9.0

IBM Message Service Client for .NET

注

在使用本资料及其支持的产品之前，请阅读第 219 页的『声明』中的信息。

内容

Message Service Client for .NET 5
IBM Message Service Client for .NET 简介 5
消息传递类型 6
XMS 对象模型 7
对象的属性和特性 8
受管对象 8
XMS 消息模型 9
防止应用程序使用较新的 XMS 版本 11
设置消息传递服务器环境 11
为连接到 IBM MQ 队列管理器的应用程序配置队列管理器和代理程序 12
为使用与代理程序的实时连接的应用程序配置代理程序 13
为连接到 WebSphere Application Server 的应用程序配置服务集成总线 14
使用 XMS 样本应用程序 15
样本应用程序 15
运行样本应用程序 16
构建 .NET 样本应用程序 17
开发 XMS 应用程序 17
编写 XMS 应用程序 17
编写 XMS .NET 应用程序 36
使用受管对象 42
确保 XMS 应用程序安全通信 55
XMS 消息 58
故障诊断 69
.NET 应用程序的跟踪配置 69
针对 .NET 应用程序的 FFDC 配置 72
故障诊断技巧 73
Message Service Clients for .NET 参考 74
.NET 接口 74
XMS 对象的属性 159

声明 219
编程接口信息 220
商标 220

IBM Message Service Client for .NET 简介

IBM Message Service Client for .NET 提供一个应用程序编程接口 (API)（称为 XMS），它具有与 Java Message Service (JMS) API 相同的接口集。 IBM Message Service Client for .NET 包含一个完全受管的 XMS 实现，此实现可由任何与 .NET 兼容的语言使用。

XMS 支持：

- 点到点 消息传递
- 发布/预订 消息传递
- 同步消息传递
- 异步消息传递

XMS 应用程序可以与以下类型的应用程序交换消息：

- XMS 应用程序
- IBM MQ classes for JMS 应用程序
- 本机 IBM MQ 应用程序
- 使用 IBM MQ 缺省消息传递提供程序的 JMS 应用程序

XMS 应用程序可以连接到以下任何消息传递服务器并使用其资源：

IBM MQ 队列管理器
应用程序可以在绑定或客户机方式下连接。

WebSphere Application Server 服务集成总线
应用程序可以使用直接 TCP/IP 连接，也可以使用“基于 TCP/IP 的 HTTP”。

IBM Integration Bus
将使用 WebSphere MQ 实时传输 在应用程序和代理程序之间传输消息。 可使用 WebSphere MQ Multicast Transport 将消息传递到应用程序。

通过连接到 IBM MQ 队列管理器，XMS 应用程序可以使用 WebSphere MQ 企业传输 来与 IBM Integration Bus 进行通信。 或者，XMS 应用程序可以通过连接到 IBM MQ 来进行发布和预订。

相关概念
第 6 页的『消息传递类型』
第 7 页的『XMS 对象模型』
XMS API 是一个面向对象的接口。 XMS 对象模型基于 JMS 1.1 对象模型。

第 9 页的『XMS 消息模型』
XMS 消息模型与 IBM MQ classes for JMS 消息模型相同。

IBM Message Service Client for .NET 简介

IBM Message Service Client for .NET 提供一个应用程序编程接口 (API)（称为 XMS），它具有与 Java Message Service (JMS) API 相同的接口集。 IBM Message Service Client for .NET 包含一个完全受管的 XMS 实现，此实现可由任何与 .NET 兼容的语言使用。

XMS 支持：

- 点到点 消息传递
- 发布/预订 消息传递
- 同步消息传递
- 异步消息传递

XMS 应用程序可以与以下类型的应用程序交换消息：

- XMS 应用程序

- IBM MQ classes for JMS 应用程序
- 本机 IBM MQ 应用程序
- 使用 IBM MQ 缺省消息传递提供程序的 JMS 应用程序

XMS 应用程序可以连接到以下任何消息传递服务器并使用其资源：

IBM MQ 队列管理器
应用程序可以在绑定或客户机方式下连接。

WebSphere Application Server 服务集成总线
应用程序可以使用直接 TCP/IP 连接，也可以使用“基于 TCP/IP 的 HTTP”。

IBM Integration Bus
将使用 WebSphere MQ 实时传输 在应用程序和代理程序之间传输消息。 可使用 WebSphere MQ Multicast Transport 将消息传递到应用程序。

通过连接到 IBM MQ 队列管理器，XMS 应用程序可以使用 WebSphere MQ 企业传输 来与 IBM Integration Bus 进行通信。 或者，XMS 应用程序可以通过连接到 IBM MQ 来进行发布和预订。

相关概念
第 6 页的『消息传递类型』
第 7 页的『XMS 对象模型』
XMS API 是一个面向对象的接口。 XMS 对象模型基于 JMS 1.1 对象模型。
第 9 页的『XMS 消息模型』
XMS 消息模型与 IBM MQ classes for JMS 消息模型相同。

消息传递类型

XMS 支持 点到点 和 发布/预订 消息传递类型。

消息传递类型也称为消息传递域。

点到点 消息传递

一种常见的 点到点 消息传递形式是使用排队。 在最简单的情况下，一个应用程序通过隐式或显式标识目标队列来将消息发送到另一个应用程序。 底层的消息传递和排队系统将从发送应用程序接收消息，然后将此消息传递至其目标队列。 然后，接收应用程序可以从该队列中检索此消息。

如果底层的消息传递和排队系统包含 IBM Integration Bus，那么 IBM Integration Bus 会复制消息并将消息副本传递到不同的队列。 因此，可能有多个应用程序收到该消息。 IBM Integration Bus 可能还会转换消息并向其添加数据。

点到点 消息传递的一个关键特征是：应用程序在发送消息时会将消息放入本地队列中。 底层的消息传递和排队系统会确定要将消息发送到哪个目标队列。 接收应用程序将从目标队列中检索消息。

发布/预订 消息传递

在 发布/预订 消息传递中，有两种类型的应用程序：发布程序和预订程序。

发布程序采用发布消息的形式提供信息。 当发布程序发布消息时，它会指定主题以用于标识消息内信息的主题。

预订程序是所发布信息的使用者。 预订程序通过创建预订来指定其感兴趣的主题。

发布/预订系统将接收来自发布程序的发布和来自预订程序的预订。 它会将发布内容传递到预订程序。 预订程序仅接收其预订的那些主题的发布内容。

发布/预订 消息传递的一个关键特征是：发布程序在发布消息时标识主题。 它不会标识预订程序。 如果针对某个主题发布的消息没有预订程序，那么所有应用程序都不会收到该消息。

应用程序可以既是发布者也是订户。

XMS 对象模型

XMS API 是一个面向对象的接口。 XMS 对象模型基于 JMS 1.1 对象模型。

以下列表汇总了主要 XMS 类或对象类型：

ConnectionFactory
ConnectionFactory 对象封装了连接的一组参数。 应用程序使用 ConnectionFactory 来创建连接。 应用程序可以在运行时提供参数并创建 ConnectionFactory 对象。 或者，可以将连接参数存储在受管对象存储库中。 应用程序可以从该存储库中检索对象，并通过该对象来创建 ConnectionFactory 对象。

Connection
Connection 对象封装了从应用程序到消息传递服务器的活动连接。 应用程序使用连接来创建会话。

Destination
应用程序使用 Destination 对象来发送消息或接收消息。 在发布/预订域中，Destination 对象封装了一个主题，而在点到点域中，Destination 对象封装了一个队列。 应用程序可以提供参数以在运行时创建 Destination 对象。 或者，可以通过存储在受管对象存储库中的对象定义来创建 Destination 对象。

Session
Session 对象是用于发送和接收消息的单线程上下文。 应用程序使用 Session 对象创建 Message、MessageProducer 和 MessageConsumer 对象。

Message
Message 对象封装了应用程序使用 MessageProducer 对象发送或使用 MessageConsumer 对象接收的 Message 对象。

MessageProducer
应用程序使用 MessageProducer 对象将消息发送到目标。

MessageConsumer
应用程序使用 MessageConsumer 对象接收已发送到目标的消息。

第 7 页的图 1 显示了这些对象及其关系。 此图显示了 XMS 对象的主要类型：ConnectionFactory、Connection、Session、MessageProducer、MessageConsumer、Message 和 Destination。 应用程序使用连接工厂来创建连接，并使用连接来创建会话。 然后，应用程序可以使用会话来创建消息、消息生产者和消息使用者。 应用程序使用消息生产者向目标发送消息，并使用消息使用者接收向目标发送的消息。

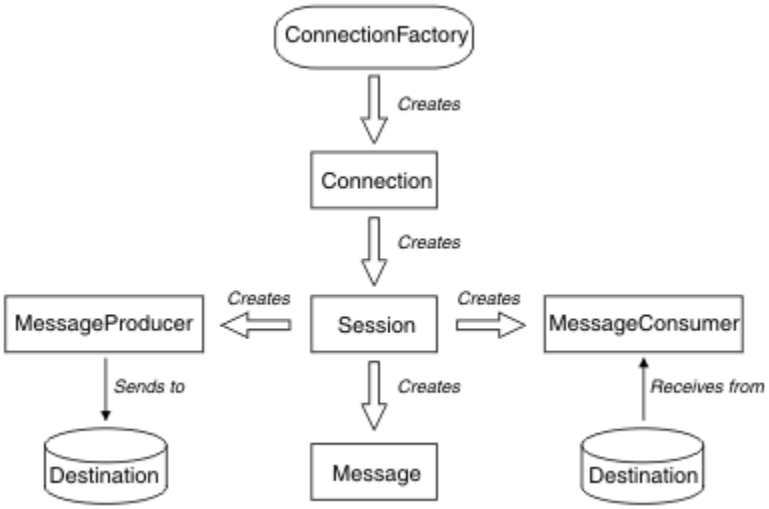

图 *1: XMS* 对象及其关系

在 .NET 中，XMS 类定义为一组 .NET 接口。 在编写 XMS .NET 应用程序的代码时，只需要已声明的接口。

XMS 对象模型基于 Java Message Service Specification V1.1 中描述的独立于域的接口。 未提供特定于域的类，如 Topic、TopicPublisher 和 TopicSubscriber。

对象的属性和特性

XMS 对象可具有属性 (attribute) 和特性 (property)，它们是对象的特征，可采用不同的方式来实现。

属性
始终存在且占用存储空间（即使属性不含值也是如此）的对象特征。 从这方面讲，属性与固定长度数据结构中的字段类似。 属性的显著特点是：每个属性都有自己的用于设置和获取其值的方法。

属性
对象的特性存在且仅在设置其值后才占用存储空间。 在设置特性的值之后，无法删除该属性或恢复其存储空间。 您可以更改其值。 XMS 提供一组常规方法来用于设置和获取特性值。

相关概念

XMS 基本类型
XMS 提供了 8 种 Java 基本类型（byte、short、int、long、float、double、char 和 boolean）的等效项。 这样便可以在 XMS 与 JMS 之间交换消息而不丢失或损坏数据。

属性值从一种数据类型到另一种数据类型的隐式转换
在应用程序获取某个属性的值后，XMS 可以将该值转换为另一种数据类型。 许多规则可控制受支持的转换以及 XMS 执行转换的方式。

相关参考

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

受管对象

通过使用受管对象，可以在中央存储库中管理您希望管理的客户机应用程序所使用的连接设置。 应用程序从中央存储库中检索对象定义，然后使用这些定义来创建 `ConnectionFactory` 和 `Destination` 对象。 通过使用受管对象，可以将应用程序与其在运行时使用的资源分离开来。

例如，可以使用受管对象（引用了测试环境中的一组连接和目标）来编写和测试 XMS 应用程序。 在部署应用程序时，可以更改受管对象，以将应用程序配置为引用生产环境中的连接和目标。

XMS 支持以下两种类型的受管对象：

- `ConnectionFactory` 对象，可供应用程序用来与服务器建立初始连接。
- `Destination` 对象，可供应用程序用来指定所发送消息的目标以及所接收消息的源。 目标是应用程序要连接到的服务器上的主题或队列。

IBM MQ 随附了管理工具 **`JMSAdmin`** 。 它用于在受管对象的中央存储库中创建和管理受管对象。

该存储库中的受管对象可由 IBM MQ classes for JMS 和 XMS 应用程序使用。 XMS 应用程序可以使用 `ConnectionFactory` 和 `Destination` 对象来连接到 IBM MQ 队列管理器。 管理员可以更改该存储库中保存的对象定义，而不影响应用程序代码。

下图显示了 XMS 应用程序通常如何使用受管对象。 该图的左侧显示存储库，其中包含使用管理控制台管理的 ConnectionFactory 和 Destination 对象定义。 该图的右侧显示 XMS 应用程序，此应用程序在该存储库中查找对象定义，然后在连接到消息传递服务器时使用这些对象定义。

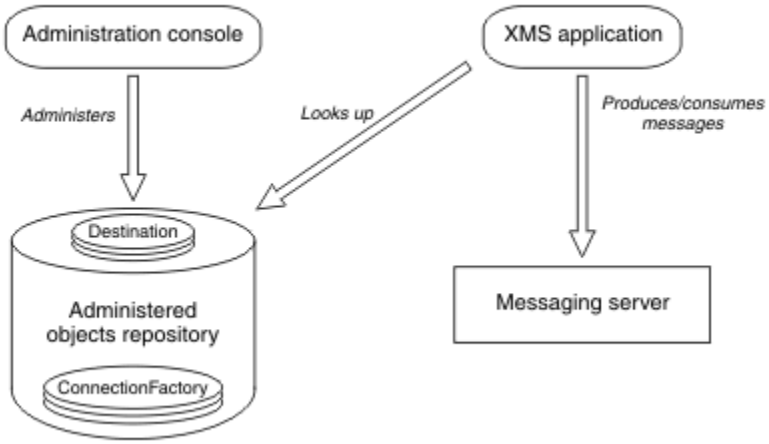

图 *2:* 受管对象在 *XMS* 应用程序中的典型用途

相关概念

使用受管对象

本部分中的主题提供了有关受管对象的信息。 XMS 应用程序可以从中央受管对象存储库中检索对象定义，并使用这些定义来创建连接工厂和目标。

受支持的受管对象存储库类型

“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

相关任务

创建受管对象

必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

XMS 消息模型

XMS 消息模型与 IBM MQ classes for JMS 消息模型相同。

特别是，XMS 实现的消息头字段和消息属性与 IBM MQ classes for JMS 实现的相同：

- JMS 头字段。 这些字段的名称以前缀 JMS 开头。
- JMS 定义的属性。 这些字段的属性名以前缀 JMSX 开头。
- IBM 定义的属性。 这些字段的属性名以前缀 JMS_IBM_ 开头。

因此，XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息。 在每条消息中，某些头字段和属性由应用程序设置，而其他头字段和属性由 XMS 或 IBM MQ classes for JMS 设置。 由 XMS 或 IBM MQ classes for JMS 设置的某些字段是在发送消息时设置，而其他字段是在接收消息时设置。 在适用情况下，通过消息传递服务器，将头字段和属性随消息一起传播。 它们可用于任何接收消息的应用程序。

使用安装向导安装 Message Service Client for .NET

安装将使用 InstallShield X/Windows MSI 安装程序。 提供了两个安装选项，因此您可以选择完整安装或定制安装。

关于此任务

要在 Windows 上安装 Message Service Client for .NET ，请遵循以下过程。

过程

1. 如果要通过 SupportPac 进行安装，请完成以下步骤，否则直接转至步骤 第 10 页的『2』。
   a) 在 Windows 上，以管理员身份登录。
   b) 运行 dotNETClientsetup.exe 安装程序。
2. 等待安装向导打开并显示以下消息：

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

   单击**下一步**。

   该向导可能会要求您阅读许可协议。
3. 如果要求您阅读许可协议，并且您接受许可协议的条款，请单击**我接受许可协议的条款**，然后单击**下一步**。

   安装向导会要求您选择最符合您需求的安装类型。
4. 选择您所需的安装类型：
   - 要安装所有程序功能部件，并且在缺省安装目录中安装，请单击**完整**。
   - 要选择您希望安装的功能部件，并指定安装位置，请单击**定制**。
5. 单击**下一步**。

   如果选择“完整”安装选项，那么安装向导将显示一条消息来表明已准备好安装，如步骤 第 10 页的『8』 中所述。 如果选择“定制”安装选项，那么安装向导会要求您选择希望安装的功能部件，并且您必须完成步骤 第 10 页的『6』 和步骤 第 10 页的『7』，然后才能转至步骤 第 10 页的『8』。
6. 仅针对定制安装，请单击功能部件列表中的图标，以指定关于您希望如何安装 Message Service Client for .NET 功能部件的任何更改。 如果您不想在建议的目录中安装 Message Service Client for .NET，请选择其他目录。

   如果选择在当前不存在的目录中安装 Message Service Client for .NET，那么安装向导会为您创建该目录。

   如果您想要开发 XMS 应用程序，请确保选择了**开发工具和样本**功能部件。 该功能部件提供了样本应用程序以及编译 .NET 应用程序所需的库和其他文件。 如果不选择该功能部件，那么只会安装运行 XMS 应用程序所需的文件。
7. 如果使用的是“定制”安装选项，请在选择所需的选项后，单击**下一步**，如步骤 第 10 页的『6』 中所述。

   安装向导将显示一条消息来表明已准备好安装。
8. 单击**安装**以开始安装。

   安装向导将显示一个进度条，用于显示安装的进度。 等待进度条完成。 成功完成安装后，窗口中将显示以下消息：

   ```
   The installation wizard has successfully installed IBM Message Service Client
   for .NET. Click Finish to exit the wizard.
   ```
9. 单击**完成**以关闭安装向导。

结果

您已成功安装 Message Service Client for .NET，现在可以开始使用。

下一步做什么

在运行任何 XMS 应用程序（包括 XMS 随附的样本应用程序）之前，必须设置消息传递服务器环境，有关详细信息，请参阅：第 11 页的『设置消息传递服务器环境』。

相关概念

JNDI 查找 Web Service

要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

设置消息传递服务器环境
本部分中的主题描述了如何设置消息传递服务器环境以允许 XMS 应用程序连接到服务器。

使用 XMS 样本应用程序
通过使用 XMS 随附的样本应用程序，可以验证安装和消息传递服务器设置，并帮助构建您自己的应用程序。 这些样本概括了每个 API 的常用功能。

XMS 应用程序连接到 WebSphere MQ 的先决条件

在 XMS 应用程序连接到 WebSphere MQ 时，需要满足一些先决条件。

对于连接到 WebSphere MQ 队列管理器的应用程序，必须在用于运行 XMS 应用程序的机器上安装相应的 WebSphere MQ 客户机库。 这些库已预安装在具有本地队列管理器的机器上。

对于 XMS Client for .NET，请使用 IBM WebSphere MQ 7.0.1.0 或更高版本随附的客户机库。 这些是 *.NET* 的 *WebSphere MQ* 类。 它们启用与 IBM WebSphere MQ 7.0， 6.0 和 5.3 队列管理器的客户机方式连接，以及与本地队列管理器的绑定方式连接 (如果也是 IBM WebSphere MQ 7.0.1.0 或更高版本)。

Microsoft .NET Framework 2.0 Redistributable Package 必须安装在要安装 XMS 的计算机上。 如果此软件包不可用，那么 XMS 安装将失败。 然后，您需要退出安装过程，在计算机上安装 Microsoft .NET Framework 2.0 Redistributable Package ，然后重新运行安装过程。

在 Microsoft 下载站点上，需要查找 dotnetfx.exe for Microsoft .NET Framework 2.0 Redistributable Package (x86) 和 NetFx64.exe for Microsoft .NET Framework 2.0 Redistributable Package (x64) (以适用者为准)。

相关概念
设置消息传递服务器环境
本部分中的主题描述了如何设置消息传递服务器环境以允许 XMS 应用程序连接到服务器。

防止应用程序使用较新的 XMS 版本

缺省情况下，在安装了较新的 XMS 版本后，使用先前版本的应用程序会自动切换到较新的版本，而无需重新编译。

关于此任务

“多版本共存”功能可确保安装较新的 XMS 版本时不会覆盖先前的 XMS 版本。 相似 XMS .NET 组合件的多个实例共存于全局组合件高速缓存 (GAC) 中，但具有不同的版本号。 在内部，GAC 使用策略文件将应用程序调用传递到最新版本的 XMS。 应用程序无需重新编译即可运行，并且可以使用较新的 XMS .NET 版本中提供的新功能。

但是，如果应用程序需要使用较早的 XMS 版本，可将应用程序配置文件中的 `publisherpolicy` 属性设置为 no。

注: 应用程序配置文件的名称包含相关可执行程序的名称且后缀为 .config。 例如，text.exe 的应用程序配置文件的名称为 text.exe.config。

但是，在任何时候，系统的所有应用程序都使用相同版本的 XMS .NET。

设置消息传递服务器环境

本部分中的主题描述了如何设置消息传递服务器环境以允许 XMS 应用程序连接到服务器。

对于连接到 IBM MQ 队列管理器的应用程序，需要 IBM MQ 客户机（或者，如果是绑定方式，那么需要队列管理器）。

对于使用与代理程序的实时连接的应用程序，目前不存在任何先决条件。

运行任何 XMS 应用程序（包括 XMS 随附的样本应用程序）之前，必须设置消息传递服务器环境。

本部分包含以下主题：

- 第 12 页的『为连接到 IBM MQ 队列管理器的应用程序配置队列管理器和代理程序』
- 第 13 页的『为使用与代理程序的实时连接的应用程序配置代理程序』
- 第 14 页的『为连接到 WebSphere Application Server 的应用程序配置服务集成总线』

相关概念

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

使用 XMS 样本应用程序
通过使用 XMS 随附的样本应用程序，可以验证安装和消息传递服务器设置，并帮助构建您自己的应用程序。 这些样本概括了每个 API 的常用功能。

相关任务

使用安装向导安装 Message Service Client for .NET
安装将使用 InstallShield X/Windows MSI 安装程序。 提供了两个安装选项，因此您可以选择完整安装或定制安装。

相关参考

XMS 应用程序连接到 WebSphere MQ 的先决条件
在 XMS 应用程序连接到 WebSphere MQ 时，需要满足一些先决条件。

为连接到 IBM MQ 队列管理器的应用程序配置队列管理器和代理程序

本部分假定您使用的是 IBM WebSphere MQ 7.0.1 或更高版本。 在运行连接到 IBM MQ 队列管理器的应用程序之前，必须先配置该队列管理器。 对于发布/预订应用程序，如果使用的是排队式发布/预订接口，那么需要进行一些额外配置。

开始之前

XMS 使用 IBM Integration Bus 或 WebSphere Message Broker 6.1 或更高版本

在开始此任务之前，请执行以下步骤：

- 确保应用程序有权访问正在运行的队列管理器。
- 如果应用程序是发布/预订应用程序并且使用排队式发布/预订接口，请确保队列管理器上的“PSMODE”属性设置为“ENABLED”。
- 确保已相应地设置了应用程序所用连接工厂的属性以连接到队列管理器。 如果应用程序是发布/预订应用程序，请确保已设置了相应的连接工厂属性，以便使用代理程序。 有关连接工厂的属性的更多信息，请参阅第 160 页的『ConnectionFactory 属性』。

关于此任务

按照配置队列管理器和排队式发布/预订接口以运行 IBM MQ JMS 应用程序的相同方式，配置队列管理器和代理程序以运行 XMS 应用程序。 以下步骤概括了您需要执行的操作。

过程

1. 在队列管理器上，创建应用程序所需的队列。

   有关如何创建队列的概述，请参阅定义队列。

   如果应用程序是发布/预订应用程序，并且使用需要访问 IBM MQ classes for JMS 系统队列的排队式发布/预订接口，请等待完成步骤 4a，然后才能创建队列。
2. 向与应用程序相关的用户标识授予可连接到队列管理器和访问队列的相应权限。

   有关授权的概述，请参阅保护。 如果应用程序以客户机方式连接到队列管理器，请参阅客户机和服务器。

3. 如果应用程序以客户机方式连接到队列管理器，请确保已在队列管理器中定义服务器连接通道，并且已启动侦听器。

   您无需针对连接到队列管理器的每个应用程序都执行此步骤。 通过一个服务器连接通道定义和一个侦听器，就可以支持在客户机方式下连接的所有应用程序。
4. 如果应用程序是发布/预订应用程序，并且使用排队式发布/预订接口，请执行以下步骤。
   a) 在队列管理器上，通过运行 IBM MQ 随附的 MQSC 命令脚本来创建 IBM MQ classes for JMS 系统队列。 确保与 IBM Integration Bus 或 WebSphere Message Broker 相关的用户标识有权访问这些队列。

      有关可在何处找到该脚本以及如何运行该脚本的信息，请参阅使用 IBM MQ classes for Java。

      针对队列管理器仅执行该步骤一次。 通过同一组 IBM MQ classes for JMS 系统队列，就可以支持连接到队列管理器的所有 XMS 和 IBM MQ classes for JMS 应用程序。

   b) 向与应用程序相关的用户标识授予可访问 IBM MQ classes for JMS 系统队列的相应权限。

      有关用户标识需要哪些权限的信息，请参阅使用 IBM MQ classes for JMS。

   c) 对于 IBM Integration Bus 或 WebSphere Message Broker 代理程序，请创建并部署消息流，以便服务于可供应用程序发送其发布的消息的队列。

      基本消息流包括一个用于读取已发布消息的 MQInput 消息处理节点和一个用于发布消息的 Publication 消息处理节点。

      有关如何创建和部署消息流的信息，请参阅 IBM Integration Bus 产品文档库 Web 页面中提供的 IBM Integration Bus 或 WebSphere Message Broker 产品文档。

      如果代理程序上已部署了合适的消息流，那么无需执行该步骤。

结果

现在，您可以启动应用程序。

相关任务

为使用与代理程序的实时连接的应用程序配置代理程序
在运行使用与代理程序的实时连接的应用程序之前，必须先配置该代理程序。

为连接到 WebSphere Application Server 的应用程序配置服务集成总线
在运行连接到 WebSphere Application Server service integration technologies 服务集成总线的应用程序之前，必须先配置该服务集成总线，其配置方法与配置服务集成总线以运行使用缺省消息传递提供程序的 JMS 应用程序的方式相同。

为使用与代理程序的实时连接的应用程序配置代理程序

在运行使用与代理程序的实时连接的应用程序之前，必须先配置该代理程序。

开始之前

在开始此任务之前，请执行以下步骤：

- 确保应用程序有权访问正在运行的代理程序。
- 确保已根据与代理程序的实时连接相应地设置了应用程序所用连接工厂的属性。 有关连接工厂的属性的更多信息，请参阅第 160 页的『ConnectionFactory 属性』。

关于此任务

按照配置代理程序以运行 IBM MQ classes for JMS 应用程序的相同方式，配置代理程序以运行 XMS 应用程序。 以下步骤概括了您需要执行的操作：

过程

1. 创建并部署消息流，以从代理程序侦听的 TCP/IP 端口读取消息并发布这些消息。

   您可以使用以下任一方式来执行此操作：

- 创建包含 **Real-timeOptimizedFlow** 消息处理节点的消息流。
- 创建包含 **Real-timeInput** 消息处理节点和 Publication 消息处理节点的消息流。

必须配置 **Real-timeOptimizedFlow** 或 **Real-timeInput** 节点，才能侦听用于实时连接的端口。 在 XMS 中，实时连接的缺省端口号为 1506。

如果代理程序上已部署了合适的消息流，那么无需执行该步骤。

2. 如果需要使用 IBM MQ classes for JMS 将消息传递到应用程序，请配置代理程序以启用多点广播。 配置那些需要启用多点广播的主题，为需要可靠多点广播的主题指定可靠的服务质量。
3. 如果应用程序在连接到代理程序时提供用户标识和密码，并且您希望代理程序使用这些信息来认证应用程序，请对用户名服务器和代理程序配置简单的 telnet 类型密码认证。

结果

现在，您可以启动应用程序。

相关任务

为连接到 IBM MQ 队列管理器的应用程序配置队列管理器和代理程序
本部分假定您使用的是 IBM WebSphere MQ 7.0.1 或更高版本。 在运行连接到 IBM MQ 队列管理器的应用程序之前，必须先配置该队列管理器。 对于发布/预订应用程序，如果使用的是排队式发布/预订接口，那么需要进行一些额外配置。

为连接到 WebSphere Application Server 的应用程序配置服务集成总线
在运行连接到 WebSphere Application Server service integration technologies 服务集成总线的应用程序之前，必须先配置该服务集成总线，其配置方法与配置服务集成总线以运行使用缺省消息传递提供程序的 JMS 应用程序的方式相同。

为连接到 WebSphere Application Server 的应用程序配置服务集成总线

在运行连接到 WebSphere Application Server service integration technologies 服务集成总线的应用程序之前，必须先配置该服务集成总线，其配置方法与配置服务集成总线以运行使用缺省消息传递提供程序的 JMS 应用程序的方式相同。

开始之前

在开始此任务之前，必须执行以下步骤：

- 确保已创建消息传递总线，并且您的服务器已作为总线成员添加到该总线中。
- 确保应用程序有权访问至少包含一个正在运行的消息传递引擎的服务集成总线。
- 如果需要 HTTP 操作，那么必须定义 HTTP 消息传递引擎入站传输通道。 缺省情况下，会在服务器安装期间定义用于 SSL 和 TCP 的通道。
- 确保已相应地设置了应用程序所用连接工厂的属性，以便使用引导程序服务器连接到服务集成总线。 必须至少提供以下信息：
  - 提供程序端点，用于描述在协商与消息传递服务器的连接（即，通过引导程序服务器）时要使用的位置和协议。 在最简单的形式中，对于使用缺省设置安装的服务器，提供程序端点可设置为服务器的主机名。
  - 用于发送消息的总线的名称。

  有关连接工厂的属性的更多信息，请参阅第 160 页的『ConnectionFactory 属性』。

关于此任务

必须定义所需的任何队列或主题空间。 缺省情况下，会在服务器安装期间定义名为 Default.Topic.Space 的主题空间，但如果需要更多主题空间，您必须自行创建。 您不需要预定义主题空间中的主题，因为服务器会根据需要动态实例化这些主题。

以下步骤概括了您需要执行的操作。

过程

1. 创建在应用程序执行点到点消息传递时需要的队列。
2. 创建在应用程序执行发布/预订消息传递时需要的任何额外主题空间。

结果

现在，您可以启动应用程序。

相关任务

为连接到 IBM MQ 队列管理器的应用程序配置队列管理器和代理程序
本部分假定您使用的是 IBM WebSphere MQ 7.0.1 或更高版本。 在运行连接到 IBM MQ 队列管理器的应用程序之前，必须先配置该队列管理器。 对于发布/预订应用程序，如果使用的是排队式发布/预订接口，那么需要进行一些额外配置。

为使用与代理程序的实时连接的应用程序配置代理程序
在运行使用与代理程序的实时连接的应用程序之前，必须先配置该代理程序。

使用 XMS 样本应用程序

通过使用 XMS 随附的样本应用程序，可以验证安装和消息传递服务器设置，并帮助构建您自己的应用程序。 这些样本概括了每个 API 的常用功能。

相关概念

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

设置消息传递服务器环境
本部分中的主题描述了如何设置消息传递服务器环境以允许 XMS 应用程序连接到服务器。

相关任务

使用安装向导安装 Message Service Client for .NET
安装将使用 InstallShield X/Windows MSI 安装程序。 提供了两个安装选项，因此您可以选择完整安装或定制安装。

样本应用程序

样本应用程序概述了每个 API 的常用功能。 使用这些样本应用程序，可以验证安装和消息传递服务器设置，并帮助您构建自己的应用程序。

如果您在创建自己的应用程序时需要帮助，那么可以使用这些样本应用程序作为起点。 已提供了每个样本应用程序的源代码和已编译的版本。 请查看样本源代码，并确定关键步骤，以便为应用程序创建所有必需对象（ConnectionFactory、Connection、Session、Destination 以及 Producer 和/或 Consumer）并根据需要设置任何特定属性以指定应用程序的工作方式。 有关更多信息，请参阅第 17 页的『编写 XMS 应用程序』。
在 XMS 的未来发行版中可随时更改这些样本。

下表显示了 XMS 随附的三组样本应用程序（每个 API 各一组）。

样本名称	描述
SampleConsumerCS	从队列中获取消息或预订主题的消息使用者应用程序。
SampleProducerCS	将消息生成到队列中或针对主题生成消息的消息生产者应用程序。
SampleConfigCS	可用于创建基于文件的受管对象存储库的配置应用程序。 该应用程序包含适用于特殊连接设置的连接工厂和目标。 然后，可以将此受管对象存储库与各个样本使用者应用程序和生成者应用程序一起使用。

支持各种 API 中相同功能的样本存在一些语法差异。

- 样本消息使用者和生产者应用程序都支持以下功能：

  - 连接到 IBM MQ、IBM Integration Bus（使用与代理程序的实时连接）和 WebSphere Application Server service integration bus
  - 使用初始上下文接口执行受管对象存储库查找
  - 连接到队列（IBM MQ 和 WebSphere Application Server service integration bus）和主题（IBM MQ、与代理程序的实时连接和 WebSphere Application Server service integration bus）
  - 基本、字节、映射、对象、流和文本消息
- 样本消息使用者应用程序支持同步和异步接收方式以及 SQL 选择器语句。
- 样本消息生产者应用程序支持持久性和非持久性传递方式。

运行方式

这些样本可以在以下两种方式下运行：

简单方式
您可以使用最少的用户输入来运行样本。

高级方式
您可以更细微地定制样本的运行方式。

所有样本都兼容，因此可以跨语言运行。

相关概念
构建自己的应用程序
您可以像构建样本应用程序那样构建自己的应用程序。

相关任务
运行样本应用程序
您可以在简单或高级方式下，以交互方式运行 .NET 样本应用程序，或者通过自动生成或定制的响应文件，以非交互方式运行这些样本应用程序。

构建 .NET 样本应用程序
构建 .NET 样本应用程序时，会创建所选样本的可执行版本。

运行样本应用程序

您可以在简单或高级方式下，以交互方式运行 .NET 样本应用程序，或者通过自动生成或定制的响应文件，以非交互方式运行这些样本应用程序。

开始之前

在运行所提供的任何样本应用程序之前，必须首先设置消息传递服务器环境，以便应用程序可以连接到服务器。 请参阅第 11 页的『设置消息传递服务器环境』。

过程

要运行 .NET 样本应用程序，请完成以下步骤：

提示: 在运行样本应用程序时，输入? 随时获取有关下一步操作的帮助。

1. 选择要用于运行样本应用程序的方式。
   输入 `Advanced` 或 `Simple`。
2. 回答问题。
   要选择缺省值（显示在问题结尾处的方括号中），请按 Enter 键。 要选择其他值，请输入相应的值，然后按 Enter 键。
   下面是一个示例问题：

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

   在这种情况下，缺省值为 `wpm` (连接到 WebSphere Application Server service integration bus)。

结果

运行样本应用程序时，会在当前工作目录中自动生成响应文件。 响应文件名的格式为 *connection_type-sample_type*.rsp；例如 wpm-producer.rsp。 如果需要，可以使用生成的响应文件，采用相同的选项来重新运行样本应用程序，因此不需要再次输入这些选项。

相关概念

样本应用程序

样本应用程序概述了每个 API 的常用功能。 使用这些样本应用程序，可以验证安装和消息传递服务器设置，并帮助您构建自己的应用程序。

相关任务

构建 .NET 样本应用程序

构建 .NET 样本应用程序时，会创建所选样本的可执行版本。

构建 .NET 样本应用程序

构建 .NET 样本应用程序时，会创建所选样本的可执行版本。

开始之前

安装相应的编译器。 此任务假定您已安装了 Microsoft Visual Studio 2012，并且熟悉它的使用方法。

过程

要构建 .NET 样本应用程序，请完成以下步骤：

1. 单击 .NET 样本随附的 Samples.sln 解决方案文件。
2. 在“解决方案资源管理器”窗口中右键单击解决方案**样本**，然后选择**构建解决方案**。

结果

此时会在该样本的相应子文件夹（bin/Debug 或 bin/Release，取决于您所选的配置）中创建可执行程序。 此程序与该文件夹同名，其后缀为 CS。 例如，如果要构建 C# 版本的消息生产者样本应用程序，那么将在 SampleProducer 文件夹中创建 SampleProducerCS.exe。

相关概念

样本应用程序

样本应用程序概述了每个 API 的常用功能。 使用这些样本应用程序，可以验证安装和消息传递服务器设置，并帮助您构建自己的应用程序。

第 36 页的『构建自己的应用程序』

您可以像构建样本应用程序那样构建自己的应用程序。

相关任务

运行样本应用程序

您可以在简单或高级方式下，以交互方式运行 .NET 样本应用程序，或者通过自动生成或定制的响应文件，以非交互方式运行这些样本应用程序。

开发 XMS 应用程序

本部分中的主题提供了在编写 XMS 应用程序时可能有用的信息。

有关编写 XMS 应用程序的信息，请参阅以下主题：

编写 XMS 应用程序

本部分中的主题所提供的信息可帮助您编写 XMS 应用程序。

本部分包含编写 XMS 应用程序时涉及的常用概念。 另请参阅第 36 页的『编写 XMS .NET 应用程序』，以获取与创建 .NET 应用程序有关的信息。

本部分包含以下主题：

- 第 18 页的『线程技术模型』
- 第 18 页的『ConnectionFactories 和 Connection 对象』
- 第 21 页的『会话』
- 第 24 页的『目标』
- 第 28 页的『消息生产者』
- 第 28 页的『消息使用者』
- 第 31 页的『队列浏览器』
- 第 31 页的『请求程序』
- 第 32 页的『删除对象』
- 第 33 页的『XMS 基本类型』
- 第 33 页的『属性值从一种数据类型到另一种数据类型的隐式转换』
- 第 35 页的『迭代器』
- 第 36 页的『编码字符集标识』
- 第 36 页的『XMS 错误和异常代码』
- 第 36 页的『构建自己的应用程序』

相关概念

编写 XMS .NET 应用程序
本部分中的主题所提供的信息可帮助您编写 XMS .NET 应用程序。

相关参考

.NET 接口
本部分记录了 .NET 类接口及其属性和方法。

线程技术模型

一般规则可控制多线程应用程序如何使用 XMS 对象。

- 只能在不同线程上同时使用以下类型的对象：
  - ConnectionFactory
  - Connection
  - ConnectionMetaData
  - Destination
- 在任何时候，都只能在单个线程上使用 Session 对象。

这些规则的异常由 第 74 页的『Message Service Clients for .NET 参考』中方法的接口定义中标有 “线程上下文” 的条目指示。

相关概念

可在运行时处理的错误情况
来自 API 调用的返回码是可在运行时处理的错误情况。 处理此类错误的方式取决于您使用的是 C 还是 C++ API。

ConnectionFactories 和 Connection 对象

ConnectionFactory 对象提供一种模板以供应用程序用于创建 Connection 对象。 应用程序可使用 Connection 对象来创建 Session 对象。

对于 .NET，XMS 应用程序首先使用 XMSFactoryFactory 对象获取对所需协议类型对应的 ConnectionFactory 对象的引用。 然后，该 ConnectionFactory 对象会建立仅适用于该协议类型的连接。

XMS 应用程序可以创建多个连接，而多线程应用程序可在多个线程上同时使用同一个 Connection 对象。 Connection 对象用于封装应用程序与消息传递服务器之间的通信连接。

连接可提供多种用途：

- 在应用程序创建连接时，可认证该应用程序。
- 应用程序可将唯一的客户机标识与连接相关联。 客户机标识用于支持发布/预订域中的持久预订。 可通过以下两种方式设置客户机标识：

  分配连接客户机标识的首选方法是使用属性在特定于客户机的 ConnectionFactory 对象中进行配置，并采用透明方式将该标识分配给其创建的连接。

  分配客户机标识的另一种方法是使用 Connection 对象上设置的特定于提供者的值。 该值不会覆盖由管理人员配置的标识。 在管理人员未指定标识的情况下，可提供该值。 如果管理人员指定的标识确实存在，那么尝试使用特定于提供者的值覆盖该值会导致抛出异常。 如果应用程序要显式设置标识，那么必须在创建连接之后且在连接上执行其他操作之前立即执行该操作，否则会抛出异常。

XMS 应用程序通常会创建一个连接、一个或多个会话，以及大量的消息生产者和消息使用者。

创建连接需要使用大量的系统资源，因为这不仅要建立通信连接，还要认证应用程序。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

相关参考

IConnectionFactory（适用于 .NET 接口）
应用程序使用连接工厂来创建连接。

ConnectionFactory 属性
下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

连接启动和停止方式

可以在启动或停止方式下运行连接。

应用程序创建连接时，该连接处于停止方式。 如果连接处于停止方式，那么应用程序可以初始化会话，可以发送消息但无法接收消息（同步或异步）。

应用程序可以通过调用 `Start Connection` 方法来启动连接。 如果连接处于启动方式，那么应用程序可以发送和接收消息。 应用程序随后可通过调用 Stop Connection 和 `Start Connection` 方法来停止和重新启动连接。

相关概念

关闭连接
应用程序可通过调用 Close Connection 方法来关闭连接。

异常处理
如果应用程序仅使用连接来异步使用消息，那么它只能通过异常侦听器来了解连接问题。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

关闭连接

应用程序可通过调用 Close Connection 方法来关闭连接。

在应用程序关闭连接时，XMS 会执行以下操作：

- 关闭与连接相关联的所有会话，并删除与这些会话相关联的某些对象。 有关删除哪些对象的更多信息，请参阅第 32 页的『删除对象』。 同时，XMS 会回滚这些会话中当前正在执行的所有事务。
- 结束与消息传递服务器的通信连接。
- 释放连接所使用的内存及其他内部资源。

在关闭连接之前，对于在会话期间确认失败的消息，XMS 不会确认收到这类消息。 有关确认收到消息的更多信息，请参阅第 22 页的『消息确认』。

相关概念

连接启动和停止方式
可以在启动或停止方式下运行连接。

异常处理
如果应用程序仅使用连接来异步使用消息，那么它只能通过异常侦听器来了解连接问题。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

异常处理

如果应用程序仅使用连接来异步使用消息，那么它只能通过异常侦听器来了解连接问题。

XMS .NET 异常都派生自 System.Exception。 有关更多信息，请参阅第 40 页的『.NET 中的错误处理』。

相关概念

连接启动和停止方式
可以在启动或停止方式下运行连接。

关闭连接
应用程序可通过调用 Close Connection 方法来关闭连接。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

连接到服务集成总线

XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

HTTP 协议可在无法使用直接 TCP/IP 连接的情况下使用。 一种常见的情况是通过防火墙进行通信，例如，当两个企业交换消息时。 使用 HTTP 利用防火墙通信通常称为 *HTTP* 隧道。 但是，与使用直接 TCP/IP 连接相比，HTTP 隧道的速度要慢一些，因为 HTTP 头会显著增加传输的数据量，而且 HTTP 协议需要的通信量也多于 TCP/IP。

要创建 TCP/IP 连接，应用程序可以使用其 XMSC_WPM_TARGET_TRANSPORT_CHAIN 属性设置为 XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC 的连接工厂。 这是该属性的缺省值。 如果成功创建连接，那么该连接的 XMSC_WPM_CONNECTION_PROTOCOL 属性设置为 XMSC_WPM_CP_TCP。

要创建使用 HTTP 的连接，应用程序必须使用其 XMSC_WPM_TARGET_TRANSPORT_CHAIN 属性设置为入站传输链名称（即配置为使用 HTTP 传输通道）的连接工厂。 如果成功创建连接，那么该连接的 XMSC_WPM_CONNECTION_PROTOCOL 属性设置为 XMSC_WPM_CP_HTTP。 有关如何配置传输链的信息，请参阅 WebSphere Application Server 产品文档中的配置传输链。

应用程序在连接到引导程序服务器时可使用类似的通信协议选项。 连接工厂的 XMSC_WPM_PROVIDER_ENDPOINTS 属性是引导程序服务器的一个或多个端点地址的序列。 每个端点地址的引导程序传输链组件可以是 XMSC_WPM_BOOTSTRAP_TCP（针对到引导程序服务器的 TCP/IP 连接）或 XMSC_WPM_BOOTSTRAP_HTTP（针对使用 HTTP 的连接）。

相关概念

连接启动和停止方式
可以在启动或停止方式下运行连接。

关闭连接
应用程序可通过调用 Close Connection 方法来关闭连接。

异常处理
如果应用程序仅使用连接来异步使用消息，那么它只能通过异常侦听器来了解连接问题。

相关任务

创建受管对象

必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

相关参考

IConnectionFactory（适用于 .NET 接口）
应用程序使用连接工厂来创建连接。

ConnectionFactory 属性
下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

会话

会话是用于发送和接收消息的单线程上下文。

应用程序可以使用会话来创建消息、消息生产者、消息使用者、队列浏览器和临时目标。 应用程序还可以使用会话来运行本地事务。

应用程序可以创建多个会话，其中每个会话独立于其他会话来生成和使用消息。 如果不同会话（甚至是同一个会话）中的两个消息使用者预订同一个主题，那么每个消息使用者都会收到该主题上任何已发布消息的副本。

与 Connection 对象不同，不能在不同的线程中同时使用 Session 对象。 只可以通过 Session 对象所使用线程以外的其他线程来调用 Session 对象的 Close Session 方法。 Close Session 方法将结束会话，并释放已分配给该会话的所有系统资源。

如果应用程序必须在多个线程上同时处理消息，那么应用程序必须在每个线程上创建一个会话，然后在该线程上使用该会话执行任何发送或接收操作。

事务性会话

XMS 应用程序可以运行本地事务。 本地事务只涉及更改应用程序连接到的队列管理器或服务集成总线的资源。

仅当应用程序连接到 IBM MQ 队列管理器或 WebSphere Application Server 服务集成总线时，此主题中的信息才有用。 这些信息不适用于与代理程序的实时连接。

要运行本地事务，应用程序必须首先通过调用 Connection 对象的 Create Session 方法并使用参数将会话指定为事务性来创建事务性会话。 然后，会话内发送和接收的所有消息将分组到事务序列中。 应用程序提交或回滚事务开始后所发送和接收的消息时，表明事务结束。

要落实事务，应用程序可调用 Session 对象的 Commit 方法。 提交事务时，事务中发送的所有消息可用于发送到其他应用程序，而事务内收到的所有消息将得到确认，这样一来消息传递服务器便不会尝试再次将这些消息发送到应用程序。 在点到点域中，消息传递服务器还会从队列中移除收到的消息。

要回滚事务，应用程序可调用 Session 对象的 Rollback 方法。 回滚事务时，消息传递服务器将废弃事务内发送的所有消息，而事务内收到的所有消息可用于再次发送。 在点到点域中，之前收到的消息将放回队列并可再次被其他应用程序看到。

在应用程序创建事务性会话或调用 Commit 或 Rollback 方法时，会自动启动新的事务。 因此，事务性会话总是有活动事务。

当应用程序关闭事务性会话时，会发生隐式回滚。 当应用程序关闭连接时，所有连接的事务性会话将发生隐式回滚。

事务完全包含在事务性会话中。 事务不能跨越多个会话。 这表示一个应用程序不能在两个或多个事务性会话中发送和接收消息然后作为单个事务提交或回滚所有操作。

相关概念

消息确认
每个非事务性会话都有一种确认模式，用于确定如何确认应用程序收到的消息。 共有三种确认模式可用，确认模式的选择会影响应用程序的设计。

异步消息传递
XMS 使用一个线程来处理会话的所有异步消息传递。 这意味着每次只能运行一个消息侦听器函数或一个 onMessage() 方法。

同步消息传递
如果应用程序使用 MessageConsumer 对象的 Receive 方法，那么会将消息同步传递到该应用程序。

消息传递方式
XMS 支持两种消息传递方式。

消息确认

每个非事务性会话都有一种确认模式，用于确定如何确认应用程序收到的消息。 共有三种确认模式可用，确认模式的选择会影响应用程序的设计。

仅当应用程序连接到 IBM MQ 队列管理器或 WebSphere Application Server 服务集成总线时，此主题中的信息才有用。 这些信息不适用于与代理程序的实时连接。

XMS 使用相同的机制来确认收到 JMS 使用的消息。

如果会话为非事务性，那么确认应用程序收到的消息的方法取决于该会话的确认模式。 以下段落描述了这三种确认模式：

XMSC_AUTO_ACKNOWLEDGE
会话自动确认应用程序收到的每个消息。

如果消息同步发送到应用程序，会话会在每次 Receive 调用成功完成时确认收到消息。

如果应用程序成功收到消息，但存在故障阻止进行确认，那么该消息将变为可重新发送。 因此，应用程序必须能够处理重新传递的消息。

XMSC_DUPS_OK_ACKNOWLEDGE
会话在进行选择时确认应用程序收到消息。

使用此确认模式可减少会话必须执行的工作量，但是阻止消息确认的故障可能会导致多个消息变为可重新发送。 因此，应用程序必须能够处理重新传递的消息。

XMSC_CLIENT_ACKNOWLEDGE
应用程序通过调用 Message 类的 Acknowledge 方法来确认它所收到的消息。

应用程序可以单独地确认收到每个消息，或者可以接收一批消息并只为收到的最后一条消息调用 Acknowledge 方法。 如果调用 Acknowledge 方法，自上次调用此方法以来收到的所有消息都将得到确认。

与以上任一确认模式配合使用，应用程序可以通过调用 Session 类的 Recover 方法停止并重新启动会话中的消息发送。 将重新传递先前未确认接收的消息。 但是，这些消息的发送顺序可能会与之前的发送顺序不同。 在此期间，优先级较高的消息可能已到达，而某些原始消息可能已到期。 在点到点域中，某些原始消息可以已被另一个应用程序使用。

应用程序可以通过检查消息的 JMSRedelivered 头字段的内容来确定消息是否会重新发送。 应用程序通过调用 Message 类的 Get JMSRedelivered 方法来实现此目标。

相关概念
事务性会话
XMS 应用程序可以运行本地事务。 本地事务只涉及更改应用程序连接到的队列管理器或服务集成总线的资源。

异步消息传递
XMS 使用一个线程来处理会话的所有异步消息传递。 这意味着每次只能运行一个消息侦听器函数或一个 onMessage() 方法。

同步消息传递
如果应用程序使用 MessageConsumer 对象的 Receive 方法，那么会将消息同步传递到该应用程序。

消息传递方式
XMS 支持两种消息传递方式。

异步消息传递

XMS 使用一个线程来处理会话的所有异步消息传递。 这意味着每次只能运行一个消息侦听器函数或一个 onMessage() 方法。

如果会话中的多个消息使用者正在异步接收消息，并且消息侦听器函数或 onMessage() 方法正在将消息传递给某个消息使用者，那么等待同一消息的其他消息使用者必须继续等待。 等待传递给会话的其他消息也必须继续等待。

如果应用程序需要同时传递消息，请创建多个会话，以便 XMS 能够使用多个线程来处理异步消息传递。 通过使用这种方式，可同时运行多个消息侦听器函数或 onMessage() 方法。

无法通过将消息侦听器分配给使用者来使会话异步。 只有在调用 Connection.Start 方法时，才能使会话异步。 在调用 Connection.Start 方法之前，允许所有同步调用。 在调用 Connection.Start 时，会开始将消息传递给使用者。

如果必须在异步会话上进行同步调用 (例如，创建使用者或生产者)，那么必须调用 Connection.Stop 。 可以通过调用 Connection.Start 方法来恢复会话，以开始传递消息。 这里的唯一例外情况是 Session 消息传递线程（即将消息传递到回调函数的线程）。 这个线程可以在消息回调函数中对会话发出任何调用（Close 调用除外）。

注: 在非受管方式下，回调函数中的 MQDISC 调用不受 IBM MQ .NET 客户机支持。 因此，客户机应用程序无法在异步接收方式下在 MessageListener 回调内创建或关闭会话。 可在 MessageListener 方法外创建和处置会话。

相关概念

事务性会话
XMS 应用程序可以运行本地事务。 本地事务只涉及更改应用程序连接到的队列管理器或服务集成总线的资源。

消息确认
每个非事务性会话都有一种确认模式，用于确定如何确认应用程序收到的消息。 共有三种确认模式可用，确认模式的选择会影响应用程序的设计。

同步消息传递
如果应用程序使用 MessageConsumer 对象的 Receive 方法，那么会将消息同步传递到该应用程序。

消息传递方式
XMS 支持两种消息传递方式。

同步消息传递

如果应用程序使用 MessageConsumer 对象的 Receive 方法，那么会将消息同步传递到该应用程序。

通过使用 Receive 方法，应用程序可以在指定的时间段内等待消息，也可以无限期地等待。 或者，如果应用程序不想等待消息，那么可以使用“Receive with No Wait”方法。

相关概念

事务性会话
XMS 应用程序可以运行本地事务。 本地事务只涉及更改应用程序连接到的队列管理器或服务集成总线的资源。

消息确认
每个非事务性会话都有一种确认模式，用于确定如何确认应用程序收到的消息。 共有三种确认模式可用，确认模式的选择会影响应用程序的设计。

异步消息传递
XMS 使用一个线程来处理会话的所有异步消息传递。 这意味着每次只能运行一个消息侦听器函数或一个 onMessage() 方法。

消息传递方式
XMS 支持两种消息传递方式。

消息传递方式

XMS 支持两种消息传递方式。

- 持久消息都会传递一次。 消息传递服务器会采取特殊的预防措施（例如，记录消息），以确保在传输过程中不会丢失持久消息，即使在发生故障的情况下也是如此。
- 非持久消息最多传递一次。 与持久消息相比，非持久消息的可靠性差一些，因为可能由于发生故障而导致在传输过程中丢失非持久消息。

可通过权衡可靠性与性能来选择合适的传递方式。 通常，非持久消息比持久消息传输得更快。

相关概念

事务性会话
XMS 应用程序可以运行本地事务。 本地事务只涉及更改应用程序连接到的队列管理器或服务集成总线的资源。

消息确认
每个非事务性会话都有一种确认模式，用于确定如何确认应用程序收到的消息。 共有三种确认模式可用，确认模式的选择会影响应用程序的设计。

异步消息传递
XMS 使用一个线程来处理会话的所有异步消息传递。 这意味着每次只能运行一个消息侦听器函数或一个 `onMessage()` 方法。

同步消息传递
如果应用程序使用 MessageConsumer 对象的 Receive 方法，那么会将消息同步传递到该应用程序。

目标

XMS 应用程序可使用 Destination 对象来指定所发送消息的目标以及所接收消息的源。

XMS 应用程序可以在运行时创建 Destination 对象，也可以从受管对象存储库中获取预定义的目标。

对于 ConnectionFactory，XMS 应用程序指定目标的最灵活方式是将其定义为受管对象。 通过使用此方法，用 C、C++、.NET 语言以及 Java 编写的应用程序可以共享目标的定义。 可以在不更改任何代码的情况下更改受管 Destination 对象的属性。

对于 .NET 应用程序，可以使用 CreateTopic 或 CreateQueue 方法来创建目标。 .NET API 中的 ISession 和 XMSFactoryFactory 对象中都提供了这两个方法。 有关更多信息，请参阅第 38 页的『.NET 中的目标』和第 91 页的『IDestination』。

相关参考

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

主题统一资源标识

主题统一资源标识 (URI) 指定主题的名称，还可以指定主题的一个或多个属性。

主题 URI 以序列 topic:// 开头，后跟主题名称以及（可选）用于设置其余主题属性的名称/值对列表。 主题名称不能为空。

以下示例显示了 .NET 代码片段：

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

有关主题属性的更多信息（包括可在 URI 中使用的名称和有效值），请参阅第 165 页的『Destination 属性』。

在指定要在预订中使用的主题 URI 时，可以使用通配符。 这些通配符的语法取决于连接类型和代理程序版本；提供了以下选项：

- 具有字符级别通配符格式的 IBM WebSphere MQ 7.0 队列管理器
- 具有主题级别通配符格式的 IBM WebSphere MQ 7.0 队列管理器
- WebSphere Application Server 服务集成总线

具有字符级别通配符格式的 IBM WebSphere MQ 7.0 队列管理器

具有字符级别通配符格式的 IBM WebSphere MQ 7.0 队列管理器使用以下通配符：

* 表示 0 个或更多个字符
? 表示 1 个字符
% 表示转义字符

第 25 页的表 1 提供了一些有关如何使用此通配符方案的示例。

表 *1:* 对 *IBM WebSphere MQ 7.0* 队列管理器使用字符级别通配符方案的示例 *URI*

统一资源标识	匹配项	示例
"topic://Sport*Results"	以“Sport”开头且以“Results”结束的所有主题	"topic://SportsResults" 和 "topic://Sport/Hockey/National/Div3/Results"
"topic://Sport?Results"	以“Sport”开头、后跟单个字符、再后跟“Results”的所有主题	"topic://SportsResults" 和 "topic://SportXResults"
"topic://Sport/*ball*/Div?/Results/*/???"	主题	"topic://Sport/Football/Div1/Results/2002/Nov" 和 "topic://Sport/Netball/National/Div3/Results/02/Jan"

具有主题级别通配符格式的 IBM WebSphere MQ 7.0 队列管理器

具有主题级别通配符格式的 IBM WebSphere MQ 7.0 队列管理器使用以下通配符：

可匹配多个级别
+ 可匹配单个级别

第 25 页的表 2 提供了一些有关如何使用此通配符方案的示例。

表 *2:* 对 *IBM WebSphere MQ 7.0* 队列管理器使用主题级别通配符方案的示例 *URI*

统一资源标识	匹配项	示例
"topic://Sport/+/Results"	在 Sport 与 Results 之间具有单个分层级别名称的所有主题	"topic://Sport/Football/Results" 和 "topic://Sport/Ju-Jitsu/Results"
"topic://Sport/#/Results"	以“Sport/”开头且以“/Results”结束的所有主题	"topic://Sport/Football/Results" 和 "topic://Sport/Hockey/National/Div3/Results"
"topic://Sport/Football/#"	以“Sport/Football/”开头的所有主题	"topic://Sport/Football/Results" 和 "topic://Sport/Football/TeamNews/Signings/Managerial"

WebSphere Application Server 服务集成总线

WebSphere Application Server 服务集成总线使用以下通配符：

* 可与层次结构中一个级别上的任意字符匹配
// 可与 0 个或更多个级别匹配
//. 可与 0 个或更多个级别匹配（位于主题表达式末尾）

第 26 页的表 3 提供了一些有关如何使用此通配符方案的示例。

表 3: 对 *WebSphere Application Server* 服务集成总线使用通配符方案的示例 *URI*		
统一资源标识	**匹配项**	**示例**
"topic://Sport/*ball/Results"	在 Sport 与 Results 之间具有以“ball”结尾的单个分层级别名称的所有主题	"topic://Sport/Football/Results" 和 "topic://Sport/Netball/Results"
"topic://Sport//Results"	以“Sport/”开头且以“/Results”结束的所有主题	"topic://Sport/Football/Results" 和 "topic://Sport/Hockey/National/Div3/Results"
"topic://Sport/Football//."	以“Sport/Football/”开头的所有主题	"topic://Sport/Football/Results" 和 "topic://Sport/Football/TeamNews/Signings/Managerial"
"topic://Sport/*ball//Results//."	主题	"topic://Sport/Football/Results" 和 "topic://Sport/Netball/National/Div3/Results/2002/November"

相关概念

队列统一资源标识
队列 URI 指定队列的名称，还可以指定队列的一个或多个属性。

临时目标
XMS 应用程序可以创建和使用临时目标。

目标通配符
XMS 支持目标通配符，可确保将通配符传递到需要进行匹配的位置。 XMS 可使用的每种服务器类型各有不同的通配符方案。

相关参考

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

队列统一资源标识

队列 URI 指定队列的名称，还可以指定队列的一个或多个属性。

队列 URI 以序列 queue:// 开头，后跟队列名称；它还可能包含用于设置其余队列属性的名称/值对列表。

对于 IBM MQ 队列（但不针对 WebSphere Application Server 缺省消息传递提供程序队列），可以在队列前指定队列所在的队列管理器，使用 / 将队列管理器名称与队列名称分隔开。

如果指定队列管理器，那么对于使用该队列的连接而言，它必须是 XMS 直接连接到的队列管理器，或者必须可通过该队列进行访问。 远程队列管理器仅支持从队列中检索消息，而不支持将消息放入队列中。 有关完整详细信息，请参阅 IBM MQ 队列管理器文档。

如果未指定任何队列管理器，那么可以选择使用额外的 / 分隔符，它存在与否对于队列定义没有影响。

对于 XMS 直接连接到的队列管理器 QM_A 上的 IBM MQ 队列 QB 而言，以下队列定义全都等效：

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

相关概念

主题统一资源标识
主题统一资源标识 (URI) 指定主题的名称，还可以指定主题的一个或多个属性。

临时目标
XMS 应用程序可以创建和使用临时目标。

目标通配符

XMS 支持目标通配符，可确保将通配符传递到需要进行匹配的位置。 XMS 可使用的每种服务器类型各有不同的通配符方案。

相关参考

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

临时目标

XMS 应用程序可以创建和使用临时目标。

应用程序通常使用临时目标来接收请求消息的应答。 要指定将请求消息的应答发送到的目标，应用程序可调用表示请求消息的 Message 对象的 Set JMSReplyTo 方法。 该调用上指定的目标可以是临时目标。

虽然是使用会话来创建临时目标，但是临时目标的作用域实际上是用于创建该会话的连接。 连接的任何会话都可以为临时目标创建消息生产者和消息使用者。 临时目标会一直保留，直至将其显式删除或连接终止（以先发生者为准）。

在应用程序创建临时队列时，将在应用程序所连接到的消息传递服务器中创建队列。 如果应用程序连接到队列管理器，那么通过模型队列（通过 XMSC_WMQ_TEMPORARY_MODEL 属性指定其名称）来创建动态队列，并且将通过 XMSC_WMQ_TEMP_Q_PREFIX 属性来指定用于构成动态队列名称的前缀。 如果应用程序连接到服务集成总线，那么将在总线中创建临时队列，并且将通过 XMSC_WPM_TEMP_Q_PREFIX 属性来指定用于构成临时队列名称的前缀。

当连接到服务集成总线的应用程序创建临时主题时，将通过 XMSC_WPM_TEMP_TOPIC_PREFIX 属性来指定用于构成临时主题名称的前缀。

相关概念

主题统一资源标识
主题统一资源标识 (URI) 指定主题的名称，还可以指定主题的一个或多个属性。

队列统一资源标识
队列 URI 指定队列的名称，还可以指定队列的一个或多个属性。

目标通配符
XMS 支持目标通配符，可确保将通配符传递到需要进行匹配的位置。 XMS 可使用的每种服务器类型各有不同的通配符方案。

相关参考

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

目标通配符

XMS 支持目标通配符，可确保将通配符传递到需要进行匹配的位置。 XMS 可使用的每种服务器类型各有不同的通配符方案。

这些方案如下：

连接类型	通配符方案	描述
WebSphere MQ 队列管理器	* ? %	0 个或更多个字符 1 个字符 转义字符
与代理程序的实时连接	# +	与多个级别匹配 与一个级别匹配

连接类型	通配符方案	描述
WebSphere Service Integration Bus	*	与层次结构中一个级别上的任意字符匹配
	//	与 0 个或更多个级别匹配
	//.	与 0 个或更多个级别匹配（位于 Topic 表达式结尾处）

相关概念

主题统一资源标识
主题统一资源标识 (URI) 指定主题的名称，还可以指定主题的一个或多个属性。

队列统一资源标识
队列 URI 指定队列的名称，还可以指定队列的一个或多个属性。

临时目标
XMS 应用程序可以创建和使用临时目标。

相关参考

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

消息生产者

在 XMS 中，可以创建消息生产者（具有有效目标或无相关目标）。 在使用空目标创建消息生产者时，需要在发送消息时指定有效目标。

无相关目标的消息生产者

在 XMS .NET 中，可使用空目标创建消息生产者。

要在使用 .NET API 时创建没有关联目标的消息生产者，必须将 NULL 作为参数传递到 ISession 对象的 `CreateProducer()` 方法 (例如， `session.CreateProducer(null)`)。 但是，发送消息时必须指定有效的目标。

具有相关目标的消息生产者

在此场景中，将使用有效目标来创建消息生产者。 在发送操作期间，无需指定目标。

消息使用者

消息使用者可以分为持久和非持久订户以及同步和异步消息使用者。

持久订户

持久订户是接收主题上所有已发布消息（包括当订户处于不活动状态时发布的消息）的消息使用者。

仅当应用程序连接到 IBM MQ 队列管理器或 WebSphere Application Server 服务集成总线时，此主题中的信息才有用。 这些信息不适用于与代理程序的实时连接。

要为主题创建持久订户，应用程序将调用 Session 对象的 Create Durable Subscriber 方法，并将用于标识持久预订的名称和表示主题的 Destination 对象指定为参数。 应用程序可以使用也可以不使用消息选择器来创建持久订户，并且可以指定持久订户是否接收自己的连接所发布的消息。

用于创建持久订户的会话必须具有相关的客户机标识。 客户机标识与用于创建会话的连接的相关标识相同；其指定方式如 第 18 页的『ConnectionFactories 和 Connection 对象』中所述。

用于标识持久预订的名称在客户机标识中必须唯一，因此客户机标识将构成完整且唯一的持久预订标识的一部分。 消息传递服务器将维护持久预订的记录，并确保一直保留主题上发布的所有消息，直至持久订户确认这些消息或这些消息到期。

甚至在持久订户关闭之后，消息传递服务器仍将继续维护持久预订的记录。 要复用先前创建的持久预订，应用程序必须创建一个持久订户，方法是指定与持久预订相关的预订名称并使用具有与持久预订相关的客户机标识的会话。 每个会话针对特殊的持久预订每次只能有一个持久订户。

持久预订的作用域是负责维护预订记录的消息传递服务器。 如果连接到不同消息传递服务器的两个应用程序使用相同的预订名称和客户机标识各自创建了一个持久订户，那么将创建两个完全独立的持久预订。

要删除持久预订，应用程序可调用 Session 对象的 Unsubscribe 方法，并将用于标识持久预订的名称指定为参数。 与会话相关的客户机标识必须与持久预订的相关标识相同。 消息传递服务器将删除其维护的持久预订记录，并且不会向持久订户再发送任何消息。

要更改现有预订，应用程序可以使用相同的预订名称和客户机标识但指定不同的主题和/或消息选择器来创建持久订户。 更改持久预订等效于删除预订并创建新预订。

对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序，XMS 将管理订户队列。 因此，应用程序无需指定订户队列。 如果指定了订户队列，XMS 将忽略该订户队列。

但是，对于连接到 IBM WebSphere MQ 6.0 队列管理器的应用程序，每个持久订户都必须具有指定的订户队列。 要为主题指定订户队列名称，请设置表示主题的 Destination 对象的 XMSC_WMQ_DUR_SUBQ 属性。 缺省订户队列为 SYSTEM.JMS.D.SUBSCRIBER.QUEUE。

连接到 IBM WebSphere MQ 6.0 队列管理器的持久订户可以共享同一个订户队列，或者每个持久订户都可以从自己专用的订户队列中检索其消息。 有关要为应用程序采用的方法的讨论，请参阅 *XMSIBM WebSphere MQ* 使用 *Java*。

请注意，您不能更改持久预订的订户队列。 更改订户队列的唯一方式是删除预订并创建新预订。

对于连接到服务集成总线的应用程序，每个持久订户都必须具有指定的主持久预订。 要为使用同一个连接的所有持久订户指定主持久预订，请设置用于创建连接的 ConnectionFactory 对象的 XMSC_WPM_DUR_SUB_HOME 属性。 要为个别主题指定主持久预订，请设置表示该主题的 Destination 对象的 XMSC_WPM_DUR_SUB_HOME 属性。 必须先为连接指定持久预订宿主，然后应用程序才能创建使用此连接的持久订户。 为目标指定的任何值将覆盖为连接指定的值。

非持久订户

非持久订户是仅接收在订户处于活动状态时发布的消息的消息使用者。 在订户处于不活动状态时传递的消息将会丢失。

仅当通过 IBM WebSphere MQ 6.0 队列管理器使用发布/预订消息传递时，此主题中的信息才有用。

如果未在关闭连接之前或期间删除使用者对象，那么消息可以保留在不再活动的订户的代理程序队列中。

在此情况下，可以使用“用于 JMS 的 IBM WebSphere MQ classes for JMS 类”随附的清除实用程序来清除队列中的这些消息。 在使用 *Java* 的 *IBM WebSphere MQ* 中提供了有关如何使用该实用程序的详细信息。 如果订户队列中保留了大量消息，那么可能还需要增加该队列的队列深度。

同步消息使用者

同步消息使用者从队列同步接收消息。

同步消息使用者每次接收一条消息。 如果使用 `Receive(wait interval)` 方法，那么此调用仅在指定的时间段（以毫秒计）内等待消息或者直至关闭消息使用者为止。

如果使用 ReceiveNoWait() 方法，那么同步消息使用者会立即接收消息而无任何延迟；如果下一条消息可用，将立即接收该消息，否则将返回指向空 Message 对象的指针。

异步消息使用者

异步消息使用者采用异步方式从队列接收消息。 只要队列中有可用的新消息，就会调用由应用程序注册的消息侦听器。

XMS 中的有害消息

有害消息是接收 MDB 应用程序无法处理的消息。 如果遇到有害消息，XMS MessageConsumer 对象可以根据队列属性 BOQUEUE 和 BOTHRESH 将其重新排队。

在某些情况下，传送到 MDB 的消息可能会回滚到 IBM MQ 队列上。 例如，如果在工作单元内传递消息，随后进行回滚，那么可能会发生此情况。 通常，将会再次传递已回滚的消息，但格式错误的消息可能会不断地

导致 MDB 发生故障，从而无法进行传递。 此类消息称为有害消息。 您可以配置 IBM MQ，以便将有害消息自动传输到另一个队列以供进一步调查或丢弃。 有关如何使用此方式配置 IBM MQ 的信息，请参阅在 ASF 中处理有害消息。

有时队列中会收到格式不正确的消息。 这种情况下，格式错误表示接收方应用程序无法正确处理此消息。 此类消息可能会导致接收方应用程序故障并回退此格式错误的消息。 该消息随后将重复传递到输入队列并被应用程序重复回退。 这些消息称为有害消息。 XMS MessageConsumer 对象将检测有害消息并将其重新发送到备用目标。

IBM MQ 队列管理器会记录每个消息的回退次数。 当该次数达到配置的阈值时，消息使用者会将消息重新排入指定回退队列。 如果重新排队因为任何原因失败，该消息将从输入队列中移除或重新排入死信队列，或被废弃。

XMS ConnectionConsumer 对象使用相同的队列属性以相同的方式处理有害消息。 如果有多个连接使用者监视同一队列，可能是重新排队发生之前中毒消息发送到应用程序的次数已超过阈值。 此行为是因为单个连接使用者监视队列和对中毒消息重新排队的方式。

回退队列的阈值和名称是 IBM MQ 队列的属性。 属性名称为 BackoutThreshold 和 BackoutRequeueQName。 它们适用于的队列如下所示：

- 对于点到点消息传递，这是底层本地队列。 在消息使用者和连接使用者使用队列别名时，这非常重要。
- 对于 IBM MQ 消息传递提供程序正常方式下的发布/预订消息传递，它是用于创建 Topic 受管队列的模型队列。
- 对于 IBM MQ 消息传递提供程序迁移模式中的发布/预订消息传递，适用队列为 TopicConnectionFactory 对象中定义的 CCSUB 队列，或者 Topic 主题上定义的 CCDSUB 队列。

要设置 BackoutThreshold 和 BackoutRequeueQName 属性，请发出以下 MQSC 命令：

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

对于发布/预订消息传递，如果您的系统为每个预订创建动态队列，那么从“用于 JMS 的 WebSphere MQ 类”模型队列 SYSTEM.JMS.MODEL.QUEUE 获取这些属性值。 要更改这些设置，请使用：

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

如果回退阈值是 0，那么将禁用有害消息处理，并将有害消息保留在输入队列中。 否则，回退计数值达到阈值时，消息将被发送到指定的回退队列。

如果回退计数达到阈值，但是消息无法转到回退队列中，那么会将该消息发送到死信队列，或者如果该消息是非持久消息，那么会将其丢弃。

如果没有定义回退队列，或者如果 MessageConsumer 对象无法将消息发送至回退队列，会出现此情况。

配置系统以执行有害消息处理

XMS .NET 在查询 **BOTHRESH** 和 **BOQNAME** 属性时所使用的队列取决于所执行消息传递的样式：

- 对于点到点消息传递，这是底层本地队列。 当 XMS .NET 应用程序使用来自别名队列或集群队列的消息时，这一点十分重要。
- 对于发布/预订消息传递，会创建受管队列来保存应用程序的消息。 XMS .NET 会查询受管队列以确定 **BOTHRESH** 和 **BOQNAME** 属性的值。

  受管队列是根据与应用程序预订的“主题”对象所关联的模型队列创建的，并从该模型队列继承 **BOTHRESH** 和 **BOQNAME** 属性的值。 使用的模型队列取决于接收应用程序已取得持久预订还是非持久预订。

  – 用于持久预订的模型队列由“主题”的 **MDURMDL** 属性来指定。 该属性的缺省值为 SYSTEM.DURABLE.MODEL.QUEUE。
  – 对于非持久预订，所使用的模型队列由 **MNDURMDL** 属性来指定。 **MNDURMDL** 属性的缺省值为 SYSTEM.NDURABLE.MODEL.QUEUE。

查询 **BOTHRESH** 和 **BOQNAME** 属性时，XMS .NET 会执行以下操作：

- 打开本地队列或别名队列的目标队列。
- 查询 **BOTHRESH** 和 **BOQNAME** 属性。
- 关闭本地队列或别名队列的目标队列。

打开本地队列或别名队列的目标队列时所使用的打开选项，取决于所使用的 IBM MQ 的版本：

- 对于 IBM MQ 9.0.0 Fix Pack 8 和更低版本，如果本地队列或别名队列的目标队列是集群队列，那么 XMS .NET 将使用 `MQOO_INPUT_AS_Q_DEF`， `MQOO_INQUIRE` 和 `MQOO_FAIL_IF_QUIESCING` 选项打开队列。 这意味着运行接收应用程序的用户必须对集群队列的本地实例具有查询和获取访问权。

  XMS .NET 会使用 `MQOO_INQUIRE` 和 `MQOO_FAIL_IF_QUIESCING` 打开选项来打开所有其他类型的本地队列。 为了使 XMS .NET 能够查询属性值，运行接收应用程序的用户必须对本地队列具有查询访问权。
- V 9.0.0.9 使用 XMS .NET from IBM MQ 9.0.0 Fix Pack 9 时，无论队列类型如何，运行接收应用程序的用户都必须对本地队列具有查询访问权。

要将有害消息移至回退重新排队队列或队列管理器的死信队列，必须向运行应用程序的用户授予 `put` 和 `passall` 权限。

在 *ASF* 中处理有害消息
使用应用程序服务器工具 (ASF) 时，ConnectionConsumer 而不是 MessageConsumer 会处理有害消息。 ConnectionConsumer 会根据队列的 BackoutThreshold 和 BackoutRequeueQName 属性对消息重新排队。

应用程序使用 ConnectionConsumer 时，回退消息的环境取决于应用程序服务器提供的会话。

- 会话为具有 AUTO_ACKNOWLEDGE 或 DUPS_OK_ACKNOWLEDGE 的非事务性会话时，仅会在发生系统错误后或应用程序意外终止的情况下回退消息。
- 如果会话是带有 CLIENT_ACKNOWLEDGE 的非事务性会话，那么可通过应用程序服务器调用 `Session.recover()` 来回退未确认的消息。

  通常由 MessageListener 的客户机实现或应用程序服务器调用 `Message.acknowledge()`。 `Message.acknowledge()` 确认会话上到目前为止已传递的所有消息。
- 如果会话是事务性会话，那么可通过应用程序服务器调用 `Session.rollback()` 来回退未确认的消息。

队列浏览器

应用程序使用队列浏览器浏览而不移除队列中的消息。

要创建队列浏览器，应用程序可调用 ISession 对象的 Create Queue Browser 方法，并将用于标识要浏览队列的 Destination 对象指定为参数。 应用程序可以使用也可以不使用消息选择器来创建队列浏览器。

在创建队列浏览器之后，应用程序可调用 IQueueBrowser 对象的 GetEnumerator 方法以获取队列中消息的列表。 此方法将返回用于封装 Message 对象列表的枚举符。 列表中 Message 对象的顺序与从队列中检索消息的顺序相同。 然后，应用程序可以使用该枚举符来依次浏览各条消息。

在将消息放入队列中以及从队列中除去消息时会动态更新该枚举符。 每次应用程序调用 IEnumerator.MoveNext() 来浏览队列上的下一条消息时，该消息会反映队列的当前内容。

针对给定的队列浏览器，应用程序可以多次调用 GetEnumerator 方法。 每个调用都会返回一个新枚举符。 因此，应用程序可以使用多个枚举符来浏览队列中的消息，并维护队列内的多个位置。

应用程序可以使用队列浏览器来搜索要从队列中移除的适当消息，然后将消息使用者与消息选择器一起用于移除该消息。 消息选择器可以根据 JMSMessageID 头字段的值来选择消息。 有关该头字段及其他 JMS 消息头字段的信息，请参阅 第 59 页的『XMS 消息中的头字段』。

请求程序

应用程序使用请求程序来发送请求消息，然后等待接收应答。

许多消息传递应用程序都基于用于发送请求消息并等待应答的算法。 XMS 提供一个名为 Requestor 的类，以帮助开发这种类型的应用程序。

要创建请求程序，应用程序可调用 Requestor 类的 Create Requestor 构造函数，并将用于标识请求消息要发送到何处的 Session 对象和 Destination 对象指定为参数。 此会话不能是事务性会话，也不能使用确认方式 XMSC_CLIENT_ACKNOWLEDGE。 构造函数将自动创建要将应答消息发送到的临时队列或主题。

在创建请求程序之后，应用程序可调用 Requestor 对象的 Request 方法来发送请求消息，然后等待接收来自负责接收请求消息的应用程序的应答。 此调用将一直等待，直至收到应答或该会话结束（以先发生者为准）。 针对每个请求消息，请求程序只需要一个应答。

在应用程序关闭请求程序后，将删除临时队列或主题。 但是，不会关闭相关会话。

删除对象

在应用程序删除其创建的 XMS 对象时，XMS 会释放已分配给该对象的内部资源。

在应用程序创建 XMS 对象时，XMS 会为该对象分配内存和其他内部资源。 XMS 将一直保留这些内部资源，直至应用程序通过调用该对象的 close 或 delete 方法来显式删除该对象，届时 XMS 会释放这些内部资源。 如果应用程序尝试删除已删除的对象，那么将忽略此调用。

在应用程序删除 Connection 或 Session 对象时，XMS 会自动删除某些相关对象并释放其内部资源。 这些是由 Connection 或 Session 对象创建的对象，并且其功能都依赖于该对象。 第 32 页的表 4 中显示了这些对象。

注: 如果应用程序关闭与被依赖会话的连接，那么还会删除依赖于这些会话的所有对象。 只有 Connection 或 Session 对象可以具有从属对象。

表 *4:* 自动删除的对象

已删除的对象	**方法**	**自动删除的从属对象**
Connection	关闭连接	ConnectionMetaData 和 Session 对象
Session	关闭会话	MessageConsumer、MessageProducer、QueueBrowser 和 Requestor 对象

通过 XMS 使用受管 IBM MQ XA 事务

可以通过 XMS 使用受管 IBM MQ XA 事务。

要通过 XMS 使用 XA 事务，必须创建事务性会话。 如果正在使用 XA 事务，那么通过分布式事务协调程序 (DTC) 全局事务来实现事务控制，而不是通过 XMS 会话实现事务控制。 在使用 XA 事务时，无法在 XMS 会话上发出 `Session.commit` 或 `Session.rollback`。 请改为使用 `Transscope.Commit` 或 `Transscope.Rollback` DTC 方法来落实或回滚事务。 如果对 XA 事务使用会话，那么使用该会话创建的生产者或使用者必须是 XA 事务的一部分。 它们不能用于 XA 事务作用域外的任何操作。 它们不能用于 XA 事务外的操作，如 `Producer.send` 或 `Consumer.receive`。

如果存在以下情况，那么会抛出 `IllegalStateException` 异常对象：

- XA 事务性会话用于 `Session.commit` 或 `Session.rollback`。
- 在 XA 事务作用域外使用曾在 XA 事务性会话中使用的生产者或使用者对象。

在异步使用者中不支持 XA 事务。

注:

1. 在落实 XA 事务之前，可以在 `Producer`、`Consumer`、`Session` 或 `Connection` 对象上发出 close 调用。 在这样的情况下，将回滚事务中的消息。 同样，如果在 XA 事务落实之前连接中断，那么也会回滚事务中的所有消息。 对于 `Producer` 对象，回滚意味着不将消息放入队列中。 对于 `Consumer` 对象，回滚意味着消息保留在队列中。
2. 如果 `Producer` 对象将具有 `TimeToLive` 的消息放入 `TransactionScope`，并且在该时间过后发出 `commit`，那么该消息可能会在发出 `commit` 之前到期。 在此情况下，该消息将不可用于 `Consumer` 对象。
3. 在线程中不支持 `Session` 对象。 不支持将事务与在线程间共享的 `Session` 对象一起使用。

XMS 基本类型

XMS 提供了 8 种 Java 基本类型（byte、short、int、long、float、double、char 和 boolean）的等效项。这样便可以在 XMS 与 JMS 之间交换消息而不丢失或损坏数据。

第 33 页的表 5 列出了每种 XMS 基本类型的 Java 等效数据类型、大小以及最小值和最大值。

表 *5: XMS* 数据类型及其 *Java* 等效项

XMS 数据类型	**与 Java 兼容的数据类型**	**大小**	**最小值：**	**最大值：**
System.Boolean	布尔值	32 位	false	true
System.SBYTE	字节	8 位	-2^7 (-128)	2^7-1 (127)
System.BYTE	字节	8 位	-2^7 (-128)	2^7-1 (127)
System.CHAR	字节	8 位	-2^7 (-128)	2^7-1 (127)
System.Int16	短整型	16 位	-2^{15} (-32768)	2^{15}-1 (32767)
System.Int32	int	32 位	-2^{31} (-2147483648)	2^{31}-1 (2147483647)
System.Int64	long	64 位	-2^{63} (-9223372036854775808)	2^{63}-1 (9223372036854775807)
System.Single	浮点值	32 位	-3.402823E-38（精度为 7 位数）	3.402823E+38（精度为 7 位数）
System.Double	双精度值	64 位	-1.79769313486231E-308（精度为 15 位数）	1.79769313486231E+308（精度为 15 位数）

相关概念

对象的属性和特性
XMS 对象可具有属性 (attribute) 和特性 (property)，它们是对象的特征，可采用不同的方式来实现。

属性值从一种数据类型到另一种数据类型的隐式转换
在应用程序获取某个属性的值后，XMS 可以将该值转换为另一种数据类型。 许多规则可控制受支持的转换以及 XMS 执行转换的方式。

相关参考

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

属性值从一种数据类型到另一种数据类型的隐式转换

在应用程序获取某个属性的值后，XMS 可以将该值转换为另一种数据类型。 许多规则可控制受支持的转换以及 XMS 执行转换的方式。

对象属性具有名称和值；值具有相关的数据类型，属性的值也称为属性类型。

应用程序使用 PropertyContext 类的方法来获取和设置对象的属性。 要获取某个属性的值，应用程序可调用适用于此属性类型的方法。 例如，要获取整数属性的值，应用程序通常会调用 GetIntProperty 方法。

但是，在应用程序获取某个属性的值后，XMS 可以将该值转换为另一种数据类型。 例如，要获取整数属性的值，应用程序可以调用 GetStringProperty 方法以将此属性的值作为字符串返回。 第 34 页的表 6 中显示了 XMS 支持的转换。

表 6: 从属性类型到其他数据类型的受支持转换	
属性类型	**受支持的目标数据类型**
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 数组	System.String
System.Int16	System.String, System.Int32, System.Int64

以下一般规则可控制受支持的转换：

- 如果在转换期间未丢失数据，那么可以将数字属性值从一种数据类型转换为另一种数据类型。 例如，可以将数据类型为 System.Int32 的属性值转换为数据类型为 System.Int64 的值，但不能将其转换为数据类型为 System.Int16 的值。
- 可以将任何数据类型的属性值转换为字符串。
- 可以将字符串属性值转换为任何其他数据类型，前提是已针对转换正确格式化了该字符串。 如果应用程序尝试转换未正确格式化的字符串属性值，那么 XMS 可能会返回错误。
- 如果应用程序尝试不受支持的转换，那么 XMS 可能会返回错误。

将属性值从一种数据类型转换为另一种数据类型时，以下规则适用：

- 将布尔属性值转换为字符串时，值 true 将转换为字符串 “true”，值 false 将转换为字符串 “false”。
- 在将布尔属性值转换为数字数据类型（包括 System.SByte）时，会将值 true 转换为 1，并将值 false 转换为 0。
- 将字符串属性值转换为布尔值时，字符串 “true”（不区分大小写）或 “1” 将转换为 true，字符串 “false”（不区分大小写）或 “0” 将转换为 false。 无法转换所有其他字符串。
- 在将字符串属性值转换为数据类型为 System.Int32、System.Int64、System.SByte 或 System.Int16 的值时，该字符串必须采用以下格式：

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

  此字符串的各个组成部分定义如下：

  blanks
  可选前导空白字符

  sign
  可选的加号 (+) 或减号 (-) 字符。

  digits
  连续的数字字符序列 (0-9)。 必须至少存在一个数字字符。

  在数字字符序列之后，该字符串可包含其他非数字字符，但是在到达这些字符中的第一个字符后会立即停止转换。 假设该字符串表示十进制整数。

  如果未正确格式化该字符串，那么 XMS 可能会返回错误。
- 在将字符串属性值转换为数据类型为 System.Double 或 System.Float 的值时，该字符串必须采用以下格式：

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

此字符串的各个组成部分定义如下：

blanks
（可选）前导空格字符。

sign
（可选）加号 (+) 或减号 (-) 字符。

digits
连续的数字字符序列 (0-9)。 *digits* 或 *d_digits* 中必须至少存在一个数字字符。

point
（可选）小数点 (.)。

d_digits
连续的数字字符序列 (0-9)。 *digits* 或 *d_digits* 中必须至少存在一个数字字符。

e_char
阶符，为 *E* 或 *e*。

e_sign
（可选）该指数的加号 (+) 或减号 (-) 字符。

e_digits
该指数的连续数字字符序列 (0-9)。 如果该字符串包含指数字符，那么必须至少存在一个数字字符。

在数字字符序列或表示指数的可选字符之后，该字符串可包含其他非数字字符，但是在到达这些字符中的第一个字符后会立即停止转换。 假设该字符串表示十进制浮点数，指数幂为 10。

如果未正确格式化该字符串，那么 XMS 可能会返回错误。

- 在将数字属性值转换为字符串（包括数据类型为 System.SByte 的属性值）时，会将该值转换为字符串表示形式（即十进制数字），而不是转换为包含该值的 ASCII 字符的字符串。 例如，整数 65 将转换为字符串 “65”，而不是字符串 “A”。
- 在将字节数组属性值转换为字符串时，会将每个字节转换为表示该字节的 2 个十六进制字符。 例如，字节数组 {0xF1, 0x12, 0x00, 0xFF} 将转换为字符串 “F11200FF”。

Property 和 PropertyContext 类的方法支持从属性类型转换为其他数据类型。

相关概念

对象的属性和特性
XMS 对象可具有属性 (attribute) 和特性 (property)，它们是对象的特征，可采用不同的方式来实现。

XMS 基本类型
XMS 提供了 8 种 Java 基本类型（byte、short、int、long、float、double、char 和 boolean）的等效项。这样便可以在 XMS 与 JMS 之间交换消息而不丢失或损坏数据。

相关参考

映射消息
映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

流消息
流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

迭代器

迭代器封装了一个对象列表和一个用于维护列表中当前位置的光标。 IBM Message Service Client for C/C++ 中的“迭代器”概念是通过 IBM Message Service Client for .NET 中的 IEnumerator 接口实现。

在创建迭代器时，光标的位置位于第一个对象之前。 应用程序使用迭代器来依次检索每个对象。

IBM Message Service Client for C/C++ 的 Iterator 类等同于 Java 中的 Enumerator 类。XMS .NET 类似于 Java 并使用 IEnumerator 接口。

应用程序可以使用 IEnumerator 来执行以下任务：

- 获取消息的属性
- 获取映射消息主体中的名称/值对
- 浏览队列中的消息
- 获取连接支持的由 JMS 定义的消息属性的名称

编码字符集标识

在 XMS .NET 中，使用本机 .NET 字符串来传递所有字符串。 由于这采用固定编码，所以不需要其他信息即可解释该字符串。 因此，XMS .NET 应用程序不需要 XMSC_CLIENT_CCSID 属性。

XMS 错误和异常代码

XMS 可使用各种错误代码来指示故障。 本文档并未明确列出这些错误代码，因为它们可能因发行版而异。仅记录 XMS 异常代码（格式为 XMS_X_...），因为它们在 XMS 的各发行版中保持相同。

构建自己的应用程序

您可以像构建样本应用程序那样构建自己的应用程序。

如第 17 页的『构建 .NET 样本应用程序』中所述（它还列出了构建自己的 .NET 应用程序所需要满足的先决条件），构建自己的 .NET 应用程序。 有关如何构建自己的应用程序的其他指导，请使用为每个样本应用程序提供的 makefile。

提示: 为了在出现故障的情况下帮助诊断问题，您可能会发现使用包含的符号编译应用程序很有帮助。

相关概念

样本应用程序
样本应用程序概述了每个 API 的常用功能。 使用这些样本应用程序，可以验证安装和消息传递服务器设置，并帮助您构建自己的应用程序。

相关参考

.NET 接口
本部分记录了 .NET 类接口及其属性和方法。

XMS 对象的属性
本章记录了 XMS 定义的对象属性。

通过 *XMS* 自动重新连接 *IBM MQ* 客户机

在使用 IBM WebSphere MQ 7.1 客户机和更高版本作为消息提供者的情况下，将 XMS 客户机配置为在网络、队列管理器或服务器故障后自动重新连接。

可使用 MQConnectionFactory 类的 WMQ_CONNECTION_NAME_LIST 和 WMQ_CLIENT_RECONNECT_OPTIONS 属性来将客户机配置为自动重新连接。 自动客户机重新连接功能可在发生连接故障后重新连接客户机，也可以作为一个选项在停止队列管理器后使用。 某些客户机应用程序的设计可能导致其不适合于自动重新连接。

可自动重新连接的客户机连接在建立连接后变为可重新连接。

注: 也可以通过客户机通道定义表 (CCDT) 或通过使用 mqclient.ini 文件启用客户机重新连接功能来设置属性**客户机重新连接选项**、**客户机重新连接超时**和**连接名称列表**。

注: 如果在 ConnectionFactory 对象上以及在 CCDT 中设置了重新连接属性，那么优先规则如下。 如果在 ConnectionFactory 对象上设置了“连接名称列表”属性的缺省值，那么优先采用 CCDT。 如果未将“连接名称列表”属性设置为其缺省值，那么优先采用 ConnectionFactory 对象上设置的属性值。 “连接名称列表”属性的缺省值为 localhost(1414)。

编写 XMS .NET 应用程序

本部分中的主题所提供的信息可帮助您编写 XMS .NET 应用程序。

本部分提供了与编写 XMS .NET 应用程序有关的信息。 有关编写 XMS 应用程序的常规信息，请参阅第 17 页的『编写 XMS 应用程序』。

本部分包含以下主题：

- 第 37 页的『.NET 的数据类型』
- 第 38 页的『.NET 中的受管和非受管操作』
- 第 38 页的『.NET 中的目标』
- 第 39 页的『.NET 中的属性』
- 第 40 页的『在 .NET 中处理不存在的属性』
- 第 40 页的『.NET 中的错误处理』
- 第 40 页的『.NET 中的消息和异常侦听器』

相关概念

编写 XMS 应用程序
本部分中的主题所提供的信息可帮助您编写 XMS 应用程序。

相关参考

.NET 接口
本部分记录了 .NET 类接口及其属性和方法。

.NET 的数据类型

XMS .NET 支持 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 和 System.Object。 XMS .NET 的数据类型与 XMS C/C++ 的数据类型有所不同。您可以使用本主题来标识相应的数据类型。

下表显示了对应的 XMS .NET 和 XMS C/C++ 数据类型及其简要描述。

表 *7: XMS .NET* 和 *XMS C/C++* 的数据类型

XMS .NET 类型	**XMS C/C++ 类型**	**描述**
System.SByte	xmsSBYTE xmsINT8	有符号的 8 位值
System.Byte	xmsBYTE xmsUINT8	无符号的 8 位值
System.Int16	xmsINT16 xmsSHORT	有符号的 16 位值
System.UInt16	xmsUINT16 xmsUSHORT	无符号的 16 位值
System.Int32	xmsINT32 xmsINT	有符号的 32 位值
System.UInt32	xmsUINT32 xmsUINT	无符号的 32 位值
System.Int64	xmsLONG xmsINT64	有符号的 64 位值

表 7: *XMS .NET* 和 *XMS C/C++* 的数据类型 *(继续)*		
XMS .NET 类型	**XMS C/C++ 类型**	**描述**
System.UInt64	xmsULONG xmsUINT64	无符号的 64 位值
System.Char	xmsCHAR16	无符号的 16 位字符（针对 .NET 的 Unicode）
System.Single	xmsFLOAT	IEEE 32 位浮点数
System.Double	xmsDOUBLE	IEEE 64 位浮点数
System.Boolean	xmsBOOL	True/False 值
不适用	xmsCHAR	有符号或无符号的 8 位值（有符号或无符号取决于平台）
System.Decimal	不适用	96 位有符号的整数倍（10^0 到 10^{28}）
System.Object	不适用	所有类型的基础
System.String	不适用	字符串类型

.NET 中的受管和非受管操作

受管代码专门在 .NET 通用语言运行时环境中执行，并且完全依赖于该运行时提供的服务。 如果应用程序的任何部分运行或调用 .NET 通用语言运行时环境外部的服务，那么此应用程序归类为非受管应用程序。

受管 .NET 环境中当前不支持某些高级功能。

如果应用程序需要使用完全受管环境中当前不支持的某些功能，那么可以将应用程序更改为使用非受管环境，而无需对应用程序进行实质性更改。 但请注意，做出此选择后，XMS 堆栈将使用非受管代码。

与 IBM MQ 队列管理器的连接

与 WMQ_CM_CLIENT 的受管连接不支持非 TCP 通信和通道压缩。 但是，可通过非受管连接 (WMQ_CM_CLIENT_UNMANAGED) 来支持这些连接。 有关更多信息，请参阅 开发 .NET 应用程序。

如果通过非受管环境中的受管对象来创建连接工厂，那么必须手动将连接方式的值更改为 XMSC_WMQ_CM_CLIENT_UNMANAGED。

与 WebSphere Application Server 服务集成总线消息传递引擎的连接

对于需要使用 SSL 协议（包括 HTTPS）的与 WebSphere Application Server 服务集成总线消息传递引擎的连接，当前不支持将其用作受管代码。

相关参考

XMSC_WMQ_CONNECTION_MODE

.NET 中的目标

在 .NET 中，可根据协议类型创建目标，而且只能在创建目标的协议类型上使用该目标。

提供了两个用于创建目标的函数：一个用于主题，另一个用于队列：

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

API 中的以下两个对象上提供了这两个函数：

- ISession
- XMSFactoryFactory

在这两种情况下，这些方法都可以接受 URI 样式字符串，它可以包含以下格式的参数：

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

或者，这些方法只能接受目标名称（即不带 topic:// 或 queue:// 前缀且不含参数的名称）。

这意味着将使用以下 URI 样式字符串：

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

将生成与以下目标名称相同的结果：

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

对于 WebSphere Application Server 服务集成总线 JMS，还可以指定简短格式的主题，其中包括 *topicname* 和 *topicspace*，但不能包含参数：

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

.NET 中的属性

.NET 应用程序使用 PropertyContext 接口中的方法来获取和设置对象的属性。

PropertyContext 接口封装了用于获取和设置属性的方法。 以下类直接或间接继承了这些方法：

- BytesMessage
- 连接
- ConnectionFactory
- ConnectionMetaData
- Destination
- MapMessage
- 消息
- MessageConsumer
- MessageProducer
- ObjectMessage
- QueueBrowser
- Session
- StreamMessage
- TextMessage

如果应用程序设置了某个属性的值，那么新值将替换此属性的旧值。

有关 XMS 属性的更多信息，请参阅 第 159 页的『XMS 对象的属性』。

为便于使用，XMS 中的 XMS 属性名称和值已预定义为 XMSC struct 中的公共常量。 这些常量的名称采用 XMSC.*constant* 格式；例如，XMSC.USERID（属性名称常量）和 XMSC.DELIVERY_AS_APP（值常量）。

另外，您可以使用 IBM.XMS.MQC struct 访问 IBM MQ 常量。 如果已导入 IBM.XMS 名称空间，那么可以访问这些属性的值（采用格式 MQC.*constant*）。 例如，MQC.MQRO_COA_WITH_FULL_DATA。

另外，如果您具有使用 XMS .NET 和用于 .NET 的 IBM MQ 类且会导入 IBM.XMS 和 IBM.WMQ 名称空间的混合应用程序，那么必须完全限定 MQC struct 名称空间以确保每次出现都是唯一的。

受管 .NET 环境中当前不支持某些高级功能。 请参阅 第 38 页的『.NET 中的受管和非受管操作』，以获取更多详细信息。

在 .NET 中处理不存在的属性

XMS .NET 中针对不存在属性的处理方式与 JMS 规范大体一致，也与 XMS 的 C 和 C++ 实现一致。

在 JMS 中，当方法尝试将不存在的（空）值转换为所需类型时，访问不存在的属性可能会导致 Java 系统异常。 如果属性不存在，那么会发生以下异常：

- getStringProperty 和 getObjectProperty 返回空值
- getBooleanProperty 返回 false，因为 Boolean.valueOf(null) 返回 false
- getIntProperty 等抛出 java.lang.NumberFormatException，因为 Integer.valueOf(null) 抛出异常

如果属性不存在于 XMS .NET 中，那么会发生以下异常：

- GetStringProperty 和 GetObjectProperty（和 GetBytesProperty）返回空值（与 Java 相同）
- GetBooleanProperty 抛出 System.NullReferenceException
- GetIntProperty 等抛出 System.NullReferenceException

此实现与 Java 不同，但与 JMS 规范大体一致，并且与 XMS C 和 C++ 接口一致。 与 Java 实现相似，XMS .NET 会将任何异常从 System.Convert 调用传播到调用者。 但与 Java 不同，XMS 会通过将空值传递到系统转换例程来显式抛出 NullReferenceExceptions，而不仅仅使用 .NET 框架的本机行为。 如果应用程序将属性设置为诸如 “abc” 之类的字符串并调用 GetIntProperty，那么 Convert.ToInt32("abc") 抛出的 System.FormatException 将传播到调用者（这与 Java 一致）。 只有当用于 setProperty 和 getProperty 的类型不兼容时，才会抛出 MessageFormatException。 此行为也与 Java 一致。

.NET 中的错误处理

XMS .NET 异常都来自 System.Exception。XMS 方法调用可以抛出特定 XMS 异常，例如 MessageFormatException、常规 XMSExceptions 或系统异常（例如 NullReferenceException）。

编写应用程序，以根据应用程序需求在特定的 catch 块或常规 `System.Exception` catch 块中捕获所有这些错误。

.NET 中的消息和异常侦听器

.NET 应用程序使用消息侦听器异步接收消息，并使用异常侦听器异步通知连接问题。

对于 .NET 和 C++，消息和异常侦听器的功能是相同的。但是，存在一些小的实现差异。

.NET 中的消息侦听器

要异步接收消息，必须完成以下步骤：

1. 定义与消息侦听器委托的特征符匹配的方法。 您定义的方法可以是静态方法，也可以是实例方法，并且可以在任何可访问类中进行定义。 委托特征符如下所示：

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

   因此，您可以将方法定义为：

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

2. 使用与下面类似的内容将此方法实例化为委托：

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

3. 通过设置为使用者的 MessageListener 属性，向一个或多个使用者注册该委托：

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

您可以通过将 MessageListener 重新设置为 null 来移除该委托：

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

.NET 中的异常侦听器

异常侦听器的工作方式与消息侦听器大致相同，它具有不同的委托定义，并且分配给连接而不是分配给消息使用者。 这与 C++ 相同。

1. 定义方法。 委托特征符如下所示：

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

   因此，定义的方法可以是：

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

2. 使用与下面类似的内容将此方法实例化为委托：

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

3. 通过设置 ExceptionListener 属性来向连接注册该委托：

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

   您可以按如下方式重新设置 ExceptionListener 来移除该委托：

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

如果不存在对异常或消息的引用，那么系统垃圾回收器会自动删除这些异常或消息。

下面是样本代码：

```
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);
    }
}
```

使用受管对象

本部分中的主题提供了有关受管对象的信息。 XMS 应用程序可以从中央受管对象存储库中检索对象定义，并使用这些定义来创建连接工厂和目标。

本部分提供的信息可帮助创建和管理受管对象，并描述了 XMS 支持的受管对象储存库类型。 本部分还说明了 XMS 应用程序如何与受管对象存储库建立连接以检索所需的受管对象。

本部分包含以下主题：

- 第 42 页的『受支持的受管对象存储库类型』
- 第 43 页的『受管对象的属性映射』
- 第 46 页的『受管 ConnectionFactory 对象的必需属性』
- 第 47 页的『受管 Destination 对象的必需属性』
- 第 48 页的『创建受管对象』
- 第 50 页的『InitialContext 对象』
- 第 51 页的『InitialContext 属性』
- 第 51 页的『XMS 初始上下文的 URI 格式』
- 第 53 页的『JNDI 查找 Web Service』
- 第 54 页的『检索受管对象』

相关概念

受管对象

通过使用受管对象，可以在中央存储库中管理您希望管理的客户机应用程序所使用的连接设置。 应用程序从中央存储库中检索对象定义，然后使用这些定义来创建 `ConnectionFactory` 和 `Destination` 对象。 通过使用受管对象，可以将应用程序与其在运行时使用的资源分离开来。

受支持的受管对象存储库类型

“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

“文件系统”对象目录可采用序列化的 Java Naming Directory Interface (JNDI) 对象形式。 LDAP 对象目录是包含 JNDI 对象的目录。 可使用 JMSAdmin 工具（随 IBM WebSphere MQ 6.0 一起提供）或 IBM MQ Explorer（随 IBM WebSphere MQ 7.0 和更高版本一起提供）来管理“文件系统”和 LDAP 对象目录。 可以使用“文件系统”和 LDAP 对象目录，以通过集中 IBM WebSphere MQ 连接工厂和目标来管理客户机连接。 网络管理员可以部署引用了同一中央存储库的多个应用程序，这些应用程序将自动进行更新以反映中央存储库中的连接设置更改。

“COS 命名”目录包含 WebSphere Application Server service integration bus 连接工厂和目标，并且可使用 WebSphere Application Server 管理控制台进行管理。 如果 XMS 应用程序需要从“COS 命名”目录中检索对象，那么必须部署 JNDI 查找 Web Service。 此 Web Service 并不适用于所有 WebSphere Application Server service integration technologies。 请参阅产品文档以了解详细信息。

注: 重新启动应用程序连接，使该对象目录的更改生效。

相关概念

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

受管对象
通过使用受管对象，可以在中央存储库中管理您希望管理的客户机应用程序所使用的连接设置。 应用程序从中央存储库中检索对象定义，然后使用这些定义来创建 `ConnectionFactory` 和 `Destination` 对象。 通过使用受管对象，可以将应用程序与其在运行时使用的资源分离开来。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

受管对象的属性映射

为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

例如，要使用从 IBM MQ JMS 连接工厂检索的属性来创建 XMS 连接工厂，必须在这二者之间映射这些属性。

将自动执行所有属性映射。

下表展示了连接工厂和目标的部分最常见属性之间的映射。 此表中显示的属性只是一小部分示例，并非所有这些属性都适用于所有连接类型和服务器。

表 *8:* 连接工厂和目标属性的名称映射示例

IBM MQ JMS 属性名称	**XMS 属性名称**	**WebSphere Application Server service integration bus 属性名称**
PERSISTENCE (PER)	XMSC_DELIVERY_MODE	
EXPIRY (EXP)	XMSC_TIME_TO_LIVE	
PRIORITY (PRI)	XMSC_PRIORITY	

表 8: 连接工厂和目标属性的名称映射示例 (继续)		
IBM MQ JMS 属性名称	**XMS 属性名称**	**WebSphere Application Server service integration bus 属性名称**
	XMSC_WPM_HOST_NAME	serverName
	XMSC_WPM_BUS_NAME	busName
	XMSC_WPM_TOPIC_SPACE	topicName

表 9: *XMS .NET* 属性					
属性	**对象类型**				
	CF	**QCF**	**TCF**	**队列**	**Topic**
APPLICATIONNAME	Y	Y	Y	不适用	不适用
ASYNCEXCEPTION	Y	Y	Y	不适用	不适用
CCDTURL	Y	Y	Y	不适用	不适用
CHANNEL	Y	Y	Y	不适用	不适用
CONNECTIONNAMELIST	Y	Y	Y	不适用	不适用
CLIENTRECONNECTOPTIONS	Y	Y	Y	不适用	不适用
CLIENTRECONNECTTIMEOUT	Y	Y	Y	不适用	不适用
CLIENTID	不适用	Y	不适用	不适用	不适用
COMPHDR 第 45 页的『1』	Y	不适用	Y	不适用	不适用
COMPMSG 第 45 页的『1』	Y	Y	Y	不适用	不适用
CONNOPT 第 45 页的『1』	Y	Y	Y	不适用	不适用
CONNTAG 第 45 页的『1』	Y	Y	Y	不适用	不适用
DESCRIPTION 第 45 页的『1』	不适用	Y	不适用	Y	Y
到期第 45 页的『1』	不适用	不适用	不适用	Y	Y
FAILIFQUIESCE	Y	Y	Y	Y	Y
HOSTNAME	不适用	Y	不适用	不适用	不适用
LOCALADDRESS	不适用	Y	不适用	不适用	不适用
持久性	不适用	不适用	不适用	Y	Y
PORT	不适用	Y	不适用	不适用	不适用

表 9: XMS .NET 属性 (继续)					
属性	**对象类型**				
	CF	**QCF**	**TCF**	**队列**	**Topic**
优先级第 45 页的『1』	不适用	不适用	不适用	Y	Y
PROVIDERVERSION 第 45 页的『1』	不适用	Y	不适用	不适用	不适用
QMANAGER	Y	Y	Y	Y	不适用
队列第 45 页的『1』	不适用	不适用	不适用	Y	不适用
SHARECONVALLOWED	Y	Y	Y	不适用	不适用
主题第 45 页的『1』	不适用	不适用	不适用	不适用	Y
Transport 第 45 页的『1』	不适用	Y	不适用	不适用	不适用

注:

1. 这些属性没有应用程序级别属性，但可以选择使用受管属性进行设置。

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

IConnectionFactory（适用于 .NET 接口）
应用程序使用连接工厂来创建连接。

ConnectionFactory 属性
下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

受管 ConnectionFactory 对象的必需属性

应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

以下表格中列出的属性是应用程序与消息传递服务器建立连接的最低设置要求。 如果要定制用于建立连接的方式，那么应用程序可以根据需要来设置 ConnectionFactory 对象的任何其他属性。 有关更多信息，请参阅第 160 页的『ConnectionFactory 属性』。 其中提供了可用属性的完整列表。

与 IBM MQ 队列管理器的连接

表 *10:* 用于与 *IBM MQ* 队列管理器建立连接的受管 *ConnectionFactory* 对象的属性设置

必需 XMS	**等效的必需 IBM MQ JMS 属性**
XMSC_CONNECTION_TYPE	XMS 通过连接工厂类名和 TRANSPORT (TRAN) 属性来确定此项。
XMSC_WMQ_HOST_NAME	HOSTNAME (HOST)
XMSC_WMQ_PORT	端口
XMSC_WMQ_QUEUE_MANAGER	队列管理器的名称

与代理程序的实时连接

表 *11:* 用于与代理程序建立实时连接的受管 *ConnectionFactory* 对象的属性设置

必需 XMS	**等效的必需 IBM MQ JMS 属性**
XMSC_CONNECTION_TYPE	XMS 通过连接工厂类名和 TRANSPORT (TRAN) 属性来确定此项。
XMSC_RTT_HOST_NAME	HOSTNAME (HOST)
XMSC_RTT_PORT	端口

与 WebSphere Application Server service integration bus

表 *12:* 用于与 *WebSphere Application Server service integration bus* 建立连接的受管 *ConnectionFactory* 对象的属性设置

XMS 属性	**描述**
XMSC_CONNECTION_TYPE	与应用程序相连的消息传递服务器的类型。。这可通过连接工厂类名称来确定。
XMSC_WPM_BUS_NAME	对于连接工厂，这是应用程序连接到的服务集成总线的名称；对于目标，这是存在目标的服务集成总线的名称。

相关概念
受支持的受管对象存储库类型

“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

与 IBM MQ 队列管理器的安全连接
要使 XMS .NET 应用程序能够与 IBM MQ 队列管理器建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

与 WebSphere Application Server service integration bus 消息传递引擎的安全连接
要使 XMS .NET 应用程序能够与 WebSphere Application Server service integration bus 消息传递引擎建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

相关任务
创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考
受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

IConnectionFactory（适用于 .NET 接口）
应用程序使用连接工厂来创建连接。

ConnectionFactory 属性
下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

受管 Destination 对象的必需属性

用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

表 *13:* 受管 *Destination* 对象的属性设置

连接类型	**属性**	**描述**
IBM MQ 队列管理器	QUEUE (QU) TOPIC (TOP)	要连接到的队列 应用程序用作目标的主题
与代理程序的实时连接	TOPIC (TOP)	应用程序用作目标的主题
WebSphere Application Server service integration bus	topicName queueName	如果应用程序正在连接到主题 如果应用程序正在连接到队列

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

创建受管对象

必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

开始之前

有关 XMS 支持的不同类型的受管对象存储库的更多详细信息，请参阅第 42 页的『受支持的受管对象存储库类型』。

关于此任务

要创建 IBM MQ 的受管对象，请使用 IBM MQ Explorer 或 IBM MQ JMS 管理 (JMSAdmin) 工具。

要创建 IBM MQ 或 IBM Integration Bus 的受管对象，请使用 IBM MQ JMS 管理 (JMSAdmin) 工具。

要创建 WebSphere Application Server service integration bus 的受管对象，请使用 WebSphere Application Server 管理控制台。

以下步骤概括了在创建受管对象时要执行的操作。

过程

1. 创建连接工厂并定义必需的属性，以创建应用程序到所选服务器的连接。
   第 46 页的『受管 ConnectionFactory 对象的必需属性』中定义了 XMS 建立连接时所需的最少属性。
2. 在应用程序连接到的消息传递服务器上创建必需的目标：
   - 对于与 IBM MQ 队列管理器的连接，请创建队列或主题。
   - 对于与代理程序的实时连接，请创建主题。
   - 对于与 WebSphere Application Server service integration bus 的连接，请创建队列或主题。

   第 47 页的『受管 Destination 对象的必需属性』中定义了 XMS 建立连接时所需的最少属性。

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

受管对象
通过使用受管对象，可以在中央存储库中管理您希望管理的客户机应用程序所使用的连接设置。 应用程序从中央存储库中检索对象定义，然后使用这些定义来创建 `ConnectionFactory` 和 `Destination` 对象。 通过使用受管对象，可以将应用程序与其在运行时使用的资源分离开来。

ConnectionFactories 和 Connection 对象
ConnectionFactory 对象提供一种模板以供应用程序用于创建 Connection 对象。 应用程序可使用 Connection 对象来创建 Session 对象。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

相关任务

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

IConnectionFactory（适用于 .NET 接口）
应用程序使用连接工厂来创建连接。

ConnectionFactory 属性

下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

IDestination（适用于 .NET 接口）
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

Destination 属性
下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

InitialContext 对象

应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

关于此任务

InitialContext 对象封装了与该存储库的连接。 XMS API 提供了用于执行以下任务的方法：

- 创建 InitialContext 对象
- 在受管对象存储库中查找受管对象。

有关创建 InitialContext 对象的更多详细信息，请参阅 第 94 页的『InitialContext』 for .NET 和 第 167 页的『InitialContext 属性』。

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

InitialContext（适用于 .NET 接口）
应用程序使用 InitialContext 对象，通过从受管对象存储库中检索的对象定义来创建对象。

InitialContext 属性
下面概括了 InitialContext 对象属性，并提供了指向更详细参考信息的链接。

InitialContext 属性

InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

在 JNDI 中以及在 XMS 的 .NET 实施中，通过环境散列表向该构造函数提供其他信息。

受管对象存储库的位置是在 XMSC_IC_URL 属性中定义的。 通常会将该属性传递给 Create 调用，但可以修改该属性以在查找之前连接到其他命名目录。 对于 FileSystem 或 LDAP 上下文，该属性将定义该目录的地址。 对于 COS 命名，这是使用这些属性连接到 JNDI 目录的 Web Service 的地址。

以下属性在不做修改的情况下传递给 Web Service，后者将使用这些属性来连接到 JNDI 目录。

- XMSC_IC_PROVIDER_URL
- XMSC_IC_SECURITY_CREDENTIALS
- XMSC_IC_SECURITY_AUTHENTICATION
- XMSC_IC_SECURITY_PRINCIPAL
- XMSC_IC_SECURITY_PROTOCOL

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

InitialContext（适用于 .NET 接口）
应用程序使用 InitialContext 对象，通过从受管对象存储库中检索的对象定义来创建对象。

InitialContext 属性
下面概括了 InitialContext 对象属性，并提供了指向更详细参考信息的链接。

XMS 初始上下文的 URI 格式

将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

FileSystem 上下文

对于 FileSystem 上下文，URL 提供基于文件系统的目录的位置。 该 URL 的结构由 RFC 1738 统一资源定位符 *(URL)*定义: 该 URL 具有前缀 `file://`，此前缀后面的语法是可以在运行 XMS 的系统上打开的文件的有效定义。

该语法可特定于平台，并且可使用“/”或“\”分隔符。 如果使用“\”，那么需要使用额外的“\”来转义每个分隔符。 这可防止 .NET 框架尝试将分隔符解释为其后面内容的转义字符。

以下示例说明了该语法:

```
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 上下文

对于 LDAP 上下文，URL 的基本结构由 RFC 2255“*LDAP URL* 格式”定义，它具有不区分大小写的前缀 `ldap://`。

以下示例说明了精确的语法:

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

此语法由该 RFC 定义，但不支持任何属性、作用域、过滤器或扩展。

此语法的示例包括:

```
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 上下文

对于 WSS 上下文，URL 采用 Web Service 端点形式，它具有前缀 `http://`。

也可以使用前缀 `cosnaming://` 或 `wsvc://`。

这两个前缀可解释为以下含义：您正在使用可通过 http 访问其 URL 的 WSS 上下文，这样便可直接从 URL 轻松派生出初始上下文类型。

此语法的示例包括:

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

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

JNDI 查找 Web Service
要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

InitialContext（适用于 .NET 接口）
应用程序使用 InitialContext 对象，通过从受管对象存储库中检索的对象定义来创建对象。

InitialContext 属性
下面概括了 InitialContext 对象属性，并提供了指向更详细参考信息的链接。

JNDI 查找 Web Service

要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

该 Web Service 以企业归档文件 SIBXJndiLookupEAR.ear（位于安装目录中）形式提供。 对于 IBM Message Service Client for .NET 的当前发行版，可以在 *install_dir*\java\lib 目录中找到 SIBXJndiLookupEAR.ear 。 可使用管理控制台或 wsaadmin 脚本编制工具，将该 Web Service 安装在 WebSphere Application Server service integration bus 服务器中。 请参阅产品文档，以获取有关部署 Web Service 应用程序的更多信息。

要在 XMS 应用程序中定义该 Web Service，只需将 InitialContext 对象的 XMSC_IC_URL 属性设置为 Web Service 端点 URL。 例如，如果在名为 MyServer 的服务器主机上部署该 Web Service，那么 Web Service 端点 URL 示例如下：

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

通过设置 XMSC_IC_URL 属性，InitialContext 查找调用可以在定义的端点上调用该 Web Service，进而从 COS 命名服务中查找所需的受管对象。

.NET 应用程序可以使用该 Web Service。 XMS C、/C++ 和 XMS .NET 的服务器端部署相同。XMS .NET 直接通过 Microsoft .NET 框架调用 Web Service。

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性

InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

设置消息传递服务器环境
本部分中的主题描述了如何设置消息传递服务器环境以允许 XMS 应用程序连接到服务器。

使用 XMS 样本应用程序
通过使用 XMS 随附的样本应用程序，可以验证安装和消息传递服务器设置，并帮助构建您自己的应用程序。 这些样本概括了每个 API 的常用功能。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

使用安装向导安装 Message Service Client for .NET
安装将使用 InstallShield X/Windows MSI 安装程序。 提供了两个安装选项，因此您可以选择完整安装或定制安装。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

检索受管对象

XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

要检索的对象可能具有以下类型的名称：

- 用于描述 Destination 对象的简单名称（例如，名为 SalesOrders 的队列目标）
- 复合名称（可能由 SubContext 构成，用“/”分隔，且必须以对象名结尾）。 复合名称的示例是“Warehouse/PickLists/DispatchQueue2”（其中 Warehouse 和 Picklists 是命名目录中的 SubContext，DispatchQueue2 是 Destination 对象的名称）。

相关概念

受支持的受管对象存储库类型
“文件系统”和 LDAP 受管对象可用于连接到 IBM MQ 和 WebSphere Application Server，而“COS 命名”只能用于连接到 WebSphere Application Server。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

JNDI 查找 Web Service

要从 XMS 访问“COS 命名”目录，必须在 WebSphere Application Server service integration bus 服务器上部署 JNDI 查找 Web Service。 此 Web Service 可将来自 COS 命名服务的 Java 信息转换为 XMS 应用程序可读格式。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

InitialContext（适用于 .NET 接口）
应用程序使用 InitialContext 对象，通过从受管对象存储库中检索的对象定义来创建对象。

InitialContext 属性
下面概括了 InitialContext 对象属性，并提供了指向更详细参考信息的链接。

确保 XMS 应用程序安全通信

本部分介绍了如何设置安全通信以使 XMS 应用程序能够通过安全套接字层 (SSL) 连接到 WebSphere Application Server service integration bus 消息传递引擎或 IBM MQ 队列管理器。

本部分包含以下主题：

- 第 55 页的『与 IBM MQ 队列管理器的安全连接』
- 第 56 页的『与 IBM MQ 队列管理器的连接的 CipherSuite 和 CipherSpec 名称映射』
- 第 56 页的『与 WebSphere Application Server service integration bus 消息传递引擎的安全连接』
- 第 57 页的『与 WebSphere Application Server service integration bus 的连接的 CipherSuite 和 CipherSpec 名称映射』

与 IBM MQ 队列管理器的安全连接

要使 XMS .NET 应用程序能够与 IBM MQ 队列管理器建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

在加密协商中使用的协议可以是安全套接字层 (SSL) 或传输层安全性 (TLS)，这取决于您在 ConnectionFactory 对象中指定的 CipherSuite。

如果您使用 IBM WebSphere MQ 7.0.0 Fix Pack 1 和更高版本的客户机库并连接到 IBM WebSphere MQ 7.0 队列管理器，那么可以在 XMS 应用程序中与同一队列管理器建立多个连接。 但是，不允许与不同的队列管理器建立连接。 如果您尝试这样做，那么会收到 `MQRC_SSL_ALREADY_INITIALIZED` 错误。

如果您使用 IBM WebSphere MQ 6.0 和更高版本的客户机库，那么只有在先关闭任何先前的 SSL 连接时才可以创建 SSL 连接。 不允许在同一个进程中与相同或不同的队列管理器建立多个并发 SSL 连接。 如果您尝试多个请求，那么会收到警告 `MQRC_SSL_ALREADY_INITIALIZED`，这意味着已忽略 SSL 连接的某些请求参数。

下表显示了通过 SSL 与 IBM MQ 队列管理器建立的连接的 ConnectionFactory 属性以及简短描述。

表 *14:* 通过 *SSL* 与 *IBM MQ* 队列管理器建立的连接的 *ConnectionFactory* 属性

属性的名称	**描述**
XMSC_WMQ_SSL_CERT_STORES	用于保存与队列管理器的 SSL 连接上使用的证书撤销列表 (CRL) 的服务器的位置。

表 *14:* 通过 *SSL* 与 *IBM MQ* 队列管理器建立的连接的 *ConnectionFactory* 属性 *(*继续*)*	
属性的名称	**描述**
XMSC_WMQ_SSL_CIPHER_SPEC	到队列管理器的安全连接上要使用的 CipherSpec 名称。
XMSC_WMQ_SSL_CIPHER_SUITE	到队列管理器的 TLS 连接上要使用的 CipherSuite 的名称。协商安全连接时使用的协议取决于指定的 CipherSuite。
XMSC_WMQ_SSL_CRYPTO_HW	下面是连接到客户机系统的加密硬件的配置详细信息。
XMSC_WMQ_SSL_FIPS_REQUIRED	该属性的值用于确定应用程序能否使用符合非 FIPS 标准的密码套件。 如果将该属性设置为 true，那么客户机/服务器连接只能使用 FIPS 算法。
XMSC_WMQ_SSL_KEY_REPOSITORY	用于存储密钥和证书的密钥数据库文件的位置。
XMSC_WMQ_SSL_KEY_RESETCOUNT	KeyResetCount 表示在重新协商密钥之前在 SSL 对话期间发送和接收的未加密字节总数。
XMSC_WMQ_SSL_PEER_NAME	到队列管理器的 SSL 连接上要使用的对等方名称。

相关参考

IConnectionFactory（适用于 .NET 接口）
应用程序使用连接工厂来创建连接。

ConnectionFactory 属性
下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

与 *IBM MQ* 队列管理器的连接的 *CipherSuite* 和 *CipherSpec* 名称映射

InitialContext 可在 JMSAdmin 连接工厂属性 SSLCIPHERSUITE 与 XMS 近似等效项 XMSC_WMQ_SSL_CIPHER_SPEC 之间进行转换。 如果指定了 XMSC_WMQ_SSL_CIPHER_SUITE 的值但省略了 XMSC_WMQ_SSL_CIPHER_SPEC 的值，那么需要进行类似转换。

第 56 页的表 15 列出了可用的 CipherSpec 及其 JSSE CipherSuite 等效项。

表 *15:* 可用的 *CipherSpec* 及其 *JSSE CipherSuite* 等效项	
CipherSpec	**等效的 JSSE CipherSuite**
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA

注:

- 不推荐 TLS_RSA_WITH_3DES_EDE_CBC_SHA。 但是，它仍可用于传输最多 32 GB 数据，超过此数据量之后，连接将因错误 AMQ9288 而终止。 要避免此错误，您需要避免使用三重 DES，或在使用此 CipherSpec 时启用密钥重置。

与 WebSphere Application Server service integration bus 消息传递引擎的安全连接

要使 XMS .NET 应用程序能够与 WebSphere Application Server service integration bus 消息传递引擎建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

针对与 WebSphere Service Integration Bus 的连接，XMS 提供 SSL 和 HTTPS 支持。 SSL 和 HTTPS 提供安全连接以便进行认证和确保机密性。

与 WebSphere 安全性相似，XMS 安全性是根据 JSSE 安全标准和命名约定进行配置，包括使用 CipherSuite 指定在协商安全连接时使用的算法。 在加密协商中使用的协议可以是 SSL 或 TLS，这取决于您在 ConnectionFactory 对象中指定的 CipherSuite。

第 57 页的表 16 列出了必须在 ConnectionFactory 对象中定义的属性。

表 *16:* 与 *WebSphere Application Server service integration bus* 消息传递引擎的安全连接的 *ConnectionFactory* 属性

属性的名称	描述
XMSC_WPM_SSL_CIPHER_SUITE	到 WebSphere Service Integration Bus 消息传递引擎的 TLS 连接上要使用的 CipherSuite 名称。 协商安全连接时使用的协议取决于指定的 CipherSuite。
XMSC_WPM_SSL_KEYRING_LABEL	向服务器进行认证时要使用的证书。

下面是与 WebSphere Application Server service integration bus 消息传递引擎的安全连接的 ConnectionFactory 属性示例：

```
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);
```

其中，应将 chain_name 设置为 BootstrapTunneledSecureMessaging 或 BootstrapSecureMessaging，port_number 是引导程序服务器侦听入局请求时所使用的端口号。

下面是与 WebSphere Application Server service integration bus 消息传递引擎的安全连接的 ConnectionFactory 属性示例（已插入示例值）：

```
  /* 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");
```

相关参考

IConnectionFactory（适用于 .NET 接口）
应用程序使用连接工厂来创建连接。

ConnectionFactory 属性
下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

与 WebSphere Application Server service integration bus 的连接的 CipherSuite 和 CipherSpec 名称映射

由于 GSKit 使用 CipherSpec（而不是 CipherSuite），因此必须将 XMSC_WPM_SSL_CIPHER_SUITE 属性中指定的 JSSE 样式的 CipherSuite 名称映射到 GSKit 样式的 CipherSpec 名称。

第 57 页的表 17 列出了每个公认 CipherSuite 的等效 CipherSpec。

表 *17:* 可用的 *CipherSuite* 及其等效的 *CipherSpec*

CipherSuite	CipherSpec 等效项
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

表 17: 可用的 *CipherSuite* 及其等效的 *CipherSpec (继续)*	
CipherSuite	**CipherSpec 等效项**
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

注:

- 不推荐 TLS_RSA_WITH_3DES_EDE_CBC_SHA。 但是，它仍可用于传输最多 32 GB 数据，超过此数据量之后，连接将因错误 AMQ9288 而终止。 要避免此错误，您需要避免使用三重 DES，或在使用此 CipherSpec 时启用密钥重置。

XMS 消息

本部分描述了 XMS 消息的结构和内容，并解释了应用程序如何处理 XMS 消息。

本部分包含以下主题:

- 第 58 页的『XMS 消息的组成部分』
- 第 59 页的『XMS 消息中的头字段』
- 第 59 页的『XMS 消息的属性』
- 第 62 页的『XMS 消息的主体』
- 第 67 页的『消息选择器』
- 第 68 页的『将 XMS 消息映射到 IBM MQ 消息』

相关参考

IMessage（适用于 .NET 接口）
Message 对象表示应用程序发送或接收的消息。 IMessage 是诸如 IMapMessage 等消息类的超类。

XMS 消息的组成部分

XMS 消息由一个头、一组属性和一个主体组成。

头
消息的头包含字段，并且所有消息都包含同一组头字段。 XMS 和应用程序使用头字段的值来识别和传递消息。 有关头字段的更多信息，请参阅 第 59 页的『XMS 消息中的头字段』。

属性集
消息的属性指定有关消息的其他信息。 虽然所有消息都具有同一组头字段，但是每条消息可以具有一组不同的属性。 有关更多信息，请参阅 第 59 页的『XMS 消息的属性』。

正文
消息的主体包含应用程序数据。 有关更多信息，请参阅 第 62 页的『XMS 消息的主体』。

应用程序可以选择要接收的消息。 使用可指定选择条件的消息选择器。 条件可以基于某些头字段的值以及消息的任何属性的值。 有关消息选择器的更多信息，请参阅第 67 页的『消息选择器』。

相关参考

XMS 消息中的头字段
要允许 XMS 应用程序与 WebSphere JMS 应用程序交换消息，XMS 消息的头包含 JMS 消息头字段。

XMS 消息的属性
XMS 支持三类消息属性：JMS 定义的属性、IBM 定义的属性和应用程序定义的属性。

XMS 消息的主体
消息的主体包含应用程序数据。 但是，消息也可以没有主体，而是只包含头字段和属性。

消息选择器
XMS 应用程序使用消息选择器来选择要接收的消息。

将 XMS 消息映射到 IBM MQ 消息
将 XMS 消息的 JMS 头字段和属性映射到 IBM MQ 消息的头结构中的字段。

XMS 消息中的头字段

要允许 XMS 应用程序与 WebSphere JMS 应用程序交换消息，XMS 消息的头包含 JMS 消息头字段。

这些头字段的名称以前缀 JMS 开头。 有关 JMS 消息头字段的描述，请参阅 *Java Message Service* 规范。

XMS 会将 JMS 消息头字段实现为 Message 对象的属性。 每个头字段都有自己的用于设置和获取其值的方法。 有关这些方法的描述，请参阅 第 106 页的『IMessage』。 头字段始终可读且可写。

第 59 页的表 18 列出了 JMS 消息头字段，并指示如何为传输的消息设置每个字段的值。 在应用程序发送消息或者在 JMSRedelivered 情况下应用程序接收消息时，XMS 会自动设置其中某些字段。

表 *18: JMS* 消息头字段

JMS 消息头字段的名称	**如何为传输的消息设置值（采用格式 *method* [*class*]）**
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]

相关参考

XMS 消息的组成部分
XMS 消息由一个头、一组属性和一个主体组成。

XMS 消息的属性
XMS 支持三类消息属性：JMS 定义的属性、IBM 定义的属性和应用程序定义的属性。

XMS 消息的主体
消息的主体包含应用程序数据。 但是，消息也可以没有主体，而是只包含头字段和属性。

消息选择器
XMS 应用程序使用消息选择器来选择要接收的消息。

将 XMS 消息映射到 IBM MQ 消息
将 XMS 消息的 JMS 头字段和属性映射到 IBM MQ 消息的头结构中的字段。

XMS 消息的属性

XMS 支持三类消息属性：JMS 定义的属性、IBM 定义的属性和应用程序定义的属性。

XMS 应用程序可以与 WebSphere JMS 应用程序交换消息，因为 XMS 支持 Message 对象的下列预定义属性：

- 受 WebSphere JMS 支持的 JMS 定义的属性。 这些属性的名称以前缀 JMSX 开头。
- 受 WebSphere JMS 支持的 IBM 定义的属性。 这些属性的名称以前缀 JMS_IBM_ 开头。

每个预定义属性都具有两个名称：

- JMS 名称（针对 JMS 定义的属性）或 WebSphere JMS 名称（针对 IBM 定义的属性）。

  这是在 JMS 或 WebSphere JMS 中用于标识属性的名称，也是与具有此属性的消息一起传输的名称。 XMS 应用程序使用此名称识别消息选择器表达式中的属性。

- 在所有情况（消息选择器表达式除外）下用于标识属性的 XMS 名称。 每个 XMS 名称都定义为 `IBM.XMS.XMSC` 类中的命名常量。 命名常量的值是对应的 JMS 或 WebSphere JMS 名称。

除了预定义的属性外，XMS 应用程序还可以创建和使用自己的消息属性集。 这些属性称为应用程序定义的属性。

在应用程序创建消息之后，消息属性为可读且可写。 在应用程序发送消息之后，这些属性仍然可读且可写。 在应用程序接收消息时，消息属性为只读。 如果应用程序在消息的属性为只读时调用 Message 类的 `Clear Properties` 方法，那么这些属性将变为可读且可写。 该方法还会清除这些属性。

在清除消息属性后转发已收到消息的行为方式与转发已清除消息属性的 .NET BytesMessage 的标准 WMQ XMS 的行为方式一致。

但是，由于将会丢失以下属性，所以建议不要这样做：

- JMS_IBM_Encoding 属性值，意味着无法采用有意义的方式解码消息数据。
- JMS_IBM_Format 属性值，意味着（MQMD 或新的 MQRFH2）消息头与现有头之间的头链将断开。

要确定消息的所有属性的值，应用程序可以调用 Message 类的 Get Properties 方法。 该方法将创建用于封装 Property 对象列表的迭代器，其中每个 Property 对象都表示消息的一个属性。 然后，应用程序可以使用 Iterator 类的方法来依次检索各个 Property 对象，它可以使用 Property 类的方法来检索每个属性的名称、数据类型和值。

相关参考

XMS 消息的组成部分
XMS 消息由一个头、一组属性和一个主体组成。

XMS 消息中的头字段
要允许 XMS 应用程序与 WebSphere JMS 应用程序交换消息，XMS 消息的头包含 JMS 消息头字段。

XMS 消息的主体
消息的主体包含应用程序数据。 但是，消息也可以没有主体，而是只包含头字段和属性。

消息选择器
XMS 应用程序使用消息选择器来选择要接收的消息。

将 XMS 消息映射到 IBM MQ 消息
将 XMS 消息的 JMS 头字段和属性映射到 IBM MQ 消息的头结构中的字段。

消息的 *JMS* 定义的属性

XMS 和 WebSphere JMS 支持消息的多个 JMS 定义的属性。

第 60 页的表 19 列出了 XMS 和 WebSphere JMS 支持的消息的 JMS 定义的属性。 有关 JMS 定义的属性的描述，请参阅 *Java Message Service* 规范。 JMS 定义的属性不适用于与代理程序的实时连接。

该表列出了每个属性的数据类型，并指示如何为传输的消息设置此属性的值。 当应用程序发送消息或者在 JMSXDeliveryCount 情况下应用程序接收消息时，XMS 会自动设置其中某些属性。

表 *19:* 消息的 *JMS* 定义的属性

JMS 定义的属性的 XMS 名称	**JMS 名称**	**数据类型**	**如何为传输的消息设置值（采用格式 *method* [*class*]）**
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]

消息的 *IBM* 定义的属性

XMS 和 WebSphere JMS 支持消息的多个 IBM 定义的属性。

第 61 页的表 20 列出了 XMS 和 WebSphere JMS 支持的消息的 IBM 定义的属性。 有关 IBM 定义的属性的更多信息，请参阅 IBM MQ 或 WebSphere Application Server 产品文档。

该表列出了每个属性的数据类型，并指示如何为传输的消息设置此属性的值。 应用程序发送消息时，XMS 会自动设置其中某些属性。

表 *20:* 消息的 *IBM* 定义的属性

IBM 定义的属性的 XMS 名称	**WebSphere JMS 名称**	**数据类型**	**如何为传输的消息设置值（采用格式 *method* [*class*]）**
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_EXCEPTIONMESSAGE	JMS_IBM_ExceptionMessage	System.String	Receive [MessageConsumer]
JMS_IBM_EXCEPTIONREASON	JMS_IBM_ExceptionReason	System.Int32	Receive [MessageConsumer]
JMS_IBM_EXCEPTIONTIMESTAMP	JMS_IBM_ExceptionTimestamp	System.Int64	Receive [MessageConsumer]
JMS_IBM_EXCEPTIONPROBLEM 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]

表 20: 消息的 IBM 定义的属性 (继续)

IBM 定义的属性的 XMS 名称	WebSphere JMS 名称	数据类型	如何为传输的消息设置值（采用格式 *method* [*class*]）
JMS_IBM_REPORT_EXPIRATION	JMS_IBM_Report_Expiration	System.Int32	Set Integer Property [PropertyContext]
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_ 标识	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]

消息的应用程序定义的属性

XMS 应用程序可以创建和使用自己的消息属性集。 在应用程序发送消息时，这些属性也随该消息一起传输。 然后，接收应用程序可以根据这些属性的值，使用消息选择器来选择要接收的消息。

要允许 WebSphere JMS 应用程序选择和处理 XMS 应用程序发送的消息，应用程序定义的属性的名称必须遵循在消息选择器表达式中构成标识的相关规则，如 IBM MQ 产品文档中所述。 应用程序定义的属性的值必须具有以下某种数据类型：System.Boolean、System.SByte、System.Int16、System.Int32、System.Int64、System.Float、System.Double 或 System.String。

XMS 消息的主体

消息的主体包含应用程序数据。 但是，消息也可以没有主体，而是只包含头字段和属性。

XMS 支持以下五种类型的消息体：

字节
主体包含字节流。 具有此类型主体的消息称为字节消息。 IBytesMessage 接口包含用于处理字节消息体的方法。 有关更多信息，请参阅第 64 页的『字节消息』。

映射
主体包含一组名称/值对，其中每个值都具有相关的数据类型。 具有此类型主体的消息称为映射消息。 IMapMessage 接口包含用于处理映射消息体的方法。 有关更多信息，请参阅第 64 页的『映射消息』。

Object
主体包含序列化的 Java 或 .NET 对象。 具有此类型主体的消息称为对象消息。 IObjectMessage 接口包含用于处理对象消息体的方法。 有关更多信息，请参阅第 65 页的『对象消息』。

流
主体包含值流，其中每个值都具有相关的数据类型。 具有此类型主体的消息称为流消息。 IStreamMessage 接口包含用于处理流消息体的方法。 有关更多信息，请参阅第 65 页的『流消息』。

文本
主体包含字符串。 具有此类型主体的消息称为文本消息。 ITextMessage 接口包含用于处理文本消息体的方法。 有关更多信息，请参阅第 66 页的『文本消息』。

IMessage 接口是所有 Message 对象的父项，可以在消息传递函数中用于表示任何 XMS 消息类型。

有关所有这些数据类型的大小以及最大值和最小值的信息，请参阅 第 33 页的表 5。

相关参考

XMS 消息的组成部分
XMS 消息由一个头、一组属性和一个主体组成。

XMS 消息中的头字段
要允许 XMS 应用程序与 WebSphere JMS 应用程序交换消息，XMS 消息的头包含 JMS 消息头字段。

XMS 消息的属性
XMS 支持三类消息属性：JMS 定义的属性、IBM 定义的属性和应用程序定义的属性。

消息选择器
XMS 应用程序使用消息选择器来选择要接收的消息。

将 XMS 消息映射到 IBM MQ 消息
将 XMS 消息的 JMS 头字段和属性映射到 IBM MQ 消息的头结构中的字段。

应用程序数据元素的数据类型

为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

由于这个原因，XMS 应用程序在消息体中写入的应用程序数据的每个元素都必须具有第 63 页的表 21 中所列的一种数据类型。 对于每种数据类型，该表显示了兼容的 Java 数据类型。 XMS 提供了仅使用这些数据类型来写入应用程序数据元素的方法。

表 *21: 与 Java* 数据类型兼容的 *XMS* 数据类型

XMS 数据类型	**表示**	**与 Java 兼容的数据类型**
System.Boolean	布尔值 true 或 false	布尔值
System.Char16	双字节字符	char
System.SByte	有符号的 8 位整数	字节
System.Int16	有符号的 16 位整数	短整型
System.Int32	有符号的 32 位整数	int
System.Int64	有符号的 64 位整数	long
System.Float	有符号的浮点数	浮点值
System.Double	有符号的双精度浮点数	双精度值
System.String	字符串	字符串

有关所有这些数据类型的大小以及最大值和最小值的信息，请参阅 第 33 页的『XMS 基本类型』。

相关概念

对象的属性和特性
XMS 对象可具有属性 (attribute) 和特性 (property)，它们是对象的特征，可采用不同的方式来实现。

XMS 基本类型
XMS 提供了 8 种 Java 基本类型（byte、short、int、long、float、double、char 和 boolean）的等效项。这样便可以在 XMS 与 JMS 之间交换消息而不丢失或损坏数据。

属性值从一种数据类型到另一种数据类型的隐式转换
在应用程序获取某个属性的值后，XMS 可以将该值转换为另一种数据类型。 许多规则可控制受支持的转换以及 XMS 执行转换的方式。

相关参考

字节消息
字节消息的主体包含一连串字节。 主体仅包含实际数据，由发送和接收应用程序负责解释这些数据。

映射消息
映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

对象消息
对象消息的主体包含序列化的 Java 或 .NET 对象。

流消息
流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

文本消息
文本消息的主体包含一个字符串。

字节消息

字节消息的主体包含一连串字节。 主体仅包含实际数据，由发送和接收应用程序负责解释这些数据。

如果 XMS 应用程序需要与不使用 XMS 或 JMS 应用程序编程接口的应用程序交换消息，那么字节消息将很有用。

在应用程序创建字节消息之后，消息体为只写。 应用程序通过对 .NET 调用 IBytesMessage 接口的相应写方法，将应用程序数据组合到主体中。 每次应用程序将值写入字节消息流时，都会直接在应用程序写入的前一个值后面组合该值。 XMS 会维护一个内部光标来记住组合的最后一个字节的位置。

在应用程序发送消息时，消息体将变为只读。 在此方式下，应用程序可以重复发送消息。

在应用程序接收字节消息时，消息体为只读。 应用程序可以使用 IBytesMessage 接口的相应读方法来读取字节消息流的内容。 应用程序将按顺序读取字节，并且 XMS 会维护一个内部光标来记住读取的最后一个字节的位置。

如果应用程序在字节消息体为可写时调用 IBytesMessage 接口的 Reset 方法，该主体将变为只读。 该方法还会将光标重新定位到字节消息流的开头。

如果应用程序在字节消息体为只读时调用 .NET IMessage 接口的 `Clear Body` 方法，该主体将变为可写。 该方法还会清除主体。

相关参考

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

映射消息
映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

对象消息
对象消息的主体包含序列化的 Java 或 .NET 对象。

流消息
流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

文本消息
文本消息的主体包含一个字符串。

IBytesMessage（适用于 .NET 接口）
字节消息是其主体由字节流组成的消息。

映射消息

映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

在每个名称/值对中，名称是用于标识值的字符串，值是具有第 63 页的表 21 中所列的一种 XMS 数据类型的应用程序数据元素。 未定义名称/值对的顺序。 MapMessage 类包含用于设置和获取名称/值对的方法。

应用程序可以通过指定名称来随机访问名称/值对。

.NET 应用程序可以使用 MapNames 属性来获取映射消息体中名称的枚举。

在应用程序获取名称/值对的值后，XMS 可以将该值转换为另一种数据类型。 例如，要从映射消息体中获取某个整数，应用程序可以调用 MapMessage 类的 GetString 方法，以将该整数作为字符串返回。 受支持的转换与 XMS 将属性值从一种数据类型转换为另一种数据类型时支持的转换相同。 有关受支持的转换的更多信息，请参阅第 33 页的『属性值从一种数据类型到另一种数据类型的隐式转换』。

在应用程序创建映射消息之后，消息体为可读且可写。 在应用程序发送消息之后，主体仍然可读且可写。 在应用程序接收映射消息时，消息体为只读。 如果应用程序在映射消息体为只读时调用 Message 类的 Clear Body 方法，该主体将变为可读且可写。 该方法还会清除主体。

相关概念

属性值从一种数据类型到另一种数据类型的隐式转换
在应用程序获取某个属性的值后，XMS 可以将该值转换为另一种数据类型。 许多规则可控制受支持的转换以及 XMS 执行转换的方式。

相关参考

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

字节消息
字节消息的主体包含一连串字节。 主体仅包含实际数据，由发送和接收应用程序负责解释这些数据。

对象消息
对象消息的主体包含序列化的 Java 或 .NET 对象。

流消息
流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

文本消息
文本消息的主体包含一个字符串。

IMapMessage（适用于 .NET 接口）
映射消息是消息主体由一组名称/值对组成的消息，其中每个值都有关联的数据类型。

对象消息

对象消息的主体包含序列化的 Java 或 .NET 对象。

XMS 应用程序可以接收对象消息，更改其头字段和属性，然后将其发送到其他目标。 应用程序也可以复制对象消息的主体，并使用它来构成其他对象消息。 XMS 会将对象消息的主体视为字节数组。

在应用程序创建对象消息之后，消息体为可读且可写。 在应用程序发送消息之后，主体仍然可读且可写。 在应用程序接收对象消息时，消息体为只读。 如果应用程序在对象消息的主体为只读时调用 .NET 的 IMessage 接口的 `Clear Body` 方法，那么该主体将变为可读且可写。 该方法还会清除主体。

相关参考

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

字节消息
字节消息的主体包含一连串字节。 主体仅包含实际数据，由发送和接收应用程序负责解释这些数据。

映射消息
映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

流消息
流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

文本消息
文本消息的主体包含一个字符串。

IObjectMessage（适用于 .NET 接口）
对象消息是其主体包含序列化 Java 或 .NET 对象的消息。

流消息

流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

值的数据类型是第 63 页的表 21 中所列的一种 XMS 数据类型。

在应用程序创建流消息之后，消息体为可写。 应用程序通过调用 .NET IStreamMessage 接口的相应写方法，将应用程序数据组合到主体中。 每次应用程序将值写入消息流时，都会直接在应用程序写入的前一个值后面组合该值及其数据类型。 XMS 会维护一个内部光标来记住组合的最后一个值的位置。

在应用程序发送消息时，消息体将变为只读。 在此方式下，应用程序可以多次发送消息。

在应用程序接收流消息时，消息体为只读。 应用程序可以使用 .NET IStreamMessage 接口的相应读方法来读取消息流的内容。 应用程序将按顺序读取值，并且 XMS 会维护一个内部光标来记住读取的最后一个值的位置。

在应用程序从消息流中读取值时，该值可由 XMS 转换为其他数据类型。 例如，要从消息流中读取某个整数，应用程序可以调用 ReadString 方法以将该整数作为字符串返回。 受支持的转换与 XMS 将属性值从一种数据类型转换为另一种数据类型时支持的转换相同。 有关受支持的转换的更多信息，请参阅第 33 页的『属性值从一种数据类型到另一种数据类型的隐式转换』。

如果在应用程序尝试从消息流中读取值时发生错误，那么不会将光标前移。 应用程序可以通过尝试将值作为另一种数据类型读取来从该错误中恢复。

如果应用程序在流消息体为只写时调用 XMS IStreamMessage 接口的 Reset 方法，该主体将变为只读。 该方法还会将光标重新定位到消息流的开头。

如果应用程序在流消息的主体为只读时调用 XMS 的 IMessage 接口的 `Clear Body` 方法，那么该主体将变为只读。 该方法还会清除主体。

相关概念

属性值从一种数据类型到另一种数据类型的隐式转换
在应用程序获取某个属性的值后，XMS 可以将该值转换为另一种数据类型。 许多规则可控制受支持的转换以及 XMS 执行转换的方式。

相关参考

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

字节消息
字节消息的主体包含一连串字节。 主体仅包含实际数据，由发送和接收应用程序负责解释这些数据。

映射消息
映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

对象消息
对象消息的主体包含序列化的 Java 或 .NET 对象。

文本消息
文本消息的主体包含一个字符串。

IStreamMessage（适用于 .NET 接口）
流消息是消息主体由一连串值组成的消息，其中每个值都有关联的数据类型。 将按顺序读写主体的内容。

文本消息

文本消息的主体包含一个字符串。

在应用程序创建文本消息之后，消息体为可读且可写。 在应用程序发送消息之后，主体仍然可读且可写。 在应用程序接收文本消息时，消息体为只读。 如果应用程序在文本消息体为只读时调用 .NET IMessage 接口的 Clear Body 方法，该主体将变为可读且可写。 该方法还会清除主体。

相关参考

应用程序数据元素的数据类型
为确保 XMS 应用程序可以与 IBM MQ classes for JMS 应用程序交换消息，这两个应用程序必须能够以相同的方式解释消息体中的应用程序数据。

字节消息
字节消息的主体包含一连串字节。 主体仅包含实际数据，由发送和接收应用程序负责解释这些数据。

映射消息
映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

对象消息
对象消息的主体包含序列化的 Java 或 .NET 对象。

流消息
流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

ITextMessage（适用于 .NET 接口）
文本消息是消息主体由字符串组成的消息。

消息选择器

XMS 应用程序使用消息选择器来选择要接收的消息。

在应用程序创建消息使用者时，它可以将消息选择器表达式与该使用者相关联。 消息选择器表达式可指定选择条件。

当应用程序连接到 IBM WebSphere MQ 7.0 队列管理器时，将在队列管理器端完成消息选择。 XMS 将不做任何选择，只是传递其从队列管理器收到的消息，因此能够提供更好的性能。

应用程序可以创建多个消息使用者，每个都有自己的消息选择器表达式。 如果入局消息满足多个消息使用者的选择条件，那么 XMS 会将该消息传递到所有这些使用者。

消息选择器表达式可以引用消息的以下属性：

- JMS 定义的属性
- IBM 定义的属性
- 应用程序定义的属性

它还可以引用以下消息头字段：

- JMSCorrelationID
- JMSDeliveryMode
- JMSMessageID
- JMSPriority
- JMSTimestamp
- JMSType

但是，消息选择器表达式无法引用消息体中的数据。

下面是消息选择器表达式的示例：

```
JMSPriority > 3 AND manufacturer = 'Jaguar' AND model in ('xj6','xj12')
```

仅当消息的优先级大于 3 时， XMS 才会使用此消息选择器表达式将消息传递给消息使用者; 应用程序定义的属性，制造商 (值为 `Jaguar;`) 和另一个应用程序定义的属性，模型 (值为 `xj6` 或 `xj12.`)

XMS 中构成消息选择器表达式的语法规则与 IBM MQ classes for JMS 中的相同。 有关如何构造消息选择器表达式的信息，请参阅 IBM MQ 产品文档。请注意，在消息选择器表达式中，JMS 定义的属性的名称必须为 JMS 名称，IBM 定义的属性的名称必须为 IBM MQ classes for JMS 名称。 不能在消息选择器表达式中使用 XMS 名称。

相关参考

XMS 消息的组成部分
XMS 消息由一个头、一组属性和一个主体组成。

XMS 消息中的头字段
要允许 XMS 应用程序与 WebSphere JMS 应用程序交换消息，XMS 消息的头包含 JMS 消息头字段。

XMS 消息的属性
XMS 支持三类消息属性：JMS 定义的属性、IBM 定义的属性和应用程序定义的属性。

XMS 消息的主体
消息的主体包含应用程序数据。 但是，消息也可以没有主体，而是只包含头字段和属性。

将 XMS 消息映射到 IBM MQ 消息
将 XMS 消息的 JMS 头字段和属性映射到 IBM MQ 消息的头结构中的字段。

将 XMS 消息映射到 IBM MQ 消息

将 XMS 消息的 JMS 头字段和属性映射到 IBM MQ 消息的头结构中的字段。

在将 XMS 应用程序连接到 IBM MQ 队列管理器时，将采用在类似情况下将 IBM MQ classes for JMS 消息映射到 IBM MQ 消息的相同方式，将发送到队列管理器的消息映射到 IBM MQ 消息。

如果将 Destination 对象的 XMSC_WMQ_TARGET_CLIENT 属性设置为 XMSC_WMQ_TARGET_DEST_JMS，那么会将发送到目标的消息的 JMS 头字段和属性映射到 IBM MQ 消息的 MQMD 和 MQRFH2 头结构中的字段。 如果使用该方式设置 XMSC_WMQ_TARGET_CLIENT 属性，即是假定接收消息的应用程序可以处理 MQRFH2 头。 因此，接收应用程序可能是另一个 XMS 应用程序、IBM MQ classes for JMS 应用程序或设计为处理 MQRFH2 头的本机 IBM MQ 应用程序。

如果将 Destination 对象的 XMSC_WMQ_TARGET_CLIENT 属性转而设置为 XMSC_WMQ_TARGET_DEST_MQ，那么会将发送到目标的消息的 JMS 头字段和属性映射到 IBM MQ 消息的 MQMD 头结构中的字段。 消息不包含 MQRFH2 头，并且将忽略那些无法映射到 MQMD 头结构中字段的 JMS 头字段和属性。 因此，接收消息的应用程序可能是未设计为处理 MQRFH2 头的本机 IBM MQ。

将采用在类似情况下将 IBM MQ 消息映射到 IBM MQ classes for JMS 消息的相同方式，将从队列管理器收到的 IBM MQ 消息映射到 XMS 消息。

如果入局 IBM MQ 消息具有 MQRFH2 头，那么生成的 XMS 消息的主体类型由 MQRFH2 头的 `mcd` 文件夹中的 **Msd** 属性值确定。 如果 **Msd** 属性不存在于 MQRFH2 头中，或者如果 IBM MQ 消息不包含 MQRFH2 头，那么生成的 XMS 消息的主体类型由 MQMD 头中的 *Format* 字段值确定。 如果将 *Format* 字段设置为 MQFMT_STRING，那么 XMS 消息为文本消息。 否则，XMS 消息为字节消息。 如果 IBM MQ 消息不包含 MQRFH2 头，那么只会设置可从 MQMD 头中的字段派生的 JMS 头字段和属性。

有关将 IBM MQ classes for JMS 消息映射到 IBM MQ 消息的更多信息，请参阅 IBM MQ 产品文档。

相关参考

XMS 消息的组成部分
XMS 消息由一个头、一组属性和一个主体组成。

XMS 消息中的头字段
要允许 XMS 应用程序与 WebSphere JMS 应用程序交换消息，XMS 消息的头包含 JMS 消息头字段。

XMS 消息的属性
XMS 支持三类消息属性：JMS 定义的属性、IBM 定义的属性和应用程序定义的属性。

XMS 消息的主体
消息的主体包含应用程序数据。 但是，消息也可以没有主体，而是只包含头字段和属性。

消息选择器
XMS 应用程序使用消息选择器来选择要接收的消息。

通过 *IBM Message Service Client for .NET* 应用程序读写消息描述符

您可以访问 IBM MQ 消息的除 StrucId 和 Version 以外的所有其他消息描述符 (MQMD) 字段；可以读取但不能写入 BackoutCount。 仅当连接到 IBM WebSphere MQ 6.0 或更高版本的队列管理器时，此功能才可用，它由稍后所述的目标属性来控制。

IBM Message Service Client for .NET 提供的消息属性不仅可以帮助 XMS 应用程序设置 MQMD 字段，还可以驱动 IBM WebSphere MQ 应用程序。

使用发布/预订消息传递时，存在一些限制。 例如，将忽略诸如 MsgID 和 CorrelId 的 MQMD 字段。

当连接到 IBM WebSphere MQ 6.0 队列管理器时，此主题中描述的功能不可用于发布/预订消息传递。 当 **PROVIDERVERSION** 属性设置为 6 时，该功能同样不可用。

通过 *IBM Message Service Client for .NET* 应用程序访问 *IBM MQ* 消息数据

您可以在 IBM Message Service Client for .NET 应用程序中访问充当 JMSBytesMessage 主体的完整 IBM MQ 消息数据，其中包括 MQRFH2 头（如果存在）和任何其他 IBM MQ 头（如果存在）。

仅当连接到 IBM WebSphere MQ 7.0 或更高版本的队列管理器并且 WebSphere MQ 消息传递提供程序处于正常方式时，本主题中描述的功能才可用。

Destination 对象属性将确定 XMS 应用程序如何访问充当 JMSBytesMessage 主体的整个 IBM MQ 消息（包括 MQRFH2 头（如果存在））。

故障诊断

本部分提供的信息可帮助您检测和处理在使用 IBM Message Service Client for .NET 时遇到的问题。

本部分包含以下主题：

- 第 69 页的『.NET 应用程序的跟踪配置』
- 第 72 页的『针对 .NET 应用程序的 FFDC 配置』
- 第 73 页的『故障诊断技巧』

.NET 应用程序的跟踪配置

对于 XMS .NET 应用程序，您可以通过应用程序配置文件以及通过 XMS 环境变量来配置跟踪。 您可以选择要跟踪的组件。 通常在 IBM 支持人员的指导下使用跟踪。

XMS .NET 的跟踪基于标准 .NET 跟踪基础结构。

缺省情况下，将禁用除错误跟踪以外的所有其他跟踪。 您可以使用以下任一方式来开启跟踪和配置跟踪设置：

- 使用其名称包含相关可执行程序名称且后缀为 `.config` 的应用程序配置文件。 例如，text.exe 的应用程序配置文件的名称为 text.exe.config。 使用应用程序配置文件是为 XMS .NET 应用程序启用跟踪的首选方法。 要获取更多详细信息，请参阅第 70 页的『使用应用程序配置文件的跟踪配置』。
- 对 XMS C 或 C++ 应用程序使用 XMS 环境变量。 要获取更多详细信息，请参阅第 71 页的『使用 XMS 环境变量的跟踪配置』。

活动跟踪文件名的格式为 xms_trace*PID*.log，其中 *PID* 表示应用程序的进程标识。 缺省情况下，活动跟踪文件的大小限制为 20 MB。 达到此限制后，将对该文件进行重命名和归档。 归档文件名的格式为 xms_trace*PID_YY.MM.DD_HH.MM.SS*.log

缺省情况下，保留的跟踪文件数为 4（1 个活动文件和 3 个归档文件）。 这 4 个文件将用作滚动缓冲区直至应用程序停止，最早的文件将被移除并替换为最新的文件。 您可以通过在应用程序配置文件中指定不同的数目来更改跟踪文件的数目。 但是，必须至少存在 2 个文件（1 个活动文件和 1 个归档文件）。

有两种可用的跟踪文件格式：

- 基本格式的跟踪文件是人类可读的文件，采用 WebSphere Application Server 格式。 此格式是缺省跟踪文件格式。 基本格式与跟踪分析器工具不兼容。
- 高级格式的跟踪文件与跟踪分析器工具兼容。 您必须指定要在应用程序配置文件中采用高级格式生成跟踪文件。

跟踪条目包含以下信息：

- 记录跟踪的日期和时间
- 类名
- 跟踪类型
- 跟踪消息

以下示例显示了来自某个跟踪的摘录：

```
[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
```

在上一个示例中，格式为：

```
[Date Time:Microsecs]         Thread-id    Classname        Trace-type    Methodname    Entry
or Exit
```

其中，Trace-type 为：

> 表示入口
< 表示出口
d 表示调试信息

使用应用程序配置文件的跟踪配置

为 XMS .NET 应用程序配置跟踪的首选方法是使用应用程序配置文件。 该文件的 trace 节包含用于定义要跟踪内容的参数、跟踪文件位置和最大允许大小、已使用的跟踪文件数以及跟踪文件格式。

要使用应用程序配置文件来开启跟踪，只需将该文件与应用程序的可执行文件放在同一个目录中即可。

可以按组件和跟踪类型来启用跟踪。 也可以对整个跟踪组开启跟踪。 可以对层次结构中的组件单独或集体开启跟踪。 可用的跟踪类型包括：

- 调试跟踪
- 异常跟踪
- 警告、参考消息和错误消息
- 方法入口和出口跟踪

以下示例显示了在应用程序配置文件的 Trace 节中定义的跟踪设置：

```
<?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>
```

第 71 页的表 22 更详细地描述了参数设置。

表 *22:* 应用程序配置文件跟踪参数设置

参数	描述
traceSpecification=*ComponentName*=*type*=*state*	*ComponentName* 是想要跟踪的类的名称。 您可以在该名称中使用 * 通配符。 例如，*=all=enabled 指定您要跟踪所有类，IBM.XMS.impl.*=all=enabled 指定您只需要 API 跟踪。 *type* 可以是以下任何跟踪类型： • all • debug • 事件 • EntryExit *state* 可以启用或禁用。 您可使用“:”（冒号）定界符将多个跟踪元素串在一起。
traceFilePath=“*filename*”	如果您未指定 traceFilePath 或者如果 traceFilePath 存在但包含空字符串，那么会将跟踪文件放入当前目录中。 要将跟踪文件存储在指定目录中，请在 traceFilePath 中指定该目录名称，例如： `traceFilePath="c:\somepath"`
traceFileSize=“*size*”	允许的跟踪文件的最大大小。 当文件达到此大小时，会将其归档并重命名。 缺省情况下，最大值为 20 KB，即指定为 traceFileSize=“20000000”。
traceFileNumber=“*number*”	要保留的跟踪文件的数量。 缺省值为 4（1 个活动文件和 3 个归档文件）。 允许的最小数目是 2。
traceFormat=“*format*”	缺省跟踪格式为 basic。 在指定了 traceFormat=“basic” 时、未指定 traceFormat 时或者 traceFormat 存在但包含空字符串时，都将采用此格式生成跟踪文件。 如果需要与跟踪分析器工具兼容的跟踪，那么必须指定 traceFormat=“advanced”。

应用程序配置文件中的跟踪设置是动态的，每次保存或替换该文件时都会重新读入跟踪设置。 如果在编辑该文件后在其中发现错误，那么会将跟踪文件设置还原为其缺省值。

相关概念

使用 XMS 环境变量的跟踪配置

作为使用应用程序配置文件的替代方法，您可以使用 XMS 环境变量来开启跟踪。 仅当应用程序配置文件中不包含跟踪规范时，才可以使用这些环境变量。

使用 XMS 环境变量的跟踪配置

作为使用应用程序配置文件的替代方法，您可以使用 XMS 环境变量来开启跟踪。 仅当应用程序配置文件中不包含跟踪规范时，才可以使用这些环境变量。

要为 XMS .NET 应用程序配置跟踪，请在运行该应用程序之前设置以下环境变量：

表 *23:* 针对 *.NET* 跟踪的环境变量设置

环境变量	缺省	设置	含义
XMS_TRACE_ON	不适用	不适用：将忽略该变量的值	如果设置了 XMS_TRACE_ON，那么缺省情况下将启用所有跟踪。
XMS_TRACE_FILE_PATH	当前工作目录	/dirpath/	跟踪和 FFDC 记录将写入的目录路径。 XMS 将在当前工作目录中创建 FFDC 和跟踪文件，除非您指定了备用位置。 您可以通过将环境变量 XMS_TRACE_FILE_PATH 设置为希望 XMS 在其中创建 FFDC 和跟踪文件的目录的标准路径名。 必须在启动要跟踪的应用程序之前设置该环境变量。 您必须确保应用程序运行时所使用的用户标识有权对 XMS 将在其中创建 FFDC 和跟踪文件的目录执行写操作。
XMS_TRACE_FORMAT	基本	BASIC 和 ADVANCED	指定所需的跟踪格式，可以是 BASIC 或 ADVANCED。 缺省格式为 BASIC。 ADVANCED 格式与跟踪分析器工具兼容。
XMS_TRACE_SPECIFICATION	不适用	请参阅 第 70 页的『使用应用程序配置文件的跟踪配置』	覆盖遵循第 70 页的『使用应用程序配置文件的跟踪配置』中所指定格式的跟踪规范。

相关概念

使用应用程序配置文件的跟踪配置

为 XMS .NET 应用程序配置跟踪的首选方法是使用应用程序配置文件。 该文件的 trace 节包含用于定义要跟踪内容的参数、跟踪文件位置和最大允许大小、已使用的跟踪文件数以及跟踪文件格式。

针对 .NET 应用程序的 FFDC 配置

对于 XMS 的 .NET 实现，将为每个 FFDC 生成一个 FFDC 文件。

采用人类可读的文本文件形式存储首次故障数据捕获 (FFDC) 文件。 这些文件名的格式为 `xmsffdc`*`processID_DateTTimestamp`*`.txt`。 文件名示例为 `xmsffdc264_2006.01.06T13.18.52.990955.txt`。 时间戳记精确到微秒。

文件以异常发生的日期和时间开头，后跟异常类型。 文件包含唯一的短 probeId，可用于查找在何处发生此 FFDC。

无需进行任何配置即可开启 FFDC。 缺省情况下，会将所有 FFDC 文件写入当前目录中。 但是，如果需要，您可以通过在应用程序配置文件的 Trace 节中更改 ffdcDirectory 来指定其他目录。 在以下示例中，会将所有跟踪文件记录到目录 c:client\ffdc 中：

```
<IBM.XMS>
       <Trace ffdc=true ffdcDirectory="c:\client\ffdc"/>
</IBM.XMS>
```

您可以通过在应用程序配置文件的 Trace 节中将 FFDC 设置为 false 来禁用跟踪。

如果您没有使用应用程序配置文件，那么 FFDC 处于开启状态，而跟踪处于关闭状态。

故障诊断技巧

使用以下技巧可帮助您对使用 XMS 时遇到的问题进行故障诊断。

XMS 应用程序无法连接到队列管理器 (MQRC_NOT_AUTHORIZED)

XMS .NET 客户机的行为可能与 IBM MQ JMS 客户机的行为不同。 因此，您可能会发现 XMS 应用程序无法连接到队列管理器，但 JMS 应用程序可以。

- 这个问题的最简单解决方案是尝试使用长度小于 12 个字符且在队列管理器的权限列表中已完全授权的用户标识。 如果此解决方案不是很理想，那么另一种较为复杂的方法是使用安全出口。 如果您对于此问题需要获得进一步的帮助，请联系 IBM 支持人员以获取帮助。
- 如果设置了连接工厂的 XMSC_USERID 属性，那么此属性必须与已登录用户的用户标识和密码匹配。 如果未设置此属性，那么缺省情况下队列管理器将使用已登录用户的用户标识。
- IBM MQ 的用户认证是通过当前已登录用户的详细信息来完成的，而不是通过 XMSC.USERID 和 XMSC.PASSWORD 字段中提供的信息来完成。 这旨在与 IBM MQ 保持一致。 有关认证的更多信息，请参阅 IBM MQ 联机产品文档中的认证信息。

重定向到消息传递引擎的连接

当连接到 WebSphere Application Server 6.0.2 服务集成总线时，可以将所有连接从原始提供程序端点重定向到总线为该客户机连接选择的消息传递引擎。 执行此操作时，始终将连接重定向到由主机名（而不是 IP 地址）指定的主机服务器。 因此，如果无法解析主机名，那么您可能会遇到连接问题。

要成功连接到 WebSphere Application Server 6.0.2 服务集成总线，您可能需要在客户机主机上提供主机名与 IP 地址的映射。 例如，可以在客户机主机上的本地主机表中指定该映射。

支持 telnet 类型的密码认证

XMS .NET 实时传输协议仅支持简单的 telnet 类型密码认证。 XMS .NET 实时传输协议不支持“保护质量”。

设置 double 类型属性的值

在 Windows 64 位平台上，在设置或获取 double 类型属性的值时，如果值小于 Double.Epsilon，那么 SetDoubleProperty() 或 GetDoubleProperty() 方法可能无法正常运行。

例如，如果为 double 类型的属性设置值 4.9E-324，那么 Windows 64 位平台会将其视为 0.0。 因此，在分布式消息传递环境中，如果 JMS 或其他应用程序在任何 UNIX 或 Windows 32 位机器上设置 double 属性的值，而 XMS .NET 在 64 位机器上运行，那么 GetDoubleProperty() 返回的值将为 0.0。 这是 Microsoft .NET Framework 2.0 Framework 存在的已知问题。

可在运行时处理的错误情况

来自 API 调用的返回码是可在运行时处理的错误情况。 处理此类错误的方式取决于您使用的是 C 还是 C++ API。

如何在运行时检测错误

如果应用程序调用 C API 函数并且此调用失败，那么将返回包含返回码而非 XMS_OK 的响应，还返回包含与失败原因相关的更多信息的 XMS 错误块。

C++ API 将在使用方法时抛出异常。

应用程序使用异常侦听器异步通知连接问题。 将向 XMS C 或 C++ API 提供异常侦听器，并使用该 API 来初始化异常侦听器。

如何在运行时处理错误

有些错误情况指示某些资源不可用，并且应用程序可以执行的操作取决于应用程序正在调用的 XMS 函数。例如，如果连接无法连接到服务器，那么应用程序可能希望定期重试，直至建立连接为止。 XMS 错误块或异常可能未包含足够的信息来确定要执行的操作，在这样的情况下，通常会提供包含更具体的诊断信息的链接错误块或异常。

在 C API 中，始终测试具有 XMS_OK 以外的返回码的响应，并且始终传递有关 API 调用的错误块。 所执行的操作通常取决于应用程序正在使用的 API 函数。

在 C++ API 中，始终在 try 块中包含方法调用以捕获所有类型的 XMS 异常，并在 catch 构造中指定 Exception 类。

异常侦听器是可随时启动的异步错误情况路径。 在自己的线程上启动异常侦听器函数后，通常会指示比普通 XMS API 错误情况更严重的故障。 可以执行任何相关操作，但是您必须严格遵循 XMS 线程技术模型的规则，如第 18 页的『线程技术模型』中所述。

相关概念

线程技术模型

一般规则可控制多线程应用程序如何使用 XMS 对象。

Message Service Clients for .NET 参考

本参考部分提供的信息可帮助您使用 Message Service Client for .NET。 此信息可帮助您完成 XMS 编程中涉及的任务。

.NET 接口

本部分记录了 .NET 类接口及其属性和方法。

下表汇总了 `IBM.XMS` 名称空间内定义的所有接口。

表 *24: .NET* 类接口汇总

接口	描述
第 76 页的『IBytesMessage』	字节消息是其主体由字节流组成的消息。
第 86 页的『IConnection』	Connection 对象表示应用程序到消息传递服务器的活动连接。
第 89 页的『IConnectionFactory』	应用程序使用连接工厂来创建连接。
第 90 页的『IConnectionMetaData』	ConnectionMetaData 对象提供有关连接的信息。
第 91 页的『IDestination』	目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。
第 93 页的『ExceptionListener』	应用程序使用异常侦听器来获得连接问题的异步通知。
第 93 页的『IllegalStateException』	如果应用程序在不正确或不适当的时间调用方法，或者如果 XMS 未处于请求的操作的适当状态，那么 XMS 将抛出此异常。
第 94 页的『InitialContext』	应用程序使用 InitialContext 对象，通过从受管对象存储库中检索的对象定义来创建对象。
第 96 页的『InvalidClientIDException』	如果应用程序尝试为连接设置客户机标识，但该客户机标识无效或已在使用中，那么 XMS 将抛出此异常。
第 96 页的『InvalidDestinationException』	如果应用程序指定的目标无效，那么 XMS 将抛出此异常。

表 *24: .NET* 类接口汇总 *(*继续*)*	
接口	**描述**
第 96 页的『InvalidSelectorException』	如果应用程序提供了语法无效的消息选择器表达式，那么 XMS 将抛出此异常。
第 97 页的『IMapMessage』	映射消息是消息主体由一组名称/值对组成的消息，其中每个值都有关联的数据类型。
第 106 页的『IMessage』	Message 对象表示应用程序发送或接收的消息。IMessage 是诸如 IMapMessage 等消息类的超类。
第 112 页的『IMessageConsumer』	应用程序使用消息使用者接收向目标发送的消息。
第 114 页的『MessageEOFException』	如果 XMS 在应用程序读取字节消息的主体时迁到字节消息流的结尾，那么 XMS 将抛出此异常。
第 114 页的『MessageFormatException』	如果 XMS 迁到格式无效的消息，那么 XMS 将抛出此异常。
第 115 页的『IMessageListener（委托）』	应用程序使用消息侦听器以异步方式接收消息。
第 115 页的『MessageNotReadableException』	如果应用程序尝试读取仅写入的消息体，那么 XMS 将抛出此异常。
第 115 页的『MessageNotWritableException』	如果应用程序尝试写入只读消息的主体，那么 XMS 将抛出此异常。
第 116 页的『IMessageProducer』	应用程序使用消息生产者向目标发送消息。
第 121 页的『IObjectMessage』	对象消息是其主体包含序列化 Java 或 .NET 对象的消息。
第 122 页的『IPropertyContext』	IPropertyContext 是一个抽象超类，其中包含用于获取和设置属性的方法。 其他类将继承这些方法。
第 131 页的『IQueueBrowser』	应用程序使用队列浏览器来浏览队列中的消息，而不将其除去。
第 133 页的『请求者』	应用程序可以使用请求程序来发送请求消息，然后等待接收应答。
第 134 页的『ResourceAllocationException』	如果 XMS 无法分配方法所需的资源，那么 XMS 将抛出此异常。
第 135 页的『SecurityException』	XMS 将抛出此异常。 如果权限检查失败并导致方法无法完成，那么 XMS 也会抛出此异常。
第 135 页的『ISession』	会话是用于发送和接收消息的单线程上下文。
第 145 页的『IStreamMessage』	流消息是消息主体由一连串值组成的消息，其中每个值都有关联的数据类型。
第 155 页的『ITextMessage』	文本消息是消息主体由字符串组成的消息。
第 156 页的『TransactionInProgressException』	XMS 在应用程序请求由于事务正在进行而无效的操作时抛出此异常。
第 156 页的『TransactionRolledBackException』	如果应用程序调用 Session.commit() 以落实当前事务，但该事务随后回滚，那么 XMS 将抛出此异常。
XMSC	对于 .NET，XMS 属性名称和值在此类中定义为公共常量。 要获取更多详细信息，请参阅第 159 页的『XMS 对象的属性』。

表 24: .NET 类接口汇总 (继续)	
接口	**描述**
第 156 页的『XMSException』	如果 XMS 在处理对 .NET 方法的调用时检测到错误，那么 XMS 将抛出异常。 异常是用于封装错误相关信息的对象。 存在不同类型的 XMS 异常，而 XMSException 对象只是一种类型的异常。 但是，XMSException 类是其他 XMS 异常类的超类。 在没有任何其他类型的异常适用的情况下，XMS 会抛出 XMSException 对象。
第 157 页的『XMSFactoryFactory』	如果应用程序没有使用受管对象，请使用此类来创建连接工厂、队列和主题。

每个方法的定义都列出了 XMS 在处理方法调用的过程中检测到错误时可能返回的异常代码。 每个异常代码都由其命名常量表示，而命名常量都有各自对应的异常。

相关概念

构建自己的应用程序
您可以像构建样本应用程序那样构建自己的应用程序。

编写 XMS 应用程序
本部分中的主题所提供的信息可帮助您编写 XMS 应用程序。

编写 XMS .NET 应用程序
本部分中的主题所提供的信息可帮助您编写 XMS .NET 应用程序。

相关参考

XMS 对象的属性
本章记录了 XMS 定义的对象属性。

IBytesMessage

字节消息是其主体由字节流组成的消息。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IBytesMessage
```

相关参考

字节消息
字节消息的主体包含一连串字节。 主体仅包含实际数据，由发送和接收应用程序负责解释这些数据。

.NET 属性

BodyLength - 获取主体长度

接口：

```
Int64 BodyLength
    {
      get;
    }
```

在消息主体为只读时，用于获取消息主体的长度（以字节计）。

返回的值是整个主体的长度，与读取消息时光标当前所在的位置无关。

异常：

- XMSException
- MessageNotReadableException

方法

ReadBoolean - 读取布尔值

接口：

```
Boolean ReadBoolean();
```

从字节消息流中读取布尔值。

参数：
None

返回：
读取的布尔值。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadSignedByte - 读取单个字节

接口：

```
Int16   ReadSignedByte();
```

从字节消息流中读取下一个字节作为有符号的 8 位整数。

参数：
None

返回：
读取的字节。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - 读取多个字节

接口：

```
Int32   ReadBytes(Byte[] array);
Int32   ReadBytes(Byte[] array, Int32 length);
```

从字节消息流中读取字节数组，从光标当前位置开始。

参数：

array（输出）
包含所读取的字节数组的缓冲区。 如果在调用前要从流中读取的剩余字节数大于或等于缓冲区长度，那么缓冲区将填满。 否则，将使用所有剩余字节部分填充缓冲区。

如果在输入上指定空指针，那么此方法会跳过字节，而不读取这些字节。 如果在调用前要从流中读取的剩余字节数大于或等于缓冲区长度，那么跳过的字节数等于缓冲区长度。 否则，将跳过所有剩余字节。 光标保留在字节消息流中要读取的下一个位置。

length（输入）
缓冲区的长度（以字节计）

返回：
读入到缓冲区的字节数。 如果部分填充缓冲区，那么该值小于缓冲区长度，这表明不存在待读取的剩余字节。 如果在调用前流中不存在待读取的剩余字节，那么该值为 XMSC_END_OF_STREAM。

如果在输入上指定空指针，那么此方法不返回任何值。

异常：

- XMSException
- MessageNotReadableException

ReadChar - 读取字符

接口：

```
Char    ReadChar();
```

从字节消息流中读取接下来的 2 个字节作为一个字符。

参数：
None

返回：
读取的字符。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - 读取双精度浮点数

接口：

```
Double  ReadDouble();
```

从字节消息流中读取接下来的 8 个字节作为一个双精度浮点数。

参数：
None

返回：
读取的双精度浮点数。

异常：

- XMSException
- MessageNotReadableException

- MessageEOFException

ReadFloat - 读取浮点数

接口：

```
Single  ReadFloat();
```

从字节消息流中读取接下来的 4 个字节作为一个浮点数。

参数：
None

返回：
读取的浮点数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - 读取整数

接口：

```
Int32   ReadInt();
```

从字节消息流中读取接下来的 4 个字节作为一个有符号的 32 位整数。

参数：
None

返回：
读取的整数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - 读取长整数

接口：

```
Int64   ReadLong();
```

从字节消息流中读取接下来的 8 个字节作为一个有符号的 64 位整数。

参数：
None

返回：
读取的长整数。

异常：

- XMSException

- MessageNotReadableException
- MessageEOFException

ReadShort - 读取短整数

接口：

```
Int16   ReadShort();
```

从字节消息流中读取接下来的 2 个字节作为一个有符号的 16 位整数。

参数：
None

返回：
读取的短整数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - 读取无符号的字节

接口：

```
Byte    ReadByte();
```

从字节消息流中读取下一个字节作为无符号的 8 位整数。

参数：
None

返回：
读取的字节。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadUnsignedShort - 读取无符号的短整数

接口：

```
Int32   ReadUnsignedShort();
```

从字节消息流中读取接下来的 2 个字节作为一个无符号的 16 位整数。

参数：
None

返回：
读取的无符号的短整数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadUTF - 读取 *UTF* 字符串

接口：

```
String  ReadUTF();
```

从字节消息流中读取采用 UTF-8 编码的字符串。

注： 在调用 ReadUTF() 之前，请确保缓冲区的光标正指向字节消息流的开始位置。

参数：
None

返回：
用于封装读取的字符串的 String 对象。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - 重置

接口：

```
void Reset();
```

将消息主体置于只读方式，并将光标重新定位在字节消息流的开始位置。

参数：
None

返回：
无效

异常：

- XMSException
- MessageNotReadableException

WriteBoolean - 写入布尔值

接口：

```
void  WriteBoolean(Boolean value);
```

将布尔值写入字节消息流中。

参数：

value（输入）
要写入的布尔值。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteByte - 写入单个字节

接口：

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

将一个字节写入字节消息流中。

参数：

value（输入）
要写入的字节。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteBytes - 写入多个字节

接口：

```
void  WriteBytes(Byte[] value);
```

将字节数组写入字节消息流中。

参数：

value（输入）
要写入的字节数组。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteBytes - 写入部分字节数组

接口：

```
void  WriteBytes(Byte[] value, int offset, int length);
```

将部分字节数组写入字节消息流中（如指定长度所定义）。

参数：

value（输入）
要写入的字节数组。

offset（输入）
要写入的字节数组的起点。

length（输入）
要写入的字节数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteChar - 写入字符

接口：

```
void  WriteChar(Char value);
```

将一个字符作为 2 个字节（首先是高位字节）写入字节消息流中。

参数：

value（输入）
要写入的字符。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteDouble - 写入双精度浮点数

接口：

```
void  WriteDouble(Double value);
```

将双精度浮点数转换为长整数，并将此长整数作为 8 个字节（首先是高位字节）写入字节消息流中。

参数：

value（输入）
要写入的双精度浮点数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteFloat - 写入浮点数

接口：

```
void  WriteFloat(Single value);
```

将浮点数转换为整数，并将此整数作为 4 个字节（首先是高位字节）写入字节消息流中。

参数：

value（输入）
要写入的浮点数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteInt - 写入整数

接口：

```
void  WriteInt(Int32 value);
```

将整数作为 4 个字节（首先是高位字节）写入字节消息流中。

参数：

value（输入）
要写入的整数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteLong - 写入长整数

接口：

```
void  WriteLong(Int64 value);
```

将长整数作为 8 个字节（首先是高位字节）写入字节消息流中。

参数：

value（输入）
要写入的长整数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteObject - 写入对象

接口：

```
void  WriteObject(Object value);
```

将指定的对象写入字节消息流中。

参数：

value（输入）
要写入的对象，此对象必须是对基本类型的引用。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteShort - 写入短整数

接口：

```
void  WriteShort(Int16 value);
```

将短整数作为 2 个字节（首先是高位字节）写入字节消息流中。

参数：

value（输入）
要写入的短整数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteUTF - 写入 *UTF* 字符串

接口：

```
void  WriteUTF(String value);
```

将采用 UTF-8 编码的字符串写入字节消息流中。

参数：

value（输入）
用于封装待写入字符串的 String 对象。

返回：
无效

异常：

- XMSException

- MessageNotWritableException

继承的属性和方法

从 IMessage 接口继承以下属性：

JMSCorrelationID、JMSDeliveryMode、JMSDestination、JMSExpiration、JMSMessageID、JMSPriority、JMSRedelivered、JMSReplyTo、JMSTimestamp、JMSType 或 Properties

从 IMessage 接口继承以下方法：

clearBody, clearProperties, PropertyExists

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IConnection

Connection 对象表示应用程序到消息传递服务器的活动连接。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnection
```

要获取 XMS 定义的 Connection 对象属性列表，请参阅 第 159 页的『Connection 属性』。

.NET 属性

ClientID - 获取和设置客户机标识

接口：

```
String ClientID
      {
        get;
        set;
      }
```

获取和设置连接的客户机标识。

客户机标识可由管理员在 ConnectionFactory 中预配置或者通过设置 ClientID 来指定。

客户机标识仅用于在发布/预订域中支持持久预订，而在点到点域中会忽略此标识。

如果应用程序要为连接设置客户机标识，那么该应用程序必须在创建连接后且在对连接执行任何其他操作之前立即设置此标识。 如果该应用程序尝试在这个时间点后设置客户机标识，那么调用将抛出异常 IllegalStateException。

对于与代理程序的实时连接，该属性无效。

异常：

- XMSException
- IllegalStateException
- InvalidClientIDException

ExceptionListener - 获取和设置异常侦听器

接口：

```
ExceptionListener ExceptionListener
    {
      get;
      set;
    }
```

获取向连接注册的异常侦听器，并向连接注册异常侦听器。

如果未向连接注册异常侦听器，那么此方法将返回空值。 如果已向连接注册异常侦听器，可以通过指定空值（代替异常侦听器）来取消注册。

有关使用异常侦听器的更多信息，请参阅第 40 页的『.NET 中的消息和异常侦听器』。

异常：

- XMSException

Metadata - 获取元数据

接口：

```
IConnectionMetaData MetaData
    {
      get;
    }
```

获取连接的元数据。

异常：

- XMSException

方法

Close - 关闭连接

接口：

```
void  Close();
```

关闭连接。

如果应用程序尝试关闭已关闭的连接，那么将忽略此调用。

参数：
None

返回：
无效

异常：

- XMSException

CreateSession - 创建会话

接口：

```
ISession CreateSession(Boolean transacted,
                       AcknowledgeMode acknowledgeMode);
```

创建会话。

参数：

transacted（输入）
值 `True` 表示此会话是事务性会话。 值 `False` 表示此会话不是事务性会话。

对于与代理程序的实时连接，该值必须为 `False`。

acknowledgeMode（输入）
指示如何确认应用程序接收的消息。 该值必须是来自 AcknowledgeMode 枚举符的以下值之一：

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

对于与代理程序的实时连接，该值必须为 `AcknowledgeMode.AutoAcknowledge` 或 `AcknowledgeMode.DupsOkAcknowledge`

如果会话是事务性会话，那么将忽略此参数。 有关确认方式的更多信息，请参阅第 22 页的『消息确认』。

返回：
Session 对象

异常：

- XMSException

Start - 启动连接

接口：

```
void  Start();
```

对连接启动或重新启动入局消息传递。 如果连接已启动，那么将忽略此调用。

参数：
None

返回：
无效

异常：

- XMSException

Stop - 停止连接

接口：

```
void  Stop();
```

对连接停止入局消息传递。 如果连接已停止，那么将忽略此调用。

参数：
None

返回：
无效

异常：

- XMSException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IConnectionFactory

应用程序使用连接工厂来创建连接。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionFactory
```

要获取 XMS 定义的 ConnectionFactory 对象属性列表，请参阅 第 160 页的『ConnectionFactory 属性』。

相关概念

ConnectionFactories 和 Connection 对象
ConnectionFactory 对象提供一种模板以供应用程序用于创建 Connection 对象。 应用程序可使用 Connection 对象来创建 Session 对象。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

与 IBM MQ 队列管理器的安全连接
要使 XMS .NET 应用程序能够与 IBM MQ 队列管理器建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

与 WebSphere Application Server service integration bus 消息传递引擎的安全连接
要使 XMS .NET 应用程序能够与 WebSphere Application Server service integration bus 消息传递引擎建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

方法

CreateConnection - 创建连接工厂（使用缺省用户身份）

接口：

```
IConnection CreateConnection();
```

使用缺省属性创建连接工厂。

如果要连接到 WebSphere MQ，并且未设置 XMSC_USERID，那么缺省情况下队列管理器将使用已登录用户的 userID。 如果需要个别用户通过进一步的连接级别认证，可以编写在 WebSphere MQ 中配置的客户机认证出口。

参数：
None

异常：

- XMSException

CreateConnection - 创建连接（使用指定的用户身份）

接口：

```
IConnection CreateConnection(String userId, String password);
```

使用指定的用户身份创建连接。

如果要连接到 WebSphere MQ，并且未设置 XMSC_USERID，那么缺省情况下队列管理器将使用已登录用户的 userID。 如果需要个别用户通过进一步的连接级别认证，可以编写在 WebSphere MQ 中配置的客户机认证出口。

使用停止方式来创建该连接。 在应用程序调用 **Connection.start()** 后才会传递消息。

参数：

userID（输入）
用于封装当认证应用程序时要使用的用户标识的 String 对象。 如果提供空值，那么会尝试在不进行认证的情况下创建连接。

password（输入）
用于封装当认证应用程序时要使用的密码的 String 对象。 如果提供空值，那么会尝试在不进行认证的情况下创建连接。

返回：
Connection 对象。

异常：

- XMSException
- XMS_X_SECURITY_EXCEPTION

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IConnectionMetaData

ConnectionMetaData 对象提供有关连接的信息。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionMetaData
```

要获取 XMS 定义的 ConnectionMetaData 对象属性列表，请参阅 第 164 页的『ConnectionMetaData 属性』。

.NET 属性

JMSXPropertyNames - 获取 *JMS* 定义的消息属性

接口：

```
System.Collections.IEnumerator JMSXPropertyNames
    {
      get;
    }
```

返回连接支持的 JMS 定义的消息属性名称的枚举。

JMS 定义的消息属性不受与代理程序的实时连接支持。

异常：

- XMSException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IDestination

目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IDestination
```

要获取 XMS 定义的 Destination 对象属性列表，请参阅 第 165 页的『Destination 属性』。

相关概念

ConnectionFactories 和 Connection 对象
ConnectionFactory 对象提供一种模板以供应用程序用于创建 Connection 对象。 应用程序可使用 Connection 对象来创建 Session 对象。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

目标
XMS 应用程序可使用 Destination 对象来指定所发送消息的目标以及所接收消息的源。

目标通配符

XMS 支持目标通配符，可确保将通配符传递到需要进行匹配的位置。 XMS 可使用的每种服务器类型各有不同的通配符方案。

主题统一资源标识
主题统一资源标识 (URI) 指定主题的名称，还可以指定主题的一个或多个属性。

队列统一资源标识
队列 URI 指定队列的名称，还可以指定队列的一个或多个属性。

临时目标
XMS 应用程序可以创建和使用临时目标。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

相关任务
创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

相关参考
受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

.NET 属性

Name - 获取目标名称

接口：

```
String Name
{
  get;
}
```

获取目标的名称。 此名称是用于封装队列名称或主题名称的字符串。

异常：

- XMSException

TypeId - 获取目标类型

接口：

```
DestinationType TypeId
{
  get;
}
```

获取目标的类型。 目标类型可为以下值之一：

```
DestinationType.Queue
DestinationType.Topic
```

异常：

- XMSException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

ExceptionListener

继承层次结构：

None

应用程序使用异常侦听器来获得连接问题的异步通知。

如果应用程序仅将连接用于以异步方式使用消息，而不用于其他用途，那么应用程序获悉连接问题的唯一方法是使用异常侦听器。 在其他情况下，异常侦听器可以提供一种较为直接的方式来获悉连接问题，而无需等到下一次同步调用 XMS。

代理人

ExceptionListener - 异常侦听器

接口：

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

向应用程序通知连接问题。

可向连接注册用于实现此委托的方法。

有关使用异常侦听器的更多信息，请参阅第 40 页的『.NET 中的消息和异常侦听器』。

参数：

exception（输入）
指向由 XMS 创建的异常的指针。

返回：

无效

IllegalStateException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.Exception
           |
           +----IBM.XMS.IllegalStateException
```

如果应用程序在不正确或不适当的时间调用方法，或者如果 XMS 未处于请求的操作的适当状态，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

InitialContext

应用程序使用 InitialContext 对象，通过从受管对象存储库中检索的对象定义来创建对象。

继承层次结构：

None

相关概念

InitialContext 属性

InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式

将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

检索受管对象

XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

相关任务

InitialContext 对象

应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

.NET 属性

Environment - 获取环境

接口：

```
Hashtable Environment
{
     get;
}
```

获取环境。

异常：

- 异常特定于所使用的目录服务。

构造函数

InitialContext - 创建初始上下文

接口：

```
InitialContext(Hashtable env);
```

创建 InitialContext 对象。

参数：

向环境散列表中的构造函数提供与受管对象存储库建立连接所需的信息。

异常：

- XMSException

方法

AddToEnvironment - 向环境添加新属性

接口：

```
Object AddToEnvironment(String propName, Object propVal);
```

向环境添加新属性。

参数：

propName（输入）
用于封装要添加的属性名称的 String 对象。

propVal（输入）
要添加的属性值。

返回：
属性的原值。

异常：

- 异常特定于所使用的目录服务。

Close - 关闭此上下文

接口：

```
void Close()
```

关闭此上下文。

参数：
None

返回：
None

异常：

- 异常特定于所使用的目录服务。

Lookup - 在初始上下文中查找对象

接口：

```
Object Lookup(String name);
```

通过从受管对象存储库检索的对象定义来创建对象。

参数：

name（输入）
用于封装要检索的受管对象名称的 String 对象。 此名称可以是简单名称或复杂名称。 要获取更多详细信息，请参阅第 54 页的『检索受管对象』。

返回：
IConnectionFactory 或 IDestination（取决于检索的对象类型）。 如果函数可以访问目录，但找不到所需的对象，那么将返回空值。

异常：

- 异常特定于所使用的目录服务。

RemoveFromEnvironment - 从环境中除去属性

接口：

```
Object RemoveFromEnvironment(String propName);
```

从环境中除去属性。

参数：

propName（输入）
用于封装要除去的属性名称的 String 对象。

返回：
已除去的对象。

异常：

- 异常特定于所使用的目录服务。

InvalidClientIDException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidClientIDException
```

如果应用程序尝试为连接设置客户机标识，但该客户机标识无效或已在使用中，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

InvalidDestinationException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidDestinationException
```

如果应用程序指定的目标无效，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

InvalidSelectorException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidSelectorException
```

如果应用程序提供了语法无效的消息选择器表达式，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

IMapMessage

映射消息是消息主体由一组名称/值对组成的消息，其中每个值都有关联的数据类型。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IMapMessage
```

当应用程序获取名称/值对的值时，该值可由 XMS 转换为其他数据类型。 有关这种格式的隐式转换的更多信息，请参阅第 64 页的『映射消息』。

相关参考

映射消息

映射消息的主体包含一组名称/值对，其中每个值都具有相关的数据类型。

.NET 属性

MapNames - 获取映射名称

接口：

```
System.Collections.IEnumerator MapNames
{
  get;
}
```

获取映射消息主体中名称的枚举。

异常：

- XMSException

方法

GetBoolean - 获取布尔值

接口：

```
Boolean GetBoolean(String name);
```

从映射消息主体中获取由名称标识的布尔值。

参数：

name（输入）
用于封装可标识布尔值的名称的 String 对象。

返回：
从映射消息主体中检索的布尔值。

异常：

- XMSException

GetByte - 获取单个字节

接口：

```
Byte    GetByte(String name);
    Int16   GetSignedByte(String name);
```

从映射消息主体中获取由名称标识的字节。

参数：

name（输入）
用于封装可标识字节的名称的 String 对象。

返回：
从映射消息主体中检索的字节。 将不会对该字节执行任何数据转换。

异常：

- XMSException

GetBytes - 获取多个字节

接口：

```
Byte[]  GetBytes(String name);
```

从映射消息主体中获取由名称标识的字节数组。

参数：

name（输入）
用于封装可标识字节数组的名称的 String 对象。

返回：
数组中的字节数。

异常：

- XMSException

GetChar - 获取字符

接口：

```
Char    GetChar(String name);
```

从映射消息主体中获取由名称标识的字符。

参数：

name（输入）
用于封装可标识字符的名称的 String 对象。

返回：
从映射消息主体中检索的字符。

异常：

- XMSException

GetDouble - 获取双精度浮点数

接口：

```
Double  GetDouble(String name);
```

从映射消息主体中获取由名称标识的双精度浮点数。

参数：

name（输入）
用于封装可标识双精度浮点数的名称的 String 对象。

返回：
从映射消息主体中检索的双精度浮点数。

异常：

- XMSException

GetFloat - 获取浮点数

接口：

```
Single  GetFloat(String name);
```

从映射消息主体中获取由名称标识的浮点数。

参数：

name（输入）
用于封装可标识浮点数的名称的 String 对象。

返回：
从映射消息主体中检索的浮点数。

异常：

- XMSException

GetInt - 获取整数

接口：

```
Int32   GetInt(String name);
```

从映射消息主体中获取由名称标识的整数。

参数：

name（输入）
用于封装可标识整数的名称的 String 对象。

返回：
从映射消息主体中检索的整数。

异常：

- XMSException

GetLong - 获取长整数

接口：

```
Int64   GetLong(String name);
```

从映射消息主体中获取由名称标识的长整数。

参数：

name（输入）
用于封装可标识长整数的名称的 String 对象。

返回：
从映射消息主体中检索的长整数。

异常：

- XMSException

GetObject - 获取对象

接口：

```
Object  GetObject(String name);
```

从映射消息主体中获取对名称/值对中的值的引用。 此名称/值对由名称标识。

参数：

name（输入）
用于封装名称/值对中的名称的 String 对象。

返回：
以下某种对象类型的值：

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

异常：
XMSException

GetShort - 获取短整数

接口：

```
Int16   GetShort(String name);
```

从映射消息主体中获取由名称标识的短整数。

参数：

name（输入）
用于封装可标识短整数的名称的 String 对象。

返回：
从映射消息主体中检索的短整数。

异常：

- XMSException

GetString - 获取字符串

接口：

```
String  GetString(String name);
```

从映射消息主体中获取由名称标识的字符串。

参数：

name（输入）
用于封装可标识映射消息主体中的字符串的名称的 String 对象。

返回：
用于封装从映射消息主体中检索的字符串的 String 对象。 如果需要进行数据转换，那么该值是转换后的字符串。

异常：

- XMSException

ItemExists - 检查名称/值对是否存在

接口：

```
Boolean ItemExists(String name);
```

检查映射消息主体是否包含具有指定名称的名称/值对。

参数：

name（输入）
用于封装名称/值对中的名称的 String 对象。

返回：

- `True`（如果映射消息主体包含具有指定名称的名称/值对）。
- `False`（如果映射消息主体不包含具有指定名称的名称/值对）。

异常：

- XMSException

SetBoolean - 设置布尔值

接口：

```
void  SetBoolean(String name, Boolean value);
```

在映射消息主体中设置布尔值。

参数：

name（输入）
用于封装可标识映射消息主体中的布尔值的名称的 String 对象。

value（输入）
要设置的布尔值。

返回：
无效

异常：

- XMSException

SetByte - 设置单个字节

接口：

```
void  SetByte(String name, Byte value);
void  SetSignedByte(String name, Int16 value);
```

在映射消息主体中设置单个字节。

参数：

name（输入）
用于封装可标识映射消息主体中的字节的名称的 String 对象。

value（输入）
要设置的字节。

返回：
无效

异常：

- XMSException

SetBytes - 设置多个字节

接口：

```
void  SetBytes(String name, Byte[] value);
```

在映射消息主体中设置字节数组。

参数：

name（输入）
用于封装可标识映射消息主体中的字节数组的名称的 String 对象。

value（输入）
要设置的字节数组。

返回：
无效

异常：

- XMSException

SetChar - 设置字符

接口：

```
void  SetChar(String name, Char value);
```

在映射消息主体中设置 2 字节字符。

参数：

name（输入）
用于封装可标识映射消息主体中的字符的名称的 String 对象。

value（输入）
要设置的字符。

返回：
无效

异常：

- XMSException

SetDouble - 设置双精度浮点数

接口：

```
void  SetDouble(String name, Double value);
```

在映射消息主体中设置双精度浮点数。

参数：

name（输入）
用于封装可标识映射消息主体中的双精度浮点数的名称的 String 对象。

value（输入）
要设置的双精度浮点数。

返回：
无效

异常：

- XMSException

SetFloat - 设置浮点数

接口：

```
void  SetFloat(String name, Single value);
```

在映射消息主体中设置浮点数。

参数：

name（输入）
用于封装可标识映射消息主体中的浮点数的名称的 String 对象。

value（输入）
要设置的浮点数。

返回：
无效

异常：

- XMSException

SetInt - 设置整数

接口：

```
void  SetInt(String name, Int32 value);
```

在映射消息主体中设置整数。

参数：

name（输入）
用于封装可标识映射消息主体中的整数的名称的 String 对象。

value（输入）
要设置的整数。

返回：
无效

异常：

- XMSException

SetLong - 设置长整数

接口：

```
void  SetLong(String name, Int64 value);
```

在映射消息主体中设置长整数。

参数：

name（输入）
用于封装可标识映射消息主体中的长整数的名称的 String 对象。

value（输入）
要设置的长整数。

返回：
无效

异常：

- XMSException

SetObject - 设置对象

接口：

```
void  SetObject(String name, Object value);
```

在映射消息主体中设置必须为 XMS 基本类型的值。

参数：

name（输入）
用于封装可标识映射消息主体中的值的名称的 String 对象。

value（输入）
包含要设置的值的字节数组。

返回：
无效

异常：

- XMSException

SetShort - 设置短整数

接口：

```
void  SetShort(String name, Int16 value);
```

在映射消息主体中设置短整数。

参数：

name（输入）
用于封装可标识映射消息主体中的短整数的名称的 String 对象。

value（输入）
要设置的短整数。

返回：
无效

异常：

- XMSException

SetString - 设置字符串

接口：

```
void  SetString(String name, String value);
```

在映射消息主体中设置字符串。

参数：

name（输入）
用于封装可标识映射消息主体中的字符串的名称的 String 对象。

value（输入）
用于封装要设置的字符串的 String 对象。

返回：
无效

异常：

- XMSException

继承的属性和方法

从 IMessage 接口继承以下属性：

JMSCorrelationID、JMSDeliveryMode、JMSDestination、JMSExpiration、JMSMessageID、JMSPriority、JMSRedelivered、JMSReplyTo、JMSTimestamp、JMSType 或 Properties

从 IMessage 接口继承以下方法：

clearBody, clearProperties, PropertyExists

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IMessage

Message 对象表示应用程序发送或接收的消息。 IMessage 是诸如 IMapMessage 等消息类的超类。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
```

要获取 Message 对象中 JMS 消息头字段的列表，请参阅 第 59 页的『XMS 消息中的头字段』。 要获取 JMS 定义的 Message 对象属性列表，请参阅 第 60 页的『消息的 JMS 定义的属性』。 要获取 IBM 定义的 Message 对象属性列表，请参阅 第 61 页的『消息的 IBM 定义的属性』。 要获取 Message 对象的 JMS_IBM_MQMD* 属性列表，请参阅 第 168 页的『JMS_IBM_MQMD* 属性』

消息由垃圾回收器进行删除。 删除消息时，将释放其使用的资源。

相关参考

XMS 消息

本部分描述了 XMS 消息的结构和内容，并解释了应用程序如何处理 XMS 消息。

.NET 属性

GetJMSCorrelationID - 获取和设置 *JMSCorrelationID*

接口：

```
String JMSCorrelationID
{
  get;
  set;
}
```

获取消息的相关标识并将其设置为 String 对象。

异常：

- XMSException

JMSDeliveryMode - 获取和设置 *JMSDeliveryMode*

接口：

```
DeliveryMode JMSDeliveryMode
{
  get;
  set;
}
```

获取和设置消息的传递方式。

消息的传递方式可为以下值之一：

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

对于尚未发送的新创建的消息，传递方式为 DeliveryMode.Persistent，但与代理程序的实时连接除外，其传递方式为 DeliveryMode.NonPersistent。 对于已收到的消息，此方法将返回由 IMessageProducer.send() 调用在发送消息时设置的传递方式，除非接收应用程序通过设置 JMSDeliveryMode 更改了传递方式。

异常：

- XMSException

JMSDestination - 获取和设置 *JMSDestination*

接口：

```
IDestination JMSDestination
{
  get;
  set;
}
```

获取和设置消息的目标。

目标是由 IMessageProducer.send() 调用在发送消息时设置的。 将忽略 JMSDestination 的值。 但是，可以使用 JMSDestination 来更改已接收的消息的目标。

对于尚未发送的新创建的消息，此方法将返回空的 Destination 对象，除非发送应用程序通过设置 JMSDestination 设置了目标。 对于已收到的消息，此方法将针对由 IMessageProducer.send() 调用在发送消息时设置的目标返回 Destination 对象，除非接收应用程序通过设置 JMSDestination 更改了目标。

异常：

- XMSException

JMSExpiration - 获取和设置 *JMSExpiration*

接口：

```
Int64 JMSExpiration
{
  get;
  set;
}
```

获取和设置消息的到期时间。

到期时间是由 IMessageProducer.send() 调用在发送消息时设置的。 其值的计算方式为：在消息发送时间上加上发送应用程序所指定的生存时间。 到期时间将以自 1970 年 1 月 1 日格林威治标准时间 00:00:00 起的毫秒数表示。

对于尚未发送的新创建的消息，到期时间为 0，除非发送应用程序通过设置 JMSExpiration 设置了不同的到期时间。 对于已收到的消息，此方法将返回由 IMessageProducer.send() 调用在发送消息时设置的到期时间，除非接收应用程序通过设置 JMSExpiration 更改了到期时间。

如果生存时间为 0，那么 IMessageProducer.send() 调用会将到期时间设置为 0，以指示此消息不会到期。

XMS 将丢弃到期的消息，而不会将其传递到应用程序。

异常：

- XMSException

JMSMessageID - 获取和设置 *JMSMessageID*

接口：

```
String JMSMessageID
{
  get;
  set;
}
```

获取消息的消息标识，并将其设置为用于封装消息标识的 String 对象。

消息标识是由 IMessageProducer.send() 调用在发送消息时设置的。 对于已收到的消息，此方法将返回由 IMessageProducer.send() 调用在发送消息时设置的消息标识，除非接收应用程序通过设置 JMSMessageID 更改了消息标识。

如果消息没有消息标识，那么此方法将返回空值。

异常：

- XMSException

JMSPriority - 获取和设置 *JMSPriority*

接口：

```
Int32 JMSPriority
{
  get;
  set;
}
```

获取和设置消息优先级。

优先级是由 IMessageProducer.send() 调用在发送消息时设置的。 该值是范围 0（表示最低优先级）到 9（表示最高优先级）内的整数。

对于尚未发送的新创建的消息，优先级为 4，除非发送应用程序通过设置 JMSPriority 设置了不同的优先级。 对于已收到的消息，此方法将返回由 IMessageProducer.send() 调用在发送消息时设置的优先级，除非接收应用程序通过设置 JMSPriority 更改了优先级。

异常：

- XMSException

JMSRedelivered - 获取和设置 *JMSRedelivered*

接口：

```
Boolean JMSRedelivered
{
  get;
  set;
}
```

获取是否正在重新传递消息的指示，并指示是否正在重新传递消息。 此指示是由 IMessageConsumer.receive() 调用在接收消息时设置的。

该属性具有以下值：

- `True`（如果正在重新传递消息）。
- `False`（如果未在重新传递消息）。

对于与代理程序的实时连接，该值始终为 `False`。

IMessageProducer.send() 调用在发送消息时会忽略在发送消息之前由 JMSRedelivered 设置的重新传递指示，并且 IMessageConsumer.receive()调用在接收消息时会忽略和替换该指示。 但是，可以使用 JMSRedelivered 来更改已接收的消息指示。

异常：

- XMSException

JMSReplyTo - 获取和设置 *JMSReplyTo*

接口：

```
IDestination JMSReplyTo
{
  get;
  set;
}
```

获取和设置要将消息应答发送到的目标。

该属性的值是一个 Destination 对象，用于表示要将消息应答发送到的目标。 空的 Destination 对象表示不期望收到应答。

异常：

- XMSException

JMSTimestamp - 获取和设置 *JMSTimestamp*

接口：

```
Int64 JMSTimestamp
{
  get;
  set;
}
```

获取和设置消息发送时间。

此时间戳记是由 IMessageProducer.send() 调用在发送消息时设置的，并且将以从 1970 年 1 月 1 日格林威治标准时间 00:00:00 起的毫秒数表示。

对于尚未发送的新创建的消息，此时间戳记为 0，除非发送应用程序通过设置 JMSTimestamp 设置了不同的时间戳记。 对于已收到的消息，此方法将返回由 IMessageProducer.send() 调用在发送消息时设置的时间戳记，除非接收应用程序通过设置 JMSTimestamp 更改了时间戳记。

异常：

- XMSException

注意：

1. 如果未定义时间戳记，那么此方法将返回 0，但不抛出任何异常。

JMSType - 获取和设置 *JMSType*

接口：

```
String JMSType
{
  get;
```

```
  set;
}
```

获取和设置消息类型。

JMSType 的值是用于封装消息类型的字符串。 如果需要进行数据转换，那么该值是转换后的类型。

异常：

- XMSException

PropertyNames - 获取属性

接口：

```
System.Collections.IEnumerator PropertyNames
{
  get;
}
```

获取消息的名称属性的枚举。

异常：

- XMSException

方法

Acknowledge - 确认

接口：

```
void  Acknowledge();
```

确认此消息以及会话先前接收到的所有未确认消息。

如果会话确认方式为 AcknowledgeMode.ClientAcknowledge，那么应用程序可以调用此方法。 如果会话采用任何其他确认方式或者属于事务性会话，那么将忽略此方法调用。

可能会重新传递已接收但未确认的消息。

有关确认消息的更多信息，请参阅第 22 页的『消息确认』。

参数：
None

返回：
无效

异常：

- XMSException
- IllegalStateException

ClearBody - 清除主体

接口：

```
void  ClearBody();
```

清除消息主体。 将不清除头字段和消息属性。

如果应用程序清除了消息主体，那么该主体的状态将与新创建消息中的空主体保持一致。 新创建消息中的空主体的状态取决于消息主体的类型。 有关更多信息，请参阅第 62 页的『XMS 消息的主体』。

无论消息主体处于何种状态，应用程序都可随时清除消息主体。 如果消息主体为只读，那么应用程序写入主体的唯一方式是应用程序先清除该主体。

参数：
None

返回：
无效

异常：

- XMSException

ClearProperties - 清除属性

接口：

```
void  ClearProperties();
```

清除消息属性。 将不清除头字段和消息主体。

如果应用程序清除了消息属性，那么这些属性将变为可读且可写。

无论消息属性处于何种状态，应用程序都可随时清除消息属性。 如果消息属性为只读，那么属性变为可写的唯一方式是应用程序先清除这些属性。

参数：
None

返回：
无效

异常：

- XMSException

PropertyExists - 检查属性是否存在

接口：

```
Boolean PropertyExists(String propertyName);
```

检查消息是否包含具有指定名称的属性。

参数：

propertyName（输入）
用于封装属性名称的 String 对象。

返回：

- `True`（如果消息包含具有指定名称的属性）。
- `False`（如果消息不包含具有指定名称的属性）。

异常：

- XMSException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IMessageConsumer

应用程序使用消息使用者接收向目标发送的消息。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageConsumer
```

要获取 XMS 定义的 MessageConsumer 对象属性列表，请参阅 第 171 页的『MessageConsumer 属性』。

.NET 属性

MessageListener - 获取和设置消息侦听器

接口：

```
MessageListener MessageListener
{
  get;
  set;
}
```

获取向消息使用者注册的消息侦听器，并向消息使用者注册消息侦听器。

如果没有向消息使用者注册任何消息侦听器，那么 MessageListener 为空。 如果已向消息使用者注册了消息侦听器，那么可以通过改为指定空值来取消注册。

有关使用消息侦听器的更多信息，请参阅第 40 页的『.NET 中的消息和异常侦听器』。

异常：

- XMSException

MessageSelector - 获取消息选择器

接口：

```
String MessageSelector
{
  get;
}
```

获取消息使用者的消息选择器。 返回值是用于封装消息选择器表达式的 String 对象。 如果需要进行数据转换，那么该值是转换后的消息选择器表达式。 如果消息使用者不具有消息选择器，那么 MessageSelector 的值是空的 String 对象。

异常：

- XMSException

方法

Close - 关闭消息使用者

接口：

```
void  Close();
```

关闭消息使用者。

如果应用程序尝试关闭已关闭的消息使用者，那么将忽略此调用。

参数：
None

返回：
无效

异常：

- XMSException

Receive - 接收

接口：

```
IMessage Receive();
```

接收消息使用者的下一条消息。 此调用会无限期地等待消息或者直至关闭消息使用者为止。

参数：
None

返回：
指向 Message 对象的指针。 如果在调用等待消息期间关闭了消息使用者，那么此方法将返回一个指向空 Message 对象的指针。

异常：

- XMSException

Receive - 接收（带有等待时间间隔）

接口：

```
IMessage Receive(Int64 delay);
```

接收消息使用者的下一条消息。 此调用仅在指定的时间段内等待消息或者直至关闭消息使用者为止。

参数：

delay（输入）
调用等待消息的时间（以毫秒计）。 如果将等待时间间隔指定为 0，那么此调用会无限期地等待消息。

返回：
指向 Message 对象的指针。 如果在等待时间间隔内没有消息到达，或者如果在调用等待消息期间关闭了消息使用者，那么此方法将返回一个指向空 Message 对象的指针，但不会抛出任何异常。

异常：

- XMSException

ReceiveNoWait - 接收（无等待）

接口：

```
IMessage ReceiveNoWait();
```

如果有立即可用的消息，那么将接收消息使用者的下一条消息。

参数：
None

返回：
指向 Message 对象的指针。 如果没有立即可用的消息，那么此方法将返回一个指向空 Message 对象的指针。

异常：

- XMSException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

MessageEOFException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageEOFException
```

如果 XMS 在应用程序读取字节消息的主体时迁到字节消息流的结尾，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

MessageFormatException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageFormatException
```

如果 XMS 迁到格式无效的消息，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

IMessageListener（委托）

继承层次结构：

None

应用程序使用消息侦听器以异步方式接收消息。

代理人

MessageListener - 消息侦听器

接口：

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

以异步方式将消息传递到消息使用者。

可向连接注册用于实现此委托的方法。

有关使用消息侦听器的更多信息，请参阅第 40 页的『.NET 中的消息和异常侦听器』。

参数：

mesg（输入）

Message 对象。

返回：

无效

MessageNotReadableException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
            |
            +----IBM.XMS.MessageNotReadableException
```

如果应用程序尝试读取仅写入的消息体，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

MessageNotWritableException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
            |
            +----IBM.XMS.MessageNotWritableException
```

如果应用程序尝试写入只读消息的主体，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

IMessageProducer

应用程序使用消息生产者向目标发送消息。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageProducer
```

要获取 XMS 定义的 MessageProducer 对象属性列表，请参阅 第 171 页的『MessageProducer 属性』。

.NET 属性

DeliveryMode - 获取和设置缺省传递方式

接口：

```
DeliveryMode DeliveryMode
{
  get;
  set;
}
```

获取和设置由消息生产者发送的消息的缺省传递方式。

缺省传递方式可为以下值之一：

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

对于与代理程序的实时连接，该值必须为 `DeliveryMode.NonPersistent`。

缺省值为 `DeliveryMode.Persistent`，但与代理程序的实时连接除外，其缺省值为 `DeliveryMode.NonPersistent`。

异常：

- XMSException

Destination - 获取目标

接口：

```
IDestination Destination
{
  get;
}
```

获取消息生产者的目标。

参数：
None

返回：
Destination 对象。 如果消息生产者没有目标，那么此方法将返回空的 Destination 对象。

异常：

- XMSException

DisableMsgID - 获取和设置“禁用消息标识”标志

接口：

```
Boolean DisableMessageID
{
  get;
  set;
}
```

获取接收应用程序是否要求在由消息生产者发送的消息中包含消息标识的指示，并指示接收应用程序是否要求在由消息生产者发送的消息中包含消息标识。

在与队列管理器的连接上或者与代理程序的实时连接上，将忽略此标志。 在与服务集成总线的连接上，支持此标志。

DisabledMsgID 可为以下值：

- `True`（如果接收应用程序不要求在由消息生产者发送的消息中包含消息标识）。
- `False`（如果接收应用程序要求在由消息生产者发送的消息中包含消息标识）。

异常：

- XMSException

DisableMsgTS - 获取和设置“禁用时间戳记”标志

接口：

```
Boolean DisableMessageTimestamp
{
  get;
  set;
}
```

获取接收应用程序是否要求在由消息生产者发送的消息中包含时间戳记的指示，并指示接收应用程序是否要求在由消息生产者发送的消息中包含时间戳记。

在与代理程序的实时连接上，将忽略此标志。 在与队列管理器的连接上或者与服务集成总线的连接上，支持此标志。

DisableMsgTS 可为以下值：

- `True`（如果接收应用程序不要求在由消息生产者发送的消息中包含时间戳记）。
- `False`（如果接收应用程序要求在由消息生产者发送的消息中包含时间戳记）。

返回：

异常：

- XMSException

Priority - 获取和设置缺省优先级

接口：

```
Int32 Priority
{
```

```
  get;
  set;
}
```

获取和设置由消息生产者发送的消息的缺省优先级。

缺省消息优先级的值是范围 0（表示最低优先级）到 9（表示最高优先级）内的整数。

在与代理程序的实时连接上，将忽略消息优先级。

异常：

- XMSException

TimeToLive - 获取和设置缺省生存时间

接口：

```
Int64 TimeToLive
{
  get;
  set;
}
```

获取和设置消息在到期前的缺省存在时间。

从消息生产者发送消息之时开始计算此时间，此时间是缺省生存时间（以毫秒计）。 值 0 表示消息永不到期。

对于与代理程序的实时连接，该值始终为 0。

异常：

- XMSException

方法

Close - 关闭消息生产者

接口：

```
void  Close();
```

关闭消息生产者。

如果应用程序尝试关闭已关闭的消息生产者，那么将忽略此调用。

参数：
None

返回：
无效

异常：

- XMSException

Send - 发送

接口：

```
void Send(IMessage msg) ;
```

将消息发送到创建消息生产者时指定的目标。 使用消息生产者缺省传递方式、优先级和生存时间来发送消息。

参数：

msg（输入）
Message 对象。

返回：
无效

异常：

- XMSException
- MessageFormatException
- InvalidDestinationException

Send - 发送（指定传递方式、优先级和生存时间）

接口：

```
void Send(IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive);
```

将消息发送到创建消息生产者时指定的目标。 使用指定的传递方式、优先级和生存时间来发送消息。

参数：

msg（输入）
Message 对象。

deliveryMode（输入）
消息的传递方式，它必须是以下值之一：

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

对于与代理程序的实时连接，该值必须为 `DeliveryMode.NonPersistent`。

priority（输入）
消息的优先级。 该值可以是范围 0（表示最低优先级）到 9（表示最高优先级）内的整数。 在与代理程序的实时连接上，将忽略该值。

timeToLive（输入）
消息的生存时间（以毫秒计）。 值 0 表示消息永不到期。 对于与代理程序的实时连接，该值必须为 0。

返回：
无效

异常：

- XMSException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Send - 发送（到指定的目标）

接口：

```
void Send(IDestination dest, IMessage msg) ;
```

如果您正在使用在创建消息生产者时没有为其指定目标的消息生产者，请将消息发送到指定目标。 使用消息生产者缺省传递方式、优先级和生存时间来发送消息。

通常，在创建消息生产者时指定目标，但如果未指定，那么在每次发送消息时就必须指定目标。

参数：

dest（输入）
Destination 对象。

msg（输入）
Message 对象。

返回：
无效

异常：

- XMSException
- MessageFormatException
- InvalidDestinationException

Send - 发送（到指定的目标，同时指定传递方式、优先级和生存时间）

接口：

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

如果您正在使用在创建消息生产者时没有为其指定目标的消息生产者，请将消息发送到指定目标。 使用指定的传递方式、优先级和生存时间来发送消息。

通常，在创建消息生产者时指定目标，但如果未指定，那么在每次发送消息时就必须指定目标。

参数：

dest（输入）
Destination 对象。

msg（输入）
Message 对象。

deliveryMode（输入）
消息的传递方式，它必须是以下值之一：

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

对于与代理程序的实时连接，该值必须为 `DeliveryMode.NonPersistent`。

priority（输入）
消息的优先级。 该值可以是范围 `0`（表示最低优先级）到 `9`（表示最高优先级）内的整数。 在与代理程序的实时连接上，将忽略该值。

timeToLive（输入）
消息的生存时间（以毫秒计）。 值 0 表示消息永不到期。 对于与代理程序的实时连接，该值必须为 0。

返回：
无效

异常：

- XMSException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IObjectMessage

对象消息是其主体包含序列化 Java 或 .NET 对象的消息。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
            |
            +----IBM.XMS.IObjectMessage
```

相关参考
对象消息
对象消息的主体包含序列化的 Java 或 .NET 对象。

.NET 属性

Object - 获取对象和将对象设置为字节

接口：

```
System.Object Object
    {
      get;
      set;
    }

    Byte[] GetObject();
```

获取和设置构成对象消息主体的对象。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

- MessageNotWritableException

继承的属性和方法

从 IMessage 接口继承以下属性：

JMSCorrelationID、JMSDeliveryMode、JMSDestination、JMSExpiration、JMSMessageID、JMSPriority、JMSRedelivered、JMSReplyTo、JMSTimestamp、JMSType 或 Properties

从 IMessage 接口继承以下方法：

clearBody, clearProperties, PropertyExists

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IPropertyContext

IPropertyContext 是一个抽象超类，其中包含用于获取和设置属性的方法。 其他类将继承这些方法。

继承层次结构：
None

方法

GetBooleanProperty - 获取布尔值属性

接口：

```
Boolean GetBooleanProperty(String property_name);
```

获取具有指定名称的布尔值属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetByteProperty - 获取字节属性

接口：

```
Byte    GetByteProperty(String property_name) ;
Int16   GetSignedByteProperty(String property_name) ;
```

获取由名称标识的字节属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetBytesProperty - 获取字节数组属性

接口：

```
Byte[]  GetBytesProperty(String property_name) ;
```

获取由名称标识的字节数组属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
数组中的字节数。

线程上下文：
由子类决定

异常：

- XMSException

GetCharProperty - 获取字符属性

接口：

```
Char    GetCharProperty(String property_name) ;
```

获取由名称标识的 2 字节字符属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetDoubleProperty - 获取双精度浮点属性

接口：

```
Double  GetDoubleProperty(String property_name) ;
```

获取由名称标识的双精度浮点属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetFloatProperty - 获取浮点属性

接口：

```
Single  GetFloatProperty(String property_name) ;
```

获取由名称标识的浮点属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetIntProperty - 获取整数属性

接口：

```
Int32   GetIntProperty(String property_name) ;
```

获取由名称标识的整数属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetLongProperty - 获取长整数属性

接口：

```
Int64   GetLongProperty(String property_name) ;
```

获取由名称标识的长整数属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetObjectProperty - 获取对象属性

接口：

```
Object  GetObjectProperty( String property_name) ;
```

获取由名称标识的属性的值和数据类型。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
以下某种对象类型的属性值：

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

线程上下文：
由子类决定

异常：

- XMSException

GetShortProperty - 获取短整数属性

接口：

```
Int16   GetShortProperty(String property_name) ;
```

获取由名称标识的短整数属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
属性的值。

线程上下文：
由子类决定

异常：

- XMSException

GetStringProperty - 获取字符串属性

接口：

```
String  GetStringProperty(String property_name) ;
```

获取由名称标识的字符串属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

返回：
用于封装作为属性值的字符串的 String 对象。 如果需要进行数据转换，那么该值是转换后的字符串。

线程上下文：
由子类决定

异常：

- XMSException

SetBooleanProperty - 设置布尔值属性

接口：

```
void SetBooleanProperty( String property_name,  Boolean value) ;
```

设置由名称标识的布尔值属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetByteProperty - 设置字节属性

接口：

```
void SetByteProperty( String property_name,  Byte value) ;
      void SetSignedByteProperty( String property_name,  Int16 value) ;
```

设置由名称标识的字节属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetBytesProperty - 设置字节数组属性

接口：

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

设置由名称标识的字节数组属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
字节数组形式的属性值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetCharProperty - 设置字符属性

接口：

```
void SetCharProperty( String property_name,  Char value) ;
```

设置由名称标识的 2 字节字符属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetDoubleProperty - 设置双精度浮点属性

接口：

```
void SetDoubleProperty( String property_name,  Double value) ;
```

设置由名称标识的双精度浮点属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetFloatProperty - 设置浮点属性

接口：

```
void SetFloatProperty( String property_name,  Single value) ;
```

设置由名称标识的浮点属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetIntProperty - 设置整数属性

接口：

```
void SetIntProperty( String property_name,  Int32 value) ;
```

设置由名称标识的整数属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetLongProperty - 设置长整数属性

接口：

```
void SetLongProperty( String property_name,  Int64 value) ;
```

设置由名称标识的长整数属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetObjectProperty - 设置对象属性

接口：

```
void SetObjectProperty( String property_name,  Object value) ;
```

设置由名称标识的属性的值和数据类型。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

objectType（输入）
以下某种对象类型的属性值：

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value（输入）
字节数组形式的属性值。

length（输入）
数组中的字节数。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetShortProperty - 设置短整数属性

接口：

```
void SetShortProperty( String property_name,  Int16 value) ;
```

设置由名称标识的短整数属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
属性的值。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

SetStringProperty - 设置字符串属性

接口：

```
void SetStringProperty( String property_name,  String value);
```

设置由名称标识的字符串属性的值。

参数：

property_name（输入）
用于封装属性名称的 String 对象。

value（输入）
用于封装作为属性值的字符串的 String 对象。

返回：
无效

线程上下文：
由子类决定

异常：

- XMSException
- MessageNotWritableException

IQueueBrowser

应用程序使用队列浏览器来浏览队列中的消息，而不将其除去。

继承层次结构：

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
   |
   +----IBM.XMS.IQueueBrowser
```

.NET 属性

MessageSelector - 获取消息选择器

接口：

```
String MessageSelector
```

```
{
  get;
}
```

获取队列浏览器的消息选择器。

消息选择器是用于封装消息选择器表达式的 String 对象。 如果需要进行数据转换，那么该值是转换后的消息选择器表达式。 如果队列浏览器不具有消息选择器，那么此方法将返回空的 String 对象。

异常：

- XMSException

Queue - 获取队列

接口：

```
IDestination Queue
{
  get;
}
```

以表示队列的 Destination 对象形式，获取与队列浏览器相关联的队列。

异常：

- XMSException

方法

Close - 关闭队列浏览器

接口：

```
void  Close();
```

关闭队列浏览器。

如果应用程序尝试关闭已关闭的队列浏览器，那么将忽略此调用。

参数：
None

返回：
无效

异常：

- XMSException

GetEnumerator - 获取消息

接口：

```
IEnumerator GetEnumerator();
```

获取队列中消息的列表。

此方法将返回用于封装 Message 对象列表的枚举符。 Message 对象的顺序与从队列中检索消息的顺序相同。 然后，应用程序可以使用该枚举符来依次浏览各条消息。

在将消息放入队列中以及从队列中除去消息时会动态更新该枚举符。 每次应用程序调用 IEnumerator.MoveNext() 来浏览队列中的下一条消息时，该消息会反映队列的当前内容。

如果应用程序针对队列浏览器多次调用了此方法，那么每次调用都会返回一个新的枚举符。 因此，应用程序可以使用多个枚举符来浏览队列中的消息，并维护队列内的多个位置。

参数：
None

返回：
Iterator 对象。

异常：

- XMSException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

请求者

应用程序可以使用请求程序来发送请求消息，然后等待接收应答。

继承层次结构：
None

构造函数

Requestor - 创建请求程序

接口：

```
Requestor(ISession sess, IDestination dest);
```

创建请求程序。

参数：

sess（输入）
Session 对象。 此会话不能是事务性会话，并且必须使用以下确认方式之一：

```
AcknowledgeMode.AutoAcknowledge
AcknowledgeMode.DupsOkAcknowledge
```

dest（输入）
表示应用程序可将请求消息发送到的目标的 Destination 对象。

线程上下文：
与请求程序相关联的会话

异常：

- XMSException

方法

Close - 关闭请求程序

接口：

```
void  Close();
```

关闭请求程序。

如果应用程序尝试关闭已关闭的请求程序，那么将忽略此调用。

注： 在应用程序关闭请求程序时，不会同时关闭相关的会话。 从这方面讲，XMS 与 JMS 的行为方式不同。

参数：
None

返回：
无效

线程上下文：
任何

异常：

- XMSException

Request - 请求响应

接口：

```
IMessage Request(IMessage requestMessage);
```

发送请求消息，然后等待接收来自负责接收请求消息的应用程序的应答。

此方法调用将会停滞，直至收到应答或该会话结束（以两者中较早的一个为准）。

参数：

requestMessage（输入）
用于封装请求消息的 Message 对象。

返回：
指向用于封装应答消息的 Message 对象的指针。

线程上下文：
与请求程序相关联的会话

异常：

- XMSException

ResourceAllocationException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.ResourceAllocationException
```

如果 XMS 无法分配方法所需的资源，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

SecurityException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.SecurityException
```

如果为认证应用程序而提供的用户标识和密码被拒绝，那么 XMS 将抛出此异常。 如果权限检查失败并导致方法无法完成，那么 XMS 也会抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

ISession

会话是用于发送和接收消息的单线程上下文。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.ISession
```

要获取 XMS 定义的 Session 对象属性列表，请参阅 第 172 页的『Session 属性』。

.NET 属性

AcknowledgeMode - 获取确认方式

接口：

```
AcknowledgeMode AcknowledgeMode
    {
      get;
    }
```

获取会话的确认方式。

在创建会话时指定确认方式。

如果会话不是事务性会话，确认方式将为以下值之一：

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

有关确认方式的更多信息，请参阅第 22 页的『消息确认』。

事务性会话没有确认方式。 如果会话是事务性会话，那么此方法将改为返回 `AcknowledgeMode.SessionTransacted`。

异常：

- XMSException

Transacted - 确定是否为事务性

接口：

```
Boolean Transacted
    {
      get;
    }
```

确定会话是否为事务性会话。

Transacted 可为：

- `True`（如果会话是事务性会话）。
- `False`（如果会话不是事务性会话）。

对于与代理程序的实时连接，此方法始终返回 `False`。

异常：

- XMSException

方法

Close - 关闭会话

接口：

```
void Close();
```

关闭会话。 如果会话是事务性会话，那么将回滚执行中的任何事务。

如果应用程序尝试关闭已关闭的会话，那么将忽略此调用。

参数：
None

返回：
无效

线程上下文：
任何

异常：

- XMSException

Commit - 落实

接口：

```
void Commit();
```

落实当前事务中处理的所有消息。

会话必须是事务性会话。

参数：
None

返回：
无效

异常：

- XMSException
- IllegalStateException
- TransactionRolledBackException

CreateBrowser - 创建队列浏览器

接口：

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

为指定的队列创建队列浏览器。

参数：

queue（输入）
表示队列的 Destination 对象。

返回：
QueueBrowser 对象。

异常：

- XMSException
- InvalidDestinationException

CreateBrowser - 创建队列浏览器（使用消息选择器）

接口：

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

使用消息选择器为指定的队列创建队列浏览器。

参数：

queue（输入）
表示队列的 Destination 对象。

selector（输入）
用于封装消息选择器表达式的 String 对象。 只会将其属性与消息选择器表达式相匹配的消息传递到队列浏览器。

空的 String 对象表示该队列浏览器没有消息选择器。

返回：
QueueBrowser 对象。

异常：

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateBytesMessage - 创建字节消息

接口：

```
IBytesMessage CreateBytesMessage();
```

创建字节消息。

参数：

None

返回：

BytesMessage 对象。

异常：

- XMSException
- IllegalStateException（已关闭会话）

CreateConsumer - 创建使用者

接口：

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

为指定的目标创建消息使用者。

参数：

dest（输入）

Destination 对象。

返回：

MessageConsumer 对象。

异常：

- XMSException
- InvalidDestinationException

CreateConsumer - 创建使用者（使用消息选择器）

接口：

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector) ;
```

使用消息选择器为指定的目标创建消息使用者。

参数：

dest（输入）

Destination 对象。

selector（输入）

用于封装消息选择器表达式的 String 对象。 只会将其属性与消息选择器表达式相匹配的消息传递到消息使用者。

空的 String 对象表示该消息使用者没有消息选择器。

返回：

MessageConsumer 对象。

异常：

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateConsumer - 创建使用者（使用消息选择器和本地消息标志）

接口：

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector,
                                Boolean noLocal) ;
```

使用消息选择器为指定的目标创建消息使用者，并在目标为主题时指定消息使用者是否接收由其自己的连接发布的消息。

参数：

dest（输入）
Destination 对象。

selector（输入）
用于封装消息选择器表达式的 String 对象。 只会将其属性与消息选择器表达式相匹配的消息传递到消息使用者。

空的 String 对象表示该消息使用者没有消息选择器。

noLocal（输入）
值 `True` 表示消息使用者不接收由其自己的连接发布的消息。 值 `False` 表示消息使用者将接收由其自己的连接发布的消息。 缺省值为 `False`。

返回：
MessageConsumer 对象。

异常：

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateDurableSubscriber - 创建持久订户

接口：

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription) ;
```

为指定的主题创建持久订户。

对于与代理程序的实时连接，此方法无效。

有关持久订户的更多信息，请参阅第 28 页的『持久订户』。

参数：

dest（输入）
表示主题的 Destination 对象。 主题不能是临时主题。

subscription（输入）
用于封装可标识持久预订的名称的 String 对象。 在连接的客户机标识内，此名称必须是唯一的。

返回：
表示持久订户的 MessageConsumer 对象。

异常：

- XMSException
- InvalidDestinationException

CreateDurableSubscriber - 创建持久订户（使用消息选择器和本地消息标志）

接口：

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription,
                                         String selector,
                                         Boolean noLocal) ;
```

使用消息选择器为指定的主题创建持久订户，并指定持久订户是否接收由其自己的连接发布的消息。

对于与代理程序的实时连接，此方法无效。

有关持久订户的更多信息，请参阅第 28 页的『持久订户』。

参数：

dest（输入）
表示主题的 Destination 对象。 主题不能是临时主题。

subscription（输入）
用于封装可标识持久预订的名称的 String 对象。 在连接的客户机标识内，此名称必须是唯一的。

selector（输入）
用于封装消息选择器表达式的 String 对象。 只会将其属性与消息选择器表达式相匹配的消息传递到持久订户。

空的 String 对象表示该持久订户没有消息选择器。

noLocal（输入）
值 `True` 表示持久订户不接收由其自己的连接发布的消息。 值 `False` 表示持久订户将接收由其自己的连接发布的消息。 缺省值为 `False`。

返回：
表示持久订户的 MessageConsumer 对象。

异常：

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateMapMessage - 创建映射消息

接口：

```
IMapMessage CreateMapMessage();
```

创建映射消息。

参数：
None

返回：
MapMessage 对象。

异常：

- XMSException
- IllegalStateException（已关闭会话）

CreateMessage - 创建消息

接口：

```
IMessage CreateMessage();
```

创建不含主体的消息。

参数：
None

返回：
Message 对象。

异常：

- XMSException
- IllegalStateException（已关闭会话）

CreateObjectMessage - 创建对象消息

接口：

```
IObjectMessage CreateObjectMessage();
```

创建对象消息。

参数：
None

返回：
ObjectMessage 对象。

异常：

- XMSException
- IllegalStateException（已关闭会话）

CreateProducer - 创建生产者

接口：

```
IMessageProducer CreateProducer(IDestination dest) ;
```

创建要将消息发送到指定目标的消息生产者。

参数：

dest（输入）
Destination 对象。

如果指定了空的 Destination 对象，那么将创建不含目标的消息生产者。 在这种情况下，应用程序在每次使用消息生产者发送消息时都必须指定目标。

返回：
MessageProducer 对象。

异常：

- XMSException
- InvalidDestinationException

CreateQueue - 创建队列

接口：

```
IDestination CreateQueue(String queue) ;
```

创建表示消息传递服务器中的队列的 Destination 对象。

此方法不会在消息传递服务器中创建队列。 您必须在应用程序调用此方法之前创建队列。

参数：

queue（输入）
用于封装队列名称的 String 对象，或用于封装可标识队列的统一资源标识 (URI) 的 String 对象。

返回：
表示队列的 Destination 对象。

异常：

- XMSException

CreateStreamMessage - 创建流消息

接口：

```
IStreamMessage CreateStreamMessage();
```

创建流消息。

参数：
None

返回：
StreamMessage 对象。

异常：

- XMSException
- XMS_ILLEGAL_STATE_EXCEPTION

CreateTemporaryQueue - 创建临时队列

接口：

```
IDestination CreateTemporaryQueue() ;
```

创建临时队列。

临时队列的作用域是连接。 只有通过连接创建的会话才可以使用临时队列。

临时队列会一直保留，直至将其显式删除或连接中断（以两者中较早的一个为准）。

有关临时队列的更多信息，请参阅第 27 页的『临时目标』。

参数：
None

返回：
表示临时队列的 Destination 对象。

异常：

- XMSException

CreateTemporaryTopic - 创建临时主题

接口：

```
IDestination CreateTemporaryTopic() ;
```

创建临时主题。

临时主题的作用域是连接。 只有通过连接创建的会话才可以使用临时主题。

临时主题会一直保留，直至将其显式删除或连接中断（以两者中较早的一个为准）。

有关临时主题的更多信息，请参阅第 27 页的『临时目标』。

参数：
None

返回：
表示临时主题的 Destination 对象。

异常：

- XMSException

CreateTextMessage - 创建文本消息

接口：

```
ITextMessage CreateTextMessage();
```

创建包含空主体的文本消息。

参数：
None

返回：
TextMessage 对象。

异常：

- XMSException

CreateTextMessage - 创建文本消息（已初始化）

接口：

```
ITextMessage CreateTextMessage(String initialValue);
```

创建已使用指定文本初始化消息主体的文本消息。

参数：

initialValue（输入）
用于封装可用来初始化文本消息主体的文本的 String 对象。

None

返回：
TextMessage 对象。

异常：

- XMSException

CreateTopic - 创建主题

接口：

```
IDestination CreateTopic(String topic) ;
```

创建表示主题的 Destination 对象。

参数：

topic（输入）
用于封装主题名称的 String 对象，或用于封装可标识主题的统一资源标识 (URI) 的 String 对象。

返回：
表示主题的 Destination 对象。

异常：

- XMSException

Recover - 恢复

接口：

```
void Recover();
```

恢复会话。 消息传递已停止，然后通过最早的未确认消息来重新启动。

会话不能是事务性会话。

有关恢复会话的更多信息，请参阅第 22 页的『消息确认』。

参数：
None

返回：
无效

异常：

- XMSException
- IllegalStateException

Rollback - 回滚

接口：

```
void Rollback();
```

回滚当前事务中处理的所有消息。

会话必须是事务性会话。

参数：
None

返回：
无效

异常：

- XMSException
- IllegalStateException

Unsubscribe - 取消预订

接口：

```
void Unsubscribe(String subscription);
```

删除持久预订。 消息传递服务器将删除其维护的持久预订记录，并且不会向持久订户再发送任何消息。

在以下任一情况下，应用程序无法删除持久预订：

- 该持久预订存在活动的消息使用者时
- 使用的消息是暂挂事务的一部分时
- 未确认所使用的消息时

对于与代理程序的实时连接，此方法无效。

参数：

subscription（输入）
用于封装可标识持久预订的名称的 String 对象。

返回：
无效

异常：

- XMSException
- InvalidDestinationException
- IllegalStateException

继承的属性和方法

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IStreamMessage

流消息是消息主体由一连串值组成的消息，其中每个值都有关联的数据类型。 将按顺序读写主体的内容。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
```

```
           |
           +----IBM.XMS.IStreamMessage
```

在应用程序从消息流中读取值时，该值可由 XMS 转换为其他数据类型。 有关这种格式的隐式转换的更多信息，请参阅第 65 页的『流消息』。

相关参考
流消息
流消息的主体包含一连串值，其中每个值都具有相关的数据类型。

方法

ReadBoolean - 读取布尔值

接口：

```
Boolean ReadBoolean();
```

从消息流中读取布尔值。

参数：
None

返回：
读取的布尔值。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - 读取单个字节

接口：

```
Int16   ReadSignedByte();
    Byte    ReadByte();
```

从消息流中读取有符号的 8 位整数。

参数：
None

返回：
读取的字节。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - 读取多个字节

接口：

```
Int32   ReadBytes(Byte[] array);
```

从消息流中读取字节数组。

参数：

array（输入）
包含所读取的字节数组和缓冲区长度（以字节计）的缓冲区。

如果数组中的字节数小于或等于缓冲区长度，那么会将整个数组读入到缓冲区中。 如果数组中的字节数大于缓冲区长度，那么会用数组的部分内容填满缓冲区，并且内部光标会标记要读取的下一个字节的位置。 随后的一个 readBytes() 调用会从光标当前位置开始读取数组中的字节。

如果在输入上指定空指针，那么此调用将跳过字节数组，而不读取该数组。

返回：
读入到缓冲区的字节数。 如果部分填充缓冲区，那么该值小于缓冲区长度，这表明不存在待读取的剩余字节。 如果在调用前数组中不存在待读取的剩余字节，那么该值为 XMSC_END_OF_BYTEARRAY。

如果在输入上指定空指针，那么此方法不返回任何值。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadChar - 读取字符

接口：

```
Char    ReadChar();
```

从消息流中读取 2 字节字符。

参数：
None

返回：
读取的字符。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - 读取双精度浮点数

接口：

```
Double  ReadDouble();
```

从消息流中读取 8 字节双精度浮点数。

参数：
None

返回：
读取的双精度浮点数。

异常：

- XMSException

- MessageNotReadableException
- MessageEOFException

ReadFloat - 读取浮点数

接口：

```
Single  ReadFloat();
```

从消息流中读取 4 字节浮点数。

参数：
None

返回：
读取的浮点数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - 读取整数

接口：

```
Int32   ReadInt();
```

从消息流中读取有符号的 32 位整数。

参数：
None

返回：
读取的整数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - 读取长整数

接口：

```
Int64   ReadLong();
```

从消息流中读取有符号的 64 位整数。

参数：
None

返回：
读取的长整数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadObject - 读取对象

接口：

```
Object  ReadObject();
```

从消息流中读取值并返回其数据类型。

参数：
None

返回：
以下某种对象类型的值：

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

异常：
XMSException

ReadShort - 读取短整数

接口：

```
Int16   ReadShort();
```

从消息流中读取有符号的 16 位整数。

参数：
None

返回：
读取的短整数。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadString - 读取字符串

接口：

```
String  ReadString();
```

从消息流中读取字符串。 如果需要，XMS 会将该字符串中的字符转换为本地代码页。

参数：
None

返回：
用于封装读取的字符串的 String 对象。 如果需要进行数据转换，那么该值是转换后的字符串。

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - 重置

接口：

```
void  Reset();
```

将消息主体置于只读方式，并将光标重新定位在消息流的开始位置。

参数：
None

返回：
无效

异常：

- XMSException
- MessageNotReadableException
- MessageEOFException

WriteBoolean - 写入布尔值

接口：

```
void  WriteBoolean(Boolean value);
```

将布尔值写入消息流中。

参数：

value（输入）
要写入的布尔值。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteByte - 写入单个字节

接口：

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

将一个字节写入消息流中。

参数：

value（输入）
要写入的字节。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteBytes - 写入多个字节

接口：

```
void  WriteBytes(Byte[] value);
```

将字节数组写入消息流中。

参数：

value（输入）
要写入的字节数组。

length（输入）
数组中的字节数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteChar - 写入字符

接口：

```
void  WriteChar(Char value);
```

将一个字符作为 2 个字节（首先是高位字节）写入消息流中。

参数：

value（输入）
要写入的字符。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteDouble - 写入双精度浮点数

接口：

```
void  WriteDouble(Double value);
```

将双精度浮点数转换为长整数，并将此长整数作为 8 个字节（首先是高位字节）写入消息流中。

参数：

value（输入）
要写入的双精度浮点数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteFloat - 写入浮点数

接口：

```
void  WriteFloat(Single value);
```

将浮点数转换为整数，并将此整数作为 4 个字节（首先是高位字节）写入消息流中。

参数：

value（输入）
要写入的浮点数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteInt - 写入整数

接口：

```
void  WriteInt(Int32 value);
```

将整数作为 4 个字节（首先是高位字节）写入消息流中。

参数：

value（输入）
要写入的整数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteLong - 写入长整数

接口：

```
void  WriteLong(Int64 value);
```

将长整数作为 8 个字节（首先是高位字节）写入消息流中。

参数：

value（输入）
要写入的长整数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteObject - 写入对象

接口：

```
void  WriteObject(Object value);
```

将指定数据类型的值写入消息流中。

参数：

objectType（输入）
以下某种对象类型的值：

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value（输入）
包含要写入的值的字节数组。

length（输入）
数组中的字节数。

返回：
无效

异常：

- XMSException

WriteShort - 写入短整数

接口：

```
void  WriteShort(Int16 value);
```

将短整数作为 2 个字节（首先是高位字节）写入消息流中。

参数：

value（输入）
要写入的短整数。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

WriteString - 写入字符串

接口：

```
void  WriteString(String value);
```

将字符串写入消息流中。

参数：

value（输入）
用于封装待写入字符串的 String 对象。

返回：
无效

异常：

- XMSException
- MessageNotWritableException

继承的属性和方法

从 IMessage 接口继承以下属性：

JMSCorrelationID、JMSDeliveryMode、JMSDestination、JMSExpiration、JMSMessageID、JMSPriority、JMSRedelivered、JMSReplyTo、JMSTimestamp、JMSType 或 Properties

从 IMessage 接口继承以下方法：

clearBody, clearProperties, PropertyExists

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、

GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

ITextMessage

文本消息是消息主体由字符串组成的消息。

继承层次结构：

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.ITextMessage
```

相关参考

文本消息
文本消息的主体包含一个字符串。

.NET 属性

Text - 获取和设置文本

接口：

```
String Text
{
  get;
  set;
}
```

获取和设置用于构成文本消息主体的字符串。

如果需要，XMS 会将该字符串中的字符转换为本地代码页。

异常：

- XMSException
- MessageNotReadableException
- MessageNotWritableException
- MessageEOFException

继承的属性和方法

从 IMessage 接口继承以下属性：

JMSCorrelationID、JMSDeliveryMode、JMSDestination、JMSExpiration、JMSMessageID、JMSPriority、JMSRedelivered、JMSReplyTo、JMSTimestamp、JMSType 或 Properties

从 IMessage 接口继承以下方法：

clearBody, clearProperties, PropertyExists

从 IPropertyContext 接口继承以下方法：

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

TransactionInProgressException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionInProgressException
```

XMS 在应用程序请求由于事务正在进行而无效的操作时抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

TransactionRolledBackException

继承层次结构：

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionRolledBackException
```

如果应用程序调用 Session.commit() 以落实当前事务，但该事务随后回滚，那么 XMS 将抛出此异常。

继承的属性和方法

从 XMSException 接口继承以下方法：

GetErrorCode 或 GetLinkedException

XMSException

如果 XMS 在处理对 .NET 方法的调用时检测到错误，那么 XMS 将抛出异常。 异常是用于封装错误相关信息的对象。

继承层次结构：

```
System.Exception
   |
   +----IBM.XMS.XMSException
```

存在不同类型的 XMS 异常，而 XMSException 对象只是一种类型的异常。 但是，XMSException 类是其他 XMS 异常类的超类。 在没有任何其他类型的异常适用的情况下，XMS 会抛出 XMSException 对象。

.NET 属性

ErrorCode - 获取错误代码

接口：

```
public String ErrorCode
{
  get {return errorCode_;}
}
```

获取错误代码。

异常：

- XMSException

LinkedException - 获取链接异常

接口：

```
public Exception LinkedException
{
  get { return linkedException_;}
  set { linkedException_ = value;}
}
```

获取异常链中的下一个异常。

如果链中没有其他异常，那么此方法将返回空值。

异常：

- XMSException

XMSFactoryFactory

如果应用程序没有使用受管对象，请使用此类来创建连接工厂、队列和主题。

继承层次结构：

None

.NET 属性

Metadata - 检索元数据

接口：

```
IConnectionMetaData MetaData
```

获取 XMSFactoryFactory 对象的连接类型对应的元数据。

异常：

None

方法

CreateConnectionFactory - 创建连接工厂

接口：

```
IConnectionFactory CreateConnectionFactory();
```

创建已声明类型的 ConnectionFactory 对象。

参数：

None

返回：

ConnectionFactory 对象。

异常：

- XMSException

CreateQueue - 创建队列

接口：

```
IDestination CreateQueue(String name);
```

创建表示消息传递服务器中的队列的 Destination 对象。

此方法不会在消息传递服务器中创建队列。 您必须在应用程序调用此方法之前创建队列。

参数：

name（输入）
用于封装队列名称的 String 对象，或用于封装可标识队列的统一资源标识 (URI) 的 String 对象。

返回：
表示队列的 Destination 对象。

异常：

- XMSException

CreateTopic - 创建主题

接口：

```
IDestination CreateTopic(String name);
```

创建表示主题的 Destination 对象。

参数：

name（输入）
用于封装主题名称的 String 对象，或用于封装可标识主题的统一资源标识 (URI) 的 String 对象。

返回：
表示主题的 Destination 对象。

异常：

- XMSException

GetInstance - 获取 *XMSFactoryFactory* 实例

接口：

```
static XMSFactoryFactory GetInstance(int connectionType);
```

创建 XMSFactoryFactory 的实例。 XMS 应用程序使用 XMSFactoryFactory 对象获取与所需协议类型对应的 ConnectionFactory 对象的引用。 然后，此 ConnectionFactory 对象可以仅针对该协议类型建立连接。

参数：

connectionType（输入）
ConnectionFactory 对象要建立的连接的连接类型。

- XMSC.CT_WPM

- XMSC.CT_RTT
- XMSC.CT_WMQ

返回：
专用于已声明连接类型的 XMSFactoryFactory 对象。

异常：

- NotSupportedException

XMS 对象的属性

本章记录了 XMS 定义的对象属性。

本章包含以下部分：

- 第 159 页的『Connection 属性』
- 第 160 页的『ConnectionFactory 属性』
- 第 164 页的『ConnectionMetaData 属性』
- 第 165 页的『Destination 属性』
- 第 167 页的『InitialContext 属性』
- 第 167 页的『Message 属性』
- 第 171 页的『MessageConsumer 属性』
- 第 171 页的『MessageProducer 属性』
- 第 172 页的『Session 属性』

每个部分列出了一种指定类型对象的属性，并提供了每个属性的简短描述。

本章还包含第 172 页的『属性定义』 部分，其中提供了每个属性的定义。

如果应用程序针对本章中的所述对象定义了自己的属性，那么这不会导致错误，但可能导致出现无法预测的结果。

注： 此部分中的属性名称和值以 XMSC.*NAME* 格式显示，这是用于 C 和 C++ 的格式。但是，在 .NET 中，属性名称的格式可以是 XMSC.*NAME* 或 XMSC_*NAME*，具体取决于您使用属性名称的方式：

- 如果要指定属性，那么属性名称的格式必须为 XMSC.*NAME* ，如以下示例中所示：

```
cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
```

- 如果要指定字符串，那么属性名称的格式必须为 XMSC_*NAME* ，如以下示例中所示：

```
cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
```

  在 .NET 中，属性名称和值作为 XMSC 类中的常量提供。 这些常量标识字符串并将由任何 XMS .NET 应用程序使用。 如果您正在使用这些预定义的常量，那么属性名称和值的格式为 XMSC.*NAME*，例如，您将使用 XMSC.USERID，而不是 XMSC_USERID。

数据类型也采用用于 C/C + + 的格式。您可以在第 37 页的『.NET 的数据类型』中找到 .NET 的相应值。

相关概念
构建自己的应用程序
您可以像构建样本应用程序那样构建自己的应用程序。

相关参考
.NET 接口
本部分记录了 .NET 类接口及其属性和方法。

Connection 属性

下面概括了 Connection 对象属性，并提供了指向更详细参考信息的链接。

表 *25: Connection* 属性	
属性的名称	**描述**
第 201 页的『XMSC_WMQ_RESOLVED_QUEUE_MANAGER』	此属性用于获取与其相连的队列管理器的名称。
第 202 页的『XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID』	在连接后使用队列管理器标识填充此属性。
XMSC_WPM_CONNECTION_PROTOCOL	到消息传递引擎的连接所使用的通信协议。 该属性为只读。
XMSC_WPM_HOST_NAME	包含应用程序连接到的消息传递引擎的系统的主机名或 IP 地址。 该属性为只读。
XMSC_WPM_ME_NAME	应用程序连接到的消息传递引擎的名称。 该属性为只读。
XMSC_WPM_PORT	应用程序连接到的消息传递引擎所侦听的端口号。 该属性为只读。

Connection 对象还有一些只读属性，这些属性派生自用于创建连接的连接工厂的属性。 这些属性不仅派生自在创建连接时设置的连接工厂属性，还派生自未设置的属性的缺省值。 这些属性仅包括适用于与应用程序相连的消息传递服务器类型的属性。 这些属性的名称与连接工厂属性的名称相同。

ConnectionFactory 属性

下面概括了 ConnectionFactory 对象属性，并提供了指向更详细参考信息的链接。

表 *26: ConnectionFactory* 属性	
属性的名称	**描述**
第 180 页的『XMSC_ASYNC_EXCEPTIONS』	此属性确定仅在连接中断时还是在 XMS API 调用出现异步异常时，XMS 通知 ExceptionListener。 此属性适用于通过该 ConnectionFactory 创建的且已注册 ExceptionListener 的所有连接。
XMSC_CLIENT_ID	连接的客户机标识。
XMSC_CONNECTION_TYPE	与应用程序相连的消息传递服务器的类型。
XMSC_PASSWORD	可用于在应用程序尝试连接到消息传递服务器时对其进行认证的密码。
第 185 页的『XMSC_RTT_BROKER_PING_INTERVAL』	XMS .NET 检查与实时消息传递服务器的连接以检测任何活动前的时间间隔（以毫秒计）。
XMSC_RTT_CONNECTION_PROTOCOL	与代理程序的实时连接所使用的通信协议。
XMSC_RTT_HOST_NAME	运行代理程序的系统的主机名或 IP 地址。
XMSC_RTT_LOCAL_ADDRESS	与代理程序的实时连接所使用的本地网络接口的主机名或 IP 地址。
XMSC_RTT_MULTICAST	连接工厂或目标的多点广播设置。
XMSC_RTT_PORT	代理程序侦听入局请求所使用的端口号。
XMSC_USERID	可用于在应用程序尝试连接到消息传递服务器时对其进行认证的用户标识。

表 *26: ConnectionFactory* 属性 *(*继续*)*	
属性的名称	**描述**
XMSC_WMQ_BROKER_CONTROLQ	代理程序使用的控制队列的名称。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
XMSC_WMQ_BROKER_PUBQ	受代理程序监控且应用程序将其发布的消息发送到的队列的名称。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
XMSC_WMQ_BROKER_QMGR	代理程序连接到的队列管理器的名称。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
XMSC_WMQ_BROKER_SUBQ	非持久消息使用者的订户队列的名称。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
XMSC_WMQ_BROKER_VERSION	应用程序针对连接或目标使用的代理程序类型。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
第 189 页的『XMSC_WMQ_CCDTURL』	统一资源定位符 (URL)，用于标识包含客户机通道定义表的文件的名称和位置并指定该文件的访问方式。
XMSC_WMQ_CHANNEL	连接要使用的通道的名称。
第 190 页的『XMSC_WMQ_CLIENT_RECONNECT_OPTIONS』	此属性可为该工厂所创建的新连接指定客户机重新连接选项。
第 190 页的『XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT』	此属性可指定客户机连接尝试重新连接的持续时间（以秒计）。
XMSC_WMQ_CONNECTION_MODE	应用程序连接到队列管理器所使用的方式。
第 191 页的『XMSC_WMQ_CONNECTION_NAME_LIST』	此属性可指定客户机在中断连接后尝试重新连接的主机。
XMSC_WMQ_FAIL_IF_QUIESCE	当应用程序连接到的队列管理器处于停顿状态时，某些方法调用是否会失败。

表 *26: ConnectionFactory* 属性 *(*继续*)*	
属性的名称	**描述**
XMSC_WMQ_HOST_NAME	运行队列管理器的系统的主机名或 IP 地址。
XMSC_WMQ_LOCAL_ADDRESS	对于到队列管理器的连接，该属性指定要使用的本地网络接口和/或要使用的本地端口/本地端口范围。
XMSC_WMQ_MESSAGE_SELECTION	确定消息选择是由 XMS 客户机完成还是由代理完成。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
XMSC_WMQ_MSG_BATCH_SIZE	在使用异步消息传递时要成批从队列中检索的最大消息数。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
XMSC_WMQ_POLLING_INTERVAL	如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么此值是每个消息侦听器再次尝试从其队列中获取消息前经过的最大时间间隔（以毫秒计）。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
第 199 页的『XMSC_WMQ_PROVIDER_VERSION』	应用程序要连接到的队列管理器的版本、发行版、修订版级别和修订包。
XMSC_WMQ_PORT	队列管理器侦听入局请求的端口号。
XMSC_WMQ_PUB_ACK_INTERVAL	发布者在 XMS 客户机请求代理程序应答之前发布的消息数。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
第 195 页的『XMSC_WMQ_PUT_ASYNC_ALLOWED』	此属性确定是否允许消息生产者使用异步放置来将消息发送到此目标。
XMSC_WMQ_QMGR_CCSID	编码字符集或代码页的标识 (CCSID)，其中在消息队列接口 (MQI) 中定义的字符数据字段在 XMS 客户机与 WebSphere MQ 客户机之间交换。
XMSC_WMQ_QUEUE_MANAGER	要连接的队列管理器的名称。
XMSC_WMQ_RECEIVE_EXIT	标识要运行的通道接收出口。
XMSC_WMQ_RECEIVE_EXIT_INIT	调用通道接收出口时传递到通道接收出口的用户数据。
XMSC_WMQ_SECURITY_EXIT	标识通道安全出口。
XMSC_WMQ_SECURITY_EXIT_INIT	调用通道安全出口时传递到通道安全出口的用户数据。

表 *26: ConnectionFactory* 属性 *(*继续*)*	
属性的名称	**描述**
第 203 页的『XMSC_WMQ_SEND_CHECK_COUNT』	单个非事务性 XMS 会话内两次检查异步放置错误之间允许的 Send 调用次数。
XMSC_WMQ_SEND_EXIT	标识通道发送出口。
XMSC_WMQ_SEND_EXIT_INIT	调用通道发送出口时传递到通道发送出口的用户数据。
第 203 页的『XMSC_WMQ_SHARE_CONV_ALLOWED』	如果通道定义匹配，那么客户机连接是否可以与从同一进程到同一队列管理器的其他顶级 XMS 连接共享其套接字。 根据应用程序开发、维护或运行方面的需要，使用此属性可在不同套接字中完全隔离连接。
XMSC_WMQ_SSL_CERT_STORES	用于保存与队列管理器的 SSL 连接上使用的证书撤销列表 (CRL) 的服务器的位置。
XMSC_WMQ_SSL_CIPHER_SPEC	到队列管理器的安全连接上要使用的 CipherSpec 名称。
XMSC_WMQ_SSL_CIPHER_SUITE	到队列管理器的 TLS 连接上要使用的 CipherSuite 的名称。 协商安全连接时使用的协议取决于指定的 CipherSuite。
XMSC_WMQ_SSL_CRYPTO_HW	下面是连接到客户机系统的加密硬件的配置详细信息。
XMSC_WMQ_SSL_FIPS_REQUIRED	该属性的值用于确定应用程序能否使用符合非 FIPS 标准的密码套件。 如果将该属性设置为 true，那么客户机/服务器连接只能使用 FIPS 算法。
XMSC_WMQ_SSL_KEY_REPOSITORY	用于存储密钥和证书的密钥数据库文件的位置。
XMSC_WMQ_SSL_KEY_RESETCOUNT	KeyResetCount 表示在重新协商密钥之前在 SSL 对话期间发送和接收的未加密字节总数。
XMSC_WMQ_SSL_PEER_NAME	到队列管理器的 SSL 连接上要使用的对等方名称。
XMSC_WMQ_SYNCPOINT_ALL_GETS	是否必须从同步点控制范围内的队列中检索所有消息。
第 209 页的『XMSC_WMQ_TARGET_CLIENT』	
XMSC_WMQ_TEMP_Q_PREFIX	用于构成应用程序创建 XMS 临时队列时创建的 WebSphere MQ 动态队列的名称的前缀。
XMSC_WMQ_TEMP_TOPIC_PREFIX	创建临时主题时， XMS 将生成格式为 "TEMP/TEMPTOPICPREFIX/unique_id" 的主题字符串，或者如果此属性包含缺省值，那么将生成此字符串 "TEMP/unique_id"。 通过指定非空值，可以定义特定的模型队列，以便为在该连接下创建的临时主题的订户创建受管队列。
XMSC_WMQ_TEMPORARY_MODEL	应用程序创建 XMS 临时队列时从中创建动态队列的 WebSphere MQ 模型队列的名称。
XMSC_WPM_BUS_NAME	对于连接工厂，这是应用程序连接到的服务集成总线的名称；对于目标，这是存在目标的服务集成总线的名称。
XMSC_WPM_CONNECTION_PROXIMITY	连接的连接距离设置。
XMSC_WPM_DUR_SUB_HOME	用于管理连接或目标的所有持久预订的消息传递引擎的名称。
XMSC_WPM_LOCAL_ADDRESS	对于到服务集成总线的连接，该属性指定要使用的本地网络接口和/或要使用的本地端口/本地端口范围。
XMSC_WPM_NON_PERSISTENT_MAP	通过连接发送的非持久消息的可靠性级别。

表 *26: ConnectionFactory* 属性 *(*继续*)*	
属性的名称	**描述**
XMSC_WPM_PERSISTENT_MAP	通过连接发送的持久消息的可靠性级别。
XMSC_WPM_PROVIDER_ENDPOINTS	由引导程序服务器的一个或多个端点地址组成的序列。
XMSC_WPM_TARGET_GROUP	消息传递引擎的目标组名称。
XMSC_WPM_TARGET_SIGNIFICANCE	消息传递引擎的目标组的重要性。
XMSC_WPM_TARGET_TRANSPORT_CHAIN	应用程序连接到消息传递引擎时必须使用的入站传输链的名称。
XMSC_WPM_TARGET_TYPE	消息传递引擎的目标组类型。
XMSC_WPM_TEMP_Q_PREFIX	用于构成应用程序创建 XMS 临时队列时在服务集成总线中创建的临时队列的名称的前缀。
XMSC_WPM_TEMP_TOPIC_PREFIX	用于构成应用程序创建的临时主题名称的前缀。

相关概念

ConnectionFactories 和 Connection 对象
ConnectionFactory 对象提供一种模板以供应用程序用于创建 Connection 对象。 应用程序可使用 Connection 对象来创建 Session 对象。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

与 IBM MQ 队列管理器的安全连接
要使 XMS .NET 应用程序能够与 IBM MQ 队列管理器建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

与 WebSphere Application Server service integration bus 消息传递引擎的安全连接
要使 XMS .NET 应用程序能够与 WebSphere Application Server service integration bus 消息传递引擎建立安全连接，必须在 ConnectionFactory 对象中定义相关属性。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

相关参考

受管 ConnectionFactory 对象的必需属性
应用程序创建连接工厂时，必须定义一些属性来与消息传递服务器建立连接。

ConnectionMetaData 属性

下面概括了 ConnectionMetaData 对象属性，并提供了指向更详细参考信息的链接。

表 *27: ConnectionMetaData* 属性	
属性的名称	**描述**
XMSC_JMS_MAJOR_VERSION	XMS 所基于的 JMS 规范的主要版本号。 该属性为只读。
XMSC_JMS_MINOR_VERSION	XMS 所基于的 JMS 规范的次要版本号。 该属性为只读。
XMSC_JMS_VERSION	XMS 所基于的 JMS 规范的版本标识。 该属性为只读。

表 27: *ConnectionMetaData* 属性 (继续)	
属性的名称	**描述**
XMSC_MAJOR_VERSION	XMS 客户机的版本号。 该属性为只读。
XMSC_MINOR_VERSION	XMS 客户机的发行版号。 该属性为只读。
XMSC_PROVIDER_NAME	XMS 客户机的提供者。 该属性为只读。
XMSC_VERSION	XMS 客户机的版本标识。 该属性为只读。

Destination 属性

下面概括了 Destination 对象属性，并提供了指向更详细参考信息的链接。

表 28: *Destination* 属性	
属性的名称	**描述**
XMSC_DELIVERY_MODE	发送到目标的消息的传递方式。
XMSC_PRIORITY	发送到目标的消息的优先级。
XMSC_RTT_MULTICAST	连接工厂或目标的多点广播设置。
XMSC_TIME_TO_LIVE	发送到目标的消息的生存时间。
XMSC_WMQ_BROKER_VERSION	应用程序针对连接或目标使用的代理程序类型。
XMSC_WMQ_CCSID	当 XMS 客户机将消息转发到目标时，消息主体中的字符数据字符串所在的编码字符集或代码页的标识 (CCSID)。
XMSC_WMQ_DUR_SUBQ	正在从目标接收消息的持久订户的订户队列名称。 **注:** 此属性可以与 IBM Message Service Client for .NET 的 V 2.0 配合使用，但对于连接到 IBM WebSphere MQ 7.0 队列管理器的应用程序不起作用，除非连接工厂的 XMSC_WMQ_PROVIDER_VERSION 属性设置为小于 7 的版本号。
XMSC_WMQ_ENCODING	当 XMS 客户机将消息转发到目标时，如何表示消息体中的数字数据。
XMSC_WMQ_FAIL_IF_QUIESCE	当应用程序连接到的队列管理器处于停顿状态时，某些方法调用是否会失败。
第 193 页的『XMSC_WMQ_MESSAGE_BODY』	此属性确定 XMS 应用程序是否将 IBM WebSphere MQ 消息的 MQRFH2 作为消息有效内容的一部分 (即，作为消息体的一部分) 进行处理。
第 193 页的『XMSC_WMQ_MQMD_MESSAGE_CONTEXT』	确定 XMS 应用程序要设置的消息上下文级别。 应用程序必须以相应的上下文权限运行才能使属性生效。
第 194 页的『XMSC_WMQ_MQMD_READ_ENABLED』	此属性确定 XMS 应用程序是否可以抽取 MQMD 字段的值。
第 195 页的『XMSC_WMQ_MQMD_WRITE_ENABLED』	此属性确定 XMS 应用程序是否可以写入 MQMD 字段的值。
第 195 页的『XMSC_WMQ_READ_AHEAD_ALLOWED』	此属性确定是否允许消息使用者和队列浏览器在接收消息之前使用预读功能从此目标获取非事务性的非持久消息并将其放入内部缓冲区。
第 196 页的『XMSC_WMQ_READ_AHEAD_CLOSE_POLICY』	对于正在传递到异步消息侦听器的消息，此属性确定当关闭消息使用者时在内部预读缓冲区中对消息执行的操作。

表 *28: Destination* 属性 *(*继续*)*	
属性的名称	**描述**
第 200 页的『XMSC_WMQ_RECEIVE_CCSID』	用于设置队列管理器消息转换的目标 CCSID 的目标属性。 除非将 XMSC_WMQ_RECEIVE_CONVERSION 设置为 WMQ_RECEIVE_CONVERSION_QMGR，否则将忽略此值。
第 201 页的『XMSC_WMQ_RECEIVE_CONVERSION』	用于确定数据转换即将由队列管理器执行的目标属性。
XMSC_WMQ_TARGET_CLIENT	发送到目标的消息是否包含 MQRFH2 头。
XMSC_WMQ_TEMP_TOPIC_PREFIX	创建临时主题时， XMS 将生成格式为 "TEMP/TEMPTOPICPREFIX/unique_id" 的主题字符串，或者如果此属性包含缺省值，那么将生成此字符串 "TEMP/unique_id"。 通过指定非空值，可以定义特定的模型队列，以便为在该连接下创建的临时主题的订户创建受管队列。
XMSC_WPM_BUS_NAME	对于连接工厂，这是应用程序连接到的服务集成总线的名称；对于目标，这是存在目标的服务集成总线的名称。
XMSC_WPM_TOPIC_SPACE	包含主题的主题空间的名称。

相关概念

ConnectionFactories 和 Connection 对象
ConnectionFactory 对象提供一种模板以供应用程序用于创建 Connection 对象。 应用程序可使用 Connection 对象来创建 Session 对象。

连接到服务集成总线
XMS 应用程序可以通过使用直接 TCP/IP 连接或使用“基于 TCP/IP 的 HTTP”来连接到 WebSphere Application Server 服务集成总线。

目标
XMS 应用程序可使用 Destination 对象来指定所发送消息的目标以及所接收消息的源。

目标通配符
XMS 支持目标通配符，可确保将通配符传递到需要进行匹配的位置。 XMS 可使用的每种服务器类型各有不同的通配符方案。

主题统一资源标识
主题统一资源标识 (URI) 指定主题的名称，还可以指定主题的一个或多个属性。

队列统一资源标识
队列 URI 指定队列的名称，还可以指定队列的一个或多个属性。

临时目标
XMS 应用程序可以创建和使用临时目标。

受管对象的属性映射
为了使应用程序能够使用 IBM MQ JMS 和 WebSphere Application Server 连接工厂以及 Destination 对象定义，必须将从这些定义中检索的属性映射到可在 XMS 连接工厂和目标上设置的对应 XMS 属性。

相关任务

创建受管对象
必须使用相应的管理工具来创建 XMS 应用程序与消息传递服务器建立连接时所需的 ConnectionFactory 和 Destination 对象定义。

相关参考

受管 Destination 对象的必需属性
用于创建目标的应用程序必须在受管 Destination 对象上设置多个属性。

InitialContext 属性

下面概括了 InitialContext 对象属性，并提供了指向更详细参考信息的链接。

表 *29: InitialContext* 属性	
属性的名称	**描述**
XMSC_IC_PROVIDER_URL	用于查找 JNDI 命名目录，因此 COS 命名服务不需要与 Web Service 在同一服务器上。
XMSC_IC_SECURITY_AUTHENTICATION	基于 Java 上下文接口 SECURITY_AUTHENTICATION。 此属性仅适用于 COS 命名上下文。
XMSC_IC_SECURITY_CREDENTIALS	基于 Java 上下文接口 SECURITY_CREDENTIALS。 此属性仅适用于 COS 命名上下文。
XMSC_IC_SECURITY_PRINCIPAL	基于 Java 上下文接口 SECURITY_PRINCIPAL。 此属性仅适用于 COS 命名上下文。
XMSC_IC_SECURITY_PROTOCOL	基于 Java 上下文接口 SECURITY_PROTOCOL 此属性仅适用于 COS 命名上下文。
XMSC_IC_URL	对于 LDAP 和 FileSystem 上下文，这是包含受管对象的存储库的地址。 对于 COS 命名上下文，这是在目录中查找对象的 Web Service 的地址。

相关概念

InitialContext 属性
InitialContext 构造函数的参数中包括受管对象存储库的位置（其指定为统一资源指示符 (URI)）。 如果要使应用程序与该存储库建立连接，那么需要提供的信息可能要多于 URI 中包含的信息。

XMS 初始上下文的 URI 格式
将采用统一资源指示符 (URI) 形式提供受管对象存储库的位置。 URI 的格式取决于上下文类型。

检索受管对象
XMS 使用在创建 InitialContext 对象时提供的地址或 InitialContext 属性中的地址，从存储库中检索受管对象。

相关任务

InitialContext 对象
应用程序必须创建用于与受管对象存储库建立连接的初始上下文，以便检索所需的受管对象。

Message 属性

下面概括了 Message 对象属性，并提供了指向更详细参考信息的链接。

表 *30: Message* 属性	
属性的名称	**描述**
JMS_IBM_CHARACTER_SET	当 XMS 客户机将消息转发到其预期目标时，消息主体中的字符数据字符串所在的编码字符集或代码页的标识 (CCSID)。 在 XMS 中，此属性具有数字值并映射到 CCSID。 但是，此属性基于 JMS 属性，因此，它具有字符串类型值，并映射到表示此数字 CCSID 的 Java 字符集。
JMS_IBM_ENCODING	当 XMS 客户机将消息转发到其预期目标时，如何表示消息主体中的数字数据。
JMS_IBM_EXCEPTIONMESSAGE	用于描述为何将消息发送到异常目标的文本。 该属性为只读。
JMS_IBM_EXCEPTIONPROBLEMDESTINATION	消息在发送到异常目标之前所处目标的名称。

表 30: Message 属性 (继续)	
属性的名称	**描述**
JMS_IBM_EXCEPTIONREASON	用于指示为何将消息发送到异常目标的原因码。
JMS_IBM_EXCEPTIONTIMESTAMP	将消息发送到异常目标的时间。
JMS_IBM_FEEDBACK	用于指示报告消息性质的代码。
JMS_IBM_FORMAT	消息中应用程序数据的性质。
JMS_IBM_LAST_MSG_IN_GROUP	指示消息是否是消息组中的最后一条消息。
JMS_IBM_MSGTYPE	消息的类型。
JMS_IBM_PUTAPPLTYPE	发送消息的应用程序类型。
JMS_IBM_PUTDATE	发送消息的日期。
JMS_IBM_PUTTIME	发送消息的时间。
JMS_IBM_REPORT_COA	请求“到达时确认”报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。
JMS_IBM_REPORT_COD	请求“传递时确认”报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。
JMS_IBM_REPORT_DISCARD_MSG	请求在无法将消息传递到其预期目标时丢弃此消息。
JMS_IBM_REPORT_EXCEPTION	请求异常报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。
JMS_IBM_REPORT_EXPIRATION	请求到期报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。
JMS_IBM_REPORT_NAN	请求负面操作通知报告消息。
JMS_IBM_REPORT_PAN	请求正面操作通知报告消息。
JMS_IBM_REPORT_PASS_CORREL_ID	请求任何报告或应答消息的相关标识与原始消息的相关标识相同。
JMS_IBM_REPORT_PASS_MSG_ID	请求任何报告或应答消息的消息标识与原始消息的消息标识相同。
JMS_IBM_RETAIN	设置此属性可指示队列管理器将消息视作“保留发布”。
JMS_IBM_SYSTEM_MESSAGEID	用于在服务集成总线内唯一识别消息的标识。 该属性为只读。
JMSX_APPID	发送消息的应用程序名称。
JMSX_DELIVERY_COUNT	尝试传递消息的次数。
JMSX_GROUPID	消息所属消息组的标识。
JMSX_GROUPSEQ	消息在消息组中的序号。
JMSX_USERID	与发送消息的应用程序相关联的用户标识。

JMS_IBM_MQMD* 属性

通过 IBM Message Service Client for .NET，客户机应用程序可以使用 API 读/写 MQMD 字段。 它也允许访问 MQ 消息数据。 缺省情况下，已禁用对 MQMD 的访问，必须由应用程序使用 Destination 属性 XMSC_WMQ_MQMD_WRITE_ENABLED 和 XMSC_WMQ_MQMD_READ_ENABLED 来明确进行启用。 这两个属性相互独立。

除 StrucId 和 Version 以外的所有其他 MQMD 字段都作为额外的 Message 对象属性公开，并添加了前缀 JMS_IBM_MQMD。

JMS_IBM_MQMD* 属性的优先级高于其他属性（如上表中所述的 JMS_IBM*）。

发送消息

表示除 StrucId 和 Version 之外的所有 MQMD 字段。 这些属性仅仅是指 MQMD 字段；属性同时出现在 MQMD 和 MQRFH2 头中，不会设置或抽取 MQRFH2 中的版本。 除了 JMS_IBM_MQMD_BackoutCount 之外，以上任意属性都可以设置。 将忽略对 JMS_IBM_MQMD_BackoutCount 设置的任何值。

如果属性具有最大长度限制，而您提供的值过长，则该值会被截断。

对于某些属性，还必须在 Destination 对象上设置 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 属性。 应用程序必须以相应的上下文权限运行才能使属性生效。 如果未将 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 设置为相应的值，那么将忽略此属性值。 如果将 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 设置为相应的值，但您对于队列管理器没有足够的上下文权限，那么将发出异常。 下面是一些要求 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 具有特定值的属性。

以下属性要求将 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 设置为 XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT 或 XMSC_WMQ_MDCTX_SET_ALL_CONTEXT：

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityData

以下属性要求将 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 设置为 XMSC_WMQ_MDCTX_SET_ALL_CONTEXT：

- JMS_IBM_MQMD_PutApplType
- JMS_IBM_MQMD_PutApplName
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginData

接收消息

如果将 XMSC_WMQ_MQMD_READ_ENABLED 属性设置为 true，那么所有这些属性在收到的消息上均可用，而与生产应用程序设置的实际属性无关。 应用程序无法修改所收到消息的属性，除非首先根据 JMS 规范清除了所有属性。 可以在不修改属性的情况下转发已接收的消息。

注: 如果应用程序从将 XMSC_WMQ_MQMD_READ_ENABLED 属性设置为 true 的目标处收到消息，并将其转发到将 XMSC_WMQ_MQMD_WRITE_ENABLED 设置为 true 的目标，那么这会导致将所收到消息的所有 MQMD 字段值复制到转发的消息中。 属性表

表 *31:* 表示 *MQMD* 字段的 *Message* 对象属性

属性	**描述**	**类型**
JMS_IBM_MQMD_REPORT	报告消息的选项	System.Int32
JMS_IBM_MQMD_MSGTYPE	消息类型	System.Int32
JMS_IBM_MQMD_EXPIRY	消息生命周期	System.Int32
JMS_IBM_MQMD_FEEDBACK	反馈或原因码	System.Int32
JMS_IBM_MQMD_ENCODING	消息数据的数字编码	System.Int32
JMS_IBM_MQMD_CODEDCHARSETID	消息数据的字符集标识	System.Int32
JMS_IBM_MQMD_FORMAT	消息数据的格式名称	System.String

表 31: 表示 *MQMD* 字段的 *Message* 对象属性 (继续)

属性	描述	类型
JMS_IBM_MQMD_PRIORITY **注:** 如果为 JMS_IBM_MQMD_PRIORITY 指定了范围 0-9 以外的值，那么此值违反了 JMS 规范。	消息优先级	System.Int32
JMS_IBM_MQMD_PERSISTENCE	消息持久性	System.Int32
JMS_IBM_MQMD_MSGID **注:** JMS 规范声明消息标识必须由 JMS 提供程序进行设置且必须唯一或为空值。 如果为 JMS_IBM_MQMD_MSGID 指定了值，那么此值将复制到 JMSMessageID。 因此，此值不是由 JMS 提供程序设置，并且可能不唯一：此值违反了 JMS 规范。	消息标识	字节数组 **注:** 在消息上使用字节数组属性违反了 JMS 规范。
JMS_IBM_MQMD_CORRELID **注:** 如果为 JMS_IBM_MQMD_CORRELID 指定了以字符串“ID:”开头的值，那么此值违反了 JMS 规范。	相关标识	字节数组 **注:** 在消息上使用字节数组属性违反了 JMS 规范。
JMS_IBM_MQMD_BACKOUTCOUNT	回退计数器	System.Int32
JMS_IBM_MQMD_REPLYTOQ	应答队列的名称	System.String
JMS_IBM_MQMD_REPLYTOQMGR	应答队列管理器的名称	System.String
JMS_IBM_MQMD_USERIDENTIFIER	用户标识	System.String
JMS_IBM_MQMD_ACCOUNTINGTOKEN	记帐标记	字节数组 **注:** 在消息上使用字节数组属性违反了 JMS 规范。
JMS_IBM_MQMD_APPLIDENTITYDATA	与身份有关的应用程序数据	System.String
JMS_IBM_MQMD_PUTAPPLTYPE	放置消息的应用程序的类型	System.Int32
JMS_IBM_MQMD_PUTAPPLNAME	放置消息的应用程序的名称	System.String
JMS_IBM_MQMD_PUTDATE	消息的放置日期	System.String
JMS_IBM_MQMD_PUTTIME	消息放置的时间	System.String
JMS_IBM_MQMD_APPLORIGINDATA	与源有关的应用程序数据	System.String
JMS_IBM_MQMD_GROUPID	组标识	字节数组 **注:** 在消息上使用字节数组属性违反了 JMS 规范。
JMS_IBM_MQMD_MSGSEQNUMBER	组内本地消息的序号	System.Int32
JMS_IBM_MQMD_OFFSET	从逻辑消息开始的物理消息数据偏移量	System.Int32
JMS_IBM_MQMD_MSGFLAGS	消息标志	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	原始消息的长度	System.Int32

请参阅 MQMD，以获取更多详细信息。

示例

以下示例生成的消息将放入 MQMD.UserIdentifier 设置为“JoeBloggs”的队列或主题中。

```
// 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
// ...
```

在设置 JMS_IBM_MQMD_USERIDENTIFIER 之前，需要先设置 XMSC_WMQ_MQMD_MESSAGE_CONTEXT。 有关使用 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 的更多信息，请参阅 Message 对象属性。

同样，可通过在收到消息前将 XMSC_WMQ_MQMD_READ_ENABLED 设置为 true，然后使用消息获取方法（如 getStringProperty）来抽取 MQMD 字段内容。 收到的任何属性均为只读。

此示例将从队列或主题中获取用于保存消息 MQMD.ApplIdentityData 字段值的值字段。

```
// 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 required MQMD field value using a property
System.String value = rcvMsg.getStringProperty(JMS_IBM_MQMD_APPLIDENTITYDATA);
```

MessageConsumer 属性

下面概括了 MessageConsumer 对象属性，并提供了指向更详细参考信息的链接。

表 *32: MessageConsumer* 属性

属性的名称	描述
XMSC_IS_SUBSCRIPTION_MULTICAST	指示是否正在使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 该属性为只读。
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST	指示是否使用具有可靠服务质量的 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 该属性为只读。

请参阅 .NET 中的 IMessageConsumer 属性，以获取更多详细信息。

MessageProducer 属性

下面概括了 MessageProducer 对象属性，并提供了指向更详细参考信息的链接。

请参阅 。IMessageProducer 的 NET 属性以获取更多详细信息。

Session 属性

下面概括了 Session 对象属性，并提供了指向更详细参考信息的链接。

请参阅 。ISession 的 NET 属性 以获取更多详细信息。

属性定义

本部分提供了每个对象属性的定义。

每个属性定义均包含以下信息：

- 该属性的数据类型
- 具有该属性的对象的类型
- 对于 Destination 的属性，可在统一资源标识 (URI) 中使用的名称
- 该属性的更详细的描述
- 该属性的有效值
- 此属性的缺省值

其名称以下列某个前缀开头的属性仅适用于指定类型的连接：

XMSC_RTT
这些属性仅适用于与代理程序的实时连接。 这些属性的名称将定义为头文件 `xmsc_rtt.h` 中的命名常量。

XMSC_WMQ
这些属性仅适用于应用程序连接到 WebSphere MQ 队列管理器的情况。 这些属性的名称将定义为头文件 `xmsc_wmq.h` 中的命名常量。

XMSC_WPM
这些属性仅适用于应用程序连接到 WebSphere 服务集成总线的情况。 这些属性的名称将定义为头文件 `xmsc_wpm.h` 中的命名常量。

除非在定义中另行说明，否则其余属性适用于所有类型的连接。 这些属性的名称将定义为头文件 `xmsc.h` 中的命名常量。 其名称以前缀 JMSX 开头的属性是 JMS 定义的消息属性，而其名称以前缀 JMS_IBM 开头的属性是 IBM 定义的消息属性。 有关消息属性的更多信息，请参阅第 59 页的『XMS 消息的属性』。

除非在定义中另行说明，否则每个属性同时适用于点到点和发布/预订域。

除非属性指定为只读属性，否则应用程序可以获取和设置该属性的值。

JMS_IBM_CHARACTER_SET

数据类型：
System.Int32

相关属性：
消息

当 XMS 客户机将消息转发到其预期目标时，消息主体中的字符数据字符串所在的编码字符集或代码页的标识 (CCSID)。 在 XMS 中，此属性具有数字值并映射到 CCSID。 但是，此属性基于 JMS 属性，因此，它具有字符串类型值，并映射到表示此数字 CCSID 的 Java 字符集。 此属性将覆盖 XMSC_WMQ_CCSID 属性为目标指定的任何 CCSID。

缺省情况下，不设置该属性。

此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_ENCODING

数据类型：
System.Int32

相关属性：
消息

当 XMS 客户机将消息转发到其预期目标时，如何表示消息主体中的数字数据。 此属性将覆盖 XMSC_WMQ_ENCODING 属性为目标指定的任何编码。 此属性指定二进制整数、压缩十进制整数和浮点数的表示法。

此属性的有效值与可在消息描述符的 **Encoding** 字段中指定的值相同。

应用程序可使用以下命名常量来设置此属性：

命名常量	含义
MQENC_INTEGER_NORMAL	标准整数编码
MQENC_INTEGER_REVERSED	反向整数编码
MQENC_DECIMAL_NORMAL	标准压缩十进制编码
MQENC_DECIMAL_REVERSED	反向压缩十进制编码
MQENC_FLOAT_IEEE_NORMAL	标准 IEEE 浮点编码
MQENC_FLOAT_IEEE_REVERSED	反向 IEEE 浮点编码
MQENC_FLOAT_S390	z/OS 体系结构浮点编码
MQENC_NATIVE	本机编码

要构成该属性的值，应用程序可添加其中的三个常量，如下所示：

- 名称以 MQENC_INTEGER 开头的常量，用于指定二进制整数表示法
- 名称以 MQENC_DECIMAL 开头的常量，用于指定压缩十进制整数表示法
- 名称以 MQENC_FLOAT 开头的常量，用于指定浮点数表示法

或者，应用程序可将此属性设置为 MQENC_NATIVE（其值与环境相关）。

缺省情况下，不设置该属性。

此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_EXCEPTIONMESSAGE

数据类型：
字符串

相关属性：
消息

用于描述为何将消息发送到异常目标的文本。 该属性为只读。

此属性仅适用于应用程序连接到服务集成总线并从异常目标接收消息的情况。

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

数据类型：
字符串

相关属性：
消息

消息在发送到异常目标之前所处目标的名称。

此属性仅适用于应用程序连接到服务集成总线并从异常目标接收消息的情况。

JMS_IBM_EXCEPTIONREASON

数据类型：
System.Int32

相关属性：
消息

用于指示为何将消息发送到异常目标的原因码。

此属性仅适用于应用程序连接到服务集成总线并从异常目标接收消息的情况。

JMS_IBM_EXCEPTIONTIMESTAMP

数据类型：
System.Int64

相关属性：
消息

将消息发送到异常目标的时间。

此时间表示为自 1970 年 1 月 1 日格林威治标准时间 00:00:00 起的毫秒数。

此属性仅适用于应用程序连接到服务集成总线并从异常目标接收消息的情况。

JMS_IBM_FEEDBACK

数据类型：
System.Int32

相关属性：
消息

用于指示报告消息性质的代码。

此属性的有效值是可在消息描述符的 **Feedback** 字段中指定的反馈代码和原因码。

缺省情况下，不设置该属性。

JMS_IBM_FORMAT

数据类型：
字符串

相关属性：
消息

消息中应用程序数据的性质。

此属性的有效值与可在消息描述符的 **Format** 字段中指定的值相同。

缺省情况下，不设置该属性。

此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_LAST_MSG_IN_GROUP

数据类型：
System.Boolean

相关属性：
消息

指示消息是否是消息组中的最后一条消息。

如果消息是消息组中的最后一条消息，请将此属性设置为 true。 否则，请将此属性设置为 false，或者请勿设置此属性。 缺省情况下，不设置该属性。

值 true 与可在消息描述符的 **MsgFlags** 字段中指定的状态标志 MQMF_LAST_MSG_IN_GROUP 相对应。

在发布/预订域中将忽略此属性，并且此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_MSGTYPE

数据类型：
System.Int32

相关属性：
消息

消息的类型。

该属性的有效值如下所示：

有效值	含义
MQMT_DATAGRAM	消息是不需要应答的消息。
MQMT_REQUEST	消息是需要应答的消息。
MQMT_REPLY	消息是应答消息。
MQMT_REPORT	消息是报告消息。

这些值与可在消息描述符的 **MsgType** 字段中指定的消息类型相对应。

缺省情况下，不设置该属性。

此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_PUTAPPLTYPE

数据类型：
System.Int32

相关属性：
消息

发送消息的应用程序类型。

此属性的有效值是可在消息描述符的 **PutApplType** 字段中指定的应用程序类型。

缺省情况下，不设置该属性。

此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_PUTDATE

数据类型：
字符串

相关属性：
消息

发送消息的日期。

此属性的有效值与可在消息描述符的 **PutDate** 字段中指定的值相同。

缺省情况下，不设置该属性。

此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_PUTTIME

数据类型：
字符串

相关属性：
消息

发送消息的时间。

此属性的有效值与可在消息描述符的 **PutTime** 字段中指定的值相同。

缺省情况下，不设置该属性。

此属性不适用于应用程序连接到服务集成总线的情况。

JMS_IBM_REPORT_COA

数据类型：
System.Int32

相关属性：
消息

请求“到达时确认”报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。

该属性的有效值如下所示：

有效值	含义
MQRO_COA	请求“到达时确认”报告消息，报告消息中不包含原始消息中的应用程序数据。
MQRO_COA_WITH_DATA	请求“到达时确认”报告消息，报告消息中包含原始消息中的前 100 个字节的应用程序数据。
MQRO_COA_WITH_FULL_DATA	请求“到达时确认”报告消息，报告消息中包含原始消息中的所有应用程序数据。

这些值与可在消息描述符的 **Report** 字段中指定的报告选项相对应。 有关这些选项的更多信息，请参阅报告 (MQLONG)。

缺省情况下，不设置该属性。

JMS_IBM_REPORT_COD

数据类型：
System.Int32

相关属性：
消息

请求“传递时确认”报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。

该属性的有效值如下所示：

有效值	含义
MQRO_COD	请求“传递时确认”报告消息，报告消息中不包含原始消息中的应用程序数据。
MQRO_COD_WITH_DATA	请求“传递时确认”报告消息，报告消息中包含原始消息中的前 100 个字节的应用程序数据。
MQRO_COD_WITH_FULL_DATA	请求“传递时确认”报告消息，报告消息中包含原始消息中的所有应用程序数据。

这些值与可在消息描述符的 **Report** 字段中指定的报告选项相对应。

缺省情况下，不设置该属性。

JMS_IBM_REPORT_DISCARD_MSG

数据类型：
System.Int32

相关属性：
消息

请求在无法将消息传递到其预期目标时丢弃此消息。

将此属性设置为 MQRO_DISCARD_MSG，以请求在无法将消息传递到其预期目标时丢弃此消息。 如果您需要转而将消息放入死信队列中或发送到异常目标，请勿设置此属性。 缺省情况下，不设置该属性。

值 MQRO_DISCARD_MSG 与可在消息描述符的 **Report** 字段中指定的报告选项相对应。

JMS_IBM_REPORT_EXCEPTION

数据类型：
System.Int32

相关属性：
消息

请求异常报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。

该属性的有效值如下所示：

有效值	含义
MQRO_EXCEPTION	请求异常报告消息，报告消息中不包含原始消息中的应用程序数据。
MQRO_EXCEPTION_WITH_DATA	请求异常报告消息，报告消息中包含原始消息中的前 100 个字节的应用程序数据。
MQRO_EXCEPTION_WITH_FULL_DATA	请求异常报告消息，报告消息中包含原始消息中的所有应用程序数据。

这些值与可在消息描述符的 **Report** 字段中指定的报告选项相对应。

缺省情况下，不设置该属性。

JMS_IBM_REPORT_EXPIRATION

数据类型：
System.Int32

相关属性：
消息

请求到期报告消息，并指定报告消息中必须包含的原始消息的应用程序数据量。

该属性的有效值如下所示：

有效值	含义
MQRO_EXPIRATION	请求到期报告消息，报告消息中不包含原始消息中的应用程序数据。
MQRO_EXPIRATION_WITH_DATA	请求到期报告消息，报告消息中包含原始消息中的前 100 个字节的应用程序数据。
MQRO_EXPIRATION_WITH_FULL_DATA	请求到期报告消息，报告消息中包含原始消息中的所有应用程序数据。

这些值与可在消息描述符的 **Report** 字段中指定的报告选项相对应。

缺省情况下，不设置该属性。

JMS_IBM_REPORT_NAN

数据类型：
System.Int32

相关属性：
消息

请求负面操作通知报告消息。

将此属性设置为 MQRO_NAN，以请求负面操作通知报告消息。 如果您不需要负面操作通知报告消息，请勿设置此属性。 缺省情况下，不设置该属性。

值 MQRO_NAN 与可在消息描述符的 **Report** 字段中指定的报告选项相对应。

JMS_IBM_REPORT_PAN

数据类型：
System.Int32

相关属性：
消息

请求正面操作通知报告消息。

将此属性设置为 MQRO_PAN，以请求正面操作通知报告消息。 如果您不需要正面操作通知报告消息，请勿设置此属性。 缺省情况下，不设置该属性。

值 MQRO_PAN 与可在消息描述符的 **Report** 字段中指定的报告选项相对应。

JMS_IBM_REPORT_PASS_CORREL_ID

数据类型：
System.Int32

相关属性：
消息

请求任何报告或应答消息的相关标识与原始消息的相关标识相同。

该属性的有效值如下所示：

有效值	含义
MQRO_PASS_CORREL_ID	请求任何报告或应答消息的相关标识与原始消息的相关标识相同。
MQRO_COPY_MSG_ID_TO_CORREL_ID	请求任何报告或应答消息的相关标识与原始消息的消息标识相同。

这些值对应于可在消息描述符的 **Report** 字段中指定的报告选项。

此属性的缺省值为 MQRO_COPY_MSG_ID_TO_CORREL_ID。

JMS_IBM_REPORT_PASS_MSG_ID

数据类型：
System.Int32

相关属性：
消息

请求任何报告或应答消息的消息标识与原始消息的消息标识相同。

该属性的有效值如下所示：

有效值	含义
MQRO_PASS_MSG_ID	请求任何报告或应答消息的消息标识与原始消息的消息标识相同。
MQRO_NEW_MSG_ID	请求为每个报告或应答消息生成新的消息标识。

这些值与可在消息描述符的 Report 字段中指定的报告选项相对应。

此属性的缺省值为 MQRO_NEW_MSG_ID。

JMS_IBM_RETAIN

数据类型：
System.Int32

相关属性：
消息

设置此属性可指示队列管理器将消息视作“保留发布”。 订户收到来自主题的消息时，除了前发行版中收到的消息外，还可能在预订后立即收到其他消息。 这些消息是已预订主题的可选保留发布。 对于符合预订要求的每个主题，如果存在保留发布，那么该发布可供传递到预订消息使用者。

RETAIN_PUBLICATION 是此属性唯一的有效值。 缺省情况下，不设置此属性。

注: 仅在发布/预订域中，此属性才适用。

JMS_IBM_SYSTEM_MESSAGEID

数据类型:
字符串

相关属性:
消息

用于在服务集成总线内唯一识别消息的标识。 该属性为只读。

此属性仅适用于应用程序连接到服务集成总线的情况。

JMSX_APPID

数据类型:
字符串

相关属性:
消息

发送消息的应用程序名称。

此属性是由 JMS 定义的具有 JMS 名称 JMSXAppID 的属性。 有关此属性的更多信息，请参阅 *Java* 消息服务规范 *V 1.1*。

缺省情况下，不设置该属性。

对于与代理程序的实时连接，该属性无效。

JMSX_DELIVERY_COUNT

数据类型:
System.Int32

相关属性:
消息

尝试传递消息的次数。

此属性是由 JMS 定义的具有 JMS 名称 JMSXDeliveryCount 的属性。 有关此属性的更多信息，请参阅 *Java* 消息服务规范 *V 1.1*。

缺省情况下，不设置该属性。

对于与代理程序的实时连接，该属性无效。

JMSX_GROUPID

数据类型:
字符串

相关属性:
消息

消息所属消息组的标识。

此属性是由 JMS 定义的具有 JMS 名称 JMSXGroupID 的属性。 有关此属性的更多信息，请参阅 *Java* 消息服务规范 *V 1.1*。

缺省情况下，不设置该属性。

对于与代理程序的实时连接，该属性无效。

JMSX_GROUPSEQ

数据类型:
System.Int32

相关属性:
消息

消息在消息组中的序号。

此属性是由 JMS 定义的具有 JMS 名称 JMSXGroupSeq 的属性。 有关此属性的更多信息，请参阅 *Java* 消息服务规范 *V 1.1*。

缺省情况下，不设置该属性。

对于与代理程序的实时连接，该属性无效。

JMSX_USERID

数据类型:
字符串

相关属性:
消息

与发送消息的应用程序相关联的用户标识。

此属性是由 JMS 定义的具有 JMS 名称 JMSXUserID 的属性。 有关此属性的更多信息，请参阅 *Java* 消息服务规范 *V 1.1*。

缺省情况下，不设置该属性。

对于与代理程序的实时连接，该属性无效。

XMSC_ASYNC_EXCEPTIONS

数据类型:
System.Int32

相关属性:
ConnectionFactory

适用对象:
JMS 管理工具长名称 :ASYNCEXCEPTION

JMS 管理工具短名称 :AEX

此属性确定仅在连接中断时还是在 XMS API 调用出现异步异常时，XMS 通知 ExceptionListener。 此属性适用于通过该 ConnectionFactory 创建的且已注册 ExceptionListener 的所有连接。

此属性的有效值为:

XMSC_ASYNC_EXCEPTIONS_ALL
在同步 API 调用作用域外部异步检测到的任何异常以及所有连接中断异常都会发送到 ExceptionListener。

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN
只有指示连接中断的异常才会发送到 ExceptionListener。 在异步处理期间发生的任何其他异常都不会报告给 ExceptionListener，因此，不会向应用程序通知这些异常。

缺省情况下，此属性设置为 XMSC_ASYNC_EXCEPTIONS_ALL。

XMSC_CLIENT_ID

数据类型:
字符串

相关属性:
ConnectionFactory

适用对象:

JMS 管理工具长名称 :CLIENTID

JMS 管理工具短名称 :CID

连接的客户机标识。

客户机标识仅用于在发布/预订域中支持持久预订，而在点到点域中会忽略此标识。 有关设置客户机标识的更多信息，请参阅第 18 页的『ConnectionFactories 和 Connection 对象』。

此属性不适用于与代理程序的实时连接。

XMSC_CONNECTION_TYPE

数据类型:

System.Int32

相关属性:

ConnectionFactory

与应用程序相连的消息传递服务器的类型。

该属性的有效值如下所示:

有效值	含义
XMSC_CT_RTT	与代理程序的实时连接。
XMSC_CT_WMQ	与 WebSphere MQ 队列管理器的连接。
XMSC_CT_WPM	与 WebSphere 服务集成总线的连接。

缺省情况下，不设置该属性。

XMSC_DELIVERY_MODE

数据类型:

System.Int32

相关属性:

Destination

URI 中使用的名称:

`persistence`（适用于 WebSphere MQ 目标）
`deliveryMode`（适用于 WebSphere 缺省消息传递提供程序目标）

适用对象:

JMS 管理工具长名称 :PERSISTENCE

JMS 管理工具短名称 :PER

发送到目标的消息的传递方式。

该属性的有效值如下所示:

有效值	含义
XMSC_DELIVERY_NOT_PERSISTENT	发送到目标的消息是非持久消息。 将忽略消息生产者的缺省传递方式或 Send 调用上指定的任何传递方式。 如果目标为 WebSphere MQ 队列，那么还会忽略队列属性 *DefPersistence* 的值。
XMSC_DELIVERY_PERSISTENT	发送到目标的消息是持久消息。 将忽略消息生产者的缺省传递方式或 Send 调用上指定的任何传递方式。 如果目标为 WebSphere MQ 队列，那么还会忽略队列属性 *DefPersistence* 的值。

有效值	含义
XMSC_DELIVERY_AS_APP	发送到目标的消息采用 Send 调用上指定的传递方式。 如果 Send 调用未指定传递方式，那么将改为使用消息生产者的缺省传递方式。 如果目标为 WebSphere MQ 队列，那么将忽略队列属性 *DefPersistence* 的值。
XMSC_DELIVERY_AS_DEST	如果目标为 WebSphere MQ 队列，那么放入队列的消息将采用队列属性 *DefPersistence* 值指定的传递方式。 将忽略消息生产者的缺省传递方式或 Send 调用上指定的任何传递方式。 如果目标不是 WebSphere MQ 队列，那么其含义与 XMSC_DELIVERY_AS_APP 的含义相同。

缺省值为 XMSC_DELIVERY_AS_APP。

XMSC_IC_PROVIDER_URL

数据类型：
字符串

相关属性：
InitialContext

用于查找 JNDI 命名目录，因此 COS 命名服务不需要与 Web Service 在同一服务器上。

XMSC_IC_SECURITY_AUTHENTICATION

数据类型：
字符串

相关属性：
InitialContext

基于 Java 上下文接口 SECURITY_AUTHENTICATION。 此属性仅适用于 COS 命名上下文。

XMSC_IC_SECURITY_CREDENTIALS

数据类型：
字符串

相关属性：
InitialContext

基于 Java 上下文接口 SECURITY_CREDENTIALS。 此属性仅适用于 COS 命名上下文。

XMSC_IC_SECURITY_PRINCIPAL

数据类型：
字符串

相关属性：
InitialContext

基于 Java 上下文接口 SECURITY_PRINCIPAL。 此属性仅适用于 COS 命名上下文。

XMSC_IC_SECURITY_PROTOCOL

数据类型：
字符串

相关属性：
InitialContext

基于 Java 上下文接口 SECURITY_PROTOCOL 此属性仅适用于 COS 命名上下文。

XMSC_IC_URL

数据类型：
字符串

相关属性：
InitialContext

对于 LDAP 和 FileSystem 上下文，这是包含受管对象的存储库的地址。

对于 COS 命名上下文，这是在目录中查找对象的 Web Service 的地址。

XMSC_IS_SUBSCRIPTION_MULTICAST

数据类型：
System.Boolean

相关属性：
MessageConsumer

指示是否正在使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 该属性为只读。

如果正在使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者，那么此属性的值为 true。 否则，值为 false。

此属性仅适用于与代理程序的实时连接。

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

数据类型：
System.Boolean

相关属性：
MessageConsumer

指示是否使用具有可靠服务质量的 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 该属性为只读。

如果在提供可靠服务质量的情况下使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者，那么此属性的值为 true。 否则，值为 false。

此属性仅适用于与代理程序的实时连接。

XMSC_JMS_MAJOR_VERSION

数据类型：
System.Int32

相关属性：
ConnectionMetaData

XMS 所基于的 JMS 规范的主要版本号。 该属性为只读。

XMSC_JMS_MINOR_VERSION

数据类型：
System.Int32

相关属性：
ConnectionMetaData

XMS 所基于的 JMS 规范的次要版本号。 该属性为只读。

XMSC_JMS_VERSION

数据类型：
字符串

相关属性：
ConnectionMetaData

XMS 所基于的 JMS 规范的版本标识。 该属性为只读。

XMSC_MAJOR_VERSION

数据类型：
System.Int32

相关属性：
ConnectionMetaData

XMS 客户机的版本号。 该属性为只读。

XMSC_MINOR_VERSION

数据类型：
System.Int32

相关属性：
ConnectionMetaData

XMS 客户机的发行版号。 该属性为只读。

XMSC_PASSWORD

数据类型：
字节数组

相关属性：
ConnectionFactory

可用于在应用程序尝试连接到消息传递服务器时对其进行认证的密码。 此密码与 XMSC_USERID 属性结合使用。

缺省情况下，不设置该属性。

Multi 如果要连接到 多平台上的 WebSphere MQ，并且设置了连接工厂的 XMSC_USERID 属性，那么该属性必须与已登录用户的 **userid** 相匹配。 如果未设置这些属性，那么缺省情况下队列管理器会使用已登录用户的 **userid**。 如果需要个别用户通过进一步的连接级别认证，可以编写在 WebSphere MQ 中配置的客户机认证出口。 您可以在 WebSphere MQ 客户机手册中的“认证”主题中了解有关创建客户机认证出口的更多信息。

要在连接到 z/OS 上的 WebSphere MQ 时认证用户，需要使用安全出口。

XMSC_PRIORITY

数据类型：
System.Int32

相关属性：
Destination

URI 中使用的名称：
priority

发送到目标的消息的优先级。

该属性的有效值如下所示：

有效值	含义
范围 0（表示最低优先级）到 9（表示最高优先级）内的整数	发送到目标的消息具有指定的优先级。 将忽略消息生产者的缺省优先级或 Send 调用上指定的任何优先级。 如果目标为 WebSphere MQ 队列，那么还会忽略队列属性 **DefPriority** 的值。

有效值	含义
XMSC_PRIORITY_AS_APP	发送到目标的消息具有 Send 调用上指定的优先级。 如果 Send 调用未指定优先级，那么将改为使用消息生产者的缺省优先级。 如果目标为 WebSphere MQ 队列，那么将忽略队列属性 **DefPriority** 的值。
XMSC_PRIORITY_AS_DEST	如果目标为 WebSphere MQ 队列，那么放入队列的消息具有队列属性 **DefPriority** 值指定的优先级。 将忽略消息生产者的缺省优先级或 Send 调用上指定的任何优先级。 如果目标不是 WebSphere MQ 队列，那么其含义与 XMSC_PRIORITY_AS_APP 的含义相同。

缺省值为 XMSC_PRIORITY_AS_APP。

根据消息优先级，WebSphere MQ Real-Time Transport 和 WebSphere MQ Multicast Transport 将不采取任何操作。

XMSC_PROVIDER_NAME

数据类型：
字符串

相关属性：
ConnectionMetaData

XMS 客户机的提供者。 该属性为只读。

XMSC_RTT_BROKER_PING_INTERVAL

数据类型：
System.Int32

相关属性：
ConnectionFactory

XMS .NET 检查与实时消息传递服务器的连接以检测任何活动前的时间间隔（以毫秒计）。 如果未检测到任何活动，那么客户机将启动 ping；如果未检测到对 ping 的响应，那么将关闭连接。

此属性的缺省值为 30000。

XMSC_RTT_CONNECTION_PROTOCOL

数据类型：
System.Int32

相关属性：
ConnectionFactory

与代理程序的实时连接所使用的通信协议。

该属性的值必须为 XMSC_RTT_CP_TCP，表示通过 TCP/IP 与代理程序建立实时连接。 缺省值为 XMSC_RTT_CP_TCP。

XMSC_RTT_HOST_NAME

数据类型：
字符串

相关属性：
ConnectionFactory

运行代理程序的系统的主机名或 IP 地址。

此属性与 XMSC_RTT_PORT 属性一起用于标识代理程序。

缺省情况下，不设置该属性。

XMSC_RTT_LOCAL_ADDRESS

数据类型：
字符串

相关属性：
ConnectionFactory

与代理程序的实时连接所使用的本地网络接口的主机名或 IP 地址。

仅当运行应用程序的系统有两个或更多个网络接口并且您需要能够指定实时连接必须使用的接口时，此属性才有用。 如果系统只有一个网络接口，那么只能使用此接口。 如果系统有两个或更多个网络接口，并且未设置此属性，那么将随机选择接口。

缺省情况下，不设置该属性。

XMSC_RTT_MULTICAST

数据类型：
System.Int32

相关属性：
ConnectionFactory 和 Destination

URI 中使用的名称：
mulicast

连接工厂或目标的多点广播设置。 只有作为主题的目标才具有该属性。

应用程序使用此属性来启用与代理程序实时连接相关联的多点广播；如果已启用多点广播，还会指定使用多点广播将消息从代理程序传递到消息使用者的确切方式。 此属性不会影响消息生产者向代理程序发送消息的方式。

该属性的有效值如下所示：

有效值	含义
XMSC_RTT_MULTICAST_DISABLED	不使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 该值是 ConnectionFactory 对象的缺省值。
XMSC_RTT_MULTICAST_ASCF	根据与消息使用者相关联的连接工厂的多点广播设置，将消息传递到消息使用者。 创建连接时，已记录连接工厂的多点广播设置。 该值仅适用于 Destination 对象，并且是 Destination 对象的缺省值。
XMSC_RTT_MULTICAST_ENABLED	如果在代理程序中对主题配置了多点广播，那么将使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 如果对主题配置了可靠的多点广播，那么将使用可靠的服务质量。
XMSC_RTT_MULTICAST_RELIABLE	如果在代理程序中对主题配置了可靠的多点广播，那么会在提供可靠服务质量的情况下使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 如果没有对主题配置可靠的多点广播，那么无法为该主题创建消息使用者。
XMSC_RTT_MULTICAST_NOT_RELIABLE	如果在代理程序中对主题配置了多点广播，那么将使用 WebSphere MQ Multicast Transport 将消息传递到消息使用者。 即使对主题配置了可靠的多点广播，也不会使用可靠服务质量。

XMSC_RTT_PORT

数据类型：
System.Int32

相关属性：
ConnectionFactory

代理程序侦听入局请求所使用的端口号。 在代理程序上，必须配置 Real-timeInput 或 Real-timeOptimizedFlow 消息处理节点以侦听此端口。

此属性与 XMSC_RTT_HOST_NAME 属性一起用于标识代理程序。

此属性的缺省值为 XMSC_RTT_DEFAULT_PORT 或 1506。

XMSC_TIME_TO_LIVE

数据类型：
System.Int32

相关属性：
Destination

URI 中使用的名称：

expiry（适用于 WebSphere MQ 目标）
timeToLive（适用于 WebSphere 缺省消息传递提供程序目标）

发送到目标的消息的生存时间。

该属性的有效值如下所示：

有效值	含义
0	发送到目标的消息永不过期。
正整数	发送到目标的消息具有指定的生存时间（以毫秒计）。 将忽略消息生产者的缺省生存时间或 Send 调用上指定的任何生存时间。
XMSC_TIME_TO_LIVE_AS_APP	发送到目标的消息具有 Send 调用上指定的生存时间。 如果 Send 调用未指定生存时间，那么将改为使用消息生产者的缺省生存时间。

缺省值为 XMSC_TIME_TO_LIVE_AS_APP。

XMSC_USERID

数据类型：
字符串

相关属性：
ConnectionFactory

可用于在应用程序尝试连接到消息传递服务器时对其进行认证的用户标识。 此用户标识与 XMSC_PASSWORD 属性结合使用。

缺省情况下，不设置该属性。

Multi 如果要连接到 IBM MQ for Multiplatforms，并设置连接工厂的 XMSC_USERID 属性，那么该属性必须与已登录用户的 **userid** 相匹配。 如果未设置这些属性，那么缺省情况下队列管理器会使用已登录用户的 **userid**。 如果需要对个别用户进行进一步的连接级别认证，那么可以编写在 IBM MQ 中配置的客户机认证出口。

z/OS 要在连接到 IBM MQ for z/OS 时认证用户，需要使用安全出口。

XMSC_VERSION

数据类型：
字符串

相关属性：
ConnectionMetaData

XMS 客户机的版本标识。 该属性为只读。

XMSC_WMQ_BROKER_CONTROLQ

数据类型：
字符串

相关属性：
ConnectionFactory

代理程序使用的控制队列的名称。

此属性的缺省值为 SYSTEM.BROKER.CONTROL.QUEUE。

仅在发布/预订域中，该属性才适用。

XMSC_WMQ_BROKER_PUBQ

数据类型：
字符串

相关属性：
ConnectionFactory

受代理程序监控且应用程序将其发布的消息发送到的队列的名称。

此属性的缺省值为 SYSTEM.BROKER.DEFAULT.STREAM。

仅在发布/预订域中，该属性才适用。

XMSC_WMQ_BROKER_QMGR

数据类型：
字符串

相关属性：
ConnectionFactory

代理程序连接到的队列管理器的名称。

缺省情况下，不设置该属性。

仅在发布/预订域中，该属性才适用。

XMSC_WMQ_BROKER_SUBQ

数据类型：
字符串

相关属性：
ConnectionFactory

非持久消息使用者的订户队列的名称。

订户队列的名称必须以下列字符开头：

SYSTEM.JMS.ND.

如果您希望所有非持久消息使用者共享同一个订户队列，请指定共享队列的完整名称。 在应用程序创建非持久消息使用者之前，必须存在具有指定名称的队列。

如果您希望每个非持久消息使用者从其自己的独占订户队列中检索消息，请指定以星号 (*) 结尾的队列名称。 然后，当应用程序创建非持久消息使用者时， XMS 客户机将创建动态队列以供消息使用者独占使用。 XMS 客户机使用此属性的值来设置用于创建动态队列的对象描述符中的 **DynamicQName** 字段内容。

此属性的缺省值为 SYSTEM.JMS.ND.SUBSCRIBER.QUEUE，这表示缺省情况下 XMS 使用共享队列方法。

仅在发布/预订域中，该属性才适用。

XMSC_WMQ_BROKER_VERSION

数据类型：
System.Int32

相关属性：
ConnectionFactory 和 Destination

URI 中使用的名称：
brokerVersion

应用程序针对连接或目标使用的代理程序类型。 只有作为主题的目标才具有该属性。

该属性的有效值如下所示：

有效值	含义
XMSC_WMQ_BROKER_V1	应用程序正在使用 WebSphere MQ 发布/预订代理程序。 如果从 WebSphere MQ 发布/预订迁移到 WebSphere Message Broker，但未更改应用程序，那么应用程序也可以使用该值。
XMSC_WMQ_BROKER_V2	应用程序正在使用 IBM Integration Bus 代理程序。
XMSC_WMQ_BROKER_UNSPECIFIED	迁移代理程序后，请设置此属性以便不再使用 RFH2 头。 迁移后，此属性不再相关。

连接工厂的缺省值为 XMSC_WMQ_BROKER_UNSPECIFIED，但在缺省情况下，未针对目标设置此属性。 如果针对目标设置此属性，它将覆盖连接工厂属性指定的任何值。

XMSC_WMQ_CCDTURL

数据类型：
System.String

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :CCDTURL

JMS 管理工具短名称 :CCDT

统一资源定位符 (URL)，用于标识包含客户机通道定义表的文件的名称和位置并指定该文件的访问方式。

缺省情况下，不设置该属性。

XMSC_WMQ_CCSID

数据类型：
System.Int32

相关属性：
Destination

URI 中使用的名称：
CCSID

当 XMS 客户机将消息转发到目标时，消息主体中的字符数据字符串所在的编码字符集或代码页的标识 (CCSID)。 如果针对单条消息进行设置，那么 JMS_IM_CHARACTER_SET 属性将覆盖此属性针对目标指定的 CCSID。

此属性的缺省值为 1208。

此属性仅适用于发送到目标的消息，而不适用于从目标接收的消息。

XMSC_WMQ_CHANNEL

数据类型：
字符串

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :CHANNEL

JMS 管理工具短名称: 陈

连接要使用的通道的名称。

缺省情况下，不设置该属性。

此属性仅适用于应用程序在客户机方式下连接到队列管理器的情况。

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS

数据类型：
字符串

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :CLIENTRECONNECTOPTIONS

JMS 管理工具短名称 :CROPT

此属性可为该工厂所创建的新连接指定客户机重新连接选项。 它存在于 XMSC 中，是以下之一:

- WMQ_CLIENT_RECONNECT_AS_DEF（缺省值）。 使用 `mqclient.ini` 文件中指定的值。 可使用 Channels 节中的 **DefRecon** 属性来设置该值。 它可设置为以下值之一:
  1. 是。 其行为方式类似 WMQ_CLIENT_RECONNECT 选项
  2. 否。 缺省值。 请勿指定任何重新连接选项
  3. QMGR。 其行为方式类似 WMQ_CLIENT_RECONNECT_Q_MGR 选项
  4. DISABLED。 其行为方式类似 WMQ_CLIENT_RECONNECT_DISABLED 选项
- WMQ_CLIENT_RECONNECT。 重新连接到连接名称列表中指定的任何队列管理器。
- WMQ_CLIENT_RECONNECT_Q_MGR。 重新连接到其最初连接到的同一队列管理器。 如果其尝试连接到的队列管理器（在连接名称列表中指定）的 QMID 不同于最初连接到的队列管理器，那么它将返回 `MQRC_RECONNECT_QMID_MISMATCH`。
- WMQ_CLIENT_RECONNECT_DISABLED。 禁用重新连接。

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT

数据类型：
字符串

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :CLIENTRECONNECTTIMEOUT

JMS 管理工具短名称 :CRT

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT 属性仅对受管 XMS .NET 客户机有效。

此属性可指定客户机连接尝试重新连接的持续时间（以秒计）。

在此持续时间内尝试重新连接后，客户机将失败，并返回 MQRC_RECONNECT_FAILED。 此属性的缺省设置为 XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT。

此属性的缺省值为 1800。

XMSC_WMQ_CONNECTION_MODE

数据类型：
System.Int32

相关属性：
ConnectionFactory

应用程序连接到队列管理器所使用的方式。

该属性的有效值如下所示：

有效值	含义
XMSC_WMQ_CM_BINDINGS	在绑定方式下连接到队列管理器以获得最佳性能。 此值为 C/C++ 的缺省值。
XMSC_WMQ_CM_CLIENT	在客户机方式下连接到队列管理器以确保堆栈完全受管。 此值为 .NET 的缺省值。
XMSC_WMQ_CM_CLIENT_UNMANAGED（仅限 .NET）	连接到强制使用非受管客户机堆栈的队列管理器。

相关概念
.NET 中的受管和非受管操作
受管代码专门在 .NET 通用语言运行时环境中执行，并且完全依赖于该运行时提供的服务。 如果应用程序的任何部分运行或调用 .NET 通用语言运行时环境外部的服务，那么此应用程序归类为非受管应用程序。

XMSC_WMQ_CONNECTION_NAME_LIST

数据类型：
字符串

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :CONNECTIONNAMELIST

JMS 管理工具短名称 :CNLIST

此属性可指定客户机在中断连接后尝试重新连接的主机。

连接名称列表是主机/IP 端口对的逗号分隔列表。 此属性的缺省设置为 WMQ_CONNECTION_NAME_LIST_DEFAULT。

例如，`127.0.0.1(1414),host2.example.com(1400)`

此属性的缺省设置为 `localhost(1414)`。

XMSC_WMQ_DUR_SUBQ

数据类型：
字符串

相关属性：
Destination

正在从目标接收消息的持久订户的订户队列名称。 只有作为主题的目标才具有该属性。

订户队列的名称必须以下列字符开头：

SYSTEM.JMS.D.

如果您希望所有持久订户共享同一个订户队列，请指定共享队列的完整名称。 在应用程序创建持久订户之前，必须存在具有指定名称的队列。

如果您希望每个持久订户从其自己的互斥订户队列中检索消息，请指定以星号 (*) 结尾的队列名称。 然后，当应用程序创建持久订户时， XMS 客户机将创建动态队列以供持久订户独占使用。 XMS 客户机使用此属性的值来设置用于创建动态队列的对象描述符中的 **DynamicQName** 字段内容。

此属性的缺省值为 SYSTEM.JMS.D.SUBSCRIBER.QUEUE，这表示缺省情况下 XMS 使用共享队列方法。

仅在发布/预订域中，该属性才适用。

XMSC_WMQ_ENCODING

数据类型：
System.Int32

相关属性：
Destination

当 XMS 客户机将消息转发到目标时，如何表示消息体中的数字数据。 如果针对单条消息进行设置，那么 JMS_IBM_ENCODING 属性将覆盖此属性针对目标指定的编码。 此属性指定二进制整数、压缩十进制整数和浮点数的表示法。

该属性的有效值与可在消息描述符的 **Encoding** 字段中指定的值相同。

应用程序可使用以下命名常量来设置此属性：

命名常量	含义
MQENC_INTEGER_NORMAL	标准整数编码
MQENC_INTEGER_REVERSED	反向整数编码
MQENC_DECIMAL_NORMAL	标准压缩十进制编码
MQENC_DECIMAL_REVERSED	反向压缩十进制编码
MQENC_FLOAT_IEEE_NORMAL	标准 IEEE 浮点编码
MQENC_FLOAT_IEEE_REVERSED	反向 IEEE 浮点编码
MQENC_FLOAT_S390	z/OS 体系结构浮点编码
MQENC_NATIVE	本机编码

要构成该属性的值，应用程序可添加其中的三个常量，如下所示：

- 名称以 MQENC_INTEGER 开头的常量，用于指定二进制整数表示法
- 名称以 MQENC_DECIMAL 开头的常量，用于指定压缩十进制整数表示法
- 名称以 MQENC_FLOAT 开头的常量，用于指定浮点数表示法

或者，应用程序可将此属性设置为 MQENC_NATIVE（其值与环境相关）。

此属性的缺省值为 MQENC_NATIVE。

此属性仅适用于发送到目标的消息，而不适用于从目标接收的消息。

XMSC_WMQ_FAIL_IF_QUIESCE

数据类型：
System.Int32

相关属性：
ConnectionFactory 和 Destination

URI 中使用的名称：
failIfQuiesce

适用对象：
JMS 管理工具长名称: FAILIFQUIESCE

JMS 管理工具短名称 :FIQ

当应用程序连接到的队列管理器处于停顿状态时，某些方法调用是否会失败。

该属性的有效值如下所示：

有效值	含义
XMSC_WMQ_FIQ_YES	队列管理器处于停顿状态时，某些方法调用将失败。 当应用程序检测到队列管理器处于停顿状态时，应用程序可以完成其即时任务并关闭连接，以允许队列管理器停止。
XMSC_WMQ_FIQ_NO	由于队列管理器处于停顿状态，所有方法调用均未失败。 如果指定了此值，那么应用程序将检测不到队列管理器是否处于停顿状态。 应用程序可能会继续对队列管理器执行操作，因而会阻止队列管理器停止运行。

连接工厂的缺省值为 XMSC_WMQ_FIQ_YES，但在缺省情况下，未针对目标设置此属性。 如果针对目标设置此属性，它将覆盖连接工厂属性指定的任何值。

XMSC_WMQ_MESSAGE_BODY

数据类型：
System.Int32

相关属性：
Destination

此属性确定 XMS 应用程序是否将 IBM WebSphere MQ 消息的 MQRFH2 作为消息有效内容的一部分 (即，作为消息体的一部分) 进行处理。

注： 在将消息发送到目标时，XMSC_WMQ_MESSAGE_BODY 属性将取代现有的 XMS 目标属性 XMSC_WMQ_TARGET_CLIENT。

此属性的有效值为：

XMSC_WMQ_MESSAGE_BODY_JMS
接收： 入站 XMS 消息类型和主体由收到的 IBM WebSphere MQ 消息中的 MQRFH2（如果存在）或 MQMD（如果不存在 MQRFH2）的内容来确定。

发送： 出站 XMS 消息体包含基于 XMS 消息属性和头字段预先准备和自动生成的 MQRFH2 头。

XMSC_WMQ_MESSAGE_BODY_MQ
接收： 入站 XMS 消息类型始终为 ByteMessage，与收到的 IBM WebSphere MQ 消息的内容和收到的 MQMD 的 format 字段无关。 XMS 消息体是底层消息传递提供程序 API 调用返回的未作更改的消息数据。 消息体中数据的字符集和编码由 MQMD 的 CodedCharSetId 和 Encoding 字段确定。 消息体中数据的格式由 MQMD 的 Format 字段确定。

发送： 出站 XMS 消息体按原样包含应用程序有效内容；主体中不添加任何自动生成的 IBM WebSphere MQ 头。

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED
接收： XMS 客户机将为此属性确定合适的值。 对于接收路径，该值为 WMQ_MESSAGE_BODY_JMS 属性值。

发送： XMS 客户机将为此属性确定合适的值。 对于发送路径，该值为 XMSC_WMQ_TARGET_CLIENT 属性值。

缺省情况下，此属性设置为 XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED。

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

数据类型：
System.Int32

相关属性：
Destination

确定 XMS 应用程序要设置的消息上下文级别。 应用程序必须以相应的上下文权限运行才能使属性生效。

此属性的有效值为：

XMSC_WMQ_MDCTX_DEFAULT
对于出站消息，MQOPEN API 调用和 MQPMO 结构未指定任何显式消息上下文选项。

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT
MQOPEN API 调用指定消息上下文选项 MQOO_SET_IDENTITY_CONTEXT，而 MQPMO 结构指定 MQPMO_SET_IDENTITY_CONTEXT。

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT
MQOPEN API 调用指定消息上下文选项 MQOO_SET_ALL_CONTEXT，而 MQPMO 结构指定 MQPMO_SET_ALL_CONTEXT。

缺省情况下，此属性设置为 XMSC_WMQ_MDCTX_DEFAULT。

注: 此属性不适用于应用程序连接到系统集成总线的情况。

发送消息时，为达到您想要的效果，以下属性要求 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 属性设置为 XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT 属性值或 XMSC_WMQ_MDCTX_SET_ALL_CONTEXT 属性值：

- JMS_IBM_MQMD_USERIDENTIFIER
- JMS_IBM_MQMD_ACCOUNTINGTOKEN
- JMS_IBM_MQMD_APPLIDENTITYDATA

发送消息时，为达到您想要的效果，以下属性要求 XMSC_WMQ_MQMD_MESSAGE_CONTEXT 属性设置为 XMSC_WMQ_MDCTX_SET_ALL_CONTEXT 属性值：

- 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

数据类型：
System.Int32

相关属性：
Destination

此属性确定 XMS 应用程序是否可以抽取 MQMD 字段的值。

此属性的有效值为：

XMSC_WMQ_READ_ENABLED_NO
发送消息时，将不会更新所发送消息中的 JMS_IBM_MQMD* 属性，从而不会反映 MQMD 中已更新的字段值。

接收消息时，在收到的消息上未提供 JMS_IBM_MQMD* 属性，即使发送方设置了其中的部分或全部属性也是如此。

XMSC_WMQ_READ_ENABLED_YES
发送消息时，将更新所发送消息上的所有 JMS_IBM_MQMD* 属性（包括发送方未显式设置的那些属性）以反映 MQMD 中已更新的字段值。

接收消息时，在收到的消息上提供了所有 JMS_IBM_MQMD* 属性（包括发送方未显式设置的那些属性）。

缺省情况下，此属性设置为 XMSC_WMQ_READ_ENABLED_NO。

XMSC_WMQ_MQMD_WRITE_ENABLED

数据类型：
System.Int32

相关属性：
Destination

此属性确定 XMS 应用程序是否可以写入 MQMD 字段的值。

此属性的有效值为：

XMSC_WMQ_WRITE_ENABLED_NO
将忽略所有 JMS_IBM_MQMD* 属性，并且不会将它们的值复制到底层的 MQMD 结构。

XMSC_WMQ_WRITE_ENABLED_YES
将处理 JMS_IBM_MQMD* 属性。 它们的值将复制到底层的 MQMD 结构。

缺省情况下，此属性设置为 XMSC_WMQ_WRITE_ENABLED_NO。

XMSC_WMQ_PUT_ASYNC_ALLOWED

数据类型：
System.Int32

相关属性：
Destination

此属性确定是否允许消息生产者使用异步放置来将消息发送到此目标。

此属性的有效值为：

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST
通过参考队列或主题定义来确定是否允许异步放置。

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_Q_DEF
通过参考队列定义来确定是否允许异步放置。

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_TOPIC_DEF
通过参考主题定义来确定是否允许异步放置。

XMSC_WMQ _PUT_ASYNC_ALLOWED_DISABLED
不允许异步放置。

XMSC_WMQ _PUT_ASYNC_ALLOWED_ENABLED
允许异步放置。

缺省情况下，此属性设置为 XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST。

注： 此属性不适用于应用程序连接到系统集成总线的情况。

XMSC_WMQ_READ_AHEAD_ALLOWED

数据类型：
System.Int32

相关属性：
Destination

此属性确定是否允许消息使用者和队列浏览器在接收消息之前使用预读功能从此目标获取非事务性的非持久消息并将其放入内部缓冲区。

此属性的有效值为：

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF
通过参考队列定义来确定是否允许预读。

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_ TOPIC _DEF
通过参考主题定义来确定是否允许预读。

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST
通过参考队列或主题定义来确定是否允许预读。

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED
使用或浏览消息期间不允许预读。

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED
允许预读。

缺省情况下，此属性设置为 XMSC_WMQ _READ_AHEAD_ALLOWED_AS_DEST。

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

数据类型：
System.Int32

相关属性：
Destination

对于正在传递到异步消息侦听器的消息，此属性确定当关闭消息使用者时在内部预读缓冲区中对消息执行的操作。

此属性适用于在使用来自目标的消息时指定关闭队列选项，而不适用于将消息发送到目标的情况。

队列浏览器将忽略此属性，因为在浏览期间消息在队列中仍可用。

此属性的有效值为：

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT
只有当前消息侦听器调用会在返回之前完成，内部预读缓冲区中可能会留有一些消息，随后将丢弃这些消息。

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL
在返回之前，会将内部预读缓冲区中的所有消息传递到应用程序消息侦听器。

缺省情况下，此属性设置为 XMSC_WMQ _READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT。

注：

- **异常应用程序终止**

  当 XMS 应用程序突然终止时，预读缓冲区中的所有消息都会丢失。
- **对事务的影响**

  应用程序使用事务时会禁用预读功能。 因此，在使用事务性会话时，应用程序不会觉察到行为方面的差异。
- **会话确认方式的影响**

  当确认方式为 XMSC_AUTO_ACKNOWLEDGE 或 XMSC_DUPS_OK_ACKNOWLEDGE 时，会针对非事务性会话启用预读功能。 如果会话确认方式为 XMSC_CLIENT_ACKNOWLEDGE，那么不论是事务性还是非事务性会话，都会禁用预读功能。
- **对队列浏览器和队列浏览器选择器的影响**

  用于 XMS 应用程序的队列浏览器和队列浏览器选择器都可通过预读功能来提高性能。 关闭队列浏览器不会降低性能，因为队列中的消息仍可用于其他操作。 除了预读功能带来的性能优势外，对队列浏览器和队列浏览器选择器没有其他影响。

XMSC_WMQ_HOST_NAME

数据类型：
字符串

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :HOSTNAME

JMS 管理工具短名称 :HOST

运行队列管理器的系统的主机名或 IP 地址。

此属性仅适用于应用程序在客户机方式下连接到队列管理器的情况。 此属性与 XMSC_WMQ_PORT 属性一起用于标识队列管理器。

此属性的缺省值为 `localhost`。

XMSC_WMQ_LOCAL_ADDRESS

数据类型：
字符串

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :LOCALADDRESS

JMS 管理工具短名称 :LA

对于到队列管理器的连接，该属性指定要使用的本地网络接口和/或要使用的本地端口/本地端口范围。

该属性的值是以下格式的字符串：

[*host_name*][(*low_port*)[,*high_port*])]

变量含义如下所示：

host_name
要用于连接的本地网络接口的主机名或 IP 地址。

仅当运行应用程序的系统有两个或更多个网络接口并且您需要指定连接必须使用的接口时，才需要提供此信息。 如果系统只有一个网络接口，那么只能使用此接口。 如果系统有两个或更多个网络接口，并且未指定必须使用的接口，那么会随机选择接口。

low_port
要用于连接的本地端口号。

如果还指定了 *high_port*，那么会将 *low_port* 解释为端口号范围内的最小端口号。

high_port
端口号范围内的最大端口号。 连接必须使用指定范围内的某一个端口。

此字符串的最大长度为 48 个字符。

下面是该属性的几个有效值示例：

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

缺省情况下，不设置该属性。

此属性仅适用于应用程序在客户机方式下连接到队列管理器的情况。

XMSC_WMQ_MESSAGE_SELECTION

数据类型：
System.Int32

相关属性：
ConnectionFactory

确定消息选择是由 XMS 客户机完成还是由代理完成。

该属性的有效值如下所示：

有效值	含义
XMSC_WMQ_MSEL_CLIENT	消息选择由 XMS 客户机完成。
XMSC_WMQ_MSEL_BROKER	消息选择由代理程序完成。

缺省值为 XMSC_WMQ_MSEL_CLIENT。

仅在发布/预订域中，该属性才适用。 如果将 XMSC_WMQ_BROKER_VERSION 属性设置为 XMSC_WMQ_BROKER_V1，那么不支持由代理程序选择消息。

XMSC_WMQ_MSG_BATCH_SIZE

数据类型：
System.Int32

相关属性：
ConnectionFactory

在使用异步消息传递时要成批从队列中检索的最大消息数。

当应用程序使用异步消息传递时，在某些情况下，XMS 客户机会从队列中检索一批消息，然后单独将每一条消息转发到应用程序。 此属性可指定可批量操作的最大消息数。

此属性的值是正整数，缺省值为 10。 仅当遇到特定的性能问题需要解决时，才考虑将此属性设置为其他值。

如果应用程序通过网络与队列管理器相连接，那么增大此属性的值可降低网络开销、缩短响应时间，但也会增加在客户机系统上存储消息所需的内存量。 相反，减小此属性的值可能会增加网络开销、延长响应时间，但也会减小存储消息所需的内存量。

XMSC_WMQ_POLLING_INTERVAL

数据类型：
System.Int32

相关属性：
ConnectionFactory

如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么此值是每个消息侦听器再次尝试从其队列中获取消息前经过的最大时间间隔（以毫秒计）。

如果没有合适的消息可用于会话中的任何消息侦听器的情况频繁发生，请考虑增大此属性的值。

此属性的值是正整数。 缺省值是 5000。

XMSC_WMQ_PORT

数据类型：
System.Int32

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :PORT

JMS 管理工具短名称 :PORT

队列管理器侦听入局请求的端口号。

此属性仅适用于应用程序在客户机方式下连接到队列管理器的情况。 此属性与 XMSC_WMQ_HOST_NAME 属性一起用于标识队列管理器。

此属性的缺省值为 XMSC_WMQ_DEFAULT_CLIENT_PORT 或 1414。

XMSC_WMQ_PROVIDER_VERSION

数据类型：
字符串

相关属性：
ConnectionFactory

应用程序要连接到的队列管理器的版本、发行版、修订版级别和修订包。 此属性的有效值为：

- Unspecified

或者以下某个格式的字符串：

- V.R.M.F
- V.R.M
- V.R
- V

其中，V、R、M 和 F 是大于或等于零的整数值。

值 7 或更高表示此版本旨在用作与 IBM WebSphere MQ 7.0 队列管理器的连接的 IBM WebSphere MQ 7.0 ConnectionFactory 。 小于 7 的值（如“6.0.2.0”）表示此版本将与 V7.0 之前的队列管理器一起使用。 缺省值 unspecified 允许与任何级别的队列管理器建立连接，并根据队列管理器的功能来确定适用的属性和可用的功能。

缺省情况下，此属性设置为“unspecified”。

注：

- 如果 XMSC_WMQ_PROVIDER_VERSION 设置为 6.2，那么不进行套接字共享。
- 如果 XMSC_WMQ_PROVIDER_VERSION 设置为 7 并且服务器上的通道 SHARECNV 设置为 0，那么连接将失败。
- 如果 XMSC_WMQ_PROVIDER_VERSION 设置为 UNSPECIFIED 并且 SHARECNV 设置为 0，那么将禁用 IBM WebSphere MQ 7.0 特有功能。

IBM WebSphere MQ Client 的版本在 XMS 客户机应用程序是否可以使用 IBM WebSphere MQ 7.0 特定功能方面也起着主要作用。 下表描述了该行为。

注： 系统属性 XMSC_WMQ_OVERRIDEPROVIDERVERSION 将覆盖 XMSC_WMQ_PROVIDER_VERSION 属性。 如果您无法更改连接工厂设置，那么可以使用此属性。

表 *33: XMS* 客户机 - 使用 *IBM WebSphere MQ 7.0* 特有功能的能力。

#	XMSC_WMQ_PROVIDER_VERSION	IBM WebSphere MQ 客户机版本	IBM WebSphere MQ 7.0 功能
1	未指定	7	启用
2	未指定	6	关闭
3	7	7	启用
4	7	6	异常
5	6	6	关闭
6	6	7	关闭

XMSC_WMQ_PUB_ACK_INTERVAL

数据类型：
System.Int32

相关属性：
ConnectionFactory

发布者在 XMS 客户机请求代理程序应答之前发布的消息数。

如果减小此属性的值，那么客户机请求确认的频率会更高，而发布程序的性能会因此降低。 如果您提高该值，那么当代理失败时，客户机将需更多的时间来抛出异常。

此属性的值是正整数。 缺省值是 25。

XMSC_WMQ_QMGR_CCSID

数据类型：
System.Int32

相关属性：
ConnectionFactory

编码字符集或代码页的标识 (CCSID)，其中在消息队列接口 (MQI) 中定义的字符数据字段在 XMS 客户机与 WebSphere MQ 客户机之间交换。 此属性不适用于消息主体中的字符数据字符串。

XMS 应用程序在客户机方式下连接到队列管理器时，XMS 客户机会链接到 WebSphere MQ 客户机。 两个客户机之间交换的信息包含 MQI 中定义的字符数据字段。 在正常情况下，WebSphere MQ 客户机假定这些字段采用的是运行客户机的系统的代码页。 如果 XMS 客户机提供并预期收到采用其他代码页的这些字段，那么必须设置此属性以通知 WebSphere MQ 客户机。

当 WebSphere MQ 客户机将这些字符数据字段转发到队列管理器时，必须根据需要将其中的数据转换为队列管理器使用的代码页。 同样，当 WebSphere MQ 客户机从队列管理器接收这些字段时，必须根据需要将其中的数据转换为 XMS 客户机预期接收数据时所用的代码页。 WebSphere MQ 客户机使用此属性来转换这些数据。

缺省情况下，不设置该属性。

设置此属性相当于为支持本机 WebSphere MQ 客户机应用程序的 WebSphere MQ 客户机设置 MQCCSID 环境变量。 有关此环境变量的更多信息，请参阅 *WebSphere MQ* 客户机。

XMSC_WMQ_QUEUE_MANAGER

数据类型：
字符串

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :QMANAGER

JMS 管理工具短名称 :QMGR

要连接的队列管理器的名称。

缺省情况下，不设置该属性。

XMSC_WMQ_RECEIVE_CCSID

用于设置队列管理器消息转换的目标 CCSID 的目标属性。 除非将 XMSC_WMQ_RECEIVE_CONVERSION 设置为 WMQ_RECEIVE_CONVERSION_QMGR，否则将忽略此值。

数据类型：
整数

值：
任何正整数。

缺省值为 1208。

在消息中可选择指定 GMO_CONVERT 值。 如果指定了 GMO_CONVERT 值，那么将根据指定的值来执行转换。

XMSC_WMQ_RECEIVE_CONVERSION

用于确定数据转换即将由队列管理器执行的目标属性。

数据类型：
整数

值：
XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG（缺省值）：仅在 XML 客户机上执行数据转换。 始终使用代码页 1208 执行转换。

XMSC_WMQ_RECEIVE_CONVERSION_QMGR：在将消息发送到 XMS 客户机之前在队列管理器上执行数据转换。

XMSC_WMQ_RECEIVE_EXIT

数据类型：
字符串

相关属性：
ConnectionFactory

标识要运行的通道接收出口。

此属性的值是用于标识通道接收出口的字符串，格式如下：

libraryName(entryPointName)

其中，

- **libraryName** 是受管出口 .dll 的完整路径
- entryPointName 是名称空间所限定的类名

例如，C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)

缺省情况下，不设置该属性。

此属性仅适用于应用程序在受管客户机方式下连接到队列管理器的情况。 此外，仅支持受管出口。

XMSC_WMQ_RECEIVE_EXIT_INIT

数据类型：
字符串

相关属性：
ConnectionFactory

调用通道接收出口时传递到通道接收出口的用户数据。

此属性的值是字符串。 缺省情况下，不设置该属性。

此属性仅适用于应用程序在受管客户机方式下连接到队列管理器且已设置 第 201 页的『XMSC_WMQ_RECEIVE_EXIT』 属性的情况。

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

数据类型：
字符串

相关属性：
ConnectionFactory

此属性用于获取与其相连的队列管理器的名称。

在与 CCDT（客户机通道定义表）一起使用时，此名称可能与“连接工厂”中指定的队列管理器名称不同。

XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID

数据类型：
字符串

相关属性：
ConnectionFactory

在连接后使用队列管理器标识填充此属性。

XMSC_WMQ_SECURITY_EXIT

数据类型：
字符串

相关属性：
ConnectionFactory

标识通道安全出口。

此属性的值是用于标识通道安全出口的字符串，格式如下：

libraryName(entryPointName)

其中，

- **libraryName** 是受管出口 .dll 的完整路径
- entryPointName 是名称空间所限定的类名

例如，C:\MySecurityExit.dll(MySecurityExitNameSpace.MySecurityExitClassName)

此字符串的最大长度为 128 个字符。

缺省情况下，不设置该属性。

此属性仅适用于应用程序在受管客户机方式下连接到队列管理器的情况。 此外，仅支持受管出口。

XMSC_WMQ_SECURITY_EXIT_INIT

数据类型：
字符串

相关属性：
ConnectionFactory

调用通道安全出口时传递到通道安全出口的用户数据。

用户数据字符串的最大长度为 32 个字符。

缺省情况下，不设置该属性。

此属性仅适用于应用程序在受管客户机方式下连接到队列管理器且已设置 第 202 页的『XMSC_WMQ_SECURITY_EXIT』 属性的情况。

XMSC_WMQ_SEND_EXIT

数据类型：
字符串

相关属性：
ConnectionFactory

标识通道发送出口。

此属性的值是字符串。 通道发送出口采用以下格式：

libraryName(entryPointName)

其中，

- **libraryName** 是受管出口 .dll 的完整路径

• entryPointName 是名称空间所限定的类名

例如， C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)

缺省情况下，不设置该属性。

此属性仅适用于应用程序在受管客户机方式下连接到队列管理器的情况。 此外，仅支持受管出口。

XMSC_WMQ_SEND_EXIT_INIT

数据类型：
字符串

相关属性：
ConnectionFactory

调用通道发送出口时传递到通道发送出口的用户数据。

此属性的值是由一项或多项用户数据组成的字符串（用逗号分隔）。 缺省情况下，不设置该属性。

用于指定传递到一连串通道发送出口的用户数据的规则与用于指定传递到一连串通道接收出口的用户数据的规则相同。 因此，有关这些规则的信息，请参阅第 201 页的『XMSC_WMQ_RECEIVE_EXIT_INIT』。

此属性仅适用于应用程序在受管客户机方式下连接到队列管理器且已设置 第 202 页的『XMSC_WMQ_SEND_EXIT』 属性的情况。

XMSC_WMQ_SEND_CHECK_COUNT

数据类型：
System.Int32

相关属性：
ConnectionFactory

单个非事务性 XMS 会话内两次检查异步放置错误之间允许的 Send 调用次数。

缺省情况下，此属性设置为 0。

XMSC_WMQ_SHARE_CONV_ALLOWED

数据类型：
System.Int32

相关属性：
ConnectionFactory

适用对象：
JMS 管理工具长名称 :SHARECONVALLOWED

JMS 管理工具短名称: SCALD

如果通道定义匹配，那么客户机连接是否可以与从同一进程到同一队列管理器的其他顶级 XMS 连接共享其套接字。 根据应用程序开发、维护或运行方面的需要，使用此属性可在不同套接字中完全隔离连接。 设置此属性仅仅是指示 XMS 将底层套接字共享。 并不指示有多少个连接共享一个套接字。 共享套接字的连接数由在 WebSphere MQ 客户机和 WebSphere MQ 服务器之间协商的 SHARECNV 值来确定。

应用程序可以设置以下命名常量来设置此属性：

• XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE - 连接不共享套接字。

• XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE - 连接共享套接字。

缺省情况下，此属性设置为 XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED。

此属性仅适用于应用程序在客户机方式下连接到队列管理器的情况。

XMSC_WMQ_SSL_CERT_STORES

数据类型：
字符串

相关属性：
ConnectionFactory

用于保存与队列管理器的 SSL 连接上使用的证书撤销列表 (CRL) 的服务器的位置。

此属性的值是由一个或多个 URL 组成的列表（用逗号分隔）。 每个 URL 均采用以下格式：

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

此格式与基本 MQJMS 格式兼容，但扩展了基本 MQJMS 格式。

它可具有空的 `serveraddress`。 在这种情况下，XMS 假定该值为字符串“localhost”。

下面是一个示例列表：

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
ldap://
ldap://:389
```

仅适用于 .NET : 从 IBM MQ 8.0，到 IBM MQ (WMQ_CM_CLIENT) 的受管连接和到 IBM MQ (WMQ_CM_CLIENT_UNMANAGED) 的非受管连接都支持 TLS/SSL 连接。

缺省情况下，不设置该属性。

相关信息
非受管 .NET 客户机的 SSL 和 TLS 支持
针对受管 .NET 客户机的 SSL 和 TLS 支持

XMSC_WMQ_SSL_CIPHER_SPEC

数据类型：
字符串

相关属性：
ConnectionFactory

到队列管理器的安全连接上要使用的 CipherSpec 名称。

下表列出了可与 IBM WebSphere MQ TLS 支持一起使用的密码规范。 当您请求个人证书时，您为公用和专用密钥对指定密钥大小。 除非由 CipherSpec 确定，否则 SSL 握手期间使用的密钥大小即是证书中存储的大小，如下表中所述。 缺省情况下，不设置该属性。

CipherSpec 名称	使用的协议	散列算法	加密算法	加密位	FIPS[1]	套件 B 128 位	套件 B 192 位
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Yes	否	否
TLS_RSA_WITH_AES_256_CBC_SHA[2]	TLS 1.0	SHA-1	AES	256	Yes	否	否
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	否	否	否
TLS_RSA_WITH_3DES_EDE_CBC_SHA[4]	TLS 1.0	SHA-1	3DES	168	Yes	否	否
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Yes	否	否
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Yes	否	否
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	否	否
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	AES	256	Yes	否	否

CipherSpec 名称	使用的协议	散列算法	加密算法	加密位	FIPS[1]	套件 B 128 位	套件 B 192 位
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	否	否	否
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Yes	否	否
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	否	否	否
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Yes	否	否
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	否	否
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Yes	否	否
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	否	否
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Yes	否	否
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Yes	Yes	否
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Yes	否	Yes
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Yes	否	否
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Yes	否	否
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	None	0	否	否	否
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	None	0	否	否	否
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	None	0	否	否	否
TLS_RSA_WITH_NULL_NULL	TLS 1.2	None	None	0	否	否	否
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	否	否	否

CipherSpec 名称	使用的协议	散列算法	加密算法	加密位	FIPS[1]	套件 B 128 位	套件 B 192 位

注意:

1. 指定 CipherSpec 是否符合美国联邦信息处理标准 (FIPS) 140-2。 有关 FIPS 的说明以及有关如何为符合 FIPS 140-$tag7 的操作配置 WebSphere MQ 的信息，请参阅联机 IBM IBM WebSphere MQ 产品文档中的 *Federal Information Processing Standards (FIPS)* 。
2. 该 CipherSpec 无法用于保护从 WebSphere MQ Explorer 到队列管理器的连接，除非将适当的无限制策略文件应用于 Explorer 使用的 JRE。
3. 在 2007 年 5 月 19 日之前，该 CipherSpec 经 FIPS 140-2 认证。
4. 配置 WebSphere MQ 以执行符合 FIPS 140-2 标准的操作时，此 CipherSpec 可用于传输多达 32 GB 的数据，之后才终止连接并返回错误 AMQ9288。 为避免发生此错误，请避免使用三重 DES（不推荐），或在 FIPS 140-2 配置中使用此 CipherSpec 时启用密钥重置。

相关信息

保护
消息的数据完整性
指定 CipherSpec

XMSC_WMQ_SSL_CIPHER_SUITE

数据类型:
字符串

相关属性:
ConnectionFactory

到队列管理器的 TLS 连接上要使用的 CipherSuite 的名称。 协商安全连接时使用的协议取决于指定的 CipherSuite。

该属性具有以下规范值:

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- 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

该值可作为 XMSC_WMQ_SSL_CIPHER_SPEC 的替代值提供。

如果为 XMSC_WMQ_SSL_CIPHER_SPEC 指定了非空值，那么该值将覆盖 XMSC_WMQ_SSL_CIPHER_SUITE 的设置。 如果 XMSC_WMQ_SSL_CIPHER_SPEC 不包含值，那么将使用 XMSC_WMQ_SSL_CIPHER_SUITE 的值作为提供给 GSKit 的密码套件。 在这种情况下，该值会映射到等效的 CipherSpec 值，如第 56 页的『与 IBM MQ 队列管理器的连接的 CipherSuite 和 CipherSpec 名称映射』中所述。

如果 XMSC_WMQ_SSL_CIPHER_SPEC 和 XMSC_WMQ_SSL_CIPHER_SUITE 均为空，那么使用空格填充 pChDef->SSLCipherSpec 字段。

仅适用于 .NET : 从 IBM MQ 8.0，到 IBM MQ (WMQ_CM_CLIENT) 的受管连接和到 IBM MQ (WMQ_CM_CLIENT_UNMANAGED) 的非受管连接都支持 TLS/SSL 连接。

缺省情况下，不设置该属性。

相关信息
非受管 .NET 客户机的 SSL 和 TLS 支持
针对受管 .NET 客户机的 SSL 和 TLS 支持

XMSC_WMQ_SSL_CRYPTO_HW

数据类型：
字符串

相关属性：
ConnectionFactory

下面是连接到客户机系统的加密硬件的配置详细信息。

该属性具有以下规范值：

- GSK_ACCELERATOR_RAINBOW_CS_OFF
- GSK_ACCELERATOR_RAINBOW_CS_ON
- GSK_ACCELERATOR_NCIPHER_NF_OFF
- GSK_ACCELERATOR_NCIPHER_NF_ON

PKCS11 加密硬件采用下列专用格式（其中 DriverPath、TokenLabel 和 TokenPassword 是用户指定的字符串）：

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS 将不会解释或更改字符串的内容。 它会将所提供的值（最多包含 256 个单字节字符）复制到 MQSCO.CryptoHardware 字段中。

仅适用于 .NET : 从 IBM MQ 8.0，到 IBM MQ (WMQ_CM_CLIENT) 的受管连接和到 IBM MQ (WMQ_CM_CLIENT_UNMANAGED) 的非受管连接都支持 TLS/SSL 连接。

缺省情况下，不设置该属性。

相关信息
非受管 .NET 客户机的 SSL 和 TLS 支持
针对受管 .NET 客户机的 SSL 和 TLS 支持

XMSC_WMQ_SSL_FIPS_REQUIRED

数据类型：
布尔

相关属性：
ConnectionFactory

该属性的值用于确定应用程序能否使用符合非 FIPS 标准的密码套件。 如果将该属性设置为 true，那么客户机/服务器连接只能使用 FIPS 算法。

该属性可具有以下值（可转换为 MQSCO.FipsRequired 的两个规范值）：

表 *34: MQSCO.FlipsRequired* 属性值表

值	**描述**	**MQSCO.FipsRequired 的对应值**
false	可使用任何 CipherSpec。	MQSSL_FIPS_NO（缺省值）
true	此客户机连接所应用的 CipherSpec 中只能使用 FIPS 认证的密码算法。	MQSSL_FIPS_YES

XMS 会在调用 MQCONNX 之前将相关值复制到 MQSCO.FipsRequired 中。

参数 MQSCO.FipsRequired 仅在 WebSphere MQ V 6 中可用。 在 WebSphere MQ V5.3 中，如果设置了该属性，那么 XMS 不会尝试与队列管理器建立连接，而改为抛出相应的异常。

仅适用于 .NET : 从 IBM MQ 8.0，到 IBM MQ (WMQ_CM_CLIENT) 的受管连接和到 IBM MQ (WMQ_CM_CLIENT_UNMANAGED) 的非受管连接都支持 TLS/SSL 连接。

相关信息

非受管 .NET 客户机的 SSL 和 TLS 支持
针对受管 .NET 客户机的 SSL 和 TLS 支持

XMSC_WMQ_SSL_KEY_REPOSITORY

数据类型：
字符串

相关属性：
ConnectionFactory

用于存储密钥和证书的密钥数据库文件的位置。

XMS 会将最多包含 256 个单字节字符的该字符串复制到 MQSCO.KeyRepository 字段中。 WebSphere MQ 会将此字符串解释为文件名（包含完整路径）。

仅适用于 .NET : 从 IBM MQ 8.0，到 IBM MQ (WMQ_CM_CLIENT) 的受管连接和到 IBM MQ (WMQ_CM_CLIENT_UNMANAGED) 的非受管连接都支持 TLS/SSL 连接。

缺省情况下，不设置该属性。

相关信息

非受管 .NET 客户机的 SSL 和 TLS 支持
针对受管 .NET 客户机的 SSL 和 TLS 支持

XMSC_WMQ_SSL_KEY_RESETCOUNT

数据类型：
System.Int32

相关属性：
ConnectionFactory

KeyResetCount 表示在重新协商密钥之前在 SSL 对话期间发送和接收的未加密字节总数。 此字节数包括由 MCA 发送的控制信息。

XMS 会在调用 MQCONNX 之前将您为该属性提供的值复制到 MQSCO.KeyResetCount 中。

仅从 WebSphere MQ V6 开始才提供参数 MQSCO.KeyRestCount。 在 WebSphere MQ V5.3 中，如果设置了该属性，那么 XMS 不会尝试与队列管理器建立连接，而改为抛出相应的异常。

仅适用于 .NET : 从 IBM MQ 8.0，到 IBM MQ (WMQ_CM_CLIENT) 的受管连接和到 IBM MQ (WMQ_CM_CLIENT_UNMANAGED) 的非受管连接都支持 TLS/SSL 连接。

该属性的缺省值为 0，表示从不重新协商密钥。

相关信息

非受管 .NET 客户机的 SSL 和 TLS 支持
针对受管 .NET 客户机的 SSL 和 TLS 支持

XMSC_WMQ_SSL_PEER_NAME

数据类型：
字符串

相关属性：
ConnectionFactory

到队列管理器的 SSL 连接上要使用的对等方名称。

该属性没有规范值列表。 而是必须根据 SSLPEER 的规则来构建此字符串。

下面是对等方名称的示例：

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS 会在调用 MQCONNX 之前将该字符串复制到正确的单字节代码页中，并将正确的值放入 MQCD.SSLPeerNamePtr 和 MQCD.SSLPeerNameLength 中。

仅当应用程序以客户机方式连接到队列管理器时，该属性才适用。

仅适用于 .NET：从 IBM MQ 8.0，到 IBM MQ (WMQ_CM_CLIENT) 的受管连接和到 IBM MQ (WMQ_CM_CLIENT_UNMANAGED) 的非受管连接都支持 TLS/SSL 连接。

缺省情况下，不设置该属性。

相关信息

非受管 .NET 客户机的 SSL 和 TLS 支持

针对受管 .NET 客户机的 SSL 和 TLS 支持

SSLPEERNAME

XMSC_WMQ_SYNCPOINT_ALL_GETS

数据类型：
System.Boolean

相关属性：
ConnectionFactory

是否必须从同步点控制范围内的队列中检索所有消息。

该属性的有效值如下所示：

有效值	含义
false	在适当情况下，XMS 客户机可以从同步点控制范围外的队列中检索消息。
true	XMS 客户机必须从同步点控制范围内的队列中检索所有消息。

缺省值是 false。

XMSC_WMQ_TARGET_CLIENT

数据类型：
System.Int32

相关属性：
Destination

URI 中使用的名称：
targetClient

发送到目标的消息是否包含 MQRFH2 头。

如果应用程序发送了包含 MQRFH2 头的消息，那么接收应用程序必须能够处理此头。

该属性的有效值如下所示：

有效值	含义
XMSC_WMQ_TARGET_DEST_JMS	发送到目标的消息包含 MQRFH2 头。 如果应用程序正在将消息发送到另一个 XMS 应用程序、WebSphere JMS 应用程序或能够处理 MQRFH2 头的本机 WebSphere MQ 应用程序，请指定该值。

有效值	含义
XMSC_WMQ_TARGET_DEST_MQ	发送到目标的消息不包含 MQRFH2 头。 如果应用程序正在将消息发送到不能处理 MQRFH2 头的本机 WebSphere MQ 应用程序，请指定该值。

缺省值为 XMSC_WMQ_TARGET_DEST_JMS。

XMSC_WMQ_TEMP_Q_PREFIX

数据类型：
字符串

相关属性：
ConnectionFactory

用于构成应用程序创建 XMS 临时队列时创建的 WebSphere MQ 动态队列的名称的前缀。

构成前缀的规则与构成对象描述符中 **DynamicQName** 字段内容的规则相同，但最后一个非空白字符必须是星号 (*)。 如果未设置此属性，那么所使用的值为 `CSQ.*` (在 z/OS 上) 和 `AMQ.*` (在其他平台上)。 缺省情况下，不设置该属性。

仅在点到点域中，该属性才适用。

XMSC_WMQ_TEMP_TOPIC_PREFIX

数据类型：
字符串

相关属性：
ConnectionFactory 和 Destination

创建临时主题时， XMS 将生成格式为 "`TEMP/TEMPTOPICPREFIX/unique_id`" 的主题字符串，或者如果此属性包含缺省值，那么将生成此字符串 "`TEMP/unique_id`"。 通过指定非空值，可以定义特定的模型队列，以便为在该连接下创建的临时主题的订户创建受管队列。

此属性的有效值是符合以下条件的任何非空字符串：仅包含有效字符作为 IBM WebSphere MQ 主题字符串。

缺省情况下，此属性设置为 ""（空字符串）。

注： 仅在发布/预订域中，此属性才适用。

XMSC_WMQ_TEMPORARY_MODEL

数据类型：
字符串

相关属性：
ConnectionFactory

应用程序创建 XMS 临时队列时从中创建动态队列的 WebSphere MQ 模型队列的名称。

该属性的缺省值为 SYSTEM.DEFAULT.MODEL.QUEUE。

仅在点到点域中，该属性才适用。

XMSC_WMQ_WILDCARD_FORMAT

数据类型：
System.Int32

相关属性：
ConnectionFactory 和 Destination

此属性确定要使用的通配符语法版本。

将发布/预订与 IBM WebSphere MQ "*" 和 "?" 配合使用时 会被视为通配符。 然而，在将发布/预订与 IBM Integration Bus 结合使用时，会将“#”和“+”视为通配符。 此属性将取代 XMSC_WMQ_BROKER_VERSION 属性。

此属性的有效值为：

XMSC_WMQ_WILDCARD_TOPIC_ONLY
仅识别主题级别通配符，即 '#' "+" 被视为通配符。 该值与 XMSC_WMQ_BROKER_V2 相同。

XMSC_WMQ_WILDCARD_CHAR_ONLY
仅识别字符通配符，即 '*' 和 "?" 会被视为通配符。 该值与 XMSC_WMQ_BROKER_V1 相同。

缺省情况下，此属性设置为 XMSC_WMQ_WILDCARD_TOPIC_ONLY。

XMSC_WPM_BUS_NAME

数据类型：
字符串

相关属性：
ConnectionFactory 和 Destination

URI 中使用的名称：
busName

对于连接工厂，这是应用程序连接到的服务集成总线的名称；对于目标，这是存在目标的服务集成总线的名称。

对于作为主题的目标，该属性是存在相关主题空间的服务集成总线的名称。 通过 XMSC_WPM_TOPIC_SPACE 属性指定该主题空间。

如果没有对目标设置该属性，那么假定队列或相关主题空间存在于应用程序连接到的服务集成总线中。

缺省情况下，不设置该属性。

XMSC_WPM_CONNECTION_PROTOCOL

数据类型：
System.Int32

相关属性：
Connection

到消息传递引擎的连接所使用的通信协议。 该属性为只读。

该属性的可能值如下所示：

值	含义
XMSC_WPM_CP_HTTP	连接使用“基于 TCP/IP 的 HTTP”。
XMSC_WPM_CP_TCP	连接使用 TCP/IP。

XMSC_WPM_CONNECTION_PROXIMITY

数据类型：
System.Int32

相关属性：
ConnectionFactory

连接的连接距离设置。 该属性用于确定应用程序连接到的消息传递引擎与引导程序服务器之间的距离。

该属性的有效值如下所示：

有效值	连接距离设置
XMSC_WPM_CONNECTION_PROXIMITY_BUS	总线
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	集群

有效值	连接距离设置
XMSC_WPM_CONNECTION_PROXIMITY_HOST	主机
XMSC_WPM_CONNECTION_PROXIMITY_SERVER	服务器

缺省值为 XMSC_WPM_CONNECTION_PROXIMITY_BUS。

XMSC_WPM_DUR_SUB_HOME

数据类型：
字符串

相关属性：
ConnectionFactory

URI 中使用的名称：
durableSubscriptionHome

用于管理连接或目标的所有持久预订的消息传递引擎的名称。 要传递给持久订户的消息都存储在相同消息传递引擎的发布点中。

必须先为连接指定持久预订宿主，然后应用程序才能创建使用此连接的持久订户。 为目标指定的任何值将覆盖为连接指定的值。

缺省情况下，不设置该属性。

仅在发布/预订域中，该属性才适用。

XMSC_WPM_HOST_NAME

数据类型：
字符串

相关属性：
Connection

包含应用程序连接到的消息传递引擎的系统的主机名或 IP 地址。 该属性为只读。

XMSC_WPM_LOCAL_ADDRESS

数据类型：
字符串

相关属性：
ConnectionFactory

对于到服务集成总线的连接，该属性指定要使用的本地网络接口和/或要使用的本地端口/本地端口范围。

该属性的值是以下格式的字符串：

[*host_name*][(*low_port*)[,*high_port*])]

变量含义如下所示：

host_name
要用于连接的本地网络接口的主机名或 IP 地址。

仅当运行应用程序的系统有两个或更多个网络接口并且您需要指定连接必须使用的接口时，才需要提供此信息。 如果系统只有一个网络接口，那么只能使用此接口。 如果系统有两个或更多个网络接口，并且未指定必须使用的接口，那么会随机选择接口。

low_port
要用于连接的本地端口号。

如果还指定了 *high_port*，那么会将 *low_port* 解释为端口号范围内的最小端口号。

high_port
端口号范围内的最大端口号。 连接必须使用指定范围内的某一个端口。

下面是该属性的几个有效值示例：

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

缺省情况下，不设置该属性。

XMSC_WPM_ME_NAME

数据类型：
字符串

相关属性：
Connection

应用程序连接到的消息传递引擎的名称。 该属性为只读。

XMSC_WPM_NON_PERSISTENT_MAP

数据类型：
System.Int32

相关属性：
ConnectionFactory

通过连接发送的非持久消息的可靠性级别。

该属性的有效值如下所示：

有效值	可靠性级别
XMSC_WPM_MAPPING_AS_DESTINATION	由针对服务集成总线中的队列或主题空间指定的缺省可靠性级别决定。
XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT	最佳非持久
XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT	快速非持久
XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT	可靠非持久
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	可靠持久
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	有保证的持久

缺省值为 XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT。

XMSC_WPM_PERSISTENT_MAP

数据类型：
System.Int32

相关属性：
ConnectionFactory

通过连接发送的持久消息的可靠性级别。

该属性的有效值如下所示：

有效值	可靠性级别
XMSC_WPM_MAPPING_AS_DESTINATION	由针对服务集成总线中的队列或主题空间指定的缺省可靠性级别决定。
XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT	最佳非持久
XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT	快速非持久
XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT	可靠非持久
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	可靠持久
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	有保证的持久

缺省值为 XMSC_WPM_MAPPING_RELIABLE_PERSISTENT。

XMSC_WPM_PORT

数据类型：
System.Int32

相关属性：
Connection

应用程序连接到的消息传递引擎所侦听的端口号。 该属性为只读。

XMSC_WPM_PROVIDER_ENDPOINTS

数据类型：
字符串

相关属性：
ConnectionFactory

由引导程序服务器的一个或多个端点地址组成的序列。 用逗号分隔各个端点地址。

引导程序服务器是负责选择要与应用程序相连的消息传递引擎的应用程序服务器。 引导程序服务器的端点地址采用下列格式：

host_name:*port_number*:*chain_name*

端点地址的各组成部分的含义如下：

host_name
引导程序服务器所在的系统的主机名或 IP 地址。 如果未指定主机名或 IP 地址，那么缺省值为 `localhost`。

port_number
引导程序服务器侦听入局请求所使用的端口号。 如果未指定端口号，那么缺省值为 7276。

chain_name
引导程序服务器所使用的引导程序传输链的名称。 有效值如下：

有效值	引导程序传输链的名称
XMSC_WPM_BOOTSTRAP_HTTP	BootstrapTunneledMessaging
XMSC_WPM_BOOTSTRAP_HTTPS	BootstrapTunneledSecureMessaging
XMSC_WPM_BOOTSTRAP_SSL	BootstrapSecureMessaging

有效值	引导程序传输链的名称
XMSC_WPM_BOOTSTRAP_TCP	BootstrapBasicMessaging

如果未指定名称，那么缺省值为 XMSC_WPM_BOOTSTRAP_TCP。

如果未指定端点地址，那么缺省值为 localhost:7276:BootstrapBasicMessaging。

XMSC_WPM_SSL_CIPHER_SUITE

数据类型：
字符串

相关属性：
ConnectionFactory

到 WebSphere Service Integration Bus 消息传递引擎的 TLS 连接上要使用的 CipherSuite 名称。 协商安全连接时使用的协议取决于指定的 CipherSuite。

表 *35:* 到 *WebSphere Service Integration Bus* 消息传递引擎的连接所使用的 *CipherSuite* 选项

密码套件	使用的协议
TLS_RSA_WITH_DES_CBC_SHA	TLSv1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1

注意：

1. 仅在 Windows 或 Solaris 上才支持 TLS_RSA_WITH_AES_128_CBC_SHA 和 TLS_RSA_WITH_AES_256_CBC_SHA CipherSuite。 （这由 GSKit 控制。）
2. 不推荐 TLS_RSA_WITH_3DES_EDE_CBC_SHA。 但是，它仍可用于传输最多 32 GB 数据，超过此数据量之后，连接将因错误 AMQ9288 而终止。 要避免此错误，您需要避免使用三重 DES，或在使用此 CipherSpec 时启用密钥重置。

该属性没有缺省值。 如果要使用 SSL 或 TLS，那么必须指定该属性的值，否则应用程序无法成功连接到服务器。

XMSC_WPM_SSL_FIPS_REQUIRED

数据类型：
布尔

相关属性：
ConnectionFactory

该属性的值用于确定应用程序能否使用符合非 FIPS 标准的密码套件。 如果将该属性设置为 true，那么客户机/服务器连接只能使用 FIPS 算法。如果将该属性的值设置为 TRUE，可阻止应用程序使用符合非 FIPS 标准的密码套件。

缺省情况下，该属性设置为 FALSE（即关闭 FIPS 方式）。

XMSC_WPM_SSL_KEY_REPOSITORY

数据类型：
字符串

相关属性：
ConnectionFactory

包含要用于安全连接的公用和专用密钥的密钥环文件的路径。

如果将密钥环文件属性设置为特殊值 XMSC_WPM_SSL_MS_CERTIFICATE_STORE，即指定使用 Microsoft Windows 密钥数据库。 如果使用 Microsoft Windows 密钥数据库（可在**控制面板** > **Internet 选项** > **内容** > **证书**下找到），那么就不需要独立的密钥文件数据库。 不允许在 Windows x64 和其他平台上使用此常量。

缺省情况下，不设置该属性。

XMSC_WPM_SSL_KEYRING_LABEL

数据类型：
字符串

相关属性：
ConnectionFactory

向服务器进行认证时要使用的证书。 如果未指定值，那么将使用缺省证书。

缺省情况下，不设置该属性。

XMSC_WPM_SSL_KEYRING_PW

数据类型：
字符串

相关属性：
ConnectionFactory

密钥环文件的密码。

该属性可代替 XMSC_WPM_SSL_KEYRING_STASH_FILE，以用于配置密钥环文件的密码。

缺省情况下，不设置该属性。

XMSC_WPM_SSL_KEYRING_STASH_FILE

数据类型：
字符串

相关属性：
ConnectionFactory

包含密钥存储库文件密码的二进制文件的名称。

该属性可代替 XMSC_WPM_SSL_KEYRING_PW，以用于配置密钥环文件的密码。

缺省情况下，不设置该属性。

XMSC_WPM_TARGET_GROUP

数据类型：
字符串

相关属性：
ConnectionFactory

消息传递引擎的目标组名称。 目标组的性质由 XMSC_WPM_TARGET_TYPE 属性决定。

如果要将消息传递引擎搜索范围限制为服务集成总线中的消息传递引擎子组，请设置该属性。 如果您希望应用程序能够连接到服务集成总线中的任何消息传递引擎，请勿设置该属性。

缺省情况下，不设置该属性。

XMSC_WPM_TARGET_SIGNIFICANCE

数据类型：
System.Int32

相关属性：
ConnectionFactory

消息传递引擎的目标组的重要性。

该属性的有效值如下所示：

有效值	含义
XMSC_WPM_TARGET_SIGNIFICANCE_ PREFERRED	如果目标组包含可用的消息传递引擎，那么将选择该消息传递引擎。 否则，将选择目标组外的消息传递引擎，前提是此消息传递引擎位于相同的服务集成总线中。
XMSC_WPM_TARGET_SIGNIFICANCE_ 必需	所选消息传递引擎必须位于目标组中。 如果目标组中的消息传递引擎不可用，那么连接过程将失败。

该属性的缺省值为 XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED。

XMSC_WPM_TARGET_TRANSPORT_CHAIN

数据类型：
字符串

相关属性：
ConnectionFactory

应用程序连接到消息传递引擎时必须使用的入站传输链的名称。

该属性的值可以是用于托管消息传递引擎的应用程序服务器中任何可用入站传输链的名称。 针对每个预定义的入站传输链，提供了以下命名常量：

命名常量	传输链名称
XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC	InboundBasicMessaging

该属性的缺省值为 XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC。

XMSC_WPM_TARGET_TYPE

数据类型：
System.Int32

相关属性：
ConnectionFactory

消息传递引擎的目标组类型。 该属性将决定由 XMSC_WPM_TARGET_GROUP 属性标识的目标组的性质。

该属性的有效值如下所示：

有效值	含义
XMSC_WPM_TARGET_TYPE_BUSMEMBER	目标组的名称是总线成员的名称。 目标组由总线成员中的所有消息传递引擎组成。
XMSC_WPM_TARGET_TYPE_CUSTOM	目标组的名称是用户定义的消息传递引擎组的名称。 目标组由向用户定义组注册的所有消息传递引擎组成。
XMSC_WPM_TARGET_TYPE_ME	目标组的名称是消息传递引擎的名称。 目标组由指定的消息传递引擎组成。

缺省情况下，不设置该属性。

XMSC_WPM_TEMP_Q_PREFIX

数据类型：
字符串

相关属性：
ConnectionFactory

用于构成应用程序创建 XMS 临时队列时在服务集成总线中创建的临时队列的名称的前缀。 此前缀最多可包含 12 个字符。

临时队列名称以字符“_Q”开头，后接此前缀。 此名称的其余部分由系统生成的字符组成。

缺省情况下，不设置该属性，这意味着临时队列名称没有前缀。

仅在点到点域中，该属性才适用。

XMSC_WPM_TEMP_TOPIC_PREFIX

数据类型：
字符串

相关属性：
ConnectionFactory

用于构成应用程序创建的临时主题名称的前缀。 此前缀最多可包含 12 个字符。

临时主题名称以字符“_T”开头，后接此前缀。 此名称的其余部分由系统生成的字符组成。

缺省情况下，不设置该属性，这意味着临时主题名称没有前缀。

仅在发布/预订域中，该属性才适用。

XMSC_WPM_TOPIC_SPACE

数据类型：
字符串

相关属性：
Destination

URI 中使用的名称：
topicSpace

包含主题的主题空间的名称。 只有作为主题的目标才具有该属性。

缺省情况下，不设置该属性，这意味着将使用缺省主题空间。

仅在发布/预订域中，该属性才适用。

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能。 有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。 任何对 IBM 产品、程序或服务的引用并非意在明示或默示只能使用 IBM 的产品、程序或服务。 只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以代替 IBM 产品、程序或服务。 但是， 评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。 提供本文档并未授予用户使用这些专利的任何许可。 您可以以书面形式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

知识产权许可
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 063-8506 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的 (无论是明示的还是暗含的) 保证，包括但不限于暗含的有关非侵权，适销和适用于某种特定用途的保证。 某些国家或地区在某些交易中不允许免除明示或暗含的保证。 因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。 此处的信息将定期更改；这些更改将编入本资料的新版本中。 IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式 充当对那些 Web 站点的保证。 那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Corporation
软件互操作性协调员，部门 49XA
北纬 3605 号公路
罗切斯特，明尼苏达州 55901
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。 因此，在其他操作环境中获得的数据可能会有明显的不同。 有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。 此外，有些测量是通过推算而估计的， 实际结果可能会有差异。 本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中 获取。 IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。 有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含日常商业运作所使用的数据和报表的示例。 为了尽可能全面地说明这些数据和报表，这些示例包括个人、公司、品牌和产品的名称。 所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有任何雷同，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。 如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。 这些示例并未在所有条件下作全面测试。 因此，IBM 不能担保或默示这些程序的可靠性、可维护性或功能。

如果您正在查看本信息的软拷贝，图片和彩色图例可能无法显示。

编程接口信息

编程接口信息 (如果提供) 旨在帮助您创建用于此程序的应用软件。

本书包含有关允许客户编写程序以获取 WebSphere MQ 服务的预期编程接口的信息。

但是，该信息还可能包含诊断、修改和调优信息。 提供诊断、修改和调优信息是为了帮助您调试您的应用程序软件。

要点: 请勿将此诊断，修改和调整信息用作编程接口，因为它可能会发生更改。

商标

IBM IBM 徽标 ibm.com 是 IBM Corporation 在全球许多管辖区域的商标。 当前的 IBM 商标列表可从 Web 上的 “Copyright and trademark information”www.ibm.com/legal/copytrade.shtml 获取。 其他产品和服务名称可能是 IBM 或其他公司的商标。

Microsoft 和 Windows 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 Open Group 在美国和其他国家或地区的注册商标。

Linux® 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

此产品包含由 Eclipse 项目 (http://www.eclipse.org/) 开发的软件。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其附属公司的商标或注册商标。

部件号:

(1P) P/N: