

# Novell exteNd Composer™ JMS Connect

4.2

ユーザガイド

[www.novell.com](http://www.novell.com)



Novell®

## 保証と著作権

Copyright ©1999, 2000, 2001, 2002, 2003 SilverStream Software, LLC. All rights reserved.

SilverStream ソフトウェア製品は、SilverStream Software, LLC により著作権とすべての権利が保留されています。

SilverStream は SilverStream Software, LLC の登録商標です。Novell は、Novell, Inc. の登録商標です。

ソフトウェアとマニュアルの所有権、および特許、著作権、およびそれに関連するその他のすべての財産権は常に、単独で排他的に SilverStream とそのライセンサーに保留され、当該所有権と矛盾するいかなる行為も行わないものとします。本ソフトウェアは、著作権法と国際条約規定で保護されています。ソフトウェアならびにそのマニュアルからすべての著作権に関する通知とその他の所有権に関する通知を削除してはならず、ソフトウェアとそのマニュアルのすべてのコピーまたは抜粋に当該通知を複製しなければなりません。本ソフトウェアのいかなる所有権も取得するものではありません。

Jakarta-Regexp Copyright ©1999 The Apache Software Foundation. All rights reserved. Ant Copyright ©1999 The Apache Software Foundation. All rights reserved. Xalan Copyright ©1999 The Apache Software Foundation. All rights reserved. Xerces Copyright ©1999-2000 The Apache Software Foundation. All rights reserved. Jakarta-Regexp, Ant, Xalan, Crimson, および Xerces ソフトウェアは、The Apache Software Foundation によりライセンスを付与され、Jakarta-Regexp, Ant, Xalan, Crimson, および Xerces のソースおよびバイナリ形式での再配布および使用は、変更のあるなしにかかわらず、以下の条件が満たされることを前提として許可されます。1. ソースコードの再配布に上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知が記載されていること。2. バイナリ形式の再配布では上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知がマニュアルまたは配布の際に提供されるその他の資料、あるいはその両方に記載されていること。3. エンドユーザの資料には、適宜、以下の通知を再配布の際に含めてください。「この製品には、Apache Software Foundation (<http://www.apache.org/>) により開発されたソフトウェアが含まれています」代わりに、この謝辞をソフトウェア自体に表示し、当該サードパーティに対する謝辞が通常表示される場所に表示することもできます。4. 「The Jakarta Project」、「Jakarta-Regexp」、「Xerces」、「Xalan」、「Ant」、および「Apache Software Foundation」は、書面による事前の許可なく、このソフトウェアから派生する製品を推薦したり、販売促進したりするのに使用してはなりません。書面による許可については、[apache@apache.org](mailto:apache@apache.org) <<mailto:apache@apache.org>> にお問い合わせください。5. 本ソフトウェアから派生する製品は「Apache」と呼ばれてはならず、「Apache」は The Apache Software Foundation の事前の書面による許可なくその名前に使用することはできません。本ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性や特定の目的に対する適合性に対する暗黙の保証も行われません。いかなる場合でも、Apache Software Foundation またはその関係者はいかなる直接的、間接的、偶発的、特別な、免除的、または結果的な損害（代替品やサービスの調達、使用機会、データ、または利益の喪失、または業務の中断などを含む）についても、理論上責任がある場合でも、契約上の責任がある場合でも、厳密な責任、または瑕疵（怠慢などを含む）があった場合でも、ソフトウェアの使用の過程で生じ、当該損害の可能性を助言した場合であっても、責任をもちません。

Copyright ©2000 Brett McLaughlin & Jason Hunter. All rights reserved. ソースおよびバイナリ形式での再配布および使用は、変更のあるなしにかかわらず、以下の条件が満たされることを前提として許可されます。1. ソースコードの再配布に上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知が記載されていること。2. バイナリ形式の再配布では上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知がマニュアルまたは配布の際に提供されるその他の資料、あるいはその両方に記載されていること。3. 「JDOM」という名前は、書面による事前の許可なく、このソフトウェアから派生する製品を推薦したり、販売促進したりするのに使用してはなりません。書面による許可については、[license@jdom.org](mailto:license@jdom.org) <<mailto:license@jdom.org>> にお問い合わせください。4. 本ソフトウェアから派生する製品は「JDOM」と呼ばれてはならず、「JDOM」は JDOM Project Management ([pm@jdom.org](mailto:pm@jdom.org)) <<mailto:pm@jdom.org>> の事前の書面による許可なくその名前に使用することはできません。本ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性や特定の目的に対する適合性に対する暗黙の保証も行われません。いかなる場合でも、Apache Software Foundation またはその関係者はいかなる直接的、間接的、偶発的、特別な、免除的、または結果的な損害（代替品やサービスの調達、使用機会、データ、または利益の喪失、または業務の中断などを含む）についても、理論上責任がある場合でも、契約上の責任がある場合でも、厳密な責任、または瑕疵（怠慢などを含む）があった場合でも、ソフトウェアの使用の過程で生じ、当該損害の可能性を助言した場合であっても、責任をもちません。

Sun Microsystems, Inc. Sun, Sun Microsystems, Sun Logo Sun、Sun のロゴ、Sun Microsystems、JavaBeans、Enterprise JavaBeans、JavaServer Pages、Java Naming and Directory Interface、JDK、JDBC、Java、HotJava、HotJava Views、Visual Java、Solaris、NEO、Joe、Netra、NFS、ONC、ONC+、OpenWindows、PC-NFS、SNM、SunNet Manager、Solaris sunburst design、Solstice、SunCore、SolarNet、SunWeb、Sun Workstation、The Network Is The Computer、ToolTalk、Ultra、Ultracomputing、Ultraserver、Where The Network Is Going、SunWorkShop、XView、Java WorkShop、Java Coffee Cup のロゴ、Visual Java、および NetBeans は、米国およびその他の国の Sun Microsystems, Inc. の商標ならびに登録商標です。

Copyright ©2001 Extreme!Lab, Indiana University License. <http://www.extreme.indiana.edu>. 同社により許可が無料で、Indiana University ソフトウェアと関連する Indiana University のドキュメントファイル (「IU Software」) のコピーを取得したすべての人に、制限なく IU Software を取り扱うために付与されます。その際に、IU Software の使用、コピー、変更、マージ、公開、配布、サブライセンス、または販売、あるいはそれらのすべてに関する権利に制限はなく、IU Software が指定した人に以下の条件に基づき権利を付与します。上記の著作権に関する通知とその許可に関する通知は、IU Software のすべてのコピーおよび主要部分に含まれる必要があります。本 IU ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性、特定の目的に対する適合性や権利侵害がないことに対する暗黙の保証も行われません。いかなる場合でも、作成者または著作権所有者は、契約上の責任がある場合でも、厳密な責任、または瑕疵 (怠慢などを含む) があつた場合でも、UI Software に関連して、または UI Software の使用やその他の取引の過程で生じた場合であっても、クレーム、損害、その他の責任について責任をもちません。

本ソフトウェアは、著作権をもつ SSLavaTM Toolkit の一部です。Copyright ©1996-1998 by Phaos Technology Corporation. All rights reserved.

Copyright © 1994-2002 W3C® (Massachusetts Institute of Technology, Institut National de Recherche Informatique et en Automatique, Keio University), all Rights Reserved. <http://www.w3.org/consortium/legal>. この W3C の成果物 (ソフトウェア、ドキュメント、またはその他の関連品目を含む) は、以下のライセンスの下で著作権所有者により提供されています。この成果物の取得、使用、またはコピー、あるいはそれらのすべてにより、ライセンサーは以下の条件を読み、理解し、遵守することに合意するものとします。本ソフトウェアとそのドキュメントの使用、コピー、変更、および配布は、変更のあるなしにかかわらず、いかなる目的でも無料または本契約で許可された使用料をもって許可されます。ただし、変更箇所を含む本ソフトウェアとドキュメントのすべてまたはその一部に以下のとおり記述することを前提とします。1. この通知の全文は、再配布物または派生物のユーザが見やすい場所に掲示しなければなりません。2. すべての前もって存在する知的所有権の放棄、通知、または条件。存在しない場合は、以下の形式の短い通知 (ハイパーテキストが望ましい、テキストでも良い) を再配布または派生コードの本文内で使用しなければなりません。「Copyright © [Date-of-software] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All rights reserved. <http://www.w3.org/Consortium/Legal/>」3. W3C のファイルに変更または修正を加えた場合はその日付を含む通知 (コードが派生する場所への URI を示すことをお勧めします)。本ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性、特定の目的に対する適合性やサードパーティの特許、著作権、商標またはそのたの権利を侵害しないことに対する暗黙の保証も行われません。著作権の所有者は本ソフトウェアまたはマニュアルの使用の結果生じる、直接的、間接的、特殊な、または結果的な損害に対していかなる責任も負いません。著作権所有者の名前および商標は、特別な書面による事前の承諾なしにソフトウェアに関する広告や広報に使用してはなりません。本ソフトウェアおよび関連する資料の著作権の所有権は常に、著作権所有者に帰属するものとします。

米国 Novell, Inc.  
1800 South Novell Place  
Provo, UT 85606

[www.novell.com](http://www.novell.com)

JMS Connect ユーザガイド  
2003 年 1 月  
000-000000-000

**オンラインマニュアル**： この製品およびその他の Novell 製品のオンラインマニュアルや更新情報については、[www.novell.com/documentation](http://www.novell.com/documentation) を参照してください。

# 目次

このガイドについて	9
<b>1 exteNd Composer と JMS へようこそ</b>	<b>11</b>
exteNd Connect について	12
JMS Connect とは	13
JMS で対応できるニーズ	13
企業メッセージングとは	14
メッセージキューとは	15
メッセージベースのアプリケーションの遅延	16
メッセージングの信頼性	16
メッセージをトランザクションの一部として使用する	16
ポイントツーポイントメッセージングとは	17
パブリッシュ / サブスクライブメッセージングとは	18
配信の保証について	19
メッセージ構造	20
ヘッダ情報	20
本文のタイプ	21
メッセージの取り出し	22
メッセージフィルタ	22
リクエスト - 応答と保存 / 転送	23
JMS の定義範囲	24
exteNd の JMS コンポーネントについて	24
<b>2 JMS コンポーネントエディタをお使いになる前に</b>	<b>25</b>
JMS 接続リソースの作成	25
式駆動型の接続について	26
キュー接続について	27
トピック接続について	34
コンポーネントに対する XML テンプレートの作成	40
<b>3 JMS コンポーネントの作成</b>	<b>41</b>
JMS コンポーネントを作成する前に	41
JMS コンポーネントエディタウィンドウについて	45
ネイティブ環境ペインについて	46
<b>4 JMS アクションの作成</b>	<b>49</b>
アクションについて	49
JMS コンポーネントエディタに固有なアクション	50
オプションタブ	50
[Message Body] タブ	51
[Message Header] タブ	52

Send Message アクション	53
優先度、モード、および有効期限	53
送信先キュー/トピック	53
返信用アドレス	56
Browse Messages アクション	62
Receive Message アクション	67
Message Transaction アクション	73
Commit ステートメントを発行した場合	73
Rollback ステートメントを発行した場合	73
セッションを未解決のままにした場合	73
Message Transaction に含まれるアクション	74
Message Transaction アクションの対象	75
JMS コンポーネントエディタでの他のアクションの使用	77

## **5 メッセージの操作** **79**

メッセージヘッダへのデータのマップ	80
ヘッダにマップする場合の制約	81
カスタムプロパティへのデータのマップ	82
プロパティのマップの制約	83
XML メッセージの操作	84
コピーブックメッセージの操作	89
コピーブックメッセージの設定	89
コピーブックとネイティブ環境ペイン	90
コピーブック固有のコンテキストメニュー項目	92
コピーブックと DOM 間のデータのマップ	94
メッセージフィルタ (セレクタ) の操作	96
フィルタの制約	98
本文のタイプによるフィルタ	98
リクエスト - 応答メッセージ	99
一時キュー	100
ECMAScript と JMS Connect	102
ECMAScript メソッドの概要	105

## **6 JMS サービス** **109**

JMS サービスについて	109
複数のリスナ	110
JMS サービスの作成	110
JMS サービスの配備	114
配備された JMS サービスをどのように管理しますか？	115

## **A JMS 用語集** **117**

## **B メッセージセレクタ構文** **123**

リテラル	123
識別子	123

## **6 JMS Connect ユーザガイド**

式	124
比較	125
Null 値	126
特記事項	126
<b>C</b> <b>メッセージヘッダおよびプロパティ</b>	<b>127</b>
JMS によって定義されるヘッダフィールド	127
JMSCorrelationID	127
JMSDeliveryMode	127
JMSDestination	127
JMSExpiration	128
JMSMessageID	128
JMSPriority	128
JMSRedelivered	128
JMSReplyTo	128
JMSTimestamp	129
JMSType	129
メッセージプロパティ	129
JMS 定義のプロパティ	129
プロバイダ固有のプロパティ	130
ユーザ定義のプロパティ	130





# このガイドについて

## 目的

このガイドでは、exteNd Composer JMS Connect の設計時アプリケーションの一部である JMS コンポーネントエディタの使用方法を説明します。

## 対象読者

このガイドは、メッセージ指向ミドルウェア (MOM) を必要とするアプリケーションまたはサービスを構築するシステムアナリスト、プログラマ、およびその他の担当者を対象としています (対象となるホストの MOM システムが Sun Microsystems 社の Java Message Service API に対応していることが必要です)。

## 前提条件

このガイドでは、exteNd Composer の設計時環境および Composer のアプリケーション構築例についての予備知識が必要です。また、MOM および JMS の概念についてすでに理解されていることが前提となります。

## 追加のドキュメント

[Novell exteNd Director](#) に関する完全なドキュメンテーションについては、次の Novell マニュアルの Web サイトを参照してください。

<http://www.novell.com/documentation-index/index.jsp>



# 1

## exteNd Composer と JMS へようこそ

『Novell exteNd JMS Connect ユーザガイド』へようこそ。このガイドは、Composer の全機能(いくつかの Connect コンポーネントエディタを除く)の使用 방법이詳しく説明されている『exteNd Composer ユーザガイド』に付属しています。そのため、『Composer ユーザガイド』をご覧になっていない場合は、このガイドを使用する前に読んで内容を確認してください。

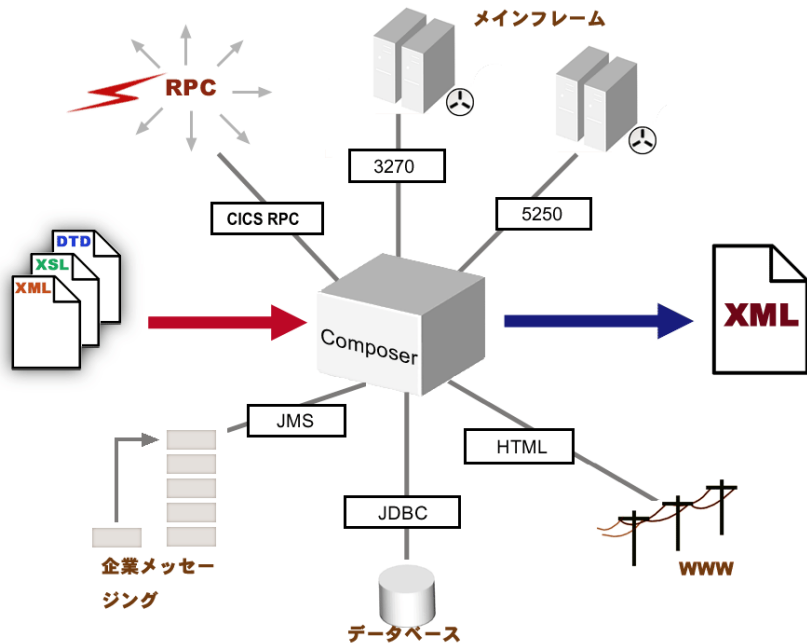
exteNd Composer には、JMS Connector など、Connector ごとに異なるコンポーネントエディタが用意されています。各コンポーネントエディタの特殊な機能は、これと同じような個々のガイドで説明されています。

作業を始める前に、まず JMS Connect を既存の exteNd Composer にインストールしておく必要があります。また、この Connector で作成されたサービスを exteNd Server 環境で実行するには、この Connector 用のサーバ側ソフトウェアが exteNd Server にインストールされている必要があります。

**注記：** このコンポーネントエディタを正しく使用するには、メッセージ指向ミドルウェア (MOM) の概念、および配備する特定の MOM 環境 (MQSeries など) に慣れ親しんでおく必要があります。ここからは、企業メッセージングの概念を簡単に説明しますが、包括的な手順についてはこのマニュアルでは説明しません。このマニュアルでの説明は、JMS プロバイダのマニュアルに替わるものではありません。

## exteNd Connect について

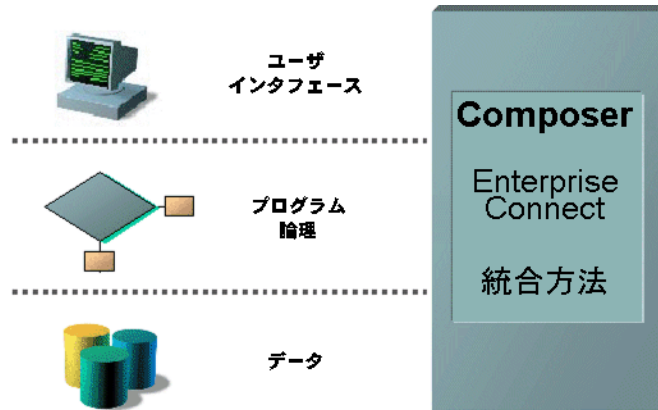
Novell exteNd は、単純なハブ & スポークアーキテクチャに基づいて構築されています(次の図を参照)。ハブは、XML ドキュメントを受け付けてドキュメントを処理し、XML ドキュメントを返す、強力な XML 変換エンジンです。スポーク(つまり Connect)は、XML 対応でないデータのソースを「XML に対応させる」プラグインモジュールで、データをハブに送信して XML として処理します。これらのデータソースには、レガシー COBOL/VSAM で管理されている情報から、HTML ページに対するメッセージキューまで何でも使用できます。exteNd Connect は、情報ソースを XML に対応させるために各製品で使用されている統合方法に従って分類できます。統合方法は、インターネットベースのコンピュータアーキテクチャに対する現在のシステム設計において使用される主要な区分を反映したものです。exteNd では、B2Bi のニーズに応じて、ユーザインタフェースレベル、プログラム論理レベル、またはデータレベルでビジネスシステムを統合できます。



ハブとスポークの構造により、exteNd では、Connect (EE) を介して企業規模で XML を統合できます。

# JMS Connect とは

Java Messaging Service (JMS) は、IBM の MQSeries や Progress Software の SonicMQ などで提供されるメッセージ指向ミドルウェア (MOM) サービス用の Java ベースのインタフェースです。Java アプリケーションサーバで実行する分散アプリケーションでメッセージシステムを十分に利用できるように、Java 言語クライアントおよび Java 中間層サービスでは、企業メッセージング製品と「通話」する共通の方法が必要です。JMS では、この機能を提供しています。



*Novell exteNd では、ユーザインタフェースレベル、プログラム論理レベル、またはデータレベルでビジネスシステムを統合できます。*

exteNd JMS Connect では、トランザクションまたは非トランザクションセッションを使用して、JMS ベースの MOM システムにより管理されるキューでメッセージを送信、受信、または参照するコンポーネントを作成できます。JMS 対応の exteNd サービスは、企業メッセージング特有の非同期処理およびトランスポート層の両方を利用できます。JMS Connect を使用すると、オブジェクトのリモート呼び出し、「確実な一度だけ」の通知の配信および分散トランザクションなどの複雑になる可能性のある操作を行いながら、システムリソースを十分に利用できる強力で柔軟なアプリケーションを作成できます。

## JMS に対応できるニーズ

JMS 標準は、次のような、いくつかの目的のために作成されました。

- ◆ 既存の MOM 製品に対応するメッセージの作成および操作に適切なアプリケーションプログラミングインタフェースを提供する。
- ◆ Java オブジェクトを含むメッセージなど、異なるメッセージ内容タイプをサポートする。

- ◆ オペレーティングシステム、マシンアーキテクチャ、転送メカニズム、およびコンピュータ言語などの様々なアプリケーションの開発を単純化する。

JMS は、JNDI (Java 名およびディレクトリインタフェース) が既存の名前およびディレクトリサービスの層を成すのと同じように、広範囲の既存および将来のメッセージ指向ミドルウェアシステムの層を成す、広範囲対応の Java API です。

JMS の完全な仕様は、<http://java.sun.com/products/jms/> で利用できます。

## 企業メッセージングとは

企業メッセージングシステムは、メッセージの転送および保存をサポートします。ここでは、「メッセージ」は、主に (ユーザではなく) 企業アプリケーションにより生産および消費される情報のパケットを表します。メッセージは、キーと値のペア、XML ドキュメント、シリアル化 Java オブジェクト、または任意のバイトストリームを含みます。

メッセージ指向ミドルウェアの主な特徴の 1 つに、通信プロトコルの異なるネットワーク上で通信する必要のある様々なクライアントからのメッセージ転送および配信の詳細を隠す抽象層となる機能があります。通信ゲートウェイとして機能することにより、MOM では、クライアントの分散アプリケーション開発の妨げとなる接続問題が解決されます。

また、MQSeries や SonicMQ のような製品をサポートする企業メッセージングには、「非同期」でプロセスをリンクする機能があります。非同期プロセスとは、パーティ間でのデータ交換が直接的に、リアルタイムでも行われなことを表します。非同期プロセスの他には、「同期」プロセスがあります。同期プロセスでは、ホストとクライアント (または個人と個人) が、そのセッション中に中断されることなく、連続して通信している「必要があります」(同期交換の例としては、CICS 環境でのリモートプロシージャコールの使用があります)。同期操作は一部の通信で必要ですが、参加者間での同期通信が「必要でない」ビジネスプロセスはたくさんあります。このようなプロセスでは、通常、非同期通信は、リソースを最大限効率的に使用し、システムの生産性を向上させます。

同期プロセスは、すべての客が会話で直接シェフに料理を注文し、調理場では料理中以外の注文を受け入れないレストランのようなものです。客は、列に並んで、1 人ずつ順番を待たなければならない、料理は 1 つずつ提供されます。また、非同期プロセスは、客、調理場、バーに同時に注文を伝えるウェイターとウェイトレスにも例えることができます。この例では、ウェイターは、調理場へのメッセージングチャンネルとして機能し、注文は「並べられ」、料理は別々に提供されます。また、この例では (多くのビジネスプロセスと同様に)、注文を同期ではなく非同期で処理した方が、より効率的です。

メッセージングシステムでの時間領域の分割により、操作が可能な限り堅固で、安全なものになります。パーティは、他のパーティがメッセージを送信（または受信）すると、ビジー状態（またはオフライン）になります。送信側は、受信側により認知を待たずに、処理を続けることができます。ネットワークやサーバは、ダウンしますが、メッセージ転送や受信には影響ありません。

## メッセージキューとは

非同期メッセージングでは、メッセージがクライアント自体ではなく、「キュー」に送信されます。キューは、キューを使用するクライアントプロセスとは独立して存在します。

「キュー」とは、データ要素（この場合メッセージ）が最終的な受信のために保存される、待機領域または待機場のことです。MOM 環境では、クライアントアプリケーションで、メッセージキューがどのように構造、保守、保存されるかを認識する必要はありません。キュー管理の詳細は、MOM ベンダ（または「JMS プロバイダ」）が扱います。通常、キューは、サーバノードのように、信頼性、スケーラビリティ、および負荷バランスのために「クラスタ化」されます。

FIFO（先入れ先出し）および LIFO（後入れ先出し）キューは、コンピューティングで有名な構造ですが、「メッセージキューでは取り出し順序は決まっています」。取り出し順序は、自由です。つまり、取り出し順序（つまり消費順序）がクライアントのニーズにより決まるように、メッセージの優先度をカスタムで指定できます。正しく説明すると、この機能は、全体のシステム運用をより効率的にします。優先度の低いメッセージの処理は、システムリソースが使用できるようになるまで遅らせることができます。そのため、優先度の低いメッセージが、優先度の高いメッセージに影響を与えることはありません。

メッセージをキューから削除せずに参照することを「ブラウズ」と呼びます。

**注記：**メッセージがキューに入っている時間、キューに入れることができるメッセージの最大数、リソース超過の処理方法は、JMS 標準では定義されます。これらの詳細については、MOM ベンダのドキュメンテーションを参照してください。

## メッセージベースのアプリケーションの遅延

すべてのメッセージングシステムで遅延が発生することがありますが、これは、メッセージングを使用するアプリケーションで必ず遅延が発生するというものではありません。メッセージングで使用できる非同期プロセスでは、アプリケーション内でのマルチタスク処理が可能です。これにより、スループットが向上されます(アプリケーションにより異なります)。たとえば、顧客がショッピングカートに品物を入れる、ショッピングカートアプリケーションで在庫チェックコンポーネントを起動する、コンポーネントで送料を計算する、コンポーネントで顧客情報をデータベースから取り出すなど、これらすべての操作を同時に処理することができます。

メッセージングモデル(ポイントツーポイントとパブリッシュ/サブスクライブ)の選択は、待ち時間にとって重要になります。以下の「ポイントツーポイントメッセージングとは」および「パブリッシュ/サブスクライブメッセージングとは」を参照してください。

## メッセージングの信頼性

メッセージ指向ミドルウェアで提供されるサービス保証の品質は様々ですし、実際の信頼性は管理問題(クラスタサイズや使用できるリソースなど)により異なりますが、「すべての」JMS ベースのメッセージングサービスでは、信頼性が優先されるアプリケーションのオプションとして、「確実な一度だけ」のメッセージ配信を提供する必要があります。また、JMS では、サービスの品質を落とすこともできるので、確実な一度だけの送信よりも、配信の「速度」が重要な場合に、このようなソリューションを構築できます。そのため、JMS ベースのメッセージングソリューションの信頼性は、設定により様々です。

強い信頼性の保証は、JMS ベースのシステムで共通した特徴です。

## メッセージをトランザクションの一部として使用する

信頼性において JMS メッセージングを魅力的なものにする要因の 1 つに、メッセージセッションに「トランザクション制御」を追加できるということがあります。トランザクションセッションは、生成または消費されるメッセージの任意のセットを 1 つの論理的な作業単位にまとめます。トランザクションがコミットされると、メッセージに関するすべての入力の確認され、すべての出力が送信されます。Transaction アクションのメッセージセッションがロールバックされると、生成されたメッセージはすべて破棄され、セッション中に使用されたメッセージはすべて回復されます。



たとえば、アプリケーションで、5 つメッセージを1つのグループにまとめるとします。このようなアプリケーションでは、5 つのメッセージのグループ全体をまとめて送信する、または送信しない場合は5 つのメッセージをすべて送信しない必要があります。JMS コンポーネントを使用すると、メッセージが個別に作成および送信されるアプリケーションを構築できますが、接続が途中で閉じられると(または5 つのメッセージのいずれかでエラーが発生すると)、グループ全体がロールバックされます。

**注記：** JMS では、MOM 製品が「分散」トランザクションをサポートする必要はありません。ただし、このようなサポートがある場合、JMS では、サポートが JTA (Java トランザクション API) *XAResource* を介して実装される必要があります。MOM 環境で使用できる分散トランザクションのサポート(存在する場合)については、JMS プロバイダのマニュアルを参照してください。

**注記：** 分散トランザクションは JTA を介して制御されるので、メッセージセッション *commit* または *rollback* を使用すると、JMS *TransactionInProgressException* が発生します。

## ポイントツーポイントメッセージングとは

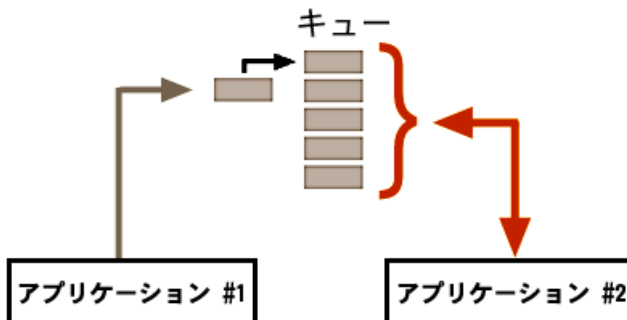
「ポイントツーポイント」(PTP) メッセージングおよび「パブリッシュ/サブスクライブ」メッセージングの2つのメインパラダイムが MOM ベンダにより実装されています。ベンダによっては、どちらか一方だけを実装していることや、両方とも実装していることもあります。

**注記：** ポイントツーポイントは、メッセージングにおける同期接続を表しているものではありません(RPCの説明などでは、表すこともあります)。

PTP モデルでは、理論上、任意の JMS クライアントが、管理上の制限のみに従い、他の JMS にメッセージを送信できます。PTP は、非同期なキューベースのピアツーピアモデルで、通常、キューが管理的に作成され、キューの寿命は無制限です。キューは、これを使用するクライアントがオンラインであってもそうでなくても、送信されたメッセージの受信および保持に使用できます。

PTP では、キューが、メールボックスのように機能します。アプリケーションで、メッセージをキューに送信したり、別のアプリケーションで、同じキューからメッセージを取り出したりすることがあります。通常、クライアントは、送信されるすべてのメッセージを1つのキューに入れます。

## ポイントツーポイント メッセージング



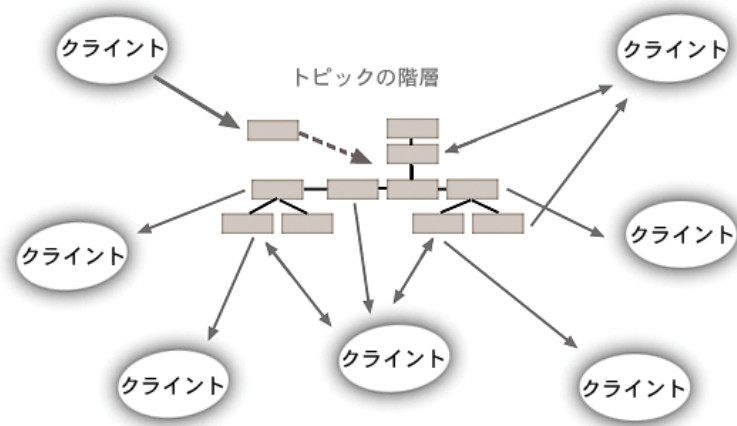
ポイントツーポイントモデルは、キューベースのピアツーピアモデルで、キューは実質的にメールボックスとして機能します。クライアントアプリケーションは、キューにメッセージを送信、(上記のアプリケーション2のように)メッセージを1つずつ取り出す、連続してメッセージキューをポーリングすることができます。また、クライアントは、受信するときにメッセージで機能する `MessageListener` を実装できます。

## パブリッシュ / サブスクライブメッセージングとは

パブリッシュ / サブスクライブメッセージングモデルは、一部 (すべてではありません) の MOM ベンダにより実装され、次のような点で、ポイントツーポイントと異なります。

- ◆ キューが通常、複数のクライアントにより共有される。
- ◆ キューが「トピック」と呼ばれるノードに階層的に編成される (これは、通常の実装計画ですが、JMS では、実際、トピックでの表現に制限はありません)。
- ◆ 各トピックが、送信されたメッセージをまとめ、分配するミニメッセージブローカーとして機能する。

## パブリッシュ/サブスクライブ



パブリッシュ/サブスクライブシステムでは、通常、キューはトピックと呼ばれるノードに階層的に編成されます。クライアントは、いくつものトピックをサブスクライブおよびパブリッシュできます。

このようなシステムのクライアントは、*TopicPublisher* および *TopicSubscriber* と呼ばれるメッセージの生成 / 消費側オブジェクトを使用します。クライアントは、複数のトピックに加入したり、加入者および発行者の両方になることができます。

*TopicSubscriber* は、「正統的」または「一時的」にすることができます。クライアントが、トピックの「すべて」のメッセージ (加入者がオフラインのときに発行されるメッセージも含む) にアクセスする場合、永続的 *TopicSubscriber* を使用する必要があります。これ以外の場合、クライアントは、メッセージ取り出しセッション中キューに保持されるメッセージにだけアクセスします。

**注記：** メッセージは、シリアルで加入者に送信されます。トピックは共有リソースなので (また一度に扱われる加入者は 1 人だけなので)、遅延の可能性は、PTP よりパブリッシュ/サブスクライブメッセージングの方が高くなります。

## 配信の保証について

JMS は、2 種類のメッセージ配信モードをサポートしています。

- ◆ **PERSISTENT** モードは、メッセージを安全な補間場所に送るようにメッセージブローカに伝えます。これにより、システム障害によりメッセージが転送中に損失することがなくなります。

- ◆ NON\_PERSISTENT では、JMS プロバイダが安定した保管場所にメッセージを保存する必要がありません。そのため、理論上、メッセージが損失することがあります (その代わりパフォーマンスは向上し、NON\_PERSISTENT メッセージではオーバーヘッドも少なくなります)。

JMS プロバイダは、NON\_PERSISTENT メッセージを「一度だけ」配信するときに必要です。つまり、メッセージが損失することはありますが、2 度配信されることはありません。

対称的に、PERSISTENT メッセージの配信は、「確実に一度だけ」行われます。つまり、プロバイダ障害により転送中のメッセージが失われることがなく、メッセージが複数回配信されることもありません。

**注記：** 一度および一度だけの配信には、メッセージの有効期限切れ、リソース超過、または管理上の削除によるメッセージの損失が保証されないという重要な制限があります。メッセージシステムで最高の信頼性を実現するには、使用している MOM ソリューションの管理問題を完全に理解する必要があります。

## メッセージ構造

企業メッセージング製品は、ヘッダ、本文、および (JMS 対応製品の場合) プロパティリストで構成される持続的で軽量のエンティティとしてメッセージを扱います。「ヘッダ」には、メッセージのルーティングと識別に使用されるフィールドが含まれます。「本文」には、送信されるアプリケーションデータが含まれます。「プロパティ」は、任意の記述子をメッセージに追加するメカニズムを提供します。そのため、クライアントまたはプロバイダは、アプリケーション固有の基準に基づき、メッセージを選択または「フィルタ」することができます。

## ヘッダ情報

ヘッダで定義されるメッセージの特徴は、「期限」(メッセージを使用できる期間を設定)、「メッセージ優先度」(0 ~ 9 の数値)、および「配信モード」(PERSISTENT または NON\_PERSISTENT) があることです。

JMS で定義されるヘッダフィールドは次のとおりです。

- ◆ JMSCorrelationID
- ◆ JMSDestination
- ◆ JMSDeliveryMode
- ◆ JMSExpiration
- ◆ JMSPriority
- ◆ JMSMessageID
- ◆ JMSTimestamp

- ◆ JMSRedelivered
- ◆ JMSReplyTo
- ◆ JMSType

これらの定義済みヘッダフィールド (すべての JMS メッセージに必要です) のほかに、JMS 定義のプロパティフィールド (ほとんどはオプションです)、プロバイダ固有のプロパティ、およびユーザプロパティがあります。

様々なヘッダおよびプロパティフィールドの意味については、付録 C で詳しく説明します。

## 本文のタイプ

JMS では、5つのメッセージ本文のタイプを定義しています。

- 1 MapMessage** — 本文のメッセージは、キーと値のペアで構成されます。ここで、キーは *Java String* で、値は *Java* プリミティブタイプです。エントリーには、シーケンシャル一覧または名前によりランダムでアクセスできます(キーと値のペアの順序は定義されません)。
- 2 TextMessage** — 本文が *java.lang.String* のメッセージ
- 3 StreamMessage** — 本文が一連の *Java* プリミティブ値 (シーケンシャルに入力および読み取られます) で構成されるメッセージ
- 4 ObjectMessage** — シリアル化が可能な *Java* オブジェクトを含むメッセージ (オブジェクトのコレクションには、JDK 1.2 で定義される「コレクションクラス」が使用されます)
- 5 BytesMessage** — 一連の連続するデータで構成されるメッセージ (このカテゴリは、ベンダ側のメッセージ形式と一致するように本文をエンコードするためのものです)

**注記:** タイプに関係なく、キューに送信されたすべての JMS メッセージは「読み込み専用」です。

JMS Connect では、5つの標準 JMS 本文タイプのいずれかを使用して、メッセージペイロードを定義できます。また、JMS Connect は、2つの定義済みメッセージタイプを提供します (実際は、2つの定義済み JMS 本文タイプのラップです)。

- ◆ **XML** — XMLドキュメントを(選択したXMLテンプレートに基いて)メッセージとして送受信できます。XMLドキュメントの完全な DOM 表現は、ECMAScript および XPath を介したマッピングまたは操作に使用できます。また、新しいノードをテンプレートドキュメントに追加できます。このメッセージタイプは、JMS 定義の *TextMessage* タイプのラップです。
- ◆ **Copybook** — COBOL コピーブックを (*BytesMessage* として) 送受信できます。

詳細については、第 4 章を参照してください。

## メッセージの取り出し

JMS では、クライアントのメッセージ取り出しに 2 種類のメカニズム (次を参照) を提供しています。

- 1 同期メッセージ取り出し。応答のないセッションの終了にタイムアウト値を指定できます。
- 2 `onMessage()` メソッドに着信メッセージ処理用のアプリケーションロジックが含まれる `MessageListener` オブジェクトを介した非同期取り出し

**注記：**ここでは、「同期」および「非同期」という用語は、「送信者」とクライアント間の関係ではなく、キューとクライアントの通信セッションを表します。送信者は、受信側クライアントがオンラインであるかどうかに関係なく、常に、キューにメッセージを送ります。この意味では、すべてのメッセージが非同期に受信されます。

同期取り出しでは、タイムアウト値をミリ秒単位で指定できます。タイムアウト値を指定しないと、「受信」セッションは、メッセージが到着するまで、永久的にブロックされます。また、「ゼロ」を指定すると、該当する選択条件 (指定した場合) を満たすキューのメッセージが取り出され、セッションはすぐにタイムアウトになります。

非同期取り出しでは、メッセージがセッションのように扱われ、メッセージの到着直後にクライアントに通知 (およびアクションを実行) できます。`onMessage()` ハンドラにより起動されるアプリケーションロジックでは、待ち時間を最小限に抑え、透過的にメッセージを処理します。この場合、ブロードキャスト / リスナメタフォが適用されます。

この例として、クライアントがキューからデータを「取り出す」同期取り出しがあります。非同期では、キューはクライアント側にデータを「送信します」。

## メッセージフィルタ

アプリケーションによっては、送受信するメッセージをフィルタまたは分類する必要があります。受信側アプリケーションによっては、メッセージの本文を簡単に検査し、その内容からメッセージを送るか、破棄するかを決定できます。しかし、メッセージを送るかどうかを定めるためにメッセージの本文を分析する必要がないように、選択条件をメッセージヘッダに指定した方が効率的です。

メッセージのヘッダ部分にメッセージ選択条件を指定することは、複数の受信側アプリケーションが同じキューを使用する場合の通常の方法です。あるメッセージタイプの扱いに最適なアプリケーションが、必要なメッセージだけを受け取りつつ、他のアプリケーションで、それらに適したメッセージを送ることができません。管理上、それぞれに専用の受信者をもつように複数のキューを設定するより、1つのキューを設定した (複数の受信者が1つのキューにアクセスする) 方が効率的です。

また、選択条件を ( ヘッダを介して ) JMS プロバイダに表示できる場合、そのプロバイダがメッセージを必要とするクライアントだけにメッセージを送信できるという利点もあります。実際、フィルタを JMS プロバイダの代理として使用できます ( この方法は、パブリッシュ / サブスクライブメッセージングで重要です )。

JMS は、キューのメッセージの選択に使用できる「メッセージセクタ」を定義します。このセクタは、ヘッダフィールドまたはプロパティ値がセクタで対応する ID として使用されるときに、true か false を評価する式です ( 構文は SQL92 と同じです )。

exteNd JMS Connect は、すべての Browse および Receive アクションに対してネイティブ環境ペインの [Message Filter] タブのセクタを実装します。これにより、選択した条件に従って、着信メッセージをフィルタできます。

詳細については、第 4 章および付録 B を参照してください。

## リクエスト - 応答と保存 / 転送

「リクエスト - 応答」の例として、応答の受信を期待してメッセージを送信するアプリケーションがあります。たとえば、信用交換アプリケーションは、顧客情報をメッセージにまとめ、そのメッセージをキューに送ります。一方、受信側アプリケーションは、そのメッセージを取り出し、必要なデータベースクエリおよびその他の処理を実行して、オリジナルメッセージに応答します。

これは、メッセージの作成側が、メッセージをキューに置くだけで終了する ( または他の処理に移る ) 「保存 / 転送」または「起動と削除」の例とは異なります。このように送信されるメッセージは、「データグラム」と呼ばれます。

JMS Connect は、どちらの場合でもサポートします。しかし、リクエスト - 応答の例は、個々の Send Message および Receive Message アクションを使用して実装する必要があります ( つまり、リクエスト - 応答セッションをカプセル化するアクションタイプはありません )。リクエストメッセージおよび応答メッセージが同じキューを共有する場合、関連する Send および Receive アクションは、同じ JMS コンポーネントで連続して実行する必要があります。しかし、発信メッセージが着信メッセージとは異なるキューに置かれる場合、キューは JMS コンポーネントごとに使用する必要があるため、2 つの異なる JMS コンポーネントを作成する必要があります。

詳細については、第 4 章を参照してください。

## JMS の定義範囲

JMS 標準は、多数のメッセージシステム動作およびデータタイプを定義しますが、管理的な概念、パフォーマンス調整、セキュリティ、構成問題、その他の様々な JMS プロバイダ機能は扱いません。

JMS では「扱われない」範囲は次のとおりです。

- ◆ 負荷バランス
- ◆ スケーラビリティ
- ◆ 透過的なフェイルオーバー
- ◆ システム側のエラー通知または計画
- ◆ ユーザ認証
- ◆ メッセージ (プライバシ) の安全な転送
- ◆ 通信プロトコル
- ◆ リポジトリに保存されるメッセージタイプ定義

これらの機能については、MOM ベンダのドキュメンテーションを参照してください。

## exteNd の JMS コンポーネントについて

JMS Connect は、exteNd Service に追加できる JMS コンポーネントを作成します。XML Map コンポーネントと同様に、JMS コンポーネントは、着信または発信メッセージと XML テンプレート間でデータをマップ、変換、および転送するために設計されています。また、JMS 対応メッセージングシステムへの JMS 呼び出しを作成、設定ウィザードを介して提供した情報に基づいて必要なヘッダ情報を自動的に入力、JMS の制限に従ってメッセージ内容のパッケージング化詳細を扱うことができます。

任意のデータ交換操作と同様、JMS コンポーネントは、接続リソースに依存しています。接続リソースは、ポート、チャンネル、ユーザ認識、パスワード、キューの位置などに関する重要な情報を指定します。JMS 接続リソースを設定すると、これを使用して、リソースに指定したキューにメッセージを送る (またはメッセージを取り出す) ように JMS コンポーネントを設定できます。



# 2

## JMS コンポーネントエディタをお使いになる前に

他の exteNd Connect と同様、使用する JMS コンポーネントの作成は、接続がメッセージキューまたはメッセージトピックから発生するように、実際には JMS 接続リソースを作成することから始まります。また、コンポーネントで使用する任意の XML テンプレートドキュメント (XML スケルトン、DTD、または XSL スタイルシート、あるいはこれらすべて) も準備します。この章では、これらの項目の準備について説明します。

### JMS 接続リソースの作成

JMS コンポーネントを使用するには、メッセージを送受信するキューまたはトピックにアクセスするための接続リソースを作成する必要があります。

JMS Connect 含む各 Connect アプリケーションでは、独自の接続リソースタイプが使用されます。JDBC、JMS、ECI など異なる接続リソースには、対象となる外部のデータソースに対応するさまざまな数やタイプのパラメータが必要です。これを反映するために、セットアップウィザードの表示が動的に変化します。

接続リソースを作成すると、JMS コンポーネントを作成するたびに新しい接続を作成する必要なく、さまざまな JMS コンポーネントに再使用できます。また、接続リソースは一度作成するとある程度自己設定が可能で、接続に関連付けられているパラメータ値を制御する ECMAScript 式にデータフィールドをリンクすることができます (次を参照)。

## 式駆動型の接続について

「Create a New Connection Resource」ウィザードを使用すると、定数または式の 2 つの方法で接続パラメータを指定できます。デフォルトでは、ウィザードのパラメータ入力フィールドは定数ベースで、接続が使用されるたびに指定された任意のパラメータに対し入力した値がそのまま使用されます。対照的に式ベースのパラメータの場合、設計時にウィザードに組み込まれている ECMAScript 式のプログラムによって、値がランタイム時に決定されます。したがって、ランタイム時の状況に応じて、式駆動型のパラメータの値は接続が使用されるたびに異なる可能性があります。

たとえば、JMS Connection における式駆動型パラメータの簡単な例として、PROJECT 変数として Connection User 名の定義があります (Composer のメインメニューバーから、[Tools]、[Configuration]、[Project Variables] タブの順に選択します)。ここで、Connection User パラメータに PROJECT 変数の値を割り当てることができます。プロジェクトを配備する際、このように Deployment Wizard の「Project Variable Remapping Panel」機能を使用して、最終的な運用環境に適した値に Connection User 名を更新します。

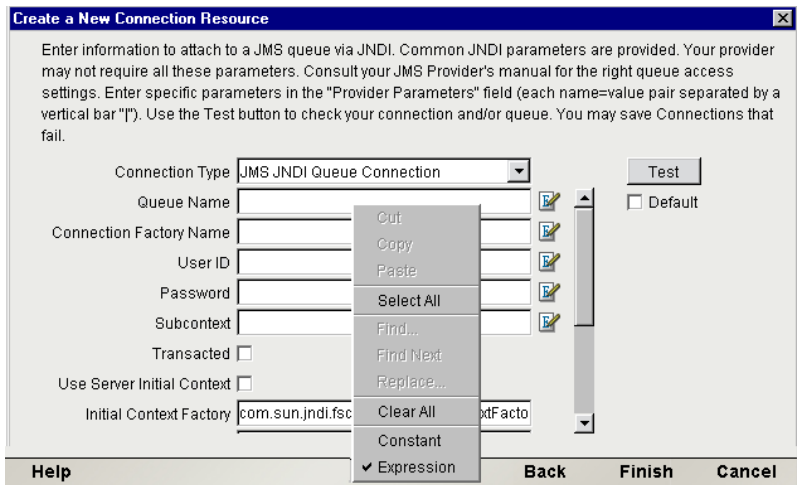
別の例として、毎月 15 日にメンテナンスを行うために MOM (メッセージ指向型ミドルウェア) 環境で Queue1 が設定されていたとすると、代わりに Queue2 が使用されます。接続の Queue Name パラメータに次の式を割り当てた場合です。

```
(new Date()).getDate() == 15 ? "Queue1" : "Queue2"
```

また、ECMAScript 式を使用して、ディスク上のファイルから情報を読み込み、アプリケーションサーバなどに存在する Java オブジェクトを呼び出すことができます。このように、パラメータの情報提供に式を使用すると、柔軟性および機能が向上します。

### ➤ パラメータを式駆動型モードに切り替える

- 1 式を添付するフィールドにカーソルを置きます (この操作は、[Connection Type] フィールド、チェックボックスにはいずれも適用されません)。
- 2 マウスの右ボタンをクリックして、コンテキストメニューを表示します。



- 3 メニューから **[Expression]** を選択します。パラメータフィールドの右側に、青色の **[Expression Editor]** アイコンが表示されます。
- 4 フィールドに ECMAScript 式を入力するか、**[Expression Editor]** ボタンをクリックして Expression Editor のリストを使用し、ランタイム時に正しいパラメータ値を評価する式を作成します。

## キュー接続について

メッセージ指向ミドルウェアのシステムでは、キューは管理されたリソースで、JMS はそれぞれの管理オブジェクトからキューにアクセスします。ポータブルなままインタフェースからクライアントアプリケーションでオブジェクトを使用できるように、JMS 管理オブジェクトにより送信先および接続に関する情報がまとめられます。

JMS では、管理オブジェクト ( 接続ファクトリおよび送信先 ) を JNDI ネームスペースに配置する必要があります。したがって、アプリケーションにより要求される接続リソースは常に JNDI から取得されます。ただし、プロバイダの接続ファクトリのオブジェクト名が対象の Java アプリケーションですでに認識されている場合は、JNDI を経由することなくそのアプリケーションで接続が作成されます。

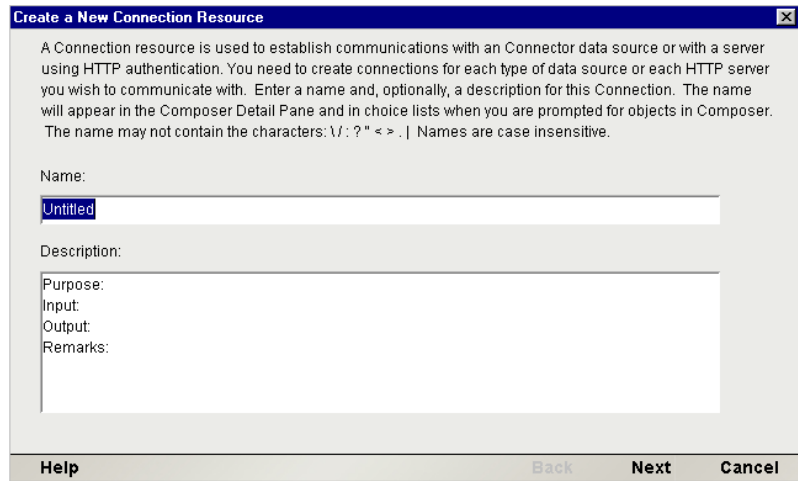
exteNd では、デフォルトで JNDI を経由する接続が設定されています ( JNDI を利用する管理オブジェクトへのアクセスは、すべての JMS MOM で使用できることが保証されているためです )。ただし、IBM の MQSeries の場合、MQSeries クラスを使用して ( つまり JNDI を経由せずに ) キュー接続を直接取得するオプションがあります。ユーザは、ベンダにより設定されたユーザインタフェースを利用して簡単なセットアップオプションを選択できます。

**注記：** ベンダ固有の「直接接続」機能が使用可能な場合、exteNd Composer の「Create a New Connection Resource」ウィザードにあるプルダウンメニューにリストが表示されます。また、xconfig.xml ファイルの JMS <COMPONENT\_FACTORY> ノードで <PROVIDERS> の内容を確認すると、プロバイダ固有の接続機能を検証できます (xconfig.xml ファイルは Composer/bin ディレクトリにあります)。

➤ **JNDI を使用して JMS キュー接続リソースを作成する**

**注記：** 次のグラフィックに示されている設定は、JbrokerMQ への一般的な JNDI 接続です。

- 1 [File]、[New xObject]、[Resource]、[Connection] の順に選択します。「Create a New Connection Resource」ウィザードが表示されます。



The screenshot shows a dialog box titled "Create a New Connection Resource". The dialog contains the following text: "A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \/: ? " < > . | Names are case insensitive." Below this text, there is a "Name:" label followed by a text input field containing "Untitled". Below that is a "Description:" label followed by a larger text area containing "Purpose:", "Input:", "Output:", and "Remarks:". At the bottom of the dialog, there are four buttons: "Help", "Back", "Next", and "Cancel".

- 2 [Name] に、接続オブジェクトの名前を入力します。
- 3 オプションで、[Description] に説明のテキストを入力します。
- 4 [Next] をクリックします。

Enter information to attach to a JMS queue via JNDI. Common JNDI parameters are provided. Your provider may not require all these parameters. Consult your JMS Provider's manual for the right queue access settings. Enter specific parameters in the "Provider Parameters" field (each name=value pair separated by a vertical bar "|"). Use the Test button to check your connection and/or queue. You may save Connections that fail.

Connection Type: JMS JNDI Queue Connection

Queue Name: \_\_\_\_\_

Connection Factory Name: \_\_\_\_\_

User ID: \_\_\_\_\_

Password: \_\_\_\_\_

Subcontext: \_\_\_\_\_

Transacted:

Use Server Initial Context:

Initial Context Factory: com.sun.jndi.fscontext.ReffSContextFacto

Test

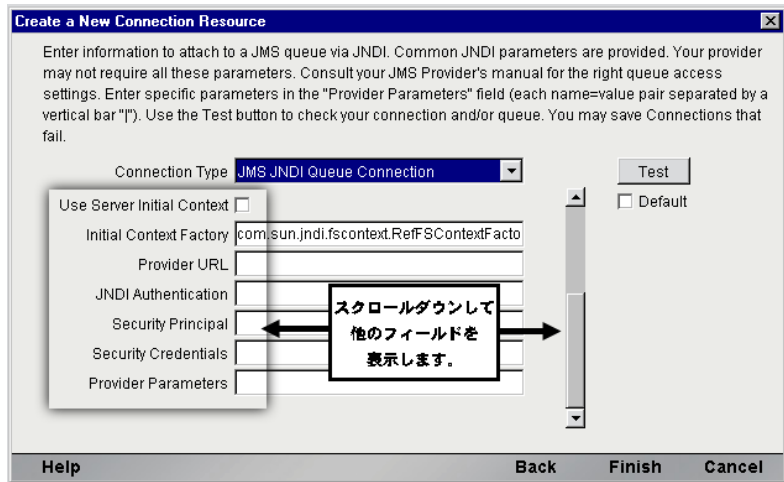
Default

Help Back Finish Cancel

- 5 [Connection Type] プルダウンメニューから、**JMS JNDI Queue Connection** (ポイントツーポイントメッセージング用) を選択します。ペインの内容が更新され、選択した特定の接続タイプに必要なセットアップ情報が表示されます。
- 6 最初のフィールド ([Queue Name]) に、使用するキューの名前を入力します。
- 7 必要に応じて、[ClientID] を入力します (オプション)。
- 8 [Connection Factory Name] フィールドに、キュー接続を作成する場合はキュー接続ファクトリ名、トピック接続を作成する場合はトピック接続ファクトリ名を入力します。
- 9 必要に応じて、接続ユーザおよび接続パスワードの情報を入力します (オプション)。
- 10 必要に応じて、[Subcontext] に JNDI のサブコンテキスト情報を入力します (オプション)。
- 11 JMS コンポーネントにおいて、セッションレベルで Commit または Rollback コマンドを発行する場合は [Transacted] チェックボックスをオンにします。

**注記:** [Transacted] チェックボックスをチェックせずに JMS コンポーネントのアクションモデルで Commit または Rollback ステートメントを発行すると、例外が発生します。

- 12** サービスの配備後、サーバでランタイム時にローカルで接続ファクトリを取得する場合、[Use Server Initial Context] にチェックを付けます。つまり、配備されたサービスでキューまたはトピック接続を取得する場合、手順 15 から 19 までを実行する必要はありません。ただし、設計時に接続でライブメッセージを送受信する場合は、リモートホストで **Composer** に必要な接続ファクトリオブジェクトが検索可能であることが必要なため、次の該当する手順をすべて完了する必要があります。次の設定は、**Composer** によりリモートで接続を構築する場合を想定しています。
- 13** テキストフィールドの右側にある**垂直方向のスクロールバー**を下にスクロールして、**ダイアログボックスの残りのフィールド**を表示します。次の図を参照してください。



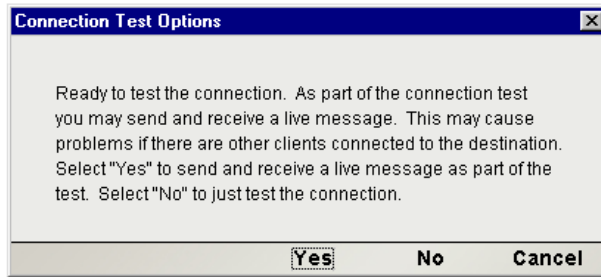
- 14** [Initial Context Factory] フィールドに、**com.sun.jndi.fscontext.RefFSContextFactory** など、使用しているシステムの JNDI コンテキストファクトリの名前を入力します (この情報を入手するには、管理者に連絡してください)。
- 15** [Provider URI] フィールドに、JMS プロバイダ (または MOM プロバイダ) の JNDI コンテキストリソースの場所を表す URI を入力します。たとえば、このフィールドは **iiop://localhost:3506** または **file:///D:/MQSeries/java/fscontext** のようになります。
- 16** (オプション) [JNDI Authentication] フィールドに、必要な JNDI 認証文字列を (管理者の指定どおりに) 入力します。
- 17** (オプション) [Security Principal] フィールドに、必要な JNDI セキュリティプリンシパル名を (管理者の指定どおりに) 入力します。
- 18** (オプション) [Security Credentials] フィールドに、必要な JNDI セキュリティ資格情報文字列を (管理者の指定どおりに) 入力します。

- 19 ( オプション ) [Provider Parameters] フィールドに、操作を実行している MOM 環境に必要なプロバイダ固有の name/value の対を入力します。パイプ文字 (|) で name/value の対を区切ります。たとえば、LDAP プロバイダのパラメータは次のようになります。

```
java.naming.security.authentication = value |
java.naming.security.credentials = value |
java.naming.security.principal = value
```

**注記：**ここでは、わかりやすく表示するためにスペースが使用されています。プロバイダパラメータの文字列にはスペースを使用しないでください。

- 20 JMS コンポーネントの作成時に、デフォルトでこの接続リソースをセットアップダイアログボックスに表示する場合は、[Default] チェックボックスをオンにします (通常チェックはオフです)。
- 21 正常に接続されるかどうかを確認するため、[Test] をクリックします。[Test Options] ダイアログボックスが表示されます。



- 22 [Test Options] ダイアログボックスに、接続の整合性に関するテストの一部としてライブメッセージを送信するかどうか確認するメッセージが表示されます。[Yes] ボタンをクリックすると、Composer により接続を構築しているキューまたはトピックにライブメッセージ (固有な CorrelationID を持つ *TextMessage* タイプ) が送信されます。

**注記：**運用環境において既存のアプリケーションに悪影響が及ばないという合理的な確信がない限り、運用環境 (多数のリソースが存在する可能性があり、ライブキューを使用している環境) でこのテストメッセージを送信しないように注意してください。

必要な接続オブジェクトを作成するもののテストメッセージを送信しない場合は、[No] をクリックします。

- 23 [Finish] をクリックします。新しく作成された接続リソースの xObject が、Composer の接続リソースの詳細ペインに表示されます。

#### ➤ MQSeries のキュー接続リソースを作成する

- 1 [File]、[New xObject]、[Resource]、[Connection] の順に選択します。「Create a New Connection Resource」ウィザードが表示されます。

**Create a New Connection Resource**

A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \/: ?" < > . | Names are case insensitive.

Name:

Description:

Purpose:  
Input:  
Output:  
Remarks:

**Help** **Back** **Next** **Cancel**

- 2 接続オブジェクトの名前を入力します。
- 3 オプションで、[Description] に説明のテキストを入力します。
- 4 [Next] をクリックして、ウィザードの接続情報ペインに移動します。

**Create a New Connection Resource**

Enter the information required to attach to an MQSeries topic via JMS Bridge. All the specific MQSeries parameters are provided as they are defined. You will need to enter the Topic name and host name of the queue manager. For all other parameters refer to the user documentation. Checking 'Default' makes this Connection the initial selection when requesting a JMS MQSeries Topic Connection. Use the Test button to check your connection. You may save connections that fail the test.

Connection Type: **JMS MQSeries Queue Connection**

Queue Name:

User ID:

Password:

Host Name:

Port:

Queue Manager:

Channel:

Temporary Model Queue:

Default

**Help** **Back** **Finish** **Cancel**

- 5 [Connection Type] プルダウンメニューから、**JMS MQSeries Queue** (ポイントツーポイントメッセージング用) を選択します。ペインの内容が更新され、選択した特定の接続タイプに必要なセットアップ情報が表示されます。
- 6 最初のフィールド ([Queue Name]) に、使用するキューの名前を入力します。
- 7 オプションで、[Connection User] フィールドにユーザ名を入力します。

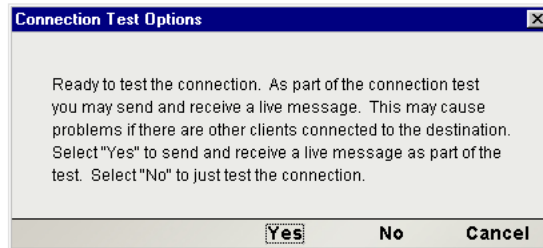


- 8 オプションで、[**Connection Password**] フィールドにパスワードを入力します。
- 9 [**Host Name**] フィールドに、使用しているシステムのMQSeries ホストマシン名を入力します ( 必要に応じて、この情報については管理者に連絡してください)。
- 10 [**Port**] フィールドに、MQSeries ホストマシンのポート番号を入力します ( 必要に応じて、この情報については管理者に連絡してください)。
- 11 [**Queue Manager**] フィールドに、MQSeries キューマネージャ名を ( 管理者の指定どおりに ) 入力します。
- 12 [**Channel**] フィールドに、MQSeries ホストマシンのチャンネル名を ( 管理者の指定どおりに ) 入力します。
- 13 一時的なモデルのキューを指定する場合は、[**Temporary Model Queue**] フィールドに指定します。
- 14 垂直方向のスクロールバーを下にスクロールして、ダイアログボックスの残りのフィールドを表示します ( この例では 2 つのチェックボックスが表示されています )。次の図を参照してください。

- 15 JMS コンポーネントにおいて、セッションレベルで Commit または Rollback コマンドを発行する場合は [**Transacted**] チェックボックスをオンにします ( デフォルトではオフです )。

**注記：** [Transacted] チェックボックスをチェックせずに JMS コンポーネントのアクションモデルで Commit または Rollback ステートメントを発行すると、例外が発生します。

- 16 MOM 本来の機能を使用して最適な接続を行う場合は、[Non-JMS Client] チェックボックスをオンにします (この例では、ベンダ製の呼び出しを使用して MQSeries のオブジェクトを直接取得するということです)。
- 17 JMS コンポーネントの作成時に、デフォルトでこの接続リソースをセットアップダイアログボックスに表示する場合は、[Default] チェックボックスをオンにします (通常チェックはオフです)。
- 18 正常に接続されるかどうかを確認するために、[Test] をクリックします。[Test Options] ダイアログボックスが表示されます。



- 19 [Test Options] ダイアログボックスに、接続の整合性に関するテストの一部としてライブメッセージを送信するかどうか確認するメッセージが表示されます。[Yes] ボタンをクリックすると、Composer により接続を構築しているキューまたはトピックにライブメッセージ(固有な CorrelationID を持つ *TextMessage* タイプ)が送信されます。

**注記:** 運用環境において既存のアプリケーションに悪影響が及ばないという合理的な確信がない限り、運用環境 (多数のリスナが存在する可能性があり、ライブキューを使用している環境) でこのテストメッセージを送信しないように注意してください。

必要な接続オブジェクトを作成するもののテストメッセージを送信しない場合は、[No] をクリックします。

- 20 [Finish] をクリックします。新しく作成された接続リソースの *xObject* が、Composer の接続リソースの詳細ペインに表示されます。

## トピック接続について

キューがパブリッシュ / サブスクライブコンテキストに使用される場合 (「パブリッシュ / サブスクライブメッセージングとは」を参照)、「トピック」と呼ばれます。キューとトピックの違いは、基本的に機能面よりも管理面にあります。したがって、パブリッシュ / サブスクライブコンテキストで、ブラウザ表示が定義されない点を除き、前のセクションにあるキュー接続に関するコメントはすべてトピック接続にも同様に適用されます。Browse アクションを使用する必要がある場合は、トピックではなく、キューに接続してください。

## ➤ JNDI トピック接続リソースを作成する

- 1 [File] > [New xObject] > [Resource] > [Connection] の順に選択します。「Create a New Connection Resource」ウィザードが表示されます。
- 2 すでに説明した「JNDI を使用して JMS キュー接続リソースを作成する」(28 ページ)の**手順 2、3、および 4**に従って、ウィザードの最初のパネルを入力します。
- 3 ウィザードの最初のパネルで、[Finish] をクリックします。新しいパネルが表示されます。

Enter information required to attach to a JMS topic via JNDI. Common JNDI parameters are provided. Your provider may not require all the listed parameters. Consult your JMS Provider's manual for the right setting to access your queue. Enter provider specific parameters in the "Provider Parameters" field (each name=value pair separated by a vertical bar "|"). Select transacted to have session level control of transactions. Check "Default" makes this Connection the initial selection when requesting a JMS JNDI Topic Connection.

Connection Type: JMS JNDI Topic Connection

Topic Name: \_\_\_\_\_

Durable Subscriber: \_\_\_\_\_

Client ID: \_\_\_\_\_

Connection Factory Name: \_\_\_\_\_

User ID: \_\_\_\_\_

Password: \_\_\_\_\_

Subcontext: \_\_\_\_\_

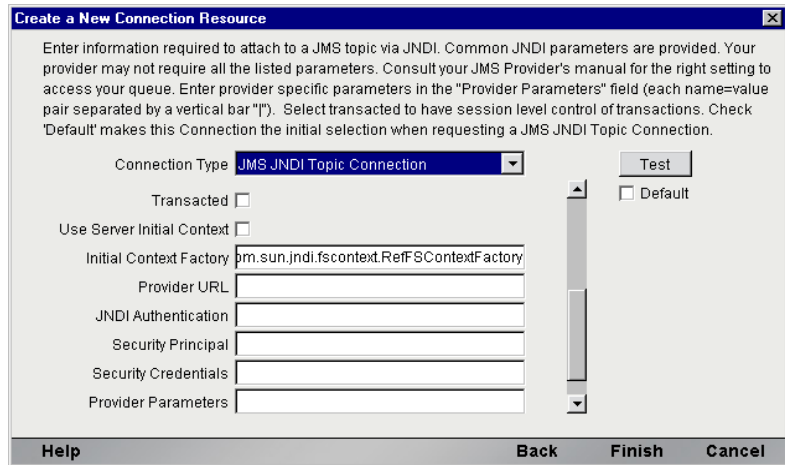
No Local Messages:

Test  Default

Help Back Finish Cancel

- 4 [Connection Type] プルダウンメニューから、**JMS JNDI Topic Connection** (パブリッシュ / サブスクライブメッセージング用) を選択します。ペインの内容が更新され、この接続タイプに必要なセットアップ情報が表示されます。
- 5 最初のフィールド ( [Topic Name] ) に、トピックの名前を入力します。
- 6 [Durable Subscriber] に名前を入力します (オプション)。
- 7 [Client ID] を入力します。
- 8 [Connection Factory Name] フィールドに、トピック接続ファクトリ名を入力します。
- 9 必要に応じて、接続ユーザおよび接続パスワードの情報を入力します (オプション)。
- 10 必要に応じて、[Subcontext] に JNDI のサブコンテキスト情報を入力します (オプション)。

- 11 リスンしているトピックに対しコンポーネントまたはサービスにより送信されるメッセージが、コンポーネントまたはサービスで受信されないようにする場合は、**[No Local Messages]** チェックボックスをオンにします（つまり、コンポーネントが自身のメッセージを受信しないようにするには、このチェックボックスをオンにします）。
- 12 下にスクロールして、パネルの残りのフィールドを表示します。次の図を参照してください。



- 13 JMS コンポーネントにおいて、セッションレベルで Commit または Rollback コマンドを発行する場合は **[Transacted]** チェックボックスをオンにします（デフォルトではオフです）。

**注記：** **[Transacted]** チェックボックスをチェックせずに JMS コンポーネントのアクションモデルで Commit または Rollback ステートメントを発行すると、例外が発生します。

- 14 サービスの配備後、サーバでランタイム時にローカルで接続ファクトリを取得する場合、**[Use Server Initial Context]** にチェックを付けます。つまり、配備されたサービスでキューまたはトピック接続を取得する場合、手順 15 から 19 までを実行する必要はありません。ただし、設計時に接続でライブメッセージを送受信する場合は、リモートホストで **Composer** に必要な接続ファクトリオブジェクトが検索可能であることが必要なため、次の該当する手順をすべて完了する必要があります。次の設定は、Composer によりリモートで接続を構築する場合を想定しています。

- 15 **[Initial Context Factory]** フィールドに、`com.sun.jndi.fscontext.ReffSContextFactory` など、使用しているシステムの JNDI コンテキストファクトリの名前を入力します（この情報入手するには、管理者に連絡してください）。

- 16 [Provider URI]フィールドに、JMS プロバイダ (または MOM プロバイダ) の JNDI コンテキストリソースの場所を表す URI を入力します。たとえば、このフィールドは `iiop://localhost:3506` または `file:///D:/MQSeries/java/fscontext` のようになります。
- 17 (オプション) [JNDI Authentication] フィールドに、必要な JNDI 認証文字列を (管理者の指定どおりに) 入力します。
- 18 (オプション) [Security Principal] フィールドに、必要な JNDI セキュリティプリンシパル名を (管理者の指定どおりに) 入力します。
- 19 (オプション) [Security Credentials] フィールドに、必要な JNDI セキュリティ資格情報文字列を (管理者の指定どおりに) 入力します。
- 20 (オプション) [Provider Parameters] フィールドに、操作を実行している MOM 環境に必要なプロバイダ固有の name/value の対を入力します。パイプ文字 (|) で name/value の対を区切ります。たとえば、LDAP プロバイダのパラメータは次のようになります。

```
java.naming.security.authentication = value |  
java.naming.security.credentials = value |  
java.naming.security.principal = value
```

**注記：**ここでは、わかりやすく表示するためにスペースが使用されています。プロバイダパラメータの文字列にはスペースを使用しないでください。

- 21 JMS コンポーネントの作成時に、デフォルトでこの接続リソースをセットアップダイアログボックスに表示する場合は、[Default] チェックボックスをオンにします (通常チェックはオフです)。
- 22 オプションで、[Test] ボタンを押して接続をテストします。
- 23 [Finish] をクリックしてウィザードを終了します。

#### ➤ MQSeries トピック接続リソースを作成する

- 1 [File]、[New xObject]、[Resource]、[Connection] の順に選択します。「Create a New Connection Resource」ウィザードが表示されます。
- 2 すでに説明した「JNDI を使用して JMS キュー接続リソースを作成する」の手順 2、3、および 4 に従って、ウィザードの最初のパネルを入力します。
- 3 ウィザードの最初のパネルで、[Finish] をクリックします。新しいパネルが表示されます。

**Create a New Connection Resource**

Enter the information required to attach to an MQSeries queue via JMS Bridge. All the specific MQSeries parameters are provided as they are defined. You will need to enter the Queue name and host name of the queue manager. For all other parameters refer to the user documentation. Checking 'Default' makes this Connection the initial selection when requesting a JMS MQSeries Topic Connection. Use the Test button to check your connection. You may save connections that fail the test.

Connection Type: **JMS MQSeries Topic Connection** [Test]  Default

Topic Name

Durable Subscriber

Client ID

User ID

Password

Host Name

Port

Queue Manager

Help Back Finish Cancel

- 4 最初のフィールド ( [Topic Name] ) に、トピックの名前を入力します。
- 5 [Durable Subscriber] に名前を入力します ( オプション )。
- 6 [Client ID] を入力します ( オプション )。
- 7 必要に応じて、**ユーザ ID** および**パスワード** の情報を入力します ( オプション )。
- 8 [Host Name] を入力します ( オプション )。
- 9 [Port] を入力します ( オプション )。
- 10 MQSeries のキューマネージャ名を指定する場合、[Queue Manager] に入力します ( キューマネージャの関連付けを行う場合、行わない場合については、MQSeries のドキュメンテーションを参照してください )。
- 11 下にスクロールして、ウィザードの残りのパネルを表示します。次の図を参照してください。

Enter the information required to attach to an MQSeries queue via JMS Bridge. All the specific MQSeries parameters are provided as they are defined. You will need to enter the Queue name and host name of the queue manager. For all other parameters refer to the user documentation. Checking 'Default' makes this Connection the initial selection when requesting a JMS MQSeries Topic Connection. Use the Test button to check your connection. You may save connections that fail the test.

Connection Type: JMS MQSeries Topic Connection

Host Name: \_\_\_\_\_

Port: \_\_\_\_\_

Queue Manager: \_\_\_\_\_

Channel: \_\_\_\_\_

IBM CCSID: \_\_\_\_\_

No Local Messages

Transacted

Non-JMS client

Test  Default

Help Back Finish Cancel

- 12** オプションで [Channel] を入力します。
- 13** リスしているトピックに対しコンポーネントまたはサービスにより送信されるメッセージが、コンポーネントまたはサービスで受信されないようにする場合は、[No Local Messages] チェックボックスをオンにします (つまり、コンポーネントが自身のメッセージを受信しないようにするには、このチェックボックスをオンにします)。
- 14** JMS コンポーネントにおいて、セッションレベルで Commit または Rollback コマンドを発行する場合は [Transacted] チェックボックスをオンにします (デフォルトではオフです)。
- 注記:** [Transacted] チェックボックスをチェックせずに JMS コンポーネントのアクションモデルで Commit または Rollback ステートメントを発行すると、例外が発生します。
- 15** このトピックについてプロバイダ本来のメッセージ配信機能を使用する (JMS ヘッダ情報を無視する) 場合は、[Non-JMS Client] チェックボックスをオンにします。コンポーネントにより MQSeries 本来の (非 JMS 対応の) メッセージングを使用する受信者にメッセージを送信する場合、このオプションが便利です。
- 16** JMS コンポーネントの作成時に、デフォルトでこの接続リソースをセットアップダイアログボックスに表示する場合は、[Default] チェックボックスをオンにします (通常チェックはオフです)。
- 17** オプションで、[Test] ボタンを押して接続をテストします。
- 18** [Finish] をクリックしてウィザードを終了します。

## コンポーネントに対する XML テンプレートの作成

接続リソースに加えて、JMS コンポーネントで XML スタブドキュメント、関連付けられている DTD または XML スタイルシート、あるいはこれらすべてを使用して、メッセージ情報のマップに役立てることもできます。このようなドキュメントを使用する場合、コンポーネントの設計のためサンプルドキュメントに利用できるこの時点で XML テンプレートリソースにこれらを追加する必要があります ( 詳細については、『exteNd Composer ユーザガイド』の第 5 章の「XML テンプレートの作成」を参照してください )。

また、コンポーネント設計によって別の xObject リソース ( カスタムスクリプトやコードテーブルマップなど ) が要求される場合は、JMS コンポーネントを作成する前にこれらのリソースを作成するようお勧めします。詳細については、『Composer ユーザガイド』の「カスタムスクリプトの作成」を参照してください。



# 3

## JMS コンポーネントの作成

この章では、exteNd Web サービスで使用するための JMS コンポーネントの作成プロセスについて説明します。また、メッセージヘッダフィールドやプロパティのセマンティックおよび使用についても、JMS メッセージングのコンテキストで説明し、JMS ベースの exteNd Web サービスを最も有効に使用する方法のヒントも記載されています。JMS Connect で作成されたコンポーネントに依存する Web サービスを作成して配備する前に、この章を読んで内容を確認してください。

### JMS コンポーネントを作成する前に

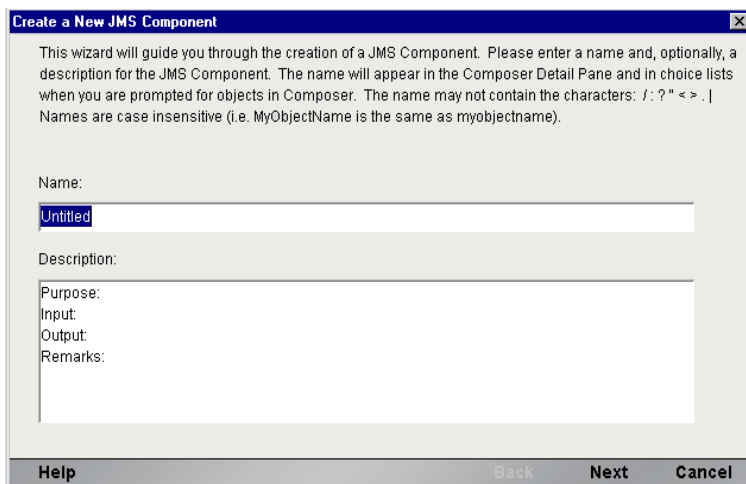
JMS コンポーネントを作成する場合は、次の質問に対する回答を把握している必要があります。

- ◆ メッセージへまたはメッセージからデータをマップするために必要な XML テンプレートドキュメント ( または COBOL コピーブック、あるいはその両方 )。XML テンプレートリソースの詳細については、『*Composer ユーザガイド*』の「*Creating a New XML Template*」を参照してください。
- ◆ コンポーネントで使用する JMS 接続リソース。JMS コンポーネントの作成プロセスの一部として、既存の JMS 接続を選択したり、新しい接続を作成したりすることが可能です ( 接続は、事前に作成した場合、新しい JMS コンポーネントすべてに対して使用可能になります )。JMS 接続リソースのセットアップ方法の段階的な情報については、前の章を参照してください。
- ◆ Browse Messages アクションを作成するかどうか ( 「Browse Messages アクション」を参照 )。作成する場合は、接続リソースとして「キュー」接続を選択する必要があります。参照は、「トピック」接続で定義されません。
- ◆ キューまたはトピックに到着したメッセージによってサービスがトリガされるかどうか。トリガされる場合は、サービスを JMS サービスとして配備する必要があります ( 第 6 章「JMS サービス」を参照 )。

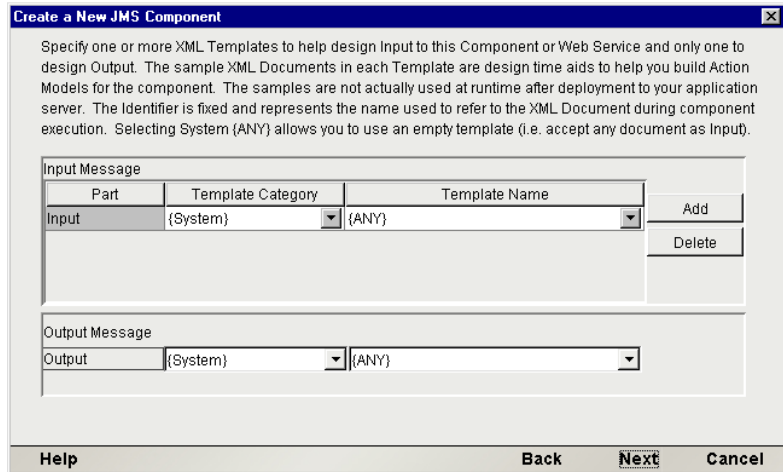
- ◆ メッセージセッションが処理されるかどうか。Commit または Rollback コマンドを発行する場合は、使用する特定の JMS 接続リソースのトランザクションを有効にする (Connection Resource セットアップウィザードで [Transacted] チェックボックスをオンにする) 必要があります。詳細については、31 ページ [Transacted] チェックボックスの説明を参照してください。
- ◆ メッセージを「送信」するのか、または「受信」するのかどうか。単一の JMS コンポーネント内では、同じキュー (または接続リソース) を使用する場合にはのみ、メッセージを送信および受信できます。2 つ以上の異なるキューまたはトピックに対して送信または受信を行う場合は、各接続リソースに 1 つずつ、別のコンポーネントを作成する必要があります。

### ➤ 新しい JMS コンポーネントを作成する

- 1 [File] > [New xObject] > [Component] > [JMS] の順に選択します。「Create a New JMS Component」ウィザードが表示されます。

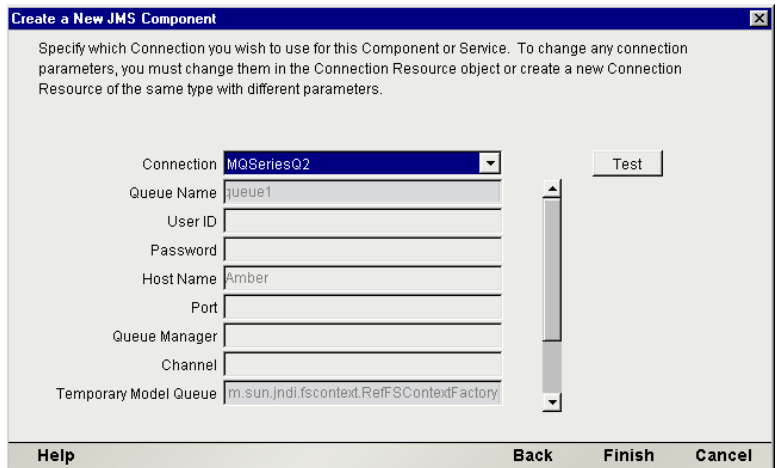


- 2 新しい JMS コンポーネントの「名前」を入力します。
- 3 オプションとして、[Description] に説明テキストを入力します。
- 4 [Next] をクリックします。「Create a New JMS Component」ウィザードの [XML Templates Info] パネルが表示されます。

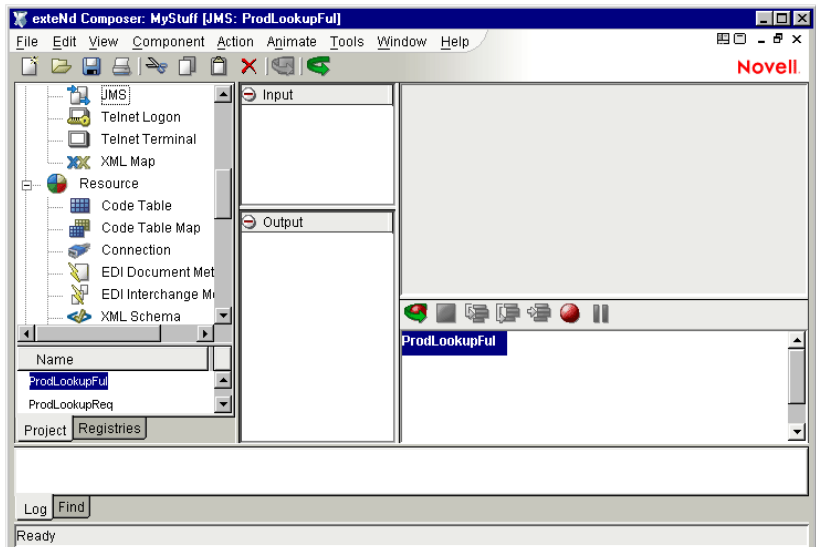


- 5 1つまたは複数の「入力」テンプレートを、次のように指定します。
  - ◆ デフォルトのカテゴリと異なる場合は、[**Template Category**] を選択します。
  - ◆ 選択した [**Template Category**] にある XML テンプレートのリストから [**Template Name**] を選択します。
  - ◆ 入力 XML テンプレートをさらに追加するには、[**Add**] をクリックして、手順 2 から 4 を繰り返します。
  - ◆ 入力 XML テンプレートを削除するには、エントリを選択して [**Delete**] をクリックします。
- 6 **Output** に対して XML テンプレートを選択します (Output という名前の出力 DOM は 1 つしかありません)。

**注記：** 出力テンプレートとして **{System}{ANY}** を選択すると、設定済みの構造が含まれない出力 XML テンプレートを指定できます。たとえば、ECMAScript を使用してランタイム時にカスタム出力 DOM を生成する場合などに、このようなテンプレートを指定します ( 詳細については、『*Composer ユーザガイド*』の XML Map コンポーネントの作成に関する章である「Creating an Output DOM without Using a Template」を参照してください)。
- 7 [**Next**] をクリックします。「Create a New JMS Component」ウィザードの [Connection Info] パネルが表示されます。



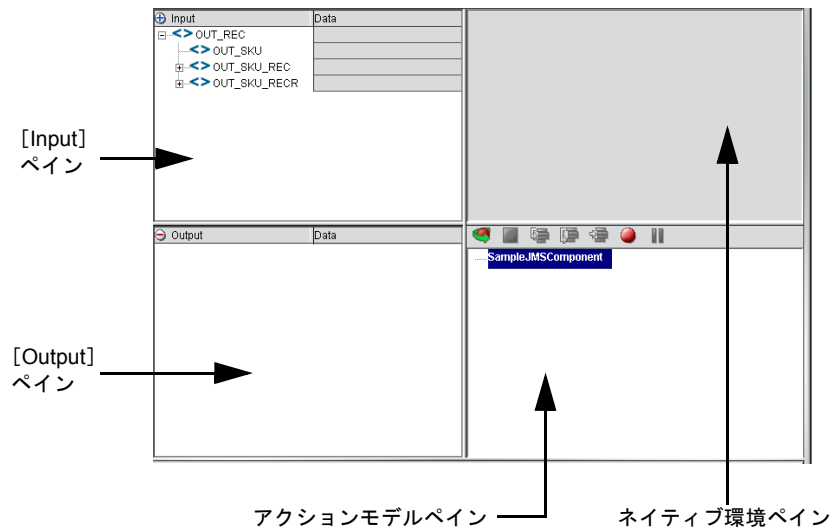
- 8 プルダウンリストで「接続」タイプを選択します。プルダウンリストの選択肢では、以前に作成した使用可能な JMS 接続リソースが反映されます。JMS 接続リソースの作成の詳細については、25 ページの「JMS 接続リソースの作成」を参照してください。
- 9 [Finish] をクリックします。コンポーネントが作成され、JMS コンポーネントエディタが表示されます。



# JMS コンポーネントエディタウィンドウについて

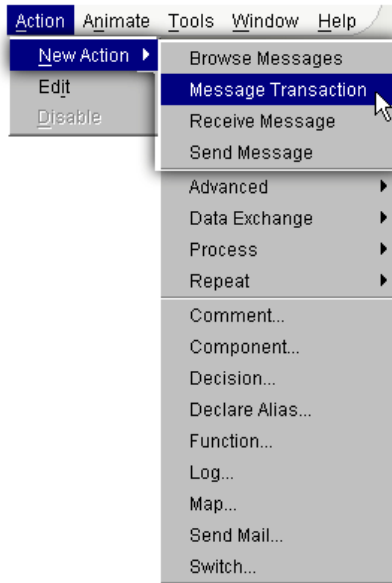
JMS コンポーネントエディタは、その外観が、XML Map コンポーネントエディタウィンドウに類似しています (XML Map コンポーネントエディタの機能がすべて含まれているだけでなく、メッセージングに固有な他のアクションタイプも含まれています)。他のあらゆるコンポーネントエディタと同様、JMS コンポーネントエディタウィンドウには、アクションモデルペイン (通常は右下角にあります) が、他の場所にあってもかまいません)、ネイティブ環境ペイン (右上)、および入力 DOM/ 出力 DOM/ 一時 DOM 用のマッピングペインがあります。

ネイティブ環境ペインは、メッセージアクションを作成または選択するまで、グレーのペインとして表示されます。メッセージアクションを作成または選択すると、現在のメッセージアクションに関連しているもの (参照、送信、または受信) に基づいて、ネイティブ環境ペインには、2 つまたは 3 つのタブを含むメッセージステータスペインが表示されます。



[Action] メニューをアクティブにして (または、アクションモデルペイン内でマウスを右クリックして) [New Action] を選択すると、XML Map コンポーネントで使用可能なアクションと同じものがすべて JMS コンポーネントでも使用可能になりますが、次の 4 つの新しいアクションタイプがさらに使用可能になります。

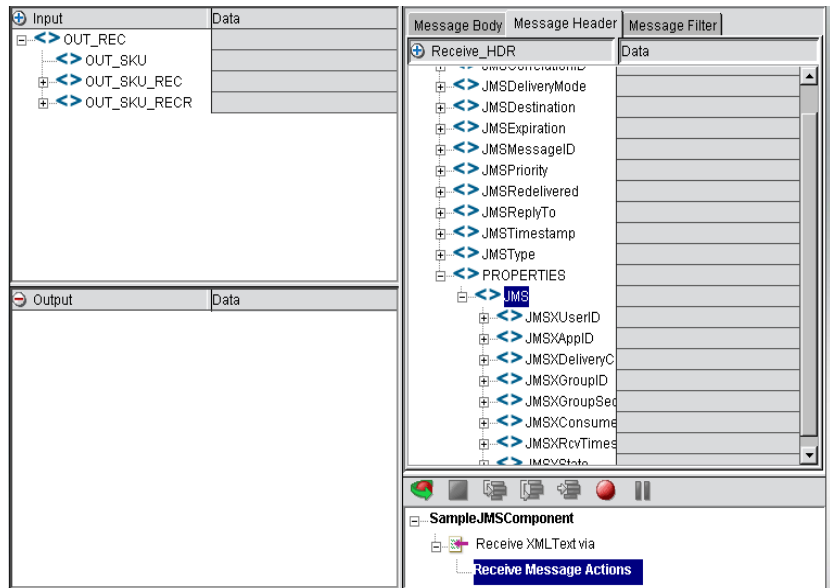
- ◆ Browse Messages
- ◆ Message Transaction
- ◆ Receive Message
- ◆ Send Message



これら4つのJMS固有のアクションについては、次の章で説明します。

## ネイティブ環境ペインについて

JMS コンポーネントエディタのネイティブ環境ペイン（最初はグレーです）には、JMS メッセージアクションがアクションモデルペインで選択されていると、メッセージに関連付けられているさまざまなタイプの情報が表示されます。使用可能な情報カテゴリ（ペインの上部にあるタブによって示される）には、[Message Body]、[Message Header]、および [Message Filter] があります。最初の2つのカテゴリは、すべてのメッセージアクション (Send、Receive、および Browse) に共通しています。ただし、[Message Filter] カテゴリは、Receive と Browse に対してのみ表示されます。



前の図では、[Message Header] タブが選択されており、アクションモデルペインでは、Receive Message アクションが選択されています。また、ネイティブ環境ペインは、使用可能なメッセージヘッダとプロパティをすべて表示するために拡大されています。メッセージヘッダおよびプロパティフィールドの使用の詳細については、127 ページ「メッセージヘッダおよびプロパティ」を参照してください。

[Message Body] タブを選択すると、ネイティブ環境ペインには、本文タイプに適した方法でコンテンツが表示されます。たとえば、メッセージに XML ドキュメントが含まれている場合、ネイティブ環境ペインには DOM ツリーが表示されます。また、メッセージに COBOL コピーブックが含まれている場合は、コピーブックのコンテンツが表示されます。詳細については、(80 ページから始まる)「メッセージヘッダへのデータのマップ」を参照してください。

[Message Filter] タブを選択すると、ネイティブ環境ペインにはセレクト編集領域が表示されます。詳細については、96 ページ「メッセージフィルタ (セクタ) の操作」を参照してください。





# 4

## JMS アクションの作成

### アクションについて

「アクション」は、プログラミングステートメントに類似しており、通常はパラメータ形式の入力により特定のタスクを実行します。一般的には関連するアクションが連係して、機能的な単位を構成します。exteNd ではこの機能的な単位をコンポーネントと呼び、コンポーネントを構成するアクションはアクションリストまたはアクションモデルの一部です (『Composer ユーザガイド』のアクションに関連する章を参照してください)。

JMS コンポーネントエディタを使用すると、送信、受信、メッセージの参照などのアクションを (オプションで「トランザクション」の一部として) 作成できます。exteNd Composer の強力な XML マップ機能により、メッセージと DOM の間で簡単に XML 情報をマップでき、ビジネス論理によるデータの転送も可能です。したがって、exteNd Composer で作成された JMS コンポーネントで XML 統合アプリケーションに優れたメッセージング機能を実現できます。

アクションモデルは、コンポーネント内で必要な結果を実現するために連係する関連したアクションのリストで構成されています。例として JMS コンポーネントでは、キューから命令データを読み込み、一時 XML ドキュメントにデータをマップし、特定の品目についてデータのトランザクションを実行して、変換されたデータを出力 XML ドキュメントにマップするアクションがアクションモデルに含まれます。

このアクションモデルは、いくつかの個々のアクションから構成されています。そのアクションは次のとおりです。

- ◆ (おそらくフィルタ機能を利用して) Read Message アクションを実行する
- ◆ メッセージの内容を一時 XML ドキュメントにマップする
- ◆ コードテーブルを使用してデータ項目を転送する
- ◆ オプションでその他のコンポーネントを実行する
- ◆ 結果を出力 XML ドキュメントにマップする

## JMS コンポーネントエディタに固有なアクション

JMS コンポーネントエディタには、exteNd Composer の XML マップコンポーネントエディタの重要な機能がすべてと、次のようなコネクタ専用の 4 つのアクションタイプが存在します。

- ◆ Browse Message
- ◆ Message Transaction
- ◆ Receive Message
- ◆ Send Message

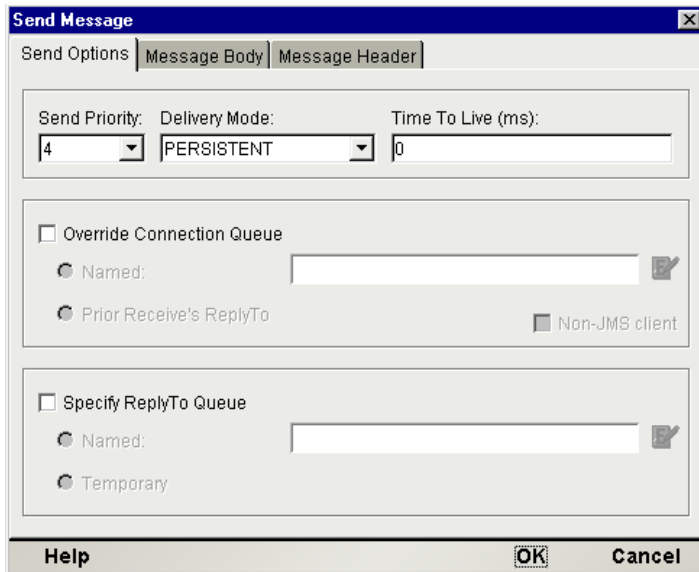
Message Transaction アクションタイプを除き、さまざまなメッセージアクションが一般的なセットアップダイアログ ボックスすべてに共通しています (Message Transaction アクションタイプを使用すると、JMS コンポーネントで Commit または Rollback、あるいはその両方のステートメントを配置できます。詳細については後で説明する「Message Transaction アクション」を参照してください)。セットアップダイアログボックスには 3 つのタブがあります。

- ◆ オプションタブ (アクションタイプに応じて [Send Options]、[Browse Options]、または [Receive Options] とラベルに表示されます)
- ◆ [Message Body] タブ
- ◆ [Message Header] タブ

[Browse Message] および [Receive Message] ダイアログボックスには [Filter] タブもあります (後から「Browse Messages アクション」および「Receive Message アクション」で説明します)。

### オプションタブ

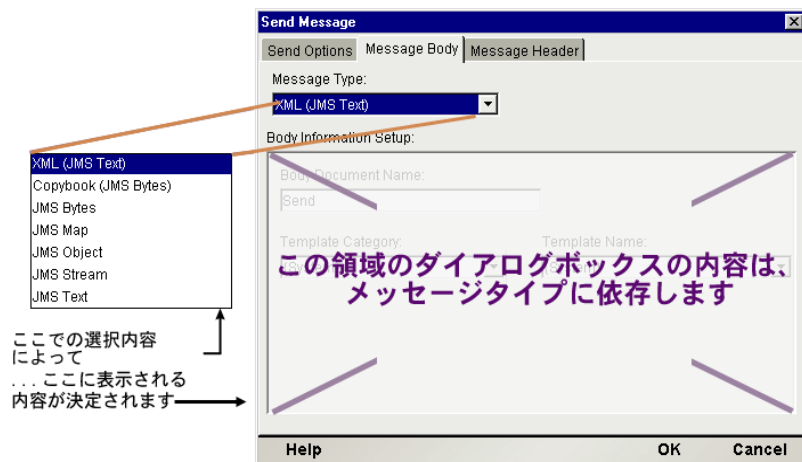
オプションタブには、対象のアクションタイプ (Send Message、Browse Message、または Receive Message) に固有なオプションが表示されます。たとえば、Send Message アクションには次の図のようなオプションパネルがあります。



このタブに表示されるオプションは、「メッセージタイプ」に固有なものではなく、「アクション」に固有です。つまり、このタブの表示は、BytesMessage または MapMessage のいずれでも同様です。

## [Message Body] タブ

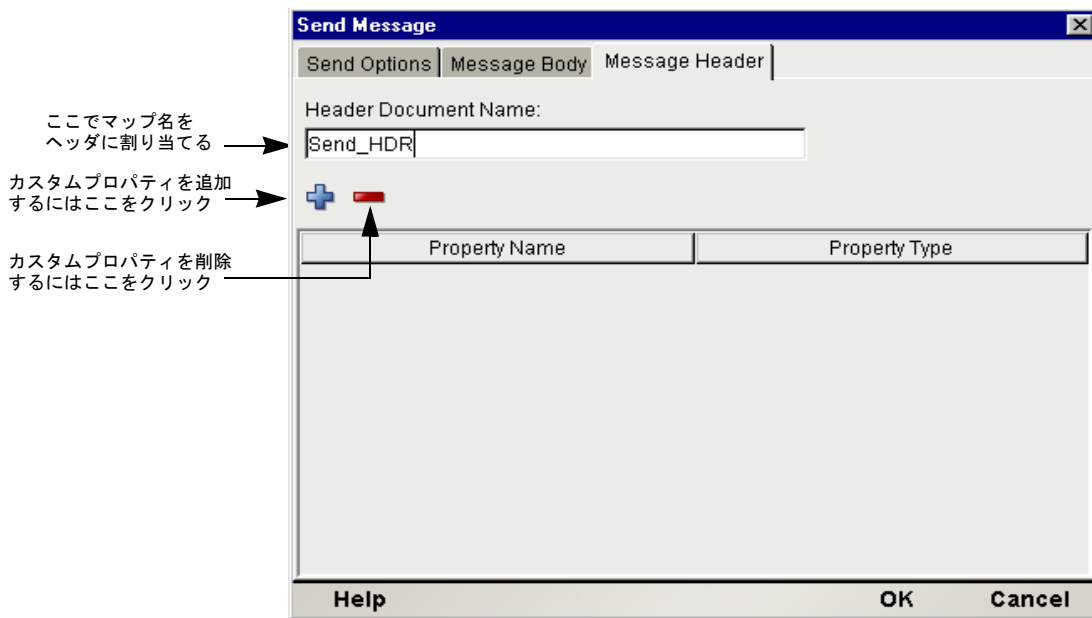
[Message Body] タブには、メッセージタイプによって異なるセットアップパラメータを含むペインが表示されます。



このペインには図のとおり **[Message Type]** プルダウンメニューがあり、XML および Copybook タイプ、さらに事前定義済みの 5 つの JMS メッセージタイプにアクセスできます。ここでの選択によって、ダイアログボックスの下の部分にある **[Body Information Setup]** ペインが変化します (さまざまなフィールドおよびその使用の詳細については、後のセクションを参照してください)。

## **[Message Header] タブ**

**[Message Header]** タブには、すべてのメッセージアクションタイプに共通なペインが表示されます。



このタブでは、JMS により定義された内蔵型のヘッダを補完するためにオプションで (カスタムヘッダフィールドと同様の) 「カスタムプロパティ」を作成します。

**[Header Document Name]** には、値をネイティブ環境ペインのプロパティフィールドにマップするためターゲットラベルとして後に使用できる名前を入力する必要があります (このダイアログボックスで直接フィールドの値を割り当てることはできません)。Send Message アクションでは、デフォルト名は Send\_HDR です。

## Send Message アクション

Send Message アクションは、キューまたはトピックにメッセージを送信する場合に使用します。メッセージの優先度、配信モード（永続的または非永続的）、および Time to Live（有効期限）をアクションごとに指定できます。また、アクションごとに送信先キューまたはトピックを指定したり、オプションで発信メッセージのヘッダにある JMSReplyTo フィールドで名前付きのキューを指定したりできます。

Send Message アクションの [Send Options] ボタングループについては、個別の説明が必要です。ボタングループには 3 つあります。上のグループでは、サービスの質に関連する設定を指定できます。中央のグループでは、送信先キューまたはコンポーネントの接続リソースで指定された別のキューを指定できます。下のグループでは、発信メッセージの「返信先」を指定できます。各ボタングループを順番に説明します。

### 優先度、モード、および有効期限

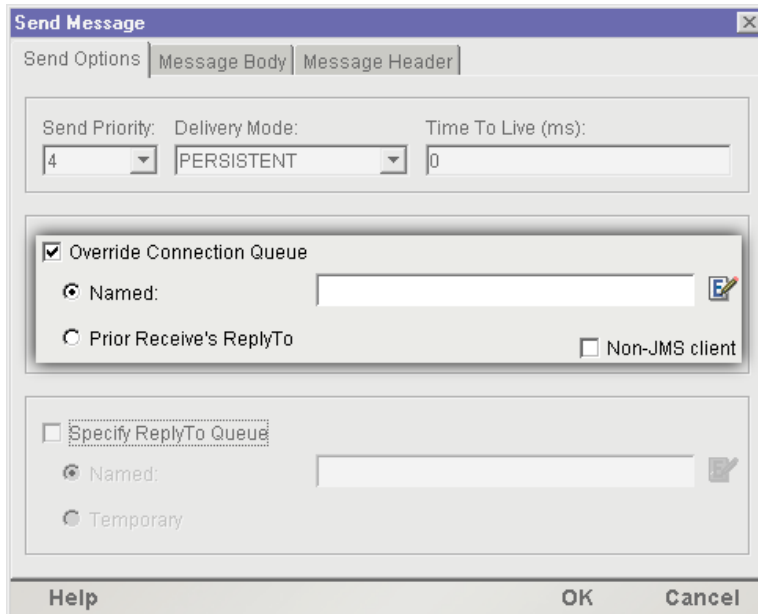
[Send Options] パネルにある上のボタングループを使用すると、サービスの質に重要なプロパティを指定できます。

- ◆ **[Send Priority]** — 1 から 9 までの数字で、メッセージに優先度を割り当てます。JMS 定義のデフォルトの優先度は 4 です。プルダウンメニューを使用すると、デフォルト値を上書きできます（メッセージの優先度に関連付けられている実装の詳細は、JMS 標準で指定されていません）。
- ◆ **[Delivery Mode]** — メッセージについて、送信先に転送される途中で保存されるよう永続的にするか（最高の信頼性）、回復能力を持たず迅速に配信するか指定できます。
- ◆ **[Time to Live]** — メッセージの最長有効期限を割り当てることができます（ミリ秒単位）。値をゼロに設定すると、メッセージの有効期限は無期限になります。

### 送信先キュー / トピック

コンポーネントの JMS 接続リソースで指定された Send Message アクション送信先キューがデフォルトです（JMS コンポーネントでデフォルトの送信先を変更するには、[File]、[Component]、[Connection Info] の順に選択し、プルダウンメニューから別のキューを選択します）。

デフォルトの動作を上書きする場合、Send アクションの送信先キューまたはトピックをアクションごとに指定できます。デフォルトの動作を「変更」するには、[Send Message] ダイアログボックスの [Send Options] パネルにある [Override Connection Queue/Topic] チェックボックスをオンにします。



[Override Connection Queue/Topic] チェックボックス ( 上の図 ) をオンにすると、その下にある 2 つのラジオボタンが有効になります。

- ◆ [Named] ラジオボタンをオンにすると、横のテキストフィールドに ( 引用符に囲まれた文字列として ) キューまたはトピック名を入力するか、ECMAScript を使用してキューまたはトピックを指定することができます。スクリプトを指定できるとは、キューまたはトピックの選択が、ランタイム時に取得される条件または値、あるいはその両方を利用してカスタム論理に基づくことが可能ということです。名前付きキューを指定するスクリプトを作成するには、テキストフィールドの右にある [Expression] アイコンをクリックし、[Expression Editor] ダイアログボックスを表示します。[Expression Editor] にスクリプトを入力して ( またはそこに表示されるリストを利用してスクリプトを作成し )、[OK] をクリックします。テキストフィールドにスクリプトが表示されます。特定のキューまたはトピックの名前を表す文字列に対しスクリプトで最終的に評価します。

**注記：** IBM MQSeries キューを使用している場合、[Named] テキストフィールドに次のようなキューの完全な形式の URI を指定できます。

```
queue://qmanager/queue1?CLIENTTYPE=0
```

- ◆ [Override Connection Queue/Topic] の下にある 2 番目のラジオボタン [Prior Receive's ReplyTo] をオンにすると、発信メッセージは *JMSReplyTo* フィールドが空白でない最後の受信メッセージの *JMSReplyTo* フィールドで指定されたキューまたはトピックに移動します。Send Message アクションが特に受信メッセージ (同じコンポーネントにおける前の Receive Message アクション) への返信を目的としている場合、このラジオボタンを選択します。

**注記:** このラジオボタンの説明に登場する *JMSReplyTo* フィールドは、(このコンポーネント内の) 最後の受信メッセージに関連付けられている最後の *JMSReplyTo* フィールドで、*JMSReplyTo* フィールドが空白でないものです。例えば、コンポーネントに 3 つの Receive Message アクションが存在し (順番に A、B、および C とします)、*JMSReplyTo* フィールドが B および C で空白、メッセージ A では空白でない場合、[Use Prior Receive's ReplyTo] ラジオボタンの設定により Send Message アクションではメッセージ A で指定されたキューが使用されます。3 つのメッセージすべてで ReplyTo フィールドが空白でなく、MessageA に返信する場合は、[Use Prior Receive] ラジオボタンを設定しません。その代わりに、[Named] ラジオボタンを選択して、メッセージ A のヘッダ DOM から *JMSReplyTo* 要素を取得する式を指定します。

## MQSeries 固有の動作

接続リソースで IBM MQSeries メッセージキューを指定すると、[Override Connection Queue/Topic] の下にあるボタングループに特別な [Non-JMS Client] チェックボックスが表示されます。メッセージを非 JMS メッセージコンシューマ、つまり MQSeries サービスのユーザに送信する場合、このチェックボックスをオンにします。このオプションを使用する場合、プロセスの受信は、(たとえば)ヘッダのエンコードやデコードに関する JMS ルールの知識がない非 Java メッセージコンシューマであることが想定されます。したがって、このオプションを使用する際、メッセージ受信者によりすべてのヘッダ情報が受信される、または受信者が情報を受信した場合でも受信者が処理方法を知っているとは限りません。たとえば、いかなる種類でもユーザ定義のヘッダプロパティを指定したり、JMSCorrelationID または JMSType ヘッダフィールドのいずれにも値をマップしたりしないようにします。

ただし、メッセージが非 JMS クライアントに送信される場合でも、キューマネージャ (JMS 対応) で特定のヘッダの値が使用されることに注意してください。一般的に、[Non-JMS Client] がオンの場合でも、サービスの質に関係するすべてのヘッダフィールドはキューマネージャに受け入れられます (JMSDeliveryMode または JMSExpiration などを除く)。

IBM MQSeries キューを使用している場合、[Override Connection Queue] コントロールグループの [Named] テキストフィールドに次のようなキューの完全な形式の URI を指定できます。

```
queue://qmanager/queue1?CLIENTTYPE=0
```

## 返信用アドレス

[Send Options] パネルの一番下にあるボタングループでは、発信メッセージで *JMSReplyTo* ヘッダフィールドに含まれる値を制御できます。

The screenshot shows the 'Send Message' dialog box with the 'Send Options' tab selected. The 'Send Priority' is set to 4, 'Delivery Mode' is PERSISTENT, and 'Time To Live (ms)' is 0. The 'Override Connection Queue' checkbox is checked, and the 'Named' radio button is selected. The 'Specify ReplyTo Queue' checkbox is also checked, and the 'Named' radio button is selected. The 'Temporary' radio button is unselected. The 'Non-JMS client' checkbox is unselected.

デフォルトでは、発信メッセージの *JMSReplyTo* フィールドは空白です。通常、発信メッセージにより受信者側の返信をトリガする場合、または発信メッセージにより ( エラーレポートなどの ) 返信が誘導される可能性がある場合は、これを変更します。デフォルトの動作を上書きするには ( そして *JMSReplyTo* フィールドで返信用アドレスを指定するには )、[ **Specify ReplyTo** ] チェックボックス ( 上の図を参照 ) をオンにしてからその下にある 2 つのラジオボタンのいずれかを選択します。

同期および非同期の 2 つのシナリオを例に考えます ( ただし、いずれの場合でも確実に返信メッセージを受信できるわけではありません )。ラジオボタンはこれらの 2 つのシナリオに対応します。

- ◆ [Named] ラジオボタンは、非同期のシナリオで使用します。このラジオボタンを選択すると、発信メッセージの *JMSReplyTo* フィールドにキューまたはトピック名を指定することになります。つまり、受信者に「返信メッセージを送信する場合、このアドレスに送信してください」と伝えるのと同じです。横にあるテキストフィールドにキュー名 ( またはトピック名 ) を引用符で囲み手動で入力するか、キューまたはトピック名に対して評価する ECMAScript の式を作成する必要があります。



- ◆ **[Temporary]** ラジオボタンは、同期的な方法で返信メッセージを受信する場合に使用します。このラジオボタンを選択すると、返信メッセージを即時に受信するために一時キューが作成され、受信プロセスでその応答の送信先が認識されるように発信メッセージの *JMSReplyTo* フィールドに一時キュー名が含まれます (一時キューは、コンポーネントの有効期限の間のみ存在します。次の章の「一時キュー」を参照してください)。

### ➤ Send Message アクションを作成する

- 1 JMS コンポーネントを作成または開きます (前の章で説明されています)。
- 2 Send Message アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。
- 3 **[Action]** メニューから、**[New Action]**、**[Send Message]** の順に選択します。**[Send Message]** ダイアログボックスに **[Send Options]** パネルが表示されます。

The screenshot shows the 'Send Message' dialog box with the 'Send Options' tab selected. The dialog has three tabs: 'Send Options', 'Message Body', and 'Message Header'. The 'Send Options' tab contains the following fields and controls:

- Send Priority:** A dropdown menu with '4' selected.
- Delivery Mode:** A dropdown menu with 'PERSISTENT' selected.
- Time To Live (ms):** A text input field with '0' entered.
- Override Connection Queue:** A checkbox that is unchecked. Below it are two radio buttons: 'Named:' (selected) and 'Prior Receive's ReplyTo'. To the right is a 'Non-JMS client' checkbox, which is also unchecked.
- Specify ReplyTo Queue:** A checkbox that is unchecked. Below it are two radio buttons: 'Named:' (selected) and 'Temporary'.

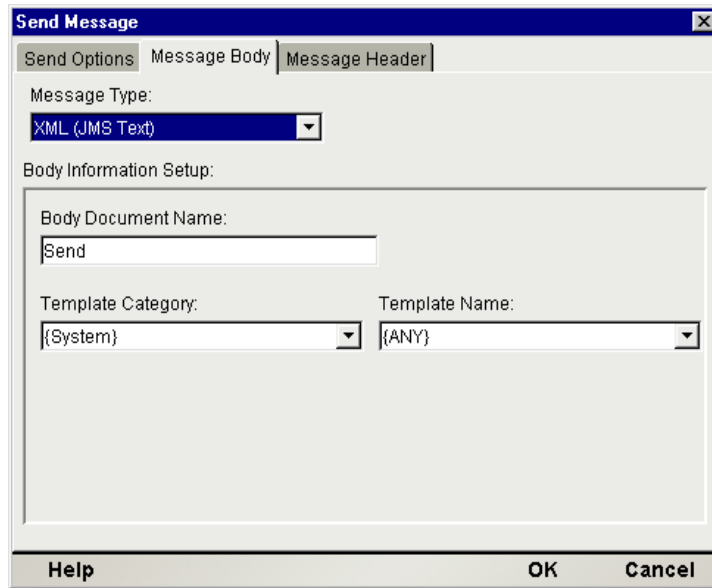
At the bottom of the dialog are 'Help', 'OK', and 'Cancel' buttons.

- 4 **[Send Priority]** プルダウンメニューから、メッセージの優先度 (0 から 9 まで) を選択します。優先度は、ゼロが最低で 9 が最高です。

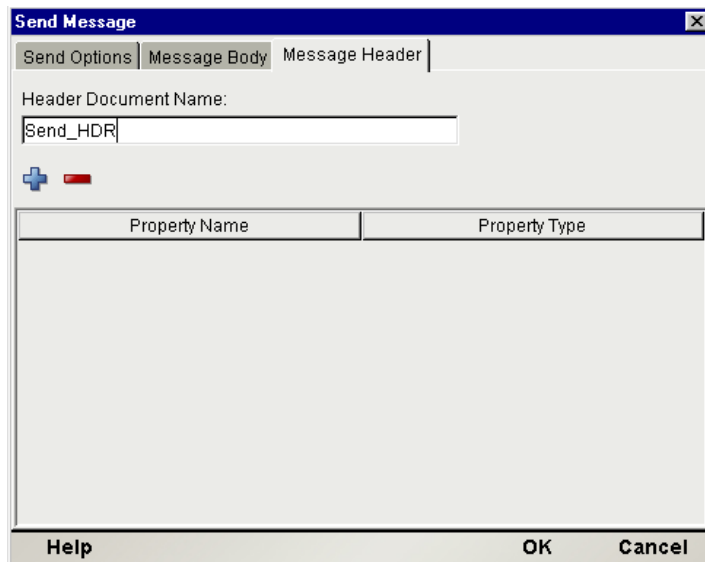
**注記:** この優先度の値の処理方法に関する詳細は、JMS 標準では定義されません。詳細については、MOM ベンダのドキュメンテーションを参照してください。

- 5 **[Delivery Mode]** プルダウンメニューから、PERSISTENT または NON\_PERSISTENT を選択します (用語の意味の詳細については、19 ページ「配信の保証について」を参照してください)。
- 6 **[Time To Live]** にミリ秒単位の値を入力して、メッセージの有効期限を設定します。有効期限を無期限にするには、ゼロを入力します (この設定の詳細については、127 ページ「メッセージヘッダおよびプロパティ」を参照してください)。

- 7 接続リソースで指定されているキューまたはトピック以外にメッセージを送信する場合は、**[Override Connection Queue]** チェックボックスをクリックして、すでに説明したように適当なラジオボタンを選択します。
- 8 発信メッセージのヘッダにある **JMSReplyTo** フィールドの値を指定する場合、**[Specify ReplyTo]** チェックボックスをクリックして、すでに説明したように適切なラジオボタンを選択します。
- 9 **[Message Body]** タブをクリックします。新しいペインが表示されます。

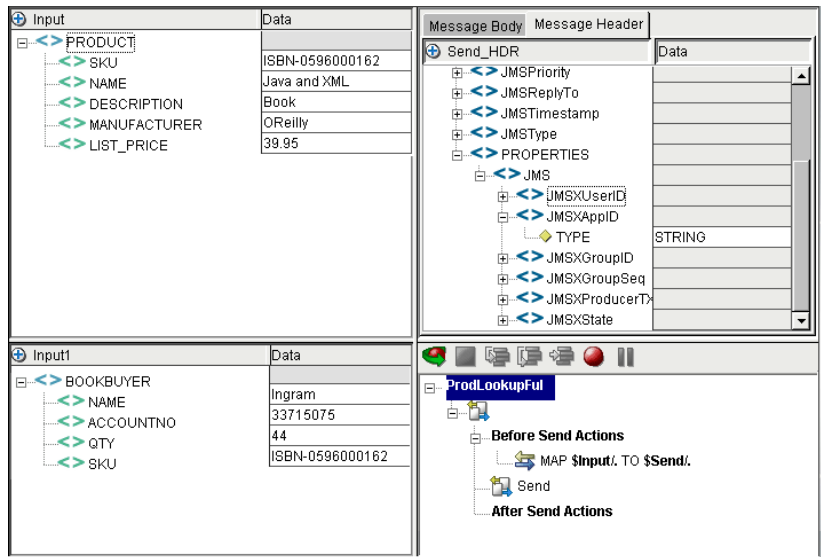


- 10 **[Message Type]** プルダウンメニューから、7つのメッセージタイプのいずれかを選択します (この例では XML タイプが選択されており、ポストされたメッセージに XML 形式のテキストドキュメントが含まれることを表しています)。
- 11 ダイアログボックスの **[Body Information Setup]** 領域に、必要な情報を入力します。ダイアログボックスのこの領域は、選択したメッセージタイプによって表示が異なります。この例のようにメッセージタイプが XML の場合、**[Body Document Name]** (メッセージ本文の DOM に適用する名前)、**[Template Category]** (XML テンプレートのリソース名)、および **[Template Name]** (必要な場合にメッセージ本文に適用する XML スタブドキュメント名) の情報を入力するボックスが表示されます。詳細については、77 ページ「**JMS** コンポーネントエディタでの他のアクションの使用」を参照してください。
- 12 メッセージのヘッダにカスタムプロパティを追加する場合は、**[Message Header]** タブをクリックします。**[Message Header]** ペインが表示されます。



- 13 [Header Document Name]を入力します (または、Send\_HDR で始まるデフォルト名を受け入れます)。この名前は、フィールドに値をマップするためにヘッダツリーの名前としてネイティブ環境ペインに表示されます。
- 14 プラス (+) アイコンをクリックして、プロパティを追加します。[Property Name] にカスタムプロパティの名前を入力して [Property Type] 列をクリックすると、使用可能なデータタイプのメニューが表示されます。プロパティに対応したデータタイプを選択します。上の例では、SKU\_PREFIX というカスタムプロパティが作成され、文字列として指定されます。

**注記：** プロパティの値を指定するには、ネイティブ環境パネルに移動する必要があります。このダイアログボックスでは、空白のプロパティが作成されます。
- 15 プラス (+) アイコンを必要な回数クリックして、その他のプロパティを追加します。それぞれに名前とタイプの情報を入力してください。
- 16 [OK] をクリックします。JMS コンポーネントエディタのメインウィンドウのアクションリストに、新しい Send アクションが表示されます。



前に登場したネイティブ環境ペインで [Message Header] タブが選択されている場合、SKU\_PREFIX というユーザプロパティが表示されます。これは、セットアップダイアログで作成したカスタムプロパティです。

## 送信前と送信後

Send Message アクションが作成されると、アクションリストに「Before Send Maps」および「After Send Maps」という行が表示されます。2 つのマッピングが存在する理由は、JMS 定義のヘッダフィールドの一部が Send アクションを実行するまで空白であるためです。実行後、同じフィールドに JMS プロバイダまたは JMS コンポーネントの内部メソッドのいずれかによりデータが入力されます。特に、Send アクションの実行後に入力されるフィールドは次のとおりです。

- ◆ JMSDestination
- ◆ JMSDeliveryMode
- ◆ JMSExpiration
- ◆ JMSPriority
- ◆ JMSMessageID
- ◆ JMSTimestamp
- ◆ JMSRedelivered

メッセージの送信前にこれらのヘッダの場所にマップされたデータは、Send アクションの実行時に上書きされます。これらのフィールドを読み取り専用にし、フィールド内のデータは Send アクションの実行後にのみ有効にすることをお勧めします。

書き込み可能なヘッダフィールドは次のとおりです。

- ◆ JMSCorrelationID
- ◆ JMSType

(ネイティブ環境ペインで) これらのフィールドをダブルクリックしてデータを手動で入力するか、ドロップ先として DOM からドラッグアンドドロップによりマップできます。

**注記：** JMSReplyTo フィールドは書き込み可能ですが、ドラッグアンドドロップまたは直接編集することはできません。このフィールドに入力するには、[Send Message] ダイアログの [Send Options] タブにあるコントロールを使用する必要があります。

次の図は、メッセージが送信された後のネイティブ環境ペインです (フィールドが自動入力された後)。

Send_HDR	Data
MSGHEADER	
JMSCorrelationID	
JMSDeliveryMode	PERSISTENT
JMSDestination	queue://clq_default_opal
JMSExpiration	Wed Dec 31 19:00:00 EST 1969
JMSMessageID	ID:414d5120514d5f670616c2e67656d6cc185a63913a00
JMSPriority	4
JMSReplyTo	
JMSTimestamp	Thu Dec 07 11:42:35 EST 2000
JMSType	
PROPERTIES	

ログ作成、デバッグ、出力 DOM へのマップなどの目的で読み取り専用フィールドのデータを使用する場合は、アクションモデルの「After Send Maps」行の下に適切な Map アクションを追加する必要があります。入力 DOM データを JMSCorrelationID などのヘッダフィールドにマップする場合、80 ページで説明したとおり、ドラッグアンドドロップにより入力 DOM の要素とヘッダフィールドの間でマップを作成できます。

## Browse Messages アクション

参照操作を使用すると、メッセージをキューから削除することなくアプリケーションでキューからメッセージを検索できます。したがって、使用中のアプリケーションにより取得できるよう、参照操作の後もすべてのメッセージをキューで使用できます。

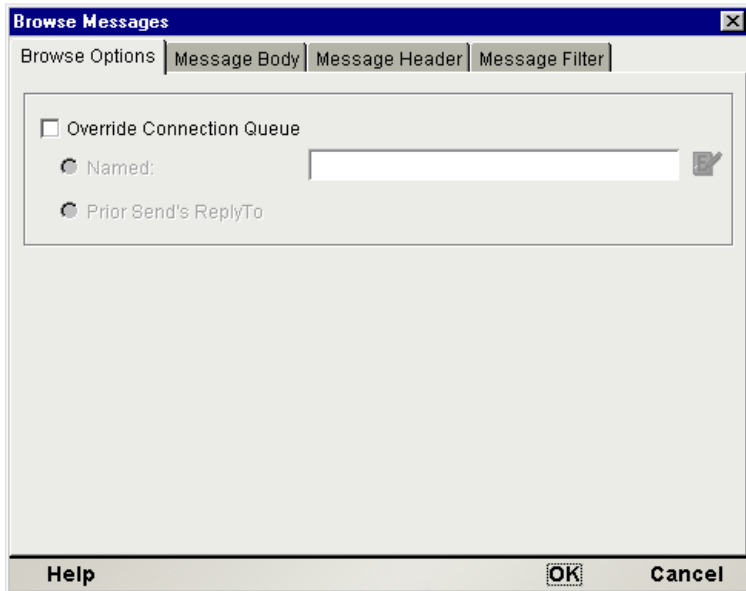
参照要求に応じて、メッセージフィルタ (または「セレクトア」) が指定されていない限り、キューマネージャにより有効なすべてのメッセージを含む `java.util.Enumeration` が返されます。メッセージフィルタ (またはセレクトア) が指定されている場合は、セレクトアステートメントに一致するメッセージのみが返されます (セレクトアの使用の詳細については、96 ページ「メッセージフィルタ (セレクトア) の操作」を参照してください)。

JMS コンポーネントでは、キューからメッセージを参照するには **Browser Messages** アクションを使用します。デフォルトでは、コンポーネントの JMS 接続リソースで指定されたキューが参照されます (JMS コンポーネントでデフォルトのキューを変更するには、**[File]**、**[Component]**、**[Connection Info]** の順に選択し、プルダウンメニューから別のキューを選択します)。また、デフォルトの設定はアクションごとに上書きできます (次を参照)。

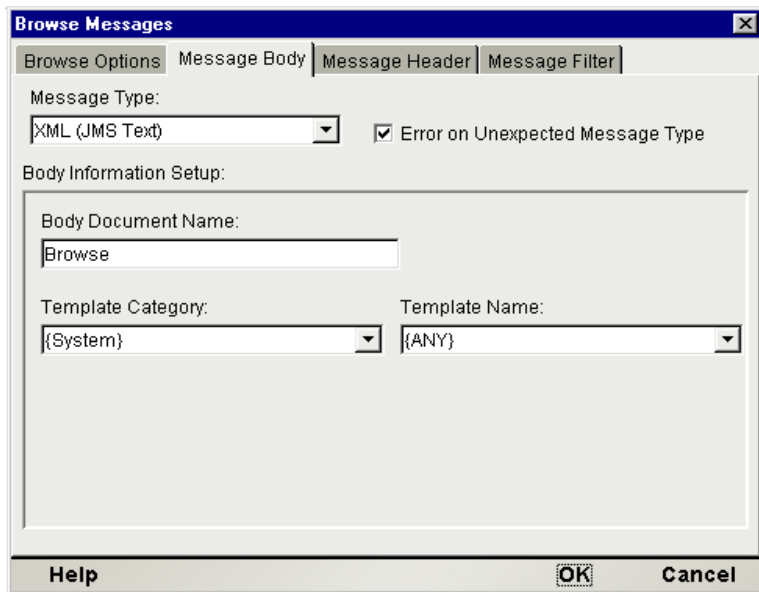
**注記：** 参照は、ポイントツーポイントの操作のみです。パブリッシュ / サブスクライブでは、参照は定義されません。Browse Messages アクションを使用する場合、(トピックではなく) キューを使用するには接続リソースを設定する必要があります。

### ➤ Browse Messages アクションを作成する

- 1 JMS コンポーネントを作成または開きます (前の章で説明されています)。
- 2 Browse Messages アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。
- 3 **[Action]** メニューから、**[New Action]**、**[Browse Messages]** の順に選択します。**[Browse Messages]** ダイアログボックスに **[Browse Options]** パネルが表示されます。



- 4 (オプション)コンポーネントの接続リソースで指定したメッセージキュー以外を参照する場合は、**[Override Connection Queue]** チェックボックスをオンにします (デフォルトではオフ)。
  - ◆ キュー名を手動で指定する場合は、**[Named]** ラジオボタンをクリックします (横にあるテキストフィールドに引用符で囲んだキュー名を入力するか、キュー名に対して評価する ECMAScript の式を作成します)。
  - ◆ 最後の **Send Message** アクションで指定されたキューを参照する場合は、**[Use Sent Message ReplyTo Field]** ラジオボタンをクリックします (送信先キュー名を取得するために、`exteNd` により `JMSReplyTo` フィールドが空白でない最後の送信メッセージが検索されます。このメッセージは最後に受信されたメッセージと異なる場合があります)。
- 5 必要に応じて、**[Filter]** のテキストフィールドにフィルタ式を指定します (詳細については、付録 B 「メッセージフィルタのシンタックス」を参照してください)。これは ECMAScript の式であるため、引用符で文字列を囲む必要があります。
- 6 **[Message Body]** タブをクリックします。新しいパネルが表示されます。



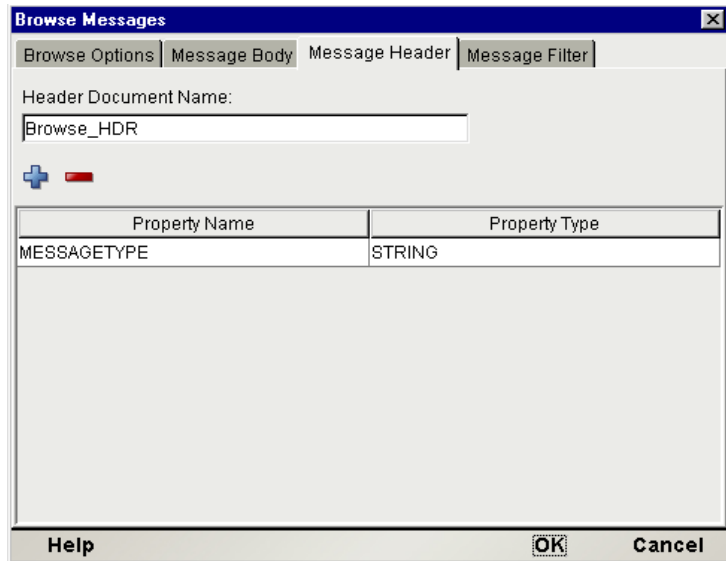
- 7 [Message Type] プルダウンメニューから、コンポーネントで処理するメッセージの種類に対応するメッセージタイプを選択します。この操作により、使用しているアプリケーションの要件を満たす形式で着信データを受信、保存できるようメッセージ本文が事前に設定されます。

**注記：** この選択では、exteNd Composer で不要なメッセージタイプをフィルタできません。JMS の参照操作では、本文のタイプに関係なく有効なメッセージすべてのリストが常に返されます。

- 8 予期しないメッセージ本文のタイプが返されても exteNd Composer で例外が発生しないようにする場合、[Error on Invalid Message Type] チェックボックスをオフにします (デフォルトはオン)。通常、JMS プロセスは JMSText などのある特定の JMS メッセージタイプを「理解」して処理するように設計されています。したがって、JMSText ではなく、JMSBytes が返された場合のように予期しないメッセージタイプが返された場合、処理エラーが発生します。一般的に、処理に関する問題はなるべく早く発見する必要があるため、このチェックボックスのデフォルト設定はオンになっています。ただし、特にテストの際など、本文のタイプに関係なく有効なすべてのメッセージを JMS コンポーネントで処理することが必要な場合もあります。そのような場合にはチェックボックスをオフにします。

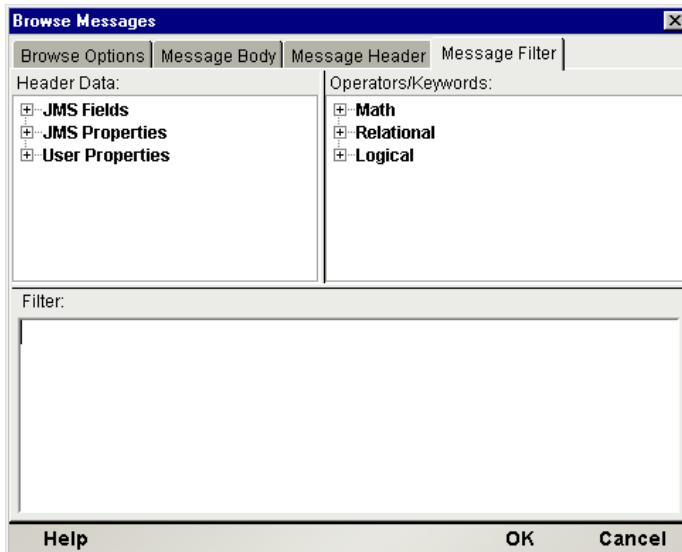


- 9 ダイアログボックスの **[Body Information Setup]** 領域で、メッセージタイプに応じて必要な情報を入力します ( ダイアログボックスのこの領域は、選択したメッセージタイプによって表示が異なります)。この例のようにメッセージタイプが XML の場合、**[Body Document Name]** ( メッセージ本文の DOM に適用する名前 )、**[Template Category]** (XML テンプレートのリソース名)、および **[Template Name]** ( 必要な場合にメッセージ本文に適用する XML スタブドキュメント名 ) の情報を入力するボックスが表示されます。詳細については、77 ページ「JMS コンポーネントエディタでの他のアクションの使用」を参照してください。
- 10 **[Message Header]** タブをクリックします。新しいパネルが表示されます。



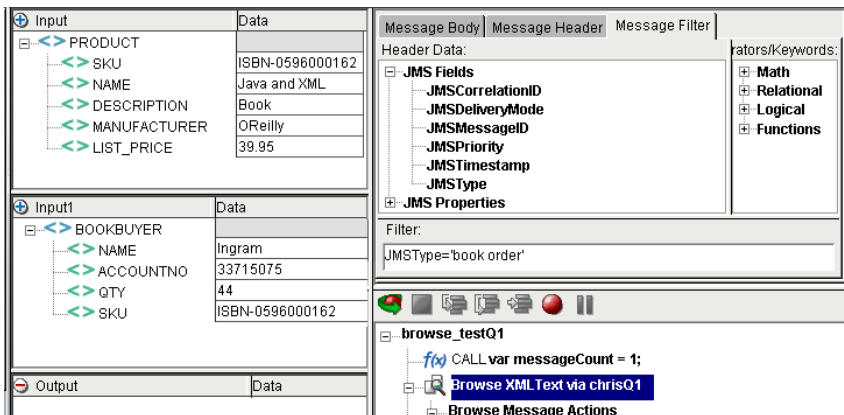
- 11 マップ名を入力します ( または、Browse\_HDR で始まるデフォルト名を受け入れます )。この名前は、マップのためにヘッダツリーの名前としてネイティブ環境ペインに表示されます。
- 12 プラス (+) アイコンをクリックして、プロパティを追加します。 **[Property Name]** にカスタムプロパティの名前を入力して **[Property Type]** 列をクリックすると、使用可能なデータタイプのメニューが表示されます。プロパティに対応したデータタイプを選択します。
- 注記：** この手順の目的は、予想される着信メッセージのプロパティリストに対応したプロパティリストを作成することです。ここで設定されていないプロパティフィールドを持つ着信メッセージに対しては、追加フィールドは無視され、関連付けられているデータは失われます。
- 13 プラス (+) アイコンを必要な回数クリックして、その他のプロパティを追加します。それぞれに名前とタイプの情報を入力してください。

- 14 [Message Filter] タブをクリックします。新しいパネルが表示されます。



- 15 参照操作に JMS メッセージフィルタを適用する場合、ダイアログボックスの下半分のテキスト領域に入力します ( またはリストの項目を使用してフィルタを作成することもできます。詳細については、96 ページ「メッセージフィルタ (セレクト) の操作」を参照してください)。

- 16 [OK] をクリックします。JMS コンポーネントエディタのメインウィンドウのアクションリストに、新しい Browse アクションが表示されます ( 次の図を参照 )。



## メッセージの繰り返し

新しい Browse アクションを作成する場合、アクションリストに「Browse Messages: Type = ...」という行が表示され、続いて「Browse Message Maps」、そして次のように始まる行が順番に表示されます。

```
WHILE JMSMESSAGE.hasMessages() . . .
```

JMSの参照操作では、有効なメッセージすべてのリストが常に返されるため(フィルタ条件に基づきます。詳細については、96 ページ「メッセージフィルタ(セレクト)の操作」を参照してください)、各 Browse Messages アクションの一部として JMS Connect により WHILE ループが自動的に構築されます。このループは、有効な各メッセージに対して繰り返されます。アクションモデルの一部として、必要な任意の Map アクション(またはその他の処理)をループに配置できます。また、コンポーネントエディタの [Break] および [Continue] コマンドを使用して、他のループと同様にループコントロールを実行できます。

次に説明する Receive Message アクションと同様、Browse Messages アクションのネイティブ環境ペインには [Message Body]、[Message Header]、および [Message Filter] タブがあります。これらのタブの操作については、次の章で説明します。

## Receive Message アクション

受信操作を使用すると、アプリケーションでキューからメッセージを受信できます。メッセージを受信するアクションによって、メッセージはキューから破壊的に削除されます。ただし、Receive Message アクションが実行されているにもかかわらずメッセージセッションが処理されロールバックされた場合のみ例外で、このような場合にはメッセージは最終的には処理されません。キューに残ります。

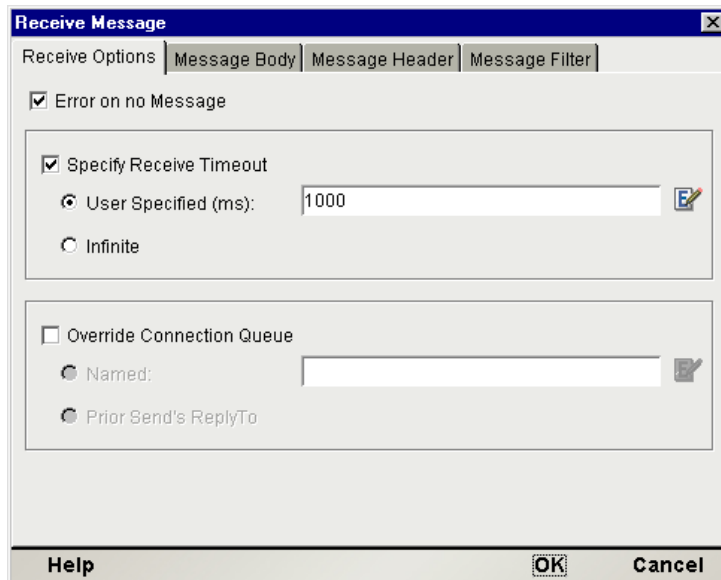
受信要求に応じて、メッセージフィルタ(または「セクタ」)が指定されていない限り、キューまたはトピックマネージャにより最初の有効なメッセージが返されます。メッセージフィルタ(または「セクタ」)が指定されている場合は、セクタステートメントに一致する最初の有効なメッセージのみが返されます。

JMS コンポーネントでは、Receive Message アクションはキューまたはトピックからメッセージを取得するために使用します。コンポーネントの JMS 接続リソースで指定されたキューまたはトピックが使用されます(JMS コンポーネントでキューまたはトピックを変更するには、[File]、[Component]、[Connection Info] の順に選択し、プルダウンメニューから別のキューまたはトピックを選択します)。

**注記:** 1回の Receive Message アクションで取得できるメッセージは1つだけであるため、キューまたはトピックからすべてのメッセージを取得するには Repeat/While ループを構築する必要があります。Repeat While アクションは、JMSMessageID ヘッダフィールドが空白になるとループが終了するように設計します。

## ➤ Receive Message アクションを作成する

- 1 JMS コンポーネントを作成または開きます (前の章で説明されています)。
- 2 Receive Message アクションを配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。
- 3 [Action] メニューから、[New Action]、[Receive Message] の順に選択します。[Receive Message] ダイアログボックスに [Receive Options] パネルが表示されます。

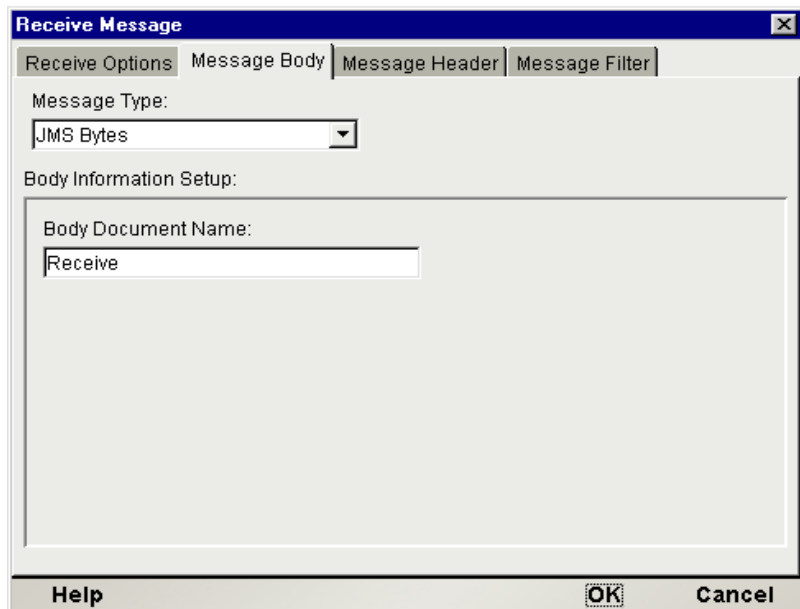


- 4 タイムアウトまでの時間にメッセージが受信されない場合でも例外が発生しないようにする場合は、[Error on No Message] チェックボックスをオフにします (デフォルトはオン)。デフォルトのタイムアウト時間はユーザ指定でない限り (手順 6 を参照してください)、NO\_WAIT またはゼロです。

**注記：** このチェックボックスをオンにした場合、Try/On Error アクションで Receive アクションを囲み、On Error 分岐で適切な回復手順を実行する必要があります。ただし、このチェックボックスのオン / オフに関係なく、Receive アクションの実行時にキューが空白である可能性も予想して、アプリケーションで適切に処理されるよう確認しておく必要があります。

- 5 タイムアウト時間の値をデフォルトの NO\_WAIT (ゼロ) 以外に設定する場合、[Specify Receive Timeout] チェックボックスをオンにします (デフォルトはオフ)。このチェックボックスがオフの場合、アクションによりキューまたはトピックが確認され、有効なメッセージが存在する場合は取得して待機することなく即時に返されます。指定した時間だけアクションをブロックするには、このチェックボックスをオンにして待機時間を指定します。

- ◆ タイムアウト時間について固有な値を入力する場合、[User Specified] ラジオボタンを選択します。タイムアウト時間の値を横にあるテキストフィールドにミリ秒単位で入力するか、適当な数に対して評価する ECMAScript の式を作成します。
  - ◆ (メッセージが受信されるまで) Receive アクションを無期限にブロックする場合、[Infinite] ラジオボタンを選択します。
- 6** コンポーネントの接続リソースで指定されたキューまたはトピック以外を指定する場合、[Override Connection Queue] チェックボックスをオンにします (デフォルトはオフ)。
- ◆ あるキューまたはトピックを明示的に指定する場合、[Named] ラジオボタンを選択します (デフォルト)。キューまたはトピック名を引用符で囲んで横にあるテキストフィールドに入力するか、キューまたはトピック名に対して評価する ECMAScript の式を入力します。
  - ◆ *JMSReplyTo* ヘッダフィールドが空白でない最後に送信されたメッセージの *JMSReplyTo* フィールドにあるキューまたはトピック名を使用する場合、[Use Prior Message ReplyTo Field] ラジオボタンを選択します。
- 注記:** アクションが前に送信されたメッセージへの返信を待機している場合、手順 6 の時点で適当なタイムアウト時間の値を指定しておく必要があります。安全性とパフォーマンスを考慮して最適なタイムアウト時間の値を決定するには、何回かテストを行う必要があります。
- 7** [Message Body] タブをクリックします。新しいパネルが表示されます。



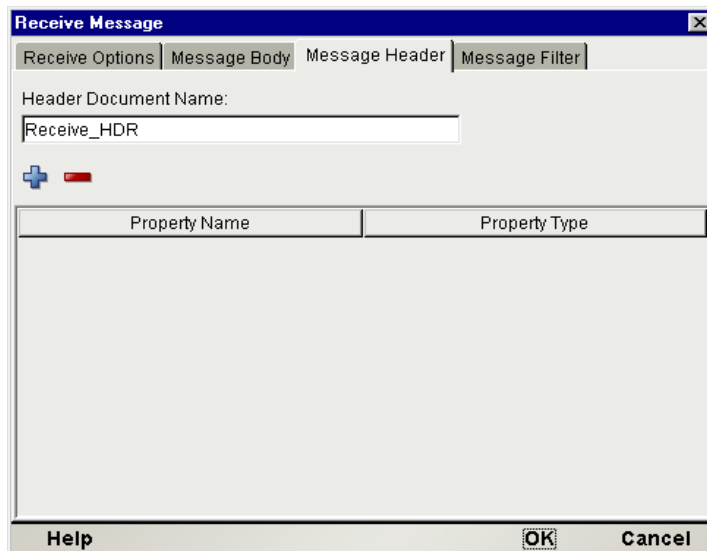
- 8 [Message Type] プルダウンメニューから、コンポーネントで受信するメッセージの種類に対応するメッセージタイプを選択します。この操作により、使用しているアプリケーションの要件を満たす形式で着信データを受信、保存できるようメッセージ本文が事前に設定されます。

**注記：**メッセージの適切な本文のタイプが受信されるように、アプリケーションを設定することが重要です。メッセージ作成アプリケーションでは一般的に、メッセージを受信できるように設計されている対応アプリケーションに対してのみメッセージが送信されるため、通常はこの点について問題ありません。[Receive Message] ダイアログボックスで選択するタイプと互換性がないタイプのメッセージを受信した場合は例外が発生します。アプリケーションで、多くの異なるメッセージ本文のタイプを含むキューからメッセージが受信される場合、使用しているアプリケーションに対応するメッセージを識別できるメッセージフィルタ(セレクトステートメント)を作成することをお勧めします。

- 9 [Body Document Name] に、DOM コンテキストのために着信メッセージの本文と関連付ける名前を入力します(または「Receive」で始まるデフォルト名を受け入れます)。

**注記：**ダイアログボックスのこの領域は、選択したメッセージタイプによって表示が異なります。この例のようにメッセージタイプが XML の場合、[Body Document Name] (メッセージ本文の DOM に適用する名前)、[Template Category] (XML テンプレートのリソース名)、および [Template Name] (必要な場合にメッセージ本文に適用する XML スタブドキュメント名)の情報を入力するボックスが表示されます。詳細については、77 ページ「JMS コンポーネントエディタでの他のアクションの使用」を参照してください。

- 10 [Message Header] タブをクリックします。[Message Header] ペインが表示されます。



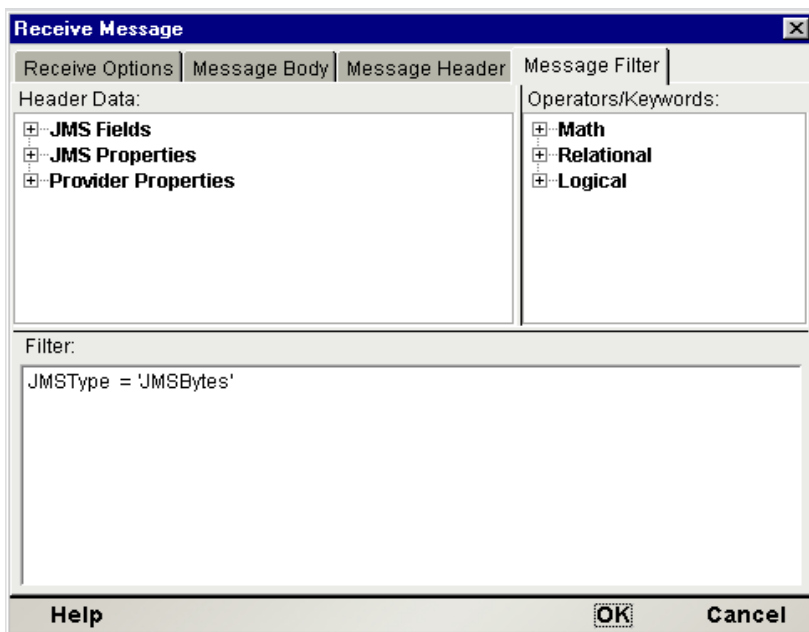
**11** マップ名を入力します ( または、「Receive\_HDR」で始まるデフォルト名を受け入れます )。この名前は、マップのためにヘッダツリーの名前としてネイティブ環境ペインに表示されます。

**12** プラス (+) アイコンをクリックして、プロパティを追加します。[Property Name] にカスタムプロパティの名前を入力して [Property Type] 列をクリックすると、使用可能なデータタイプのメニューが表示されます。プロパティに対応したデータタイプを選択します。

**注記：** この手順の目的は、予想される着信メッセージのプロパティリストに対応したプロパティリストを作成することです。ここで設定されていないプロパティフィールドを持つ着信メッセージに対しては、追加フィールドは無視され、関連付けられているデータは失われます。

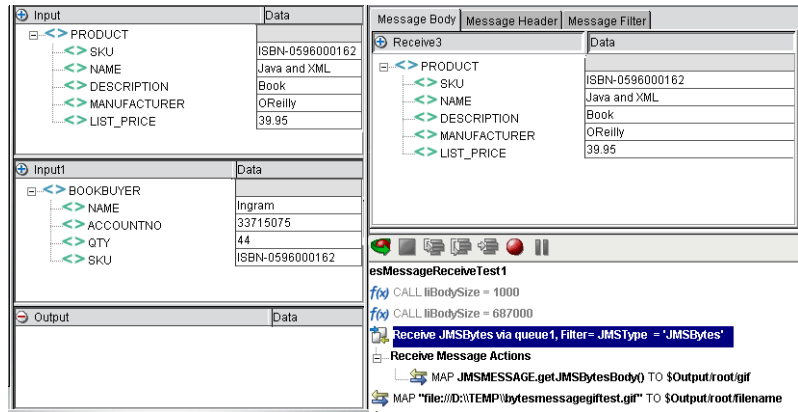
**13** プラス (+) アイコンを必要な回数クリックして、その他のプロパティを追加します。それぞれに名前とタイプの情報を入力してください。

**14** [Message Filter] タブをクリックします。新しいパネルが表示されます。



**15** 参照操作に JMS メッセージフィルタを適用する場合、ダイアログボックスの下半分のテキスト領域に入力します ( またはリストの項目を使用してフィルタを作成することもできます。詳細については、96 ページ「メッセージフィルタ (セレクト) の操作」を参照してください )。

- 16** [OK] をクリックします。JMS コンポーネントエディタのメインウィンドウのアクションリストに、新しい Receive アクションが表示されます ( 次の図を参照 )。



新しい Receive アクションを作成すると、アクションリストに「Receive Message ...」という行が表示され、続いて「Receive Message Maps」と表示されます。「Receive Message Maps」行の下に、受信されるメッセージに含まれる情報を使用するために必要な任意の Map アクションまたはその他の処理を挿入できます。

Browse Messages 操作とは異なり、Receive Message アクションでは 1 回のアクションで取得できるメッセージは 1 つに制限されています。したがって、JMS Connect には各 Receive Message アクションの一部として WHILE ループが含まれていません。

**注記：** すべての有効なメッセージに対してアクションを繰り返すには、JMSMessageID フィールドが空白な場合に終了するように Receive Message アクション ( および関連付けられている処理 ) を Repeat While アクションで囲む必要があります。

すでに説明した Browse Messages アクションと同様、Receive Message アクションのネイティブ環境ペインには [Message Body]、[Message Header]、および [Message Filter] タブがあります。これらのタブの操作については、次の章で説明します。



## Message Transaction アクション

Message Transaction アクションを使用すると、複数のメッセージ関連アクションを 1 つの論理的な作業単位にグループ化してまとめて実行 (または実行されないように) できます。トランザクションがコミットされると、メッセージに関するすべての入力を確認され、すべての出力が送信されます。Transaction アクションのメッセージセッションがロールバックされると、生成されたメッセージはすべて破棄され、セッション中に使用されたメッセージはすべて回復されます。

JMS コンポーネントで Message Transaction の呼び出しを活用するには、Transacted モードでセッションが発生することが重要です。Transacted モードは、コンポーネントの JMS 接続リソースで [Transacted] チェックボックスにより設定できます (手順については後に説明があります。第 2 章の「JMS 接続リソースの作成」も参照してください)。JMS コンポーネントにより関連付けられている JMS セッションが Transacted モードの場合、アクションモデルで (Message Transaction アクションから) Commit および Rollback ステートメントを安全に使用できます。

### Commit ステートメントを発行した場合

Commit ステートメントを発行すると、各メッセージアクション (必要に応じてアクションモデルの始め、または最後の Commit あるいは Rollback ステートメントのいずれかから開始されます) が実行されます。この時点までに作成された各メッセージは送信され、それまでに処理されたメッセージは確認されます (受信操作では、メッセージの確認によりキューからメッセージを破壊的に削除できることがキューマネージャに伝えられます)。

### Rollback ステートメントを発行した場合

Rollback ステートメントを発行すると、各メッセージアクション (アクションモデルの始め、または最後の Commit あるいは Rollback ステートメントのいずれかから開始されます) が取り消されます。この時点までに作成された各メッセージは破棄され (送信されません)、それまでに処理されたメッセージは回復されます (何も実行されずにキューに残ります)。

### セッションを未解決のままにした場合

Transaction アクションのセッションで Commit または Rollback ステートメントのいずれも使用されない場合、または適切にコミットされた (またはロールバックされた、あるいはその両方) 複数のアクションを含むアクションモデルがコミットあるいはロールバックのいずれでもないメッセージアクションで終了する場合、あいまいな状態になります。このような状態の処理は、MOM ベンダによりかなり異なります。自動的に未解決のアクションがコミットされる場合もありますし、コミットされていないものすべてがロールバックされる場合もあります。

exteNd JMS Connect では、コミットされていないトランザクションの状態には自動ロールバックプロトコルが実行されます (これは、処理された JDBC コンポーネントの最後の SQL ステートメントで Commit または Rollback のいずれも使用されていない場合に、接続リソースセットアップダイアログボックスで「Allow SQL Transactions」をオンにすると自動ロールバックが実行される JDBC Connect の動作と類似しています。詳細については、「JDBC Connect ガイド」の第 2 章を参照してください)。つまり、最後の Commit または Rollback ステートメント以降に発生したすべてのメッセージアクションは、セッションの終了時にロールバックされます。JMS により MOM のサービスが「囲まれる」ため、MOM の通常動作よりもこの動作が優先されます。MOM のデフォルト設定が自動コミットまたは自動ロールバックのいずれかに関係なく、JMS Connect ではコミットされていないメッセージについて自動ロールバックが確実に実行されます。

## Message Transaction に含まれるアクション

JMS コンポーネントでは、Message Transaction アクションの他にメッセージ関連でない Transaction アクション ( [New Action]、[Advanced]、[Transaction] の順に選択します ) も使用できるため、これら 2 つの違いを理解することが重要です。

Message Transaction アクションでは、JMS セッションを対象に commit () および rollback () メソッドが呼び出されます。したがって、これらのメソッドはメッセージ関連のプロセスにのみ実行されます。たとえば、Message Transaction コマンドによりメッセージの送信はロールバックできますが、JDBC コンポーネントへの外部呼び出しにより実行されるデータベースの操作を JMS コンポーネントのアクションモデル内でロールバックすることはできません。アクションモデルに Receive Message アクション、続いて Component アクション (JDBC コンポーネントへの呼び出し)、次に Send Message アクションが含まれ、エラーの際に (データベース操作を含めて) すべてをロールバックする場合は、[New Action]、[Message]、[Transaction] という JMS コンポーネント固有の操作ではなく、すべてのコンポーネントに使用できる [New Action]、[Advanced]、[Transaction] の操作を使用してトランザクションを区別する必要があります。JMS コンポーネント固有の操作では、メッセージ操作のみがロールバックされます。

## Message Transaction アクションの対象

JMS トランザクションの一般的な用途は、関連する一連のメッセージアクションで「すべてまたはゼロ」動作を実行することです。たとえば、( 命令確認、ベンダ通知、およびバックエンドクエリも含め ) 一連のメッセージについてすべて一度に送信するか、まったく送信しないでおく必要があるとします。複雑なビジネス論理により、メッセージの送信が認証されたかどうか判断されます。Transaction アクションに対し JMS コンポーネントを使用すると、セッションについてデフォルトでメッセージが「送信され」、エラーの場合またはプロセスにおける任意の段階で正常に返されなかった場合には Send Message ( または Receive Message、あるいはその両方の ) 操作がロールバックされるように設定できます。

Transaction セッションの別の用途として、非破壊的なメッセージの読み出しがあります。通常、アプリケーションによりキューからメッセージを読み出すと、メッセージはキューから永久的かつ回復できないように削除されます。しかし、JMS アプリケーションでは、Transaction セッションでメッセージを読み出す場合、セッションをロールバックするかどうか決定する前にメッセージの内容を確認できます。セッションがロールバックされると、何も実行されずにメッセージはキューに残ります。この機能が単に参照するよりも便利な理由は ( 参照の操作を思い出してください。受信操作と異なりメッセージは破壊的に処理されます )、アプリケーションでメッセージにアクションを実行するかどうか決定する前に、メッセージの本文を確認する必要があるためです。参照操作の場合、ワークフローは次のとおりです。

- 1 参照します。
- 2 メッセージの内容を確認します。
- 3 メッセージの処理が必要な場合、キューから読み出し ( キューから確実に削除されます )、メッセージの内容を処理します。そうでない場合は何も実行しません。

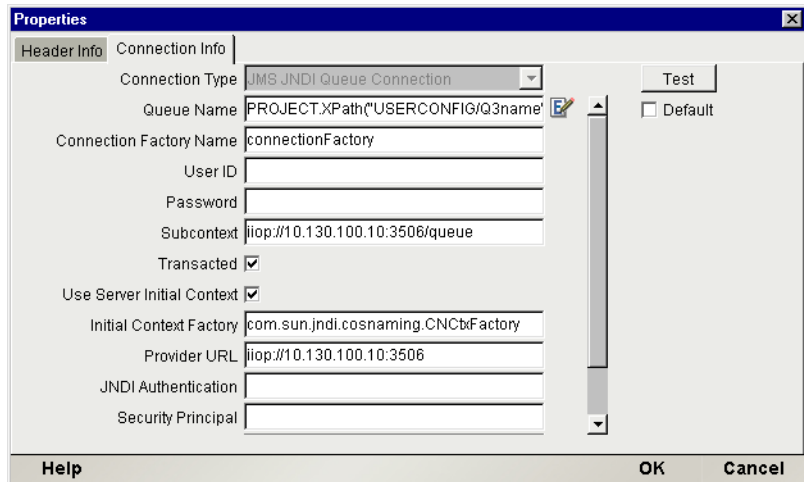
アプリケーションによりまず参照され、それから読み出し操作が行われることに注意してください。つまり、キューまでに 2 つの動作が存在します。Transaction セッションの場合、ワークフローは次のとおりです。

1. キューからメッセージを読み出します。
2. メッセージの内容を確認します。
3. メッセージの処理が必要な場合、アクションを実行します。そうでない場合、セッションをロールバックします。

この場合、キューまでの動作は 1 つのみです ( メッセージの読み出し )。メッセージが適当でない場合、ロールバックによりメッセージはキューに残され、何も実行されません。メッセージが有効かどうかに関係なく、キューまでの動作は 1 つだけです。

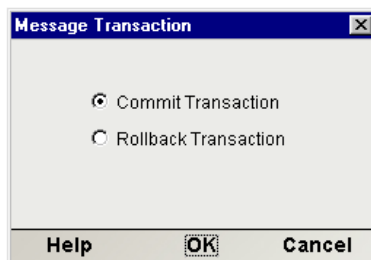
## ➤ Message Transaction アクションを作成する

- 1 JMS 接続リソースでトランザクション可能であることを確認します。exteNd Composer メインスクリーンのナビゲーションペインで、[Connection]、[Resource] の順にクリックして、詳細が表示されるペインで適当な JMS 接続リソースをダブルクリックします。[Properties] ダイアログボックスで、[Connection Info] タブを選択します。次の画面が表示されます。



ダイアログボックスの下の領域に [Transacted] チェックボックスが表示されます。JMS コンポーネントで Message Transaction コマンドを使用する場合は、このチェックボックスをオンにします。このチェックボックスをオフにすると、Message Transaction による呼び出しによりコンポーネントで例外が発生します。

- 2 メインメニューから、[Action]、[New Action]、[Message Transaction] の順に選択します。ダイアログボックスが表示されます。



- 3 必要に応じて、[Commit Transaction] または [Rollback Transaction] を選択します。

**注記:** Begin ステートメントを発行する必要はありません。接続リソースで [Transacted] チェックボックスをオンにすると ( 前の図参照 ), Begin ステートメントが暗示されます。このチェックボックスをオンにすると、JMS セッション全体が Transaction アクションのコンテキストに配置されます。

- 4 [OK] をクリックします。適当なステートメントがアクションリストに挿入されます。

## JMS コンポーネントエディタでの他のアクションの使用

Send Message、Browser Message、Receive Message、および Message Transaction アクションの他にも、JMS コンポーネントエディタの標準的なあらゆる Composer アクションを使用できます。[Action] メニューには、基本的なアクションおよび高度なアクションの両方のリストが表示されます (「exteNd Composer ユーザガイド」で説明されています)。



# 5

## メッセージの操作

JMS Connector の主な目的は、exteNd のサービスで MOM (メッセージ指向ミドルウェア) の能力を利用し、ビジネスアプリケーションのフロントエンドとバックエンド領域の間でさまざまな操作を可能にすることです。この能力を最大限に活用するには、JMS コンポーネントでメッセージによりさまざまな形式のコンテンツが伝達される仕組みを理解する必要があります。この章では、メッセージを操作するプロセスについて説明します。

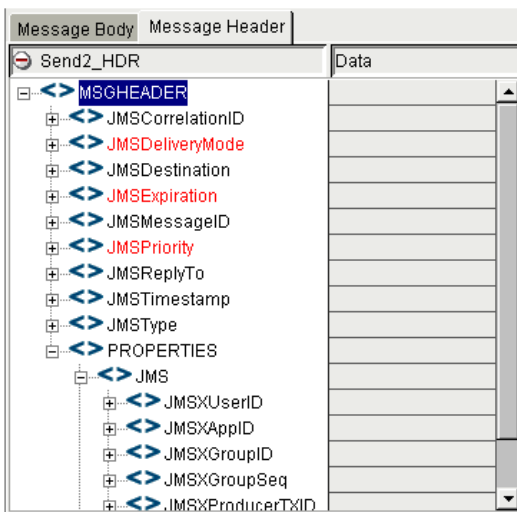
この章では次の内容について説明します。

- ◆ JMS メッセージヘッダとの間でデータをマップする方法
- ◆ カスタムプロパティとの間でデータをマップする方法
- ◆ DOM およびメッセージ本文との間でデータをマップする方法
- ◆ 特別な XML および Copybook タイプの使用方法
- ◆ メッセージフィルタ (JMS セレクタ) を使用して、アプリケーション定義の条件に基づきメッセージを選択的に受信または参照する方法
- ◆ exteNd の JMS 関連 ECMAScript 拡張機能を使用して、メッセージデータを操作する方法

この章をお読みになる前に、25 ページ始めの「JMS コンポーネントエディタをお使いになる前に」、41 ページ「JMS コンポーネントの作成」、および 49 ページ「JMS アクションの作成」をお読みになり、exteNd Composer および Map アクションの概念に関する知識を持つ必要があります。

## メッセージヘッダへのデータのマップ

ネイティブ環境ペインで [Message Header] タブをクリックすると、JMS ヘッダのツリービューが表示されます。



このヘッダ ( プロパティ情報および JMS 定義のヘッダフィールドを含みます ) は、マップのための固有な DOM を構成します。前の例では、ヘッダ DOM は Send\_HDR という名前です ( この名前は、メッセージアクションのセットアップダイアログボックスで設定されます。59 を参照してください )。Send\_HDR DOM は、メッセージヘッダにデータをマップするターゲットになります。また、出力 DOM の要素にマップするデータソースとしても使用されます。

JMSにより多くの既存のフィールドが定義されるため、メッセージ-ヘッダ DOM ツリーはすべてのメッセージと関連付けられています (そして [Message Header] タブを選択すると自動的にネイティブ環境ペインに表示されます)。したがって、ドラッグアンドドロップを使用して、入力 DOM の任意の領域からメッセージヘッダにまたはその反対方向に直接データをマップできます (制約については次に説明があります)。表示されている DOM ペインで入力ノードをクリックしたままメッセージヘッダの目的の位置までドラッグして、マウスボタンを放します。該当する Map アクションがアクションモデルに自動的に追加されます。



## ヘッダにマップする場合の制約



JMS 定義のヘッダフィールドのほとんどは読み取り専用なため (または JMS プロバイダにより使用されるため)、ドロップ先としては使用できません。

**JMSCorrelationID** および **JMSType** にのみドラッグアンドドロップを使用してマップできます (その他のフィールドに項目をドラッグすると左側に「forbidden drag operation」記号が表示され、コンポーネントエディタウィンドウのステータス行に「Write-restricted drop target」と警告が表示されます)。JMSReplyTo は書き込み可能ですが、[Send Message] ダイアログボックスの [Send Options] でのみ実行できます (後の説明を参照してください)。JMS のヘッダフィールドおよびその意味の詳細については、付録 C を参照してください。

### JMSCorrelationID

**JMSCorrelationID** フィールドは、アプリケーションのためにフィールドを追跡または制御する目的で使用されます。**JMSCorrelationID** の一般的な用途は、リクエストおよび応答目的に使用する認識文字列です。たとえば、少し後の例 (83 ページ「ユーザ定義のプロパティはすべて読み出し / 書き込み可能なため、ドラッグアンドドロップでマップする場合のドロップ先として使用できます。これについての制約は、互換性のないデータタイプをマップする場合のみです。たとえば、String 値を Integer 値として定義されているプロパティにドラッグすると、「invalid drag operation」アイコンが表示され、ステータスメッセージ (コンポーネントエディタウィンドウの左下) に「Invalid drag value for drop target: INTEGER」と表示されます。これは、すべてのヘッダフィールドに対するドラッグアンドドロップ操作に当てはまります。Extensible Composer ではドラッグ操作中に自動タイプチェック機能が実行され、タイプに関する制約違反が防止されます。」を参照) のように、別のメッセージではあり得ないような固有性を持たせることができるあいまいな追跡番号を現在のメッセージに付けることによって、送信アプリケーションで現在のメッセージによるビジネストランザクションを認識することができます。Function アクションを使用すると、次のように **JMSCorrelationID** フィールドに ECMAScript によるミリ秒単位の現在の日付で構成される固有な値と、(Input1 DOM からの) BOOKBUYER/NAME 値がハイフンで区切られて割り当てられます。

```
(Number(new Date)).toString() + '-' +  
Input1.XPath("BOOKBUYER/NAME")
```

この動作によりランタイム時に '976128839742-Ingram' のような値が発信メッセージの **JMSCorrelationID** ヘッダフィールドに表示されます。受信アプリケーションはこの値のメモを作成し、その発信メッセージに配置します。そうすると、配布されるアプリケーションの個別の要素で固有な命令としてこのカスタムの命令が認識され、一貫性した方法で操作されます。

**注記:** アプリケーションにより割り当てられる **JMSCorrelationID** の値は、「ID:」で始まることはできません。このプリフィックスは JMS プロバイダの使用のために予約されています。

## JMSType

**JMSType** フィールドには、任意の文字列を入力できます。送信メッセージでの **JMSType** の一般的な用途は、(同じキューが指定されている)受信アプリケーションがフィルタ目的で検査できる標識値を入力することです。アプリケーションの性質に応じて、入力 DOM からのマップ、コードテーブルの使用、ECMAScript の式を基礎にした入力、特定の値のハードコード化などを選択できます。

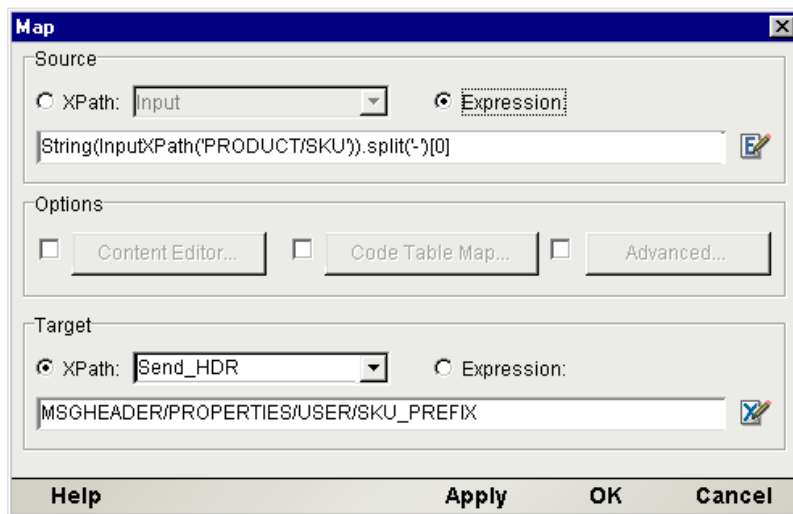
## JMSReplyTo

**JMSReplyTo** フィールドは書き込み可能ですが、ドラッグアンドドロップまたは直接編集することはできません。このフィールドに返信用アドレスを入力するには、前の章で説明したように [Send Message] ダイアログボックスの [Send Options] タブを使用します。[Send Options] タブには、[Specify ReplyTo Queue/Topic] チェックボックスがあります。このチェックボックスをオンにすると、横にあるテキストフィールドにキュー名 (引用符で囲みます) を入力するか、ランタイム時にキューに対して評価する ECMAScript の式を指定できます。

## カスタムプロパティへのデータのマップ

メッセージアクションの作成時に定義するカスタムプロパティは、すべてメッセージヘッダ DOM ツリーに自動的に表示されます。これらのプロパティには、ドラッグアンドドロップ、ECMAScript など通常の方法でデータをマップできます。

少し後の例では、SKU\_PREFIX というカスタムプロパティが使用されています。入力 DOM の PRODUCT/SKU データの一部、特に SKU の最初の部分のみ、最初のハイフンまで (ハイフンは含みません) をこの項目にマップするとします。これを実行する 1 つの方法は、Message Header ツリーの SKU\_PREFIX ノードを選択してマウスを右クリックし、[Map] ダイアログボックスを表示するために [Map...] を選択します。



[Source] 領域で [Expression] ラジオボタンをクリックし、次のように ECMAScript の式を入力します。

```
String(Input.XPath('PRODUCT/SKU')).split('-')[0]
```

ここでは、String オブジェクトの split() メソッドを使用して、ハイフンごとに SKU 文字列を分割します。split() メソッドにより最初の区切り文字 (この場合ハイフン) までの配列が返され、そのゼロ番目のメンバーが最初の下位文字列となります。したがって、この式に「ISBN-0596000162」という文字列を適用すると、「ISBN」が返されます。

受信アプリケーションでは、「ISBN」という SKU\_PREFIX 値を持つメッセージだけを処理するようにカスタマイズすることにより他のすべてのタイプを無視し、このタイプのメッセージをキューから選択的に取得できます。このようなフィルタはメッセージセレクトアで実行できます。

## プロパティのマップの制約

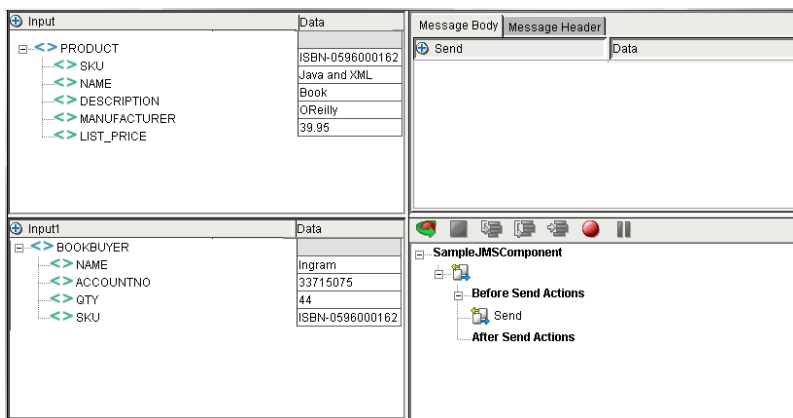
ユーザ定義のプロパティはすべて読み出し / 書き込み可能なため、ドラッグアンドドロップでマップする場合のドロップ先として使用できます。これについての制約は、互換性のないデータタイプをマップする場合のみです。たとえば、String 値を Integer 値として定義されているプロパティにドラッグすると、「invalid drag operation」アイコンが表示され、ステータスメッセージ (コンポーネントエディタウィンドウの左下) に「Invalid drag value for drop target: INTEGER」と表示されます。これは、すべてのヘッダフィールドに対するドラッグアンドドロップ操作に当てはまります。ExtJS Composer ではドラッグ操作中に自動タイプチェック機能が実行され、タイプに関する制約違反が防止されます。

## XML メッセージの操作

メッセージングの一般的な用途は、キューに XML ドキュメントを送信することです。たとえば、Web からリアルタイムで命令が発生すると、メッセージキューを経由してバックエンドの実行システムに渡されます。バックエンドのアプリケーションは、定期的に命令を取得するか、着信時にポーリングまたはリスンによって命令を取得します。

次の例では、出版社が Web で卸売業者および代理店から本の注文を受け取ります。要件は、JMS コンポーネントで 2 つの XML ソース (1 つはカスタマにより送信される情報、もう 1 つはデータベース検索で取得される商品情報) から情報を受け取って新しい XML ドキュメントに変換し、最終的にメッセージキューに送信できることです。

まず、XML メッセージタイプを使用して Send Message アクションを作成します (前の章で説明しました。「JMS アクションの作成」を参照してください)。



ネイティブ環境ペインの [Message Body] タブを選択すると、最初はメッセージの本文が空白になっています。

この特定のコンポーネントでは、商品情報を含む *Input* とカスタマ情報を含む *Input1* という 2 つの入力 DOM を使用します。これらの DOM の起源はさまざまです。Input DOM は、JDBC コンポーネントによりデータベースから取得されるデータを表します。Input1 DOM の情報は、Web (または別の JMS コンポーネントにより処理されたメッセージ) から着信するカスタマ情報です。

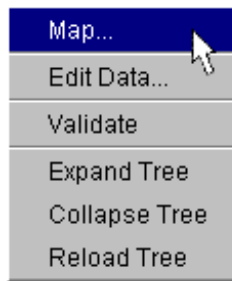
**注記:** メッセージ本文にこのコンポーネントの出力が含まれるため、この例では出力 DOM が表示されていません ([View]、[Window Layout]、[XML Layout] の操作によって、デフォルトの出力 DOM の表示が非表示になっています)。

## メッセージ本文へのデータのマップ

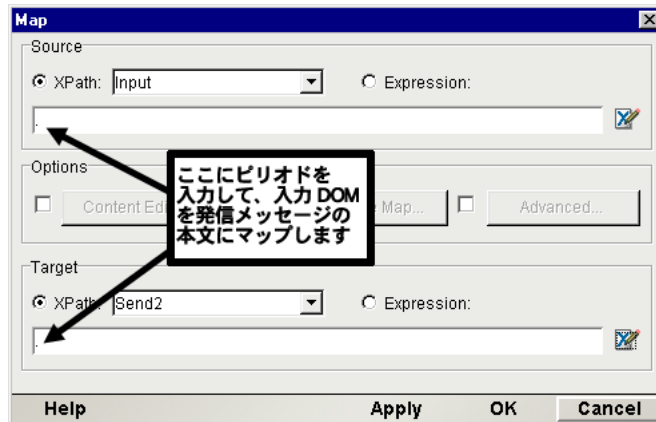
場合によっては XML ドキュメント全体をメッセージ本文にマップしたり、XML ドキュメントの一部だけをメッセージの本文にマップしたりする必要があります。両方の場合について順番に説明します。

### メッセージ本文への XML ドキュメント全体のマップ

ネイティブ環境ペインで [Message Body] タブを選択し、本文の空白になっている領域をマウスで右クリックしてコンテキストメニューが表示されたら、メニューから [Map] を選択します。

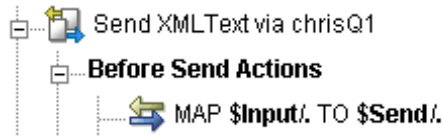


標準の XML Map アクションのダイアログボックスが表示されます。

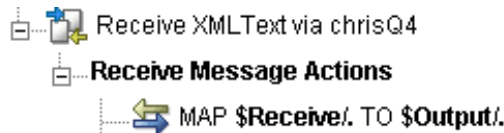


デフォルトのソース DOM は **Input** です (Input を使用しないで別の DOM を選択するには、プルダウンメニューを使用します)。[Map] ダイアログボックスの [Source] 領域内のテキストフィールドに、ピリオド (.) を入力し、ソース DOM 全体をターゲットにマップすることを設定します。

デフォルトの場所は [Send] です ( または Send アクションの作成時に指定された発信メッセージの名前です )。 [Send] の下にあるテキストフィールドにピリオドを入力します。ダイアログボックスを閉じます。アクションモデルで Map アクションが次のように表示されます。



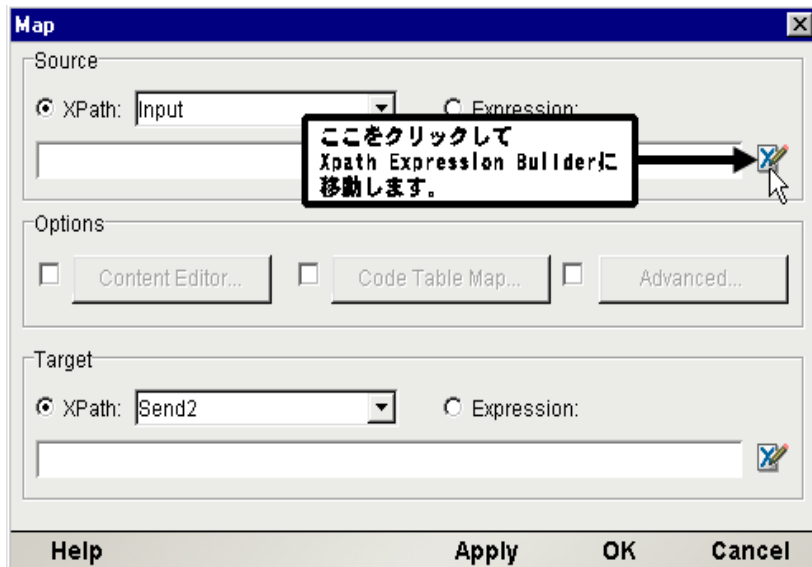
また、同じ手順を使用して、Receive アクションの一部として Receive から Output へメッセージのすべてをマップすることもできます。その場合の結果は次のようになります。



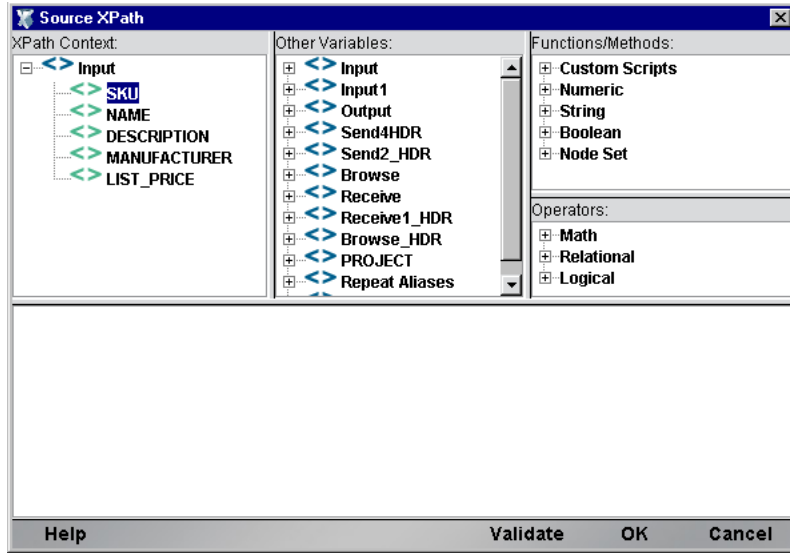
### メッセージ本文への XML ドキュメントの部分的なマップ

XML ドキュメントの一部を [Message Body] にマップするには、ネイティブ環境ペインの空白の領域を ( [Message Body] タブが選択されている状態で ) 右クリックします。コンテキストメニューが表示されます。

[Map...] コマンドを選択します。 [Map] ダイアログボックスが表示されます。

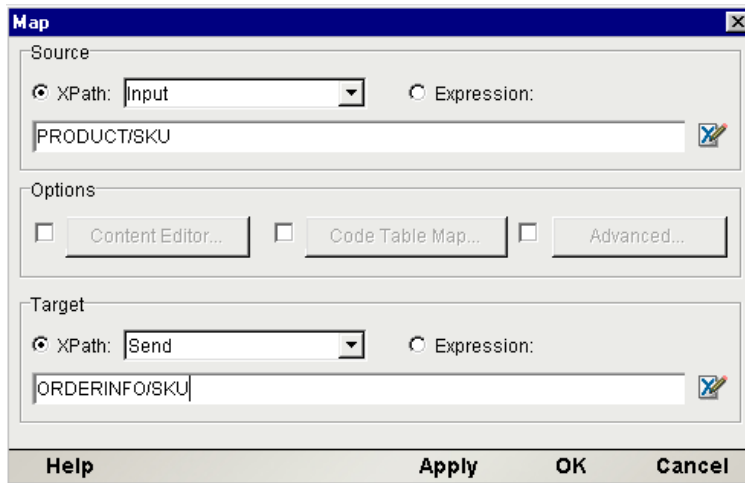


[Map] ダイアログボックスで、[Input] が [Source] 領域のデフォルトの DOM として表示され、[Send] がデフォルトのマップ先として表示されます (プルダウンメニューを使用すると、[Source] および [Target] 領域で別の DOM を選択できます)。ソースとして使用する XPath フラグメントが分かっている場合、提供されているボックスに入力します。分からない場合、右側の青い [Expression Editor] アイコンをクリックします。[Expression Editor] アイコンをクリックすると、ソースの XPath の [Expression Editor] ダイアログボックスが表示されます。

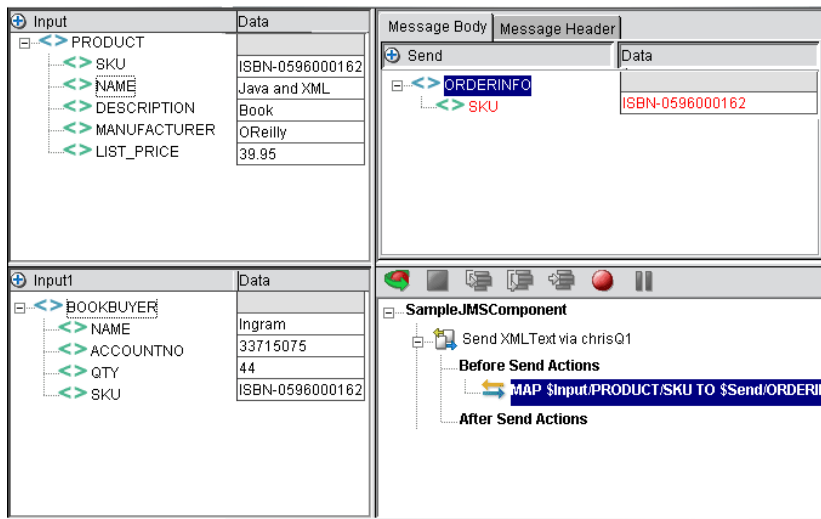


このダイアログボックスの上部にあるリストを使用すると、ポイントしてクリックするだけで、XPath フラグメントまたは ECMAScript 式、あるいはその両方を作成できます。この場合、入力 DOM (左上) のツリービューを展開して、完全な入力ツリー構造を表示します。ツリー内の SKU 項目をダブルクリックすると、ダイアログボックスの下部に PRODUCT/SKU (ツリーのその部分に対する XPath フラグメント) が自動的に表示されます。[OK] をクリックすると、[Map] ダイアログボックスの該当する箇所に XPath 情報が表示されます。

Input からメッセージ本文の ORDERINFO/SKU の XPath 位置に PRODUCT/SKU 情報をマップするには、[Map] ダイアログボックスの [Target] 領域に ORDERINFO/SKU と入力します。



[OK] をクリックすると [Map] ダイアログボックスが閉じ、JMS コンポーネントエディタのメインウィンドウでマップの結果を確認できるようになります。



この手順は、メッセージ本文にデータを入力するたびに繰り返すことができます。または、Function アクションを (ECMAScript DOM メソッドと) 使用して、プログラムによりメッセージ本文に XML ノードを作成することもできます。



## コピーブックメッセージの操作

JMS Connect の最も強力な機能の 1 つは、ペイロードが COBOL コピーブックで構成されているメッセージの送信、受信、および参照機能です。メッセージでコピーブックを使用できると、アプリケーション Composer により exteNd サービスでさまざまなレガシーシステムの操作ができます。特に、JMS Connect とともに CICS RPC Connect が使用されている場合に便利です。たとえば、JMS Component でメッセージとして受信されるコピーブックは、同じサービス内でビジネスニーズに応じて変換でき、RPC セッションへの入力として使用できます。

## コピーブックメッセージの設定

Send Message、Receive Message、または Browse Messages アクションを作成し、[Message Type] プルダウンメニューで「Copybook (JMS Bytes)」を指定すると、セットアップダイアログボックスの下の部分 ([Body Information Setup] 領域) にコピーブックとメッセージアクションを関連付けられるフィールドが表示されます。

Copybook Handle	Copybook File
-- Edit Copybook Handle --	-- Select Copybook File --

Copybook Data Parameters:

Code Page Cp037	Floating Point Format IBM
Machine Type MVS	Endian BIG

Browse

[Copybook Handle] には、JMS コンポーネントエディタでマップのためにコピーブックを認識する任意のテキスト文字列を入力できます。

コピーブックのファイル名は、[Copybook File] に表示されます。表示されない場合は、近くの [Browse] ボタンをクリックして使用するコピーブックを検索すると、[Copybook File] 領域にファイル名が表示されます。

[**Copybook Data Parameters**] には 4 つのプルダウンメニューがあり、[Code Page]、[Floating Point Format]、[Machine Type]、および [Endian] (バイトの並び順) をターゲット環境に応じて選択できます。

## Code Page

使用可能な文字エンコードは、存在する Java 2 ランタイム環境のバージョンによって多少異なります。[Code Page] プルダウンメニューに、使用されている Java ランタイムでサポートされているすべての文字エンコードが表示されます。

## Floating Point Format

JMS Connect でサポートされている浮動小数点形式は、IEEE-754 および IBM 形式の 2 種類です。

## Machine Type

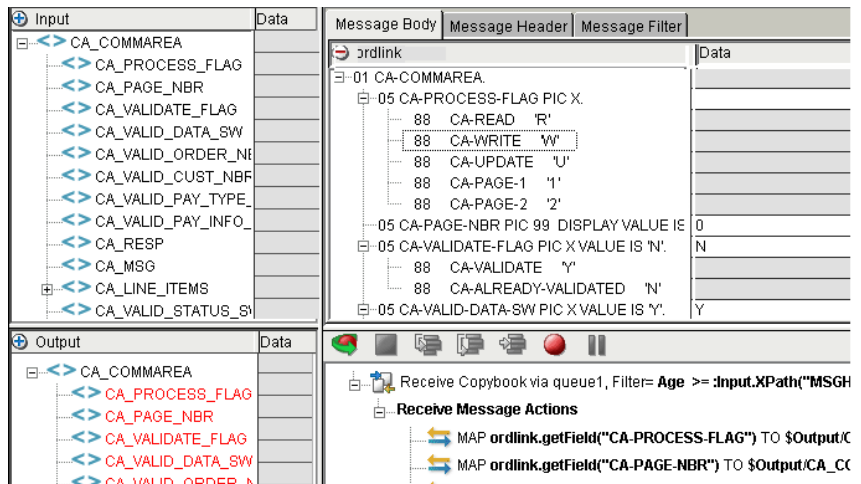
マシンタイプは、最終的に対象のコピーブックが受信または処理されるターゲットプラットフォームを指します。MVS、OS/2、NT、または AIX から選択できます。

## Endian

ターゲットプラットフォームのバイトの並び順を表し、BIG または LITTLE の 2 つの選択があります。Intel のアーキテクチャでは、マルチバイトエンティティの最下位バイトが最低のメモリアドレスになるリトルエンディアンアドレススキームが使用されています。その他ほとんどのマシンのアーキテクチャではビッグエンディアンが採用されています。

## コピーブックとネイティブ環境ペイン

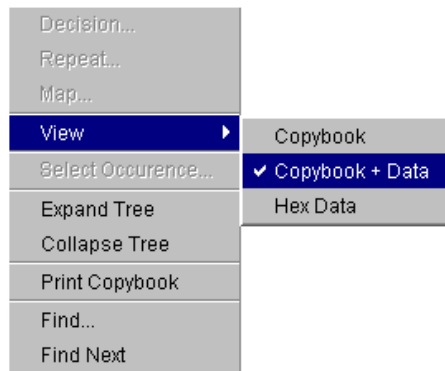
JMS コンポーネントエディタのメインウィンドウを表示して、コピーブックに関連するメッセージアクションを選択 (ハイライト) すると、メッセージアクションのセットアップダイアログボックスで選択したコピーブックに含まれる情報がネイティブ環境ペインに表示されます。



この例では、Receive Message アクションが選択されています。関連付けられているコピーブックが、右上のネイティブ環境ペインに表示されます。ネイティブ環境ペインには、コピーブックの表示モードが3種類あります。

- ◆ Copybook
- ◆ Copybook + Data (上の例)
- ◆ Hex Data

別のビューを選択するには、ネイティブ環境ペイン内でマウスを右クリックして、コンテキストメニューから [View] を選択します。



Hex Data ビューでは、コピーブックのコンテンツが標準の16進データで表示されます。

Message Body	Message Header	Message Filter
ordlink		
0000:	2030304E 59202020 20303030 30303036	00NY 0000006
0010:	38592020 20202020 20202020 20202020	8Y
0020:	20202020 20202020 20202020 20202020	
0030:	20202020 20202020 20202020 20202020	
0040:	20202020 20202020 20202020 20202020	
0050:	20202020 20202020 20202020 20204E00	N.
0060:	00000000 00000000 00000000 00000000	.....
0070:	00000000 00000000 00000000 00000000	.....
0080:	00000000 00000000 00000000 00000000	.....
0090:	00000000 00000000 00000000 00000000	.....
00A0:	00000000 00000000 00000000 00002030	..... 0
00B0:	30303030 30202020 20202020 20202020	00000
00C0:	00000000 00000000 00000000 00000000	.....

このビューでは編集またはマップのいずれも実行できません。主にトラブルシューティングやデバッグのツールとして設計されています。

## コピーブック固有のコンテキストメニュー項目

コピーブックのネイティブ環境ペインでマウスを右クリックした場合に表示されるコンテキストメニューには、次に説明するとおり、さまざまなコピーブック固有のコマンドが含まれています。

ネイティブペインメニュー	説明
Decision	コピーブックで REDEFINE ステートメントが選択されると有効になります。[Decision] をクリックすると、再定義されたデータ記述子の使用を決定する式を入力するオプションがダイアログボックスに表示されます。
Repeat	コピーブックで OCCURS ステートメントが選択されると有効になります。[Repeat] をクリックすると、Repeat アクションのターゲットを指定する情報を入力するオプションがダイアログボックスに表示されます。
Map	入力または出力ペインでアクティブになります。ステートメントを選択して [Map] をクリックすると、情報を入力するオプションがダイアログボックスに表示されます。
View - Copybook	ネイティブ環境ペインでアクティブになります。ステートメントを選択して [View/Copybook] をクリックすると、ペインにコピーブックの形式が反映されます。

ネイティブペインメニュー	説明
View - Copybook and Data	ネイティブ環境ペインでアクティブになります。ステートメントを選択して [View/Copybook and Data] をクリックすると、ペインにコピーブック、およびコピーブックにマップされたデータまたは実行中のプログラムから出力として配置されたデータが反映されます。
View Hex Data	ネイティブ環境ペインでアクティブになります。ステートメントを選択して [View/Hex Data] をクリックすると、ペインにコピーブック、およびコピーブックにマップされたデータまたは実行中のプログラムから出力として配置されたデータが 16 進形式で反映されます。
Select Occurrences	コピーブックで OCCURS ステートメントが選択されると有効になります。OCCURS 句の各オカレンスは表示されないため、[Select Occurrences] をクリックすると、データを表示するオカレンスを選択するオプションがダイアログボックスに表示されます。番号を入力します。 <b>注記:</b> 配列またはデータ構造、あるいはその両方は、0 から 5 までの番号が付いています。
Expand Tree	選択したデータ記述子の下にあるすべてのコピーブックノードを表示します。
Collapse Tree	選択したデータ記述子の下にあるコピーブックノードを非表示にします。
Copy	View/Hex Data 形式の場合に有効になります。テキストのブロックを選択して、コピー、貼り付けができます。
Print Copybook	コピーブックを印刷できます。
Find	すべてのデータビュー内にあるコピーブックを検索できます。
Find Next	すべてのデータビューで次の項目を検索できます。

## コピーブックと DOM 間のデータのマップ

ネイティブ環境ペインでコピーブックを表示すると、Hex Data ビュー以外のすべてのビューでドラッグアンドドロップにより XML DOM 要素をコピーブックにマップできます (その逆も可能です)。コピーブックへのマップは、通常 Send Message アクションの一部として入力 DOM から実行します。コピーブックからのマップは、通常 Browse または Receive アクションのターゲットとして出力 DOM を使用して実行します。

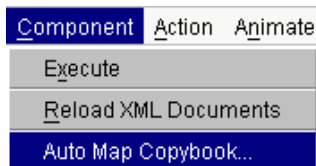
**注記：** JMS Connect では、ドラッグアンドドロップを実行するたびに、操作の一部としてある特定の状況が確認されます。特定の操作が禁止されている場合、「forbidden drag operation」記号が表示され、コンポーネントエディタのステータス行にエラーメッセージが表示されます。マウスボタンを放しても「ドロップ」は実行されません。

DOM とコピーブックフィールドの間でのマップは、メインメニューバーから [Action]、[New Action] メニューの順に選択して [Map...] コマンドを使用するか、コンテキストメニューの [Map...] コマンドを使用しても実行できます。コンテキストメニューにアクセスするには、DOM またはコピーブックの任意の要素をマウスで右クリックします。コンテキストメニューを使用する利点は、クリックした DOM またはコピーブックの要素に関する適当な Xpath 情報が、[Map] ダイアログボックスにすでに入力された状態で含まれることです。

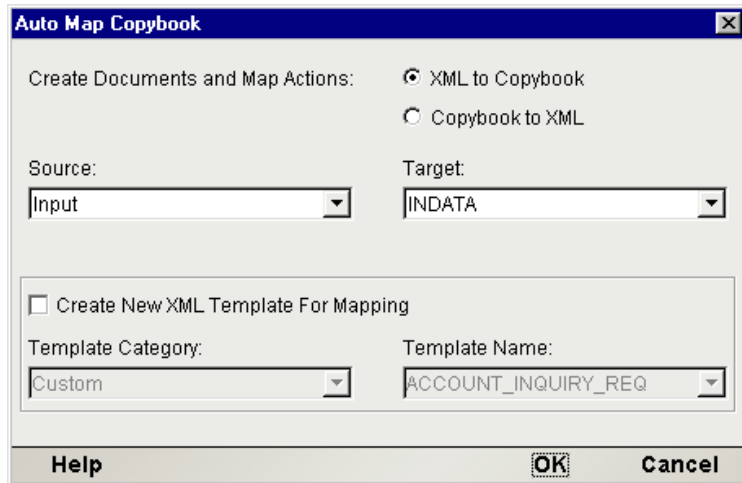
また、Function アクションの代わりに ECMAScript を使用して、プログラムによりマップを実行することもできます。

### コピーブックの自動マップ

すでに説明したマップ方法を使用する場合、キーボードまたはマウス、あるいはその両方でさまざまな操作を実行して、DOM 要素とコピーブックフィールドの間に必要なマップを作成することになります。コピーブックの行が比較的長い場合、このような操作はすぐに冗漫な作業になります。多数のマップを簡単に作成するために、JMS Connect にはメインメニューバーの [Component] メニューに [Auto Map Copybook] コマンドがあります。



このコマンドを選択すると、次のダイアログボックスが表示されます。



このダイアログボックスを使用すると、DOM とコピーブックの間でいずれの方向にも簡単かつ迅速にマップをまとめて作成できます。オプションで、コピーブックに基づいて新しい XML テンプレートドキュメントを作成し、再び使用する場合のために永久的に保存することもできます。

**注記：** [Auto Map Copybook] ダイアログボックスは、コピーブックがネイティブ環境ペインに表示されている場合にのみ表示されます (したがって、コピーブックに関連するメッセージアクションがコンポーネントエディタのアクションリストで選択されている必要があります)。

➤ **XML 要素のマップをまとめてコピーブックフィールドに作成する (またはその逆)**

- 1 [Auto Map Copybook] ダイアログボックスの右上の領域で、適当なラジオボタンを選択します ( [XML to Copybook] または [Copybook to XML] )。
- 2 [Source] および [Target] の 2 つのプルダウンメニューから必要なドキュメントを選択します。
- 3 [OK] をクリックします。コンポーネントエディタウィンドウが表示され、アクションモデルに新しい Map アクションが表示されます。

**注記：** 作成された Map アクションリストを確認し、プロジェクトに不要なアクションは削除します。

## ➤ コピーブックに基づいて新しいXML テンプレートドキュメントを作成する

- 1 [Auto Map Copybook] ダイアログボックスの右上の領域で、適当なラジオボタンを選択します ( [XML to Copybook] または [Copybook to XML] )。 [XML to Copybook] を選択すると、入力ドキュメントおよび Map アクションを作成して、データをコピーブックに移動することになります。 [Copybook to XML] を選択すると、出力ドキュメントおよび Map アクションを作成して、ホストプログラムの出力データをコピーブックから出力ドキュメントに移動することになります。
- 2 [Source] および [Target] の2つのプルダウンメニューから、必要なドキュメントを選択します。
- 3 ダイアログボックスの下の領域で、[Create New XML Template for Mapping] チェックボックスをオンにします。このチェックボックスを選択すると、すでに説明したバッチによるマップ操作に加えて、新しいXML テンプレートドキュメントを作成することになります。チェックボックスをオンにすると、ダイアログボックスの一番下にある [Template Category] および [Template Name] の項目がアクティブになります。
- 4 デフォルトのカテゴリを使用しない場合は [Template Category] を選択するか、既存のエントリに新しいテンプレートカテゴリ名を上書きします。
- 5 [Template Name] のリストからXML テンプレート名を選択するか、既存のエントリに新しいテンプレート名を上書きします。
- 6 [OK] をクリックします。コンポーネントエディタウィンドウに、作成された入力テンプレートが表示されます ( また、新しいテンプレートドキュメントはディスク上でプロジェクトのメインフォルダの下にある **xmlcategories** フォルダ内、新しいテンプレートカテゴリは Composer のメインウィンドウのナビゲーションペインにおけるXML テンプレートの下に表示されます)。

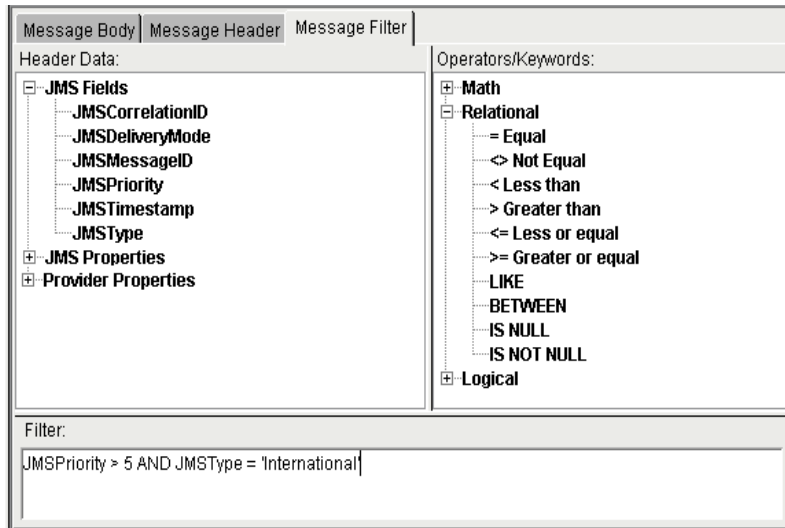
## メッセージフィルタ (セレクト) の操作

JMS 標準では、ユーザが決定した条件に基づきメッセージフィルタ (メッセージの選択的取得) を実行できます。フィルタは JMS メッセージセレクトにより実行されます。JMS メッセージセレクトは、True (真) または False (偽) について評価するステートメント (SQL のような構文) を含む文字列です。セレクトステートメントは、通常1つまたは複数の JMS メッセージヘッダフィールドまたはカスタムプロパティ、あるいはその両方を参照します (メッセージセレクトによりメッセージ本文の値を参照することはできません)。メッセージヘッダに表示されているデータに基づき、メッセージは選択されて取得されるか、選択されません。



セレクタの設定は、JMS Connect ではネイティブ環境ペインで実行します。(すべてのメッセージタイプに対する) Browse Message アクションおよび Receive Message アクションにより [Message Filter] タブが [Message Body]、[Message Header] タブとともにネイティブ環境ペインに表示されます。[Message Filter] タブには、セレクタ式を入力できるテキスト編集フィールドが表示されます。式は手動で入力することも、タブの上の領域でリストからエントリをダブルクリックして式全体または一部を構築することもできます。

例として、次のメッセージフィルタを使用します。



この例では、セレクタ式により JMSPriority が 5 より大、かつ JMSType が「International」という文字列と同一であるという条件を要求します。タブの上の領域で、[Header Data] にあるツリーの JMS Fields ノードを展開すると、使用可能なヘッダフィールドが表示されます (10 種類ある JMS 定義のヘッダフィールドすべてをフィルタに使用できるわけではありません。次の「フィルタの制約」を参照してください)。JMSType を選択します。ダブルクリックすると [Filter] フィールドに「JMSType」と表示され、カーソルの位置が移動します。同様に、右上にある [Operators/Keywords] のリストから演算子およびキーワードを「選択」すると、タイプすることなく式 (または式の一部) を作成できます。

例に表示されているフィルタ式を使用すると、Browse Messages アクションにより JMSType が「International」であるメッセージのうち、優先度が 5 以上のものだけがキューから取得されます。その他すべてのメッセージは無視されます。

Receive Message アクションでは、フィルタを使用するとキューから削除される条件を満たす「有効な最初のメッセージ」が選択されます。

その他のセレクタ式の例は、付録 B に記載されています。

**注記：** セレクタ式では、コロン文字 (:) と \ 記号をともに使用しないでください。exteNd では、SQL ステートメントおよび JMS セレクタにおけるコロンをマーカとして扱うため、( コロンの後に ) 評価可能なスクリプト式が存在することを示します。

## フィルタの制約

JMS では、フィルタに使用できるヘッダフィールドについて制約があります。セレクタステートメントによりヘッダフィールドが参照できる場所は、次のフィールドに限られています。

- ◆ JMSCorrelationID
- ◆ JMSDeliveryMode (Integer)
- ◆ JMSMessageID
- ◆ JMSPriority (Integer)
- ◆ JMSTimestamp (Long)
- ◆ JMSType

対応しないデータタイプを比較すると、ステートメントにより常に **False** ( 偽 ) が返されるため、セレクタステートメントではデータタイプが重要です。前のリストで、Integer 型である JMSPriority、Long 型である JMSTimestamp を除きすべてのフィールドは String 型です。

**注記：** 通常、JMSDeliveryMode も Integer 型ですが、セレクタコンテキストでは String 型の値「PERSISTENT」または「NON\_PERSISTENT」を持ちます。

カスタムなアプリケーション定義のプロパティフィールドは、セレクタステートメントの基礎として使用できます。ただし、セレクタで存在しないプロパティが参照された場合、操作の値は不明になります。SQL セマンティックにより NULL 値は不明として扱われ、不明な値が存在する操作にはすべて不明な値が返されるためです。便利なことに、NULL 値を参照するセレクタでは **True** ( 真 ) または **False** ( 偽 ) が返されます。ただし、IS NULL または IS NOT NULL 演算子を使用して、不明な値がブール演算の結果に変換された場合のみです。

## 本文のタイプによるフィルタ

JMS には、BytesMessage、StreamMessage などメッセージ本文のタイプそのものによるフィルタ機能は備わっていないことに注意する必要があります。クライアントアプリケーションでこのような情報にアクセスする必要がある場合、送信アプリケーションにより適当な本文のタイプがカスタムプロパティまたはヘッダフィールドに表示されます。JMS の受信アプリケーションでは、メッセージ本文のタイプを認識できる方法はありません。

しかし、JMS Connect に着信メッセージ本文の処理方法を伝えるために、Receive Message (または Browse Messages) のセットアップダイアログボックスの [Message Body] タブにある [Message Type] で何らかの値を選択する必要があります (70 ページの図を参照)。ただし、今説明したように、JMS アプリケーションでは本文のタイプによりメッセージをフィルタする方法はないため、Receive Message のセットアップダイアログボックスに指定された値はフィルタには使用されません。

**注記：** メッセージの適切な本文のタイプが受信されるように、アプリケーションを設定することが重要です。メッセージ作成アプリケーションでは、一般的にメッセージを受信できるように特に設計されている対応アプリケーションに対してのみメッセージが送信されるため、通常はこの点について問題ありません。しかし、[Receive Message] ダイアログボックスで選択するタイプと互換性がないタイプのメッセージを受信した場合は例外が発生します。

アプリケーションにより、本文のタイプがさまざまなメッセージを含むキューからメッセージを受信または参照する場合、メッセージプロパティまたはヘッダフィールドに本文のタイプに関する情報を表示し、受信アプリケーションが使用できるメッセージフィルタ (セレクトステートメント) を設計して、それらに対応するメッセージかどうかを区別できるようにすることをお勧めします。

## リクエスト - 応答メッセージ

往復リクエスト - 応答メッセージングは、メッセージングを使用するアプリケーションにおける一般的なシナリオです。一般的な使用例は次のとおりです。

- ◆ アプリケーションでキューまたはトピックにメッセージを送信し、即時に返信を受信することが予想される場合。たとえば、製造元が特定の部品を購入する場合、トピックにメッセージを公表することによって品目への入札を要求するとします。そのトピックのリスナは、確認により (または実際の入札により) 即時に応答できるよう設定します。
- ◆ アプリケーションで、応答の受信を予想せずにキューまたはトピックにメッセージを送信する場合。ただしエラー通知のために、アプリケーションにより発信メッセージで ReplyTo の送信先は指定します (ReplyTo キューはまったく異なるドメインに存在し、単にエラーレポートを蓄積するように設定します)。したがって、アプリケーションによりメッセージはキューに送信されますが、エラーレポートが別のキューに送信するように要求されます。
- ◆ アプリケーションが、キューまたはトピックでリッスンする JMS サービスで、受信メッセージを処理して返信するように設計されている場合。

どの場合でも、送信者は送信メッセージの *JMSReplyTo* ヘッダフィールドにキューまたはトピック名を入力する必要があります。入力には、[Send Message] ダイアログボックスの [Send Options] タブを使用します。前の章の「Send Message アクション」で、「返信用アドレス」節を参照してください。

アプリケーションがクエリに応答している場合、最後に受信されたメッセージの *JMSReplyTo* フィールドに存在したキューまたはトピックを自動的に使用するよう、**Send Message** アクションを設定できます。**[Send Message]** ダイアログボックスの **[Send Options]** タブには、この目的のために **[Use Prior Message ReplyTo Field]** ラジオボタンがあります ( 前の章の「**Send Message** アクション」で、「送信先キュー / トピック」を参照してください)。

## 一時キュー

アプリケーションで即時の応答が要求されていて、返信が受信されるまで ( または指定したタイムアウト時間になるまで ) ブロックしている場合、その返信を受信するための一時キューを作成すると便利です。一時キューを使用する利点は次のとおりです。

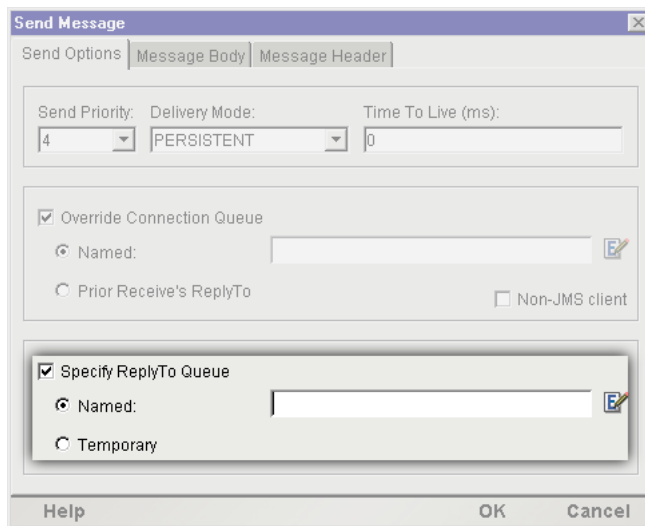
- ◆ 他のプロセスから孤立できます ( メッセージを受信するクライアント以外には、一時キューは認識されません )。メッセージフィルタの必要性を最小限または除外することにより、アプリケーションの設計を簡素化できます。
- ◆ 一時キューはすぐに作成でき、目的達成の後即時に破棄されるため、リソースおよび管理の面から見て効率的です。

メッセージの応答に使用する一時キューを指定するには、**[Send Message]** ダイアログボックスの **[Send Options]** タブにある **[Use Temporary]** ラジオボタンをオンにします ( 前の章の「**Send Message** アクション」で「返信用アドレス」を参照してください) 。一時キューは自動的に作成され、発信メッセージの *JMSReplyTo* フィールドにその名前が自動的に配置されます。このキューは、コンポーネントの有効期限にわたり存在します。

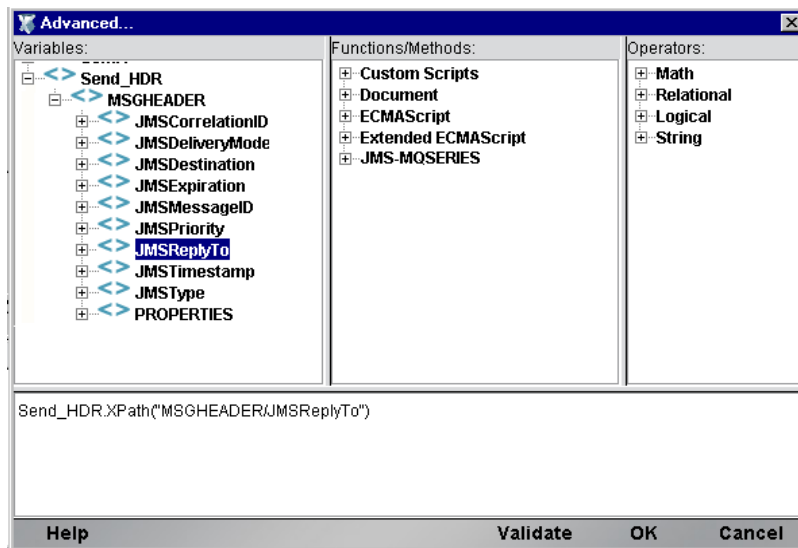
## 複数の一時キュー

1 つのコンポーネントの有効期限内に複数の送信メッセージを送信する場合、複数の一時キューを作成します。デフォルトでは、**JMS Connect** により送信メッセージが作成されるたびに固有な一時キューが作成されます ( **[Use Temporary]** ラジオボタンがオンに設定されている場合 )。複数の送信メッセージに対し **ReplyTo** フィールドで同じ一時キューを指定する場合、次の方法を実行します。

- 1** 通常の方法で最初の **Send Message** アクションを作成し、**[Send Options]** タブの **[Use Temporary]** ラジオボタンをオンにします ( 前の章の「**Send Message** アクション」で、「返信用アドレス」を参照してください) 。
- 2** 次の **Send Message** アクションで、**[Temporary]** ラジオボタンをオフにします ( オンにすると新しい固有の一時キューが使用されます )。その代わりに、**[Named]** ラジオボタンをオンにします。



- 3 テキストフィールドの右側にある [Expression] アイコンをクリックします。  
[Expression Builder] ダイアログボックスが表示されます。



- 4 左上の [Variable] ツリーで、最初の送信メッセージの送信ヘッダノード (デフォルト名 **Send\_HDR**) を開きます。**MSGHEADER** ノードを展開して、JMSヘッダフィールド名をすべて表示します。

- 5 リストの **JMSReplyTo** エントリをダブルクリックします。下の編集フィールドに、ECMAScript 式が表示されます。
- 6 **[OK]** をクリックして、**[Send Message]** ダイアログボックスに戻ります。
- 7 このメッセージに指定が必要なその他のメッセージパラメータをすべて設定します。**[OK]** をクリックして **[Send Message]** ダイアログボックスを閉じます。
- 8 コンポーネントで最初の一時キューを使用する、続く **Send Message** アクションに対して手順 2 から手順 7 を繰り返します。

## ECMAScript と JMS Connect

JMS Connect では、JMS 関連の多くの ECMAScript が表示されます。固有の Function および Map アクションで ECMAScript を使用すると、JMS コンポーネントおよびサービスの機能性を拡張できます。ほとんどの場合、拡張機能はメッセージ本文のコンテンツを操作するための「set」および「get」メソッドで構成されます。

JMS 関連の ECMAScript 拡張機能にアクセスするには、**[Expression Builder]** ダイアログボックスのリストを使用します (Expression Builder には、Map アクションおよび Function アクションのダイアログボックスからアクセスできます。次の例を参照してください)。Expression Builder のリストに表示されるメソッドは、状況に応じて異なります。たとえば、コピーブックメッセージを操作している場合、コピーブック関連の操作に対応するメソッドがツリーに表示されます。したがって、Message アクションが Bytes Message に関係がある場合、表示される ECMAScript メソッドは JMS Bytes Message タイプと関連します。

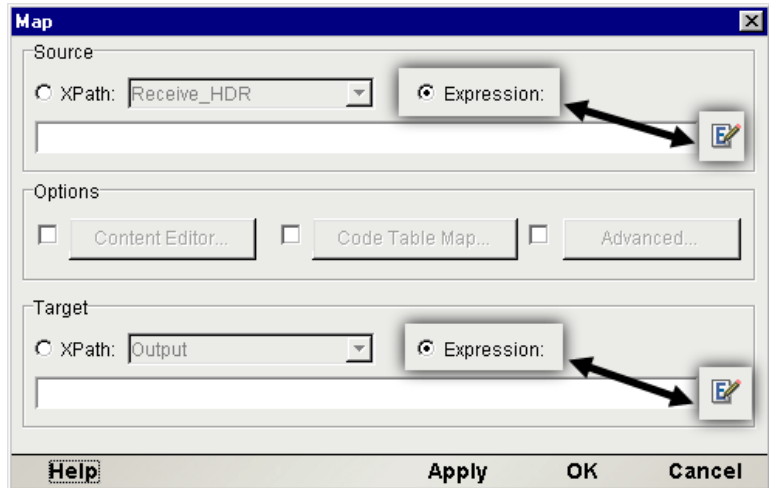
次の例では、ECMAScript を使用して JMS Bytes Message をコンテンツに添付する方法を説明します。その他のメッセージタイプと異なり、Bytes Message タイプにはマップするためのインターフェースがありません。したがって、ECMAScript は Bytes Message の本文をコンテンツに添付する唯一の方法です。

**注記：** JMS Object Message にもマップするためのユーザインターフェースはありません。ECMAScript を使用して、Object Message にコンテンツを添付する必要があります (通常、Java コードを呼び出して Object Message に関連付けられている Serializable オブジェクトを取得する場合、ECMAScript のパッケージ機能を信頼できます。『exteNd Composer ユーザガイド』の第 10 章を参照してください)。

### ➤ Expression Editor で JMS 関連の ECMAScript 拡張機能を使用する

- 1 **Send Message** アクション (またはその他の JMS アクション) を作成します。使用するメッセージのコンテンツのタイプに対応する本文のタイプを選択します。この例では Bytes Message タイプを使用しますが、ここで説明されている原理は、本文のすべてのタイプに適用できます。

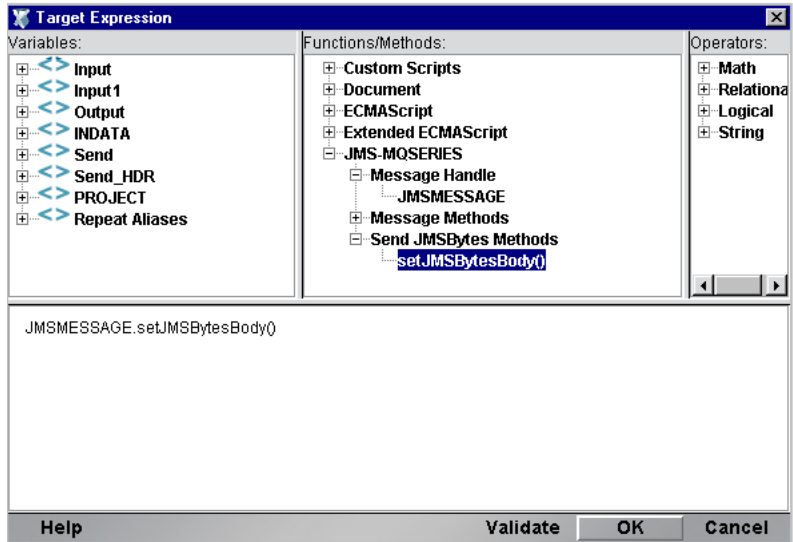
- 2 選択した Message アクションで、新しい Map アクションを作成します。次の例では、メインメニューから [Action]、[New Action]、[Map] の順に選択して新しい Map アクションを作成します。



- 3 [Source] および [Target] 領域の両方で [Expression] ラジオボタンをオンにします。
- 4 メッセージ本文のデータのソースを表すソース式を入力するか、ソースのテキスト編集領域の横にある [Expression Editor] アイコンをクリックします。[Expression Editor] ウィンドウが表示されます。

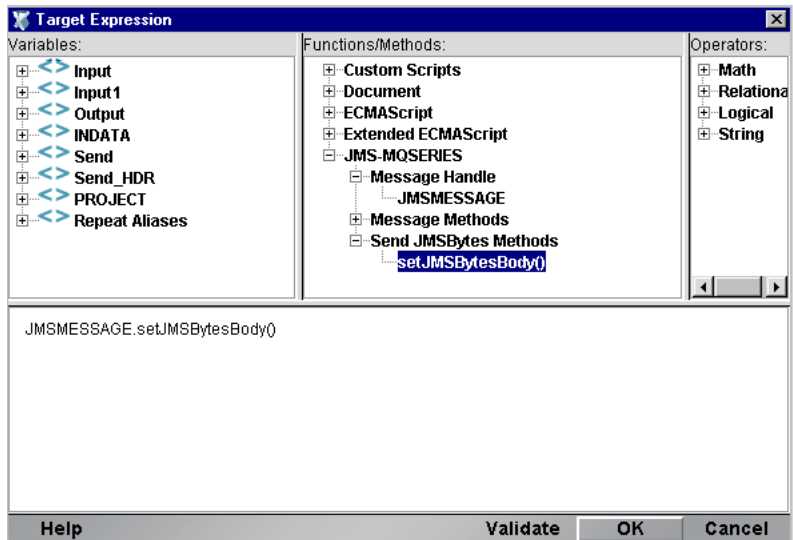
**注記：** この例では入力 DOM の PRODUCT/NAME Xpath からデータをマップしますが、他のソースからマップすることもできます。たとえば、File コンストラクタおよび readAll() メソッドを使用すると、ディスク上のファイルからデータを取得できます (「拡張 ECMAScript」リストには、ファイルの入出力およびその他のメソッドが含まれています)。

- 5 ターゲット式を入力するか、Expression Editor を使用して作成します。ターゲットのテキスト編集領域の横にある [Expression Editor] アイコンをクリックします。[Expression Editor] ウィンドウが表示されます。



- 6 Expression Editor の中上の部分で、**JMS-MQSERIES** 項目をクリックしてツリーを展開すると、Message Handle、Message Methods、および Send JMSBytes Methods とラベルが付いたノードが表示されます。

**注記：** ツリーに JMS 関連のノードが表示されない場合、Map または Function アクションが Message アクションと関連付けられていないことが考えられます。Map または Function アクションを作成する前に、Message アクションが選択されていること（または「Before Send Maps」などのアクションリストの関連セクション）を確認してください。





- 7 JMS-MQSERIES の下にあるツリーのさまざまなノードを展開します。**JMSMESSAGE**、**getJMSMessage()**、および **setJMSBytesBody()** のような名前の端末ノードが表示されます。これらの名前（または他のリストウィンドウにあるリーフノード、あるいはその両方）を必要に応じてダブルクリックして、目的の ECMAScript 式を作成します。ダブルクリックすると、ウィンドウ内のテキスト編集部分に対応するラベルが表示されます。
- 8 **[OK]** をクリックしてすべてのダイアログボックスを閉じます。Map アクションがコンポーネントのアクションリストに表示されます。

ECMAScript を使用してコンテンツをメッセージに添付するには、次のターゲット式を使用します。

```
JMSMESSAGE.setJMSBytesBody( );
```

引数は必要ありません。この「setter」メソッドは、Map アクションのソース部分からデータを取得して適当な形式に変換し (Bytes Message の場合はバイト配列)、メッセージの本文に添付します。対応する「getter」メソッドも同様の操作を行いますが、引数が必要です。呼び出し規則については、後で設定します。

## ECMAScript メソッドの概要

次に、使用可能な JMS 関連の ECMAScript 拡張機能および使用方法について説明します。メソッドのほとんどは、JMSMESSAGE ハンドルで呼び出されます。ただし、Copybook ハンドルで呼び出される Copybook メソッドおよび CopybookField オブジェクトで呼び出される CopybookField メソッドは例外です。

```
Message getJMSMessage()
```

このメソッドは、JMSMESSAGE ハンドルで呼び出されると JMS Message オブジェクトを返します。JMS Message 本文の特定のタイプを操作するには、次のように返されるメッセージを適当なタイプにキャストします。

```
TextMessage lMsg = (TextMessage) getJMSMessage()
```

```
String getJMSMsgBody()
```

JMSMESSAGE でこのメソッドを呼び出すと、メッセージの本文を文字列として取得できます。

```
String getJMSMsgType()
```

JMSMESSAGE でこのメソッドを呼び出すと、メッセージ本文のタイプを文字列として取得できます。返される値は、JMSText、JMSMap、JMSObject、または JMSStream のいずれかです。特別な XML または Copybook タイプは、JMS 定義のタイプでないため返されません。

```
CopybookField getField(String cobolDataDesc)
```

例:

次のコピーブックがあるとします。

```
COMMAREA
```

```
05 INDATA
```

```
10PARTID
```

```
05 OUTDATA
```

```
10PARTID
```

最初の PARTID (INDATA の下) は参照しますが、OUTDATA の下にある PARTID は必要ありません。重複する名前の問題を解決するには、次のように親の cobolDataDesc を参照します (MYCOPYBOOK の Copybook ハンドルであることが前提です)。

```
MYCOPYBOOK.getField("PARTID IN INDATA")
```

戻された CopybookField オブジェクトには、toString() および setValue() の 2 つのメソッドがあります。

```
void setValue(Object aValue)
```

このメソッドは、CopybookField オブジェクトの値を設定します。

```
String toString()
```

CopybookField オブジェクトに設定された値を返します。

```
String getJMSBytesBody(int aiBufSize)
```

JMS BytesMessage オブジェクトの本文の値を文字列として取得します。aiBufSize パラメータは本文のサイズ (バイト単位) です。文字列を返します。

```
setJMSBytesBody()
```

JMS BytesMessage オブジェクトの本文を設定します。

```
String getJMSMapField(String asName, String asType)
```

JMS MapMessage オブジェクトの本文フィールドの値を設定します。文字列を返します。

`setJMSMapField(String asName, String asType)`

**JMS MapMessage** オブジェクトの本文フィールドの名前およびタイプを設定します。

`Serializable getJMSObjectBody()`

**(Receive Message アクションの後 )JMS ObjectMessage** オブジェクトの本文の値を取得します。**Serializable** オブジェクトを返します。

`void setJMSMsgProperty(String asName, String asType, String asValue)`

特定の名称の **JMS** ヘッダを特定の値に設定します。

`setJMSObjectBody(Serializable aObject)`

**JMS ObjectMessage** オブジェクトの本文を設定します。

`String getJMSStreamField(String asName, String asType)`

**JMS StreamMessage** オブジェクトの本文フィールドの値を取得します。文字列を返します。

`setJMSStreamField(String asName, String asType)`

このメソッドは、**JMS StreamMessage** オブジェクトの本文フィールドの値を設定します。

`String getJMSTextBody()`

**JMS TextMessage** オブジェクトの本文の値を取得します。文字列を返します。

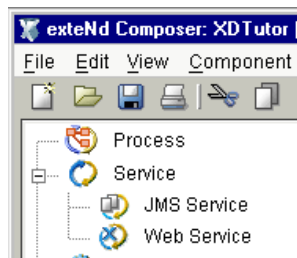


# 6

## JMS サービス

JMS は *MessageListener* オブジェクトと呼ばれるメカニズムを定義します。*MessageListener* オブジェクトは、メッセージがキューまたはトピックに公開されると、メッセージコンシューマに非同期に通知します。これにより、受信するアプリケーションに、着信メッセージをイベントとして扱う機能が追加されます。アプリケーションが送り出されてメッセージをトピックから取得する必要はなく、メッセージは、キューマネージャまたはトピックマネージャによってアプリケーションに「押し出され」ます。

この機能を利用するために、JMS Connect で、「JMS サービス」と呼ばれる新しいタイプの *exteNd Composer xObject* が導入されています。JMS サービスは、Composer メインウィンドウの [Service] にカテゴリとして表示されます。



## JMS サービスについて

他の *exteNd* サービスと同様に、JMS サービスでは、外部コンポーネントを呼び出したり、XML Interchange アクション、Log アクション、Function アクションを実行したりすることができます(『*Composer ユーザガイド*』の「Creating a Service」を参照)。ただし、JMS サービスは、さまざまな点で他のサービスとは異なります。

- ◆ JMS サービスは、(キューまたはトピックの)「着信メッセージ」によってトリガされます。

- ◆ 着信メッセージが適切に処理されるようにするには、JMS サービスに Receive Message アクションを1つ(1つだけ)含める必要があります。

**注記:** exteNd のバージョン 2.7 以降では、Send Message アクションを JMS サービスに配置して、リスナが着信メッセージに直接返信することができます (別のコンポーネントを呼び出す必要はありません)。

通常、JMS コンポーネントは、JMS サービス内にパッケージ化する必要はありません。JMS サービスの際立った特徴は、その「コンテンツ」にあるのではなく、「トリガメカニズム」にあります。JMS サービスは、メッセージがキューまたはトピックに入るとトリガされるよう設計されています。HTTP サーブレットからトリガされるよう設計されているサービスは、JMS コンポーネントを使用する場合でも Web サービスとして作成される必要があります (最新バージョンの『Composer ユーザガイド』の「Creating Service」を参照)。

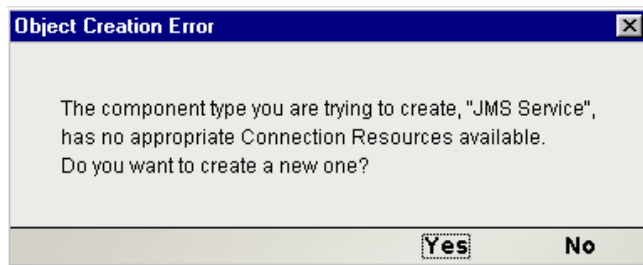
## 複数のリスナ

JMS Connector によって提供される便利な機能により、複数のリスナを持った JMS サービスを配備することができます。これにより、同じ JMS サービスの複数のインスタンスを同時に実行することが可能です。

複数のリスナサービスは、JMS サービスを設計して、通常使用しているものと同じ配備ウィザードを使用して配備するだけで作成できます。配備ウィザードの JMS サービスパネルには、[Count] フィールドがあります (「JMS サービスの配備」という節にある図を参照)。[Count] フィールドでは、配備するリスナの数を指定できます。配備すると、リスナは HTML ベースのコンソールウィンドウから管理できます (「配備された JMS サービスをどのように管理しますか?」の説明を参照)。

## JMS サービスの作成

JMS サービスは、他のサービスと同じように作成されます。ただし、JMS サービスを作成する前に、サービスがメッセージを受信する元のキューまたはトピックに対して JMS 接続リソースを作成しておく必要があります (詳細については、「JMS 接続リソースの作成」を参照してください)。この手順を省略すると、次のようなエラーメッセージが表示されます。

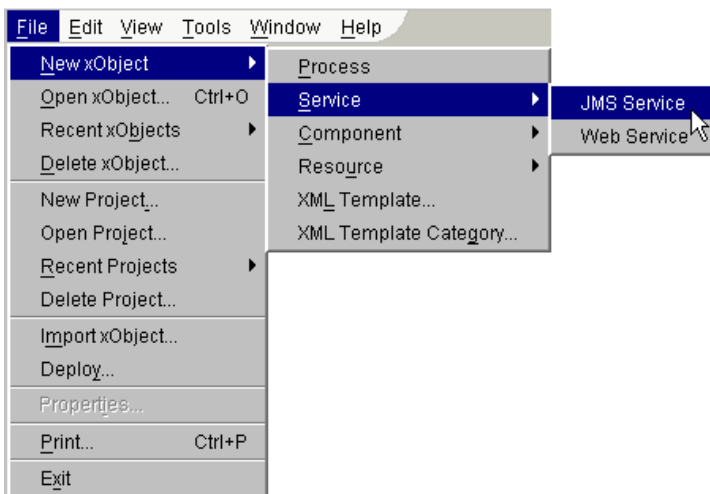


新しい JMS 接続リソースをその場で作成する場合は、このダイアログボックスで [Yes] をクリックします。

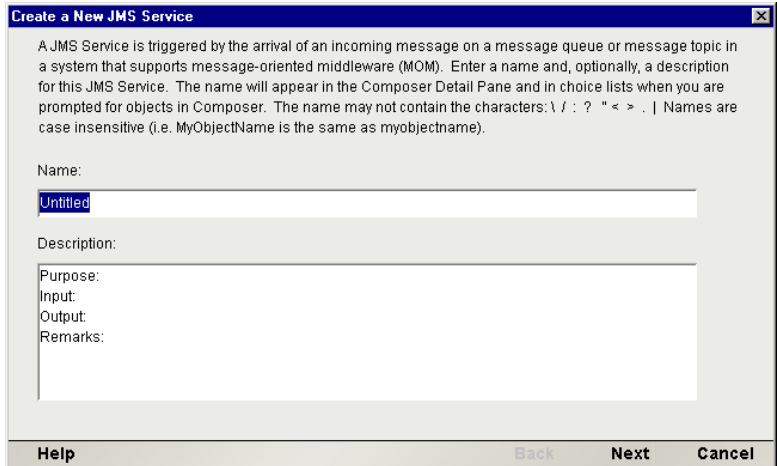
次の説明では、JMS サービスで使用する接続リソースがすでに作成されていることを前提にしています。

### ➤ 新しい JMS サービスを作成する

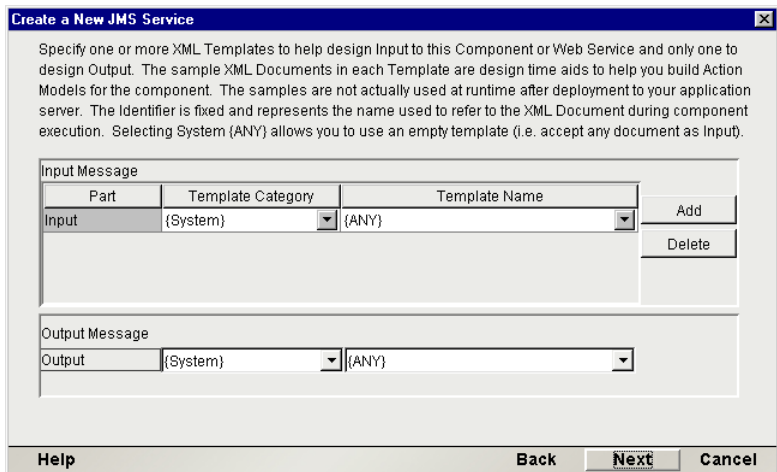
- 1 Composer の [File] メニューから、[New xObject]、[Service]、[JMS Service] の順に選択します。



- 2 「Create a New JMS Service Component」ウィザードの最初のパネルで、サービスの「名前」と「説明」(オプション)を入力します。

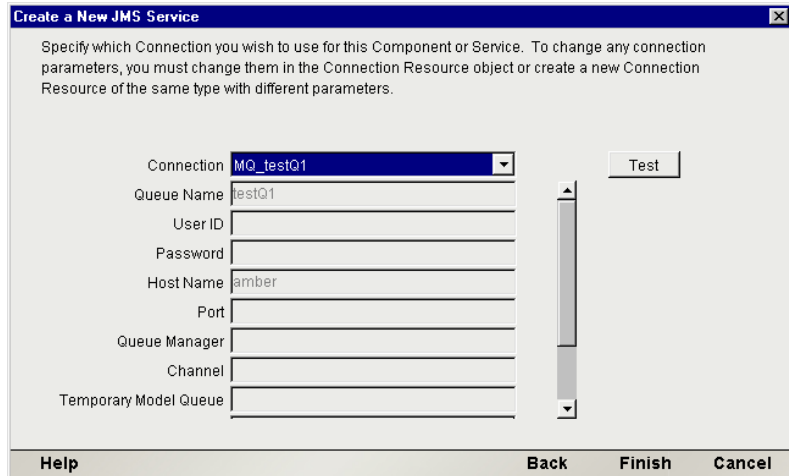


- 3 [Next] をクリックして、テンプレートパネルを表示します。



- 4 プルダウンメニューを使用して、入力テンプレートおよび出力テンプレートを必要に応じて指定します。デフォルトのカテゴリと異なる場合は、[Template Category] を選択し、[Template Name] で使用できるプルダウンリストからテンプレートを選択します。入力 XML テンプレートをさらに追加するには、[Add] をクリックして、これらの手順を繰り返します。入力テンプレートを削除するには、エントリを選択して [Delete] をクリックします。
- 5 必要な場合は、Output に対して XML テンプレートを選択します。
- 6 [Next] をクリックして、ウィザードの最後のパネルを表示します。

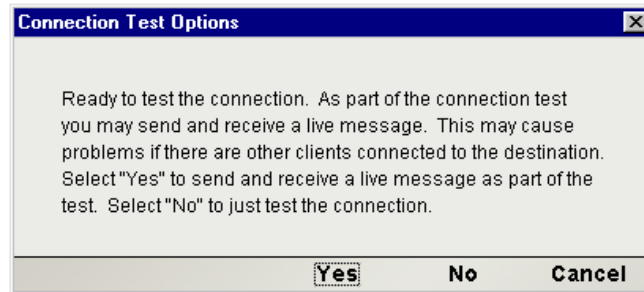




- 7 プルダウンメニューに表示されている使用可能なキューまたはトピック、あるいはその両方から「接続」を選択します。

**注記：** [Connection] メニューの下にあるフィールドはグレー表示 (無効) になります。これらのフィールドに表示された情報を変更する必要がある場合は、(このダイアログボックスを閉じた後) Composer のメインウィンドウから適切な接続リソースを開くことによって変更できます。

- 8 [Test] をクリックして、正常に接続されるかどうかを確認します。[Test] をクリックすると、[Test Options] ダイアログボックスが表示されます。



- 9 [Test Options] ダイアログボックスでは、接続の整合性テストの一環としてライブメッセージを送信するかどうか尋ねられます。[Yes] ボタンをクリックすると、接続を確立しているキューまたはトピックに対して、Composer から (*TextMessage* タイプで固有の CorrelationID を持った) メッセージが送信されます。

**注記：** 運用環境 (つまり、多数のリスナを持つ可能性があるライブキューを使用) では、その環境において既存アプリケーションが悪影響を受けないことが確かでない限り、テストメッセージを送信しないよう注意してください。

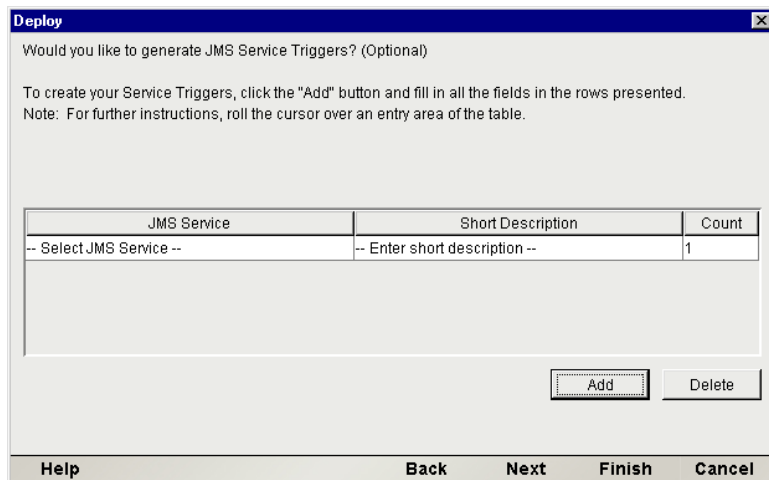
必要な接続オブジェクトを作成してもテストメッセージは送信しない場合は、[No] をクリックします。

- 10 [Finish] をクリックします。JMS サービスコンポーネントが作成され、サービスエディタウィンドウが表示されます。

## JMS サービスの配備

JMS サービスオブジェクトを含むプロジェクトは、他のプロジェクトと同じ方法で、同じ配備ウィザードを使用して配備されます。プロジェクトの配備方法の段階的な手順については、『*exteNd Composer Enterprise Server ユーザガイド*』の「Deploying a Project」という章を参照してください。

JMS Connect のユーザに対して配備ウィザードが異なっているのは、このウィザードには次の特別なパネルが含まれているという点においてのみです。



このパネルによって、各 JMS サービスの *MessageListener* オブジェクトが作成されます。このオブジェクトは、各サービスと適切なリスナオブジェクトの `onMessage()` ハンドラを自動的に関連付けて配備します。このとき、リスナは該当する JMS サービスコンポーネントで指定された JMS のキュー接続またはトピック接続に登録されます。

### ➤ JMS サービストリガを作成する

- 1 前のパネルで [Add] をクリックします。新しい列が挿入ペインに表示されます。

- 2 「-- Select JMS Service --」というスペースをクリックします。プロジェクトで使用できるすべての JMS サービスオブジェクトの名前を一覧表示したプルダウンメニューが表示されます。
- 3 JMS サービスオブジェクトを選択して、マウスボタンを放します。
- 4 「Short Description」の下に、この JMS サービスに関連付けるプレーンテキストの記述情報を入力します。  
**注記：** このフィールドを空白にはできません。
- 5 「Count」の下に、このサービスに関連付けるリスナの数を入力します。デフォルトの値は、1 です。
- 6 配備する JMS サービスがすべて追加されるまでこの手順を繰り返します。
- 7 必要なだけ追加すると、[Next] または [Finish] をクリックします。

サーバにプロジェクトの各 JMS サービスのリスナオブジェクトが追加されました。Novell アプリケーションサーバを使用している場合は、サーバを再起動する必要はありません。配備が完了するとリスナオブジェクトがアクティブ状態になります。

## 配備された JMS サービスをどのように管理しますか？

JMS サービスを含むプロジェクトが配備されると、サーバを起動するたびに、さまざまなサービスの *MessageListener* オブジェクトがメッセージをリッスンします。これらのサービスを個別に起動または停止したり、サーバ全体からこれらのサービスを削除したりするためには、exteNdJMS サービスコンソールにアクセスする必要があります。このブラウザベースのコンソールによって、JMS サービスの一覧（および配備ウィザードで入力した記述情報）、各サービスのステータス（有効または無効）、受信されるメッセージの合計数（Count）、および他の管理情報を表示できます。また、（各サービスに 1 つずつ）[Start/Stop] ボタンおよび [Remove] ボタンが表示されます。

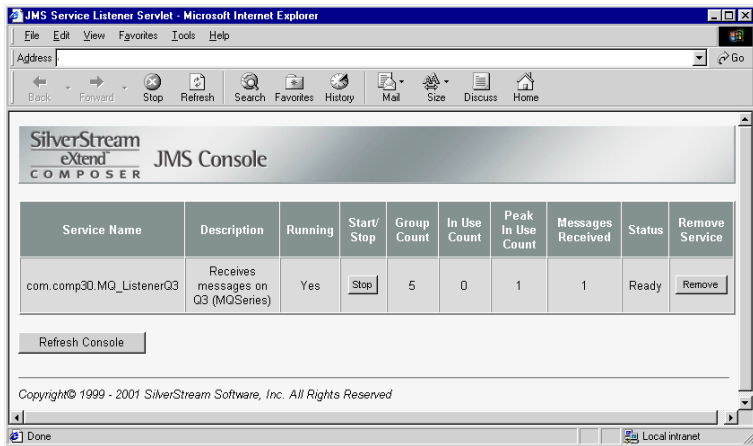
### ➤ exteNdJMS サービスコンソールにアクセスする

- 1 アプリケーションサーバが稼動していることを確認してください。
- 2 Web ブラウザを起動して移動します。

**http://*hostname*/extendComposer/jmsConsole**

[hostname] はサーバの名前（: ポート）に置き換えます。「localhost:80」と入力します。

- 3 配備された JMS サービスを一覧表示したコンソールウィンドウが表示されます。



- 4 JMS サービスを停止するには、適切な **[Stop]** ボタンをクリックします。ボタンが **[Start]** に変更されます。

**注記：** メッセージがサービスによって処理されているときに **[Stop]** コマンドを使用すると、サービスが実際に終了する時に遅延がおこることがあります。該当するサービスのコンソールの **[Running]** 欄が **[No]** になるまで、定期的に **[Refresh]** ボタンをクリックしてください。待ち時間はトラフィック条件およびベンダ固有の JMS 実装内容に依存するため、サービスを停止するときの正確な待ち時間は予測できません。pub/sub トピックの実行を保留する方法の詳細については、プロバイダのマニュアルを参照してください。

- 5 JMS サービスを完全に削除するには、適切な **[Remove]** ボタンをクリックします。

メッセージがキューまたはトピックで待機中の場合、( サービスを再開するために ) コンソールページ **[Start]** をクリックすると、サービスの `onMessage()` メソッドが即座に呼び出されます。ただし、コンソールの **[Count]** フィールド ( 通常は、現在までに受信したメッセージの合計を表示 ) は自動的に更新されません。**[Count]** を正しく表示するには、サービスを起動してからブラウザの **[Refresh]** ボタンをクリックします。

# A

## JMS 用語集

### BytesMessage

5つある JMS 定義メッセージタイプの 1つ。このタイプのメッセージの本文は、バイト配列で文字どおりに構成されるため、あらゆる種類のペイロードを表すことができます。

### CICS

Customer Information Control System。メインフレームのトランザクションを処理、監視するための IBM プロトコルです。

### DOM

Document Object Model。XML または HTML ファイルのコンテンツの階層を記述または表す業界標準の方法。

### Durable Subscriber

パブリッシュ/サブスクライブ設定（「パブリッシュ/サブスクライブ」参照）では、メッセージの受信者はオフラインの場合でもメッセージを受信できるよう登録できます。このようなサブスクライバは、受信者のステータスがどのようなセッションでも維持されるため「永続的」と呼ばれます。

### JMS

Java Messaging Service。Sun により開発されたメッセージサービス用 Java インタフェースで、メッセージ指向ミドルウェア (MOM) に関する業界標準のオブジェクトおよび動作を定義します。JMS 対応 MOM は、JMS で定義されたインタフェースを実装します。

### JMS プロバイダ

JMS インタフェースを実装する任意の MOM システム。

## JNDI

Java Naming and Directory Interface。Java プラットフォームに対する標準的な拡張機能で、Java アプリケーションに複数の命名およびディレクトリサービスに対する統一されたインタフェースを提供します。

## JTA

Java Transactions API。分散トランザクションを区切るための Java API です。

## MapMessage

5 つある JMS 定義メッセージタイプの 1 つ。Name/value の対で構成されています。キーは Java 文字列で、値は Java のプリミティブタイプです。

## MessageListener

アプリケーションがパブリッシュ / サブスクライブシステムで実装できる Java インタフェースで、アプリケーションにより着信メッセージの自動通知を受信できます。

## MOM

Message Oriented Middleware。企業メッセージングを実装するソフトウェアシステムです (IBM の MQSeries ソフトウェアなど)。

## NON\_PERSISTENT

2 つある JMS 定義の配信モードの 1 つ (もう 1 つは PERSISTENT です。次を参照してください)。最大で 1 回の配信が保証されます。メッセージはどの段階でも非揮発性のストレージには書き込まれないため、このモードを使用するとシステムの中断によりメッセージが失われる可能性があります。ただし、オーバーヘッドは、PERSISTENT モードよりもこのモードの方が低くなります。

## ObjectMessage

5 つある JMS 定義メッセージタイプの 1 つで、本文に連続した Java オブジェクトを含みます。

## PERSISTENT

2 つある JMS 定義の配信モードの 1 つ (もう 1 つは NON\_PERSISTENT です)。PERSISTENT モードを使用すると、メッセージは 1 度だけの配信が保証されます。メッセージは、非揮発性のストレージに書き込まれ、転送が失われる可能性を防止します。

## RPC

Remote Procedure Call。プログラムまたはプロシージャが、メインフレームまたはサーバの同期セッションからリモートで実行されるプロトコルです。

## SQL92

データベースのクエリに一般的に使用されている SQL (Structured Query Language) の実装で、JMS メッセージセクタ構文の基礎。

## StreamMessage

5 つある JMS 定義メッセージタイプの 1 つ。メッセージ本文が Java のプリミティブ値で構成されます。このタイプのメッセージ本文は、`readLong()`、`readString()` などのメソッドを使用して連続的に読み込まれます。

## TextMessage

5 つある JMS 定義のメッセージタイプの 1 つ。`TextMessage` の本文は `String` 型です。

## Time-To-Live

メッセージの有効期限。メッセージの有効期限は、メッセージが送信された時間プラス `Time-to-Live` の値に基づいて計算されます。

## 宛先

`javax.jms.Destination` を展開する JMS のキューおよびトピック。したがって、JMS の宛先はキューまたはトピックと同義です。宛先は管理機能で作成され、作成時の JNDI 名に指定されます。

## エンディアン

コンピュータメモリにバイトを格納する順序を記述するために使用される用語。ビッグエンディアンは、シーケンス内の最上位バイトが下位のメモリアドレスに最初に格納されます (リトルエンディアンはその反対になります)。Intel では伝統的にリトルエンディアンアーキテクチャが使用されていますが、他のほとんどのチップメーカーではビッグエンディアンアーキテクチャが使用されています。

## 管理オブジェクト

JMS では、*Destinations* および *ConnectionFactory*s という 2 つの管理オブジェクトを定義します。*Destinations* のような種類のオブジェクトにより、トピック名またはキューと物理リソースが関連付けられます。*ConnectionFactory*s により、クライアントが JMS プロバイダのサービスデーモンに接続するメソッドが表示されます。いずれのオブジェクトも管理制御下にあるリソースに関係します。JMS クライアントは、JNDI を使用するネームスペースで管理オブジェクトを参照することにより、オブジェクトを検索できます。

## キュー

JMS ベースのメッセージングシステムでは、メッセージはクライアントではなく、キューに送信されます。つまりキューはメッセージを処理するためにセットアップされた集積所です。送信者と受信者の間にキューを置くと、クライアントが応答できない場合でも、そのクライアントに送信されたメッセージを後の取得のためにキャッシュできます。通常、キューは管理上の目的で作成され、静的なリソースとしてメッセージクライアントに表示されます。

## コピーブック

個別に定義された COBOL データ記述子で構成される記録構造です。

## コミット

処理されるメッセージセッションで、セッションの `commit()` メソッドを呼び出すと、作成されたメッセージはすべて送信され (元に戻すことはできません)、使用されたメッセージは確認されます (したがってキューから削除されます)。トランザクションおよびロールバックも参照してください。

## セッション

セッションは、メッセージを作成および使用する軽量の JMS オブジェクトです。セッションは、確認されるまでメッセージを取得します。すべての送信および受信アクションは、セッションの範囲で実行されます。

## 接続

接続は、セッションを構築して管理するために必要な、サーバ側とクライアント側のオブジェクトの集合です。接続は、`QueueConnection` タイプまたは `TopicConnection` タイプのいずれかです。接続は、JNDI によりアクセス可能なオブジェクトから発生します。

## セレクトタ

ポイントツーポイントメッセージでは、クライアントはヘッダの内容に基づきセレクトタを使用してメッセージをフィルタします。セレクトタは基本的にヘッダまたはプロパティ値に関する条件的なステートメント (True または False について評価するステートメント) で、SQL92 と類似する構文で記述されます。

## データグラム

JMS 定義の用語ではありませんが、ダイアグラムという用語はメッセージングで頻繁に使用されます。一般的には、管理に関する通知など短いメッセージを表し、「実行後削除」の方法で送信されます (返信を予期していません)。

## トピック

パブリッシュ/サブスクライブメッセージング (前の「パブリッシュ/サブスクライブ」を参照) では、通常メッセージキューは「トピック」と呼ばれます。つまり、トピックはキューのことであり、キューとの違いは主に管理される方法にあります。トピックは一般的に多くのユーザにより共有され、コンテナの階層でノードを形成します (ただし、JMS の要件ではありません)。多くのユーザは 1 つまたは複数のトピックを「パブリッシュ」できます。反対に、トピックには多くの「サブスクライブ」が存在します。



## トランザクション

JMS メッセージセッションでは、1 つまたは複数の受信、送信、または参照アクションをトランザクションにグループ化でき、すべての操作は 1 つの単位として処理されます。トランザクションが正常に実行された場合、構成する操作すべてが正常に行われたこととなります。トランザクションが失敗した場合、全ての操作は「ロールバック」され、セッションが開始される前の元の状態に回復されません。JMS のコミットおよびロールバックメソッドは、JMS セッションの範囲で実行されるため、他のプログラム操作には影響ありません。

## ネイティブ環境ペイン

JMS コンポーネントエディタ内のペインで、メッセージと関連付けられているさまざまな属性 (ヘッダフィールド、本文のコンテンツなど) を表示します。

## パブリッシュ / サブスクライブ

一般的に使用される 2 つの主要なメッセージングパラダイムの 1 つ (もう 1 つはポイントツーポイントメッセージングです)。パブリッシュ / サブスクライブメッセージングでは、キューはトピックとも呼ばれます (該当する項目を参照)。トピックは、従来のキューとは異なり多数の「リスナ」で共有するように設計されている一方、ポイントツーポイントメッセージングでは通常キューは受信アプリケーション (または小規模な、明確な数のユーザ) に関連付けられます。トピックは共有されるため、メッセージはすべての登録されたリスナにより受信されるまでトピックから削除されません。また、フィルタ (ポイントツーポイントメッセージングではメッセージセレクタにより実行されます) は、受信アプリケーションの制御下というよりはパブリッシュ / サブスクライブシステムの管理制御下となります。トピックにメッセージをポストするクライアントは「パブリッシャ」、メッセージを使用するクライアントは「サブスクライバ」と呼ばれます。

## 非同期の配信

パブリッシュ / サブスクライブメッセージングでは、メッセージブローカ (またはトピックマネージャ) により *MessageListener* の *onMessage()* メソッドが呼び出されると、非同期の配信が発生します。対照的に、同期の配信では受信アプリケーションはメッセージを要求して取得します。

## ポイントツーポイント (PTP)

一般的に使用される 2 つの主要なメッセージングパラダイムの 1 つ (もう 1 つはパブリッシュ / サブスクライブメッセージングです)。PTP システムでは、キューはトピックごとに整理されませんが、その代わり、通常はキューをメールボックスのように扱う専用の受信者 (クライアントアプリケーション) に「属し」ます。クライアントは、管理による中断を最小限に抑えて他のクライアントとの間でメッセージを送受信します。オプションで、受信アプリケーションにセレクタ (またはフィルタ) を実装すると特別な条件に基づきメッセージの選択的な取得ができます。

## ロールバック

処理されるメッセージセッションで、セッションの *rollback()* メソッドを呼び出すと、作成されたメッセージは破棄され (送信されません)、使用されたメッセージは何も処理されずキューに残ります。コミットおよびトランザクションも参照してください。



# B

## メッセージセレクトタ構文

メッセージセレクトタは、`TRUE` と評価した場合はメッセージを選択し、`FALSE` と評価した場合はメッセージを無視する式を含む「文字列」です。JMS セレクトタ式の構文は、SQL92 のサブセットに基づきます。メッセージセレクトタの評価は、優先順位レベル内で左から右へと順に行われますが、評価順序を変更するために括弧を使用することもできます。一貫性を保つため、定義済みのセレクトタのリテラルと演算子名は、ここでは大文字で表示します (ただし、実際には大文字と小文字は区別されません)。セレクトタには、次に説明するルールに従ってトークン、演算子、および式を含めることができます。

### リテラル

文字列リテラルは、一重引用符で囲まれます。文字列リテラルに一重引用符が含まれる場合は、一重引用符を二重に使用して表すことができます (例: `'it's'` → `'it's'`)。Java *String* リテラルでは、Unicode 文字エンコードが使用されます。

「正確」な数値リテラルは、小数点のない数値です (例: `59`、`-257`、`+82`)。サポートされているものは、Java *long* の範囲の数字です。正確な数値リテラルでは、Java 整数リテラル構文が使用されます。

「近似」の数値リテラルは、科学的表記の数値 (例: `7E4`、`-27.9E2`) または小数点のある数値 (例: `7`、`-95.7`、`+16.2`) で、サポートされているものは、Java *double* の範囲の数字です。近似のリテラルでは、Java 浮動小数点リテラル構文が使用されます。

ブールリテラルには、`TRUE` または `FALSE` という値を使用できます。

### 識別子

識別子は、ヘッダフィールド参照またはプロパティ参照のいずれかです。識別子とは、Java 識別子開始文字で始まり Java 識別子部分文字である文字が後に続く文字のシーケンスです。識別子開始文字は、メソッド `Character.isJavaIdentifierStart()` によって `true` が返される文字すべてです。このような文字には、下線や `$` も含まれます。識別子部分文字は、メソッド `Character.isJavaIdentifierPart()` によって `true` が返される文字すべてです。

識別子には、`NULL`、`TRUE`、または `FALSE` を使用することはできません。

また、識別子には、`NOT`、`AND`、`OR`、`BETWEEN`、`LIKE`、`IN`、または `IS` を使用することもできません。

識別子では、大文字と小文字が区別されます。

メッセージヘッダフィールド参照は、*JMSDeliveryMode*、*JMSPriority*、*JMSMessageID*、*JMSTimestamp*、*JMSCorrelationID*、および *JMSType* に制限されています。

*JMSMessageID*、*JMSCorrelationID*、および *JMSType* の値は *null* にすることができますが、このような場合、NULL 値として扱われます。

「JMSX」で始まる名前は、すべて JMS 定義のプロパティ名です。

「JMSX\_」で始まる名前は、すべてプロバイダ固有のプロパティ名です。

「JMS」で始まらない名前は、すべてアプリケーション固有のプロパティ名です。存在しないプロパティが参照された場合、その値は NULL となります。存在する場合は、その値が対応するプロパティ値になります。

「空白文字」は、Java に対して定義されているものと同じで、スペース、水平タブ、フォームフィード、ラインターミネータが含まれます。

## 式

セレクトタは、条件付き式です。true と評価するセレクトタはすべて一致し、false または unknown と評価するセレクトタは一致しません。

算術式は、算術演算子、数値を含む識別子、数値リテラル、または他の算術式、あるいはこれらすべてから構成されます。

「条件付き」式は、比較演算子、論理演算子、ブール値を含む識別子、ブールリテラル、または他の条件付き式、あるいはこれらすべてから構成されます。

標準の括弧 () は、式評価を順序付けるためにサポートされています。

優先順位の論理演算子は、NOT、AND、OR です。

比較演算子は、=、>、>=、<、<=、<> (等しくない) です。

比較できるのは、like タイプの値のみです。唯一の例外であるのは、正確な数値と近似の数値を比較するには有効であるということです (必要なタイプの変換は、Java 数値昇格のルールに従って行われます)。like 以外のタイプの値を比較すると、セレクトタは常に false になります。

String と Boolean の比較は、= (等しい) および <> (等しくない) に制限されています。2 つの文字列は、含まれる文字のシーケンスが同じ場合にのみ等しくなります。

優先順の算術演算子は、次のとおりです。

+, -( 単項 )

\*, / ( 乗算および除算 )

+, - ( 加算および減算 )

**注記：** 算術演算子では、Java 数値昇格が使用されなければなりません。

## 比較

- *arithmetic-expr1* [NOT] BETWEEN *arithmetic-expr2* および *arithmetic-expr3*

例:

`age BETWEEN 15 and 19` は、`age >= 15 AND age <= 19` に等しくなります。

`age NOT BETWEEN 15 and 19` は、`age < 15 OR age > 19` に等しくなります。

- *identifier* [NOT] IN (*string-literal1*, *string-literal2*,...) (*identifier* は *String* または NULL 値)

例: `Country IN ('UK', 'US', 'France')`

これは、「UK」に対しては `true`、「Peru」に対しては `false` となり、次の式に等しくなります。

`(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`

例: `Country NOT IN ('UK', 'US', 'France')`

これは、「UK」に対しては `false`、「Peru」に対しては `true` となり、次の式に等しくなります。

`NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`

IN または NOT IN 演算の *identifier* が NULL の場合、演算の値は `unknown` になります。

- *identifier* [NOT] LIKE *pattern-value* [ESCAPE *escape-character*] (*identifier* は *String* 値、*pattern-value* は「\_」(下線)が任意の単一文字を表す文字列リテラル、「%」は文字の任意のシーケンス(空のシーケンスを含む)、その他のすべての文字はそれ自体。オプションの *escape-character* は、*pattern-value* で「\_」と「%」の特別な意味をエスケープするために文字が使用される単一文字の文字列リテラル)

`phone LIKE '12%3'` は、「123」または「12993」に対しては `true` で、「1234」に対しては `false` になります。

`phone NOT LIKE '12%3'` は、「123」および「12993」に対しては `false` で、「1234」に対しては `true` になります。

`word LIKE 'l_se'` は、「lose」に対しては `true` で、「loose」に対しては `false` になります。

`underscored LIKE '\_%' ESCAPE '\'` は、「\_foo」に対しては `true`、「bar」に対しては `false` になります。

LIKE または NOT LIKE 演算の *identifier* が NULL の場合、演算の値は `unknown` になります。

- *identifier IS NULL* は、`null` のヘッダフィールド値または見つからないプロパティ値をテストします。

• `identifier IS NOT NULL` は、`null` ではないヘッダフィールド値またはプロパティ値の存在をテストします。

次のメッセージセクタは、メッセージタイプが `car`、色が `red`、重量が `3500 lbs` を超えているメッセージを選択します。

```
"JMSType = 'car' AND color = 'red' AND weight > 3500"
```

## Null 値

前に説明したとおり、ヘッダフィールドとプロパティ値は `NULL` にすることができます。 `NULL` 値を含むセクタ式の評価は、`SQL92 NULL` セマンティックによって定義されます。つまり、`SQL` では、`NULL` 値を `unknown` として扱います。 `unknown` の値を含む比較または算術では、結果として `unknown` の値が常に生成されます。 `IS NULL` および `IS NOT NULL` 演算子は、`unknown` のヘッダまたはプロパティ値を `TRUE` または `FALSE` の値に変換します。

## 特記事項

メッセージセクタで使用されると、`JMSDeliveryMode` は、「`PERSISTENT`」または「`NON_PERSISTENT`」という値を持ちます。

日付と時刻の値では、標準の `Java long millis` 値を使用します。日付または時刻リテラルをメッセージセクタに含む場合、このようなリテラルは `millis` 値の整数リテラルでなければなりません。 `millis` 値を生成する標準の手段は、`java.util.Calendar` を使用することです。 `SQL` では固定小数点比較および算術をサポートしていますが、`JMS` メッセージセクタではサポートしていません (これは、正確な数値リテラルを非小数点に制限するためです)。

`SQL` コメントはサポートされていません。

# C

## メッセージヘッダおよびプロパティ

### JMS によって定義されるヘッダフィールド

JMS では、定義済みのヘッダフィールドの同じセットがすべてのメッセージでサポートされています (次を参照)。ほとんどのヘッダフィールドには、`exteNd Composer` または MOM ベンダのいずれかによってランタイム時に自動的に設定された値があります。

#### JMSCorrelationID

JMS クライアントでは、*JMSCorrelationID* ヘッダを使用して、メッセージを互いに関連付けることができます (リクエスト/応答の場合に対して)。このフィールドには、入力 DOM からノード値をマップすることによって取得したものや、ECMAScript を使用して動的に作成したものなど、任意の文字列値を含めることができます。このフィールドの使用は必須ではありません。

#### JMSDeliveryMode

このヘッダフィールドには、メッセージが送信されたときに指定された配信モード (PERSISTENT または NON\_PERSISTENT) が含まれます。送信セッションの開始では、このヘッダフィールドは無視され、送信が完了すると、送信方法によって指定された配信モードがこのフィールドに含まれます。

このフィールドは、Send Message セットアップウィザードで選択した持続性の値を使用して、自動的に入力されます。

#### JMSDestination

*JMSDestination* ヘッダフィールドは、メッセージの送信時には無視され、送信後は、送信メッセージによって指定されたあて先オブジェクトがこのフィールドに含まれます。

このフィールドには、手動で何かを入力する必要はありません。これは、必要な接続情報 (つまり、移動先キューの選択) が、JMS コンポーネントを初めて作成したときに自動的に設定されるためです。

## JMSExpiration

このフィールドには、手動で何かを入力する必要はありません。Send Message セットアップウィザードでは、(特に) 発信メッセージに対する Time-to-Live を指定するよう求められます。「送信」セッション中、実際にメッセージを送信する準備が整うと、exteNd では、メッセージの有効期限を Time-to-Live 値と現在の UTC 時間 (両方の値の単位はミリ秒) の合計として計算します。送信が完了すると、メッセージの *JMSExpiration* ヘッダフィールドにこの合計値が含まれます。Send Message アクションが作成されたときに Time-to-Live 値がゼロであった場合、メッセージには有効期限がありません (つまり、期限が切れることはありません)。

**注記:** ほとんどの JMS ベースの MOM では、期限の切れたメッセージをクライアントが受信することはありません。

## JMSMessageID

*JMSMessageID* 値は、MOM 環境においてメッセージを一意に識別します。この値は、JMS プロバイダによって自動的に設定され、読み込み専用です (メッセージが送信された後でのみ)。

## JMSPriority

*JMSPriority* フィールドには、メッセージの優先度を反映した 10 個の値 (「0」から「9」) の中の 1 つがある文字列値が含まれます。このフィールドには、Send Message セットアップウィザードで指定した値が自動的に入力されます。「0」から「4」の値は、通常の優先度の範囲であることを示し (「4」がデフォルト)、「5」から「9」の値は、「急送」優先度の段階を示します。

**注記:** キュー内におけるメッセージの順序が優先度設定によって決定される方法は、JMS では定義されません。メッセージの順序においてこの機能がどのように影響するかについては、MOM ベンダのマニュアルを参照してください。

## JMSRedelivered

クライアントアプリケーションで、*JMSRedelivered* マーカーセットを含むメッセージが受信された場合、キューマネージャでは予定よりも早くメッセージを配信しようとし、クライアントによってそのメッセージが認識されない可能性があります (システム障害のため)。このフィールドは、キューマネージャまたはメッセージブローカによって制御されるもので、アプリケーション制御によるものではありません。

## JMSReplyTo

*JMSReplyTo* フィールドは、メッセージの送信時にクライアントによって提供されたあて先を含むよう設計されています。このフィールドは、メッセージに対する返信 (存在する場合) が送信されるべきあて先を表します。

**注記:** このヘッダフィールドは、exteNd Composer の JMS コンポーネントエディタで書き込み可能な項目として現在は表示されません。



## JMSTimestamp

このフィールドには、送信するメッセージがプロバイダに渡された時刻が含まれます。これは、たとえば、メッセージの「送信」セッションがトランザクション制御によるものかどうかに基づいて、実際の送信時刻になる場合とならない場合があります。

## JMSType

ユーザによる設定が可能なこのフィールドには、メッセージの作成時にクライアントによって提供される任意の文字列が含まれます。送信者は、受信者にとって便利な可能性のある *JMSType* にどんな値でも割り当てることができます。たとえば、アプリケーション定義の *JMSType* 値は、あらゆる受信者がさまざまな特定のメッセージタイプを処理できるようにすることによって、メッセージフィルタリングを容易にすることが可能です。

**注記：**一部の JMS プロバイダでは、レポジトリにメッセージタイプ定義を保存し、そのようなタイプ定義に対応する *JMSType* のランタイム値を期待することがあります。このような状況が MOM 環境で発生する場合は、該当するレポジトリで定義されているリーガル値に対応する *JMSType* にシンボリック値を使用します（詳細については、MOM のマニュアルを参照してください）。

## メッセージプロパティ

メッセージプロパティは、事実上、追加のヘッダフィールドとして機能します。JMS には、JMS 定義のプロパティ、プロバイダ固有のプロパティ、およびユーザ定義のプロパティという、3 つのおおまかなプロパティのカテゴリがあります。JMS Connect ではこれらのカテゴリをすべてサポートしていますが、JMS は、プロパティをサポートするアプリケーションを、*JMSXGroupID* および *JMSXGroupSeq* 以外は必要としません（次を参照）。

プロパティ値 (null でない場合) のタイプは、*boolean*、*byte*、*short*、*int*、*long*、*float*、*double*、または *String* でなければなりません。特定の定義済みのプロパティに対して使用可能な値については、次に説明します。

プロパティ値は、存在する場合、メッセージを送信する前に送信者によって設定されます。メッセージがクライアントによって受信された場合、プロパティはすべて読み込み専用となります。取得したメッセージのプロパティ値をクライアントが設定しようとすると、*MessageNotWriteableException* がスローされます。

## JMS 定義のプロパティ

メッセージクライアントまたはプロバイダ、あるいはその両方によりオプションとして入力される可能性のある JMS 固有のプロパティフィールドの数は、JMS によって定義（および JMS Connect によって表示）されます。これらの JMS 定義のプロパティ（「JMSX」というプリフィックスが付いている）には、次が含まれます。

- ◆ *JMSXUserID* (String) — メッセージを送信しているユーザを識別する任意の文字列（これは、送信操作中にプロバイダによって設定されます）。

- ◆ **JMSXAppID (String)** — 送信アプリケーションの ID (これは、送信操作中にプロバイダによって設定されます)。
- ◆ **JMSXDeliveryCount (int)** — メッセージ配信試行の回数 (プロバイダによって設定されます)。
- ◆ **JMSXGroupID (String)** — このメッセージが属するメッセージグループの (クライアントによる設定が可能な) ID (バッチ処理でメッセージを送信しているクライアントによって使用されます)。
- ◆ **JMSXGroupSeq (int)** — グループ内におけるこのメッセージの (クライアントによる設定が可能な) シーケンス番号。
- ◆ **JMSXProducerTXID (String)** — このメッセージが生成されたトランザクションの ID (プロバイダによって設定されます)。
- ◆ **JMSXConsumerTXID (String)** — このメッセージが使用されたトランザクションの ID (プロバイダによって設定されます)。
- ◆ **JMSXRcvTimestamp (long)** — 最終的な使用者にメッセージが配信された時刻 (プロバイダによって設定されます)。
- ◆ **JMSXState (int)** — 1 (待機中)、2 (準備完了)、3 (期限切れ)、4 (保留) のうちの 1 つ (クライアントアプリケーションとは関連がなく、プロバイダによって内部的に使用されます)。

## プロバイダ固有のプロパティ

JMS では、プロバイダが独自のパブリックプロパティ名を定義し、「JMS\_<ベンダ名>」という形式のプリフィックスを付けることが可能です (たとえば、JMS\_IBM は、IBM 定義のプロパティのプリフィックスです)。これらのフィールドは、JMS Connect によって JMS メッセージヘッダツリービューに表示されますが、実際には JMS プロバイダが使用することを目的としています。

IBM の MQSeries はプロバイダで、exteNd の JMS Connect では、3 つのベンダ固有のプロパティである JMS\_IBM\_MsgType、JMS\_IBM\_PutAppType、および JMS\_IBM\_Format を表示します。JMS コンポーネントによってメッセージが受信されると、これらのフィールドには、MQSeries 固有の制御情報が通常は入力されます。発信メッセージに関しては、これらのフィールドに適切な値を自分で入力したり、空白のままにしたりすることができます。これらのフィールドのセマンティックの詳細については、MQSeries の『*Application Programming Reference*』を参照してください。

## ユーザ定義のプロパティ

JMS では、ユーザが独自のカスタムプロパティを定義することが可能です。この機能は、JMS Connect によって JMS コンポーネントエディタに表示されます (次を参照)。メッセージに添付できるユーザ定義のプロパティフィールドの数または種類には制限がありません。ただし、ユーザ定義のプロパティの名前は、(他のヘッダやプロパティの場合と同様に) メッセージセレクトラ識別子の構文ルールに従う必要があります。

# 索引

## A

After Send Maps 60

## B

Before Send Maps 60

[Break] コマンド 67

Browse Messages アクション 62

キュー 41

Bytes Message 102

BytesMessage 21

## C

CICS 14

CICS RPC Enterprise Enabler 89

COBOL 89, 120

COBOL コピーブック、「コピーブック」を参照

commit 29, 33, 36, 39

[Continue] コマンド 67

## D

DTD 40

durable subscriber 117

## E

ECMAScript 79, 81, 87

getters と setters 105

メソッドの概要 105

ECMAScript による拡張機能 102

[Error on No Message] チェックボックス 68

Expand Tree 93

Expression Editor 87

## F

FIFO (先入れ先だし) 15

forbidden drag operation 81

## G

getField() 106

getJMSBytesBody() 106

getJMSMapField() 106

getJMSObjectBody() 107

getJMSStreamField() 107

getter メソッド 105

## H

hasMessages() 67

[Header Document Name] 59

HTTP サブレット 110

## I

IBM 13, 27

IBM メッセージプロパティ 130

IS NULL 126

## J

java.util.Enumeration 62

JDBC 74

JMSCorrelationID 20, 61, 81, 98, 127

JMSDeliveryMode 20, 60, 98, 127

JMSDestination 20, 60, 127

JMSExpiration 20, 60, 128

JMS\_IBM\_Format 130

JMS\_IBM\_MsgType 130

JMS\_IBM\_PutApplType 130

JMSMESSAGE 67

JMSMessageID 20, 60, 67, 98, 128

ループの終了 72

JMS-MQSERIES 104

JMSPriority 20, 60, 97, 98, 128

JMSRedelivered 21, 60, 128

JMSReplyTo 21, 128

JMSTimestamp 20, 60, 98, 129

JMSType 21, 81, 82, 97, 98, 129

JMSXAppID 130

JMSXConsumerTXID 130  
JMSXDeliveryCount 130  
JMSXGroupID 130  
JMSXGroupSeq 130  
JMSXProducerTXID 130  
JMSXRcvTimestamp 130  
JMSXState 130  
JMSXUserID 129  
JMS コンポーネント  
  新規作成 42  
JMS サービス 109  
  開始および停止 115  
  削除 115  
JMS サービスコンソール 115  
JMS サービストリガ 114  
JMS サービスリスナの停止 116  
JMS 接続リソース 111  
JMS 標準 13  
  定義範囲 24  
JNDI 27  
JTA (Java トランザクション API) 17

## M

MapMessage 21  
  [Map] コマンド 86  
  [Message Filter] タブ 23  
MessageListener 22, 109, 114, 115  
Message Transaction アクション 73, 76  
  [Message Transaction] ダイアログボックス 76  
  [Message Type] プルダウンメニュー 64  
MQSeries ホストマシン 33  
MQSeries 11, 13, 14, 27, 32, 130  
MQSeries キュー 31

## N

NON\_PERSISTENT 98  
NON\_PERSISTENT モード 20

## O

Object Message 102  
ObjectMessage 21  
onMessage() 22  
OS/2 90  
[Override Connection Queue] 63

## P

PERSISTENT 98  
PERSISTENT モード 19  
Progress Software 13  
  [Property Name] 65  
  [Property Type] 65  
PTP、「ポイントツーポイント (PTP)」を参照

## R

Receive Message Maps 72  
Receive Message アクション 67  
Repeat While アクション 67  
rollback 29, 33, 36, 39  
RPC 17, 118

## S

Select Occurrences 93  
SendMessage アクション 53  
setJMSBytesBody() 106  
setJMSMapField() 106  
setJMSMsgProperty() 107  
setJMSObjectBody() 107  
setJMSStreamField() 107  
setter メソッド 105  
setValue() 106  
SonicMQ 13, 14  
SQL92 23, 126  
StreamMessage 21

## T

[Template Category] 43  
[Template Name] 43  
[Test Options] ダイアログ ボックス 34  
TextMessage 21  
TopicPublisher 19  
TopicSubscriber 19  
[Transacted] チェックボックス 29, 33, 36, 39, 42, 76  
TransactionInProgressException 17  
Try/On Error アクション 68

## U

Use Sent Message ReplyTo Field 63, 100

## W

WHILE ループ 67

## X

XAResource インタフェース 17  
xconfig.xml 28  
XML Map コンポーネント 45  
XML スタブドキュメント 58, 65  
XML テンプレート 40, 65  
XML 要素のバッチによるマップ 95  
XPath 81, 87  
XPath() メソッド 83  
XSL 40

## あ

あいまいなトランザクション状態 73  
アクション 49  
アクションモデル 49  
アクションモデルペイン 45  
宛先 119

## い

一時的なモデルのキュー 33  
一度だけ 20

## え

永続的加入者 19  
エラー通知 24

## か

確実な一度だけの配信 16  
確実に一度だけ 20  
確認 73  
カスタムプロパティ 52, 65  
カスタムヘッダプロパティ 82  
管理オブジェクト 27, 119

## き

起動と削除 23  
キュー  
2つ以上の使用 42  
一時的なモデル 33  
空白 68  
クラスタ化 15  
参照 62  
パブリッシュ/サブスクライブ 18  
変更 62  
キューマネージャ 33

## こ

コピーブック 21, 47, 120  
コミット 16, 73  
自動 73  
コロン 98  
コンポーネントエディタ 45

## さ

サービストリガ 114  
サービスの品質 16  
参照 62  
参照と読み出し 75  
サンプルドキュメント 40

## し

- 式駆動型の接続 26
- 識別子、セレクタ 123
- 式、セレクタ 124
- 自動ロールバックプロトコル 74
- 受信
  - ブロック 22
- 受信のループ 67
- 状況に応じたピックリスト 102
- 初期コンテキストファクトリ、JNDI 30, 36
- ショッピングカートアプリケーション 16
- シリアル化が可能な Java オブジェクト 21
- 信頼性 16, 20

## す

- スケラビリティ 24
- スタイルシート 40
- スタブドキュメント 40
- すべてのメッセージに対して繰り返し 67

## せ

- 制約
  - プロパティのマップ 83
- セキュリティプリンシパル 30, 37
- 接続 27
  - MQSeries トピック 37
  - キュー 27
  - トピック 34
  - プロバイダ固有 31
- 接続ファクトリ 27
- 接続リソース
  - 作成 25
- セレクタ 23, 62, 67, 70
  - 文法 123

## そ

- 送信先 27
- 送信先、変更 53

## た

- タイムアウト値 22

## て

- データグラム 23, 120
- データタイプ 83
- データの取り出しと送信 22
- データベース操作 74

## と

- トピック 18
- トピック接続 34
  - 参照は許可されていない 41
- ドラッグアンドドロップ 81
- トランザクション制御 16, 73
  - 範囲 74
- トランザクション制御の範囲 74
- ドロップ先 81

## に

- 認証 24, 30, 37

## ね

- ネイティブ環境ペイン 45, 46, 61

## は

- 配達を保証 19
- 配備の問題 114
- 破壊的削除 67
- パスワード 33
- パフォーマンス問題 16
- ハブとスポークの構造 12
- パブリッシュ/サブスクライブ 121, 18
  - 参照定義なし 62

## ひ

非 JMS Client 34  
比較演算子 124  
ピックリスト 102  
否定不可能 81  
非同期取り出し 22  
非同期のトリガ 109  
非同期の配信 121  
非同期プロセス 14  
非破壊的な読み出し 75

## ふ

フィルタ 22, 83  
    制約 98  
    本文のタイプ 98  
フェイルオーバ 24  
負荷バランス 24  
複数のリスナ 110  
プライバシー 24  
ブラウザコンソール 115  
ブロードキャスタ/リスナ 22  
ブロックとポーリング 22  
プロバイダ 28  
プロバイダ URL 30, 37  
プロバイダ固有のプロパティ 130  
プロパティ 20, 129  
プロパティ、カスタム 59  
分散トランザクション 17

## へ

ヘッダ、メッセージ  
    送信前 60  
    データのマップ 80, 81

## ほ

ポイントツーポイント  
    参照 62  
ポイントツーポイント (PTP) 17, 121  
ホスト名 33  
保存/転送 23  
本文のタイプ 21  
    フィルタ 70

## ま

待ち時間 16, 19, 22  
マップ名 65  
マップ、ヘッダ 81  
マルチタスク処理 16

## み

未解決のトランザクション 73  
ミリ秒の値 57

## め

メールボックス 17  
メッセージ  
    XML 21  
    キュー内での最大数 15  
    繰り返し 67  
    構造 20  
    コピーブック 21  
    寿命 15  
    セレクトタ 23  
    タイプ定義 24  
    非同期取り出し 22  
    フィルタ 22  
    読み込み専用 21  
メッセージキュー 15  
メッセージセレクトタ、「セレクトタ」を参照  
メッセージに対する返信 99  
メッセージの確認 73  
メッセージの繰り返し 67  
メッセージフィルタ 62, 67, 70, 97  
メッセージブローカ 18  
メッセージプロパティ 129  
メッセージ本文の DOM 65

## も

モデルのキュー 33

## ゆ

ユーザ名 32  
優先度 20

## リ

- リクエスト - 応答 23, 81
- リクエスト - 応答メッセージング 99
- リソース超過 15, 20
- リテラル、セクタ 123
- リトルエンディアン 90
- リポジトリ 24
- リモートプロシージャコール 14

## る

- ループコントロール 67

## れ

- 例外 29, 36, 39, 70, 99
  - TransactionInProgressException 17

## ろ

- ロールバック 16, 67, 73