (Eingabe)
Das Zielobjekt (Destination).

Rückgabe:
Das MessageConsumer-Objekt.

Exceptions:

- XMSException
- InvalidDestinationException

CreateConsumer - Konsumenten erstellen (mit Nachrichtenselektor)

Schnittstelle:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector) ;
```

Erstellen eines Nachrichtenkonsumenten für das angegebene Ziel unter Verwendung eines Nachrichtenselektors.

Parameter:

dest (Eingabe)
Das Zielobjekt (Destination).

selector (Eingabe)
Ein Zeichenfolgeobjekt, das einen Nachrichtenselektorausdruck enthält. Es werden nur die Nachrichten mit Eigenschaften, die mit dem Nachrichtenselektorausdruck übereinstimmen, an den Nachrichtenkonsumenten übergeben.

Ein leeres Zeichenfolgeobjekt bedeutet, dass es keinen Nachrichtenselektor für den Nachrichtenkonsumenten gibt.

Rückgabe:
Das MessageConsumer-Objekt.

Exceptions:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateConsumer - Konsumenten erstellen (mit Nachrichtenselektor und lokalem Nachrichtenflag)

Schnittstelle:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector,
                                Boolean noLocal) ;
```

Erstellen eines Nachrichtenkonsumenten für das angegebene Ziel unter Verwendung eines Nachrichtenselektors und, wenn das Ziel ein Thema ist, mit der Angabe, ob der Nachrichtenkonsument die von seiner eigenen Verbindung veröffentlichten Nachrichten empfängt.

Parameter:

dest (Eingabe)
Das Zielobjekt (Destination).

selector (Eingabe)
Ein Zeichenfolgeobjekt, das einen Nachrichtenselektorausdruck enthält. Es werden nur die Nachrichten mit Eigenschaften, die mit dem Nachrichtenselektorausdruck übereinstimmen, an den Nachrichtenkonsumenten übergeben.

Ein leeres Zeichenfolgeobjekt bedeutet, dass es keinen Nachrichtenselektor für den Nachrichtenkonsumenten gibt.

noLocal (Eingabe)
Der Wert `True` bedeutet, dass der Nachrichtenkonsument die Nachrichten, die von seiner eigenen Verbindung veröffentlicht werden, nicht empfängt. Der Wert `False` bedeutet, dass der Nachrichtenkonsument die Nachrichten, die von seiner eigenen Verbindung veröffentlicht werden, empfängt. Der Standardwert ist `False`.

Rückgabe:
Das MessageConsumer-Objekt.

Exceptions:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateDurableSubscriber - Permanenten Subskribenten erstellen

Schnittstelle:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription) ;
```

Erstellen eines permanenten Subskribenten für das angegebene Thema.

Diese Methode ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

Weitere Informationen zu permanenten Subskribenten finden Sie im Abschnitt „Permanente Subskribenten" auf Seite 38.

Parameter:

dest (Eingabe)
Ein Zielobjekt, das das Thema angibt. Es darf kein temporäres Thema sein.

subscription (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der permanenten Subskription angibt. Der Name muss innerhalb der Client-ID für die Verbindung eindeutig sein.

Rückgabe:
Das MessageConsumer-Objekt, das den permanenten Subskribenten angibt.

Exceptions:

- XMSException
- InvalidDestinationException

CreateDurableSubscriber - Permanenten Subskribenten erstellen (mit Nachrichtenselektor und lokalem Nachrichtenflag)

Schnittstelle:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription,
                                         String selector,
                                         Boolean noLocal) ;
```

Erstellen eines permanenten Subskribenten für das angegebene Thema unter Verwendung eines Nachrichtenselektors und mit der Angabe, ob der permanente Subskribent die von seiner eigenen Verbindung veröffentlichten Nachrichten empfängt.

Diese Methode ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

Weitere Informationen zu permanenten Subskribenten finden Sie im Abschnitt „Permanente Subskribenten" auf Seite 38.

Parameter:

dest (Eingabe)
Ein Zielobjekt, das das Thema angibt. Es darf kein temporäres Thema sein.

subscription (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der permanenten Subskription angibt. Der Name muss innerhalb der Client-ID für die Verbindung eindeutig sein.

selector (Eingabe)
Ein Zeichenfolgeobjekt, das einen Nachrichtenselektorausdruck enthält. Es werden nur die Nachrichten mit Eigenschaften, die mit dem Nachrichtenselektorausdruck übereinstimmen, an den permanenten Subskribenten übergeben.

Ein leeres Zeichenfolgeobjekt bedeutet, dass es keinen Nachrichtenselektor für den permanenten Subskribenten gibt.

noLocal (Eingabe)
Der Wert `True` bedeutet, dass der permanente Subskribent die Nachrichten, die von seiner eigenen Verbindung veröffentlicht werden, nicht empfängt. Der Wert `False` bedeutet, dass der permanente Subskribent die Nachrichten, die von seiner eigenen Verbindung veröffentlicht werden, empfängt. Der Standardwert ist `False`.

Rückgabe:
Das MessageConsumer-Objekt, das den permanenten Subskribenten angibt.

Exceptions:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateMapMessage - Zuordnungsnachricht erstellen

Schnittstelle:

```
IMapMessage CreateMapMessage();
```

Erstellen einer Zuordnungsnachricht.

Parameter:
--

Rückgabe:
Das MapMessage-Objekt.

Exceptions:

- XMSException
- IllegalStateException (Sitzung ist geschlossen)

CreateMessage - Nachricht erstellen

Schnittstelle:

```
IMessage CreateMessage();
```

Erstellen einer Nachricht, die keinen Hauptteil hat.

Parameter:
--

Rückgabe:
Das Message-Objekt.

Exceptions:

- XMSException

- IllegalStateException (Sitzung ist geschlossen)

CreateObjectMessage - Objektnachricht erstellen

Schnittstelle:

```
IObjectMessage CreateObjectMessage();
```

Erstellen einer Objektnachricht.

Parameter:
--

Rückgabe:
Das ObjectMessage-Objekt.

Exceptions:

- XMSException
- IllegalStateException (Sitzung ist geschlossen)

CreateProducer - Produzenten erstellen

Schnittstelle:

```
IMessageProducer CreateProducer(IDestination dest) ;
```

Erstellen eines Nachrichtenproduzenten, um Nachrichten an das angegebene Ziel zu senden.

Parameter:

dest (Eingabe)
Das Zielobjekt (Destination).

Wenn Sie ein leeres Zielobjekt angeben, wird der Nachrichtenproduzent ohne ein Ziel erstellt. In diesem Fall muss die Anwendung jedes Mal, wenn sie den Nachrichtenproduzenten zum Senden einer Nachricht verwendet, ein Ziel angeben.

Rückgabe:
Das MessageProducer-Objekt.

Exceptions:

- XMSException
- InvalidDestinationException

CreateQueue - Warteschlange erstellen

Schnittstelle:

```
IDestination CreateQueue(String queue) ;
```

Erstellen eines Zielobjekt, das eine Warteschlange im Messaging-Server darstellt.

Mit dieser Methode wird die Warteschlange im Messaging-Server nicht erstellt. Sie müssen die Warteschlange erstellen; erst dann kann eine Anwendung diese Methode aufrufen.

Parameter:

queue (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Warteschlange oder einen Uniform Resource Identifier (URI), der die Warteschlange angibt, enthält.

Rückgabe:
Das Zielobjekt, das die Warteschlange angibt.

Exceptions:

- XMSException

CreateStreamMessage - Datenstromnachricht erstellen

Schnittstelle:

```
IStreamMessage CreateStreamMessage();
```

Erstellen einer Datenstromnachricht.

Parameter:
--

Rückgabe:
Das StreamMessage-Objekt.

Exceptions:

- XMSException
- XMS_ILLEGAL_STATE_EXCEPTION

CreateTemporaryQueue - Temporäre Warteschlange erstellen

Schnittstelle:

```
IDestination CreateTemporaryQueue() ;
```

Erstellen einer temporären Warteschlange.

Der Geltungsbereich der temporären Warteschlange ist die Verbindung. Nur die Sitzungen, die von der Verbindung erstellt werden, können die temporäre Warteschlange verwenden.

Die temporäre Warteschlange bleibt so lange bestehen, bis sie explizit gelöscht oder die Verbindung beendet wird, je nachdem, was früher eintritt.

Weitere Informationen zu temporären Warteschlangen finden Sie im Abschnitt „Temporäre Ziele” auf Seite 36.

Parameter:
--

Rückgabe:
Ein Zielobjekt, das die temporäre Warteschlange angibt.

Exceptions:

- XMSException

CreateTemporaryTopic - Temporäres Thema erstellen

Schnittstelle:

```
IDestination CreateTemporaryTopic() ;
```

Erstellen eines temporären Themas.

Der Geltungsbereich des temporären Themas ist die Verbindung. Nur die Sitzungen, die von der Verbindung erstellt werden, können das temporäre Thema verwenden.

Das temporäre Thema bleibt so lange bestehen, bis es explizit gelöscht oder die Verbindung beendet wird, je nachdem, was früher eintritt.

Weitere Informationen zu temporären Themen finden Sie im Abschnitt „Temporäre Ziele” auf Seite 36.

Parameter:
--

Rückgabe:
Ein Zielobjekt, das das temporäre Thema angibt.

Exceptions:

- XMSException

CreateTextMessage - Textnachricht erstellen

Schnittstelle:

```
ITextMessage CreateTextMessage();
```

Erstellen einer Textnachricht mit einem leeren Hauptteil.

Parameter:
--

Rückgabe:
Das TextMessage-Objekt.

Exceptions:

- XMSException

CreateTextMessage - Textnachricht erstellen (initialisiert)

Schnittstelle:

```
ITextMessage CreateTextMessage(String initialValue);
```

Erstellen einer Textnachricht, deren Hauptteil mit dem angegebenen Text initialisiert wird.

Parameter:

initialValue (Eingabe)
Ein Zeichenfolgeobjekt, das den Text enthält, mit dem der Hauptteil der Textnachricht initialisiert werden soll.

--

Rückgabe:
Das TextMessage-Objekt.

Exceptions:

- XMSException

CreateTopic - Thema erstellen

Schnittstelle:

```
IDestination CreateTopic(String topic) ;
```

Erstellen eines Zielobjekts, um ein Thema darzustellen.

Parameter:

topic (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Themas oder einen Uniform Resource Identifier (URI), der das Thema angibt, enthält.

Rückgabe:
Das Zielobjekt, das das Thema angibt.

Exceptions:

- XMSException

Recover - Wiederherstellen

Schnittstelle:

```
void Recover();
```

Wiederherstellen der Sitzung. Die Nachrichtenübermittlung wird gestoppt und anschließend mit der ältesten, nicht bestätigten Nachricht erneut gestartet.

Es darf sich nicht um eine Sitzung mit Transaktionsunterstützung handeln.

Weitere Informationen zum Wiederherstellen einer Sitzung finden Sie im Abschnitt „Nachrichtenbestätigung" auf Seite 29.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException
- IllegalStateException

Rollback - Rückgängig machen

Schnittstelle:

```
void Rollback();
```

Rückgängigmachen aller Nachrichten, die in der aktuellen Transaktion verarbeitet wurden.

Es muss sich um eine Sitzung mit Transaktionsunterstützung handeln.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException
- IllegalStateException

Unsubscribe - Subskription beenden

Schnittstelle:

```
void Unsubscribe(String subscription);
```

Löschen einer permanenten Subskription. Der Messaging-Server löscht den aufgezeichneten Datensatz der permanenten Subskription und sendet danach keine weiteren Nachrichten an den permanenten Subskribenten.

Eine Anwendung kann eine permanente Subskription nicht löschen, wenn eine der folgenden Bedingungen zutrifft:

- Solange es einen aktiven Nachrichtenkonsumenten für die permanente Subskription gibt
- Solange eine konsumierte Nachricht Teil einer anstehenden Transaktion ist
- Solange eine konsumierte Nachricht nicht bestätigt wurde

Diese Methode ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

Parameter:

subscription (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der permanenten Subskription angibt.

Rückgabe:
Void

Exceptions:

- XMSException
- InvalidDestinationException
- IllegalStateException

Geerbte Eigenschaften und Methoden

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IStreamMessage

Eine Datenstromnachricht ist eine Nachricht, deren Hauptteil aus einem Strom von Werten besteht, wobei jedem Wert ein Datentyp zugeordnet ist. Der Inhalt des Hauptteils wird nacheinander geschrieben und gelesen.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IStreamMessage
```

Wenn eine Anwendung den Wert eines Nachrichtendatenstroms abruft, kann der Wert durch XMS in einen anderen Datentyp konvertiert werden. Weitere Informationen zu dieser Form der impliziten Konvertierung finden Sie im Abschnitt „Datenstromnachrichten” auf Seite 85.

Zugehörige Verweise
Datenstromnachrichten
Der Hauptteil einer Datenstromnachricht enthält einen Datenstrom von Werten, wobei jedem Wert ein bestimmter Datentyp zugeordnet ist.

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
ReadBoolean	Lesen eines booleschen Werts aus dem Nachrichtendatenstrom.
ReadByte	Lesen einer 8-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.
ReadBytes	Lesen eines Byte-Arrays aus dem Nachrichtendatenstrom.
ReadChar	Lesen eines 2-Byte-Zeichens aus dem Nachrichtendatenstrom.
ReadDouble	Lesen einer 8-Byte-Gleitkommazahl mit doppelter Genauigkeit aus dem Nachrichtendatenstrom.
ReadFloat	Lesen einer 4-Byte-Gleitkommazahl aus dem Nachrichtendatenstrom.

Methode	Beschreibung
ReadInt	Lesen einer 32-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.
ReadLong	Lesen einer 64-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.
ReadObject	Lesen eines Werts aus dem Nachrichtendatenstrom und Rückgabe seines Datentyps.
ReadShort	Lesen einer 16-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.
ReadString	Lesen einer Zeichenfolge aus dem Nachrichtendatenstrom.
Reset	Zurücksetzen des Hauptteils der Nachricht in den Lesezugriffsmodus und erneutes Positionieren des Cursors auf den Anfang des Nachrichtendatenstroms.
WriteBoolean	Schreiben eines booleschen Werts in den Nachrichtendatenstrom.
WriteByte	Schreiben eines Bytes in den Nachrichtendatenstrom.
WriteBytes	Schreiben eines Byte-Arrays in den Nachrichtendatenstrom.
WriteChar	Schreiben eines Zeichens in den Nachrichtendatenstrom als 2 Bytes, das höchstwertige Byte zuerst.
WriteDouble	Konvertieren einer Gleitkommazahl mit doppelter Genauigkeit in eine lange Ganzzahl und Schreiben der langen Ganzzahl in den Nachrichtendatenstrom als 8 Bytes, das höchstwertige Byte zuerst.
WriteFloat	Konvertieren einer Gleitkommazahl in eine Ganzzahl und Schreiben der Ganzzahl in den Nachrichtendatenstrom als 4 Bytes, das höchstwertige Byte zuerst.
WriteInt	Schreiben einer Ganzzahl in den Nachrichtendatenstrom als 4 Bytes, das höchstwertige Byte zuerst.
WriteLong	Schreiben einer langen Ganzzahl in den Nachrichtendatenstrom als 8 Bytes, das höchstwertige Byte zuerst.
WriteObject	Schreiben eines Werts mit einem angegebenen Datentyp in den Nachrichtendatenstrom.
WriteShort	Schreiben einer kurzen Ganzzahl in den Nachrichtendatenstrom als 2 Bytes, das höchstwertige Byte zuerst.
WriteString	Schreiben einer Zeichenfolge in den Nachrichtendatenstrom.

ReadBoolean - Booleschen Wert lesen

Schnittstelle:

```
Boolean ReadBoolean();
```

Lesen eines booleschen Werts aus dem Nachrichtendatenstrom.

Parameter:

--

Rückgabe:

Der boolesche Wert, der gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException

- MessageEOFException

ReadByte - Byte lesen

Schnittstelle:

```
Int16   ReadSignedByte();
    Byte    ReadByte();
```

Lesen einer 8-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.

Parameter:
--

Rückgabe:
Das Byte, das gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Bytes lesen

Schnittstelle:

```
Int32   ReadBytes(Byte[] array);
```

Lesen eines Byte-Arrays aus dem Nachrichtendatenstrom.

Parameter:

array (Eingabe)
Der Puffer, der das Byte-Array, das gelesen wird, und die Länge des Puffers in Bytes enthält.

Wenn die Anzahl der Bytes im Array kleiner-gleich der Länge des Puffers ist, wird das gesamte Array in den Puffer eingelesen. Ist die Anzahl der Bytes im Array größer als die Länge des Puffers, wird der Puffer mit einem Teil des Arrays gefüllt und ein interner Cursor markiert die Position des nächsten zu lesenden Bytes. Bei einem nachfolgenden Aufruf von readBytes() werden Bytes aus dem Array ab der aktuellen Position des Cursors gelesen.

Wenn Sie bei der Eingabe einen Nullzeiger angeben, überspringt der Aufruf das Byte-Array, ohne es zu lesen.

Rückgabe:
Die Anzahl Bytes, die in den Puffer gelesen werden. Wenn der Puffer teilweise gefüllt ist, ist der Wert kleiner als die Länge des Puffers, was bedeutet, dass es im Array keine weiteren Bytes gibt, die noch gelesen werden müssen. Wenn vor dem Aufruf keine weiteren Bytes mehr aus dem Array gelesen werden müssen, lautet der Wert XMSC_END_OF_BYTEARRAY.

Wenn Sie bei der Eingabe einen Nullzeiger angeben, gibt die Methode keinen Wert zurück.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadChar - Zeichen lesen

Schnittstelle:

```
Char    ReadChar();
```

Lesen eines 2-Byte-Zeichens aus dem Nachrichtendatenstrom.

Parameter:
--

Rückgabe:
Das Zeichen, das gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Gleitkommazahl mit doppelter Genauigkeit lesen

Schnittstelle:

```
Double  ReadDouble();
```

Lesen einer 8-Byte-Gleitkommazahl mit doppelter Genauigkeit aus dem Nachrichtendatenstrom.

Parameter:
--

Rückgabe:
Die Gleitkommazahl mit doppelter Genauigkeit, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Gleitkommazahl lesen

Schnittstelle:

```
Single  ReadFloat();
```

Lesen einer 4-Byte-Gleitkommazahl aus dem Nachrichtendatenstrom.

Parameter:
--

Rückgabe:
Die Gleitkommazahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Ganzzahl lesen

Schnittstelle:

```
Int32   ReadInt();
```

Lesen einer 32-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.

Parameter:
--

Rückgabe:
Die Ganzzahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Lange Ganzzahl lesen

Schnittstelle:

```
Int64   ReadLong();
```

Lesen einer 64-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.

Parameter:
--

Rückgabe:
Die lange Ganzzahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadObject - Objekt lesen

Schnittstelle:

```
Object  ReadObject();
```

Lesen eines Werts aus dem Nachrichtendatenstrom und Rückgabe seines Datentyps.

Parameter:
--

Rückgabe:
Der Wert, bei dem es sich um einen der folgenden Objektypen handelt:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Exceptions:
XMSException

ReadShort - Kurze Ganzzahl lesen

Schnittstelle:

```
Int16   ReadShort();
```

Lesen einer 16-Bit-Ganzzahl mit Vorzeichen aus dem Nachrichtendatenstrom.

Parameter:
--

Rückgabe:
Die kurze Ganzzahl, die gelesen wird.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadString - Zeichenfolge lesen

Schnittstelle:

```
String  ReadString();
```

Lesen einer Zeichenfolge aus dem Nachrichtendatenstrom. Falls erforderlich, konvertiert XMS die Zeichen in der Zeichenfolge in die lokale Codepage.

Parameter:
--

Rückgabe:
Ein Zeichenfolgeobjekt, das die gelesene Zeichenfolge enthält. Wenn eine Datenkonvertierung erforderlich ist, ist dies die Zeichenfolge nach der Konvertierung.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - Zurücksetzen

Schnittstelle:

```
void  Reset();
```

Zurücksetzen des Hauptteils der Nachricht in den Lesezugriffsmodus und erneutes Positionieren des Cursors auf den Anfang des Nachrichtendatenstroms.

Parameter:
--

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotReadableException
- MessageEOFException

WriteBoolean - Booleschen Wert schreiben

Schnittstelle:

```
void  WriteBoolean(Boolean value);
```

Schreiben eines booleschen Werts in den Nachrichtendatenstrom.

Parameter:

value (Eingabe)
Der boolesche Wert, der geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteByte - Byte schreiben

Schnittstelle:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Schreiben eines Bytes in den Nachrichtendatenstrom.

Parameter:

value (Eingabe)
Das Byte, das geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteBytes - Bytes schreiben

Schnittstelle:

```
void  WriteBytes(Byte[] value);
```

Schreiben eines Byte-Arrays in den Nachrichtendatenstrom.

Parameter:

value (Eingabe)
Das Byte-Array, das geschrieben werden soll.

length (Eingabe)
Die Anzahl der Bytes im Array.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteChar - Zeichen schreiben

Schnittstelle:

```
void  WriteChar(Char value);
```

Schreiben eines Zeichens in den Nachrichtendatenstrom als 2 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Das Zeichen, das geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteDouble - Gleitkommazahl mit doppelter Genauigkeit schreiben

Schnittstelle:

```
void  WriteDouble(Double value);
```

Konvertieren einer Gleitkommazahl mit doppelter Genauigkeit in eine lange Ganzzahl und Schreiben der langen Ganzzahl in den Nachrichtendatenstrom als 8 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die Gleitkommazahl mit doppelter Genauigkeit, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteFloat - Gleitkommazahl schreiben

Schnittstelle:

```
void  WriteFloat(Single value);
```

Konvertieren einer Gleitkommazahl in eine Ganzzahl und Schreiben der Ganzzahl in den Nachrichtendatenstrom als 4 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die Gleitkommazahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteInt - Ganzzahl schreiben

Schnittstelle:

```
void  WriteInt(Int32 value);
```

Schreiben einer Ganzzahl in den Nachrichtendatenstrom als 4 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die Ganzzahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException

- MessageNotWritableException

WriteLong - Lange Ganzzahl schreiben

Schnittstelle:

```
void  WriteLong(Int64 value);
```

Schreiben einer langen Ganzzahl in den Nachrichtendatenstrom als 8 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die lange Ganzzahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteObject - Objekt schreiben

Schnittstelle:

```
void  WriteObject(Object value);
```

Schreiben eines Werts mit einem angegebenen Datentyp in den Nachrichtendatenstrom.

Parameter:

objectType (Eingabe)
Der Wert, bei dem es sich um einen der folgenden Objektypen handeln muss:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value (Eingabe)
Ein Byte-Array, das den zu schreibenden Wert enthält.

length (Eingabe)
Die Anzahl der Bytes im Array.

Rückgabe:
Void

Exceptions:

- XMSException

WriteShort - Kurze Ganzzahl schreiben

Schnittstelle:

```
void  WriteShort(Int16 value);
```

Schreiben einer kurzen Ganzzahl in den Nachrichtendatenstrom als 2 Bytes, das höchstwertige Byte zuerst.

Parameter:

value (Eingabe)
Die kurze Ganzzahl, die geschrieben werden soll.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

WriteString - Zeichenfolge schreiben

Schnittstelle:

```
void  WriteString(String value);
```

Schreiben einer Zeichenfolge in den Nachrichtendatenstrom.

Parameter:

value (Eingabe)
Ein Zeichenfolgeobjekt, das die zu schreibende Zeichenfolge enthält.

Rückgabe:
Void

Exceptions:

- XMSException
- MessageNotWritableException

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden von der Schnittstelle IMessage übernommen:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Die folgenden Methoden werden aus der Schnittstelle IMessage übernommen:

clearBody, clearProperties, PropertyExists

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ITextMessage

Eine Textnachricht ist eine Nachricht, deren Hauptteil aus einer Zeichenfolge besteht.

Vererbungshierarchie:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.ITextMessage
```

Zugehörige Verweise

Textnachrichten
Der Hauptteil einer Textnachricht enthält eine Zeichenfolge.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
Text	Abrufen und Festlegen der Zeichenfolge, die den Hauptteil der Textnachricht bildet.

Text - Text abrufen und festlegen

Schnittstelle:

```
String Text
{
  get;
  set;
}
```

Abrufen und Festlegen der Zeichenfolge, die den Hauptteil der Textnachricht bildet.

Falls erforderlich, konvertiert XMS die Zeichen in der Zeichenfolge in die lokale Codepage.

Exceptions:

- XMSException
- MessageNotReadableException
- MessageNotWritableException
- MessageEOFException

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden von der Schnittstelle IMessage übernommen:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Die folgenden Methoden werden aus der Schnittstelle IMessage übernommen:

clearBody, clearProperties, PropertyExists

Folgende Methoden werden aus der Schnittstelle IPropertyContext übernommen:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

TransactionInProgressException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionInProgressException
```

XMS löst diese Ausnahme aus, wenn eine Anwendung eine Operation anfordert, die nicht gültig ist, weil eine Transaktion in Bearbeitung ist.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

TransactionRolledBackException

Vererbungshierarchie:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionRolledBackException
```

XMS löst diese Ausnahme aus, wenn eine Anwendung die Methode Session.commit() aufruft, um die aktuelle Transaktion festzuschreiben, die Transaktion dann aber rückgängig gemacht wird.

Geerbte Eigenschaften und Methoden

Die folgenden Methoden werden aus der Schnittstelle XMSException übernommen:

GetErrorCode, GetLinkedException

XMSException

Wenn XMS beim Verarbeiten eines Aufrufs an eine .NET-Methode einen Fehler erkennt, löst XMS eine Ausnahmebedingung aus. Eine Ausnahme ist ein Objekt, das Informationen über den Fehler enthält.

Vererbungshierarchie:

```
System.Exception
   |
   +----IBM.XMS.XMSException
```

Es gibt verschiedene Typen von XMS-Ausnahmebedingungen, und bei einem XMSException-Objekt handelt es sich um einen dieser Typen. Allerdings ist die Klasse XMSException eine Superklasse für die anderen XMS-Ausnahmeklassen. XMS löst ein XMSException-Objekt in Situationen aus, in denen keiner der anderen Ausnahmetypen geeignet ist.

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
ErrorCode	Abrufen des Fehlercodes.
LinkedException	Abrufen der nächsten Ausnahme in der Ausnahmenkette.

ErrorCode - Fehlercode abrufen

Schnittstelle:

```
public String ErrorCode
{
  get {return errorCode_;}
}
```

Abrufen des Fehlercodes.

Exceptions:

- XMSException

LinkedException - Verlinkte Ausnahmebedingung abrufen

Schnittstelle:

```
public Exception LinkedException
{
  get { return linkedException_;}
  set { linkedException_ = value;}
}
```

Abrufen der nächsten Ausnahme in der Ausnahmenkette.

Die Methode gibt eine Null zurück, wenn die Kette keine weiteren Ausnahmen enthält.

Exceptions:

- XMSException

XMSFactoryFactory

Wenn eine Anwendung keine verwalteten Objekte verwendet, können Sie mithilfe dieser Klasse Verbindungsfactorys, Warteschlangen und Themen erstellen.

Vererbungshierarchie:

--

.NET-Eigenschaften

Zusammenfassung der Eigenschaften von .NET:

.NET-Eigenschaft	Beschreibung
MetaData	Abrufen der Metadaten, die für den Verbindungstyp des XMSFactoryFactory-Objekts geeignet sind.

Metadaten - Metadaten abrufen

Schnittstelle:

```
IConnectionMetaData MetaData
```

Abrufen der Metadaten, die für den Verbindungstyp des XMSFactoryFactory-Objekts geeignet sind.

Exceptions:

--

Methoden

Zusammenfassung der Methoden:

Methode	Beschreibung
CreateConnectionFactory	Erstellen eines ConnectionFactory-Objekt des deklarierten Typs.
CreateQueue	Erstellen eines Zielobjekt, das eine Warteschlange im Messaging-Server darstellt.
CreateTopic	Erstellen eines Zielobjekts, um ein Thema darzustellen.
GetInstance	Erstellen Sie eine Instanz von XMSFactoryFactory. Eine XMS-Anwendung verwendet ein XMSFactoryFactory-Objekt, um einen Verweis auf ein ConnectionFactory-Objekt abzurufen, das für den erforderlichen Protokolltyp geeignet ist. Dieses ConnectionFactory-Objekt kann dann nur für diesen Protokolltyp Verbindungen herstellen.

CreateConnectionFactory - Verbindungsfactory erstellen

Schnittstelle:

```
IConnectionFactory CreateConnectionFactory();
```

Erstellen eines ConnectionFactory-Objekt des deklarierten Typs.

Parameter:
--

Rückgabe:
Das ConnectionFactory-Objekt.

Exceptions:

- XMSException

CreateQueue - Warteschlange erstellen

Schnittstelle:

```
IDestination CreateQueue(String name);
```

Erstellen eines Zielobjekt, das eine Warteschlange im Messaging-Server darstellt.

Mit dieser Methode wird die Warteschlange im Messaging-Server nicht erstellt. Sie müssen die Warteschlange erstellen; erst dann kann eine Anwendung diese Methode aufrufen.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen der Warteschlange oder einen Uniform Resource Identifier (URI), der die Warteschlange angibt, enthält.

Rückgabe:
Das Zielobjekt, das die Warteschlange angibt.

Exceptions:

- XMSException

CreateTopic - Thema erstellen

Schnittstelle:

```
IDestination CreateTopic(String name);
```

Erstellen eines Zielobjekts, um ein Thema darzustellen.

Parameter:

name (Eingabe)
Ein Zeichenfolgeobjekt, das den Namen des Themas oder einen Uniform Resource Identifier (URI), der das Thema angibt, enthält.

Rückgabe:
Das Zielobjekt, das das Thema angibt.

Exceptions:

- XMSException

GetInstance - Instanz von XMSFactoryFactory abrufen

Schnittstelle:

```
static XMSFactoryFactory GetInstance(int connectionType);
```

Erstellen Sie eine Instanz von XMSFactoryFactory. Eine XMS-Anwendung verwendet ein XMSFactoryFactory-Objekt, um einen Verweis auf ein ConnectionFactory-Objekt abzurufen, das für den erforderlichen Protokolltyp geeignet ist. Dieses ConnectionFactory-Objekt kann dann nur für diesen Protokolltyp Verbindungen herstellen.

Parameter:

connectionType (Eingabe)
Der Verbindungstyp, für den das ConnectionFactory-Objekt Verbindungen herstellt:

- XMSC.CT_WPM
- XMSC.CT_RTT
- XMSC.CT_WMQ

Rückgabe:
Das XMSFactoryFactory-Objekt, das dem deklarierten Verbindungstyp zugeordnet ist.

Exceptions:

- NotSupportedException

Eigenschaften von XMS-Objekten

In diesem AbschnittKapitel werden die mit XMS definierten Objekteigenschaften dokumentiert.

Der AbschnittKapitel enthält folgende ThemenAbschnitte:

- „Eigenschaften von Connection” auf Seite 190
- „Eigenschaften von ConnectionFactory” auf Seite 190
- „Eigenschaften von ConnectionMetaData” auf Seite 197
- „Eigenschaften von Destination” auf Seite 198
- „Eigenschaften von InitialContext” auf Seite 200
- „Eigenschaften von Message” auf Seite 201
- „Eigenschaften von MessageConsumer” auf Seite 207
- „Eigenschaften von MessageProducer” auf Seite 207
- „Eigenschaften von Session” auf Seite 207

In jedem Thema werden die Eigenschaften eines Objekts mit dem angegebenen Typ aufgeführt und es wird eine kurze Beschreibung jeder Eigenschaft bereitgestellt.

Dieser AbschnittKapitel enthält auch den Thema „Eigenschaftsdefinitionen” auf Seite 207, in dem eine Definition jeder Eigenschaft bereitgestellt wird.

Wenn eine Anwendung ihre eigenen Eigenschaften der in diesem AbschnittKapitel beschriebenen Objekte definiert, führt dies nicht zu einem Felder, aber es treten möglicherweise unvorhersehbare Ergebnisse auf.

Anmerkung: Die Eigenschaftsnamen und -werte in diesem Abschnitt werden im Formular XMSC.*NAME* angezeigt. Dies ist das für C und C++ verwendete Formular. In .NET kann es sich jedoch um XMSC.*NAME* oder XMSC_*NAME* handeln, je nachdem, wie Sie es verwenden:

- Wenn Sie eine Eigenschaft angeben, muss der Eigenschaftsname das Format XMSC.*NAME* haben, wie im folgende Beispiel gezeigt wird:

```
cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
```

- Wenn Sie eine Zeichenfolge angeben, muss der Eigenschaftsname das Format XMSC_*NAME* haben, wie im folgenden Beispiel gezeigt wird:

```
cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
```

In .NET werden Eigenschaftsnamen und -werte als Konstanten in der XMSC-Klasse bereitgestellt. Diese Konstanten geben Zeichenfolgen an und werden von jeder XMS .NET -Anwendung verwendet. Wenn

Sie diese vordefinierten Konstanten verwenden, haben die Eigenschaftsnamen und -werte das Format XMSC.*NAME*, d. h. Sie würden beispielsweise XMSC.USERID statt XMSC_USERID verwenden.

Die Datentypen haben auch das Format, das für C/C++ verwendet wird. Die entsprechenden Werte für .NET finden Sie in„Datentypen für .NET” auf Seite 49.

Zugehörige Konzepte

Eigene Anwendung erstellen
Sie können Ihre eigenen Anwendungen auf die gleiche Weise erstellen, wie Sie die Beispielanwendungen erstellen.

Zugehörige Verweise

.NET-Schnittstellen
In diesem Thema werden die Schnittstellen der .NET-Klasse und die zugehörigen Eigenschaften und Methoden dokumentiert.

Eigenschaften von Connection

Eine Übersicht über die Eigenschaften des Objekts Connection mit Links zu detaillierteren Referenzinformationen.

Tabelle 28. Eigenschaften von Connection

Name der Eigenschaft	Beschreibung
„XMSC_WMQ_RESOLVED_QUEUE_MANAGER” auf Seite 245	Diese Eigenschaft wird verwendet, um den Namen des Warteschlangenmanagers abzurufen, mit dem sie verbunden ist.
„XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID” auf Seite 245	Diese Eigenschaft wird nach der Verbindung mit der ID des Warteschlangenmanagers gefüllt.
XMSC_WPM_CONNECTION_PROTOCOL	Das Kommunikationsprotokoll, das für die Verbindung zur Messaging-Engine verwendet wird. Diese Eigenschaft ist schreibgeschützt.
XMSC_WPM_HOST_NAME	Der Hostname oder die IP-Adresse des Systems, das die Messaging-Engine enthält, mit der die Anwendung verbunden ist. Diese Eigenschaft ist schreibgeschützt.
XMSC_WPM_ME_NAME	Der Name der Messaging-Engine, mit der die Anwendung verbunden ist. Diese Eigenschaft ist schreibgeschützt.
XMSC_WPM_PORT	Die Nummer des Ports, an dem die Messaging-Engine, mit der die Anwendung verbunden ist, empfangsbereit ist. Diese Eigenschaft ist schreibgeschützt.

Ein Connection-Objekt verfügt auch über schreibgeschützte Eigenschaften, die aus den Eigenschaften der Verbindungsfactory abgeleitet werden, die zum Erstellen der Verbindung verwendet wurde. Diese Eigenschaften werden nicht nur von den Eigenschaften der Verbindungsfactory, die zum Zeitpunkt der Erstellung der Verbindung festgelegt wurden, sondern auch von den Standardwerten der Eigenschaften, die nicht festgelegt wurden, abgeleitet. Zu den Eigenschaften gehören nur die Eigenschaften, die für den Typ des Messaging-Servers relevant sind, mit dem die Anwendung verbunden ist. Die Namen der Eigenschaften sind mit den Namen der Eigenschaften der Verbindungsfactory identisch.

Eigenschaften von ConnectionFactory

Eine Übersicht über die Eigenschaften des Objekts ConnectionFactory mit Links zu detaillierteren Referenzinformationen.

Tabelle 29. Eigenschaften von ConnectionFactory

Name der Eigenschaft	Beschreibung
„XMSC_ASYNC_EXCEPTIONS” auf Seite 220	Diese Eigenschaft bestimmt, ob XMS einen ExceptionListener nur dann informiert, wenn eine Verbindung unterbrochen wird oder wenn eine Ausnahmebedingung asynchron zu einem XMS-API-Aufruf auftritt. Diese Eigenschaft gilt für alle Connections (Verbindungen), die ausdieser ConnectionFactory erstellt werden und für die ein ExceptionListenerregistriert ist.
XMSC_CLIENT_ID	Die Client-ID für eine Verbindung.
XMSC_CONNECTION_TYPE	Der Typ des Messaging-Servers, zu dem eine Anwendung eine Verbindung herstellt.
XMSC_PASSWORD	Ein Kennwort, das zur Authentifizierung der Anwendung verwendet werden kann, wenn versucht wird, eine Verbindung zu einem Messaging-Server herzustellen.
„XMSC_RTT_BROKER_PING_INTERVAL” auf Seite 226	Das Zeitintervall in Millisekunden, nach dem XMS .NET die Verbindung zum Echtzeit-Messaging-Server überprüft, um eine Aktivität zu erkennen.
XMSC_RTT_CONNECTION_PROTOCOL	Das Kommunikationsprotokoll, das für eine Echtzeitverbindung zu einem Broker verwendet wird.
XMSC_RTT_HOST_NAME	Der Hostname oder die IP-Adresse des Systems, auf dem ein Broker ausgeführt wird.
XMSC_RTT_LOCAL_ADDRESS	Der Hostname oder die IP-Adresse der lokalen Netzschnittstelle, die für eine Echtzeitverbindung zu einem Broker verwendet werden soll.
XMSC_RTT_MULTICAST	Die Multicasteinstellung für eine Verbindungsfactory oder ein Ziel.
XMSC_RTT_PORT	Die Nummer des Ports, an dem ein Broker für eingehende Anforderungen empfangsbereit ist.
XMSC_USERID	Eine Benutzer-ID, die zur Authentifizierung der Anwendung verwendet werden kann, wenn versucht wird, eine Verbindung zu einem Messaging-Server herzustellen.
XMSC_WMQ_BROKER_CONTROLQ	Der Name der Steuerwarteschlange, die von einem Broker verwendet wird. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.

Tabelle 29. Eigenschaften von ConnectionFactory (Forts.)	
Name der Eigenschaft	**Beschreibung**
XMSC_WMQ_BROKER_PUBQ	Der Name der Warteschlange, die von einem Broker überwacht wird, wo Anwendungen Nachrichten senden, die sie veröffentlichen. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
XMSC_WMQ_BROKER_QMGR	Der Name des Warteschlangenmanagers, mit dem ein Broker verbunden ist. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
XMSC_WMQ_BROKER_SUBQ	Der Name der Subskribentenwarteschlange für einen nicht permanenten Nachrichtenkonsumenten. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
XMSC_WMQ_BROKER_VERSION	Der Typ des Brokers, der von der Anwendung für eine Verbindung oder für das Ziel verwendet wird. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
„XMSC_WMQ_CCDTURL” auf Seite 231	Eine URL (Uniform Resource Locator), die den Namen und die Position der Datei angibt, die die Kanaldefinitionstabelle des Clients enthält, und auch angibt, wie auf die Datei zugegriffen werden kann.
XMSC_WMQ_CHANNEL	Der Name des Kanals, der für eine Verbindung verwendet werden soll.
„XMSC_WMQ_CLIENT_RECONNECT_OPTIONS” auf Seite 231	Diese Eigenschaft gibt die Clientverbindungswiederholungsoptionen für neue Verbindungen an, die von dieser Factory erstellt werden.

Tabelle 29. Eigenschaften von ConnectionFactory (Forts.)	
Name der Eigenschaft	**Beschreibung**
„XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT” auf Seite 232	Diese Eigenschaft gibt den Zeitraum (in Sekunden) an, in dem eine Clientverbindung versucht, die Verbindung wiederherzustellen.
XMSC_WMQ_CONNECTION_MODE	Der Modus, in dem eine Anwendung eine Verbindung zu einem Warteschlangenmanager herstellt.
„XMSC_WMQ_CONNECTION_NAME_LIST” auf Seite 233	Diese Eigenschaft gibt die Hosts an, zu denen der Client die Verbindung wiederherzustellen versucht, nachdem die Verbindung unterbrochen wurde.
XMSC_WMQ_FAIL_IF_QUIESCE	Gibt an, ob Aufrufe bestimmter Methoden fehlschlagen, wenn sich der Warteschlangenmanager, mit dem die Anwendung verbunden ist, im Quiesce-Zustand befindet.
XMSC_WMQ_HOST_NAME	Der Hostname oder die IP-Adresse des Systems, auf dem ein Warteschlangenmanager ausgeführt wird.
XMSC_WMQ_LOCAL_ADDRESS	Für eine Verbindung zu einem Warteschlangenmanager gibt diese Eigenschaft die zu verwendende lokale Netzschnittstelle und/oder den zu verwendenden lokalen Port bzw. den Bereich der zu verwendenden lokalen Ports an.
XMSC_WMQ_MESSAGE_SELECTION	Es wird bestimmt, ob die Nachrichtenauswahl vom XMS-Client oder vom Broker vorgenommen wird. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
XMSC_WMQ_MSG_BATCH_SIZE	Die maximale Anzahl Nachrichten, die bei Verwendung einer asynchronen Nachrichtenübermittlung in einem einzigen Stapel aus einer Warteschlange abgerufen werden sollen. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.

Tabelle 29. Eigenschaften von ConnectionFactory (Forts.)	
Name der Eigenschaft	**Beschreibung**
XMSC_WMQ_POLLING_INTERVAL	Wenn sich bei den einzelnen Nachrichtenlistenern innerhalb einer Sitzung keine geeignete Nachricht in der zugehörigen Warteschlange befindet, ist dieser Wert das maximale Intervall in Millisekunden, das verstreicht, bevor die einzelnen Nachrichtenlistener erneut versuchen, eine Nachricht aus der zugehörigen Warteschlange abzurufen. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
„XMSC_WMQ_PROVIDER_VERSION" auf Seite 242	Die Version, das Release, die Modifikationsstufe und das Fixpack des Warteschlangenmanagers, zu dem die Anwendung eine Verbindung herstellen soll.
XMSC_WMQ_PORT	Die Nummer des Ports, an dem ein Warteschlangenmanager für eingehende Anforderungen empfangsbereit ist.
XMSC_WMQ_PUB_ACK_INTERVAL	Die Anzahl der Nachrichten, die von einem Publisher veröffentlicht werden, bevor der XMS-Client eine Bestätigung vom Broker anfordert. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
„XMSC_WMQ_PUT_ASYNC_ALLOWED" auf Seite 237	Diese Eigenschaft gibt an, ob Nachrichtenproduzenten asynchrone PUT-Operationen verwenden dürfen, um Nachrichten an diese Zieladresse zu senden.
XMSC_WMQ_QMGR_CCSID	Die ID des codierten Zeichensatzes (CCSID) oder die Codepage, in der Zeichendatenfelder, die in der MQI (Message Queue Interface) definiert sind, zwischen dem XMS -Client und dem IBM WebSphere MQ -Client ausgetauscht werden.
XMSC_WMQ_QUEUE_MANAGER	Der Name des Warteschlangenmanagers, zu dem eine Verbindung hergestellt werden soll.
XMSC_WMQ_RECEIVE_EXIT	Gibt einen Kanalempfangsexit an, der ausgeführt werden soll.
XMSC_WMQ_RECEIVE_EXIT_INIT	Die Benutzerdaten, die beim Aufruf eines Kanalempfangsexits an diesen übergeben werden.
XMSC_WMQ_SECURITY_EXIT	Gibt einen Kanalsicherheitsexit an.
XMSC_WMQ_SECURITY_EXIT_INIT	Die Benutzerdaten, die beim Aufruf des Kanalsicherheitsexits an diesen übergeben werden.

Tabelle 29. Eigenschaften von ConnectionFactory (Forts.)	
Name der Eigenschaft	**Beschreibung**
„XMSC_WMQ_SEND_CHECK_COUNT” auf Seite 247	Die Anzahl Sendeaufrufe, die innerhalb einer einzelnen XMS-Sitzung ohne Transaktionsunterstützung zwischen Überprüfungen auf Fehler bei asynchronen Put-Operationen zugelassen werden sollen.
XMSC_WMQ_SEND_EXIT	Gibt einen Kanalsendeexit an.
XMSC_WMQ_SEND_EXIT_INIT	Die Benutzerdaten, die beim Aufruf von Kanalsendeexits an diese übergeben werden.
„XMSC_WMQ_SHARE_CONV_ALLOWED” auf Seite 247	Gibt an, ob eine Clientverbindung ihr Socket mit anderen XMS -Verbindungen der höchsten Ebene von demselben Prozess zu demselben Warteschlangenmanager gemeinsam nutzen kann, wenn die Kanaldefinitionen übereinstimmen. Diese Eigenschaft wird bereitgestellt, um eine vollständige Isolierung von Verbindungen in separaten Sockets zu ermöglichen, wenn dies für die Anwendungsentwicklung oder Wartung oder aus betriebsbedingten Gründen erforderlich ist.
XMSC_WMQ_SSL_CERT_STORES	Die Positionen der Server, auf denen sich die Zertifikatswiderrufslisten (CRLs) befinden, die für eine SSL-Verbindung zu einem Warteschlangenmanager verwendet werden sollen.
XMSC_WMQ_SSL_CIPHER_SPEC	Der Name der CipherSpec, die für eine sichere Verbindung zu einem Warteschlangenmanager verwendet werden soll.
XMSC_WMQ_SSL_CIPHER_SUITE	Der Name der CipherSuite , die für eine SSL-oder TLS-Verbindung zu einem Warteschlangenmanager verwendet werden soll. Welches Protokoll bei der Vereinbarung der sicheren Verbindung verwendet wird, ist von der angegebenen Cipher-Suite abhängig.
XMSC_WMQ_SSL_CRYPTO_HW	Konfigurationsdetails für die Verschlüsselungshardware, die mit dem Clientsystem verbunden ist.
XMSC_WMQ_SSL_FIPS_REQUIRED	Der Wert dieser Eigenschaft legt fest, ob eine Anwendung nicht FIPS-konforme Cipher-Suites verwenden kann oder nicht. Wenn diese Eigenschaft auf 'true' gesetzt ist, werden nur FIPS-Algorithmen für die Client-Server-Verbindung verwendet.
XMSC_WMQ_SSL_KEY_REPOSITORY	Gibt die Position der Schlüsseldatenbankdatei an, in der Schlüssel und Zertifikate gespeichert werden.
XMSC_WMQ_SSL_KEY_RESETCOUNT	Der Wert von KeyResetCount gibt die Gesamtzahl unverschlüsselter Bytes an, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL-Dialog gesendet oder empfangen werden.
XMSC_WMQ_SSL_PEER_NAME	Der Peername, der für eine SSL-Verbindung zu einem Warteschlangenmanager verwendet werden soll.
XMSC_WMQ_SYNCPOINT_ALL_GETS	Gibt an, ob alle Nachrichten innerhalb der Synchronisationspunktsteuerung aus Warteschlangen abgerufen werden müssen.
„XMSC_WMQ_TARGET_CLIENT” auf Seite 255	

Tabelle 29. Eigenschaften von ConnectionFactory (Forts.)	
Name der Eigenschaft	**Beschreibung**
XMSC_WMQ_TEMP_Q_PREFIX	Das Präfix, das verwendet wird, um den Namen der dynamischen IBM WebSphere MQ -Warteschlange zu bilden, die erstellt wird, wenn die Anwendung eine temporäre Eine XMS -Warteschlange erstellt.
XMSC_WMQ_TEMP_TOPIC_PREFIX	Beim Erstellen temporärer Themen generiert XMS eine Themenzeichenfolge im Format "`TEMP/TEMPTOPICPREFIX/unique_id`" oder, wenn diese Eigenschaft den Standardwert hat, nur "`TEMP/unique_id`". Die Angabe eines nichtleeren Werts ermöglicht die Definition bestimmter Modellwarteschlangen für die Erstellung der verwalteten Warteschlangen für Abonnenten temporärer Themen, die unter dieser Verbindung erstellt werden.
XMSC_WMQ_TEMPORARY_MODEL	Der Name der IBM WebSphere MQ -Modellwarteschlange, aus der eine dynamische Warteschlange erstellt wird, wenn die Anwendung eine temporäre Eine XMS -Warteschlange erstellt.
XMSC_WPM_BUS_NAME	Für eine Verbindungsfactory der Name des Service Integration Bus, zu dem die Anwendung eine Verbindung herstellt, bzw. für ein Ziel der Name des Service Integration Bus, in dem sich das Ziel befindet.
XMSC_WPM_CONNECTION_PROXIMITY	Die Verbindungsabstandseinstellung für die Verbindung.
XMSC_WPM_DUR_SUB_HOME	Der Name der Messaging-Engine, auf der alle permanenten Subskriptionen für eine Verbindung oder ein Ziel verwaltet werden.
XMSC_WPM_LOCAL_ADDRESS	Für eine Verbindung zu einem Service Integration Bus gibt diese Eigenschaft die zu verwendende lokale Netzschnittstelle und/oder den zu verwendenden lokalen Port bzw. den Bereich der zu verwendenden lokalen Ports an.
XMSC_WPM_NON_PERSISTENT_MAP	Die Zuverlässigkeitsstufe von nicht persistenten Nachrichten, die über die Verbindung gesendet werden.
XMSC_WPM_PERSISTENT_MAP	Die Zuverlässigkeitsstufe von persistenten Nachrichten, die über die Verbindung gesendet werden.
XMSC_WPM_PROVIDER_ENDPOINTS	Eine Folge aus einer oder mehreren Endpunktadressen von Bootstrap-Servern.
XMSC_WPM_TARGET_GROUP	Der Name einer Zielgruppe aus Messaging-Engines.
XMSC_WPM_TARGET_SIGNIFICANCE	Die Signifikanz der Zielgruppe von Messaging-Engines.
XMSC_WPM_TARGET_TRANSPORT_CHAIN	Der Name der eingehenden Transportkette, die die Anwendung für die Verbindung zu einer Messaging-Engine verwenden muss.
XMSC_WPM_TARGET_TYPE	Der Typ der Zielgruppe von Messaging-Engines.
XMSC_WPM_TEMP_Q_PREFIX	Das Präfix, mit dem der Name der temporären Warteschlange gebildet wird, die im Service Integration Bus erstellt wird, wenn die Anwendung die temporäre Eine XMS-Warteschlange erstellt.

Tabelle 29. Eigenschaften von ConnectionFactory (Forts.)	
Name der Eigenschaft	**Beschreibung**
XMSC_WPM_TEMP_TOPIC_PREFIX	Das Präfix, das verwendet wird, um den Namen eines temporären Themas zu bilden, das von der Anwendung erstellt wird.

Zugehörige Konzepte

ConnectionFactory- und Connection-Objekte
Ein Verbindungsfactoryobjekt (ConnectionFactory) stellt eine Vorlage bereit, die von einer Anwendung zum Erstellen eines Verbindungsobjekts (Connection) verwendet wird. Das Connection-Objekt verwendet die Anwendung wiederum zum Erstellen eines Sitzungsobjekts (Session).

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager
Damit eine XMS .NET-Anwendung sichere Verbindungen zu einem IBM WebSphere MQ-Queue Manager herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

Sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine
Damit eine XMS .NET-Anwendung sichere Verbindungen zu einer WebSphere Service Integration Bus-Messaging-Engine herstellen kann, müssen die relevanten Eigenschaften im ConnectionFactory-Objekt definiert werden.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete ConnectionFactory-Objekte
Wenn eine Anwendung eine Verbindungsfactory erstellt, müssen einige Eigenschaften zum Erstellen einer Verbindung zu einem Messaging-Server definiert werden.

Eigenschaften von ConnectionMetaData

Eine Übersicht über die Eigenschaften des Objekts ConnectionMetaData mit Links zu detaillierteren Referenzinformationen.

Tabelle 30. Eigenschaften von ConnectionMetaData	
Name der Eigenschaft	**Beschreibung**
XMSC_JMS_MAJOR_VERSION	Die Hauptversionsnummer der JMS -Spezifikation, auf der XMS basiert. Diese Eigenschaft ist schreibgeschützt.
XMSC_JMS_MINOR_VERSION	Die untergeordnete Versionsnummer der JMS -Spezifikation, auf der XMS basiert. Diese Eigenschaft ist schreibgeschützt.
XMSC_JMS_VERSION	Die Versions-ID der JMS -Spezifikation, auf der XMS basiert. Diese Eigenschaft ist schreibgeschützt.

Tabelle 30. Eigenschaften von ConnectionMetaData (Forts.)	
Name der Eigenschaft	**Beschreibung**
XMSC_MAJOR_VERSION	Die Versionsnummer des XMS-Clients. Diese Eigenschaft ist schreibgeschützt.
XMSC_MINOR_VERSION	Die Releasenummer des XMS-Client. Diese Eigenschaft ist schreibgeschützt.
XMSC_PROVIDER_NAME	Der Provider des XMS-Clients. Diese Eigenschaft ist schreibgeschützt.
XMSC_VERSION	Die Versions-ID des XMS-Clients. Diese Eigenschaft ist schreibgeschützt.

Eigenschaften von Destination

Eine Übersicht über die Eigenschaften des Objekts Destination mit Links zu detaillierteren Referenzinformationen.

Tabelle 31. Eigenschaften von Destination	
Name der Eigenschaft	**Beschreibung**
XMSC_DELIVERY_MODE	Der Zustellmodus von Nachrichten, die an das Ziel gesendet werden.
XMSC_PRIORITY	Die Priorität von Nachrichten, die an das Ziel gesendet werden.
XMSC_RTT_MULTICAST	Die Multicasteinstellung für eine Verbindungsfactory oder ein Ziel.
XMSC_TIME_TO_LIVE	Die Lebensdauer für Nachrichten, die an das Ziel gesendet werden.
XMSC_WMQ_BROKER_VERSION	Der Typ des Brokers, der von der Anwendung für eine Verbindung oder für das Ziel verwendet wird.
XMSC_WMQ_CCSID	Die ID (CCSID) des codierten Zeichensatzes oder der Codepage, in dem bzw. in der sich die Zeichenfolgen mit Zeichendaten im Hauptteil einer Nachricht befinden, wenn der XMS-Client die Nachricht an das Ziel weiterleitet.
XMSC_WMQ_DUR_SUBQ	Der Name der Subskribentenwarteschlange für einen permanenten Subskribenten, der Nachrichten von dem Ziel empfängt. **Anmerkung:** Diese Eigenschaft kann mit Version 2.0 von IBM Message Service Client für .NET verwendet werden, hat aber keine Auswirkung auf eine mit einem Warteschlangenmanager von IBM WebSphere MQ Version 7.0 verbundene Anwendung, es sei denn, die Eigenschaft XMSC_WMQ_PROVIDER_VERSION der Verbindungsfactory ist auf eine kleinere Versionsnummer als 7 gesetzt.
XMSC_WMQ_ENCODING	Es wird angegeben, wie numerische Daten im Nachrichtentext dargestellt werden, wenn der XMS-Client die Nachricht an das Ziel weiterleitet.
XMSC_WMQ_FAIL_IF_QUIESCE	Gibt an, ob Aufrufe bestimmter Methoden fehlschlagen, wenn sich der Warteschlangenmanager, mit dem die Anwendung verbunden ist, im Quiesce-Zustand befindet.

Tabelle 31. Eigenschaften von Destination (Forts.)	
Name der Eigenschaft	**Beschreibung**
„XMSC_WMQ_MESSAGE_BODY” auf Seite 235	Diese Eigenschaft legt fest, ob eine XMS -Anwendung den MQRFH2 einer IBM WebSphere MQ -Nachricht als Teil der Nachrichtennutzdaten (also als Teil des Nachrichtenhauptteils) verarbeitet.
„XMSC_WMQ_MQMD_MESSAGE_CONTEXT” auf Seite 236	Legt fest, welche Stufe des Nachrichtenkontexts von der Anwendung XMS festgelegt werden soll. Die Anwendung muss mit entsprechender Kontextberechtigung ausgeführt werden, damit diese Eigenschaft in Kraft treten kann.
„XMSC_WMQ_MQMD_READ_ENABLED” auf Seite 236	Diese Eigenschaft bestimmt, ob eine XMS-Anwendung die Werte von MQMD-Feldern extrahieren kann oder nicht.
„XMSC_WMQ_MQMD_WRITE_ENABLED” auf Seite 237	Diese Eigenschaft bestimmt, ob eine XMS-Anwendung die Werte von MQMD-Feldern verarbeiten kann oder nicht.
„XMSC_WMQ_READ_AHEAD_ALLOWED” auf Seite 238	Diese Eigenschaft bestimmt, ob Nachrichtenkonsumenten und Warteschlangenbrowser die Vorauslesefunktion verwenden dürfen, um nicht permanente, nicht transaktionsorientierte Nachrichten von diesem Ziel in einen internen Puffer abzurufen, bevor sie die Nachrichten empfangen.
„XMSC_WMQ_READ_AHEAD_CLOSE_POLICY” auf Seite 238	Diese Eigenschaft gibt für Nachrichten, die an einen asynchronen Nachrichtenlistener zugestellt werden, an, was mit Nachrichten im internen Vorauslesepuffer geschehen soll, wenn der Nachrichtenkonsument geschlossen wird.
„XMSC_WMQ_RECEIVE_CCSID” auf Seite 244	Die Zieleigenschaft, die die Ziel-CCSID für die Nachrichtenkonvertierung des Warteschlangenmanagers festlegt. Der Wert wird ignoriert, außer wenn XMSC_WMQ_RECEIVE_CONVERSION auf WMQ_RECEIVE_CONVERSION_QMGR gesetzt ist.
„XMSC_WMQ_RECEIVE_CONVERSION” auf Seite 244	Eine Zieleigenschaft, die bestimmt, ob eine Datenkonvertierung vom Warteschlangenmanager durchgeführt wird.
XMSC_WMQ_TARGET_CLIENT	Gibt an, ob Nachrichten, die an das Ziel gesendet werden, einen MQRFH2-Header enthalten.
XMSC_WMQ_TEMP_TOPIC_PREFIX	Beim Erstellen temporärer Themen generiert XMS eine Themenzeichenfolge im Format "`TEMP/TEMPTOPICPREFIX/unique_id`" oder, wenn diese Eigenschaft den Standardwert hat, nur "`TEMP/unique_id`". Die Angabe eines nichtleeren Werts ermöglicht die Definition bestimmter Modellwarteschlangen für die Erstellung der verwalteten Warteschlangen für Abonnenten temporärer Themen, die unter dieser Verbindung erstellt werden.
XMSC_WPM_BUS_NAME	Für eine Verbindungsfactory der Name des Service Integration Bus, zu dem die Anwendung eine Verbindung herstellt, bzw. für ein Ziel der Name des Service Integration Bus, in dem sich das Ziel befindet.
XMSC_WPM_TOPIC_SPACE	Der Name des Themenbereichs, der das Thema enthält.

Zugehörige Konzepte

ConnectionFactory- und Connection-Objekte

Ein Verbindungsfactoryobjekt (ConnectionFactory) stellt eine Vorlage bereit, die von einer Anwendung zum Erstellen eines Verbindungsobjekts (Connection) verwendet wird. Das Connection-Objekt verwendet die Anwendung wiederum zum Erstellen eines Sitzungsobjekts (Session).

Verbindung zu einem WebSphere Service Integration Bus
Eine XMS-Anwendung kann entweder über eine direkte TCP/IP-Verbindung oder über HTTP over TCP/IP eine Verbindung zu einem WebSphere Service Integration Bus herstellen.

Ziele
XMS-Anwendungen geben mit einem Destination-Objekt das Ziel für gesendete Nachrichten und die Quelle von empfangenen Nachrichten an.

Zielplatzhalterzeichen
XMS bietet Unterstützung für Zielplatzhalterzeichen, um sicherzustellen, dass Platzhalterzeichen an die Stelle übergeben werden können, an der sie für den Abgleich benötigt werden. Es gibt für jeden Servertyp, mit dem XMS zusammenarbeiten kann, ein anderes Platzhalterschema.

Themen-URIs
Der Uniform Resource Identifier (URI) eines Themas gibt den Namen des Themas und ggf. auch eine oder mehrere seiner Eigenschaften an.

Warteschlangen-URIs
Der Uniform Resource Identifier (URI) einer Warteschlange gibt den Namen der Warteschlange und ggf. auch eine oder mehrere ihrer Eigenschaften an.

Temporäre Ziele
XMS-Anwendungen können temporäre Ziele erstellen und verwenden.

Eigenschaftszuordnung für verwaltete Objekte
Damit Anwendungen Objektdefinitionen für IBM WebSphere MQJMS- und WebSphere Application Server-Verbindungsfactorys und -Ziele verwenden können, müssen die Eigenschaften, die aus diesen Definitionen abgerufen werden, den entsprechenden XMS-Eigenschaften zugeordnet werden, die für XMS-Verbindungsfactorys und -Ziele festgelegt werden.

Zugehörige Tasks

Verwaltete Objekte erstellen
Die Definitionen der ConnectionFactory- und Destination-Objekte, die XMS-Anwendungen zum Herstellen einer Verbindung zu einem Messaging-Server benötigen, müssen mit den geeigneten Verwaltungstools erstellt werden.

Zugehörige Verweise

Erforderliche Eigenschaften für verwaltete Destination-Objekte
Eine Anwendung, die ein Ziel erstellt, muss verschiedene Eigenschaften festlegen, die die Anwendung für ein verwaltetes Destination-Objekt verwendet.

Eigenschaften von InitialContext

Eine Übersicht über die Eigenschaften des Objekts InitialContext mit Links zu detaillierteren Referenzinformationen.

Tabelle 32. Eigenschaften von InitialContext

Name der Eigenschaft	Beschreibung
XMSC_IC_URL	Für LDAP- und FileSystem-Kontexte die Adresse des Repositorys, das verwaltete Objekte enthält.

Tabelle 33. Eigenschaften von InitialContext

Name der Eigenschaft	Beschreibung
XMSC_IC_PROVIDER_URL	Wird verwendet, um das JNDI-Namensverzeichnis zu lokalisieren, sodass sich der COS-Namensservicee nicht auf demselben Server wie der Web-Service befinden muss.

Tabelle 33. Eigenschaften von InitialContext (Forts.)	
Name der Eigenschaft	**Beschreibung**
XMSC_IC_SECURITY_AUTHENTICATION	Basierend auf der Java -Kontextschnittstelle SECURITY_AUTHENTICATION. Diese Eigenschaft ist nur auf den COS-Namenskontext anwendbar.
XMSC_IC_SECURITY_CREDENTIALS	Basierend auf der Java Context-Schnittstelle SECURITY_CREDENTIALS. Diese Eigenschaft ist nur auf den COS-Namenskontext anwendbar.
XMSC_IC_SECURITY_PRINCIPAL	Basierend auf der Java Context-Schnittstelle SECURITY_PRINCIPAL. Diese Eigenschaft ist nur auf den COS-Namenskontext anwendbar.
XMSC_IC_SECURITY_PROTOCOL	Basierend auf der Java -Kontextschnittstelle SECURITY_PROTOCOL Diese Eigenschaft ist nur auf den COS-Namenskontext anwendbar.
XMSC_IC_URL	Für LDAP- und FileSystem-Kontexte die Adresse des Repositorys, das verwaltete Objekte enthält. Für COS-Namenskontexte die Adresse des Web-Service, der die Objekte im Verzeichnis sucht.

Zugehörige Konzepte

InitialContext-Eigenschaften
Zu den Parametern des Konstruktors 'InitialContext' gehört die Position des Repositorys für verwaltete Objekte, die als Uniform Resource Indicator (URI) angegeben wird. Damit eine Anwendung eine Verbindung zum Repository herstellen kann, sind zusätzlich zu den im URI enthaltenen Informationen möglicherweise weitere Angaben erforderlich.

URI-Format für XMS-Ausgangskontexte
Die Position des Repositorys für verwaltete Objekte wird als URI-Wert (Uniform Resource Indicator) bereitgestellt. Das URI-Format hängt dabei vom jeweiligen Kontexttyp ab.

Verwaltete Objekte abrufen
Zum Abrufen eines verwalteten Objekts aus dem Repository verwendet XMS die Adresse, die beim Erstellen des InitialContext-Objekts oder in den InitialContext-Eigenschaften bereitgestellt wurde.

Zugehörige Tasks

InitialContext-Objekte
Eine Anwendung muss einen Ausgangskontext erstellen, damit eine Verbindung zum Repository für verwaltete Objekte hergestellt und die erforderlichen verwalteten Objekte von dort abgerufen werden können.

Eigenschaften von Message

Eine Übersicht über die Eigenschaften des Objekts Message mit Links zu detaillierteren Referenzinformationen.

Tabelle 34. Eigenschaften von Message

Name der Eigenschaft	Beschreibung
JMS_IBM_CHARACTER_SET	Die ID (CCSID) des codierten Zeichensatzes oder der Codepage, in dem bzw. in der sich die Zeichenfolgen mit Zeichendaten im Hauptteil der Nachricht befinden, wenn der XMS-Client die Nachricht an das gewünschte Ziel weiterleitet. In XMS hat diese Eigenschaft einen numerischen Wert und ist der CCSID zugeordnet. Diese Eigenschaft basiert allerdings auf einer JMS-Eigenschaft und hat somit den Wert eines Zeichenfolgedatentyps und wird dem Java-Zeichensatz zugeordnet, der diese numerische CCSID darstellt.
JMS_IBM_ENCODING	Es wird angegeben, wie numerische Daten im Nachrichtentext dargestellt werden, wenn der XMS-Client die Nachricht an das vorgesehene Ziel weiterleitet.
JMS_IBM_EXCEPTIONMESSAGE	Text, der beschreibt, warum die Nachricht an das Ausnahmeziel gesendet wurde. Diese Eigenschaft ist schreibgeschützt.
JMS_IBM_EXCEPTIONPROBLEMDESTINATION	Der Name des Ziels, an dem sich die Nachricht befand, bevor sie an das Ausnahmeziel gesendet wurde.
JMS_IBM_EXCEPTIONREASON	Ein Ursachencode, der angibt, warum die Nachricht an das Ausnahmeziel gesendet wurde.
JMS_IBM_EXCEPTIONTIMESTAMP	Die Zeit, zu der die Nachricht an das Ausnahmeziel gesendet wurde.
JMS_IBM_FEEDBACK	Ein Code, der die Art einer Berichtsnachricht angibt.
JMS_IBM_FORMAT	Die Art der Anwendungsdaten in der Nachricht.
JMS_IBM_LAST_MSG_IN_GROUP	Gibt an, ob die Nachricht die letzte Nachricht in einer Nachrichtengruppe ist.
JMS_IBM_MSGTYPE	Der Typ der Nachricht.
JMS_IBM_PUTAPPLTYPE	Der Anwendungstyp, der die Nachricht gesendet hat.
JMS_IBM_PUTDATE	Das Datum, an dem die Nachricht gesendet wurde.
JMS_IBM_PUTTIME	Die Zeit, zu der die Nachricht gesendet wurde.
JMS_IBM_REPORT_COA	Anforderung von Berichtsnachrichten mit 'Bestätigung bei Eingang', wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.
JMS_IBM_REPORT_COD	Anforderung von Berichtsnachrichten mit 'Empfangsbestätigung', wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.
JMS_IBM_REPORT_DISCARD_MSG	Anfordern, dass die Nachricht gelöscht wird, wenn sie nicht an ihr vorgesehenes Ziel zugestellt werden kann.
JMS_IBM_REPORT_EXCEPTION	Anforderung von Ausnahmeberichtsnachrichten, wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.

Tabelle 34. Eigenschaften von Message (Forts.)	
Name der Eigenschaft	**Beschreibung**
JMS_IBM_REPORT_EXPIRATION	Anforderung von Ablaufberichtsnachrichten, wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.
JMS_IBM_REPORT_NAN	Anfordern einer Berichtsnachricht mit einer Bestätigung über eine negative Aktion.
JMS_IBM_REPORT_PAN	Anfordern einer Berichtsnachricht mit einer Bestätigung über eine positive Aktion.
JMS_IBM_REPORT_PASS_CORREL_ID	Anforderung, dass die Korrelations-ID jedes Berichts oder jeder Antwortnachricht mit der Korrelations-ID der ursprünglichen Nachricht identisch sein muss.
JMS_IBM_REPORT_PASS_MSG_ID	Anforderung, dass die Nachrichten-ID jedes Berichts oder jeder Antwortnachricht mit der Nachrichten-ID der ursprünglichen Nachricht identisch sein muss.
JMS_IBM_RETAIN	Durch Festlegen dieser Eigenschaft wird der Warteschlangenmanager angewiesen, eine Nachricht als ständige Veröffentlichung zu behandeln.
JMS_IBM_SYSTEM_MESSAGEID	Eine ID, die die Nachricht innerhalb des Service Integration Bus eindeutig identifiziert. Diese Eigenschaft ist schreibgeschützt.
JMSX_APPID	Der Name der Anwendung, die die Nachricht gesendet hat.
JMSX_DELIVERY_COUNT	Die Anzahl der Versuche zum Zustellen der Nachricht.
JMSX_GROUPID	Die ID der Nachrichtengruppe, zu der die Nachricht gehört.
JMSX_GROUPSEQ	Die Folgenummer der Nachricht innerhalb einer Nachrichtengruppe.
JMSX_USERID	Die Benutzer-ID, die der Anwendung zugeordnet ist, die die Nachricht gesendet hat.

JMS_IBM_MQMD*-Eigenschaften

IBM Message Service Client for .NET ermöglicht es Clientanwendungen, MQMD-Felder über APIs zu lesen bzw. zu schreiben. Außerdem ermöglicht es den Zugriff auf MQ-Nachrichtendaten. Der Zugriff auf MQMD ist standardmäßig inaktiviert und muss explizit von der Anwendung über die Destination-Eigenschaften XMSC_WMQ_MQMD_WRITE_ENABLED und XMSC_WMQ_MQMD_READ_ENABLED aktiviert werden. Diese beiden Eigenschaften sind voneinander unabhängig.

Alle MQMD-Felder mit Ausnahme von StrucId und Version sind als zusätzliche Message-Objekteigenschaften verfügbar und mit dem Präfix JMS_IBM_MQMD versehen.

JMS_IBM_MQMD*-Eigenschaften haben Vorrang vor anderen Eigenschaften wie etwa JMS_IBM*-Eigenschaften, die in der vorherigen Tabelle beschrieben werden.

Nachrichten senden

Mit Ausnahme von StrucId und Version werden alle MQMD-Felder dargestellt. Diese Eigenschaften beziehen sich nur auf die MQMD-Felder; kommt eine Eigenschaft sowohl im MQMD- als auch im MQRFH2-Header vor, wird die Version im MQRFH2 nicht festgelegt oder extrahiert. Mit Ausnahme von JMS_IBM_MQMD_BackoutCount kann jede dieser Eigenschaften festgelegt werden. Ein für JMS_IBM_MQMD_BackoutCount festgelegter Wert wird ignoriert.

Wenn eine Eigenschaft eine maximale Länge hat und Sie einen zu langen Wert angeben, wird der Wert abgeschnitten.

Für bestimmte Eigenschaften müssen Sie auch die Eigenschaft XMSC_WMQ_MQMD_MESSAGE_CONTEXT für das Destination-Objekt festlegen. Die Anwendung muss mit entsprechender Kontextberechtigung ausgeführt werden, damit diese Eigenschaft in Kraft treten kann. Wenn Sie XMSC_WMQ_MQMD_MESSAGE_CONTEXT nicht auf einen geeigneten Wert setzen, wird der Eigenschaftswert ignoriert. Wenn Sie XMSC_WMQ_MQMD_MESSAGE_CONTEXT auf einen geeigneten Wert setzen, aber nicht über eine ausreichende Kontextberechtigung für den Warteschlangenmanager verfügen, wird eine Ausnahme ausgegeben. Eigenschaften, die bestimmte Werte von XMSC_WMQ_MQMD_MESSAGE_CONTEXT erfordern, sind im Folgenden aufgeführt.

Für folgende Eigenschaften muss XMSC_WMQ_MQMD_MESSAGE_CONTEXT den Wert XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT oder XMSC_WMQ_MDCTX_SET_ALL_CONTEXT haben:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityData

Für folgende Eigenschaften muss XMSC_WMQ_MQMD_MESSAGE_CONTEXT den Wert XMSC_WMQ_MDCTX_SET_ALL_CONTEXT haben:

- JMS_IBM_MQMD_PutApplType
- JMS_IBM_MQMD_PutApplName
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginData

Nachrichten empfangen

Alle diese Eigenschaften sind in einer empfangenen Nachricht verfügbar, wenn die Eigenschaft XMSC_WMQ_MQMD_READ_ENABLED auf 'true' gesetzt ist, und zwar unabhängig von den tatsächlichen Eigenschaften, die die produzierende Anwendung festgelegt hat. Gemäß JMS-Spezifikation kann eine Anwendung die Eigenschaften einer empfangenen Nachricht nur dann ändern, wenn zuvor die Werte aller Eigenschaften gelöscht wurden. Die empfangene Nachricht kann ohne Änderung der Eigenschaften weitergeleitet werden.

Anmerkung: Wenn Ihre Anwendung eine Nachricht von einem Ziel empfängt, dessen Eigenschaft XMSC_WMQ_MQMD_READ_ENABLED auf 'true' gesetzt ist, und sie an ein Ziel weiterleitet, dessen Eigenschaft XMSC_WMQ_MQMD_WRITE_ENABLED auf 'true' gesetzt ist, führt dies dazu, dass alle MQMD-Feldwerte der empfangenen Nachricht in die weitergeleitete Nachricht kopiert werden. Eigenschaftentabelle

Tabelle 35. Eigenschaften des Message-Objekts, die MQMD-Felder darstellend

Eigenschaft	Beschreibung	Typ
JMS_IBM_MQMD_REPORT	Optionen für Berichtsnachrichten	System.Int32
JMS_IBM_MQMD_MSGTYPE	Nachrichtentyp	System.Int32
JMS_IBM_MQMD_EXPIRY	Nachrichtenlebensdauer	System.Int32
JMS_IBM_MQMD_FEEDBACK	Rückmeldung oder Ursachencode	System.Int32
JMS_IBM_MQMD_ENCODING	Numerische Codierung von Nachrichtendaten	System.Int32
JMS_IBM_MQMD_CODEDCHARSETID	Zeichensatzkennung von Nachrichtendaten	System.Int32
JMS_IBM_MQMD_FORMAT	Name des Formats von Nachrichtendaten	System.String

Tabelle 35. Eigenschaften des Message-Objekts, die MQMD-Felder darstellend (Forts.)

Eigenschaft	Beschreibung	Typ
JMS_IBM_MQMD_PRIORITY **Anmerkung:** Wenn Sie JMS_IBM_MQMD_PRIORITY einen Wert zuweisen, der nicht im Bereich 0-9 liegt, verstößt dieser Wert gegen die JMS-Spezifikation.	Nachrichtenpriorität	System.Int32
JMS_IBM_MQMD_PERSISTENCE	Nachrichtenpersistenz	System.Int32
JMS_IBM_MQMD_MSGID **Anmerkung:** Die JMS-Spezifikation legt fest, dass die Nachrichten-ID vom JMS-Provider festgelegt werden muss, und sie muss entweder eindeutig oder null sein. Wenn Sie JMS_IBM_MQMD_MSGID einen Wert zuweisen, wird dieser Wert in JMSMessageID kopiert. Er wird also nicht vom JMS-Provider festgelegt und ist möglicherweise nicht eindeutig: Dieser Wert verstößt gegen die JMS-Spezifikation.	Nachrichten-ID	Byte-Array **Anmerkung:** Die Verwendung von Byte-Array-Eigenschaften in einer Nachricht verstößt gegen die JMS-Spezifikation.
JMS_IBM_MQMD_CORRELID **Anmerkung:** Wenn Sie JMS_IBM_MQMD_CORRELID einen Wert zuweisen, der mit der Zeichenfolge 'ID' beginnt, verstößt dieser Wert gegen die JMS-Spezifikation.	Korrelations-ID	Byte-Array **Anmerkung:** Die Verwendung von Byte-Array-Eigenschaften in einer Nachricht verstößt gegen die JMS-Spezifikation.
JMS_IBM_MQMD_BACKOUTCOUNT	Zurücksetzungszähler	System.Int32
JMS_IBM_MQMD_REPLYTOQ	Name der Antwortwarteschlange	System.String
JMS_IBM_MQMD_REPLYTOQMGR	Name des Antwortwarteschlangenmanagers	System.String
JMS_IBM_MQMD_USERIDENTIFIER	Benutzer-ID	System.String
JMS_IBM_MQMD_ACCOUNTINGTOKEN	Abrechnung	Byte-Array **Anmerkung:** Die Verwendung von Byte-Array-Eigenschaften in einer Nachricht verstößt gegen die JMS-Spezifikation.
JMS_IBM_MQMD_APPLIDENTITYDATA	Anwendungsdaten zur Identität	System.String
JMS_IBM_MQMD_PUTAPPLTYPE	Typ der Anwendung, die die Nachricht eingereiht hat	System.Int32
JMS_IBM_MQMD_PUTAPPLNAME	Name der Anwendung, die die Nachricht eingereiht hat.	System.String
JMS_IBM_MQMD_PUTDATE	Datum der Nachrichteneinreihung	System.String
JMS_IBM_MQMD_PUTTIME	Uhrzeit, zu der die Nachricht eingereiht wurde	System.String

Tabelle 35. Eigenschaften des Message-Objekts, die MQMD-Felder darstellend (Forts.)		
Eigenschaft	**Beschreibung**	**Typ**
JMS_IBM_MQMD_APPLORIGINDATA	Anwendungsdaten zum Ursprung	System.String
JMS_IBM_MQMD_GROUPID	Gruppen-ID	Byte-Array **Anmerkung:** Die Verwendung von Byte-Array-Eigenschaften in einer Nachricht verstößt gegen die JMS-Spezifikation.
JMS_IBM_MQMD_MSGSEQNUMBER	Folgenummer einer lokalen Nachricht innerhalb einer Gruppe	System.Int32
JMS_IBM_MQMD_OFFSET	Relative Adresse von Daten in einer physischen Nachricht ab dem Anfang der logischen Nachricht	System.Int32
JMS_IBM_MQMD_MSGFLAGS	Nachrichtenmarkierungen	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	Länge der ursprünglichen Nachricht	System.Int32

Weitere Informationen finden Sie unter MQMD.

Beispiele

In diesem Beispiel wird eine Nachricht in eine Warteschlange oder ein Thema gestellt, wobei MQMD.UserIdentifier auf "JoeBloggs" gesetzt ist.

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(XMSC_WMQ_MQMD_WRITE_ENABLED,
      XMSC_WMQ_MQMD_WRITE_ENABLED_YES);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(XMSC_WMQ_MQMD_MESSAGE_CONTEXT,
  XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty(JMS_IBM_MQMD_USERIDENTIFIER, "JoeBloggs");

// Send the message
// ...
```

Es ist erforderlich, XMSC_WMQ_MQMD_MESSAGE_CONTEXT festzulegen, bevor JMS_IBM_MQMD_USERIDENTIFIER festgelegt wird. Weitere Informationen zur Verwendung von XMSC_WMQ_MQMD_MESSAGE_CONTEXT finden Sie in den Message-Objekteigenschaften.

Entsprechend können Sie die Inhalte der MQMD-Felder extrahieren, indem Sie XMSC_WMQ_MQMD_READ_ENABLED auf 'true' setzen, bevor Sie eine Nachricht empfangen, und dann die Get-Methoden der Nachricht (z. B. getStringProperty) verwenden. Alle empfangenen Eigenschaften sind schreibgeschützt.

In diesem Beispiel wird der Wert des Feldes MQMD.ApplIdentityData einer Nachricht, die aus einer Warteschlange oder einem Thema abgerufen wurde, im Wertfeld abgelegt.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...
```

```
// Enable MQMD read
dest.setBooleanProperty(XMSC_WMQ_MQMD_READ_ENABLED, XMSC_WMQ_MQMD_READ_ENABLED_YES);

// Receive a message
// ...

// Get desired MQMD field value using a property
System.String value = rcvMsg.getStringProperty(JMS_IBM_MQMD_APPLIDENTITYDATA);
```

Eigenschaften von MessageConsumer

Eine Übersicht über die Eigenschaften des Objekts MessageConsumer mit Links zu detaillierteren Referenzinformationen.

Tabelle 36. Eigenschaften von MessageConsumer

Name der Eigenschaft	Beschreibung
XMSC_IS_SUBSCRIPTION_MULTICAST	Es wird angegeben, ob Nachrichten mithilfe von WebSphere MQ Multicast Transport an den Nachrichtenkonsumenten übergeben werden. Diese Eigenschaft ist schreibgeschützt.
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST	Es wird angegeben, ob Nachrichten mithilfe von WebSphere MQ Multicast Transport mit einer zuverlässigen Servicequalität an den Nachrichtenkonsumenten übergeben werden. Diese Eigenschaft ist schreibgeschützt.

Weitere Informationen finden Sie im Abschnitt .NET-Eigenschaften von IMessageConsumer.

Eigenschaften von MessageProducer

Eine Übersicht über die Eigenschaften des Objekts MessageProducer mit Links zu detaillierteren Referenzinformationen.

Weitere Informationen finden Sie im Artikel .NET-Eigenschaften von IMessageProducer.

Eigenschaften von Session

Eine Übersicht über die Eigenschaften des Objekts Session mit Links zu detaillierteren Referenzinformationen.

Weitere Informationen finden Sie im Artikel .NET-Eigenschaften der ISession.

Eigenschaftsdefinitionen

In diesem Thema finden Sie eine Definition jeder Objekteigenschaft.

Jede Eigenschaftsdefinition umfasst folgende Informationen:

- Datentyp der Eigenschaft
- Objekttypen, die die Eigenschaft haben
- Für eine Destination-Eigenschaft den Namen, der in einem Uniform Resource Identifier (URI) verwendet werden kann
- Ausführlichere Beschreibung der Eigenschaft
- Die gültigen Werte der Eigenschaft
- Standardwert der Eigenschaft

Eigenschaften, deren Namen mit einem der folgenden Präfixe beginnen, sind nur für den angegebenen Verbindungstyp relevant:

XMSC_RTT
Die Eigenschaften sind nur für eine Echtzeitverbindung zu einem Broker relevant. Die Namen der Eigenschaften werden als benannte Konstanten in der Headerdatei `xmsc_rtt.h` definiert.

XMSC_WMQ
Die Eigenschaften sind nur relevant, wenn eine Anwendung eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager herstellt. Die Namen der Eigenschaften werden als benannte Konstanten in der Headerdatei `xmsc_wmq.h` definiert.

XMSC_WPM
Die Eigenschaften sind nur relevant, wenn eine Anwendung eine Verbindung zu einem WebSphere Service Integration Bus herstellt. Die Namen der Eigenschaften werden als benannte Konstanten in der Headerdatei `xmsc_wpm.h` definiert.

Sofern in ihren Definitionen nicht anders angegeben, sind die übrigen Eigenschaften für alle Verbindungstypen relevant. Die Namen der Eigenschaften werden als benannte Konstanten in der Headerdatei `xmsc.h` definiert. Eigenschaften, deren Namen mit dem Präfix JMSX beginnen, sind JMS definierte Eigenschaften einer Nachricht, und Eigenschaften, deren Namen mit dem Präfix JMS_IBM beginnen, sind IBM definierte Eigenschaften einer Nachricht. Weitere Informationen zu den Eigenschaften von Nachrichten finden Sie im Abschnitt „Eigenschaften der Eine XMS -Nachricht” auf Seite 77.

Sofern in den zugehörigen Definitionen nicht anders angegeben, ist jede Eigenschaft in der Punkt-zu-Punkt (Point-to-point)- und in der Publish/Subscribe-Domäne relevant.

Eine Anwendung kann den Wert jeder Eigenschaft abrufen und festlegen, es sei denn, die Eigenschaft ist als schreibgeschützt definiert.

Folgende Eigenschaften sind definiert:

JMS_IBM_CHARACTER_SET

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Die ID (CCSID) des codierten Zeichensatzes oder der Codepage, in dem bzw. in der sich die Zeichenfolgen mit Zeichendaten im Hauptteil der Nachricht befinden, wenn der XMS-Client die Nachricht an das gewünschte Ziel weiterleitet. In XMS hat diese Eigenschaft einen numerischen Wert und ist der CCSID zugeordnet. Diese Eigenschaft basiert allerdings auf einer JMS-Eigenschaft und hat somit den Wert eines Zeichenfolgedatentyps und wird dem Java-Zeichensatz zugeordnet, der diese numerische CCSID darstellt. Diese Eigenschaft überschreibt jede CCSID, die durch die Eigenschaft XMSC_WMQ_CCSID für das Ziel angegeben ist.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_ENCODING

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Es wird angegeben, wie numerische Daten im Nachrichtentext dargestellt werden, wenn der XMS-Client die Nachricht an das vorgesehene Ziel weiterleitet. Diese Eigenschaft überschreibt jede Codierung, die durch die Eigenschaft XMSC_WMQ_ENCODING für das Ziel angegeben ist. Die Eigenschaft gibt die Darstellung von binären Ganzzahlen, gepackten Dezimalganzzahlen und Gleitkommazahlen an.

Die gültigen Werte für die Eigenschaft sind mit den Werten identisch, die im Feld *Encoding* eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zum Feld *Encoding* finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Eine Anwendung kann folgende benannte Konstanten verwenden, um die Eigenschaft festzulegen:

Benannte Konstante	Bedeutet
MQENC_INTEGER_NORMAL	Normale Codierung von Ganzzahlen
MQENC_INTEGER_REVERSED	Umgekehrte Codierung von Ganzzahlen
MQENC_DECIMAL_NORMAL	Normale Codierung von gepackten Dezimalzahlen
MQENC_DECIMAL_REVERSED	Umgekehrte Codierung von gepackten Dezimalzahlen
MQENC_FLOAT_IEEE_NORMAL	Normale Codierung von IEEE-Gleitkomma
MQENC_FLOAT_IEEE_REVERSED	Umgekehrte Codierung von IEEE-Gleitkomma
MQENC_FLOAT_S390	z/OS Architektur-Gleitkomma-Kodierung
MQENC_NATIVE	Native Systemcodierung

Um einen Wert für die Eigenschaft zu bilden, kann die Anwendung drei dieser Konstanten wie folgt hinzufügen:

- Eine Konstante, deren Name mit MQENC_INTEGER beginnt, um die Darstellung von binären Ganzzahlen anzugeben
- Eine Konstante, deren Name mit MQENC_DECIMAL beginnt, um die Darstellung von gepackt dezimalen Ganzzahlen anzugeben
- Eine Konstante, deren Name mit MQENC_FLOAT beginnt, um die Darstellung von Gleitkommazahlen anzugeben

Alternativ kann die Anwendung die Eigenschaft auf MQENC_NATIVE setzen, dessen Wert umgebungsabhängig ist.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_EXCEPTIONMESSAGE

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Text, der beschreibt, warum die Nachricht an das Ausnahmeziel gesendet wurde. Diese Eigenschaft ist schreibgeschützt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt und eine Nachricht von einem Ausnahmeziel empfängt.

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Der Name des Ziels, an dem sich die Nachricht befand, bevor sie an das Ausnahmeziel gesendet wurde.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt und eine Nachricht von einem Ausnahmeziel empfängt.

JMS_IBM_EXCEPTIONREASON

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Ein Ursachencode, der angibt, warum die Nachricht an das Ausnahmeziel gesendet wurde.

Eine Liste aller möglichen Ursachencodes finden Sie in der Definition der Klasse com.ibm.websphere.sib.SIRCConstants in der vom Javadoc-Tool generierten Dokumentation, die im Lieferumfang von WebSphere Application Serverenthalten ist.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt und eine Nachricht von einem Ausnahmeziel empfängt.

JMS_IBM_EXCEPTIONTIMESTAMP

Datentyp:
System.Int64

Eigenschaft von:
Nachricht

Die Zeit, zu der die Nachricht an das Ausnahmeziel gesendet wurde.

Die Zeit wird in Millisekunden seit 00:00:00 GMT am 1. Januar 1970 ausgedrückt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt und eine Nachricht von einem Ausnahmeziel empfängt.

JMS_IBM_FEEDBACK

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Ein Code, der die Art einer Berichtsnachricht angibt.

Die gültigen Werte für die Eigenschaft sind die Rückkopplungscodes und Ursachencodes, die im Feld **Feedback** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zum Feld **Feedback** finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

JMS_IBM_FORMAT

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Die Art der Anwendungsdaten in der Nachricht.

Die gültigen Werte der Eigenschaft sind mit den Werten identisch, die im Feld **Format** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zum Feld **Format** finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_LAST_MSG_IN_GROUP

Datentyp:
System.Boolean

Eigenschaft von:
Nachricht

Gibt an, ob die Nachricht die letzte Nachricht in einer Nachrichtengruppe ist.

Setzen Sie die Eigenschaft auf "true", wenn es sich bei der Nachricht um die letzte Nachricht in einer Nachrichtengruppe handelt. Setzen Sie die Eigenschaft andernfalls auf "false" oder setzen Sie sie gar nicht. Die Eigenschaft ist standardmäßig nicht festgelegt.

Der Wert 'true' entspricht dem Statusflag MQMF_LAST_MSG_IN_GROUP, das im Feld **MsgFlags** eines Nachrichtendeskriptors angegeben werden kann. Weitere Informationen zu diesem Attribut finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Diese Eigenschaft wird in der Publish/Subscribe-Domäne ignoriert und ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_MSGTYPE

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Der Typ der Nachricht.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
MQMT_DATAGRAM	Für die Nachricht ist keine Antwort erforderlich.
MQMT_REQUEST	Für die Nachricht ist eine Antwort erforderlich.
MQMT_REPLY	Die Nachricht ist eine Antwortnachricht.
MQMT_REPORT	Die Nachricht ist eine Berichtsnachricht.

Diese Werte entsprechen den Nachrichtentypen, die im Feld **MsgType** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zum Feld **MsgType** finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_PUTAPPLTYPE

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Der Anwendungstyp, der die Nachricht gesendet hat.

Die gültigen Werte der Eigenschaft sind die Anwendungstypen, die im Feld **PutApplType** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zum Feld **PutApplType** finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_PUTDATE

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Das Datum, an dem die Nachricht gesendet wurde.

Die gültigen Werte der Eigenschaft sind mit den Werten identisch, die im Feld **PutDate** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zum Feld **PutDate** finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_PUTTIME

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Die Zeit, zu der die Nachricht gesendet wurde.

Die gültigen Werte der Eigenschaft sind mit den Werten identisch, die im Feld **PutTime** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zum Feld **PutTime** finden Sie in der Veröffentlichung *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMS_IBM_REPORT_COA

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anforderung von Berichtsnachrichten mit 'Bestätigung bei Eingang', wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
MQRO_COA	Anforderung von Berichtsnachrichten mit 'Bestätigung bei Eingang', wobei keine Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.
MQRO_COA_WITH_DATA	Anforderung von Berichtsnachrichten mit 'Bestätigung bei Eingang', wobei die ersten 100 Bytes der Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.

Gültiger Wert	Bedeutet
MQRO_COA_WITH_FULL_DATA	Anforderung von Berichtsnachrichten mit 'Bestätigung bei Eingang', wobei alle Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.

Diese Werte entsprechen den Berichtsoptionen, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zu diesen Optionen finden Sie im Handbuch *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

JMS_IBM_REPORT_COD

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anforderung von Berichtsnachrichten mit 'Empfangsbestätigung', wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
MQRO_COD	Anforderung von Berichtsnachrichten mit 'Empfangsbestätigung', wobei keine Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.
MQRO_COD_WITH_DATA	Anforderung von Berichtsnachrichten mit 'Empfangsbestätigung', wobei die ersten 100 Bytes der Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.
MQRO_COD_WITH_FULL_DATA	Anforderung von Berichtsnachrichten mit 'Empfangsbestätigung', wobei alle Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.

Diese Werte entsprechen den Berichtsoptionen, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zu diesen Optionen finden Sie im Handbuch *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

JMS_IBM_REPORT_DISCARD_MSG

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anfordern, dass die Nachricht gelöscht wird, wenn sie nicht an ihr vorgesehenes Ziel zugestellt werden kann.

Setzen Sie die Eigenschaft auf MQRO_DISCARD_MSG, um anzufordern, dass die Nachricht gelöscht wird, wenn Sie nicht an ihr vorgesehenes Ziel zugestellt werden kann. Wenn Sie möchten, dass die Nachricht stattdessen in eine Warteschlange für nicht zustellbare Nachrichten eingereiht oder an ein Ausnahmeziel gesendet wird, legen Sie die Eigenschaft nicht fest. Die Eigenschaft ist standardmäßig nicht festgelegt.

Der Wert MQRO_DISCARD_MSG entspricht einer Berichtsoption, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden kann. Weitere Informationen zu dieser Option enthält das Handbuch *IBM WebSphere MQ Application Programming Reference*.

JMS_IBM_REPORT_EXCEPTION

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anforderung von Ausnahmeberichtsnachrichten, wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
MQRO_EXCEPTION	Anforderung von Ausnahmeberichtsnachrichten, wobei keine Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.
MQRO_EXCEPTION_WITH_DATA	Anforderung von Ausnahmeberichtsnachrichten, wobei die ersten 100 Bytes der Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.
MQRO_EXCEPTION_WITH_FULL_DATA	Anforderung von Ausnahmeberichtsnachrichten, wobei alle Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.

Diese Werte entsprechen den Berichtsoptionen, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zu diesen Optionen finden Sie im Handbuch *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

JMS_IBM_REPORT_EXPIRATION

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anforderung von Ablaufberichtsnachrichten, wobei angegeben wird, wie viele Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden müssen.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
MQRO_EXPIRATION	Anforderung von Ablaufberichtsnachrichten, wobei keine Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.
MQRO_EXPIRATION_WITH_DATA	Anforderung von Ablaufberichtsnachrichten, wobei die ersten 100 Bytes der Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.

Gültiger Wert	Bedeutet
MQRO_EXPIRATION_WITH_FULL_DATA	Anforderung von Ablaufberichtsnachrichten, wobei alle Anwendungsdaten aus der ursprünglichen Nachricht in eine Berichtsnachricht eingeschlossen werden.

Diese Werte entsprechen den Berichtsoptionen, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zu diesen Optionen finden Sie im Handbuch *IBM WebSphere MQ Application Programming Reference*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

JMS_IBM_REPORT_NAN

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anfordern einer Berichtsnachricht mit einer Bestätigung über eine negative Aktion.

Setzen Sie die Eigenschaft auf MQRO_NAN, um Berichtsnachrichten mit einer Bestätigung über eine negative Aktion anzufordern. Wenn keine Berichtsnachrichten mit einer Bestätigung über eine negative Aktion angefordert werden sollen, legen Sie die Eigenschaft nicht fest. Die Eigenschaft ist standardmäßig nicht festgelegt.

Der Wert MQRO_NAN entspricht einer Berichtsoption, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden kann. Weitere Informationen zu dieser Option enthält das Handbuch *IBM WebSphere MQ Application Programming Reference*.

JMS_IBM_REPORT_PAN

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anfordern einer Berichtsnachricht mit einer Bestätigung über eine positive Aktion.

Setzen Sie die Eigenschaft auf MQRO_PAN, um Berichtsnachrichten mit einer Bestätigung über eine positive Aktion anzufordern. Wenn keine Berichtsnachrichten mit einer Bestätigung über eine positive Aktion angefordert werden sollen, legen Sie die Eigenschaft nicht fest. Die Eigenschaft ist standardmäßig nicht festgelegt.

Der Wert MQRO_PAN entspricht einer Berichtsoption, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden kann. Weitere Informationen zu dieser Option enthält das Handbuch *IBM WebSphere MQ Application Programming Reference*.

JMS_IBM_REPORT_PASS_CORREL_ID

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anforderung, dass die Korrelations-ID jedes Berichts oder jeder Antwortnachricht mit der Korrelations-ID der ursprünglichen Nachricht identisch sein muss.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
MQRO_PASS_CORREL_ID	Anforderung, dass die Korrelations-ID jedes Berichts oder jeder Antwortnachricht mit der Korrelations-ID der ursprünglichen Nachricht identisch sein muss.
MQRO_COPY_MSG_ID_TO_CORREL_ID	Anforderung, dass die Korrelations-ID jedes Berichts oder jeder Antwortnachricht mit der Nachrichten-ID der ursprünglichen Nachricht identisch sein muss.

Diese Werte entsprechen den Berichtsoptionen, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zu diesen Optionen finden Sie im Handbuch *IBM WebSphere MQ Application Programming Reference*.

Der Standardwert der Eigenschaft ist MQRO_COPY_MSG_ID_TO_CORREL_ID.

JMS_IBM_REPORT_PASS_MSG_ID

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Anforderung, dass die Nachrichten-ID jedes Berichts oder jeder Antwortnachricht mit der Nachrichten-ID der ursprünglichen Nachricht identisch sein muss.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
MQRO_PASS_MSG_ID	Anforderung, dass die Nachrichten-ID jedes Berichts oder jeder Antwortnachricht mit der Nachrichten-ID der ursprünglichen Nachricht identisch sein muss.
MQRO_NEW_MSG_ID	Anforderung, dass für jeden Bericht oder jede Antwortnachricht eine neue Nachrichten-ID generiert wird.

Diese Werte entsprechen den Berichtsoptionen, die im Feld **Report** eines Nachrichtendeskriptors angegeben werden können. Weitere Informationen zu diesen Optionen finden Sie im Handbuch *IBM WebSphere MQ Application Programming Reference*.

Der Standardwert der Eigenschaft ist MQRO_NEW_MSG_ID.

JMS_IBM_RETAIN

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Durch Festlegen dieser Eigenschaft wird der Warteschlangenmanager angewiesen, eine Nachricht als ständige Veröffentlichung zu behandeln. Wenn ein Subskribent Nachrichten von Themen empfängt, empfängt er direkt nach dem Subskribieren möglicherweise zusätzliche Nachrichten, die über die in früheren Releases empfangenen Nachrichten hinausgehen. Bei diesen Nachrichten handelt es sich um die optionalen ständigen Veröffentlichungen für die subskribierten Themen. Für jedes mit der Subskription übereinstimmende Thema wird, sofern es dafür eine ständige Veröffentlichung gibt, diese Veröffentlichung für die Zustellung an den subskribierenden Nachrichtenkonsumenten verfügbar gemacht.

RETAIN_PUBLICATION ist der einzige gültige Wert für diese Eigenschaft. Diese Eigenschaft ist standardmäßig nicht festgelegt.

Anmerkung: Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

JMS_IBM_SYSTEM_MESSAGEID

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Eine ID, die die Nachricht innerhalb des Service Integration Bus eindeutig identifiziert. Diese Eigenschaft ist schreibgeschützt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Service Integration Bus herstellt.

JMSX_APPID

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Der Name der Anwendung, die die Nachricht gesendet hat.

Bei dieser Eigenschaft handelt es sich um die JMS-definierte Eigenschaft mit dem JMS-Namen JMSXAppID. Weitere Informationen zu der Eigenschaft finden Sie in der *Java Message Service Specification Version 1.1*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

JMSX_DELIVERY_COUNT

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Die Anzahl der Versuche zum Zustellen der Nachricht.

Bei dieser Eigenschaft handelt es sich um die JMS-definierte Eigenschaft mit dem JMS-Namen JMSXDeliveryCount. Weitere Informationen zu der Eigenschaft finden Sie in der *Java Message Service Specification Version 1.1*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

JMSX_GROUPID

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Die ID der Nachrichtengruppe, zu der die Nachricht gehört.

Bei dieser Eigenschaft handelt es sich um die JMS-definierte Eigenschaft mit dem JMS-Namen JMSXGroupID. Weitere Informationen zu der Eigenschaft finden Sie in der *Java Message Service Specification Version 1.1*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

JMSX_GROUPSEQ

Datentyp:
System.Int32

Eigenschaft von:
Nachricht

Die Folgenummer der Nachricht innerhalb einer Nachrichtengruppe.

Bei dieser Eigenschaft handelt es sich um die JMS-definierte Eigenschaft mit dem JMS-Namen JMSXGroupSeq. Weitere Informationen zu der Eigenschaft finden Sie in der *Java Message Service Specification Version 1.1*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

JMSX_USERID

Datentyp:
Zeichenfolge

Eigenschaft von:
Nachricht

Die Benutzer-ID, die der Anwendung zugeordnet ist, die die Nachricht gesendet hat.

Bei dieser Eigenschaft handelt es sich um die JMS-definierte Eigenschaft mit dem JMS-Namen JMSXUserID. Weitere Informationen zu der Eigenschaft finden Sie in der *Java Message Service Specification Version 1.1*.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist für eine Echtzeitverbindung zu einem Broker nicht gültig.

XMSC_ASYNC_EXCEPTIONS

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Diese Eigenschaft bestimmt, ob XMS einen ExceptionListener nur dann informiert, wenn eine Verbindung unterbrochen wird oder wenn eine Ausnahmebedingung asynchron zu einem XMS-API-Aufruf auftritt. Diese Eigenschaft gilt für alle Connections (Verbindungen), die ausdieser ConnectionFactory erstellt werden und für die ein ExceptionListenerregistriert ist.

Gültige Werte für diese Eigenschaft sind:

XMSC_ASYNC_EXCEPTIONS_ALL
Alle Ausnahmebedingungen, die asynchron außerhalb des Bereichs eines synchronen API-Aufrufs erkannt wurden, und alle Ausnahmebedingungen durch unterbrochene Verbindungen werden an den ExceptionListener gesendet.

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN
Nur Ausnahmebedingungen, die eine unterbrochene Verbindung angeben, werden an den ExceptionListener gesendet. Alle anderen Ausnahmebedingungen, die während der asynchronen Verarbeitung auftreten, werden nicht an den ExceptionListener gemeldet, d. h. die Anwendung wird über diese Ausnahmebedingungen nicht informiert.

Diese Eigenschaft ist standardmäßig auf XMSC_ASYNC_EXCEPTIONS_ALL gesetzt.

XMSC_CLIENT_ID

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Die Client-ID für eine Verbindung.

Eine Client-ID wird nur verwendet, um permanente Subskriptionen in der Publish/Subscribe-Domäne zu unterstützen, und wird in der Punkt-zu-Punkt (Point-to-point)-Domäne ignoriert. Weitere Informationen zu Einstellung von Client-IDs finden Sie im Abschnitt „ConnectionFactory- und Connection-Objekte" auf Seite 24.

Diese Eigenschaft ist für eine Echtzeitverbindung zu einem Broker nicht relevant.

XMSC_CONNECTION_TYPE

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Der Typ des Messaging-Servers, zu dem eine Anwendung eine Verbindung herstellt.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_CT_RTT	Eine Echtzeitverbindung zu einem Broker.
XMSC_CT_WMQ	Eine Verbindung zu einem IBM WebSphere MQ-Warteschlangenmanager.
XMSC_CT_WPM	Eine Verbindung zu einem WebSphere Service Integration Bus.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_DELIVERY_MODE

Datentyp:
System.Int32

Eigenschaft von:
Ziel

In einem URI verwendeter Name:

`persistence` (für ein IBM WebSphere MQ-Ziel)
`deliveryMode` (für ein WebSphere-Standard-Messaging-Provider-Ziel)

Der Zustellmodus von Nachrichten, die an das Ziel gesendet werden.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_DELIVERY_NOT_PERSISTENT	Eine an das Ziel gesendete Nachricht ist nicht persistent. Der Standardzustellmodus des Nachrichtenproduzenten oder ein im Sendeaufruf angegebener Zustellmodus wird ignoriert. Wenn das Ziel eine IBM WebSphere MQ-Warteschlange ist, wird der Wert des Warteschlangenattributs *`DefPersistence`* ebenfalls ignoriert.

Gültiger Wert	Bedeutet
XMSC_DELIVERY_PERSISTENT	Eine an das Ziel gesendete Nachricht ist persistent. Der Standardzustellmodus des Nachrichtenproduzenten oder ein im Sendeaufruf angegebener Zustellmodus wird ignoriert. Wenn das Ziel eine IBM WebSphere MQ-Warteschlange ist, wird der Wert des Warteschlangenattributs *DefPersistence* ebenfalls ignoriert.
XMSC_DELIVERY_AS_APP	Eine Nachricht, die an das Ziel gesendet wird, hat den im Sendeaufruf angegebenen Zustellmodus. Wenn im Sendeaufruf kein Zustellmodus angegeben ist, wird stattdessen der Standardzustellmodus des Nachrichtenproduzenten verwendet. Wenn das Ziel eine IBM WebSphere MQ-Warteschlange ist, wird der Wert des Warteschlangenattributs *DefPersistence* ignoriert.
XMSC_DELIVERY_AS_DEST	Wenn das Ziel eine IBM WebSphere MQ-Warteschlange ist, hat eine Nachricht, die in die Warteschlange eingereiht wird, den Zustellmodus, der durch den Wert des Warteschlangenattributs *DefPersistence* angegeben ist. Der Standardzustellmodus des Nachrichtenproduzenten oder ein im Sendeaufruf angegebener Zustellmodus wird ignoriert. Wenn das Ziel keine IBM WebSphere MQ-Warteschlange ist, entspricht die Bedeutung der von XMSC_DELIVERY_AS_APP.

Der Standardwert ist XMSC_DELIVERY_AS_APP.

XMSC_IC_PROVIDER_URL

Datentyp:
Zeichenfolge

Eigenschaft von:
InitialContext

Wird verwendet, um das JNDI-Namensverzeichnis zu lokalisieren, sodass sich der COS-Namensservicee nicht auf demselben Server wie der Web-Service befinden muss.

XMSC_IC_SECURITY_AUTHENTICATION

Datentyp:
Zeichenfolge

Eigenschaft von:
InitialContext

Basierend auf der Java -Kontextschnittstelle SECURITY_AUTHENTICATION. Diese Eigenschaft ist nur auf den COS-Namenskontext anwendbar.

XMSC_IC_SECURITY_CREDENTIALS

Datentyp:
Zeichenfolge

Eigenschaft von:
InitialContext

Basierend auf der Java Context-Schnittstelle SECURITY_CREDENTIALS. Diese Eigenschaft ist nur auf den COS-Namenskontext anwendbar.

XMSC_IC_SECURITY_PRINCIPAL

Datentyp:
Zeichenfolge

Eigenschaft von:
InitialContext

Basierend auf der Java Context-Schnittstelle SECURITY_PRINCIPAL. Diese Eigenschaft ist nur auf den COS-Namenskontext anwendbar.

XMSC_IC_SECURITY_PROTOCOL

Datentyp:
Zeichenfolge

Eigenschaft von:
InitialContext

Basierend auf der Java -Kontextschnittstelle SECURITY_PROTOCOL Diese Eigenschaft gilt nur für den COS-Namenskontext.

XMSC_IC_URL

Datentyp:
Zeichenfolge

Eigenschaft von:
InitialContext

Für LDAP- und FileSystem-Kontexte die Adresse des Repositorys, das verwaltete Objekte enthält.

Für COS-Namenskontexte die Adresse des Web-Service, der die Objekte im Verzeichnis sucht.

XMSC_IS_SUBSCRIPTION_MULTICAST

Datentyp:
System.Boolean

Eigenschaft von:
MessageConsumer

Es wird angegeben, ob Nachrichten mithilfe von WebSphere MQ Multicast Transport an den Nachrichtenkonsumenten übergeben werden. Diese Eigenschaft ist schreibgeschützt.

Der Wert der Eigenschaft ist "true", wenn Nachrichten mithilfe von WebSphere MQ Multicast Transport an den Nachrichtenkonsumenten zugestellt werden. Andernfalls ist der Wert "false".

Diese Eigenschaft ist nur für eine Echtzeitverbindung zu einem Broker relevant.

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

Datentyp:
System.Boolean

Eigenschaft von:
MessageConsumer

Es wird angegeben, ob Nachrichten mithilfe von WebSphere MQ Multicast Transport mit einer zuverlässigen Servicequalität an den Nachrichtenkonsumenten übergeben werden. Diese Eigenschaft ist schreibgeschützt.

Der Wert der Eigenschaft ist "true", wenn Nachrichten mithilfe von WebSphere MQ Multicast Transport mit einer zuverlässigen Servicequalität an den Nachrichtenkonsumenten zugestellt werden. Andernfalls ist der Wert "false".

Diese Eigenschaft ist nur für eine Echtzeitverbindung zu einem Broker relevant.

XMSC_JMS_MAJOR_VERSION

Datentyp:
System.Int32

Eigenschaft von:
ConnectionMetaData

Die Hauptversionsnummer der JMS -Spezifikation, auf der XMS basiert. Diese Eigenschaft ist schreibgeschützt.

XMSC_JMS_MINOR_VERSION

Datentyp:
System.Int32

Eigenschaft von:
ConnectionMetaData

Die untergeordnete Versionsnummer der JMS -Spezifikation, auf der XMS basiert. Diese Eigenschaft ist schreibgeschützt.

XMSC_JMS_VERSION

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionMetaData

Die Versions-ID der JMS -Spezifikation, auf der XMS basiert. Diese Eigenschaft ist schreibgeschützt.

XMSC_MAJOR_VERSION

Datentyp:
System.Int32

Eigenschaft von:
ConnectionMetaData

Die Versionsnummer des XMS-Clients. Diese Eigenschaft ist schreibgeschützt.

XMSC_MINOR_VERSION

Datentyp:
System.Int32

Eigenschaft von:
ConnectionMetaData

Die Releasenummer des XMS-Client. Diese Eigenschaft ist schreibgeschützt.

XMSC_PASSWORD

Datentyp:
Byte-Array

Eigenschaft von:
ConnectionFactory

Ein Kennwort, das zur Authentifizierung der Anwendung verwendet werden kann, wenn versucht wird, eine Verbindung zu einem Messaging-Server herzustellen. Das Kennwort wird zusammen mit der Eigenschaft XMSC_USERID verwendet.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Wenn Sie eine Verbindung zu IBM WebSphere MQ auf verteilten Plattformen herstellen und die Eigenschaft XMSC_USERID der Verbindungsfactory festlegen, muss diese Eigenschaft mit der Benutzer-ID (**userid**) des angemeldeten Benutzers übereinstimmen. Wenn Sie diese Eigenschaft nicht festlegen, verwendet der Warteschlangenmanager standardmäßig die **userid** des angemeldeten Benutzers. Wenn eine weitere Authentifizierung einzelner Benutzer auf Verbindungsebene erforderlich ist, können Sie einen Clientauthentifizierungsexit schreiben, der in IBM WebSphere MQ konfiguriert wird. Weitere Informationen zur Erstellung eines Exits zur Clientauthentifizierung finden Sie im Handbuch für IBM WebSphere MQ-Clients im Abschnitt zur Authentifizierung.

Wenn Sie den Benutzer beim Herstellen einer Verbindung zu IBM WebSphere MQ unter z/OS authentifizieren, müssen Sie einen Sicherheitsexit verwenden.

XMSC_PRIORITY

Datentyp:
System.Int32

Eigenschaft von:
Ziel

In einem URI verwendeter Name:
priority

Die Priorität von Nachrichten, die an das Ziel gesendet werden.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
Eine Ganzzahl im Bereich 0 (niedrigste Priorität) bis 9 (höchste Priorität).	Eine Nachricht, die an das Ziel gesendet wird, hat die angegebene Priorität. Die Standardpriorität des Nachrichtenproduzenten oder eine im Sendeaufruf angegebene Priorität wird ignoriert. Wenn das Ziel eine IBM WebSphere MQ-Warteschlange ist, wird der Wert des Warteschlangenattributs **DefPriority** ebenfalls ignoriert.
XMSC_PRIORITY_AS_APP	Eine Nachricht, die an das Ziel gesendet wird, hat die im Sendeaufruf angegebene Priorität. Wenn im Sendeaufruf keine Priorität angegeben ist, wird stattdessen die Standardpriorität des Nachrichtenproduzenten verwendet. Wenn das Ziel eine IBM WebSphere MQ-Warteschlange ist, wird der Wert des Warteschlangenattributs **DefPriority** ignoriert.
XMSC_PRIORITY_AS_DEST	Wenn das Ziel eine IBM WebSphere MQ-Warteschlange ist, hat eine Nachricht, die in die Warteschlange eingereiht wird, die Priorität, die durch den Wert des Warteschlangenattributs **DefPriority** angegeben ist. Die Standardpriorität des Nachrichtenproduzenten oder eine im Sendeaufruf angegebene Priorität wird ignoriert. Wenn das Ziel keine IBM WebSphere MQ-Warteschlange ist, entspricht die Bedeutung der von XMSC_PRIORITY_AS_APP.

Der Standardwert ist XMSC_PRIORITY_AS_APP.

WebSphere MQ Real-Time Transport und WebSphere MQ Multicast Transport führen keine Aktion auf Basis der Priorität einer Nachricht aus.

XMSC_PROVIDER_NAME

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionMetaData

Der Provider des XMS-Clients. Diese Eigenschaft ist schreibgeschützt.

XMSC_RTT_BROKER_PING_INTERVAL

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Das Zeitintervall in Millisekunden, nach dem XMS .NET die Verbindung zum Echtzeit-Messaging-Server überprüft, um eine Aktivität zu erkennen. Wird keine Aktivität erkannt, initiiert der Client einen Ping; die Verbindung wird geschlossen, wenn keine Antwort auf den Ping erkannt wird.

Der Standardwert der Eigenschaft ist 30000.

XMSC_RTT_CONNECTION_PROTOCOL

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Das Kommunikationsprotokoll, das für eine Echtzeitverbindung zu einem Broker verwendet wird.

Der Wert der Eigenschaft muss XMSC_RTT_CP_TCP sein, was eine Echtzeitverbindung zu einem Broker über TCP/IP bedeutet. Der Standardwert ist XMSC_RTT_CP_TCP.

XMSC_RTT_HOST_NAME

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Hostname oder die IP-Adresse des Systems, auf dem ein Broker ausgeführt wird.

Diese Eigenschaft wird zusammen mit der Eigenschaft XMSC_RTT_PORT zur Identifizierung des Brokers verwendet.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_RTT_LOCAL_ADDRESS

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Hostname oder die IP-Adresse der lokalen Netzschnittstelle, die für eine Echtzeitverbindung zu einem Broker verwendet werden soll.

Diese Eigenschaft ist nur nützlich, wenn das System, auf dem die Anwendung aktiv ist, über zwei oder mehr Netzschnittstellen verfügt, und Sie in der Lage sein müssen, die Schnittstelle anzugeben, die für eine Echtzeitverbindung verwendet werden muss. Falls das System nur über eine Netzschnittstelle verfügt, kann nur diese Schnittstelle verwendet werden. Wenn das System über zwei oder mehr Netzschnittstellen verfügt und die Eigenschaft nicht festgelegt ist, wird die Schnittstelle nach dem Zufallsprinzip ausgewählt.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_RTT_MULTICAST

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory und Destination

In einem URI verwendeter Name:
mulicast

Die Multicasteinstellung für eine Verbindungsfactory oder ein Ziel. Nur ein Ziel, das ein Thema ist, kann diese Eigenschaft haben.

Eine Anwendung verwendet diese Eigenschaft, um Multicasting in Zusammenhang mit einer Echtzeitverbindung zu einem Broker zu aktivieren und um, falls Multicasting aktiviert ist, genau anzugeben, wie das Multicasting zum Zustellen von Nachrichten vom Broker an einen Nachrichtenkonsumenten eingesetzt wird. Die Eigenschaft hat keine Auswirkungen darauf, wie ein Nachrichtenproduzent Nachrichten an den Broker sendet.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_RTT_MULTICAST_DISABLED	Nachrichten werden nicht mithilfe von WebSphere MQ Multicast Transport an einen Nachrichtenkonsumenten gesendet. Dieser Wert ist der Standardwert für ein ConnectionFactory-Objekt.
XMSC_RTT_MULTICAST_ASCF	Die Nachrichten werden entsprechend der Multicasting-Einstellung für die Verbindungsfactory, die dem Nachrichtenkonsumenten zugeordnet ist, an den Nachrichtenkonsumenten zugestellt. Die Multicasteinstellung für die Verbindungsfactory wird zum Zeitpunkt des Erstellens der Verbindung berücksichtigt. Dieser Wert ist nur für ein Destination-Objekt gültig und ist der Standardwert für ein Destination-Objekt.
XMSC_RTT_MULTICAST_ENABLED	Wenn das Thema für Multicasting im Broker konfiguriert ist, werden Nachrichten mithilfe von WebSphere MQ Multicast Transport an den Nachrichtenkonsumenten zugestellt. Wenn das Topic-Objekt für zuverlässiges Multicasting konfiguriert ist, wird eine zuverlässige Servicequalität verwendet.
XMSC_RTT_MULTICAST_RELIABLE	Wenn das Thema für zuverlässiges Multicasting im Broker konfiguriert ist, werden Nachrichten mithilfe von WebSphere MQ Multicast Transport mit einer zuverlässigen Servicequalität an den Nachrichtenkonsumenten zugestellt. Ist das Topic-Objekt nicht für zuverlässiges Multicasting konfiguriert, können Sie keinen Nachrichtenkonsumenten für das Topic-Objekt erstellen.

Gültiger Wert	Bedeutet
XMSC_RTT_MULTICAST_NOT_RELIABLE	Wenn das Thema für Multicasting im Broker konfiguriert ist, werden Nachrichten mithilfe von WebSphere MQ Multicast Transport an den Nachrichtenkonsumenten zugestellt. Auch wenn das Topic-Objekt für zuverlässiges Multicasting konfiguriert ist, wird keine zuverlässige Servicequalität verwendet.

XMSC_RTT_PORT

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Nummer des Ports, an dem ein Broker für eingehende Anforderungen empfangsbereit ist. Auf dem Broker müssen Sie einen Real-timeInput- oder Real-timeOptimizedFlow-Nachrichtenverarbeitungsknoten konfigurieren, der an diesem Port empfangsbereit ist.

Diese Eigenschaft wird zusammen mit der Eigenschaft XMSC_RTT_HOST_NAME zur Identifizierung des Brokers verwendet.

Der Standardwert der Eigenschaft ist XMSC_RTT_DEFAULT_PORT oder 1506.

XMSC_TIME_TO_LIVE

Datentyp:
System.Int32

Eigenschaft von:
Ziel

In einem URI verwendeter Name:

expiry (für ein IBM WebSphere MQ-Ziel)
timeToLive (für ein WebSphere-Standard-Messaging-Provider-Ziel)

Die Lebensdauer für Nachrichten, die an das Ziel gesendet werden.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
0	Eine Nachricht, die an das Ziel gesendet wird, läuft nie ab.
Eine positive Ganzzahl	Eine Nachricht, die an das Ziel gesendet wird, hat die angegebene Lebensdauer in Millisekunden. Die Standardlebenszeit des Nachrichtenproduzenten oder eine im Sendeaufruf angegebene Lebenszeit wird ignoriert.
XMSC_TIME_TO_LIVE_AS_APP	Eine Nachricht, die an das Ziel gesendet wird, hat die im Sendeaufruf angegebenen Lebensdauer. Wenn im Sendeaufruf keine Lebensdauer angegeben ist, wird stattdessen die Standardlebensdauer des Nachrichtenproduzenten verwendet.

Der Standardwert ist XMSC_TIME_TO_LIVE_AS_APP.

XMSC_USERID

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Eine Benutzer-ID, die zur Authentifizierung der Anwendung verwendet werden kann, wenn versucht wird, eine Verbindung zu einem Messaging-Server herzustellen. Die Benutzer-ID wird zusammen mit der Eigenschaft XMSC_PASSWORD verwendet.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Wenn Sie eine Verbindung zu IBM WebSphere MQ auf verteilten Plattformen herstellen und die Eigenschaft XMSC_USERID der Verbindungsfactory festlegen, muss diese mit der Benutzer-ID (**userid**) des angemeldeten Benutzers übereinstimmen. Wenn Sie diese Eigenschaft nicht festlegen, verwendet der Warteschlangenmanager standardmäßig die **userid** des angemeldeten Benutzers. Wenn Sie eine weitere Authentifizierung einzelner Benutzer auf Verbindungsebene benötigen, können Sie einen Clientauthentifizierungsexit schreiben, der in IBM WebSphere MQkonfiguriert ist.

Wenn Sie den Benutzer beim Herstellen einer Verbindung zu IBM WebSphere MQ unter z/OS authentifizieren, müssen Sie einen Sicherheitsexit verwenden.

XMSC_VERSION

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionMetaData

Die Versions-ID des XMS-Clients. Diese Eigenschaft ist schreibgeschützt.

XMSC_WMQ_BROKER_CONTROLQ

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name der Steuerwarteschlange, die von einem Broker verwendet wird.

Der Standardwert der Eigenschaft ist SYSTEM.BROKER.CONTROL.QUEUE.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WMQ_BROKER_PUBQ

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name der Warteschlange, die von einem Broker überwacht wird, wo Anwendungen Nachrichten senden, die sie veröffentlichen.

Der Standardwert der Eigenschaft ist SYSTEM.BROKER.DEFAULT.STREAM.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WMQ_BROKER_QMGR

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name des Warteschlangenmanagers, mit dem ein Broker verbunden ist.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WMQ_BROKER_SUBQ

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name der Subskribentenwarteschlange für einen nicht permanenten Nachrichtenkonsumenten.

Der Name der Subskribentenwarteschlange muss mit folgenden Zeichen beginnen:

SYSTEM.JMS.ND.

Wenn Sie möchten, dass alle nicht permanenten Nachrichtenkonsumenten eine Subskribentenwarteschlange gemeinsam nutzen, geben Sie den vollständigen Namen der gemeinsam genutzten Warteschlange an. Eine Warteschlange mit dem angegebenen Namen muss vorhanden sein, bevor eine Anwendung einen nicht permanenten Nachrichtenkonsumenten erstellen kann.

Wenn jeder nicht permanente Nachrichtenkonsument Nachrichten aus einer eigenen exklusiven Subskribentenwarteschlange abrufen soll, geben Sie einen Warteschlangennamen an, der mit einem Stern (*) endet. Wenn eine Anwendung dann einen nicht permanenten Nachrichtenkonsumenten erstellt, erstellt der XMS-Client eine dynamische Warteschlange für die ausschließliche Verwendung durch den Nachrichtenkonsumenten. Der XMS-Client legt anhand des Werts der Eigenschaft den Inhalt des Feldes **DynamicQName** in dem Objektdeskriptor fest, der zum Erstellen der dynamischen Warteschlange verwendet wird.

Der Standardwert der Eigenschaft ist SYSTEM.JMS.ND.SUBSCRIBER.QUEUE. Dies bedeutet, dass XMS den Ansatz mit der gemeinsam genutzten Warteschlange standardmäßig verwendet.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WMQ_BROKER_VERSION

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory und Destination

In einem URI verwendeter Name:
brokerVersion

Der Typ des Brokers, der von der Anwendung für eine Verbindung oder für das Ziel verwendet wird. Nur ein Ziel, das ein Thema ist, kann diese Eigenschaft haben.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_WMQ_BROKER_V1	Die Anwendung verwendet einen IBM WebSphere MQ Publish/Subscribe-Broker. Die Anwendung kann diesen Wert auch verwenden, wenn Sie von IBM WebSphere MQ Publish/Subscribe auf WebSphere Event Broker oder WebSphere Message Broker migrieren, die Anwendung jedoch nicht geändert haben.
XMSC_WMQ_BROKER_V2	Die Anwendung verwendet einen Broker von WebSphere Event Broker oder WebSphere Message Broker.

Gültiger Wert	Bedeutet
XMSC_WMQ_BROKER_UNSPECIFIED	Legen Sie nach der Migration des Brokers von Version 6 auf Version 7 diese Eigenschaft so fest, dass RFH2 -Header nicht mehr verwendet werden. Nach der Migration ist diese Eigenschaft nicht mehr relevant.

Der Standardwert für eine Verbindungsfactory ist XMSC_WMQ_BROKER_UNSPECIFIED, aber standardmäßig wird die Eigenschaft nicht für ein Ziel festgelegt. Wenn Sie die Eigenschaft für ein Ziel festlegen, wird jeder Wert, der durch die Eigenschaft der Verbindungsfactory angegeben ist, überschrieben.

XMSC_WMQ_CCDTURL

Datentyp:
System.String

Eigenschaft von:
ConnectionFactory

Eine URL (Uniform Resource Locator), die den Namen und die Position der Datei angibt, die die Kanaldefinitionstabelle des Clients enthält, und auch angibt, wie auf die Datei zugegriffen werden kann.

Diese Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WMQ_CCSID

Datentyp:
System.Int32

Eigenschaft von:
Ziel

In einem URI verwendeter Name:
CCSID

Die ID (CCSID) des codierten Zeichensatzes oder der Codepage, in dem bzw. in der sich die Zeichenfolgen mit Zeichendaten im Hauptteil einer Nachricht befinden, wenn der XMS-Client die Nachricht an das Ziel weiterleitet. Wenn die Eigenschaft JMS_IBM_CHARACTER_SET für eine einzelne Nachricht festgelegt ist, überschreibt sie die CCSID, die durch diese Eigenschaft für das Ziel festgelegt wird.

Der Standardwert der Eigenschaft ist 1208.

Diese Eigenschaft ist nur für Nachrichten relevant, die an das Ziel gesendet werden, nicht für Nachrichten, die vom Ziel empfangen werden.

XMSC_WMQ_CHANNEL

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name des Kanals, der für eine Verbindung verwendet werden soll.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Clientmodus herstellt.

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Mit dieser Eigenschaft werden die Optionen zur Verbindungswiederherstellung für neue Verbindungen angegeben, die von dieser Factory erstellt wurden. Sie befindet sich in XMSC und ermöglicht die Einstellung einer der folgenden Optionen:

- WMQ_CLIENT_RECONNECT_AS_DEF (Standardwert). Verwenden Sie den in der Datei `mqclient.ini` angegebenen Wert. Legen Sie den Wert mithilfe der Eigenschaft **DefRecon** in der Zeilengruppe 'Channels' fest. Einer der folgenden Werte kann festgelegt werden:
  1. Ja. Das Verhalten entspricht dem der Option WMQ_CLIENT_RECONNECT.
  2. NEIN. Default. Es wird keine Verbindungswiederholungsoption angegeben.
  3. QMGR (Warteschlangenmanager). Das Verhalten entspricht dem der Option WMQ_CLIENT_RECONNECT_Q_MGR.
  4. DISABLED. Das Verhalten entspricht dem der Option WMQ_CLIENT_RECONNECT_DISABLED.
- WMQ_CLIENT_RECONNECT. Die Verbindung wird zu einem der Warteschlangenmanager wiederhergestellt, die in der Verbindungsnamensliste angegeben sind.
- WMQ_CLIENT_RECONNECT_Q_MGR. Die Verbindung wird zu dem Warteschlangenmanager wiederhergestellt, zu dem die Verbindung ursprünglich bestand. Es wird `MQRC_RECONNECT_QMID_MISMATCH` zurückgegeben, wenn der Warteschlangenmanager, zu dem eine Verbindung hergestellt werden soll (angegeben in der Verbindungsnamensliste), eine andere QMID hat als der Warteschlangenmanager, zu dem die Verbindung ursprünglich bestand.
- WMQ_CLIENT_RECONNECT_DISABLED. Die Verbindungswiederholung ist inaktiviert.

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Mit dieser Eigenschaft wird die Dauer in Sekunden angegeben, während der eine Clientverbindung versucht, eine Verbindung wiederherzustellen.

Nachdem er versucht hat, die Verbindung innerhalb dieses Zeitraums wiederherzustellen, schlägt der Client mit MQRC_RECONNECT_FAILED fehl. Die Standardeinstellung für diese Eigenschaft ist XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT.

Der Standardwert dieser Eigenschaft ist 1800.

XMSC_WMQ_CONNECTION_MODE

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Der Modus, in dem eine Anwendung eine Verbindung zu einem Warteschlangenmanager herstellt.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_WMQ_CM_BINDINGS	Eine Verbindung zu einem Warteschlangenmanager im Bindungsmodus, um eine optimale Leistung zu erreichen. Dieser Wert ist der Standardwert für C/C++.

Gültiger Wert	Bedeutet
XMSC_WMQ_CM_CLIENT	Eine Verbindung zu einem Warteschlangenmanager im Client-modus, um einen vollständig verwalteten Stack sicherzustellen. Dieser Wert ist der Standardwert für .NET.
XMSC_WMQ_CM_CLIENT_UNMANAGED (nur für .NET)	Eine Verbindung zu einem Warteschlangenmanager, die einen nicht verwalteten Client-Stack erzwingt.

Zugehörige Konzepte

Verwaltete und nicht verwaltete Operationen in .NET

Verwalteter Code wird ausschließlich in der Umgebung der .NET Common Language Runtime ausgeführt und ist vollständig von den in dieser Laufzeit bereitgestellten Services abhängig. Eine Anwendung wird als 'nicht verwaltet' klassifiziert, wenn ein Teil der Anwendung oder Services, die von der Anwendung aufgerufen werden, außerhalb der Umgebung der .NET Common Language Runtime ausgeführt werden.

XMSC_WMQ_CONNECTION_NAME_LIST

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Mit dieser Eigenschaft werden die Hosts angegeben, zu denen der Client eine Verbindung herstellen will, nachdem die Verbindungen unterbrochen wurden.

Die Verbindungsnamensliste ist eine durch Kommas getrennte Liste mit Host/IP-Port-Paaren. Die Standardeinstellung für diese Eigenschaft ist WMQ_CONNECTION_NAME_LIST_DEFAULT.

Beispiel: `127.0.0.1(1414),host2.example.com(1400)`

Die Standardeinstellung dieser Eigenschaft ist `localhost (1414)`.

XMSC_WMQ_DUR_SUBQ

Datentyp:
Zeichenfolge

Eigenschaft von:
Ziel

Der Name der Subskribentenwarteschlange für einen permanenten Subskribenten, der Nachrichten von dem Ziel empfängt. Nur ein Ziel, das ein Thema ist, kann diese Eigenschaft haben.

Der Name der Subskribentenwarteschlange muss mit folgenden Zeichen beginnen:

SYSTEM.JMS.D.

Wenn Sie möchten, dass alle permanenten Subskribenten eine Subskribentenwarteschlange gemeinsam nutzen, geben Sie den vollständigen Namen der gemeinsam genutzten Warteschlange an. Eine Warteschlange mit dem angegebenen Namen muss vorhanden sein, bevor eine Anwendung einen permanenten Subskribenten erstellen kann.

Wenn jeder dauerhafte Abonnent Nachrichten aus seiner eigenen exklusiven Abonnentenwarteschlange abrufen soll, geben Sie einen Warteschlangennamen an, der mit einem Stern (*) endet. Wenn eine Anwendung dann einen permanenten Abonnenten erstellt, erstellt der XMS-Client eine dynamische Warteschlange für die ausschließliche Verwendung durch den permanenten Abonnenten. Der XMS-Client legt anhand des Werts der Eigenschaft den Inhalt des Feldes **DynamicQName** in dem Objektdeskriptor fest, der zum Erstellen der dynamischen Warteschlange verwendet wird.

Der Standardwert der Eigenschaft ist SYSTEM.JMS.D.SUBSCRIBER.QUEUE. Dies bedeutet, dass XMS den Ansatz mit der gemeinsam genutzten Warteschlange standardmäßig verwendet.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WMQ_ENCODING

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Es wird angegeben, wie numerische Daten im Nachrichtentext dargestellt werden, wenn der XMS-Client die Nachricht an das Ziel weiterleitet. Wenn die Eigenschaft JMS_IBM_ENCODING für eine einzelne Nachricht festgelegt ist, überschreibt sie die Codierung, die durch diese Eigenschaft für das Ziel festgelegt wird. Die Eigenschaft gibt die Darstellung von binären Ganzzahlen, gepackten Dezimalganzzahlen und Gleitkommazahlen an.

Die gültigen Werte der Eigenschaft sind mit den Werten identisch, die im Feld **Encoding** eines Nachrichtendeskriptors angegeben werden können.

Eine Anwendung kann folgende benannte Konstanten verwenden, um die Eigenschaft festzulegen:

Benannte Konstante	Bedeutet
MQENC_INTEGER_NORMAL	Normale Codierung von Ganzzahlen
MQENC_INTEGER_REVERSED	Umgekehrte Codierung von Ganzzahlen
MQENC_DECIMAL_NORMAL	Normale Codierung von gepackten Dezimalzahlen
MQENC_DECIMAL_REVERSED	Umgekehrte Codierung von gepackten Dezimalzahlen
MQENC_FLOAT_IEEE_NORMAL	Normale Codierung von IEEE-Gleitkomma
MQENC_FLOAT_IEEE_REVERSED	Umgekehrte Codierung von IEEE-Gleitkomma
MQENC_FLOAT_S390	Gleitkommacodierung der z/OS -Architektur
MQENC_NATIVE	Native Systemcodierung

Um einen Wert für die Eigenschaft zu bilden, kann die Anwendung drei dieser Konstanten wie folgt hinzufügen:

- Eine Konstante, deren Name mit MQENC_INTEGER beginnt, um die Darstellung von binären Ganzzahlen anzugeben
- Eine Konstante, deren Name mit MQENC_DECIMAL beginnt, um die Darstellung von gepackt dezimalen Ganzzahlen anzugeben
- Eine Konstante, deren Name mit MQENC_FLOAT beginnt, um die Darstellung von Gleitkommazahlen anzugeben

Alternativ kann die Anwendung die Eigenschaft auf MQENC_NATIVE setzen, dessen Wert umgebungsabhängig ist.

Der Standardwert der Eigenschaft ist MQENC_NATIVE.

Diese Eigenschaft ist nur für Nachrichten relevant, die an das Ziel gesendet werden, nicht für Nachrichten, die vom Ziel empfangen werden.

XMSC_WMQ_FAIL_IF_QUIESCE

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory und Destination

In einem URI verwendeter Name:
failIfQuiesce

Gibt an, ob Aufrufe bestimmter Methoden fehlschlagen, wenn sich der Warteschlangenmanager, mit dem die Anwendung verbunden ist, im Quiesce-Zustand befindet.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_WMQ_FIQ_YES	Aufrufe bestimmter Methoden schlagen fehl, wenn sich der Warteschlangenmanager im Quiesce-Zustand befindet. Wenn die Anwendung erkennt, dass der Warteschlangenmanager in den Quiesce-Zustand wechselt, kann sie ihre aktuelle Task beenden und die Verbindung schließen, damit der Warteschlangenmanager gestoppt werden kann.
XMSC_WMQ_FIQ_NO	Es schlagen keine Methodenaufrufe fehl, weil sich der Warteschlangenmanager im Quiesce-Zustand befindet. Wenn Sie diesen Wert angeben, kann die Anwendung nicht erkennen, dass der Warteschlangenmanager in den Quiesce-Zustand wechselt. Die Anwendung fährt möglicherweise fort, Operationen für den Warteschlangenmanager auszuführen und verhindert so, dass der Warteschlangenmanager gestoppt wird.

Der Standardwert für eine Verbindungsfactory ist XMSC_WMQ_FIQ_YES, aber standardmäßig wird die Eigenschaft nicht für ein Ziel festgelegt. Wenn Sie die Eigenschaft für ein Ziel festlegen, wird jeder Wert, der durch die Eigenschaft der Verbindungsfactory angegeben ist, überschrieben.

XMSC_WMQ_MESSAGE_BODY

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Diese Eigenschaft legt fest, ob eine XMS -Anwendung den MQRFH2 einer IBM WebSphere MQ -Nachricht als Teil der Nachrichtennutzdaten (also als Teil des Nachrichtenhauptteils) verarbeitet.

Anmerkung: Beim Senden von Nachrichten an ein Ziel setzt die Eigenschaft XMSC_WMQ_MESSAGE_BODY die vorhandene XMS-Destination-Eigenschaft XMSC_WMQ_TARGET_CLIENT außer Kraft.

Gültige Werte für diese Eigenschaft sind:

XMSC_WMQ_MESSAGE_BODY_JMS
Empfangen: Typ und Hauptteil der eingehende XMS-Nachricht werden durch den Inhalt des MQRFH2 (falls vorhanden) oder des MQMD (falls kein MQRFH2 vorhanden ist) in der empfangenen IBM WebSphere MQ-Nachricht bestimmt.

Senden: Der Hauptteil der abgehenden XMS-Nachricht enthält einen vorangestellten und automatisch generierten MQRFH2-Header auf Basis von XMS-Nachrichteneigenschaften und Headerfeldern.

XMSC_WMQ_MESSAGE_BODY_MQ
Empfangen: Der Typ der eingehenden XMS-Nachricht ist immer ByteMessage, unabhängig vom Inhalt der empfangenen IBM WebSphere MQ-Nachricht oder dem Formatfeld des empfangenen MQMD. Der Hauptteil der XMS-Nachricht besteht aus den unveränderten Nachrichtendaten, die vom zugrunde liegenden API-Aufruf des Messaging-Providers zurückgegeben werden. Der Zeichensatz und die Codierung der Daten im Nachrichtenhauptteil werden durch die Felder 'CodedCharSetId' und 'Encoding' des MQMD bestimmt. Das Format der Daten im Nachrichtenhauptteil wird durch das Feld 'Format' des MQMD festgelegt.

Senden: Der Hauptteil der abgehenden XMS-Nachricht enthält die unveränderten Anwendungsnutzdaten; es wird kein automatisch generierter IBM WebSphere MQ-Header zum Hauptteil hinzugefügt.

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED
Empfangen: Der XMS-Client bestimmt einen geeigneten Wert für diese Eigenschaft. Im Empfangspfad ist dieser Wert der Wert der Eigenschaft WMQ_MESSAGE_BODY_JMS.

Senden: Der XMS-Client bestimmt einen geeigneten Wert für diese Eigenschaft. Im Sendepfad ist dieser Wert der Wert der Eigenschaft XMSC_WMQ_TARGET_CLIENT.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED gesetzt.

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Legt fest, welche Stufe des Nachrichtenkontexts von der Anwendung XMS festgelegt werden soll. Die Anwendung muss mit entsprechender Kontextberechtigung ausgeführt werden, damit diese Eigenschaft in Kraft treten kann.

Die gültigen Werte für diese Eigenschaft sind:

XMSC_WMQ_MDCTX_DEFAULT
Bei abgehenden Nachrichten sind im API-Aufruf MQOPEN und in der MQPMO-Struktur keine expliziten Nachrichtenkontextoptionen angegeben.

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT
Der API-Aufruf MQOPEN gibt die Nachrichtenkontextoption MQOO_SET_IDENTITY_CONTEXT an und die MQPMO-Struktur gibt MQPMO_SET_IDENTITY_CONTEXT an.

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT
Der API-Aufruf MQOPEN gibt die Nachrichtenkontextoption MQOO_SET_ALL_CONTEXT an und die MQPMO-Struktur gibt MQPMO_SET_ALL_CONTEXT an.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ_MDCTX_DEFAULT gesetzt.

Anmerkung: Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zum System Integration Bus herstellt.

Für folgende Eigenschaften muss die Eigenschaft XMSC_WMQ_MQMD_MESSAGE_CONTEXT beim Senden einer Nachricht auf den Eigenschaftswert XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT oder den Eigenschaftswert XMSC_WMQ_MDCTX_SET_ALL_CONTEXT gesetzt werden, damit die gewünschte Wirkung erzielt wird:

- JMS_IBM_MQMD_USERIDENTIFIER
- JMS_IBM_MQMD_ACCOUNTINGTOKEN
- JMS_IBM_MQMD_APPLIDENTITYDATA

Für folgende Eigenschaften muss die Eigenschaft XMSC_WMQ_MQMD_MESSAGE_CONTEXT beim Senden einer Nachricht auf den Eigenschaftswert XMSC_WMQ_MDCTX_SET_ALL_CONTEXT gesetzt werden, damit die gewünschte Wirkung erzielt wird:

- JMS_IBM_MQMD_PUTAPPLTYPE
- JMS_IBM_MQMD_PUTAPPLNAME
- JMS_IBM_MQMD_PUTDATE
- JMS_IBM_MQMD_PUTTIME
- JMS_IBM_MQMD_APPLORIGINDATA

XMSC_WMQ_MQMD_READ_ENABLED

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Diese Eigenschaft bestimmt, ob eine XMS-Anwendung die Werte von MQMD-Feldern extrahieren kann oder nicht.

Die gültigen Werte für diese Eigenschaft sind:

XMSC_WMQ_READ_ENABLED_NO
Beim Senden von Nachrichten werden die JMS_IBM_MQMD*-Eigenschaften einer gesendeten Nachricht nicht mit den aktualisierten Feldwerten in der MQMD-Struktur aktualisiert.

Beim Empfangen von Nachrichten ist keine der JMS_IBM_MQMD*-Eigenschaften in einer empfangenen Nachricht verfügbar, auch wenn der Absender einige oder alle dieser Eigenschaften festgelegt hat.

XMSC_WMQ_READ_ENABLED_YES
Beim Senden von Nachrichten werden alle JMS_IBM_MQMD*-Eigenschaften einer gesendeten Nachricht mit den aktualisierten Feldwerten in der MQMD-Struktur aktualisiert, einschließlich derjenigen Eigenschaften, die der Absender nicht explizit festgelegt hat.

Beim Empfangen von Nachrichten sind alle JMS_IBM_MQMD*-Eigenschaften in einer empfangenen Nachricht verfügbar, einschließlich derjenigen Eigenschaften, die der Absender nicht explizit festgelegt hat.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ_READ_ENABLED_NO gesetzt.

XMSC_WMQ_MQMD_WRITE_ENABLED

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Diese Eigenschaft bestimmt, ob eine XMS-Anwendung die Werte von MQMD-Feldern verarbeiten kann oder nicht.

Die gültigen Werte für diese Eigenschaft sind:

XMSC_WMQ_WRITE_ENABLED_NO
Alle JMS_IBM_MQMD*-Eigenschaften werden ignoriert; ihre Werte werden nicht in die zugrunde liegende MQMD-Struktur kopiert.

XMSC_WMQ_WRITE_ENABLED_YES
Die JMS_IBM_MQMD*-Eigenschaften werden verarbeitet. Ihre Werte werden in die zugrunde liegende MQMD-Struktur kopiert.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ_WRITE_ENABLED_NO gesetzt.

XMSC_WMQ_PUT_ASYNC_ALLOWED

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Diese Eigenschaft gibt an, ob Nachrichtenproduzenten asynchrone PUT-Operationen verwenden dürfen, um Nachrichten an diese Zieladresse zu senden.

Die gültigen Werte für diese Eigenschaft sind:

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST
Bestimmt, ob asynchrone PUT-Operationen zulässig sind, indem auf die Definition des Queue- oder des Topic-Objekts verwiesen wird.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_Q_DEF
Bestimmt, ob asynchrone PUT-Operationen zulässig sind, indem auf die Definition des Queue-Objekts verwiesen wird.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_TOPIC_DEF
Bestimmt, ob asynchrone PUT-Operationen zulässig sind, indem auf die Definition des Topic-Objekts verwiesen wird.

XMSC_WMQ _PUT_ASYNC_ALLOWED_DISABLED
Asynchrone PUT-Operationen sind nicht zulässig.

XMSC_WMQ _PUT_ASYNC_ALLOWED_ENABLED
Asynchrone PUT-Operationen sind zulässig.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST gesetzt.

Anmerkung: Diese Eigenschaft ist nicht relevant, wenn eine Anwendung eine Verbindung zum System Integration Bus herstellt.

XMSC_WMQ_READ_AHEAD_ALLOWED

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Diese Eigenschaft bestimmt, ob Nachrichtenkonsumenten und Warteschlangenbrowser die Vorauslesefunktion verwenden dürfen, um nicht permanente, nicht transaktionsorientierte Nachrichten von diesem Ziel in einen internen Puffer abzurufen, bevor sie die Nachrichten empfangen.

Die gültigen Werte für diese Eigenschaft sind:

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF
Bestimmt, ob die Vorauslesefunktion zulässig ist, indem auf die Definition des Queue-Objekts verwiesen wird.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_ TOPIC _DEF
Bestimmt, ob die Vorauslesefunktion zulässig ist, indem auf die Definition des Topic-Objekts verwiesen wird.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST
Bestimmt, ob die Vorauslesefunktion zulässig ist, indem auf die Definition des Queue- oder des Topic-Objekts verwiesen wird.

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED
Die Vorauslesefunktion ist beim Konsumieren oder Browsing von Nachrichten nicht zulässig.

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED
Die Vorauslesefunktion ist zulässig.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ _READ_AHEAD_ALLOWED_AS_DEST gesetzt.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

Datentyp:
System.Int32

Eigenschaft von:
Ziel

Diese Eigenschaft gibt für Nachrichten, die an einen asynchronen Nachrichtenlistener zugestellt werden, an, was mit Nachrichten im internen Vorauslesepuffer geschehen soll, wenn der Nachrichtenkonsument geschlossen wird.

Die Eigenschaft ist bei der Angabe von Optionen zum Schließen von Warteschlangen anwendbar, wenn Nachrichten von einem Ziel verarbeitet werden, und nicht anwendbar, wenn Nachrichten an ein Ziel gesendet werden.

Diese Eigenschaft wird für Warteschlangenbrowser ignoriert, da die Nachrichten bei diesem Vorgang weiter in den Warteschlangen verfügbar sind.

Die gültigen Werte für diese Eigenschaft sind:

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT
Vor der Rückgabe wird nur der aktuelle Nachrichtenlisteneraufruf abgeschlossen, wobei Nachrichten im internen Vorauslesepuffer verbleiben können, die anschließend gelöscht werden.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL
Alle Nachrichten im internen Vorauslesepuffer werden vor der Rückgabe an den Anwendungsnachrichtenlistener zugestellt.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ _READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT gesetzt.

Anmerkung:

- **Abnormale Anwendungsbeendigung**

  Alle Nachrichten im Vorauslesepuffer gehen verloren, wenn eine XMS-Anwendung abnormal beendet wird.

- **Auswirkungen auf Transaktionen**

  Das Vorauslesen ist inaktiviert, wenn die Anwendungen eine Transaktion verwenden. Die Anwendungen sehen daher keine Unterschiede im Verhalten, wenn Sie mit Sitzungen mit Transaktionsunterstützung arbeiten.

- **Auswirkungen auf Sitzungsbestätigungsmodi**

  Das Vorauslesen ist für eine Sitzung ohne Transaktionsunterstützung aktiviert, wenn der Bestätigungsmodus entweder XMSC_AUTO_ACKNOWLEDGE oder XMSC_DUPS_OK_ACKNOWLEDGE ist. Das Vorauslesen ist inaktiviert, wenn der Sitzungsbestätigungsmodus XMSC_CLIENT_ACKNOWLEDGE ist, unabhängig davon, ob es sich um Sitzungen mit oder ohne Transaktionsunterstützung handelt.

- **Auswirkungen auf Warteschlangenbrowser und Warteschlangenbrowserselektoren**

  Die in XMS-Anwendungen eingesetzten Warteschlangenbrowser und Warteschlangenbrowserselektoren nutzen den Leistungsvorteil durch die Vorauslesefunktion. Das Schließen des Warteschlangenbrowsers vermindert nicht die Leistung, weil die Nachricht für weitere Operationen in der Warteschlange verfügbar bleibt. Abgesehen von Leistungsvorteilen durch das Vorauslesen gibt es keine anderen Auswirkungen auf Warteschlangenbrowser und Warteschlangenbrowserselektoren.

- **Auswirkungen von Vorauslesezieleigenschaften auf WebSphere Nachrichtenbroker V6 oder frühere Warteschlangenmanager**

  Die Angabe der Zieleigenschaften XMSC_WMQ_READ_AHEAD_ALLOWED und XMSC_WMQ_READ_AHEAD_CLOSE_POLICY, wenn die XMS -Anwendung den WebSphere Nachrichtenbroker V6 -Warteschlangenmanager verwendet, kann die angegebenen Werte nicht verwenden. Diese Zieleigenschaftswerte werden stillschweigend ignoriert und die Anwendungen arbeiten ohne Vorauslesen weiter. Bei der Verwendung mit V6 -Warteschlangenmanagern werden keine Fehler ausgelöst.

XMSC_WMQ_HOST_NAME

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Hostname oder die IP-Adresse des Systems, auf dem ein Warteschlangenmanager ausgeführt wird.

Diese Eigenschaft wird nur verwendet, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Clientmodus herstellt. Die Eigenschaft wird zusammen mit der Eigenschaft XMSC_WMQ_PORT verwendet, um den Warteschlangenmanager anzugeben.

Der Standardwert der Eigenschaft ist `localhost`.

XMSC_WMQ_LOCAL_ADDRESS

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Für eine Verbindung zu einem Warteschlangenmanager gibt diese Eigenschaft die zu verwendende lokale Netzschnittstelle und/oder den zu verwendenden lokalen Port bzw. den Bereich der zu verwendenden lokalen Ports an.

Der Wert der Eigenschaft ist eine Zeichenfolge in folgenden Format:

[*Hostname*][(*niedrigster_Port*)[,*höchster_Port*])]

Die Variablen haben folgende Bedeutung:

Hostname
Der Hostname oder die IP-Adresse der lokalen Netzschnittstelle, die für die Verbindung verwendet werden soll.

Die Angabe dieser Eigenschaft ist nur dann notwendig, wenn das System, auf dem die Anwendung aktiv ist, über zwei oder mehr Netzschnittstellen verfügt und Sie in der Lage sein müssen, die Schnittstelle anzugeben, die für die Verbindung verwendet werden muss. Falls das System nur über eine Netzschnittstelle verfügt, kann nur diese Schnittstelle verwendet werden. Wenn das System über zwei oder mehr Netzschnittstellen verfügt und Sie nicht angeben, welche Schnittstelle verwendet werden soll, wird die Schnittstelle nach dem Zufallsprinzip ausgewählt.

niedrigster_Port
Die Nummer des lokalen Ports, der für die Verbindung verwendet werden soll.

Wenn *höchster_Port* ebenfalls angegeben ist, wird *niedrigster_Port* als die niedrigste Portnummer in einem Bereich von Portnummern interpretiert.

höchster_Port
Die höchste Portnummer in einem Bereich von Portnummern. Einer der Ports im angegebenen Bereich muss für die Verbindung verwendet werden.

Die maximale Länge der Zeichenfolge beträgt 48 Zeichen.

Hier einige Beispiele für gültige Werte der Eigenschaft:

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Clientmodus herstellt.

XMSC_WMQ_MESSAGE_SELECTION

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Es wird bestimmt, ob die Nachrichtenauswahl vom XMS-Client oder vom Broker vorgenommen wird.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_WMQ_MSEL_CLIENT	Die Nachrichtenauswahl erfolgt durch den XMS-Client.
XMSC_WMQ_MSEL_BROKER	Die Nachrichtenauswahl erfolgt durch den Broker.

Der Standardwert ist XMSC_WMQ_MSEL_CLIENT.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant. Die Nachrichtenauswahl durch den Broker wird nicht unterstützt, wenn die Eigenschaft XMSC_WMQ_BROKER_VERSION auf XMSC_WMQ_BROKER_V1 gesetzt ist.

XMSC_WMQ_MSG_BATCH_SIZE

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die maximale Anzahl Nachrichten, die bei Verwendung einer asynchronen Nachrichtenübermittlung in einem einzigen Stapel aus einer Warteschlange abgerufen werden sollen.

Wenn eine Anwendung mit asynchroner Nachrichtenübermittlung arbeitet, ruft der XMS-Client unter bestimmten Bedingungen einen Stapel von Nachrichten aus einer Warteschlange ab, bevor er die Nachrichten einzeln an die Anwendung weiterleitet. Diese Eigenschaft gibt die maximale Anzahl Nachrichten an, die im Stapel enthalten sein können.

Der Wert der Eigenschaft ist eine positive Ganzzahl und der Standardwert ist 10. Sie sollten die Eigenschaft nur dann auf einen anderen Wert setzen, wenn es ein spezifisches Leistungsproblem gibt, dass Sie beheben müssen.

Wenn eine Anwendung über ein Netz mit einem Warteschlangenmanager verbunden ist, kann die Erhöhung des Werts dieser Eigenschaft den Netzaufwand und die Antwortzeiten reduzieren, aber die für die Speicherung der Nachrichten auf dem Clientsystem erforderliche Speicherkapazität erhöhen. Umgekehrt kann die Verringerung des Werts dieser Eigenschaft den Netzaufwand und die Antwortzeiten erhöhen, aber die für die Speicherung der Nachrichten erforderliche Speicherkapazität reduzieren.

XMSC_WMQ_POLLING_INTERVAL

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Wenn sich bei den einzelnen Nachrichtenlistenern innerhalb einer Sitzung keine geeignete Nachricht in der zugehörigen Warteschlange befindet, ist dieser Wert das maximale Intervall in Millisekunden, das verstreicht, bevor die einzelnen Nachrichtenlistener erneut versuchen, eine Nachricht aus der zugehörigen Warteschlange abzurufen.

Wenn regelmäßig keine geeigneten Nachrichten für die Nachrichtenlistener innerhalb einer Sitzung verfügbar sind, sollten Sie einen höheren Wert für diese Eigenschaft angeben.

Der Wert der Eigenschaft ist eine positive ganze Zahl. Der Standardwert ist 5000.

XMSC_WMQ_PORT

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Nummer des Ports, an dem ein Warteschlangenmanager für eingehende Anforderungen empfangsbereit ist.

Diese Eigenschaft wird nur verwendet, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Clientmodus herstellt. Die Eigenschaft wird zusammen mit der Eigenschaft XMSC_WMQ_HOST_NAME verwendet, um den Warteschlangenmanager anzugeben.

Der Standardwert der Eigenschaft ist XMSC_WMQ_DEFAULT_CLIENT_PORT oder 1414.

XMSC_WMQ_PROVIDER_VERSION

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Die Version, das Release, die Modifikationsstufe und das Fixpack des Warteschlangenmanagers, zu dem die Anwendung eine Verbindung herstellen soll. Gültige Werte für diese Eigenschaft sind:

- Nicht angegeben

Oder eine Zeichenfolge in einem der folgenden Formate:

- V.R.M.F
- V.R.M
- V.R
- V

Dabei stehen V, R, M und F für Ganzzahlwerte größer-gleich 0.

Der Wert 7 oder höher gibt an, dass diese Version als IBM WebSphere MQ Version 7.0-Verbindungsfactory für Verbindungen zu einem IBM WebSphere MQ Version 7.0-Warteschlangenmanager vorgesehen ist. Ein kleinerer Wert als 7 (beispielsweise "6.0.2.0") gibt an, dass die Verwendung mit Warteschlangenmanagern vor der Version 7.0 erfolgen soll. Bei Angabe des Standardwerts (unspecified) sind Verbindungen zu Warteschlangenmanagern jeder Version zugelassen, wobei die gültigen Eigenschaften und die verfügbare Funktionalität von den Leistungsmerkmalen des jeweiligen Warteschlangenmanagers abhängig sind.

Diese Eigenschaft ist standardmäßig auf "unspecified" gesetzt.

Anmerkung:

- Wenn XMSC_WMQ_PROVIDER_VERSION auf 6.2 gesetzt ist, erfolgt keine Socketfreigabe.
- Die Verbindung schlägt fehl, wenn XMSC_WMQ_PROVIDER_VERSION auf 7 gesetzt ist und auf dem Server SHARECNV für den Kanal auf 0 gesetzt ist.
- IBM WebSphere MQ Version 7.0-spezifische Funktionen sind inaktiviert, wenn XMSC_WMQ_PROVIDER_VERSION auf UNSPECIFIED und SHARECNV auf 0 gesetzt ist.

Die Version des IBM WebSphere MQ-Clients spielt ebenfalls eine wichtige Rolle dabei, ob eine XMS-Clientanwendung IBM WebSphere MQ Version 7.0-spezifische Funktionen verwenden kann. In der folgenden Tabelle wird das Verhalten beschrieben.

Anmerkung: Eine Systemeigenschaft XMSC_WMQ_OVERRIDEPROVIDERVERSION überschreibt die Eigenschaft XMSC_WMQ_PROVIDER_VERSION. Diese Eigenschaft kann verwendet werden, wenn die Einstellung für die Verbindungsfactory nicht geändert werden kann.

Tabelle 37. XMS-Client - Möglichkeit der Verwendung IBM WebSphere MQ Version 7.0-spezifischer Funktionen			
#	**XMSC_WMQ_PROVIDER_VERSION**	**IBM WebSphere MQ Client-Version**	**IBM WebSphere MQ Version 7.0-Funktionen**
1	nicht angegeben	7	ON
2	nicht angegeben	6	OFF
3	7	7	ON
4	7	6	Ausnahmebedingung
5	6	6	OFF
6	6	7	OFF

XMSC_WMQ_PUB_ACK_INTERVAL

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Anzahl der Nachrichten, die von einem Publisher veröffentlicht werden, bevor der XMS-Client eine Bestätigung vom Broker anfordert.

Wenn Sie den Wert dieser Eigenschaft verringern, fordert der Client häufiger Bestätigungen an, sodass die Leistung des Publishers abnimmt. Wenn Sie den Wert erhöhen, benötigt der Client mehr Zeit, um eine Ausnahme auslösen, falls der Broker fehlschlägt.

Der Wert der Eigenschaft ist eine positive ganze Zahl. Der Standardwert ist 25.

XMSC_WMQ_QMGR_CCSID

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die ID des codierten Zeichensatzes (CCSID) oder die Codepage, in der Zeichendatenfelder, die in der MQI (Message Queue Interface) definiert sind, zwischen dem XMS -Client und dem IBM WebSphere MQ -Client ausgetauscht werden. Diese Eigenschaft gilt nicht für die Zeichendatenfolgen in den Hauptteilen von Nachrichten.

Wenn die Eine XMS-Anwendung eine Verbindung zu einem Warteschlangenmanager im Clientmodus herstellt, stellt der XMS-Client eine Verknüpfung zum IBM WebSphere MQ-Client her. Die Informationen, die zwischen den beiden Clients ausgetauscht werden, enthalten Zeichendatenfelder, die im MQI definiert sind. Unter normalen Umständen geht der IBM WebSphere MQ-Client davon aus, dass sich diese Felder in der Codepage des Systems befinden, auf dem die Clients ausgeführt werden. Wenn der XMS-Client diese Felder in einer anderen Codepage bereitstellt und empfängt, müssen Sie diese Eigenschaft festlegen, um den IBM WebSphere MQ-Client zu informieren.

Wenn der IBM WebSphere MQ-Client die Zeichendatenfelder an den Warteschlangenmanager weiterleitet, müssen die Daten in den Feldern gegebenenfalls in die vom Warteschlangenmanager verwendete Codepage konvertiert werden. Entsprechend gilt: Wenn der IBM WebSphere MQ-Client die Felder vom Warteschlangenmanager empfängt, müssen die Daten gegebenenfalls in die Codepage konvertiert werden, in der der XMS-Client die Daten beim Empfang erwartet. Der IBM WebSphere MQ-Client führt mithilfe dieser Eigenschaft diese Datenkonvertierungen durch.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Das Festlegen dieser Eigenschaft entspricht dem Setzen der Umgebungsvariable MQCCSID für einen IBM WebSphere MQ-Client, der native IBM WebSphere MQ-Clientanwendungen unterstützt. Weitere Informationen zu dieser Umgebungsvariable finden Sie im Abschnitt *IBM WebSphere MQ-Clients*.

XMSC_WMQ_QUEUE_MANAGER

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name des Warteschlangenmanagers, zu dem eine Verbindung hergestellt werden soll.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WMQ_RECEIVE_CCSID

Die Zieleigenschaft, die die Ziel-CCSID für die Nachrichtenkonvertierung des Warteschlangenmanagers festlegt. Der Wert wird ignoriert, außer wenn XMSC_WMQ_RECEIVE_CONVERSION auf WMQ_RECEIVE_CONVERSION_QMGR gesetzt ist.

Datentyp:
Integer

Wert:
Eine beliebige positive Ganzzahl.

Der Standardwert ist 1208.

XMSC_WMQ_RECEIVE_CONVERSION

Eine Zieleigenschaft, die bestimmt, ob eine Datenkonvertierung vom Warteschlangenmanager durchgeführt wird.

Datentyp:
Integer

Werte:
XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG (DEFAULT): Die Datenkonvertierung findet nur auf dem XMS-Client statt. Die Konvertierung erfolgt immer unter Verwendung der Codepage 1208.

XMSC_WMQ_RECEIVE_CONVERSION_QMGR: Die Datenkonvertierung erfolgt auf dem Warteschlangenmanager, bevor eine Nachricht an den XMS-Client gesendet wird.

XMSC_WMQ_RECEIVE_EXIT

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Gibt einen Kanalempfangsexit an, der ausgeführt werden soll.

Der Wert der Eigenschaft ist eine Zeichenfolge, die einen Kanalempfangsexit angibt und folgendes Format hat:

`libraryName``(entryPointName)`

Dabei gilt Folgendes:

- **`libraryName`** ist der vollständige Pfad des verwalteten Exits `.dll`
- `Eingangspunktname` ist der Klassenname, qualifiziert durch den Namensbereich.

Beispiel: `C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)`

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Modus des verwalteten Clients herstellt. Es werden auch nur verwaltete Exits unterstützt.

XMSC_WMQ_RECEIVE_EXIT_INIT

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Die Benutzerdaten, die beim Aufruf eines Kanalempfangsexits an diesen übergeben werden.

Der Wert der Eigenschaft ist eine Zeichenfolge. Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Modus des verwalteten Clients herstellt und die Eigenschaft „XMSC_WMQ_RECEIVE_EXIT" auf Seite 244 festgelegt ist.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Mit dieser Eigenschaft wird der Name des Warteschlangenmanagers abgerufen, zu dem eine Verbindung hergestellt ist.

Bei Verwendung mit einer CCDT (Definitionstabelle für Clientkanal) kann dieser Name von dem in der Verbindungsfactory angegebenen Warteschlangenmanagernamen abweichen.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Diese Eigenschaft wird mit der ID des Warteschlangenmanagers nach der Verbindung belegt.

XMSC_WMQ_SECURITY_EXIT

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Gibt einen Kanalsicherheitsexit an.

Der Wert der Eigenschaft ist eine Zeichenfolge, die einen Kanalsicherheitsexit angibt und folgendes Format hat:

libraryName(entryPointName)

Dabei gilt Folgendes:

- **libraryName** ist der vollständige Pfad des verwalteten Exits .dll
- Eingangspunktname ist der Klassenname, qualifiziert durch den Namensbereich.

Beispiel: C:\MySecurityExit.dll(MySecurityExitNameSpace.MySecurityExitClassName)

Die maximal zulässige Länge beträgt 128 Zeichen.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Modus des verwalteten Clients herstellt. Es werden auch nur verwaltete Exits unterstützt.

XMSC_WMQ_SECURITY_EXIT_INIT

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Die Benutzerdaten, die beim Aufruf des Kanalsicherheitsexits an diesen übergeben werden.

Die maximal zulässige Länge der Benutzerdatenzeichenfolge beträgt 32 Zeichen.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Modus des verwalteten Clients herstellt und die Eigenschaft „XMSC_WMQ_SECURITY_EXIT" auf Seite 245 festgelegt ist.

XMSC_WMQ_SEND_EXIT

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Gibt einen Kanalsendeexit an.

Der Wert der Eigenschaft ist eine Zeichenfolge. Ein Kanalsendeexit hat folgendes Format:

`libraryName``(entryPointName)`

Dabei gilt Folgendes:

- **`libraryName`** ist der vollständige Pfad des verwalteten Exits .dll
- `Eingangspunktname` ist der Klassenname, qualifiziert durch den Namensbereich.

Beispiel: `C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)`

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Modus des verwalteten Clients herstellt. Es werden auch nur verwaltete Exits unterstützt.

XMSC_WMQ_SEND_EXIT_INIT

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Die Benutzerdaten, die beim Aufruf von Kanalsendeexits an diese übergeben werden.

Der Wert der Eigenschaft ist eine Zeichenfolge, die aus einem oder mehreren Benutzerdatenelementen besteht, die durch Kommas getrennt sind. Die Eigenschaft ist standardmäßig nicht festgelegt.

Die Regeln für die Angabe von Benutzerdaten, die an eine Folge von Kanalsendeexits übergeben werden, sind dieselben wie die Regeln für die Angabe von Benutzerdaten, die an eine Folge von Kanalempfangsexits übergeben werden. Die Regeln dafür werden im Abschnitt „XMSC_WMQ_RECEIVE_EXIT_INIT" auf Seite 245 beschrieben.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Modus des verwalteten Clients herstellt und die Eigenschaft „XMSC_WMQ_SEND_EXIT" auf Seite 246 festgelegt ist.

XMSC_WMQ_SEND_CHECK_COUNT

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Anzahl Sendeaufrufe, die innerhalb einer einzelnen XMS-Sitzung ohne Transaktionsunterstützung zwischen Überprüfungen auf Fehler bei asynchronen Put-Operationen zugelassen werden sollen.

Diese Eigenschaft ist standardmäßig auf 0 gesetzt.

XMSC_WMQ_SHARE_CONV_ALLOWED

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Gibt an, ob eine Clientverbindung ihr Socket mit anderen XMS -Verbindungen der höchsten Ebene von demselben Prozess zu demselben Warteschlangenmanager gemeinsam nutzen kann, wenn die Kanaldefinitionen übereinstimmen. Diese Eigenschaft wird bereitgestellt, um eine vollständige Isolierung von Verbindungen in separaten Sockets zu ermöglichen, wenn dies für die Anwendungsentwicklung oder Wartung oder aus betriebsbedingten Gründen erforderlich ist. Die Festlegung dieser Eigenschaft ist lediglich ein Hinweis für XMS, dass das zugrunde liegende Socket gemeinsam genutzt werden soll. Es wird nicht angegeben, wie viele Verbindungen ein einzelnes Socket gemeinsam nutzen. Die Anzahl der Verbindungen, die ein Socket gemeinsam nutzen, wird durch den SHARECNV-Wert bestimmt, der zwischen IBM WebSphere MQ-Client und IBM WebSphere MQ-Server ausgehandelt wird.

Eine Anwendung kann folgende benannte Konstanten setzen, um die Eigenschaft festzulegen:

- XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE - Verbindungen nutzen ein Socket nicht gemeinsam.
- XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE - Verbindungen nutzen ein Socket gemeinsam.

Die Eigenschaft ist standardmäßig auf XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED gesetzt.

Diese Eigenschaft ist nur relevant, wenn eine Anwendung eine Verbindung zu einem Warteschlangenmanager im Clientmodus herstellt.

XMSC_WMQ_SSL_CERT_STORES

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Die Positionen der Server, auf denen sich die Zertifikatswiderrufslisten (CRLs) befinden, die für eine SSL-Verbindung zu einem Warteschlangenmanager verwendet werden sollen.

Der Wert der Eigenschaft ist eine Liste mit einer oder mehreren URLs, getrennt durch Kommas. Jede URL hat folgendes Format:

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

Dieses Format ist mit dem grundlegenden MQJMS-Format kompatibel, wurde aber erweitert.

Es ist zulässig, eine leere `serveraddress`zu haben. In diesem Fall verwendet XMS standardmäßig "localhost" als Serveradresse.

Hier eine Liste mit gültigen Beispielen:

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
```

```
ldap://
ldap://:389
```

Nur für .NET : Verwaltete Verbindungen zu IBM WebSphere MQ (WMQ_CM_CLIENT) unterstützen keine SSL-Verbindungen, können aber über eine nicht verwaltete Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt werden.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WMQ_SSL_CIPHER_SPEC

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

V 7.5.0.2 Der Name der CipherSpec, die für eine sichere Verbindung zu einem Warteschlangenmanager verwendet werden soll.

In der folgenden Tabelle sind die Verschlüsselungsspezifikationen aufgeführt, die Sie mit der SSL- und TLS-Unterstützung von IBM WebSphere MQ verwenden können. Wenn Sie ein persönliches Zertifikat anfordern, geben Sie eine Schlüsselgröße für das öffentliche und das private Schlüsselpaar an. Die Schlüsselgröße, die während des SSL-Handshakes verwendet wird, entspricht der im Zertifikat hinterlegten Größe, es sei denn, die CipherSpec bestimmt sie (siehe Tabelle). Diese Eigenschaft ist standardmäßig nicht festgelegt.

CipherSpec-Name	Verwendetes Protokoll	Hashalgorithmus	Verschlüsselungsalgorithmus	Verschlüsselungsbits	FIPS [1]	Suite B mit 128 Bit	Suite B mit 192 Bit
NULL_MD5	SSL 3.0 [8]	MD5	--	0	Nein	Nein	Nein
NULL_SHA	SSL 3.0 [8]	SHA-1	--	0	Nein	Nein	Nein
RC4_MD5_EXPORT [2]	SSL 3.0 [8]	MD5	RC4	40	Nein	Nein	Nein
RC4_MD5_US	SSL 3.0 [8]	MD5	RC4	128	Nein	Nein	Nein
RC4_SHA_US	SSL 3.0 [8]	SHA-1	RC4	128	Nein	Nein	Nein
RC2_MD5_EXPORT[2]	SSL 3.0 [8]	MD5	RC2	40	Nein	Nein	Nein
DES_SHA_EXPORT [2]	SSL 3.0 [8]	SHA-1	DES	56	Nein	Nein	Nein
RC4_56_SHA_EXPORT1024[3]	SSL 3.0 [8]	SHA-1	RC4	56	Nein	Nein	Nein
DES_SHA_EXPORT1024 [3]	SSL 3.0 [8]	SHA-1	DES	56	Nein	Nein	Nein
TRIPLE_DES_SHA_US	SSL 3.0 [8]	SHA-1	3DES	168	Nein	Nein	Nein

CipherSpec-Name	Verwendetes Protokoll	Hashalgorithmus	Verschlüsselungsalgorithmus	Verschlüsselungsbits	FIPS [1]	Suite B mit 128 Bit	Suite B mit 192 Bit
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES (Advanced Encryption Standard)	128	Ja	Nein	Nein
TLS_RSA_WITH_AES_256_CBC_SHA[4]	TLS 1.0	SHA-1	AES (Advanced Encryption Standard)	256	Ja	Nein	Nein
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	Nein[5]	Nein	Nein
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8]	TLS 1.0	SHA-1	3DES	168	Ja	Nein	Nein
FIPS_WITH_DES_CBC_SHA	SSL 3.0	SHA-1	DES	56	Nein[6]	Nein	Nein
FIPS_WITH_3DES_EDE_CBC_SHA	SSL 3.0	SHA-1	3DES	168	Nein[7]	Nein	Nein
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES (Advanced Encryption Standard)	128	Ja	Nein	Nein
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES (Advanced Encryption Standard)	256	Ja	Nein	Nein
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES (Advanced Encryption Standard)	128	Ja	Nein	Nein
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	AES (Advanced Encryption Standard)	256	Ja	Nein	Nein
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Nein	Nein	Nein
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Ja	Nein	Nein

CipherSpec-Name	Verwendetes Protokoll	Hashalgorithmus	Verschlüsselungsalgorithmus	Verschlüsselungsbits	FIPS [1]	Suite B mit 128 Bit	Suite B mit 192 Bit
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Nein	Nein	Nein
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Ja	Nein	Nein
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES (Advanced Encryption Standard)	128	Ja	Nein	Nein
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES (Advanced Encryption Standard)	256	Ja	Nein	Nein
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES (Advanced Encryption Standard)	128	Ja	Nein	Nein
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES (Advanced Encryption Standard)	256	Ja	Nein	Nein
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES (Advanced Encryption Standard)	128	Ja	Ja	Nein
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES (Advanced Encryption Standard)	256	Ja	Nein	Ja
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES (Advanced Encryption Standard)	128	Ja	Nein	Nein

CipherSpec-Name	Verwendetes Protokoll	Hashalgorithmus	Verschlüsselungsalgorithmus	Verschlüsselungsbits	FIPS [1]	Suite B mit 128 Bit	Suite B mit 192 Bit
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES (Advanced Encryption Standard)	256	Ja	Nein	Nein
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	--	0	Nein	Nein	Nein
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	--	0	Nein	Nein	Nein
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	--	0	Nein	Nein	Nein
TLS_RSA_WITH_NULL_NULL	TLS 1.2	--	--	0	Nein	Nein	Nein
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Nein	Nein	Nein

Anmerkungen:

1. Gibt an, ob die CipherSpec den Federal Information Processing Standards (FIPS) 140-2 entspricht. Eine Erläuterung von FIPS und Informationen zur Konfiguration von WebSphere MQ für einen FIPS 140-2-konformen Betrieb finden Sie unter *Federal Information Processing Standards (FIPS)* in der Online-Produktdokumentation zu IBM IBM WebSphere MQ .
2. Die maximale Größe des Handshakeschlüssels beträgt 512 Bit. Hat eines der beim SSL-Handshake ausgetauschten Zertifikate einen Schlüssel mit mehr als 512 Bits, wird ein temporärer 512-Bit-Schlüssel zur Verwendung während des Handshakes generiert.
3. Die Größe des Handshakeschlüssels beträgt 1024 Bit.
4. Mithilfe dieser Verschlüsselungsspezifikation (CipherSpec) kann eine Verbindung von WebSphere MQ Explorer zu einem Warteschlangenmanager nicht geschützt werden, es sei denn, für die vom Explorer verwendete JRE gelten die entsprechenden uneingeschränkten Richtliniendateien.
5. Diese CipherSpec wurde vor dem 19. Mai 2007 FIPS 140-2-zertifiziert.
6. Diese CipherSpec wurde vor dem 19. Mai 2007 FIPS 140-2-zertifiziert. Der Name FIPS_WITH_DES_CBC_SHA ist historisch und spiegelt die Tatsache wider, dass diese CipherSpec zuvor FIPS-konform war (aber jetzt nicht mehr). Diese CipherSpec wird nicht mehr unterstützt.
7. Der Name FIPS_WITH_3DES_EDE_CBC_SHA ist historisch und spiegelt die Tatsache wider, dass diese CipherSpec zuvor FIPS-konform war (aber jetzt nicht mehr). Die Verwendung dieser CipherSpec wird nicht weiter unterstützt.
8. Wenn WebSphere MQ für den FIPS 140-2-konformen Betrieb konfiguriert wurde, kann diese CipherSpec verwendet werden, um bis zu 32 Gigabyte Daten zu übertragen, bevor die Verbindung mit dem Fehler AMQ9288 beendet wird. Um diesen Fehler zu vermeiden, sollten Sie entweder Triple DES nicht verwenden (da es veraltet ist) oder die Zurücksetzung von geheimen Schlüsseln aktivieren, wenn diese CipherSpec in einer FIPS 140-2-Konfiguration verwendet wird.

Zugehörige Konzepte

Sicherheit

Datenintegrität von Nachrichten

Zugehörige Tasks
CipherSpecs angeben

XMSC_WMQ_SSL_CIPHER_SUITE

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name der CipherSuite , die für eine SSL-oder TLS-Verbindung zu einem Warteschlangenmanager verwendet werden soll. Welches Protokoll bei der Vereinbarung der sicheren Verbindung verwendet wird, ist von der angegebenen Cipher-Suite abhängig.

Diese Eigenschaft hat folgende kanonischen Werte:

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
- SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA

Dieser Wert kann als Alternative zu XMSC_WMQ_SSL_CIPHER_SPEC angegeben werden.

Wenn für XMSC_WMQ_SSL_CIPHER_SPEC ein nicht leerer Wert angegeben wird, überschreibt dieser Wert die Einstellung für XMSC_WMQ_SSL_CIPHER_SUITE. Wenn XMSC_WMQ_SSL_CIPHER_SPEC keinen Wert aufweist, wird der Wert von XMSC_WMQ_SSL_CIPHER_SUITE als Cipher-Suite verwendet, die an GSKit übergeben werden soll. In diesem Fall wird der Wert dem entsprechenden CipherSpec-Wert zugeordnet, wie im Abschnitt „Namenszuordnung zwischen Cipher-Suites und CipherSpecs für Verbindungen zu einem IBM WebSphere MQQueue Manager” auf Seite 71 beschrieben ist.

Wenn sowohl XMSC_WMQ_SSL_CIPHER_SPEC als auch XMSC_WMQ_SSL_CIPHER_SUITE leer sind, wird das Feld pChDef->SSLCipherSpec mit Leerzeichen gefüllt.

Nur für .NET : Verwaltete Verbindungen zu IBM WebSphere MQ (WMQ_CM_CLIENT) unterstützen keine SSL-Verbindungen, können jedoch über eine nicht verwaltete Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt werden.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WMQ_SSL_CRYPTO_HW

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Konfigurationsdetails für die Verschlüsselungshardware, die mit dem Clientsystem verbunden ist.

Diese Eigenschaft hat folgende kanonischen Werte:

- GSK_ACCELERATOR_RAINBOW_CS_OFF
- GSK_ACCELERATOR_RAINBOW_CS_ON
- GSK_ACCELERATOR_NCIPHER_NF_OFF
- GSK_ACCELERATOR_NCIPHER_NF_ON

Es gibt ein spezielles Format für PKCS11-Verschlüsselungshardware (wobei Treiberpfad, Tokenbezeichnung und Tokenkennwort benutzerdefinierte Zeichenfolgen sind):

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS interpretiert den Inhalt der Zeichenfolge nicht und ändert ihn auch nicht. Es kopiert den übergebenen Wert (maximal 256 Einzelbytezeichen) in das Feld MQSCO.CryptoHardware.

Nur für .NET : Verwaltete Verbindungen zu IBM WebSphere MQ (WMQ_CM_CLIENT) unterstützen keine SSL-Verbindungen, können aber über eine nicht verwaltete Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt werden.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WMQ_SSL_FIPS_REQUIRED

Datentyp:
Boolesch

Eigenschaft von:
ConnectionFactory

Der Wert dieser Eigenschaft legt fest, ob eine Anwendung nicht FIPS-konforme Cipher-Suites verwenden kann oder nicht. Wenn diese Eigenschaft auf 'true' gesetzt ist, werden nur FIPS-Algorithmen für die Client-Server-Verbindung verwendet.

Diese Eigenschaft kann folgende Werte haben, die in die zwei kanonischen Werte für MQSCO.FipsRequired umgesetzt werden:

Tabelle 38. Tabelle der Werte für Eigenschaft MQSCO.FipsRequired

Wert	Beschreibung	Entsprechender Wert von MQSCO.FipsRequired
false	Es kann eine beliebige CipherSpec kann verwendet werden.	MQSSL_FIPS_NO (der Standardwert)
true	In der CipherSpec für diese Clientverbindung können nur FIPS-zertifizierte Verschlüsselungsalgorithmen verwendet werden.	MQSSL_FIPS_YES

XMS kopiert den relevanten Wert in MQSCO.FipsRequired, bevor MQCONNX aufgerufen wird.

Der Parameter MQSCO.FipsRequired ist nur ab IBM WebSphere MQ Version 6 verfügbar. Wenn IBM WebSphere MQ Version 5.3, wenn diese Eigenschaft gesetzt ist, versucht XMS nicht, die Verbindung zum Warteschlangenmanager herzustellen, und löst stattdessen eine entsprechende Ausnahme aus.

Nur für .NET : Verwaltete Verbindungen zu IBM WebSphere MQ (WMQ_CM_CLIENT) unterstützen keine SSL-Verbindungen, können aber über eine nicht verwaltete Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt werden.

XMSC_WMQ_SSL_KEY_REPOSITORY

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Gibt die Position der Schlüsseldatenbankdatei an, in der Schlüssel und Zertifikate gespeichert werden.

XMS kopiert die Zeichenfolge, die bis zu 256 Einzelbytezeichen lang sein kann, in das Feld MQSCO.KeyRepository. IBM WebSphere MQ interpretiert die Zeichenfolge als Dateiname, einschließlich des vollständigen Pfads.

Nur für .NET : Verwaltete Verbindungen zu IBM WebSphere MQ (WMQ_CM_CLIENT) unterstützen keine SSL-Verbindungen, können aber über eine nicht verwaltete Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt werden.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WMQ_SSL_KEY_RESETCOUNT

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Der Wert von KeyResetCount gibt die Gesamtzahl unverschlüsselter Bytes an, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL-Dialog gesendet oder empfangen werden. Die Anzahl der Byte enthält Steuerinformationen, die vom MCA gesendet wurden.

XMS kopiert den Wert, den Sie für diese Eigenschaft angeben, in den Parameter MQSCO.KeyResetCount, bevor MQCONNX aufgerufen wird.

Der Parameter MQSCO.KeyRestCount ist nur in der Version 6 von IBM WebSphere MQ verfügbar. Wenn IBM WebSphere MQ Version 5.3, wenn diese Eigenschaft gesetzt ist, versucht XMS nicht, die Verbindung zum Warteschlangenmanager herzustellen, und löst stattdessen eine entsprechende Ausnahme aus.

Nur für .NET : Verwaltete Verbindungen zu IBM WebSphere MQ (WMQ_CM_CLIENT) unterstützen keine SSL-Verbindungen, können aber über eine nicht verwaltete Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt werden.

Der Standardwert für diese Eigenschaft ist null. Dies bedeutet, dass geheime Schlüssel nie neu verhandelt werden.

XMSC_WMQ_SSL_PEER_NAME

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Peername, der für eine SSL-Verbindung zu einem Warteschlangenmanager verwendet werden soll.

Es gibt keine Liste mit kanonischen Werte für diese Eigenschaft. Stattdessen müssen Sie diese Zeichenfolge gemäß den Regeln für SSLPEER erstellen.

Hier ein Beispiel für einen Peernamen:

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS kopiert die Zeichenfolge in die richtige Einzelbyte-Codepage und legt für MQCD.SSLPeerNamePtr und MQCD.SSLPeerNameLength die richtigen Werte fest, bevor MQCONNX aufgerufen wird.

Diese Eigenschaft ist nur relevant, wenn die Anwendung eine Verbindung zu einem Warteschlangenmanager im Clientmodus herstellt.

Nur für .NET : Verwaltete Verbindungen zu IBM WebSphere MQ (WMQ_CM_CLIENT) unterstützen keine SSL-Verbindungen, können aber über eine nicht verwaltete Verbindung (WMQ_CM_CLIENT_UNMANAGED) unterstützt werden.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WMQ_SYNCPOINT_ALL_GETS

Datentyp:
System.Boolean

Eigenschaft von:
ConnectionFactory

Gibt an, ob alle Nachrichten innerhalb der Synchronisationspunktsteuerung aus Warteschlangen abgerufen werden müssen.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
false	Wenn die Umstände geeignet sind, kann der XMS-Client Nachrichten aus Warteschlangen außerhalb der Synchronisationspunktsteuerung abrufen.
true	Der XMS-Client muss alle Nachrichten aus Warteschlangen innerhalb der Synchronisationspunktsteuerung abrufen.

Der Standardwert ist 'false'.

XMSC_WMQ_TARGET_CLIENT

Datentyp:
System.Int32

Eigenschaft von:
Ziel

In einem URI verwendeter Name:
targetClient

Gibt an, ob Nachrichten, die an das Ziel gesendet werden, einen MQRFH2-Header enthalten.

Wenn eine Anwendung eine Nachricht sendet, die einen MQRFH2-Header enthält, muss die empfangende Anwendung den Header verarbeiten können.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_WMQ_TARGET_DEST_JMS	Nachrichten, die an das Ziel gesendet werden, enthalten einen MQRFH2-Header. Geben Sie diesen Wert an, wenn die Anwendung die Nachrichten an eine andere XMS-Anwendung, eine WebSphere JMS-Anwendung oder eine native IBM WebSphere MQ-Anwendung sendet, die für die Verarbeitung eines MQRFH2-Headers konzipiert ist.
XMSC_WMQ_TARGET_DEST_MQ	Nachrichten, die an das Ziel gesendet werden, enthalten keinen MQRFH2-Header. Geben Sie diesen Wert an, wenn die Anwendung die Nachrichten an eine native IBM WebSphere MQ-Anwendung sendet, die nicht für die Verarbeitung eines MQRFH2-Headers konzipiert ist.

Der Standardwert ist XMSC_WMQ_TARGET_DEST_JMS.

XMSC_WMQ_TEMP_Q_PREFIX

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Das Präfix, das verwendet wird, um den Namen der dynamischen IBM WebSphere MQ -Warteschlange zu bilden, die erstellt wird, wenn die Anwendung eine temporäre Eine XMS -Warteschlange erstellt.

Die Regeln für die Bildung des Präfix sind mit den Regeln für die Bildung des Inhalts des Felds **DynamicQName** in einem Objektdeskriptor identisch, aber das letzte nicht leere Zeichen muss ein Stern (*) sein. Wenn die Eigenschaft nicht festgelegt ist, wird der Wert `CSQ.*` unter z/OS und `AMQ.*` auf den anderen Plattformen verwendet. Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur in der Punkt-zu-Punkt (Point-to-point)-Domäne relevant.

XMSC_WMQ_TEMP_TOPIC_PREFIX

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory, Destination

Beim Erstellen temporärer Themen generiert XMS eine Themenzeichenfolge im Format "`TEMP/TEMPTOPICPREFIX/unique_id`" oder, wenn diese Eigenschaft den Standardwert hat, nur "`TEMP/unique_id`". Die Angabe eines nichtleeren Werts ermöglicht die Definition bestimmter Modellwarteschlangen für die Erstellung der verwalteten Warteschlangen für Abonnenten temporärer Themen, die unter dieser Verbindung erstellt werden.

Jede Zeichenfolge ungleich null, die ausschließlich aus gültigen Zeichen für eine IBM WebSphere MQ-Themenzeichenfolge besteht, ist ein gültiger Wert für diese Eigenschaft.

Diese Eigenschaft ist standardmäßig auf "" (leere Zeichenfolge) gesetzt.

Anmerkung: Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WMQ_TEMPORARY_MODEL

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name der IBM WebSphere MQ -Modellwarteschlange, aus der eine dynamische Warteschlange erstellt wird, wenn die Anwendung eine temporäre Eine XMS -Warteschlange erstellt.

Der Standardwert der Eigenschaft ist SYSTEM.DEFAULT.MODEL.QUEUE.

Diese Eigenschaft ist nur in der Punkt-zu-Punkt (Point-to-point)-Domäne relevant.

XMSC_WMQ_WILDCARD_FORMAT

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory, Destination

Diese Eigenschaft gibt an, welche Version der Platzhalterzeichensyntax verwendet werden soll.

Bei Verwendung von Publish/Subscribe mit IBM WebSphere MQ '*' und '?' werden als Platzhalterzeichen behandelt. Dagegen werden '#' und '+' als Platzhalterzeichen behandelt, wenn Publish/Subscribe mit WebSphere Nachrichtenbroker verwendet wird. Diese Eigenschaft ersetzt die Eigenschaft XMSC_WMQ_BROKER_VERSION.

Die gültigen Werte für diese Eigenschaft sind:

XMSC_WMQ_WILDCARD_TOPIC_ONLY
Erkennt nur Platzhalter auf Themenebene, d. h. '#' und '+' werden als Platzhalterzeichen behandelt. Dieser Wert ist identisch mit XMSC_WMQ_BROKER_V2.

XMSC_WMQ_WILDCARD_CHAR_ONLY
Erkennt nur die Zeichenplatzhalter, d. h. '*' und '?' werden als Platzhalterzeichen behandelt. Dieser Wert ist identisch mit XMSC_WMQ_BROKER_V1.

Diese Eigenschaft ist standardmäßig auf XMSC_WMQ_WILDCARD_TOPIC_ONLY gesetzt.

Anmerkung: Diese Eigenschaft ist nicht relevant, wenn Publish/Subscribe mit IBM WebSphere MQ Version 6.0 und niedriger ausgeführt wird. Stattdessen muss die Eigenschaft XMSC_WMQ_BROKER_VERSION verwendet werden.

XMSC_WPM_BUS_NAME

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory und Destination

In einem URI verwendeter Name:
busName

Für eine Verbindungsfactory der Name des Service Integration Bus, zu dem die Anwendung eine Verbindung herstellt, bzw. für ein Ziel der Name des Service Integration Bus, in dem sich das Ziel befindet.

Wenn das Ziel ein Thema ist, gibt diese Eigenschaft den Namen des Service Integration Bus an, in dem sich der zugehörige Themenbereich befindet. Dieser Themenbereich wird durch die Eigenschaft XMSC_WPM_TOPIC_SPACE angegeben.

Wenn die Eigenschaft für ein Ziel nicht festgelegt ist, wird angenommen, dass sich die Warteschlange oder der zugehörige Themenbereich in dem Service Integration Bus befindet, zu dem die Anwendung eine Verbindung herstellt.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WPM_CONNECTION_PROTOCOL

Datentyp:
System.Int32

Eigenschaft von:
Verbindung

Das Kommunikationsprotokoll, das für die Verbindung zur Messaging-Engine verwendet wird. Diese Eigenschaft ist schreibgeschützt.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Wert	Bedeutet
XMSC_WPM_CP_HTTP	Die Verbindung verwendet HTTP über TCP/IP.
XMSC_WPM_CP_TCP	Die Verbindung verwendet TCP/IP.

XMSC_WPM_CONNECTION_PROXIMITY

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Verbindungsabstandseinstellung für die Verbindung. Diese Eigenschaft bestimmt, wie nahe die Messaging-Engine, zu der die Anwendung eine Verbindung herstellt, dem Bootstrap-Server sein muss.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Verbindungsabstands-einstellung
XMSC_WPM_CONNECTION_PROXIMITY_BUS	Bus
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	Cluster
XMSC_WPM_CONNECTION_PROXIMITY_HOST	Host
XMSC_WPM_CONNECTION_PROXIMITY_SERVER	Server

Der Standardwert ist XMSC_WPM_CONNECTION_PROXIMITY_BUS.

XMSC_WPM_DUR_SUB_HOME

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

In einem URI verwendeter Name:
durableSubscriptionHome

Der Name der Messaging-Engine, auf der alle permanenten Subskriptionen für eine Verbindung oder ein Ziel verwaltet werden. Nachrichten, die an die permanenten Subskribenten zugestellt werden sollen, werden am Veröffentlichungspunkt der betreffenden Messaging-Engine gespeichert.

Die Ausgangsposition für permanente Subskriptionen muss für eine Verbindung angegeben werden, bevor eine Anwendung einen permanenten Subskribenten erstellen kann, der diese Verbindung verwendet. Ein Wert, der für ein Ziel angegeben wird, überschreibt in jedem Fall den für eine Verbindung angegebenen Wert.

Die Eigenschaft ist standardmäßig nicht festgelegt.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WPM_HOST_NAME

Datentyp:
Zeichenfolge

Eigenschaft von:
Verbindung

Der Hostname oder die IP-Adresse des Systems, das die Messaging-Engine enthält, mit der die Anwendung verbunden ist. Diese Eigenschaft ist schreibgeschützt.

XMSC_WPM_LOCAL_ADDRESS

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Für eine Verbindung zu einem Service Integration Bus gibt diese Eigenschaft die zu verwendende lokale Netzschnittstelle und/oder den zu verwendenden lokalen Port bzw. den Bereich der zu verwendenden lokalen Ports an.

Der Wert der Eigenschaft ist eine Zeichenfolge in folgenden Format:

[*Hostname*][(*niedrigster_Port*)[,*höchster_Port*])]

Die Variablen haben folgende Bedeutung:

Hostname
Der Hostname oder die IP-Adresse der lokalen Netzschnittstelle, die für die Verbindung verwendet werden soll.

Die Angabe dieser Eigenschaft ist nur dann notwendig, wenn das System, auf dem die Anwendung aktiv ist, über zwei oder mehr Netzschnittstellen verfügt und Sie in der Lage sein müssen, die Schnittstelle anzugeben, die für die Verbindung verwendet werden muss. Falls das System nur über eine Netzschnittstelle verfügt, kann nur diese Schnittstelle verwendet werden. Wenn das System über zwei oder mehr Netzschnittstellen verfügt und Sie nicht angeben, welche Schnittstelle verwendet werden soll, wird die Schnittstelle nach dem Zufallsprinzip ausgewählt.

niedrigster_Port
Die Nummer des lokalen Ports, der für die Verbindung verwendet werden soll.

Wenn *höchster_Port* ebenfalls angegeben ist, wird *niedrigster_Port* als die niedrigste Portnummer in einem Bereich von Portnummern interpretiert.

höchster_Port
Die höchste Portnummer in einem Bereich von Portnummern. Einer der Ports im angegebenen Bereich muss für die Verbindung verwendet werden.

Hier einige Beispiele für gültige Werte der Eigenschaft:

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WPM_ME_NAME

Datentyp:
Zeichenfolge

Eigenschaft von:
Verbindung

Der Name der Messaging-Engine, mit der die Anwendung verbunden ist. Diese Eigenschaft ist schreibgeschützt.

XMSC_WPM_NON_PERSISTENT_MAP

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Zuverlässigkeitsstufe von nicht persistenten Nachrichten, die über die Verbindung gesendet werden.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Zuverlässigkeitsstufe
XMSC_WPM_MAPPING_AS_DESTINATION	Wird durch die standardmäßige Zuverlässigkeitsstufe bestimmt, die für die Warteschlange oder den Themenbereich im Service Integration Bus angegeben ist.

Gültiger Wert	Zuverlässigkeitsstufe
XMSC_WPM_MAPPING_BEST_EFFORT_NON_ PERSISTENT	Bestmöglich, nicht persistent
XMSC_WPM_MAPPING_EXPRESS_NON_ PERSISTENT	Express, nicht persistent
XMSC_WPM_MAPPING_RELIABLE_NON_ PERSISTENT	Zuverlässig, nicht persistent
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Zuverlässig, persistent
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Garantiert, persistent

Der Standardwert ist XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT.

XMSC_WPM_PERSISTENT_MAP

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Zuverlässigkeitsstufe von persistenten Nachrichten, die über die Verbindung gesendet werden.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Zuverlässigkeitsstufe
XMSC_WPM_MAPPING_AS_DESTINATION	Wird durch die standardmäßige Zuverlässigkeitsstufe bestimmt, die für die Warteschlange oder den Themenbereich im Service Integration Bus angegeben ist.
XMSC_WPM_MAPPING_BEST_EFFORT_NON_ PERSISTENT	Bestmöglich, nicht persistent
XMSC_WPM_MAPPING_EXPRESS_NON_ PERSISTENT	Express, nicht persistent
XMSC_WPM_MAPPING_RELIABLE_NON_ PERSISTENT	Zuverlässig, nicht persistent
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Zuverlässig, persistent
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Garantiert, persistent

Der Standardwert ist XMSC_WPM_MAPPING_RELIABLE_PERSISTENT.

XMSC_WPM_PORT

Datentyp:
System.Int32

Eigenschaft von:
Verbindung

Die Nummer des Ports, an dem die Messaging-Engine, mit der die Anwendung verbunden ist, empfangsbereit ist. Diese Eigenschaft ist schreibgeschützt.

XMSC_WPM_PROVIDER_ENDPOINTS

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Eine Folge aus einer oder mehreren Endpunktadressen von Bootstrap-Servern. Die Endpunktadressen werden durch Kommas getrennt.

Ein Bootstrap-Server ist ein Anwendungsserver, der dafür verantwortlich ist, die Messaging-Engine auszuwählen, zu der die Anwendung eine Verbindung herstellt. Die Endpunktadresse eines Bootstrap-Servers hat folgendes Format:

Hostname:*Portnummer*:*Kettenname*

Die Komponenten einer Endpunktadresse habe folgende Bedeutung:

Hostname
Der Hostname oder die IP-Adresse des Systems, auf dem sich der Bootstrap-Server befindet. Wird kein Hostname oder keine IP-Adresse angegeben, lautet die Standardeinstellung `localhost`.

Portnummer
Die Nummer des Ports, an dem der Bootstrap-Server für eingehende Anforderungen empfangsbereit ist. Wird keine Portnummer angegeben, wird der Standardwert 7276 verwendet.

Kettenname
Der Name einer Bootstrap-Transportkette, die vom Bootstrap-Server verwendet wird. Die gültigen Werte lauten wie folgt:

Gültiger Wert	Name der Bootstrap-Transportkette
XMSC_WPM_BOOTSTRAP_HTTP	BootstrapTunneledMessaging
XMSC_WPM_BOOTSTRAP_HTTPS	BootstrapTunneledSecureMessaging
XMSC_WPM_BOOTSTRAP_SSL	BootstrapSecureMessaging
XMSC_WPM_BOOTSTRAP_TCP	BootstrapBasicMessaging

Wenn kein Name angegeben ist, lautet der Standardwert: XMSC_WPM_BOOTSTRAP_TCP.

Wird keine Endpunktadresse angegeben, lautet der Standardwert localhost:7276:BootstrapBasicMessaging.

XMSC_WPM_TARGET_GROUP

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name einer Zielgruppe aus Messaging-Engines. Die Art der Zielgruppe wird durch die Eigenschaft XMSC_WPM_TARGET_TYPE bestimmt.

Legen Sie diese Eigenschaft fest, wenn Sie die Suche nach einer Messaging-Engine auf eine Untergruppe der Messaging-Engines im Service Integration Bus beschränken möchten. Wenn Ihre Anwendung in der Lage sein soll, eine Verbindung zu einer beliebigen Messaging-Engine im Service Integration Bus herzustellen, legen Sie diese Eigenschaft nicht fest.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WPM_TARGET_SIGNIFICANCE

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Die Signifikanz der Zielgruppe von Messaging-Engines.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_WPM_TARGET_SIGNIFICANCE_ PREFERRED	Es wird eine Messaging-Engine in der Zielgruppe ausgewählt, falls eine verfügbar ist. Andernfalls wird eine Messaging-Engine außerhalb der Zielgruppe ausgewählt, vorausgesetzt, sie befindet sich im selben Service Integration Bus.
XMSC_WPM_TARGET_SIGNIFICANCE_ ERFORDERLICH	Die ausgewählte Messaging-Engine muss sich in der Zielgruppe befinden. Wenn in der Zielgruppe keine Messaging-Engine verfügbar ist, schlägt der Verbindungsprozess fehl.

Der Standardwert der Eigenschaft ist XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED.

XMSC_WPM_TARGET_TRANSPORT_CHAIN

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Der Name der eingehenden Transportkette, die die Anwendung für die Verbindung zu einer Messaging-Engine verwenden muss.

Der Wert der Eigenschaft kann der Name einer eingehenden Transportkette sein, die auf dem Anwendungsserver verfügbar ist, der die Messaging-Engine hostet. Folgende benannte Konstante wird für eine der vordefinierten eingehenden Transportketten bereitgestellt:

Benannte Konstante	Name der Transportkette
XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC	InboundBasicMessaging

Der Standardwert der Eigenschaft ist XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC.

XMSC_WPM_TARGET_TYPE

Datentyp:
System.Int32

Eigenschaft von:
ConnectionFactory

Der Typ der Zielgruppe von Messaging-Engines. Diese Eigenschaft bestimmt die Art der Zielgruppe, die durch die Eigenschaft XMSC_WPM_TARGET_GROUP angegeben wird.

Die gültigen Werte für die Eigenschaft lauten wie folgt:

Gültiger Wert	Bedeutet
XMSC_WPM_TARGET_TYPE_BUSMEMBER	Der Name der Zielgruppe ist der Name eines Busmembers. Die Zielgruppe sind alle Messaging-Engines im Busmember.

Gültiger Wert	Bedeutet
XMSC_WPM_TARGET_TYPE_CUSTOM	Der Name der Zielgruppe ist der Name einer benutzerdefinierten Gruppe von Messaging-Engines. Die Zielgruppe sind alle Messaging-Engines, die in der benutzerdefinierten Gruppe registriert sind.
XMSC_WPM_TARGET_TYPE_ME	Der Name der Zielgruppe ist der Name einer Messaging-Engine. Die Zielgruppe ist die angegebene Messaging-Engine.

Die Eigenschaft ist standardmäßig nicht festgelegt.

XMSC_WPM_TEMP_Q_PREFIX

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Das Präfix, mit dem der Name der temporären Warteschlange gebildet wird, die im Service Integration Bus erstellt wird, wenn die Anwendung die temporäre Eine XMS-Warteschlange erstellt. Das Präfix kann bis zu 12 Zeichen enthalten.

Der Name einer temporären Warteschlange beginnt mit den Zeichen "_Q", gefolgt von dem Präfix. Der Rest des Namens besteht aus systemgenerierten Zeichen.

Die Eigenschaft ist standardmäßig nicht festgelegt, d. h., der Name einer temporären Warteschlange hat kein Präfix.

Diese Eigenschaft ist nur in der Punkt-zu-Punkt (Point-to-point)-Domäne relevant.

XMSC_WPM_TEMP_TOPIC_PREFIX

Datentyp:
Zeichenfolge

Eigenschaft von:
ConnectionFactory

Das Präfix, das verwendet wird, um den Namen eines temporären Themas zu bilden, das von der Anwendung erstellt wird. Das Präfix kann bis zu 12 Zeichen enthalten.

Der Name eines temporären Themas beginnt mit den Zeichen "_T", gefolgt von dem Präfix. Der Rest des Namens besteht aus systemgenerierten Zeichen.

Die Eigenschaft ist standardmäßig nicht festgelegt, d. h., der Name eines temporären Themas hat kein Präfix.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

XMSC_WPM_TOPIC_SPACE

Datentyp:
Zeichenfolge

Eigenschaft von:
Ziel

In einem URI verwendeter Name:
topicSpace

Der Name des Themenbereichs, der das Thema enthält. Nur ein Ziel, das ein Thema ist, kann diese Eigenschaft haben.

Die Eigenschaft ist standardmäßig nicht festgelegt, d. h., es wird der Standardthemenbereich angenommen.

Diese Eigenschaft ist nur in der Publish/Subscribe-Domäne relevant.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder andere Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Dokumentation ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Europe
IBM Europe, Middle East and Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Défense
U.S.A.

Bei Lizenzanforderungen zu Double-Byte-Information (DBCS) wenden Sie sich bitte an die IBM Abteilung für geistiges Eigentum in Ihrem Land oder senden Sie Anfragen schriftlich an folgende Adresse:

Lizenzierung von geistigem Eigentum

IBM Japan, Ltd.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in dieser Veröffentlichung werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Europe, Middle East and Africa
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesen Informationen beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Die in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Um diese so realistisch wie möglich zu gestalten, enthalten sie auch Namen von Personen, Firmen, Marken und Produkten. Sämtliche dieser Namen sind fiktiv. Ähnlichkeiten mit Namen und Adressen tatsächlicher Unternehmen oder Personen sind zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musterprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Musterprogramme kostenlos (d. h. ohne Zahlung an IBM) kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Informationen zu Programmierschnittstellen

Die bereitgestellten Informationen zur Programmierschnittstelle sollen Sie bei der Erstellung von Anwendungssoftware für dieses Programm unterstützen.

Dieses Handbuch enthält Informationen zu geplanten Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zum Abrufen der Services von IBM WebSphere MQzu schreiben.

Diese Informationen können jedoch auch Angaben über Diagnose, Bearbeitung und Optimierung enthalten. Die Informationen zu Diagnose, Bearbeitung und Optimierung sollten Ihnen bei der Fehlerbehebung für die Anwendungssoftware helfen.

Wichtig: Verwenden Sie diese Diagnose-, Änderungs-und Optimierungsinformationen nicht als Programmierschnittstelle, da sie Änderungen unterliegen.

Marken

IBM, das IBM Logo, ibm.com, sind Marken der IBM Corporation in den USA und/oder anderen Ländern. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Microsoft und Windows sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Linux® ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Dieses Produkt enthält Software, die von Eclipse Project (http://www.eclipse.org/) entwickelt wurde.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Teilenummer:

(1P) P/N: