

JEUS MQガイド

JEUS v8.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

Restricted Rights Legend

All TmaxSoft Software (JEUS®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(JEUS®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

Trademarks

JEUS® is registered trademark of TmaxSoft Co., Ltd.

JEUS®は、TmaxSoft Co., Ltd.の登録商標です。

Java and Solaris are registered trademarks of Oracle Corporation and its subsidiaries and affiliates.

Java、Solarisは、Oracle Corporation及びその子会社、関連会社の登録商標です。

Microsoft, Windows, and Windows NT are registered trademarks or trademarks of Microsoft Corporation.

Microsoft、Windows、Windows NTは、Microsoft Corporationの登録商標または商標です。

HP-UX is a registered trademark of Hewlett Packard Enterprise Company.

HP-UXは、Hewlett Packard Enterprise Companyの登録商標です。

AIX is a registered trademark of International Business Machines Corporation.

AIXは、International Business Machines Corporationの登録商標です。

UNIX is a registered trademark of X/Open Company, Ltd.

UNIXは、X/Open Company, Ltd.の登録商標です。

Linux is a registered trademark of Linus Torvalds.

Linuxは、Linus Torvaldsの登録商標です。

Other products and company names are trademarks or registered trademarks of their respective owners.

その他、記載されている会社名、製品名などは、各社の商号、商標または登録商標です。

The names of companies, systems, and products mentioned in this manual may not necessarily be indicated with a trademark symbol (TM, ®).

本マニュアルに記載されている会社名、システム名、製品名などには必ずしも商標表示(TM、®)を付記しておりません。

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

Detailed Information related to the license can be found in the following directory : \${INSTALL_PATH}/lib/licenses

この製品の一部ファイルまたはモジュールは、APACHE2.0、CDDL1.0、EDL1.0、OPEN SYMPHONY SOFTWARE1.1、TRILEAD-SSH2、Bouncy Castle、BSD、MIT、SIL OPEN FONT1.1のライセンスに準拠します。

文書情報

文書名: JEUS MQガイド

発行日: 2016年10月14日

ソフトウェアバージョン: JEUS v8.0

ガイドバージョン: v2.1.1

目次

このガイドについて	xiii
第1章 紹介	1
1.1. Java Message Service(JMS)	1
1.2. JEUS MQの特徴	2
第2章 JEUS MQクライアント・プログラミング	3
2.1. 概要	3
2.2. JMS管理対象オブジェクト	5
2.2.1. JNDIサービスの定義	6
2.2.2. コネクション・ファクトリー	7
2.2.3. デスティネーション	8
2.3. コネクションとセッション	11
2.3.1. コネクションの作成	11
2.3.2. 物理的コネクションの共有	12
2.3.3. セッションの作成	14
2.3.4. クライアント・ファシリティー・プーリング	15
2.3.5. NONE_ACKNOWLEDGEモード	15
2.3.6. JMSContext	18
2.4. メッセージ	18
2.4.1. メッセージ・ヘッダ・フィールド	18
2.4.2. メッセージ・プロパティ	19
2.4.3. メッセージ・ボディ	20
2.4.4. ファイル・メッセージ	21
2.5. トランザクション	25
2.5.1. ローカル・トランザクション	25
2.5.2. 分散トランザクション	26
第3章 JEUS MQサーバーの設定	29
3.1. 概要	29
3.1.1. ディレクトリー構造	29
3.1.2. WebAdmin	30
3.2. JMSリソースの設定	32
3.2.1. デスティネーションの設定	33
3.2.2. 永続サブスクライバーの設定	35
3.3. JMSエンジンの設定	36
3.3.1. 基本情報の設定	36
3.3.2. サービス・チャンネルの設定	38
3.3.3. コネクション・ファクトリーの設定	39
3.3.4. 永続ストアの設定	40
3.3.5. メッセージのソート設定	43
3.4. サーバー管理およびモニタリング	43

3.4.1.	サーバー管理	44
3.4.2.	サーバー・モニタリング	46
第4章	JEUS MQクラスタリング	51
4.1.	概要	51
4.2.	クラスタリングの種類	51
4.2.1.	コネクション・ファクトリー・クラスタリング	51
4.2.2.	デスティネーション・クラスタリング	52
4.3.	クラスタリングの使用	53
4.3.1.	サーバー設定	53
4.3.2.	クライアントの設定	54
4.4.	使用例	56
4.4.1.	一般的な使用例	56
4.4.2.	正しくない使用例	57
第5章	JEUS MQのフェイルオーバー	59
5.1.	概要	59
5.2.	サーバー障害の復旧	59
5.2.1.	ネットワークの構成	60
5.2.2.	コネクション・ファクトリーの設定	63
5.2.3.	永続ストアの設定	63
5.2.4.	フェイルバック	64
5.3.	クライアント障害の復旧	64
5.3.1.	再接続	64
5.3.2.	コネクション・ファクトリーの再使用	65
5.3.3.	デスティネーションの再使用	65
5.3.4.	応答待機時間	65
5.3.5.	コネクションの復旧	66
5.3.6.	セッションの復旧	67
5.3.7.	メッセージ送信の復旧	69
5.3.8.	メッセージ受信の復旧	70
5.3.9.	メッセージの紛失防止とトランザクション	72
第6章	JEUS MQの特殊機能	73
6.1.	JEUS MQのメッセージ・ブリッジ	73
6.1.1.	サーバーの設定	73
6.2.	JEUS MQのメッセージ・ソート	76
6.2.1.	サーバーの設定	77
6.2.2.	クライアントの設定	77
6.3.	JEUS MQのグローバル・オーダー	78
6.3.1.	クライアントの設定	78
6.4.	JEUS MQのメッセージ・グループ	79
6.4.1.	サーバーの設定	79
6.4.2.	クライアントの設定	79
6.5.	JEUS MQのメッセージ管理機能	80

6.5.1.	メッセージのモニタリング	81
6.5.2.	メッセージの制御	87
6.5.3.	デスティネーションの制御	94
6.6.	高信頼性メッセージの送信	97
6.6.1.	LPQの有効化	97
6.6.2.	LPQ使用の設定	98
6.6.3.	LPQリスナーの設定	100
6.6.4.	LPQの設定	101
付録 A.	ジャーナルストアの追加属性	105
A.1.	プロパティ・リファレンス	105
付録 B.	JDBC永続ストアの列	107
B.1.	メタ情報テーブル	107
B.2.	デスティネーション・テーブル	107
B.3.	永続サブスクライバー・テーブル	107
B.4.	メッセージ・テーブル	108
B.5.	サブスクリプション・メッセージ・テーブル	108
B.6.	トランザクション・テーブル	109
索引	111

図目次

[図 1.1]	JMS Messaging	1
[図 2.1]	一般的な場合とNONE_ACKNOWLEDGEモードの場合のメッセージ送信	16
[図 2.2]	AUTO_ACKNOWLEDGEモードとNONE_ACKNOWLEDGEモードでのメッセージ受信	17
[図 2.3]	ファイル・メッセージの送信例	23
[図 2.4]	JMSトランザクションの範囲	25
[図 3.1]	JEUS MQサーバー関連ファイル	29
[図 3.2]	Jms Resource画面	30
[図 3.3]	Jms Engine画面	61
[図 3.4]	Jms Resource設定画面	32
[図 3.5]	Destination設定画面	33
[図 3.6]	Durable Subscriber設定画面	36
[図 3.7]	Jms Engine設定画面	61
[図 3.8]	Jmsエンジンの詳細設定画面	37
[図 3.9]	サービス・チャンネルの設定画面	38
[図 3.10]	Connection Factory設定画面	39
[図 3.11]	Persistence Store設定画面	41
[図 3.12]	永続ストアの設定 - Journal	42
[図 3.13]	永続ストアの設定 - Jdbc	42
[図 3.14]	コネクション・ファクトリー管理	44
[図 3.15]	デスティネーション管理	44
[図 3.16]	永続サブスクライバー管理	45
[図 3.17]	デスティネーションのモニタリング	46
[図 3.18]	永続サブスクライバーのモニタリング	47
[図 3.19]	クライアントのモニタリング	47
[図 3.20]	JMSペンディング・トランザクションのモニタリング	48
[図 4.1]	コネクション・ファクトリー・クラスタリング	52
[図 4.2]	デスティネーション・クラスタリング	53
[図 4.3]	クラスター内のデスティネーションの設定例	77
[図 4.4]	JEUS MQクラスタリング構成の良い例	56
[図 4.5]	JEUS MQクラスタリング構成の悪い例	57
[図 5.1]	3台のアクティブ・サーバーと2台のスタンバイ・サーバーを利用したJEUS MQクラスタリングの構成	60
[図 5.2]	JMSフェイルオーバーの設定画面	61
[図 5.3]	フェイルオーバー設定 - アクティブ・サーバーの設定	62
[図 5.4]	フェイルオーバー設定 - スタンバイ・サーバーの設定	62
[図 5.5]	コネクション・ファクトリーの設定	63
[図 5.6]	フェイルオーバーの設定 - リクエスト・ブロッキング・タイム	65
[図 6.1]	メッセージ・ブリッジの設定	74
[図 6.2]	メッセージ・ソート設定	77
[図 6.3]	デスティネーションにメッセージ・ソートを適用	77

[図 6.4]	メッセージ・グループ設定	79
[図 6.5]	デスティネーションでのメッセージ一覧の表示	82
[図 6.6]	永続サブスクリプションでのメッセージ一覧の表示	82
[図 6.7]	表示するメッセージ一覧の設定	83
[図 6.8]	表示されたメッセージ一覧	84
[図 6.9]	表示されたメッセージ一覧で特定メッセージの詳細情報を表示	85
[図 6.10]	メッセージの詳細情報の表示	86
[図 6.11]	メッセージの移動	88
[図 6.12]	メッセージの移動設定	89
[図 6.13]	メッセージの削除	90
[図 6.14]	91
[図 6.15]	メッセージのエクスポート	92
[図 6.16]	メッセージのエクスポートの設定	93
[図 6.17]	メッセージのインポート	94
[図 6.18]	メッセージのインポート設定	94
[図 6.19]	メッセージの生産制御	95
[図 6.20]	メッセージの消費制御	96

例目次

[例 2.1]	<<jndi.properties>>	6
[例 2.2]	ファイル・メッセージの送信	23
[例 2.3]	ファイル・メッセージの受信	24

このガイドについて

対象読者

本書は、JEUS[®](以下、JEUS) MQ(Message Queue)クライアントの開発者およびJEUS MQサーバーの管理者を対象としています。

前提知識

Java Message Service¹についての詳しい内容は、『[Java Message Service Specification - version 2.0](#)』を参照してください。

JEUSの基本的な使用方法と製品を理解するには、以下のガイドについてあらかじめ熟知することをお勧めします。

- JEUS 紹介ガイド
- JEUS インストール & スタートガイド

本書のすべてのサンプルと環境構成は、UNIXスタイルに準拠します。Microsoft Windows[™](以下、Windows)など他の環境で作業を行う場合は、次のような事項を考慮してください。たとえば、Windowsプラットフォームでは、ディレクトリー区切り子をUNIXスタイルのスラッシュ(/)からWindowsスタイルのバックスラッシュ(\)に変えて使用してください。また、環境変数もWindowsスタイル(%%)に変更して使用してください。本書で触れているJEUS_HOMEは、JEUSがインストールされているディレクトリーです。

制限事項

本書の内容は、Java標準に準拠して作成されていますが、本書で触れているJava EEやJava仕様については詳しく取り上げていません。関連内容についてはJava関連ドキュメントを参照してください。

本書の構成

本書は、計6章と2つの付録で構成されています。

- [???](#)

Java Message ServiceIについての簡単な説明とJEUS MQの特徴について説明します。

- [???](#)

JEUS MQクライアントのタイプ、作成手順、および管理対象オブジェクトについて説明します。

- [???](#)

JEUS MQサーバーを実行するためのJMSエンジンとリソースの設定方法について説明します。

- [???](#)

JEUS MQサーバーの負荷を減らし、円滑なサービスを提供するために、複数のサーバーを連結して使用するクラスタリングについて説明します。

- [???](#)

JEUS MQでサーバーやネットワークに障害が発生した際の復旧方法について説明します。

- [???](#)

JEUS MQの特殊機能である、メッセージ・ブリッジ、メッセージ・ソート、グローバル・オーダー、メッセージ・グループ、メッセージ管理機能について説明します。

- [???](#)

ジャーナル・ストアに追加設定できる属性について説明します。

- [???](#)

JEUS MQで永続ストアをJDBCで設定した際に作成されるテーブルの列について説明します。

表記上の規則

表記	意味
<<AaBbCc123>>	プログラム・ソースコードのファイル名
<Ctrl>+C	CtrlキーとCキーを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
ハイパーリンク	メール・アカウント、Webサイト
>	メニューの実行順
+----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
<div>参考</div>	参照/注意事項
<div>注</div>	注意事項
[図 1.1]	図の名前
[表 1.1]	表の名前
AaBbCc123	Javaコード、XMLドキュメント
[<i>command argument</i>]	オプション・パラメータ
< xyz >	「<」と「>」の間の内容は実際に使用される特定の名前または値で置き換えられる
	構文の中の相互に排他的な選択項目の選択肢を示す 例) A B: AとBのいずれかを選択
...	パラメータ、値、または他の情報が繰り返される
\${ }	環境変数

システム要件

	要求事項
プラットフォーム	Solaris 9, 10, 11
	HP-UX 11.x, 11i, 11iV2
	IBM AIX 5L, 6L, AIX 7L
	MS Windows 2008, 2012, Vista, 7, 8
ハードウェア	最小2GB以上、推奨20GBのハードディスク容量
	推奨1GB以上のメモリー容量
JDK	JDK 7, JDK 8

関連文書

ガイド	説明
JEUS 紹介ガイド	JEUSサーバーについて全般的に紹介し、JEUSのアーキテクチャーを含む各構成要素について記述しています
JEUS インストール&スタートガイド	JEUSについて紹介し、JEUSのインストールおよび開始方法について記述しています
JEUS サーバガイド	JEUSシステムおよびサーバーの概要とシステムの管理方法について記述しています
JEUS EJBガイド	JEUS EJBエンジンおよびEJBモジュールのデプロイについて記述しています
JEUS アプリケーション&デプロイメントガイド	Java EEアプリケーションをJEUSにデプロイするための様々な方法とツールについて記述しています
JEUS アプリケーションクライアントガイド	Java EEクライアントとJEUS間の相互運用について記述しています
JEUS リファレンスガイド	JEUSを使用するために必要な詳細設定とJEUSの使用方法について記述しています
JEUS WebAdminガイド	JEUSのWeb管理ツールであるWebAdminを利用したJEUSの設定および制御、モニタリング、クラスタリング、リソースの設定および管理について記述しています

参考文献

- Java Message Service Specification - version 2.0
<http://www.oracle.com/technetwork/java/jms/index.html>
- Java EE Tutorials
<http://www.oracle.com/technetwork/java/javaee/documentation/index.html>
- XML Reference - domain.xml JMSエンジンの設定
JEUS_HOME/docs/reference/schema/index.html

お問合せ先

Korea

TmaxSoft Co., Ltd.
45, Jeongjail-ro, Bundang-gu,
Seongnam-si, Gyeonggi-do, 13613
South Korea
Tel: +82-31-8018-1000
Fax: +82-31-8018-1115
Email: info@tmax.co.kr
Web (Korean): <http://www.tmaxsoft.com>
TechNet: <http://technet.tmaxsoft.com>

USA

TmaxSoft Inc.
101 North Wacker Drive, Suite 2014,
Chicago, IL 60606
U.S.A
Tel: +1-312-525-8330
Email: info@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/us_en/home

Japan

TmaxSoft Japan Co., Ltd.
5F Sanko Bldg, 3-12-16 Mita, Minato-Ku, Tokyo, 108-0073
Japan
Tel: +81-3-5765-2550
Fax: +81-3-5765-2567
Email: info@tmaxsoft.co.jp
Web (Japanese): <http://www.tmaxsoft.co.jp>

China

Beijing TmaxSoft System Software Co., Ltd.
Room103, No.2 Huizhong Building, Seven Street Shangdi,
Haidian District, Beijing, 100085
P.R.China
Tel: +86-10-6298-8827
Email: info@tmaxsoft.com.cn
Web (Chinese): http://www.tmaxsoft.com/cn_en/home_cn_en

Brazil

Tmax Brasil Sistemas e Serviços Ltda.
Av. Copacabana, 177, sala 32~35 Empresarial 18 do Fortel
Alphaville Barueri, Sao Paulo, 06472-001
Brazil
Tel: +55-11-4191-3100
Fax: +55(11) 4191-3705 (extension#112)
Email: info.bra@tmaxsoft.com
Web (Portuguese): http://www.tmaxsoft.com/br_en/home_br_en

Russia

Tmax Rus L.L.C.
Leninsky prospekt, 113/1 (Park Place Moscow),
Office 318e, Moscow, 117198
Russia
Tel: +7(495)970-01-35
Email: info.rus@tmaxsoft.com
Web (Russian): http://www.tmaxsoft.com/ru_ru/home_ru_ru

Singapore

Tmax Singapore Pte. Ltd.
430 Lorong 6, Toa Payoh #10-02,
OrangeTee Building, 319402
Singapore
Tel: +65-6259-7223
Fax: +65-6258-7112
Email: info.sg@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/sg_en/home_sg_en

United Kingdom

TmaxSoft UK Ltd.
215 Knyvett House, Watermans Business Park,
The Causeway, Staines TW18 3BAB
United Kingdom
Tel: +44-1784-895005
Email: info.uk@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/gb_en/home_gb_en

Canada

TmaxSoft Canada, Inc.
2425 Matheson Blvd East, 8th floor,
Unit 824 Mississauga, ON, L4W 5K4
Canada
Tel: +1-905-361-2888
Email: info.canada@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/ca_en/home_ca_en

Australia

TmaxSoft Proprietary Limited
L32, 101 Miller Street, North Sydney 2060
Australia
Tel: +91-9845-330-704
Email: info.aus@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/au_en/home_au_en

India

TmaxSoft Technologies Private Limited
Sobha Alexander Plaza, 3rd Floor,
16/2 Commissariat Road, Bangalore-560025
India
Tel: +91-9845-330-704
Email: info.india@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/in_en/home_in_en

Turkey

TmaxSoft Co., Ltd. Turkey Liaison Office
Windowist Tower. Eski Buyukdere Cad. No:26,
Maslak 34467 Istanbul
Turkey
Tel: +90-544-553-6045
Email: cslee@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/tr_en/home_tr_en

第1章 紹介

本章では、Java Message Service (以下、JMS)について簡単に紹介した後、JEUS MQの特徴について説明します。

1.1. Java Message Service(JMS)

JMSは、アプリケーション間の通信をメッセージベースで行うためのJava標準APIを定義したものです。ここでは、メッセージングに必要な構成要素およびメッセージモデルに該当するインターフェースを定義し、これらの関係について説明しています。

JMSには以下のような特徴があります。

- 疎結合構造

メッセージ・プロデューサーとコンシューマは相手を知る必要がありません。

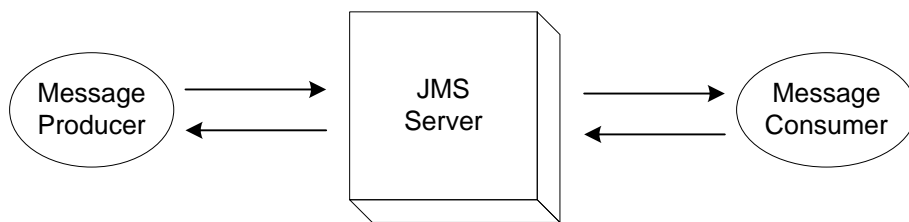
- 非同期通信

メッセージ・コンシューマは要求しなくても、メッセージがサーバーに到達次第、受信できます。

- 信頼性のあるメッセージ送信

メッセージが必ず1回(Once-and-Only-Once)送信されることを保証します。

[図 1.1] JMS Messaging



参考

MSIに関する詳細とAPIの使用方法については、JMSの仕様(<http://java.sun.com/products/jms/>)とJava EE Tutorials(<http://java.sun.com/javaee/reference/tutorials/>)を参照してください。

1.2. JEUS MQの特徴

JEUS MQ(Message Queue)は、TmaxSoftのJava EEサーバーであるJEUSに含まれているJMS仕様を実装したものです。非同期I/OおよびXAトランザクションをサポートし、セキュリティー機能を強化した、可用性の高いメッセージ・サービスを提供します。

JEUS MQは、JMSバージョン2.0に対応しており、以下のような特徴があります。

- 非同期I/Oのサポート

非同期I/Oを使用することで、同じシステム・リソースでより多くのクライアントを同時に処理できます。

- XAトランザクションのサポート

JMSの仕様で選択事項として定義されているXAトランザクションをサポートします。

- セキュリティーの強化

SSL(Secure Socket Layer)通信をサポートしており、JEUSセキュリティーを使用することができます。

- 高可用性

クライアントはネットワークやサーバーの障害から自動的にフェイルオーバーされます。

- 規模の可変性

クライアントやメッセージの処理量が増加しても、JEUS MQクラスタリングを使用して負荷を分散できます。

第2章 JEUS MQクライアント・プログラミング

本章では、JEUS MQクライアントのタイプ、作成手順および管理対象オブジェクトについて説明します。

2.1. 概要

JEUS MQクライアントのタイプと作成手順について説明します。

クライアントのタイプ

JMSクライアントは、その役割によってメッセージ送信者(メッセージ・プロデューサー)とメッセージ受信者(メッセージ・コンシューマ)に分けられ、1つのクライアントが2つの役割を果たすこともあります。

また、Javaアプリケーションをデプロイして実行する方式に応じて、JEUS MQクライアントを以下のように分類できます。

- 独立アプリケーション

Java SE環境で独立して実行されるクライアントです。

- Java EEアプリケーション

EJB(Enterprise JavaBeans)やサーブレットのように、Java EEサーバーにデプロイして実行されるクライアントです。JEUSにJava EEアプリケーションをデプロイして実行する方法については、『JEUS アプリケーション & デプロイメントガイド』を参照してください。

- メッセージ駆動型Bean(Message-Driven Bean)

EJBの一種で、Java EEアプリケーションの特殊なケースといえますが、動作方式においてはシングル・スレッド方式で作成されているJava SEアプリケーションをマルチ・スレッド方式に拡張します。メッセージ駆動型Bean(以下、MDB)についての詳細は、EJB関連書籍や『[Java EE Tutorials](#)』を参照してください。

JEUSでMDBを使用するための設定方法は、『*JEUS EJBガイド*』の「第9章 メッセージ駆動型Bean(MDB)」を参照してください。

参考

現在JEUS MQでは、Java以外の言語で作成されたクライアントには対応していません。

クライアントの作成手順

JEUS MQクライアントのタイプのうち独立アプリケーションの場合を基準にして、JEUS MQクライアントを作成する手順について説明します。

Java EE環境で実行されるクライアントの場合は、XMLデプロイメント記述子に設定して、初期JNDI(Java Naming and Directory Interface)のルックアップ手順を代替することができます。

1. JNDI初期コンテキストを作成します。

```
Context context = new InitialContext();
```

JEUSのJNDIサービスを使用するための環境設定方法は、[「2.2.1. JNDIサービスの定義」](#)を参照してください。

2. JNDIルックアップによってコネクション・ファクトリーを取得します。

```
ConnectionFactory connectionFactory =  
    (ConnectionFactory) context.lookup("jms/ConnectionFactory");
```

JEUS MQ専用のAPIを使用して、JNDIルックアップなしでコネクション・ファクトリーを取得する方法もあります。詳細については、[「2.2.2. コネクション・ファクトリー」](#)を参照してください。

3. コネクション・ファクトリーを使用してコネクションを確立します。

```
Connection connection = connectionFactory.createConnection();
```

4. コネクションからセッションを作成します。

```
Session session = connection.createSession(false, AUTO_ACKNOWLEDGE);
```

5. JNDIルックアップによってデスティネーションを取得します。

```
Destination destination = (Destination) context.lookup("ExamplesQueue");
```

コネクション・ファクトリーの場合と同様、JEUS MQ専用のAPIを使用すると、JNDIルックアップなしでデスティネーションを取得できます。これについては、[「2.2.3. デスティネーション」](#)を参照してください。

6. メッセージを送信するクライアントの場合、セッションからメッセージ・プロデューサーを作成します。

```
MessageProducer producer = session.createProducer(destination);
```

または、メッセージを受信したい場合は、セッションからメッセージ・コンシューマを作成します。

```
MessageConsumer consumer = session.createConsumer(destination);
```

メッセージを非同期的に受信して処理するには、メッセージ・コンシューマにMessageListenerインターフェースを実装したオブジェクトを追加で登録します。

```
MessageListener listener = new MyMessageListener();
consumer.setMessageListener(listener);
```

7. コネクションを開始します。

```
connection.start();
```

8. ビジネス・ロジックを実行しながらメッセージを送受信します。

メッセージはメッセージ・プロデューサーを通じて送信します。

```
TextMessage message = session.createTextMessage("Hello, World");
producer.send(msg);
```

または、同期的にメッセージを受信する場合、メッセージ・コンシューマのreceive()メソッドを呼び出します。

```
Message message = consumer.receive();
```

一方、非同期的にメッセージを受信する場合、受信したメッセージは**手順6**で登録したMessageListenerオブジェクトのonMessage()メソッドで処理されます。

9. すべてのメッセージの送受信が完了したら、コネクションを終了します。

```
connection.close();
```

2.2. JMS管理対象オブジェクト

JMS管理対象オブジェクト(Administered Object)には、コネクション・ファクトリー(Connection Factory)とデスティネーション(Destination)の2つがあります。

主にJMSサーバーの設定でサーバー内に作成され、サーバー管理者によって管理されます。JMSクライアントはJMSサービスを利用するために、この管理対象オブジェクト(または、その参照)をサーバーから取得して使用します。クライアントがJMS管理対象オブジェクトをサーバーから取得する最も一般的な方法はJNDIルックアップです。

本節では、JEUS MQサーバーからコネクション・ファクトリーやデスティネーションの参照を取得する方法と、JNDIルックアップのためにクライアントでJEUS JNDIサービスを定義する方法について説明します。また、クライアントでAPIを使用してJEUS MQサーバーにデスティネーションを動的に作成する方法と、JEUS MQサーバーのデッド・メッセージ・デスティネーションについて説明します。

2.2.1. JNDIサービスの定義

JMS管理対象オブジェクトをJNDIルックアップによって取得するには、まずJNDIサービスを定義する必要があります。

JNDIサービスは、以下の3つのプロパティで定義できます。

- java.naming.factory.initial
- java.naming.factory.url.pkgs
- java.naming.provider.url

定義された3つのプロパティは以下の方法で適用できます。

• JNDIプロパティ・ファイルの作成

JNDIサービスを定義する最も簡単な方法は、プロパティ値を指定したjndi.propertiesファイルを作成することです。

以下は、JEUSのJNDIサービスを使用するための環境を記述したjndi.propertiesファイルの例です。

[例 2.1] <<jndi.properties>>

```
java.naming.factory.initial=jeus.jndi.JEUSContextFactory
java.naming.factory.url.pkgs=jeus.jndi.jns.url
java.naming.provider.url=127.0.0.1:9736
```

このようなjndi.propertiesファイルを作成してクラスパスに格納するか、クライアント・プログラムをデプロイする際にクラス・ファイルと一緒にJARファイルにパッケージングすると、InitialContextオブジェクトがjndi.propertiesファイルの設定を読み込むことができます。

• システム・プロパティで渡す方法

クライアントの実行時にカスタマイズが必要な場合は、JARファイルに含まれているjndi.propertiesファイルを変更することは困難です。このような場合は、クライアントを実行時に以下のようにシステム・プロパティで設定値を渡す方法を使用できます。

```
java -Djava.naming.factory.initial=jeus.jndi.JEUSContextFactory \  
-Djava.naming.factory.url.pkgs=jeus.jndi.jns.url \  
-Djava.naming.provider.url=127.0.0.1:9736 \  
. . .
```

• アプレット・パラメータで渡す方法

JEUS MQクライアントがアプレット形式の場合は、jndi.propertiesファイルを作成するよりは、以下のよう
にHTMLファイルで<applet>タグの<param>タグを使用して渡す方法がカスタマイズに有効です。

```
<applet code="JeusMqApplet" width="640" height="480">
  <param name="java.naming.factory.initial"
    value="jeus.jndi.JEUSContextFactory"/>
  <param name="java.naming.factory.url.pkgs" value="jeus.jndi.jns.url"/>
  <param name="java.naming.provider.url" value="127.0.0.1:9736"/>
</applet>
```

Applet.getParameter()メソッドを使用して、アプレット内にInitialContextの作成に必要な環境を設定するコードを以下のように作成します。

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
  getParameter("java.naming.factory.initial"));
env.put(Context.URL_PKG_PREFIXES, getParameter("java.naming.factory.url.pkgs"));
env.put(Context.PROVIDER_URL, getParameter("java.naming.provider.url"));

Context context = new InitialContext(env);
```

● コードに挿入

クライアント・コード内でInitialContextの作成に必要な環境を直接設定する方法です。

以下のように、InitialContextを作成する際、プロパティが含まれたHashtableオブジェクトをパラメータとして渡します。

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "jeus.jndi.JEUSContextFactory");
env.put(Context.URL_PKG_PREFIXES, "jeus.jndi.jns.url");
env.put(Context.PROVIDER_URL, "127.0.0.1:9736");

Context context = new InitialContext(env);
```

参考

この方法を使用すると、他のJNDIサービスを使用する場合やサービスの定義が変更された場合にソースコードも変更する必要があるため、メンテナンスが困難となります。

2.2.2. コネクション・ファクトリー

コネクション・ファクトリー(Connection Factory)には、JEUS MQサーバーにコネクションを確立するのに必要な情報が含まれています。

一般的にJMSクライアントはJNDIルックアップによってコネクション・ファクトリーを取得し、これを使用してJMSサーバーへのコネクションを作成します。

```
ConnectionFactory connectionFactory =
    (ConnectionFactory) context.lookup("jms/ConnectionFactory");
QueueConnectionFactory queueConnectionFactory =
    (QueueConnectionFactory) context.lookup("jms/QueueConnectionFactory");
TopicConnectionFactory topicConnectionFactory =
    (TopicConnectionFactory) context.lookup("jms/TopicConnectionFactory");
```

JEUS MQの分散トランザクションを使用する場合、XAコネクション・ファクトリー、XAキュー・コネクション・ファクトリー、XAトピック・コネクション・ファクトリーをルックアップして使用することができます。

Java EEクライアントでは、InitialContext.lookup()メソッドを呼び出す代わりに、@Resourceアノテーションを使用して、以下のようにコネクション・ファクトリーを取得する方法もあります。

```
@Resource(mappedName="jms/ConnectionFactory")
private static ConnectionFactory connectionFactory;
```

JNDIサービスを利用できない状況では、JEUS MQが提供する専用のAPIを使用してJNDIルックアップなしでコネクション・ファクトリーを取得することができます。

```
jeus.jms.client.util.JeusConnectionFactoryCreator connectionFactoryCreator =
    new jeus.jms.client.util.JeusConnectionFactoryCreator();
connectionFactoryCreator.setFactoryName("ConnectionFactory");
connectionFactoryCreator.addBrokerAddress("127.0.0.1", 9741, "internal");
ConnectionFactory connectionFactory =
    (ConnectionFactory) connectionFactoryCreator.createConnectionFactory();
```

参考

JNDIルックアップや@Resourceアノテーションを使用する場合は、コネクション・ファクトリーを取得するためにコネクション・ファクトリーのJNDI名（本節の例では「jms/ConnectionFactory」）を参照しますが、JeusConnectionFactoryCreatorを使用する場合は、JEUS MQサーバーが内部的に使用するコネクション・ファクトリー名（最後の例の「ConnectionFactory」）で参照するので注意が必要です。コネクション・ファクトリー名とJNDI名の違いについては、[「3.3.3. コネクション・ファクトリーの設定」](#)の設定例を参照してください。

2.2.3. デスティネーション

デスティネーション(Destination)はサーバーによって管理されるメッセージを格納する領域であり、キューとトピックの2種類があります。JMSクライアントは、サーバーで管理するデスティネーション・オブジェクトの参照を取得して、デスティネーションにメッセージを送信したり、デスティネーションからのメッセージを受信したりします。

JEUS MQクライアントでデスティネーションを取得する最も一般的な方法は、JEUS MQサーバーがデスティネーション参照を登録したJNDIサーバーを使用してJNDI名でルックアップすることです。

```
Queue queue = (Queue) context.lookup("jms/ExamplesQueue");
Topic topic = (Topic) context.lookup("jms/ExamplesTopic");
```

Java EEクライアントでは、InitialContext.lookup()メソッドを呼び出す代わりに@Resourceアノテーションを使用して、以下のようにデスティネーション参照を取得する方法もあります。

```
@Resource(mappedName="jms/ExamplesQueue")
private static Queue queue;
```

コネクション・ファクトリーと同様、JEUS MQ専用のAPIを使用すると、JNDIルックアップなしでデスティネーション参照を取得できます。この場合、デスティネーションを参照するには、JEUS MQサーバーが内部的に使用するデスティネーション名を使用する必要があります。

```
jeus.jms.client.util.JeusDestinationCreator destinationCreator =
    new jeus.jms.client.util.JeusDestinationCreator();
destinationCreator.setDestinationName("ExamplesQueue");
destinationCreator.setDestinationClass(Destination.class);
Destination destination = (Destination) destinationCreator.createDestination();
```

また、JMS実装にはSession.createQueue(String)とSession.createTopic(String)メソッドを使用することができます。実際のデスティネーション名をパラメータとして渡す必要があります。

```
Queue queue = session.createQueue("ExamplesQueue");
Topic topic = session.createTopic("ExamplesTopic");
```

デスティネーションの動的作成

JEUS MQは、クライアントがSession.createQueue(String)、Session.createTopic(String)メソッドを使用して、サーバーにデスティネーションを動的に作成する方法を提供しています。

この際、パラメータとして渡す文字列は、作成するデスティネーション名の後ろに疑問符(?)を付けます。オプションを設定する場合は、続けて「param=value」という形式で記述します。オプションを2つ以上設定する場合は、アンパサンド(&)を区切り子として使用します。paramおよびvalueの値として、domain.xml環境ファイルにデスティネーションを設定する際に使用した同じ名前と値を使用することができます。

参考

現在、デスティネーションを動的に作成する際に設定できるオプションは、キューの場合はexport-nameとmultiple-receiver、トピックの場合はexport-nameがあります。

以下の例では、デスティネーション名が「DynamicQueue」、JNDI名が「jms/DynamicQueue」であり、同時に2つ以上のメッセージ・コンシューマを受け入れるキューを動的に作成しています。

```
QueueConnectionFactory queueConnectionFactory =
    (QueueConnectionFactory) context.lookup("jms/QueueConnectionFactory");
QueueConnection queueConnection = queueConnectionFactory.createQueueConnection();
QueueSession queueSession =
    queueConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
Queue queue = queueSession.createQueue(
    "DynamicQueue?export-name=jms/DynamicQueue&multiple-receiver=true");
```

このAPIには、以下のような特徴があります。

- 指定した名前のデスティネーションが既にサーバーに存在するか、指定したJNDI名で他のオブジェクトが既に登録されている場合、JMSEExceptionが発生します。JNDI名を設定していない場合、デスティネーション名がJNDI名として使用されます。
- JEUS MQサーバーがセキュリティ機能を使用する場合、コネクションを作成する際に使用したサブジェクトに「jeus.jms.destination.creation」リソースに対する「createDestination」権限がない場合はJMSEExceptionが発生します。
- その他の設定はデスティネーションの基本設定に従います。
- 動的に作成したデスティネーションはJEUS MQサーバーの終了時に削除されます。

上の例で作成したデスティネーションは、JEUS MQサーバーが終了するまで、WebAdminで設定したデスティネーションと同じく動作をします。

参考

JEUS MQクラスタリングを使用している場合はデスティネーションの動的作成を使用できません。

デッド・メッセージ・デスティネーション

デッド・メッセージ・デスティネーションは、何らかの理由で正常に処理されなかったメッセージを保管するシステム・デスティネーションです。これには、メッセージに設定されたデスティネーションが見つからないか、または、指定した回数以上のメッセージ修復が繰り返される場合などが含まれます。

メッセージ・コンシューマに送信されたメッセージがサーバー側のデスティネーションに修復される場合は、以下のとおりです。

- メッセージ・コンシューマがSession.rollback()、Session.recover()などを呼び出した場合
- XAトランザクションのロールバックのような外部要因による場合

- メッセージ・リスナーのonMessage()メソッドを実行中に例外が発生した場合

修復されたメッセージはメッセージ・コンシューマに再送信されますが、ビジネス・ロジックやクライアントのエラーによって特定のメッセージ処理に繰り返し失敗すると、デスティネーションにメッセージが溜まり続け、クライアントはメッセージを受信できなくなります。このような問題を回避するために、JEUS MQはメッセージを修復する最大回数が指定できる機能を提供しています。「JMS_JEUS_RedeliveryLimit」メッセージのプロパティ値を設定し、メッセージを送信します。

```
>message.setIntProperty("JMS_JEUS_RedeliveryLimit", <integer value>)
```

メッセージ・プロパティのデフォルト値は、メッセージ・プロデューサー・クライアントを実行する際に「jeus.jms.client.default-redelivery-limit」システム・プロパティを設定して変更できます。

```
-Djeus.jms.client.default-redelivery-limit=<integer value>
```

設定していない場合は3が使用されます。JEUS MQサーバーは、起動時に「JEUSMQ_DLQ」という名前のデッド・メッセージ・デスティネーションを作成します。デッド・メッセージ・デスティネーションに蓄積されているメッセージはシステム管理ツールで処理されますが、一般的なクライアントもこの名前でデッド・メッセージ・デスティネーションにアクセスし、目的の作業を行うことができます。

参考

デッド・メッセージ・デスティネーションにメッセージが過剰に蓄積されると、OutOfMemoryErrorの原因となる可能性があります。別のメッセージ・コンシューマ・クライアントを接続しておくか、あるいは、システム管理ツールを使用して蓄積されているメッセージを一定の周期で削除するなど、デッド・メッセージ・デスティネーションにメッセージが過剰に蓄積されないように管理する必要があります。

2.3. コネクションとセッション

JMSクライアントがサーバーとメッセージを送受信するためには、サーバーとコネクションを確立してセッションを作成する必要があります。コネクションとセッションは、JMSで最も重要なクライアント・リソースであるため、システムの性能およびクライアントの障害対策に影響を与えます。JEUS MQは、コネクションの性能に関するオプションを提供しています。

本節では、オプションを設定する方法と、オプションを設定した際のJEUS MQクライアントの動作について説明します。

2.3.1. コネクションの作成

コネクションはクライアントがJMSサーバーとの作業を行うために最初に作成するオブジェクトであり、JMSサーバーとの物理的あるいは論理的な接続を示します。コネクション・ファクトリーのcreateConnection()メソッドを呼び出してコネクションを作成します。

```
public Connection createConnection()  
    throws JMSEException;  
public Connection createConnection(String userName, String password)  
    throws JMSEException;
```

JEUS MQサーバーがセキュリティー機能を使用する場合、userName/passwordに該当するサブジェクトに jeus.jms.client.connectionFactory リソースの createConnection 権限がない場合は JMSEException が発生します。

JEUS MQクライアントを実行する際、以下のオプションをシステム・プロパティで設定できます。

```
-Djeus.jms.client.connect.timeout=<long value>
```

コネクションが確立されるまで ConnectionFactory.createConnection() メソッドが待機する時間です。この時間内にコネクションが確立されなかった場合、ConnectionFactory.createConnection() メソッドで JMSEException が発生します。0 に設定した場合、コネクションが確立されるまで無限に待機します。(デフォルト値: 5秒、単位: ms)

クライアント・コードに以下の内容を追加すると、ランタイム時にも設定変更が可能です。

```
System.setProperty("jeus.jms.client.connect.timeout", <long value>);
```

あるいは

```
System.setProperty(jeus.jms.common.JeusJMSProperties.KEY_CLIENT_CONNECT_TIMEOUT,  
    <long value>);
```

2.3.2. 物理的コネクションの共有

一般的に、アプリケーションがメッセージを送信するプロセスは以下のとおりです。

1. コネクションの作成
2. セッションの作成
3. メッセージ・プロデューサーの作成
4. メッセージの送信
5. コネクションの終了

JMSコネクションが物理的コネクション(ソケット)と1対1の関係であれば、上記のような使用パターンではメッセージを1回送信するたびに新しい物理的コネクションを作成する必要があるため、性能が低下します。実際

のメッセージ送信よりも、物理的コネクションの作成に時間がかかります。さらに、物理的コネクションが多くなると、ファイル記述子の使用が増えてIOExceptionが発生する場合もあり、安定性も低下します。

JEUS 6 Fix#6以降からは、物理的コネクションを共有することで、1対1の関係で発生し得る性能および安定性の問題を解決しました。ただし、一部の環境では、物理的コネクションを共有するよりも、毎回コネクションを作成する方が有効な場合もあるため、この機能はクライアント・オプションを設定して適用するようにしています。

基本的に、物理的コネクションはコネクション・ファクトリーと1対1の関係ですが、JEUS MQのフェイルオーバーを設定している場合や「jeus.jms.client.use-single-server-entry」システム・プロパティを「false」に設定している場合は、コネクションと物理的コネクションが1対1の関係になります。JEUS MQのフェイルオーバーの詳細については、「[第5章 JEUS MQのフェイルオーバー](#)」を参照してください。

以下は、物理的コネクションの共有に関連するJVMオプションです。

- `-Djeus.jms.client.use-single-server-entry=<boolean value>`

1つのコネクション・ファクトリー当たり1つの物理的コネクションを作成し、コネクション間の共有の可否を設定します。この値をfalseにするとコネクションと物理的コネクションは1対1の関係になります。(デフォルト値:true)

クライアント・コードに以下の内容を追加すると、ランタイム時にも設定変更が可能です。

```
System.setProperty("jeus.jms.client.use-single-server-entry", <boolean value>);
```

あるいは

```
System.setProperty(jeus.jms.common.JeusJMSProperties.USE_SINGLE_SERVER_ENTRY, <boolean value>);
```

- `-Djeus.jms.client.single-server-entry.shutdown-delay=<long value>`

共有している物理的コネクションを使用しているコネクションが0個の場合、物理的コネクションを切断する時間を設定します。JVMが終了するまで物理的コネクションを維持せず、アイドル時にリソースをシステムに返すようにします。(デフォルト値: 600000(10分)、単位: ms)

クライアント・コードに以下の内容を追加すると、ランタイム時にも設定変更が可能です。

```
System.setProperty("jeus.jms.client.single-server-entry.shutdown-delay", <long value>);
```

あるいは

```
System.setProperty(jeus.jms.common.JeusJMSProperties.SINGLE_SERVER_ENTRY_SHUTDOWN_DELAY, <long value>);
```

参考

コネクションの終了をMessageListenerのonMessage内で行うと、デッドロックの原因となる可能性があります。そのため、JMS仕様で制限しています。

2.3.3. セッションの作成

JMSセッションは、メッセージを作成してデスティネーションに送信したり、デスティネーションからメッセージを受信したりするなどのすべてのメッセージング処理の基本単位です。また、JMSクライアントがローカルおよびXAトランザクションに参加する単位でもあります。

セッションおよびセッションによるすべての作業は、基本的に単一のスレッド・コンテキストによって処理する必要があります。これは、セッション・オブジェクトがマルチスレッドに対して安全でないことを意味します。

参考

JEUS MQクライアントは、複数のスレッドが同時に1つのセッションを使用する場合に動作の安全性を保証しないため、同時実行が必要なスレッドの数だけセッションを作成して使用することを推奨します。また、複数のセッションを作成することは、複数のスレッドを使用するという意味なので、JMS仕様ではJava EE WebまたはEJBコンテナ上で動作するクライアントで1つ以上のセッションを作成しないように制限しています。

コネクション・オブジェクトから以下のAPIを使用してセッションを作成します。

```
public Session createSession(boolean transacted, int acknowledgeMode)
                        throws JMSException;
```

JMS仕様は、以下の4つの確認応答モードを定義しています。

```
Session.AUTO_ACKNOWLEDGE = 1
Session.CLIENT_ACKNOWLEDGE = 2
Session.DUPS_OK_ACKNOWLEDGE = 3
Session.SESSION_TRANSACTED = 0
```

JEUS MQは、メッセージ送受信の性能を最大限にするため、追加的なACKNOWLEDGEモードをサポートします。

```
jeus.jms.JeusSession.NONE_ACKNOWLEDGE = -1
```

参考

JMS仕様では、Java EE WebまたはEJBコンテナ上で動作するクライアントでローカル・トランザクションが使用されることを制限しているため、これらのクライアントでは、トランザクション処理されたセッションの作成が制限されます。また、確認応答モードのうち、CLEINT_ACKNOWLEDGEモードの使用を制限しています。

2.3.4. クライアント・ファシリティ・プーリング

コネクションの共有で説明したとおり、一般的なアプリケーションの使用パターンでは、クライアント・ファシリティ(コネクション、セッション、メッセージ・プロデューサー)を繰り返し作成して使用します。これらのオブジェクトは作成される度にサーバーと管理メッセージを送受信しており、ユーザー・メッセージを送信するために、より多くの管理メッセージを送受信するため、性能に影響を及ぼします。

この問題を解決するために、JEUS MQではクライアント・ファシリティをプーリングする機能を提供しています。この機能は、トランザクション環境でないか、JEUS MQのフェイルオーバーが設定されていない環境でメッセージ・プロデューサーを使用する場合にのみ適用できます。コンシューマやトランザクションなどが使用される場合は、プーリングせずに使用されていたクライアント・ファシリティはクローズ時に削除します。

この機能はオプションであり、以下のようなクライアントJVMオプションを設定することができます。

- `-Djeus.jms.client.use-pooled-connection-factory=<boolean value>`

クライアント・ファシリティ・プーリングを使用するかどうかを設定します。(デフォルト値: true)

- `-Djeus.jms.client.pooled-connection.check-period=<long value>`

使用していない、プーリングされたオブジェクトを削除するチェック周期を設定します。(デフォルト値: 60000 (1分)、単位: ms)

- `-Djeus.jms.client.pooled-connection.unused-timeout=<long value>`

現在プーリングされているオブジェクトのうち、使用されていない時間が設定時間より長い場合は削除します。(デフォルト値: 120000 (2分)、単位: ms)

2.3.5. NONE_ACKNOWLEDGEモード

トランザクション処理されたセッションを除けば、JMSの基本的なACKNOWLEDGEはメッセージを受信する場合のみ意味を持ちますが、NONE_ACKNOWLEDGEモードはメッセージの送信にも影響を与えます。

ACKNOWLEDGEモードを使用すると性能は良くなりますが、JMSの重要な特徴である信頼性のあるメッセージの送受信は難しくなります。そのため、該当のセッションを介してクライアントが送受信するメッセージの

特性、性能、信頼性、そしてその他の状況を考慮した上で、NONE_ACKNOWLEDGEモードを使用するか否かを判断してください。

参考

ファイル・メッセージを送受信したり、トランザクションの処理が行われたりしたセッション内では、NONE_ACKNOWLEDGEモードもAUTO_ACKNOWLEDGEモードと同様に動作します。

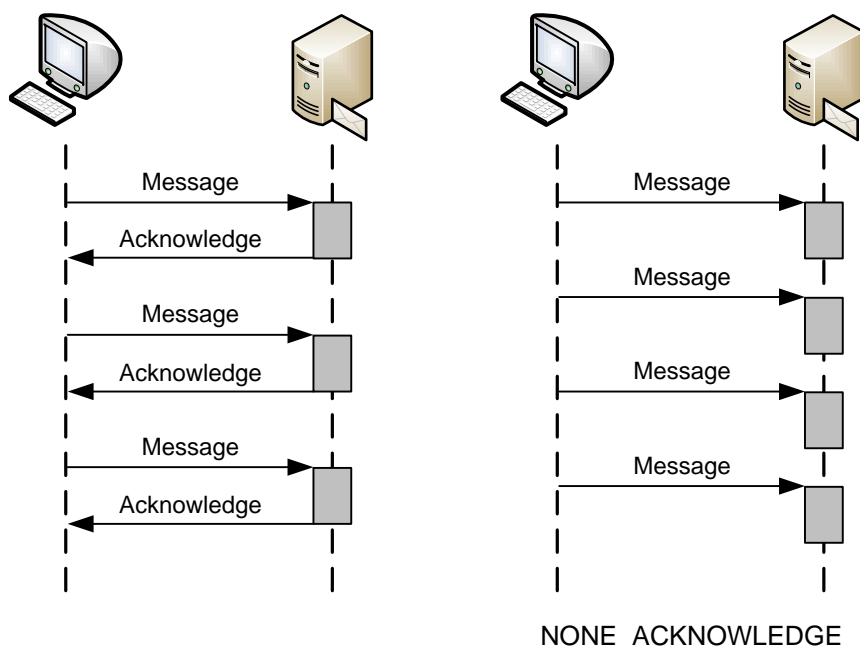
メッセージ送信時の考慮事項

クライアントが送信したメッセージがJMSサーバーに受信されたことを保証するには、サーバーからの応答があるまで、クライアント・スレッドはMessageProducer.send()メソッドを呼び出した時点で停止している必要があります。

一方、NONE_ACKNOWLEDGEモードを使用すると、MessageProducer.send()メソッドがサーバーにJMSメッセージを送信した後、即座に返します。そのため、待機時間を短縮でき、メッセージ送信の性能を高めることができます。

以下は、一般的な場合とNONE_ACKNOWLEDGEモードの場合のメッセージ送信方式の違いを示した図です。グレーで塗りつぶした部分は、サーバーにメッセージが受信されたことを記録する作業を示しています。

[図 2.1] 一般的な場合とNONE_ACKNOWLEDGEモードの場合のメッセージ送信



以下の場合、メッセージが失われることがあります。

- クライアントからサーバーにメッセージが送信される過程でネットワーク障害が発生した場合

- JEUS MQサーバーに受信されたメッセージがデスティネーションに追加される前にJEUS MQサーバーに障害が発生した場合

失われたメッセージは、メッセージ送信方式をDeliveryMode.PERSISTENTに指定した場合にも復元されません。

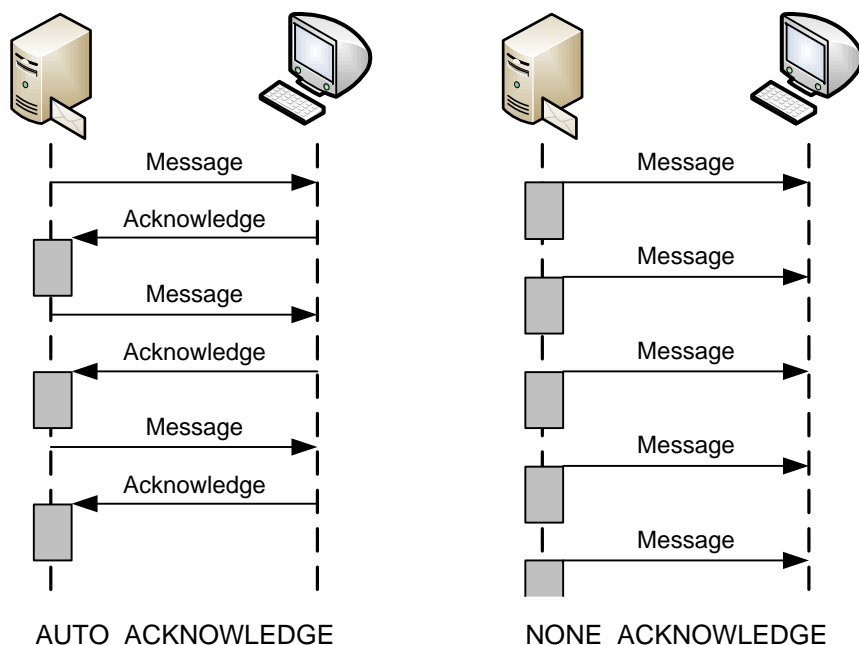
メッセージ受信時の考慮事項

JMSではメッセージの送信を保証するために、メッセージを受信したクライアントからメッセージが受信されたという確認応答を受けるまで、サーバーからメッセージ情報を削除しません。この方式では、メッセージングの信頼性は高くなりますが、もう一度ネットワーク通信が発生し、JMSサーバーによるメッセージ状態の管理が複雑になるため、性能の面では望ましくありません。

セッションのACKNOWLEDGEモードをNONE_ACKNOWLEDGEモードに設定すると、JEUS MQサーバーはメッセージを受信するネットワーク上のクライアントにJMSメッセージを送信した後、クライアントからの確認応答を待たずにサーバーからメッセージ情報を削除するので、結果的にメッセージの受信速度が向上します。

以下は、AUTO_ACKNOWLEDGEモードとNONE_ACKNOWLEDGEモードでのメッセージ受信方式の違いを示した図です。グレーで塗りつぶした部分は、クライアントに送信したメッセージ情報をサーバーから削除する作業を示しています。

[図 2.2] AUTO_ACKNOWLEDGEモードとNONE_ACKNOWLEDGEモードでのメッセージ受信



以下の場合、メッセージが失われることがあります。

- サーバーからクライアントにメッセージが送信される過程でネットワーク障害が発生した場合

- JEUS MQクライアント・ライブラリーがメッセージを処理中のクライアントがエラーを引き起こした場合
- クライアントが登録したMessageListenerオブジェクトのonMessage()メソッド内で例外が発生した場合

失われたメッセージは、Session.recover()メソッドを呼び出しても再受信されません。

2.3.6. JMSContext

JMS仕様では、利便性を高めるためにコネクションとセッションを結合したJMSContextオブジェクトを提供しています。JMSContextはコネクションとセッションの機能が結合されており、コネクションと同様、ConnectionFactoryに以下のように定義されたAPIで作成します。

```
public Connection createContext()  
    throws JMSException;  
public Connection createContext(int sessionMode)  
    throws JMSException;  
public Connection createContext(String userName, String password)  
    throws JMSException;  
public Connection createConnection(String userName, String password, int sessionMode)  
  
    throws JMSException;
```

セッションの作成時と同様、ACKNOWLEDGEモードを設定していますが、トランザクションの有無は別のパラメータではなく、ACKNOWLEDGEモードによって決まります。

JEUS MQでは、コネクションとセッションと同様、JMSContextもクライアント・ファシリティー・プーリングに含まれるので、詳細設定はコネクションとセッションのプーリングに従います。

2.4. メッセージ

本節では、JEUS MQがJMS仕様以外の付加機能を提供するためのJMSメッセージの拡張について説明します。

2.4.1. メッセージ・ヘッダ・フィールド

JMSは、以下のようなメッセージ・ヘッダ・フィールドを定義しています。

- JMSDestination
- JMSDeliveryMode
- JMSMessageID
- JMSTimestamp

- JMSCorrelationID
- JMSReplyTo
- JMSRedelivered
- JMSType
- JMSExpiration
- JMSPriority

JEUS MQはメッセージに一意のメッセージIDを付与するため、以下の機能は提供していません。

- MessageProducer.setDisableMessageID(boolean)メソッドを使用してJMSメッセージにメッセージIDを割り当てない機能
- MessageProducer.setDisableTimestamp(boolean)メソッドを使用してJMSメッセージにタイムスタンプを付与しない機能
- 管理者の設定によって、クライアントで設定したJMSDeliveryMode、JMSExpiration、JMSPriorityフィールド値を上書きする機能

確認応答モード(Acknowledge Mode)をNONE_ACKNOWLEDGEモードに設定したセッションによってメッセージを送信した後、Message.getJMSMessageID()を呼び出すと、メッセージIDはnullに表示されます。これは、MessageProducer.send()メソッドがサーバーからの応答を受信する前に返すためです。詳しい内容は、[「2.3.5. NONE_ACKNOWLEDGEモード」](#)を参照してください。

2.4.2. メッセージ・プロパティ

JMSは以下のように「JMSX」で始まるプロパティ名を定義しています。

- JMSXUserID
- JMSXAppID
- JMSXDeliveryCount
- JMSXGroupID
- JMSXGroupSeq
- JMSXProducerTXID
- JMSXConsumerTXID
- JMSXRcvTimestamp
- JMSXState

参考

「JMSX」メッセージ・プロパティは、JMSXDeliveryCountを除いて必須ではないため、JEUS MQではこのプロパティを使用しません。

以下は、JEUS MQがサポートする専用のメッセージ・プロパティです。

- JMS_JEUS_Schedule

JMSで定義するMessage Delivery Delayと同様な機能のプロパティであり、JEUS MQサーバーがメッセージ・コンシューマへのメッセージの配信を遅らせる時間です。JEUS MQサーバーは送信されたメッセージがサーバーに到達した時刻(JMSTimestampフィールド値)からこの時間が経過するまでは、メッセージをメッセージ・コンシューマに送信しません。設定値はlong型で設定します。(単位: ms)

```
Message.setLongProperty("JMS_JEUS_Schedule", <long value>);
```

- JMS_JEUS_Compaction

メッセージ・ボディを圧縮して送信するか否かを設定します。trueに設定すると、ネットワークを通じてメッセージを送信する際にZLIBライブラリーを使用してメッセージのボディ部を圧縮します。

```
Message.setBooleanProperty("JMS_JEUS_Compaction", <boolean value>);
```

- JMS_JEUS_RedeliveryLimit

メッセージをコンシューマに再送信する最大回数です。この回数を超えると、メッセージは再送信されずにデッド・メッセージ・デスティネーション([2.2.3節「デッド・メッセージ・デスティネーション」](#))に格納されます。

```
Message.setIntProperty("JMS_JEUS_RedeliveryLimit", <integer value>);
```

2.4.3. メッセージ・ボディ

JMS仕様は、メッセージ・ボディの形式によって、以下の5種類のメッセージを定義しています。

- StreamMessage
- MapMessage
- TextMessage
- ObjectMessage
- BytesMessage

各タイプのメッセージは、以下のAPIを使用してセッション・オブジェクトから作成できます。

```

public StreamMessage createStreamMessage()
    throws JMSEException;
public MapMessage createMapMessage()
    throws JMSEException;
public TextMessage createTextMessage()
    throws JMSEException;
public TextMessage createTextMessage(String text)
    throws JMSEException;
public ObjectMessage createObjectMessage()
    throws JMSEException;
public ObjectMessage createObjectMessage(Serializable object)
    throws JMSEException;
public BytesMessage createBytesMessage()
    throws JMSEException;

```

2.4.4. ファイル・メッセージ

JEUS MQはJMS基本メッセージ・タイプ以外にファイル・メッセージを追加でサポートします。JMSはメッセージ・ベースで動作するため、メッセージの内容がメモリーにロードされていないと、メッセージを送受信できません。したがって、メッセージのサイズが大きい場合は、クライアントやサーバーでメモリーのオーバーフローを引き起こす恐れがあります。

JEUS MQが提供するファイル・メッセージを使用すると、ファイルの内容をブロック単位で送信するため、このような問題を回避することができます。

メッセージの作成

jeus.jms.JeusSessionクラスに定義されている以下のメソッドを使用してファイル・メッセージを作成します。

```

public jeus.jms.FileMessage createFileMessage()
    throws javax.jms.JMSEException;
public jeus.jms.FileMessage createFileMessage(java.net.URL url)
    throws javax.jms.JMSEException;

```

JEUS MQクライアント・ライブラリーで作成したSession、QueueSession、TopicSessionオブジェクトは、jeus.jms.JeusSession、jeus.jms.JeusQueueSession、jeus.jms.JeusTopicSessionにそれぞれキャストできます。

ファイル・メッセージ・インターフェース

jeus.jms.FileMessageインターフェースの定義は以下のとおりです。

```

public interface FileMessage extends javax.jms.Message {
    public java.net.URL getURL();
    public void setURL(java.net.URL url)
        throws javax.jms.MessageNotWriteableException;
    public boolean isURLOnly();
    public void setURLOnly(boolean urlOnly);
}

```

メッセージを送信する前に、setURL()メソッドで送信するファイルのURLを指定できます。または、メッセージの作成時に、JeusSession.createFileMessage()メソッドにパラメータとして渡します。urlOnly属性は、サーバーに存在するファイルのURLのみをメッセージ・コンシューマに送信するか否かを指定します。この属性値によって、受信したファイル・メッセージに対してgetURL()メソッドを呼び出した際に取得するURLが異なります。

以下は、urlOnly属性値に応じたgetURL()値の意味です。

urlOnly	getURL()
true	JEUS MQサーバーに存在するファイルのURLです。このURLを使用して、HTTP、FTPなど、別のプロトコルでファイルを受け取ることができます
false	JEUS MQクライアント・ライブラリーがファイルの内容まで受け取ってローカルに保存した一時ファイルのURLです。詳細については、「 一時ファイルのパス 」を参照してください

ファイル・メッセージを送信する際に、MessageProducer.send()メソッドは常に(セッションの確認応答モード(Acknowledge Mode))をNONE_ACKNOWLEDGEに指定した場合も)サーバーにファイルの内容がすべて送信された後に返します。

参考

ファイル・メッセージに含まれているファイルは基本的に4KB単位で分割送信されます。このブロック・サイズは、JEUS MQクライアントを実行する際に、「-Djeus.jms.file.blocksize=<integer value>」のようにシステム・プロパティを設定して変更できます。

一時ファイルのパス

メッセージ・コンシューマがファイル・メッセージを受信時に一緒に受け取るファイルは一時ファイル・パスに格納されます。一時ファイルの格納場所は、次の条件を順次にチェックして決定します。

- JEUSにデプロイされているアプリケーションの場合

```
SERVER_HOME/.workspace/client/
```

- jeus.jms.client.workdirシステム・プロパティが設定されている場合

システム・プロパティで指定したパス

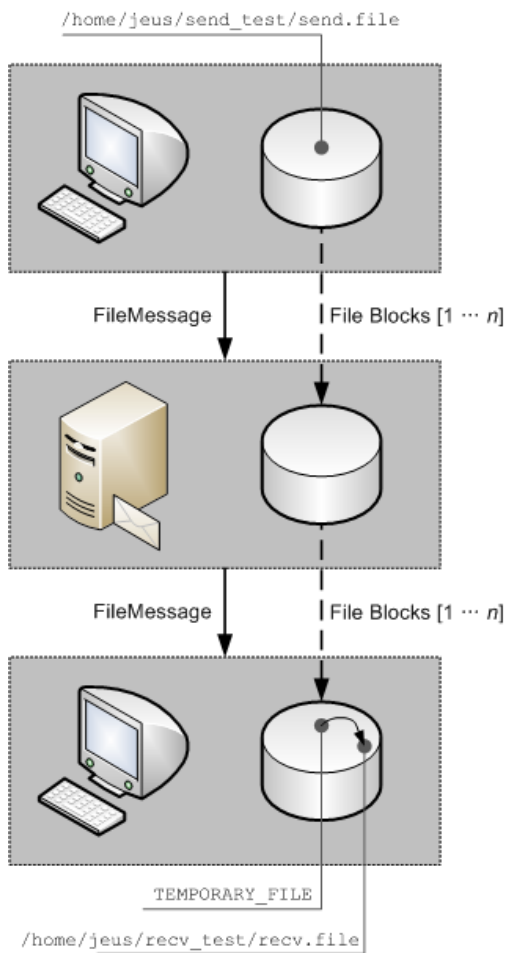
- その他の場合

USER_HOME/.jeusmq_client_work/

ファイル・メッセージの送信例

ファイル・メッセージを送信する例を通して、ファイル・メッセージAPIを使用する手順について説明します。

[図 2.3] ファイル・メッセージの送信例



以下のJavaコードは、ファイル・メッセージを使用して「/home/jeus/send_test/send.file」というファイルを送信する例です。

[例 2.2] ファイル・メッセージの送信

...

```

jeus.jms.JeusSession session = (jeus.jms.JeusSession)
    connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
MessageProducer producer = session.createProducer(destination);

jeus.jms.FileMessage message = session.createFileMessage();
File file = new File("/home/jeus/send_test/send.file");
message.setURL(file.toURI().toURL());

producer.send(message);

. . .

```

以下は、受信したファイルのURLからInputStreamを取得して、ファイルの内容を「/home/jeus/recv_test/recv.file」というファイルに書き込む例です。

[例 2.3] ファイル・メッセージの受信

```

. . .
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
MessageConsumer consumer = session.createConsumer(destination);
Message message = consumer.receive();

if (message instanceof jeus.jms.FileMessage) {
    URL url = ((jeus.jms.FileMessage) message).getURL();
    if (url != null) {
        InputStream inputStream = url.openStream();
        BufferedInputStream bufInputStream = new BufferedInputStream(inputStream);

        File outFile = new File("/home/jeus/recv_test/recv.file");
        FileOutputStream fileOutputStream = new FileOutputStream(outFile);
        BufferedOutputStream bufOutputStream =
            new BufferedOutputStream(fileOutputStream);

        int buf;
        while ((buf = bufInputStream.read()) != -1) {
            bufOutputStream.write(buf);
        }
        bufOutputStream.close();
        bufInputStream.close();
    }
}
. . .

```

2.5. トランザクション

本節では、JEUS MQがサポートするローカル・トランザクション、分散(XA)トランザクションの特性とその適用範囲、トランザクション処理など、クライアントの開発者に必要な内容について説明します。

JMSTランザクションは、1つのセッション内でメッセージを送受信する作業です。JMS仕様は、セッション内で開始/完了するローカル・トランザクションと、1つ以上のJMSセッションおよびEJB、JDBCなど、その他のトランザクション・リソースの作業を含む分散トランザクションについて定義しています。

トランザクションに参加するセッションによって送信したメッセージは実際にサーバーに到達したと見なされないため、同じセッションで作成したメッセージ・コンシューマに送信されません。また、同じセッションを利用して作成されたキューブラウザを通じても確認できません。

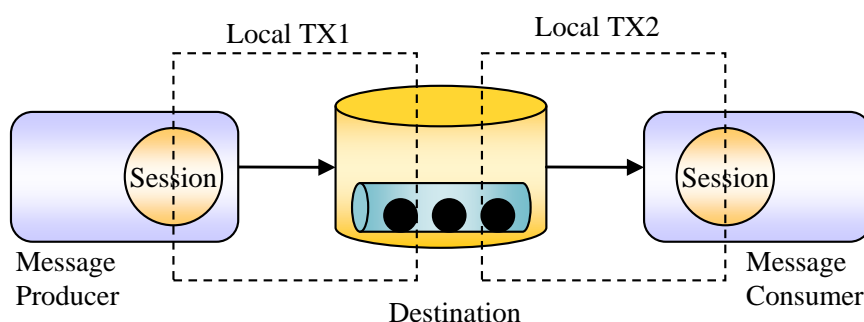
2.5.1. ローカル・トランザクション

ローカル・トランザクションは、`Connection.createSession(boolean transacted, int acknowledgeMode)`メソッドにトランザクション処理されたパラメータ値を`true`に設定して作成したセッションによって実行されます。前にコミットやロールバックが実行された時点、または初めてセッションが作成された時点から該当セッションによって実行されたすべてのメッセージング処理作業を含みます。つまり、すべての作業は、ある1つのトランザクションに属するという意味をします。

1つのローカル・トランザクション内で複数のセッションを処理することはできませんが、1つ以上のメッセージ・プロデューサーを作成して多数のデスティネーションにメッセージを送信することは可能です。同様に、多数のデスティネーションからメッセージを受信することもできます。

以下は、ローカル・トランザクションに参加する作業とその範囲を示したものです。

[図 2.4] JMSTランザクションの範囲



ローカル・トランザクションは、セッションの`commit()`、`rollback()`APIを使って行われます。トランザクション処理されたセッションは常に特定のトランザクションに参加しているため、明示的にトランザクションを開始するAPIは存在しません。

ローカル・トランザクションは、分散トランザクションとは違って、トランザクションのコミットまたはロールバックをクライアントが直接行うことがあるため、トランザクション内で非同期的なメッセージ処理が実行できます。

参考

JMS仕様では、Java EE WebまたはEJBコンテナ上で動作するクライアントでローカル・トランザクションが使用されることを制限しているため、これらのクライアントでは、トランザクション処理されたセッションの作成が制限されます。

2.5.2. 分散トランザクション

JMS仕様は、XASessionが提供するXAResourceをトランザクションに登録することでXAトランザクションに参加できます。これはJMSの付加的な仕様であり、その実装方法はベンダごとに異なります。

JEUS MQクライアントでXASessionを使用する場合は以下の事項に注意してください。

- XASessionに対してSession.getAcknowledgeMode()を呼び出すと、このXASessionがグローバル・トランザクションに参加している場合はSession.SESSION_TRANSACTEDを返し、そうでない場合はSession.AUTO_ACKNOWLEDGEを返します。
- グローバル・トランザクションに参加しているXASessionに対してSession.commit()またはSession.rollback()を呼び出すと、TransactionInProgressExceptionまたはIllegalStateExceptionが発生します。

分散トランザクションの伝播

JEUS MQクライアント・ライブラリーは、XASessionの作成時点に関係なく、JMS APIを使用するスレッドのグローバル・トランザクションにXAResourceを登録します。このようにクライアント・スレッドのトランザクションが伝播されるには、APIを明示的に呼び出す必要があります。したがって、JEUS MQの分散トランザクションの参加は**同期APIの呼び出しによってのみ**行われます。メッセージの送信や同期的な受信がこれに該当し、メッセージ・リスナーによる非同期的なメッセージの受信を分散トランザクションに含めてはなりません。

参考

分散トランザクション内で非同期的に受信したメッセージを処理するには、メッセージ駆動型Beans(以下、MDB)を使用できます。詳細については、『*JEUS EJBガイド*』の「第9章 メッセージ駆動型Bean(MDB)」を参照してください。

分散トランザクションの回復

JEUS MQサーバーは、トランザクション内で実行中のセッションの作業をストレージに保管することで、不意の障害によるサーバーの再起動時にも該当のトランザクションを回復することができます。トランザクション・マネージャーは、XASessionから取得したXAResourceによってJEUS MQ内で実行中のトランザクションIDを取得することができ、これを利用してトランザクションをコミットまたはロールバックできます。

トランザクション中に障害が発生した場合、迅速に障害を回復するため、JEUS MQのフェイルオーバー機能を使用することを推奨します。詳細については、「[第5章 JEUS MQのフェイルオーバー](#)」を参照してください。

JEUS MQサーバーは、インダウト状態ではないトランザクションの作業はブート時に自動的にロールバックします。

第3章 JEUS MQサーバーの設定

本章では、JEUS MQサーバーを実行するためにJMSエンジンとリソースを設定する方法について説明します。

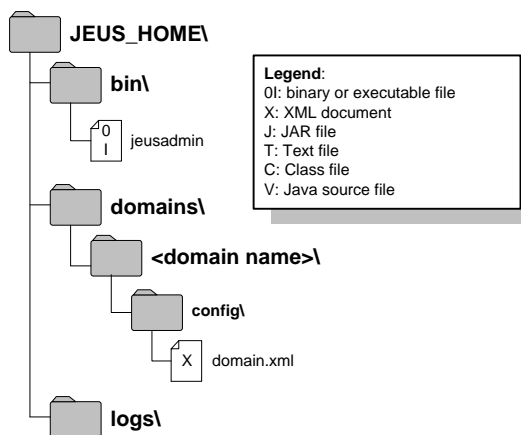
3.1. 概要

JEUS MQサーバーはJMSエンジンと表現され、サーバー当たり1つのJMSエンジンが実行されます。JEUSの各要素構成については、『JEUS サーバガイド』の「第1章 紹介」を参照してください。

3.1.1. ディレクトリー構造

以下は、JEUSおよびJEUS MQサーバーを設定し、管理するためのファイルの位置を示します。JEUS_HOMEはJEUSがインストールされているパスです。

【図 3.1】 JEUS MQサーバー関連ファイル



jeusadmin

JEUSマネージャーを管理するためのコンソール・ツールです。JEUSマネージャーとサーバー制御を含むJEUS全般の管理機能を提供します。詳細については、『JEUS リファレンスガイド』の「4.2. jeusadmin」を参照してください。

domain.xml

JEUSドメイン設定ファイルで、JEUS MQサーバーのすべての設定を行います。

domain.xmlの<jms-engine>ではJMSエンジンに関する設定を行い、<jms-resource>ではキューやトピックのようなJMSエンジンに必要なリソースを設定します。

3.1.2. WebAdmin

WebAdminの左側のメニューで[**Servers**]をクリックすると、ドメイン内に設定されているJEUSサーバーの一覧が表示されます。サーバーの一覧で、設定するサーバーの名前をクリックすると、サーバー設定画面に移動します。

JEUS WebAdminでのJEUS MQサーバー設定は、以下のメニューから構成されます。

- [**Basic**]
- [**Resource**]
- [**Engine**]

JEUS MQサーバーの設定はWebAdminを使用することを推奨します。

以下は、各メニューの画面についての説明です。**[Basic]**メニューはJMSに関連する設定箇所がないため、説明は省略します。

[Resource]

Jms Resource画面ではJMSリソースを設定します。WebAdminの[**Servers**] > [**サーバー名**] > [**Resource**] タブで[**Jms Resource**]メニューを選択します。

以下は**Jms Resource**画面です。画面の詳細については、「[3.2. JMSリソースの設定](#)」を参照してください。

[図 3.2] Jms Resource画面

The screenshot displays the JEUS WebAdmin interface for configuring JMS resources. On the left is a sidebar menu with options: jeus_domain, Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, Console, and システム状態. The main area is titled 'JMS Resource' and features a 'HISTORY' dropdown. Below the title is a warning message: 'クラスタ内で共通して使用するJMSリソースに関する設定です。クラスタに属するサーバーの場合、この項目はクラスタに設定する必要があります。クラスタの設定が優先されます。'. The interface has three main tabs: Basic, Resource (selected), and Engine. Under the Resource tab, there are sub-tabs: Listener, Jms Resource (selected), Jmx Manager, Scheduler, Lifecycle Invocation, and External Resource. The Jms Resource sub-tab contains two sections: 'Destination' and 'Durable Subscriber'. The 'Destination' section has a table with columns 'Name' and 'Type', listing 'ExamplesQueue' (QUEUE) and 'ExamplesTopic' (TOPIC). The 'Durable Subscriber' section has a table with columns 'Client ID', 'Name', and 'Destination Name', listing 'client_id' (durable) for 'ExamplesTopic'.

Name	Type
ExamplesQueue	QUEUE
ExamplesTopic	TOPIC

Client ID	Name	Destination Name
client_id	durable	ExamplesTopic

参考

本書では、JMSの設定に該当する部分についてのみ説明しているため、**[Resource]**タブの他の設定メニュー(Listener、Jmx Manager、Scheduler、Lifecycle Invocation、External Resource)の詳細については『JEUS WebAdminガイド』または関連ガイドを参照してください。

[Engine]

JMSエンジンの**[Servers] > [サーバー名] > [Engine]**タブで**[Jms Engine]**メニューを選択して、JEUS MQサーバー関連の設定を行います。

以下は、**Jms Engine**画面です。

[図 3.3] Jms Engine画面

Jms Engine画面は、以下のメニューで構成されています。

メニュー	説明
[Basic]	JEUS MQサーバーのフェイルオーバーとJMSエンジンの詳細事項を設定します。フェイルオーバーについては、「 第5章 JEUS MQのフェイルオーバー 」を参照してください。JMSエンジンの詳細設定については、「 3.3.1. 基本情報の設定 」を参照してください
[Service Config]	JEUS MQサーバーがクライアントと通信するサービス・チャンネルに関する設定です。設定方法については、「 3.3.2. サービス・チャンネルの設定 」を参照してください

メニュー	説明
[Connection Factory]	JEUS MQサーバーで使用するJMS管理オブジェクトのコネクション・ファクトリーに関する設定です。設定方法については、「 3.3.3. コネクション・ファクトリーの設定 」を参照してください
[Persistence Store]	JEUS MQサーバーで使用するPersistence Storeに関する設定です。設定方法については、「 3.3.4. 永続ストアの設定 」を参照してください
[Message Sort]	JEUS MQサーバーのメッセージ・ソート機能のための設定です。設定方法については、「 6.2. JEUS MQのメッセージ・ソート 」を参照してください

3.2. JMSリソースの設定

Jms Resource画面では、JMSリソースを設定します。WebAdminの[Servers] > [サーバー名] > [Resource] タブで[Jms Resource]メニューを選択します。

[図 3.4] Jms Resource設定画面

Jms Resource画面は、以下の2つで構成されています。

メニュー	説明
Destination	JEUS MQサーバー内のデスティネーションに関する設定です。設定方法については、「 3.2.1. デスティネーションの設定 」を参照してください
Durable Subscriber	JEUS MQサーバー内の永続サブスクライバに関する設定です。設定方法については、「 3.2.2. 永続サブスクライバーの設定 」を参照してください

3.2.1. デスティネーションの設定

キューやトピックのようなデスティネーションもJEUS MQサーバーの起動時にドメイン設定を読み込んでJEUS JNDIサービスに登録します。

Jms Resource画面の**Destination**領域で設定するリソース一覧で特定リソースの名前をクリックすると、以下のような項目を設定する画面が表示されます。各項目の情報を設定して**[確認]**ボタンをクリックすると、設定した情報が保存されます。項目を再設定するには**[再設定]**ボタンをクリックします。

[図 3.5] Destination設定画面

Destination

HISTORY

ヘルプ

デスティネーションの情報を設定します。

BasicResourceEngine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。TIP

Name	ExamplesQueue JMSブローカー内で使用されるデスティネーションの名前を指定します。
Type	queue デスティネーションのタイプを指定します。QueueまたはTopicのいずれかを指定します。
Export Name	ExamplesQueue ネーミングサーバにバインディングされるデスティネーションの名前を指定します。指定していない場合は、Name属性が使用されます。
Subscription Limit	1024 [デフォルト: 1024] デスティネーションにアクセスできるコンシューマの数を制限します。
Quota	128M [デフォルト: 128M] デスティネーションで使用できる最大のメモリ容量を指定します。メモリ使用量が指定した値を超える場合、クライアントから送られるメッセージは即時エラー処理されます。デフォルト値は128MBです。数字の後ろに単位('K'(Kilobytes), 'M'(Megabytes), 'G'(GigaBytes))を付けて設定できます。
Max Pending Limit	128 [デフォルト: 128] キューまたはトピックにメッセージリスナーが登録されている場合は、デスティネーションにメッセージが届き次第コンシューマに送信されるため、クライアントがメッセージを迅速に処理していない場合は、メッセージがクライアントメモリに溜まり、OutOfMemoryエラーが発生する可能性があります。この問題を回避するために、ディスパッチされたメッセージのうちACKNOWLEDGEを受信していないメッセージの最大の許容範囲を指定します。
Resume Dispatch Factor	0.4 [デフォルト: 0.4] 保留メッセージ数がMax Pending Limit値より大きくなると、メッセージの送信が一時的に中止されます。クライアントがメッセージを処理した後にACKNOWLEDGEを送り、保留されたメッセージ数が(Max Pending Limit * Resume Dispatch Factor)より小さいか等しくなると、ディスパッチが再開されます。
Message Sort	 デスティネーションに適用するメッセージソートを設定します。
Dead Letter Destination	JEUSMQ_DLQ [デフォルト: JEUSMQ_DLQ] このデスティネーションで正常に処理できなかったメッセージを保管するデスティネーションの名前を指定します。デフォルト値のJEUSMQ_DLQはシステムが提供するキューであり、特に設定しなくても自動で生成されます。指定したデスティネーションが存在しない場合は、デフォルト値が使用されます。
Expiration Policy	Delete [デフォルト: Delete] このデスティネーションで配信されずに期限切れになったメッセージの処理方法を設定します。(Delete: 期限切れメッセージを削除、Redirect: 期限切れメッセージをDead Letter Destinationに再送信) デフォルト値はDeleteです。
Redelivery Delay	S ロールバックまたは復旧されたメッセージが再送信されるまでの遅延時間を指定します。単位は秒です。

Message Group

デスティネーションに適用するメッセージグループを設定します。

Message Handling	<div>▼</div> <p>[デフォルト: Pass] デスティネーションがメッセージグループを処理する方式を指定します。PassまたはGatherを使用できます。Passを指定すると、一般メッセージと同様に扱います。Gatherを指定すると、メッセージグループを完成させて1つのメッセージとして送信します。</p>
Expiration Time	<div>S</div> <p>[デフォルト: -1] デスティネーションが未完成メッセージグループを保持する最大時間を秒単位で指定します。デフォルト値は-1で、メッセージグループは完成されるまで消去されません。</p>

Override Client Attributes

このデスティネーションに送信されるメッセージに限って、ここで設定する値がクライアントプログラミングによる設定値に優先して適用されます。

Expiration Time	<div>S</div> <p>この設定値はメッセージプロデューサによって定義されたTime To Live(expiration-time)設定に優先して適用されます。単位は秒です。</p>
-----------------	--

デスティネーション・メモリーの管理

クライアントに送信されていないメッセージがデスティネーションに継続して溜まると、JVMがOutOfMemoryErrorを出力し、サーバーが終了することがあります。これを防ぐために、JEUS MQはデスティネーション設定によって、デスティネーションごとにメモリー管理ポリシーを設定できる方法を提供します。

デスティネーションが使用しているメモリーが多くない場合、JEUS MQサーバーはメッセージへの強参照を保持します。デスティネーション設定内に「Quota」値を指定すると、メモリー使用量に応じてJEUS MQサーバーは以下のように動作します。

- デスティネーションが使用しているメモリーが「Quota」の設定値以上の場合、クライアントのメッセージ送信が失敗し、JMSEExceptionが発生します。「Quota」のデフォルト値は128MBです。
- デスティネーションが使用しているメモリーが「Quota」設定値の75%以上の場合、ストレージに保存したメッセージの内容はメモリーで管理しません。

デスティネーションに到達したメッセージがメモリーから削除されても、メッセージが失われるわけではありません。メッセージはストレージに格納されているため、JEUS MQサーバーは必要な際にメッセージをストレージから読み取って処理します。この際の処理速度はメッセージがメモリー上にある場合より遅くなります。

参考

JEUS MQサーバーは、ストレージに格納していないメッセージは常にメモリーに保持しているため、「Quota」の設定はストレージが設定されており、メッセージをDeliveryMode.PERSISTENTに指定した場合にのみ影響を及ぼします。

メッセージ・フローの制御

Destination領域の「**Max Pending Limit**」、「**Resume Dispatch Factor**」項目はメッセージ・フローを制御します。メッセージ・コンシューマがデスティネーションにMessage Listenerを登録した場合、JEUS MQサーバーはデスティネーションにメッセージが到着次第、該当クライアントにメッセージを送信します。しかし、サーバーがメッセージを送信する速度よりクライアントでメッセージを処理する速度が遅い場合、クライアント・メモリーにメッセージが蓄積され、クライアントJVMでOutOfMemoryErrorが発生する場合があります。

この問題を回避するために、クライアント側に処理されずに蓄積されている最大メッセージ数を設定して、この値を超過するとサーバーはクライアントへのメッセージ送信を一時的に停止させることができます。

3.2.2. 永続サブスクライバーの設定

一般的に永続サブスクライバーはクライアントが作成するため、サブスクライバーを作成する前にトピックに届いたメッセージはクライアントに送信されません。このような問題を回避するために、JEUS MQは永続サブスクライバーを設定し、サーバーの起動時に予め永続的サブスクライバーを登録しておいて、条件に合うクライアントが接続したらメッセージを送信する機能を提供します。Session.createDurableSubscriber()メソッドを呼び出す場合と同様に、JEUS WebAdminで永続サブスクライバーを設定する場合も、クライアントID、サブスクリプション名、およびトピックを設定する必要があります。追加でMessage Selectorも設定できます。

Jms Resource画面の**Durable Subscriber**領域で設定する特定のリソース名をクリックすると、以下のよう設定画面が表示されます。各項目の情報を設定して**[確認]**ボタンをクリックすると、設定した情報が保存されます。項目を再設定するには**[再設定]**ボタンをクリックします。

[図 3.6] Durable Subscriber設定画面



Durable Subscriber

HISTORY

永続的サブスクライバの情報を設定します。

ヘルプ ?

Basic Resource Engine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 * 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

Client Id *	client_id クライアントを識別するIDを指定します。このIDはコネクションファクトリで一意である必要があり、永続的サブスクライバ項目のすべてのクライアントIDとも重複してはいけません。
Name *	durable この永続的サブスクライバを識別する値を設定します。
Shared	<input type="checkbox"/> [デフォルト: false] この永続的サブスクライバを複数のクライアントが共有して使用するかどうかを設定します。共有する場合、1つのメッセージは1つのクライアントにのみ送信されます。詳細については、JMS 2.0仕様の「8.3.4 Shared Durable Subscription」を参照してください。
Destination Name *	ExamplesTopic 永続的サブスクライバがメッセージを受信するデスティネーションの名前を指定します。
Message Selector	<input type="text"/> 永続的サブスクライバのメッセージセクタを設定します。
Message Sort	<input type="text"/> 永続的サブスクライバに適用するメッセージソートを設定します。

3.3. JMSエンジンの設定

Jms Engine画面ではJEUS MQサーバーについて設定します。各JMSエンジンの[Servers] > [サーバー名] > [Engine]タブで[Jms Engine]メニューを選択します。

3.3.1. 基本情報の設定

Jms Engine画面で[Basic]メニューを選択すると、JMSエンジンの基本情報設定画面に移動します。

JEUS MQサーバーのフェイルオーバーとJMSエンジンの詳細設定画面から構成されます。上段のフェイルオーバー設定では、サーバーやネットワークに障害が発生した場合、これを復旧する情報を設定します。フェイルオーバー設定についての詳細は、「[第5章 JEUS MQのフェイルオーバー](#)」を参照してください。

[図 3.7] Jms Engine設定画面



詳細設定

JMSエンジンの詳細設定画面では、JEUS MQサーバーが使用するスレッドプールについての情報を設定します。JMSの非同期な特性に従って、クライアントとメッセージを送受信するかサーバー内部でメッセージを処理するかして、ストレージに格納するなどの作業はそれぞれ別のスレッドで実行されます。

JEUS MQサーバー内で同時に実行可能なスレッドの数は、このスレッドプールの設定で決定されます。サーバーの性能に大きく影響を及ぼすため、同時に処理が可能なクライアントの数とサーバーに掛かる負荷などを考慮し、適切な設定が必要です。

以下は、**詳細設定**画面です。

[図 3.8] Jmsエンジンの詳細設定画面



3.3.2. サービス・チャンネルの設定

JEUS MQサーバーはサービス・チャンネルを通じてクライアントと通信します。1つのJEUS MQサーバーには少なくとも1つのサービス・チャンネルが必要であり、各サービス・チャンネルごとにサービスURLなどのネットワーク設定を別々に指定できます。

サービス・チャンネルが使用するリスナーの名前は**[Servers] > [サーバー名] > [Resource] > [Listener]**で設定します。リスナーの設定に関する詳細は、『JEUS サーバガイド』の「2.3.2. リスナーの設定」を参照してください。

[Servers] > [サーバー名] > [Engine] > [Jms Engine] > [Service Config]タブを選択すると、サービス・チャンネルで使用するリスナー一覧が表示されます。リスナー一覧でチャンネルを設定する対象を選択すると、以下の設定画面が表示されます。各項目の情報を設定して**[確認]**ボタンをクリックすると、設定した情報が保存されます。項目を再設定するには**[再設定]**ボタンをクリックします。

【図 3.9】 サービス・チャンネルの設定画面

Service Config HISTORY

メッセージングサービスを提供するためのサービスチャンネルを設定します。最低1つ以上設定する必要があります。ヘルプ ?

Basic Resource **Engine**

Web Engine | **Jms Engine** | Ejb Engine

Basic **Service Config** Connection Factory Persistence Store Message Sort

動的設定 * 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

Name *	default サービスチャンネルの名前を指定します。コネクションファクトリにチャンネル情報を指定するために使用されます。
Client Limit	1000 [デフォルト: 1000] サービスチャンネルが許容する最大クライアント数を指定します。同期ソケットを使用する場合は、メッセージブローカーに設定されている最大スレッド数を越えることができません。
Client Keepalive Timeout	30 S [デフォルト: 30] クライアントとの接続が異常終了した場合に、再接続するまでの待ち時間を秒単位で指定します。指定した時間が経過すると、該当クライアントのリソースはすべてサーバに返されます。この設定を行うと、指定した時間の間クライアントのclientID値が維持されるため、ネットワーク状態が良くない場合にのみ設定してください。秒単位で設定します。0以下の値を指定すると、再接続するまで待たずに即時リソースを返します。
Listener Name	jms サービスチャンネルのリスナーを指定します。サーバにすでに設定されている名前を指定します。設定していない場合は、Baseリスナーが使用されます。
Virtual Listener 仮想リスナーを指定します。外部環境の必要に応じて、実際に存在しないアドレスを指定する場合に使用します。	
Server Address *	EX 123.123.123.123 サービスチャンネルのIPアドレスを指定します。コネクションファクトリのアドレスとして使用されます。
Port *	EX 9741 サービスチャンネルのTCPポート番号を指定します。

3.3.3. コネクション・ファクトリーの設定

コネクション・ファクトリーはJMS管理オブジェクトで、クライアントがJMSサーバーに接続するのに必要な接続設定情報とクライアントのための基本情報を保持しています。JEUS MQサーバーの起動時に作成され、JEUS JNDIサービスに登録されます。

[Servers] > [サーバー名] > [Engine] > [Jms Engine] > [Connection Factory]タブを選択すると、コネクション・ファクトリー一覧が表示されます。一覧で設定する対象を選択すると、以下の設定画面が表示されます。各項目の情報を設定して[確認]ボタンをクリックすると、設定した情報が保存されます。項目を再設定するには[再設定]ボタンをクリックします。

[図 3.10] Connection Factory設定画面

Connection Factory

HISTORY

JMSコネクションファクトリーを設定します。

ヘルプ

BasicResourceEngine

Web EngineJms EngineEjb Engine

BasicService ConfigConnection FactoryPersistence StoreMessage Sort

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。TIP

Name	ConnectionFactory JMSシステム内で使用されるコネクションファクトリーの名前を指定します。
Type	nonxa [デフォルト: nonxa] コネクションファクトリーのタイプを指定します。
Service	 コネクションファクトリーが接続を試行するサービスの名前を指定します。サービスのコンフィグレーションに設定されている名前を使用します。名前を指定していない場合は、最初に設定されたサービスが任意で設定されます。
Export Name	ConnectionFactory ネーミングサーバーにバインディングされるコネクションファクトリーの名前を指定します。指定しない場合は、Name属性が使用されます。
Client Id	 コネクションファクトリーを使って作成されるコネクションにデフォルトで設定されるクライアントID値を指定します。
Server Selection Policy	Round-robin [デフォルト: Round-robin] コネクションファクトリーがコネクションを作成時にチャンネルを選択するポリシーを設定します。Round-robinとRandomのいずれかを選択できます。
Request Blocking Time	200 [デフォルト: 200] クライアントがサーバーに要求を送ってから、応答があるまで待機する時間を指定します。指定した時間が経過するまで応答がない場合は、JMS例外がクライアントに返され、ブロックが解除されます。単位は秒、デフォルト値は200です。
Reconnect Enabled	<input type="checkbox"/> [デフォルト: false] クライアントとサーバーの接続が切断された場合に、再接続するか否かを設定します。
Reconnect Period	0 [デフォルト: 0] クライアントとサーバーの接続が切断された場合に、再接続を試行する最大時間を指定します。指定した時間の間再接続に失敗すると、すべての要求は取り消され、JMS例外が返されます。この時間は、Request Blocking Timeより常に大きい必要があります。小さい場合は、Request Blocking Timeの値に代替されます。デフォルト値は0で、無限を意味します。
Reconnect Interval	5 [デフォルト: 5] クライアントとサーバーの接続が切断された場合に、再接続を試行する時間間隔を指定します。単位は秒、デフォルト値は5です。

3.3.4. 永続ストアの設定

永続ストアは、サーバーの再起動時にメッセージやサブスクリプション、トランザクションの以前の状態をリカバリーするために必要です。永続ストアを設定しないと、DeliveryMode.PERSISTENT方式でクライアントからメッセージを送信する際にサーバーで障害が発生した場合、メッセージの送信を保証できなくなります。JEUS MQで提供するパーシステンス・ストアには、ジャーナルログとデータベースがあります。

パーシステンス・ストアの設定は、WebAdminの**[Servers] > [サーバー名] > [Engine] > [Jms Engine] > [Persistence Store]**メニューを選択します。

[図 3.11] Persistence Store設定画面

Persistence Store

HISTORY

検索

印刷

KML

共有

永続的オブジェクトのためのストア情報を設定します。永続的オブジェクトには、デスティネーション、永続的サブスクリプション、メッセージ、サブスクリプション、トランザクションがあります。

ヘルプ?

BasicResourceEngine

Web Engine | Jms EngineEjb Engine

BasicService ConfigConnection FactoryPersistence StoreMessage Sort

動的設定必須項目このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

TIP

None

Journal

Base Dir

ストアを作成するディレクトリ名を指定します。ディレクトリ名はストアごとに一意である必要があります。

Initial Log File Count

[デフォルト: 2] ジャーナルストアを作成するとき、初期作成するログファイルの数を指定します。

Max Log File Count

[デフォルト: 20] 作成するログファイルの最大数を指定します。

Log File Size

[デフォルト: 128m] ログファイルのサイズを指定します。整数型の値を指定するか、または数字の後ろに「K」(Kilobytes)、「M」(Megabytes)、「G」(Gigabytes)をつけて設定できます。

Destination Table

ジャーナルストアを使用するように設定された場合にも、デスティネーション情報は組み込みDBに保存されます。このときに使用するテーブルの名前を設定します。

Durable Subscription Table

ジャーナルストアを使用するように設定された場合にも、永続的サブスクリプション情報は組み込みDBに保存されます。このときに使用するテーブルの名前を設定します。

Property

key=value

ストアを作成するとき、追加設定が必要な場合に設定します。

Jdbc

Data Source

データベースのデータソースを設定します。

Destination Table

デスティネーションテーブルの名前を変更します。

Durable Subscription Table

永続的サブスクリプションテーブルの名前を変更します。

Message Table

メッセージテーブルの名前を変更します。

Subscription Message Table

サブスクリプションテーブルの名前を変更します。

Transaction Table

トランザクションテーブルの名前を変更します。

ジャーナルログ方式を使用するには「**Journal**」を選択し、データベース方式を使用するには「**Jdbc**」を選択して設定します。

● ジャーナルログ方式

ジャーナルログ方式でパーシステンス・ストアを使用する場合は以下のように設定します。

[図 3.12] 永続ストアの設定 - Journal

Journal	
Base Dir	<input type="text" value="/home/jeus/store/jeusmq"/> ストアを作成するディレクトリ名を指定します。ディレクトリ名はストアごとに一意である必要があります。
Initial Log File Count	<input type="text" value="2"/> [デフォルト: 2] ジャーナルストアを作成するとき、初期作成するログファイルの数を指定します。
Max Log File Count	<input type="text" value="20"/> [デフォルト: 20] 作成するログファイルの最大数を指定します。
Log File Size	<input type="text" value="128m"/> [デフォルト: 128m] ログファイルのサイズを指定します。整数型の値を指定するか、または数字の後ろに「K」(Kilobytes)、「M」(Megabytes)、「G」(Gigabytes)をつけて設定できます。
Destination Table	<input type="text"/> ジャーナルストアを使用するように設定された場合にも、デスティネーション情報は組み込みDBに保存されます。このときに使用するテーブルの名前を設定します。
Durable Subscription Table	<input type="text"/> ジャーナルストアを使用するように設定された場合にも、永続的サブスクリプション情報は組み込みDBに保存されます。このときに使用するテーブルの名前を設定します。
Property	<div><input type="text" value="key=value"/></div> ストアを作成するとき、追加設定が必要な場合に設定します。

● データベース方式

以下は、外部データベースをパーシステンス・ストアに設定した例です。この設定は、ジャーナル設定の下段に位置します。

[図 3.13] 永続ストアの設定 - Jdbc

Jdbc	
Data Source *	<input type="text" value="datasource1"/> データベースのデータソースを設定します。
Destination Table	<input type="text" value="JEUSMQ_DEST"/> デスティネーションテーブルの名前を変更します。
Durable Subscription Table	<input type="text" value="JEUSMQ_DSUB"/> 永続的サブスクリプションテーブルの名前を変更します。
Message Table	<input type="text" value="JEUSMQ_MESG"/> メッセージテーブルの名前を変更します。
Subscription Message Table	<input type="text" value="JEUSMQ_SMSG"/> サブスクリプションテーブルの名前を変更します。
Transaction Table	<input type="text" value="JEUSMQ_TRAN"/> トランザクションテーブルの名前を変更します。

現在互換可能な外部データベースは以下のとおりです。

- Oracle Database 9i以上(Enterprise Edition)
- Tiberio 3.0 SP2以上
- Altibase 4.x(5以上はサポートしていません)

「**Data Source**」項目に設定された値はパーシステンス・ストアとして使用するデータソースのJNDI名です。JEUSにデータソースを追加する方法は、『*JEUS サーバガイド*』の「第6章 DBコネクション・プールとJDBC」を参照してください。

上記のように設定した場合、該当データベース内の表の名前を変更できます。このタグを使用すると、運用中のサーバーでテストを実行した場合、別のパーシステンス・ストアを設置しなくても、使用する表の名前のみ変更してJEUS MQサーバーをテストすることが可能です。表の名前を設定していない場合、基本的に<SERVER-NAME>_DEST, <SERVER-NAME>_DSUB, <SERVER-NAME>_MESG, <SERVER-NAME>_SMSG, <SERVER-NAME>_TRANという名前の表を使用します。

JDBCを設定した際に作成されるテーブルの各列についての説明は、「[付録 B. JDBC永続ストアの列](#)」を参照してください。

3.3.5. メッセージのソート設定

メッセージのソートに関する設定は、「[6.2. JEUS MQのメッセージ・ソート](#)」の説明を参照してください。

3.4. サーバー管理およびモニタリング

JEUS MQサーバー管理は、以下の作業を通じてメッセージの損失なく持続的なサービスを可能にすることを言います。

- 運用中のJEUS MQサーバーのリソース管理およびモニタリング

以下のリソースが管理およびモニタリングの対象となります。

- コネクション・ファクトリー、デスティネーションなどのJMS管理オブジェクト
- デスティネーションと持続的サブスクリプション上のメッセージ
- クライアントに割り当てられたコネクション、セッション、メッセージのプロデューサーとコンシューマ
- JEUS MQサーバーが占めるメモリ領域
- JEUS MQサーバーが使用するパーシステント・ストレージ

- 障害への対応とJEUS MQサーバーの復旧

JEUSはコンソール・ツールのjeusadminとWebAdminを提供し、JEUS MQサーバー管理およびモニタリング機能を提供します。本章では、各ツールを使用してJEUS MQサーバーを管理およびモニタリングする方法について説明します。

3.4.1. サーバー管理

以下は、コンソール・ツール(jeusadmin)とWebAdminを使用してサーバーを管理する方法について説明します。

WebAdminの使用

WebAdminを使用して、JMSに必要なコネクション・ファクトリー、デスティネーション、永続的サブスクライバーを管理できます。各一覧で[Add]、[Delete]ボタンを使用してコネクション・ファクトリーを追加および削除できます。

- コネクション・ファクトリー関連

[Servers] > [サーバー名] > [Engine] > [Jms Engine] > [Connection Factory]メニューを選択すると、設定されたコネクション・ファクトリー一覧を表示できます。

[図 3.14] コネクション・ファクトリー管理

Connection Factory		
Name	Type	Add 
ConnectionFactory	NONXA	Delete 
XAConnectionFactory	XA	Delete 

- デスティネーション関連

[Servers] > [サーバー名] > [Resource] > [Jms Resource]メニューを選択すると、設定されたデスティネーション一覧を表示できます。

[図 3.15] デスティネーション管理

Destination		
Name	Type	Add 
ExamplesQueue	QUEUE	Delete 
ExamplesTopic	TOPIC	Delete 

- 永続サブスクライバー関連

[Servers] > [サーバー名] > [Resource] > [Jms Resource]メニューを選択すると、設定された永続サブスクライバー一覧を表示できます。

[図 3.16] 永続サブスクライバー管理

Durable Subscriber			
Client ID	Name	Destination Name	Add 
client_id	durable	ExamplesTopic	Delete 

コンソール・ツールの使用

コンソールで複数のコマンドを実行して管理作業を行うことができるツールです。以下のパスにツールが位置します。

```
JEUS_HOME/bin/
```

以下は、サーバーを管理するために提供されるコマンド一覧です。

- コネクション・ファクトリー関連

コマンド	説明
create-jms-connection-factory	コネクション・ファクトリーを追加します
delete-jms-connection-factory	コネクション・ファクトリーを削除します

- デスティネーション関連

コマンド	説明
create-jms-destination	デスティネーションを追加します
delete-jms-destination	デスティネーションを削除します

参考

コマンドの使用方法、使用例については、『*JEUS リファレンスガイド*』の「4.2.10. JMSエンジン関連コマンド」を参照してください。

3.4.2. サーバー・モニタリング

本節では、コンソール・ツールとWebAdminを使用してサーバーをモニタリングする方法について説明します。

WebAdminの使用

メニューから[Monitornig] > [JMS]を選択すると、JEUS MQサーバーの現在の状態を表示します。

- デスティネーション関連

[Destinations]タブをクリックし、[Server]または[Cluster]メニューを選択します。一覧からサーバーやクラスターを選択すると、現在のサーバーあるいはクラスターにあるデスティネーション一覧と情報が表示されます。

デスティネーションの名前をクリックすると、該当デスティネーションの詳細情報が表示されます。デスティネーションの一覧とメッセージ関連モニタリング機能についての詳細は、「[6.5. JEUS MQのメッセージ管理機能](#)」を参照してください。

[図 3.17] デスティネーションのモニタリング

The screenshot shows the 'Destinations' page in the WebAdmin interface. At the top, there's a 'HISTORY' dropdown and a search bar. Below that, a message states: '現在サーバまたはクラスターに存在するデスティネーション情報を照会します。クラスターの情報は各サーバの情報をまとめたものです。' (Retrieve destination information existing on the current server or cluster. Cluster information is a summary of information from each server). There are tabs for 'Destinations', 'Durable Subscriptions', 'JMS Clients', and 'JMS Pending Transactions'. Below the tabs, there's a dropdown for 'Server' and a text input for 'Cluster'. The selected server is 'adminServer'. The main section is titled 'Destination information in Server adminServer'. It contains a table with the following data:

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	Import Browse
ExamplesQueue	ExamplesQueue	Queue	0	JEUSMQ_DLQ	false	false	Import Browse
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	Import Browse

- 永続サブスクライバー関連

[Durable Subscriptions]タブをクリックし、サーバーを選択すると、現在のサーバーが存在する永続的サブスクライバー一覧と情報を表示します。

[図 3.18] 永続サブスクライバーのモニタリング

Durable Name	Client ID	Message Selector	Remaining Messages
durable	client_id		0

Browse

- クライアント関連

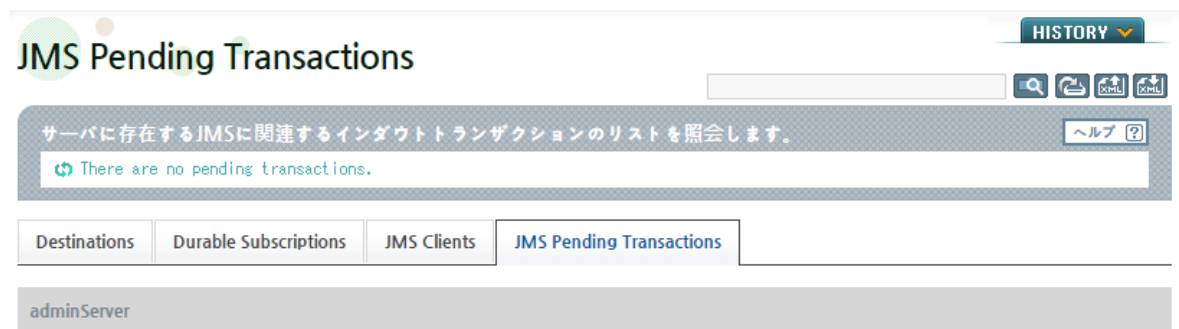
[JMS Clients]タブをクリックし、サーバーを選択すると、現在のサーバーが接続しているJMSクライアントの一覧と情報を表示します。

[図 3.19] クライアントのモニタリング

Entry Name	Remote Address	Start Time	Connection Count	Session Count	Command
該当する内容が存在しません。					

- トランザクション関連

[図 3.20] JMSベンディング・トランザクションのモニタリング



コンソール・ツールの使用

コンソール・ツールは、コンソールで様々なコマンドを実行してサーバー管理作業を行います。以下のパスにツールが位置します。

```
JEUS_HOME/bin/
```

以下は、サーバーをモニタリングするために提供されるコマンド一覧です。

- コネクション・ファクトリー関連

コマンド	説明
list-jms-connection-factories	コネクション・ファクトリー一覧を表示します。あるいは、指定されたコネクション・ファクトリーの情報を出力します

- デスティネーション関連

コマンド	説明
list-jms-destinations	デスティネーション一覧を表示します。あるいは、指定されたデスティネーション情報を出力します
list-jms-messages	指定されたデスティネーション内のメッセージ情報を表示します

- メッセージ関連

コマンド	説明
list-jms-messages	指定のデスティネーション内のメッセージの情報を表示します
view-jms-message	指定されたメッセージの詳細情報を表示します

コマンド	説明
move-jms-messages	指定されたメッセージをサーバーあるいはクラスター内の他のデスティネーションに移動します
delete-jms-messages	指定されたメッセージをデスティネーションから削除します
export-jms-messages	指定されたメッセージをXMLファイル形式でエクスポートします
import-jms-messages	エクスポートされたXMLファイルのメッセージを指定されたデスティネーションにインポートします

- 永続サブスクライバー関連

コマンド	説明
list-jms-durable-subscriptions	永続サブスクライバー一覧を表示します。あるいは、指定された永続的サブスクライバー情報を表示します

- クライアント関連

コマンド	説明
list-jms-clients	クライアント一覧を表示し、情報を出力します
ban-jms-client	クライアントからの接続を強制的に切断します

- トランザクション関連

コマンド	説明
list-jms-pending-transaction	ペンディング・トランザクション一覧を出力します
commit-jms-pending-transaction	指定されたペンディング・トランザクションを強制的にコミットします

参考

コマンドの使用方法和使用例については、『*JEUS リファレンスガイド*』の「4.2.10. JMSエンジン関連コマンド」を参照してください。

第4章 JEUS MQクラスタリング

本章では、JEUS MQサーバーの負荷を減らし、円滑なサービスを行うために、複数台のサーバーを1つにまとめて使用するクラスタリングについて説明します。

4.1. 概要

1台のJEUS MQサーバーでサービスを提供する場合に、サーバーを利用するクライアントの接続が多い場合や、サーバーに保管されているメッセージの量が非常に多い場合、ネットワークの負担が重くなったり、サーバーのメモリー使用量が増加します。

これは、全体的な性能低下やサーバーがダウンする原因になる可能性があります。このような場合、ネットワークまたはメモリーの負荷を分散し、全体的な性能を維持するために、JEUS MQサーバーを増設し、MQサーバー全体を1台のサーバーのように動作するようにすることをJEUS MQクラスタリングといいます。

この際、JEUS MQクライアントはクライアント自身が接続しようとするJEUS MQサーバーがクラスタリングになっているか否かとは関係なく、単一サーバーを利用する場合と同じ方法で接続できます。

また、JEUS MQクラスタリング機能には、JEUS MQのフェイルオーバー概念が含まれています。詳しい内容は、「[第5章 JEUS MQのフェイルオーバー](#)」を参照してください。

4.2. クラスタリングの種類

クラスタリングは、コネクション・ファクトリー・クラスタリングとデスティネーション・クラスタリングの2つに分けられます。

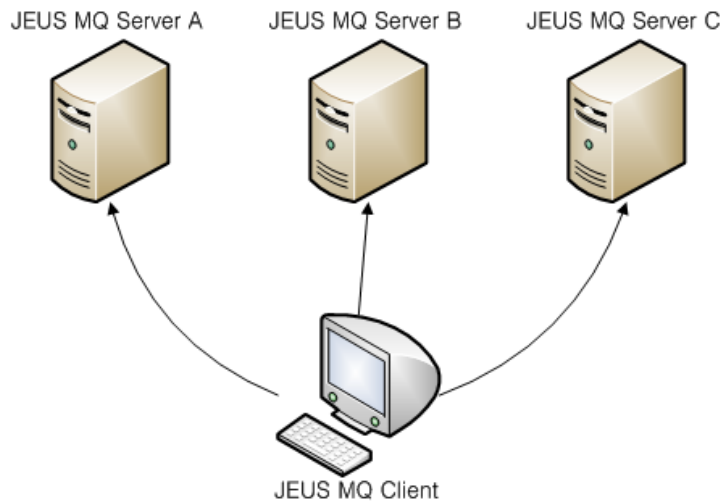
4.2.1. コネクション・ファクトリー・クラスタリング

2台以上のJEUS MQサーバーを構成した場合、クライアントからこれらのサーバーに接続するコネクションを適切に分散させる必要があります。クライアントからサーバーへの接続はコネクション・ファクトリーを通じて行われるため、コネクション・ファクトリーでこのような分散機能をサポートする必要があります。これを**コネクション・ファクトリー・クラスタリング**といいます。

コネクション・ファクトリーからコネクションを作成する際、コネクションごとに異なるサーバーに接続し、毎回異なるサーバーを選択するためのポリシーを設定できます。現在はラウンド・ロビン方式のみサポートしています。

以下の図は、コネクション・ファクトリー・クラスタリングを示しています。

[図 4.1] コネクション・ファクトリー・クラスタリング



参考

1つのJEUS MQクライアントは、クラスター内の1つのサーバーに接続され、クライアントはどのサーバーに接続されるのかを考慮する必要があります。

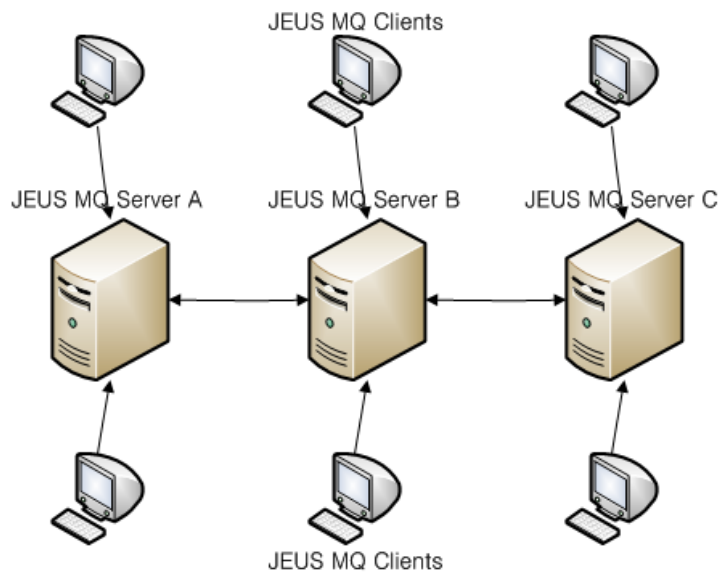
4.2.2. デスティネーション・クラスタリング

コネクション・ファクトリー・クラスタリングによってクライアント接続が適切に分散されても、クライアントの処理速度の違いやネットワーク環境によって、クライアントが接続されたJEUS MQサーバーのメッセージをすべて消費して、それ以上受信できない場合があります。また、サーバーにメッセージが蓄積する速度に比べて消費する量が少ない場合、メッセージが蓄積し続ける現象が起きることがあります。

このような状況では、残っているメッセージを、メッセージが不足するJEUS MQサーバーに移動させ、サービスが持続的に行われるようにする必要があります。このような機能を**デスティネーション・クラスタリング**といいます。

以下の図は、デスティネーション・クラスタリングを示しています。

【図 4.2】 デスティネーション・クラスタリング



4.3. クラスタリングの使用

本節では、JEUS MQクラスタリングを使用するために必要な設定について説明します。

4.3.1. サーバー設定

JEUS MQクラスタリングの全体的な構成は、JEUSサーバーのクラスタリングに従います。JEUSサーバーのクラスタリングに関する詳細は、『*JEUS ドメインガイド*』の「第5章 JEUSクラスタリング」を参照してください。

JEUS MQクラスタリング設定の注意事項

JEUS MQクラスタリングに参加するJEUS MQサーバーのデスティネーション、永続的サブスクライバー設定はWebAdminの[Clusters]メニューに設定して、それに含まれている各サーバーに同様に適用されます。サーバーごとにアドレスが異なるため、別途設定が必要なコネクション・ファクトリーはWebAdminの[Servers]メニューで設定します。ただし、この場合も名前は同一に設定する必要があります。

クラスターにデスティネーションまたは永続的サブスクライバーを設定するために[Clusters] > [クラスター名] > [Engine]タブの[Jms Resource]を選択すると、以下のような設定画面が表示されます。それぞれのデスティネーションや永続的サブスクライバーを設定する方法はサーバーに設定する方法と同じであるので、[「3.2. JMSリソースの設定」](#)の説明を参照してください。

【図 4.3】 クラスター内のデスティネーションの設定例



参考

1. クラスターにサーバーを追加する方法については、『JEUS ドメインガイド』の「5.6.1. クラスターにサーバーを追加」を参照してください。
2. クラスター内のデスティネーションを設定する場合、サーバーに設定されていたデスティネーションや永続サブスクライバーの設定は無視されますが、明確な動作のために重複した名前は使用しないようにしてください。

4.3.2. クライアントの設定

JEUS MQクラスタリングを使用するには、クライアントで考慮が必要な事項が存在します。クライアントでJEUS MQクラスタリングを利用するためにまず設定するのはコネクション・ファクトリーであるため、主にこれについて説明します。

JEUS MQのコネクション・ファクトリー・クラスタリングは、クライアントがコネクション・ファクトリーを取得する方法によって以下の2つに分けられます。

- JNDIサービスを利用する方法
- JEUS MQ専用APIを使用する方法

JNDIサービスを利用する方法

JEUS MQサーバーのクラスタリング可否と関係なく、「2.2.2. コネクション・ファクトリー」の説明と同じ方法で使えます。ただし、クライアントが毎回JNDIサービスを使用してコネクション・ファクトリーを再度ルックアップするのか、既を取得したコネクション・ファクトリーを再活用してコネクションの作成のみ行うのかによって、

動作の主体が若干異なります。また、動作の主体によってJEUS MQサーバーを選択するポリシーも異なってきます。

- 毎回JNDIサービスを利用して新しいコネクション・ファクトリーを取得する場合は、JEUS MQサーバーを選択するポリシーをJNDIサービスに依存します。この際は、JEUSクラスタリングが設定されている必要があります。

これに関する内容とJNDIクラスタリングを使用したクライアント・プログラム方法については、『*JEUSドメインガイド*』の「第5章 JEUSクラスタリング」および『*JEUS サーバガイド*』の「4.5. JNDIのプログラミング」を参照してください。

- 既に取得したコネクション・ファクトリーを再活用してコネクションの作成のみ再度行う場合は、JEUS MQサーバーの中から接続場所を選択するポリシーを設定できますが、現在はラウンド・ロビン方式のみサポートしています。

JEUS MQ専用のAPIを使用する方法

JEUS MQ専用のAPIを使用してコネクション・ファクトリーを直接作成する場合、JEUS MQサーバーがクラスタリングされていれば、「[2.2.2. コネクション・ファクトリー](#)」で説明した方法に追加作業が必要です。

以下は、`JeusConnectionFactoryCreator.addBrokerAddress()`メソッドを利用してクラスタリングに参加しているすべてのサーバーの情報を追加する方法です。

```
jeus.jms.client.util.JeusConnectionFactoryCreator connectionFactoryCreator =
new jeus.jms.client.util.JeusConnectionFactoryCreator();

connectionFactoryCreator.setFactoryName("ConnectionFactory");
connectionFactoryCreator.addBrokerAddress("192.168.1.2", 9741, <service-name>);
connectionFactoryCreator.addBrokerAddress("192.168.1.3", 9741, <service-name>);
connectionFactoryCreator.addBrokerAddress("192.168.1.4", 9741, <service-name>);

ConnectionFactory connectionFactory =
connectionFactoryCreator.createConnectionFactory();
```

4.4. 使用例

本節では、JEUS MQクラスタリングの例において、一般的な使用例と誤った使用例について説明します。

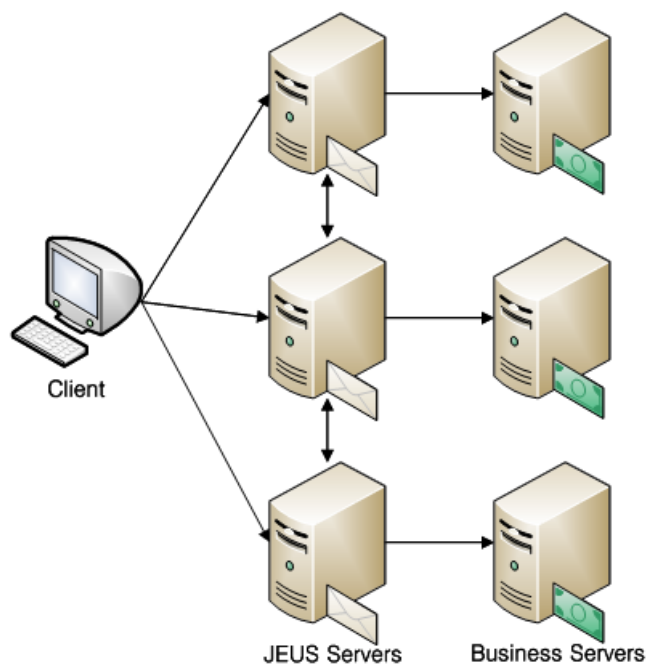
4.4.1. 一般的な使用例

JEUS MQを使用して商品の注文を処理するショッピング・モールを仮定します。

Webページで作成された注文はメッセージとして作成されて、クラスタリングで構成されているJEUS MQのキューに保管され、実際に注文を処理するビジネス・サーバーからこのメッセージを取り出します。そして、このJEUS MQはクラスタリングで構成します。

ビジネス・サーバーはそれぞれ1つずつ対応するJEUS MQサーバーからメッセージを取り出し、注文メッセージは各JEUS MQサーバーに均等に蓄積するとすれば、この構造はJEUS MQサーバーの理想的な構成例となります。特定ビジネス・サーバーの処理速度が速く、それに対応するJEUS MQサーバーのキューが他の所より先に空になれば、そのJEUS MQサーバーはデスティネーション・クラスタリングを通じて他のJEUS MQサーバーからメッセージを受け取り、該当ビジネス・サーバーにメッセージを提供します。

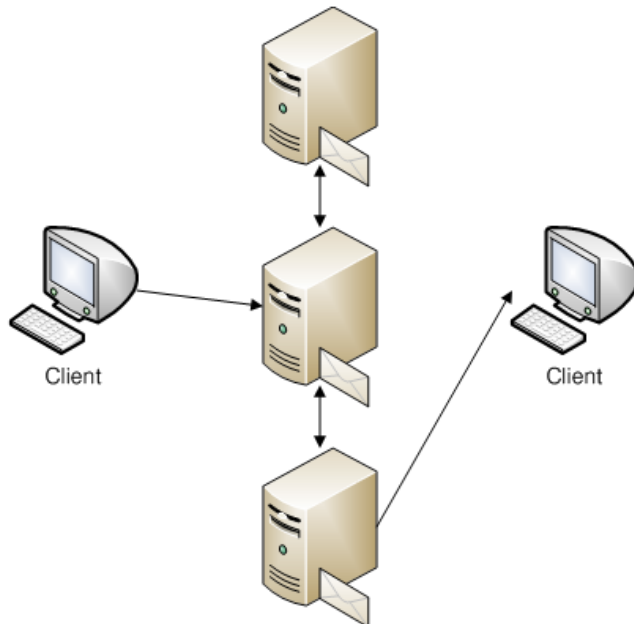
[図 4.4] JEUS MQクラスタリング構成の良い例



4.4.2. 正しくない使用例

JEUS MQクラスタリングの構成方法は非常に多様ですが、以下のような構造で使用すると効率が落ちることがあります。

[図 4.5] JEUS MQクラスタリング構成の悪い例



JEUS MQクラスタリングを構成したものの、使用するJMSクライアントが1度接続したコネクションを再活用し続けるようになると、コネクションの負荷分散が正常に動作せず、JEUS MQサーバー間で不要なメッセージ移動が起こるようになり、効率が大きく低下します。

第5章 JEUS MQのフェイルオーバー

本章では、JEUS MQでサーバーやネットワークに障害が発生し、それ以上メッセージング・サービスを行うことができない場合、JMSクライアントが再接続し、クライアント状態を復旧する方法について説明します。また、JMSクライアントの復旧のために必要なサーバー構成とサーバー障害の復旧について説明します。

5.1. 概要

JEUS MQでのフェイルオーバーとは、障害が発生した場合にクライアント・アプリケーションが自動接続を再度行い、クライアントを障害の発生前の状態に戻すことをいいます。

障害が発生する原因は、以下のように大きく2つに分けられます。

- ネットワーク障害

ネットワーク障害とは、JEUS MQサーバーとクライアント間のネットワークに問題が発生し、それ以上通信できない状態を指します。このネットワーク障害は一時的な障害であることもあり、サーバーがダウンした場合やネットワークが完全に使用不可能な場合も含まれます。

ネットワーク障害が発生した場合、JEUS MQクライアントはネットワークに障害が発生したサーバーや該当サーバーとクラスタリング関係にある他のサーバーに再接続を試行し、再接続されるとクライアントの状態を自動復旧し、サービスを継続して行います。

- サーバー障害

サーバー障害は、ネットワーク障害を除き、サーバーで発生するすべての障害を含みます。

一般的に、メッセージング・データを格納するディスクやデータベース作業を行う際に発生するエラーやメモリ不足などがこれに該当します。現在サービス中のサーバーに障害が発生した場合、待機中のバックアップ・サーバーは以前使用していたデータを自動復旧し、サービスを継続して行います。

このような障害から復旧するためには、JEUS MQサーバー間にネットワークを構成し、JEUS MQクライアントに必要な設定を行います。そして、クライアントはフェイルオーバーのためにJEUS MQで提供するクライアントAPIを呼び出し、フェイルオーバーに必要な様々な属性を直接設定できます。

5.2. サーバー障害の復旧

本節では、JEUS MQのフェイルオーバーを使用するために必要なサーバー間のネットワークの構成とその他の設定について説明します。

5.2.1. ネットワークの構成

JEUS MQのフェイルオーバーのためには、1台以上のアクティブ・サーバーが必要であり、そのサーバーはクラスタリングされている必要があります。スタンバイ・サーバーはオプションとして、障害が発生した際に全体的な処理可能容量に余裕を持たせるために設定します。JEUSのクラスタリング設定は、『JEUS ドメインガイド』の「第5章 JEUSクラスタリング」を参照してください。

- アクティブ・サーバー

正常時にクライアントのリクエストを受けて処理するメイン・サーバーです。

- スタンバイ・サーバー

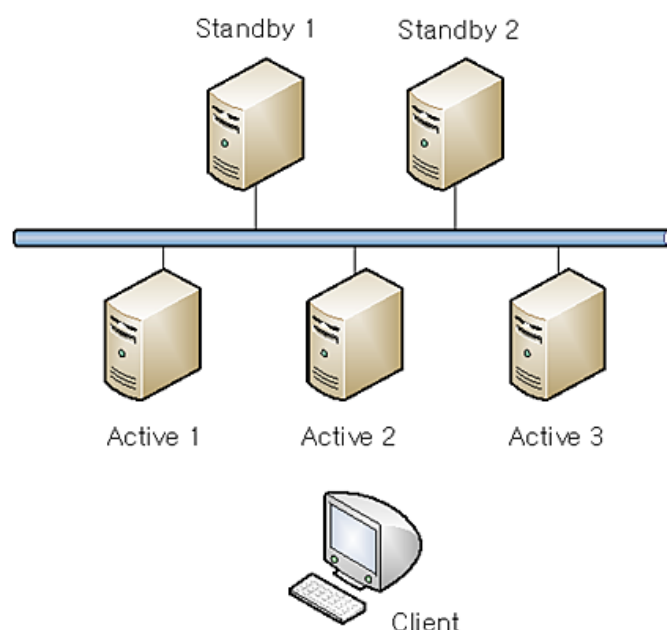
アクティブ・サーバーに障害が発生した場合、アクティブ・サーバーが行っていたサービスを引き継いで行うバックアップ・サーバーです。

JEUS MQのクラスタリングとJEUS MQのフェイルオーバーは統合された機能であり、JEUS MQのクラスタリングを設定すると、JEUS MQのフェイルオーバー機能も同時に動作します。

これにより、アクティブ・サーバーとスタンバイ・サーバーを必ずペアで構成する必要がなくなり、多様な構成が可能になります。また、アクティブ・サーバーの数とスタンバイ・サーバーの数も自由に設定できます。一般的に、多数のアクティブ・サーバに対し、少数のスタンバイ・サーバーを構成するか、あるいは、スタンバイ・サーバーなしでアクティブ・サーバーだけで構成します。

以下は、フェイルオーバーのためのMQサーバー間のネットワーク構成です。

【図 5.1】 3台のアクティブ・サーバーと2台のスタンバイ・サーバーを利用したJEUS MQクラスタリングの構成



上記のような構成で、アクティブ・サーバーのうち1つに障害が発生した場合、まずサービスを行っていないスタンバイ・サーバーのうち1つが接続を引き継いでサービスを行います。

サービスを実行しているアクティブ・サーバーとスタンバイ・サーバーのうち1つに追加で障害が発生した場合も同様に、サービスを行っていないスタンバイ・サーバーのうち1つが引き継ぎ、サービスを実行していないサーバーがない場合はサービスを実行しているアクティブ・サーバーのうち1つが引き継いで、1つのサーバーが2つ以上のサーバーの役割をします。

これは、最後の1つのサーバーのみ残るまで継続され、最後に残ったサーバーにも障害が発生した場合は JEUS MQクラスタリング・サービスは動作しなくなります。

アクティブ・サーバーとスタンバイ・サーバーの構成

以下は、アクティブ・サーバーとスタンバイ・サーバーを構成する例です。WebAdminで[Servers]>[サーバー名]>[Engine]>[Jms Engine]>[Basic]メニューを選択すると、JMSフェイルオーバーの設定画面が表示されます。

[図 5.2] JMSフェイルオーバーの設定画面

jeus_domain

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

2 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

JMS Engine

HISTORY

JMSエンジンは、該当するサーバでJMSサーバを使用するための環境を提供します。サーバの起動時に実行され、1つのサーバには1つのJMSエンジンのみサポートされます。

ヘルプ

Basic Resource Engine

Web Engine | Jms Engine | Ejb Engine

Basic Service Config Connection Factory Persistence Store Message Sort

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。TIP

Engine Roll	Active [デフォルト: Active] このJMSブローカーの役割を設定します。ActiveとStandbyを設定できます。アクティブブローカーは平常時に起動されてサービスを行い、スタンバイブローカーはアクティブブローカーが障害時に起動されてサービスを引き継ぎます。デフォルト値はActiveです。
Failover Check Timeout	5 [デフォルト: 5] 障害を検知しフェイルオーバーする前に、その対象となるJMSブローカーの生存の可否を別途の手段で確認する時間を設定します。この時間は一度の試行にかかる時間です。単位は秒、デフォルト値は5です。
Failover Check Count	0 [デフォルト: 0] 障害を検知しフェイルオーバーする前に、その対象となるJMSブローカーの生存の可否を別途の手段で確認する最大回数を設定します。設定した回数を試行しても、ブローカーの生存が確認されない場合は、障害が発生したと判断し、フェイルオーバーを開始します。デフォルト値は0であり、障害を検知すると、即時フェイルオーバーを開始します。

- アクティブ・サーバーの設定

アクティブ・サーバーの設定は、以下のように「Active」を選択します。

[図 5.3] フェイルオーバー設定 - アクティブ・サーバーの設定

The screenshot shows the JMS Engine configuration interface. The 'Engine' tab is selected, and the 'Jms Engine' sub-tab is active. The 'Basic' sub-tab is also selected. The 'Engine Roll' dropdown is set to 'Active'. The 'Failover Check Timeout' is set to 5 seconds. The 'Failover Check Count' is set to 0. A message at the top states: 'JMSエンジンは、該当するサーバでJMSサーバを使用するための環境を提供します。サーバの起動時に実行され、1つのサーバには1つのJMSエンジンのみサポートされます。' (JMS Engine provides the environment for using JMS server on the corresponding server. It is executed at the start of the server, and only one JMS engine is supported per server.)

Engine Roll	Active
Failover Check Timeout	5
Failover Check Count	0

- スタンバイ・サーバーの設定

スタンバイ・サーバーの設定は、画面で以下のように「Fail Over」をチェックして「Standby」を選択します。

[図 5.4] フェイルオーバー設定 - スタンバイ・サーバーの設定

The screenshot shows the JMS Engine configuration interface. The 'Engine' tab is selected, and the 'Jms Engine' sub-tab is active. The 'Basic' sub-tab is also selected. The 'Engine Roll' dropdown is set to 'Standby'. The 'Failover Check Timeout' is set to 5 seconds. The 'Failover Check Count' is set to 0. A message at the top states: 'JMSエンジンは、該当するサーバでJMSサーバを使用するための環境を提供します。サーバの起動時に実行され、1つのサーバには1つのJMSエンジンのみサポートされます。' (JMS Engine provides the environment for using JMS server on the corresponding server. It is executed at the start of the server, and only one JMS engine is supported per server.)

Engine Roll	Standby
Failover Check Timeout	5
Failover Check Count	0

5.2.2. コネクション・ファクトリーの設定

コネクション・ファクトリーの設定は、JEUS MQに障害が発生した際にクライアントが他のアクティブ・サーバーやスタンバイ・サーバーに試行する再接続に関連した設定です。

WebAdminで[Servers] > [サーバー名] > [Engine] > [Jms Engine] > [Connection Factory]メニューを選択します。コネクション・ファクトリー一覧で設定対象をクリックすると、設定画面が表示されます。

【図 5.5】コネクション・ファクトリーの設定

Reconnect Enabled	<input type="checkbox"/> [Default: false] クライアントとサーバの接続が切断された場合に、再接続するか否かを設定します。
Reconnect Period	0 ms [Default: 0] クライアントとサーバの接続が切断された場合に、再接続を試行する最大時間を指定します。指定した時間の間再接続に失敗すると、すべての要求は取り消され、JMS例外が返されます。この時間は、Request Blocking Timeより常に大きい必要があります。小さい場合は、Request Blocking Timeの値に代替されます。デフォルト値は0で、無限を意味します。
Reconnect Interval	5 s [Default: 5] クライアントとサーバの接続が切断された場合に、再接続を試行する時間間隔を指定します。単位は秒、デフォルト値は5です。

基本的には再接続を行わないため、画面で「**Reconnect Enabled**」をチェックして「true」に設定します。「**Reconnect Enabled**」が活性化されると、JEUS MQクライアントはアクティブ・サーバーとスタンバイ・サーバーに繰り返し再接続を試行します。

5.2.3. 永続ストアの設定

永続ストア(Persistence Store)は、DeliveryModeがPERSISTENTの場合にメッセージを保存する役割をします。

障害が発生した場合、他のアクティブ・サーバーあるいはスタンバイ・サーバーがこの永続ストアに保存されているメッセージを復旧するため、メッセージの紛失なくサービスを継続できます。したがって、JEUS MQサーバーのフェイルオーバーで最も中心となるリソースです。

JEUS MQのフェイルオーバーで永続ストアを設定するためには、アクティブ・サーバーとスタンバイ・サーバーの両方が接続可能な場所に永続ストアが位置する必要があります。

● ジャーナルログの永続ストア

ジャーナルログを永続ストアとして使用する場合は、ジャーナルログ・ベース・ディレクトリー(ジャーナルログの設定の「**Base Dir**」の値)は、アクティブ・サーバーとスタンバイ・サーバーの両方が接続可能な場所に位置する必要があります。このためには、SANのようなディスク共有ハードウェアを構成し、それ以外にジャーナル・ログのベース・ディレクトリーを設定します。

● JDBCの永続ストア

JDBCを永続ストアとして使用する場合は、WebAdminに設定されたデータソースを指定します。ただし、データベースも障害が発生してサービスが不可能な場合があるため、TiberoのTACやOracleのRACと

いったクラスタリング技術を使用して、データベースもフェイルオーバーのための構成をする必要があります。

参考

一部のサーバー間で永続ストアに互いにアクセスできない場合は、できる限りアクセス可能なサーバーを探してフェイルオーバーを試行します。

5.2.4. フェイルバック

アクティブ・サーバーの障害によって、他のサーバーやスタンバイ・サーバーのフェイルオーバー機能が活性化した場合、サーバー管理者はアクティブ・サーバーに発生した障害が何かを把握し、できる限り早く障害の原因を解決してアクティブ・サーバーを再起動します。

障害発生していたアクティブ・サーバーが再起動すると、該当サーバーのサービスを代わりに行っていたサーバーからデータをマイグレーションして、接続しているクライアントをアクティブ・サーバーに移す必要があります。この作業をフェイルバックといいます。この作業は常に自動で行われます。

5.3. クライアント障害の復旧

サーバーやネットワークに障害が発生し、JEUS MQクライアントとサーバー間の接続が切断されると、クライアントはクラスター内のアクティブ・サーバーとスタンバイ・サーバーを交互に再接続します。再接続されると、クライアントは接続が切断される前の状態への復旧を行います。このようなクライアントのフェイルオーバーは、クライアント・アプリケーションを修正せずに、JEUS MQの設定によって自動的に行われます。

本節では、クライアントのフェイルオーバーについて詳しく説明し、それに基づく制約事項とメッセージが紛失することなく障害から復旧するための方法について記述します。

5.3.1. 再接続

JEUS MQクライアントとサーバーの接続が切断された場合、再接続の可否を決定する「**Reconnect Enabled**」と関連する設定項目は、該当コネクション・ファクトリーを利用して設定するすべてのコネクションに適用されます。詳細については、「[5.2.2. コネクション・ファクトリーの設定](#)」を参照してください。

特定コネクションに対してこの設定を変更したい場合、以下の例のようにJEUS MQクライアントAPIである「`jeus.jms.client.facility.connection.JeusConnection`」クラスを利用して設定を変更できます。

```
...
import jeus.jms.client.facility.connection.JeusConnection;
...
Context ctx = new InitialContext();
ConnectionFactory factory = ctx.lookup("connection-factory");
```

```

JeusConnection connection = (JeusConnection)factory.createConnection("jeus",
"jeus");
connection.setReconnectEnabled(true);
connection.setReconnectInterval(1000); // 1秒
connection.setReconnectPeriod(3600000); // 1時間
. . .

```

サーバーのコネクション・ファクトリーの設定で「**Reconnect Enabled**」の値をtrueとした場合、すべての再接続プロセスは、クライアント・アプリケーションで自動的に行われ、クライアント・コードは修正する必要がありません。

5.3.2. コネクション・ファクトリーの再使用

JEUS MQクラスター内のアクティブ・サーバーとスタンバイ・サーバーは、同一名のコネクション・ファクトリーを持っているため、JNDIルックアップで一度取得したコネクション・ファクトリーは、サーバーやネットワークの障害が発生した場合に再度ルックアップする必要なく再使用できます。

5.3.3. デスティネーションの再使用

コネクション・ファクトリーと同様に、JEUS MQクラスター内のアクティブ・サーバーとスタンバイ・サーバーは同一名のデスティネーションを持っています。したがって、JNDIルックアップで一度取得したデスティネーションは、サーバーやネットワークに障害が発生しても再度JNDIをルックアップする必要なく、再使用できます。

障害発生前にデスティネーションに保存していたメッセージは、サーバー障害が発生した場合もすべて自動的に復旧されるため、クライアントは該当デスティネーションを通じてメッセージング作業を継続できます。

5.3.4. 応答待機時間

JEUS MQのクライアントから送られるすべてのリクエストは、「**Request Blocking Time**」という応答が来るまで待機するリクエスト応答時間を持ちます（デフォルト値：200000、単位：ms）。この時間は、JEUS MQサーバーの**Connection Factory**設定画面で「**Request Blocking Time**」で設定できます。

[図 5.6] フェイルオーバーの設定 - リクエスト・ブロッキング・タイム

Request Blocking Time	200 S
[Default: 200] クライアントがサーバに要求を送ってから、応答があるまで待機する時間を指定します。指定した時間が経過するまで応答がない場合は、JMS例外がクライアントに返され、ブロックが解除されます。単位は秒。デフォルト値は200です。	

サーバーの設定だけでなく、コネクション別に異なる設定を行う場合、JEUS MQクライアントAPIの
「jeus.jms.client.facility.connection.JeusConnection」クラスを利用して設定を変更できます。

```

. . .
import jeus.jms.client.facility.connection.JeusConnection;
. . .

```



```
Context ctx = new InitialContext();
ConnectionFactory factory = ctx.lookup("connection-factory");
JeusConnection connection = (JeusConnection)factory.createConnection("jeus",
"jeus");
connection.setRequestBlockingTime(300000); // 5分
. . .
```

リクエスト・ブロッキング・タイムは、セッション・トランザクションあるいはXAの場合にトランザクション・タイムアウトのデフォルト値としても使用されます。

5.3.5. コネクションの復旧

コネクションの復旧がされていないJEUS MQのコネクションは、基本的に物理ソケットを共有します。しかし、domain.xmlのコネクション・ファクトリーの設定で<reconnect-enabled>の値がtrueと設定されている場合、フェイルオーバーのためにそれぞれのコネクションと物理ソケットが1対1の関係になります。

参考

物理ソケットとコネクションが1対1の関係になると、接続を行う度に新しい物理ソケットを作成することになるため、性能が多少低下することがあります。この問題は、毎回コネクションを作成せずに再使用するようにクライアント・アプリケーションを作成することで解決します。

接続が復旧する際、コネクションが再接続されるだけでなく、コネクション状態もすべて復旧します。

- スタート状態

メッセージを受信するためにConnection.start()を呼び出した場合、発生した障害が復旧するとスタート状態が復旧し、メッセージの受信を継続できます。

- ストップ状態

メッセージの受信を停止するためにConnection.stop()を呼び出した場合、障害が復旧するとストップ状態が復旧し、メッセージを受信しなくなります。

コネクション状態だけでなく、コネクション以下のセッションとコネクション・コンシューマもすべて復旧します。

- セッションの復旧

コネクションを通じて作成されたセッションは、障害発生前にSession.close()が呼び出されていない場合、コネクション障害が復旧する際に一緒に復旧します。詳細については、[「5.3.6. セッションの復旧」](#)を参照してください。

- コネクション・コンシューマの復旧

コネクションを通じて作成されたコネクション・コンシューマは、障害発生前に`ConnectionConsumer.close()`が呼び出されていない場合、コネクションが復旧する際に一緒に復旧し、コネクションがスタート状態の場合はメッセージの受信を継続します。ただし、障害前に受信していたメッセージはすべてサーバーに送り返され、新しく受け取ることになるため、以降に受信したメッセージの`Message.getJMSRedelivered()`が`true`になることがあります。

これ以外にも、セッションやコネクション・コンシューマを作成するメソッドは、障害が復旧すると、リクエストを再送信して応答が来るまで待機します。`Connection.close`が呼び出された場合、応答が来ることに関係なく、発生した障害は復旧しません。

5.3.6. セッションの復旧

セッションは、`Session.close()`が呼び出されていない場合、コネクションが復旧する際に一緒に復旧します。また、セッション以下のメッセージ・コンシューマやメッセージ・プロデューサもセッションが復旧する際にすべて復旧します。

セッションでは、複数のオブジェクトを作成するメソッドを持っています。発生した障害を復旧する際、各メソッドは以下のように動作します。

- メッセージ作成メソッド

メッセージを作成するメソッドは、障害有無とは関係なく、常に即時作成されます。

```
createBytesMessage()  
createMapMessage()  
createMessage()  
createObjectMessage()  
createObjectMessage(Serializable object)  
createStreamMessage()  
createTextMessage()  
createTextMessage(String text)
```

- キューブラウザ作成メソッド

キューブラウザを作成するメソッドは、フェイルオーバー後にもリクエストを完了します。リクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合は`JMSException`が発生します。

```
createBrowser(Queue queue)  
createBrowser(Queue queue,String messageSelector)
```

- デスティネーション作成メソッド

デスティネーションを作成するメソッドは、フェイルオーバー後にも作成リクエストを完了します。リクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合は`JMSException`が発生します。

```
createQueue(String queueName)
createTopic(String topicName)
```

- 一時デスティネーション作成メソッド

一時デスティネーションを作成するメソッドは、障害とは関係なく作成リクエストを完了します。

```
createTemporaryQueue()
createTemporaryTopic()
```

- メッセージ・コンシューマ作成メソッド

メッセージ・コンシューマを作成するメソッドは、フェイルオーバー後にも作成リクエストを完了します。リクエスト・ブロッキング・タイムアウトが過ぎた後にも障害が復旧していない場合はJMSEExceptionが発生します。

```
createConsumer(Destination destination)
createConsumer(Destination destination, java.lang.String messageSelector)
createConsumer(Destination destination, java.lang.String messageSelector,boolean
NoLocal)
```

- 永続メッセージ・コンシューマ作成メソッド

永続メッセージ・コンシューマを作成するメソッドは、フェイルオーバー後にも作成リクエストを完了します。リクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合はJMSEExceptionが発生します。

```
createDurableSubscriber(Topic topic, String name)
createDurableSubscriber(Topic topic,String name, String messageSelector,boolean
noLocal)
```

- メッセージ・プロデューサー作成メソッド

メッセージ・プロデューサーを作成するメソッドは、フェイルオーバー後にも作成リクエストを完了します。リクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合はJMSEExceptionが発生します。

```
createProducer(Destination destination)
```

セッションに障害が発生した場合、セッション・トランザクションは以下のような影響を受けます。

- **commit()**

トランザクション・セッションで作成したメッセージ・コンシューマとメッセージ・プロデューサーを利用して、メッセージの受信および送信を実行中に障害が発生した場合、以降に初めて呼び出されるコミットは「javax.jms.TransactionRolledBackException」が発生してロールバックされます。コミットの時点でコミッ

トするメッセージが存在しない場合、この例外は発生しません。障害によってコミットが失敗した後に呼び出されるコミットはすべて正常に動作します。

コミット中に障害が発生し、リクエスト・ブロッキング・タイムが過ぎてからも障害が復旧していない場合、JMSEExceptionが発生します。この場合はコミットがされたか否かについては分からないので、管理ツールを利用して確認します。

- **rollback()**

ロールバックは、フェイルオーバー後にもロールバックのリクエストを完了します。ロールバック中に障害が発生したにもかかわらずリクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合は、JMSEExceptionが発生します。ロールバックの場合、JMSEExceptionが発生してもロールバックされることを保証できます。

Session.recover()メソッドの場合、フェイルオーバー後にもrecoverリクエストを完了します。recoverの呼び出し後に障害が発生したにもかかわらず、リクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合はJMSEExceptionが発生します。

セッションのACKNOWLEDGEモードをSession.CLIENT_ACKNOWLEDGEと設定した場合、Message.acknowledge()を通じてセッション内に存在するacknowledgeができなかったメッセージに対してacknowledgeを呼び出すことができます。acknowledge中に障害が発生した場合、例外リスナーを通じて、jeus.jms.common.message.MessageAcknowledgeExceptionが通知されます。この例外は、メッセージのacknowledge中に障害が発生し、メッセージが再送信(Redelivered)されることがあることを通知します。

参考

MessageAcknowledgeException.getErrorCode()を呼び出した場合、失敗したメッセージのMessageIDを取得できます。

5.3.7. メッセージ送信の復旧

本節では、メッセージ・プロデューサーを使用してメッセージを送信する際に障害が発生した場合について説明します。

メッセージ・プロデューサーのsendメソッドは、メッセージがサーバーに送信されて応答があるまでブロッキングされます。途中で障害が発生した場合、以下のような状況が発生します。

- **sendを呼び出したが、メッセージがまだ送信されていない場合**

この場合、フェイルオーバー後にメッセージはサーバーに送信され、正常処理されます。リクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合、JMSEExceptionが発生します。

- **sendを呼び出し、メッセージがサーバーで処理された後にネットワーク障害が発生した場合**

この場合、障害が復旧して再度サーバーに接続すると、応答メッセージを受信して正常に処理されます。リクエスト・ブロッキング・タイムが過ぎた後にも障害が復旧していない場合、`JMSEException`が発生します。

- **sendを呼び出し、メッセージがサーバーで処理された後に、サーバー障害が発生し、その障害から復旧した場合**

この場合、障害が復旧して再度サーバーに接続しても、メッセージの送信可否が分かりません。そのため、リクエスト・ブロッキング・タイムの分だけ待機し、例外リスナーを通じて「`jeus.jms.common.message.MessageSendException`」を通知します。

- **sendを呼び出し、メッセージがサーバーで処理される前に、ネットワークやサーバー障害が発生した場合**

この場合にも上記の場合と同様に、復旧されたサーバーに再度接続してもメッセージの送信可否が分かりません。そのため、リクエスト・ブロッキング・タイムの分だけ待機し、例外リスナーを通じて「`jeus.jms.common.message.MessageSendException`」を通知します。

参考

`MessageSendException.getErrorCode()`を呼び出した場合、失敗したメッセージのMessageIDを取得できます。

5.3.8. メッセージ受信の復旧

メッセージの受信は同期と非同期方法に分けられ、障害が復旧する場合の動作方式が少しずつ異なります。まず、同期式でメッセージを受信する場合のフェイルオーバーについて説明します。

同期式メッセージ受信の復旧

メッセージ・コンシューマには、`MessageConsumer.receive()`、`MessageConsumer.receive(long timeout)`、`MessageConsumer.receiveNoWait()`という3つの同期式メッセージ受信メソッドがあります。

各メソッドを呼び出す場合、障害が発生すると以下のように動作します。

- **receive()**

`receive()`メソッドの場合、本来のメッセージを受信するまでブロッキングされますが、障害が発生した際にいつ復旧されるか分からないため、待機時間を`RequestBlockingTime`に変更します。時間内に障害が復旧した場合、メッセージ受信をリクエストするメッセージを再送信してメッセージを受信することになり、そうでない場合は`JMSEException`が発生します。

`Session.AUTO_ACKNOWLEDGE`と設定した場合、メッセージを受信し、クライアントにメッセージを送信する前に確認応答をサーバーに送信します。この際に障害が発生した場合、確認応答に失敗したメッセージに対して例外リスナーを通じて`jeus.jms.common.message.MessageAcknowledgeException`を通知

します。この例外は、メッセージの確認応答中に障害が発生し、メッセージが再送信(Redelivered)されることがあることを通知します。

- **receive(long timeout)**

receive(long timeout)メソッドは、本来のメッセージを受信するまでブロッキングされますが、障害が発生した際にいつ復旧するか分からないため、タイムアウトがRequestBlockingTimeより大きい場合、制限時間をRequestBlockingTimeに変えます。制限時間内に障害が復旧した場合、メッセージ受信をリクエストするメッセージを再送信してメッセージを受信することになり、そうでない場合はJMSEExceptionが発生します。

Session.AUTO_ACKNOWLEDGEと設定した場合、メッセージを受信し、クライアントにメッセージを送信する前に確認応答をサーバーに送信します。この際に障害が発生した場合、例外リスナーを通じて、確認応答に失敗したメッセージに対してjeus.jms.common.message.MessageAcknowledgeExceptionを通知します。この例外は、メッセージの確認応答中に障害が発生し、メッセージが再送信(Redelivered)されることがあることを通知します。

- **receiveNoWait()**

receiveNoWait()メソッドは、本来のメッセージを受信しなくてもブロッキングされません。したがって、障害が発生してもすぐに返されます。

非同期式メッセージ受信の復旧

非同期式メッセージ受信の復旧方法で受信したメッセージはMessageListener.onMessageやMessageListener.onMessageで処理され、確認応答中や事前取得により、クライアント・キューに蓄積しているメッセージに分けることができます。

各メッセージに関してJEUS MQは以下のように障害を復旧します。

- onMessage中に障害が発生した場合、障害が復旧後に確認応答が送信されるため、正常に処理されま
す。
- onMessageが完了後、確認応答が送信中に障害が発生した場合、確認応答に失敗したメッセージに対し
て例外リスナーを通じてjeus.jms.common.message.MessageAcknowledgeExceptionを通知します。こ
の例外は、メッセージの確認応答中に障害が発生し、メッセージが再送信(Redelivered)されることがある
ことを通知します。
- 事前取得でクライアント・キューに蓄積しているメッセージをフェイルオーバー後にすべてサーバーに送り
返され、後に再度クライアントに送信されます。このメッセージのMessage.getJMSRedelivered()の値は
trueになることがあります。

参考

`MessageAcknowledgeException.getErrorCode()`を呼び出すと、失敗したメッセージのMessageIDを取得できます。

5.3.9. メッセージの紛失防止とトランザクション

JEUS MQのフェイルオーバーがクライアント・アプリケーションで自動的に処理されます。しかし、メッセージの送受信中にメッセージを紛失する恐れがあるため、例外リスナーを通じてこのようなメッセージに対して別途処理が必要です。

エンタープライズ・メッセージング・アプリケーションでのメッセージ紛失は致命的な問題を引き起こすことがあります。メッセージの紛失を防止しつつ完璧に障害から復旧できる唯一の方法は、トランザクションを使用することです。

したがって、以下の方法でアプリケーションを作成することを推奨します。

- J2EE環境では必ずトランザクション内でメッセージを送受信します。
- サンプルの場合、`UserTransaction`をJNDIからルックアップし、`UserTransaction`内でメッセージを送受信します。
- EJBの場合、EJBメソッドのトランザクション属性(`TransactionAttribute`)を「Required」か「RequireNew」に設定し、常にトランザクション内でメッセージを送受信します。

一般的なJavaクライアントでは、セッションを作成する際に`Connection.createSession(true, Session.SESSION_TRANSACTED)`を呼び出して作成します。このように作成したセッションは、`commit()`や`rollback()`の呼び出しを通じてトランザクション内でメッセージを送受信できます。

第6章 JEUS MQの特殊機能

本章では、JEUS MQの特殊機能である、メッセージ・ブリッジ、メッセージ・ソート、グローバル・オーダー、メッセージ・グループ、メッセージ管理機能について説明します。

6.1. JEUS MQのメッセージ・ブリッジ

メッセージ・ブリッジ(Message Bridge)は2つの異なるMQを接続する機能です。異なるMQとは以下の場合を含みます。

- 同じMQサービスの異なるバージョン(互いに互換していない異なるバージョンのJEUS MQ間の接続)
- それぞれ異なるMQサービス(WebLogicなど、他のベンダーのMQとJEUS MQ間の接続)

[注意事項]

接続できるMQサービスには制限がないため、JEUS MQサービスの同一バージョン間のメッセージ・ブリッジを設定することができますが、推奨しません。

JEUS MQサービスのバージョンが同じ場合、クライアントを遠隔デスティネーションに直接つけて使用の方がより安全で効率的です。同じバージョンのJEUS MQサービス間のブリッジを設定する必要がある場合は、2つのJUES MQ間の一般的なメッセージ・サービスと区別するために、ブリッジが設定されたサーバーの実行スクリプトに以下のオプションを追加する必要があります。

```
-Djeus.jms.client.use-single-server-entry=false
```

6.1.1. サーバーの設定

1つのメッセージ・ブリッジを使用するには、以下の2つの設定が必要です。

- メッセージ・ブリッジの両端で各MQサービスの接続設定を示すブリッジ・コネクションを2つ設定します。
- 2つのブリッジ・コネクションを接続する実際のブリッジを示すブリッジ・エントリーを設定します。

以下は、WebAdminのMQからWebLogic 10.3のMQへのブリッジを設定したサンプル画面です。

メッセージ・ブリッジはJEUSのMQとは別に動作するため、設定をするにはWebAdminの左側のメニューの **[Resources] > [Message Bridge]** を選択します。

Message Connections領域にMQとWebLogic 10.3のMQに関する設定を行い、**Message Entries**にこの2つの間の接続を設定します。メッセージ・コネクションやメッセージ・エントリーの詳細項目を設定するには、項目の名前をクリックします。以下のように詳細項目の設定画面に移動し、各項目の情報を設定し、**[確認]**ボタンをクリックすると、設定した情報が保存されます。項目を再設定するには**[再設定]**ボタンをクリックします。

[図 6.1] メッセージ・ブリッジの設定

jeus_domain

- Domain
- Session
- Clusters
- Servers
- Applications
- Security
- Resources
 - DataSource
 - Mail Source
 - URL Source
 - Message Bridge**
 - Custom Resource
 - External Source
 - External Resource
 - Concurrency Utilities Resource
- Monitoring
- Console
- System Status

Message Bridge

様々なJMSベンダのデスティネーション間にメッセージブリッジを設定するときに使用されます。

Message Bridge

動的設定 * 必須項目

Basedir:

ローカルトランザクション(非XA)モードでリポジトリとして使用されるジャーナル(実行履歴)ストアのディレクトリを設定します。

Message Connections

Name	Classpath	
jeus8	clientcontainer.jar	Delete
weblogic	wlirmsclient.jar	Delete

Message Entries

Name	Message Selector	Timeout	
bridge1			Delete

各詳細項目の設定画面は以下のとおりです。

- JEUSに接続するためのブリッジ・コネクションの設定

Connection

HISTORY

ブリッジコネクションはJMSベンダへの物理的な接続情報を示します。

動的設定 必須項目

確認 再設定

Name *	jeus8 ブリッジコネクションの一意の名前を指定します。
Classpath *	clientcontainer.jar ブリッジコネクションのベンダが提供するクラスパスを設定します。
Jndi Provider Url *	localhost:9736 ブリッジコネクションが利用するJNDIサービスのプロバイダURLを設定します。
Jndi Initial Context Factory *	jeus.jndi.JNSContextFactory ブリッジコネクションが利用するJNDIサービスのイニシャルコンテキストファクトリ名を指定します。
Connection Factory *	ConnectionFactory ブリッジコネクションが利用するJMSコネクションファクトリ名を指定します。
Xa Support	<input type="checkbox"/> [Default: true] ブリッジコネクションが分散トランザクション(XA)をサポートするかどうかを設定します。
Username	jeus ブリッジコネクションが利用するJMSコネクションファクトリに必要なユーザ名を指定します。
Password 入力 ブリッジコネクションが利用するJMSコネクションファクトリに必要なパスワードを指定します。

- WebLogic 10.3に接続するためのブリッジ・コネクション設定

Connection

HISTORY

ブリッジコネクションはJMSベンダへの物理的な接続情報を示します。

動的設定 必須項目

確認 再設定

Name *	weblogic ブリッジコネクションの一意の名前を指定します。
Classpath *	wljsclient.jar ブリッジコネクションのベンダが提供するクラスパスを設定します。
Jndi Provider Url *	t3:localhost:7001 ブリッジコネクションが利用するJNDIサービスのプロバイダURLを設定します。
Jndi Initial Context Factory *	weblogic.jndi.WLInitialContextFactory ブリッジコネクションが利用するJNDIサービスのイニシャルコンテキストファクトリ名を指定します。
Connection Factory *	ConnectionFactory ブリッジコネクションが利用するJMSコネクションファクトリ名を指定します。
Xa Support	<input type="checkbox"/> [Default: true] ブリッジコネクションが分散トランザクション(XA)をサポートするかどうかを設定します。
Username	weblogic ブリッジコネクションが利用するJMSコネクションファクトリに必要なユーザ名を指定します。
Password 入力 ブリッジコネクションが利用するJMSコネクションファクトリに必要なパスワードを指定します。

- 2つのブリッジ・コレクションを接続するブリッジ・エントリーの設定

HISTORY

ブリッジエントリーは、メッセージを取得して渡すブリッジデスティネーションを示します。ソースデスティネーションとターゲットデスティネーションで構成されます。

ヘルプ

動的設定
必須項目

確認 再設定

Name *	bridge1 ブリッジエントリーを区別するための名前を指定します。
Message Selector	<input type="text"/> ブリッジエントリーに設定するメッセージセレクタを指定します。
Timeout	<input type="text" value="10000"/> ms [Default: 10000] メッセージを取得するときのタイムアウトを設定します。

Source

ブリッジエントリーがメッセージを取得するブリッジデスティネーションを設定します。

Connection Name *	weblogic ブリッジデスティネーションを設定するブリッジコネクションの名前を指定します。
Destination *	Queue1 ブリッジデスティネーションのJMSデスティネーション名を指定します。
Type *	queue ブリッジデスティネーションのJMSデスティネーションのタイプを指定します。キューまたはトピックを設定できます。

Target

ブリッジエントリーがメッセージを渡すブリッジデスティネーションを設定します。

Connection Name *	jeus8 ブリッジデスティネーションを設定するブリッジコネクションの名前を指定します。
Destination *	ExamplesQueue ブリッジデスティネーションのJMSデスティネーション名を指定します。
Type *	queue ブリッジデスティネーションのJMSデスティネーションのタイプを指定します。キューまたはトピックを設定できます。

6.2. JEUS MQのメッセージ・ソート

キューにメッセージ・プロデューサーのみが存在してコンシューマが存在しない場合や、コンシューマは存在するが受信速度が相対的に遅い場合、該当キューにメッセージが蓄積することがあります。このような場合、蓄積したメッセージをユーザーが指定したキー値の順序どおりソートする機能をメッセージ・ソートといいいます。永続サブスクライバーの場合もメッセージが蓄積することがあり、キューと同様にメッセージ・ソート機能を利用することができます。指定したキー値にはJMSの基本的なプロパティ値、またはユーザーが指定したプロパティ(User Property)値が含まれます。

メッセージ・ソートの設定は、サーバーとクライアントで方法が異なります。メッセージ・ソートに関連するサーバーの設定はWebAdminを使用し、クライアント設定はソート対象メッセージにプロパティ値で設定します。

6.2.1. サーバーの設定

サーバーのメッセージ・ソート設定は、WebAdminの[Servers] > [サーバー名] > [Engine] > [Jms Engine] > [Message Sort]メニューを選択します。メッセージ・ソート機能を使用するには、以下のようにサーバーに基準を設定します。

[図 6.2] メッセージ・ソート設定

Message Sort

HISTORY

デスティネーション内でメッセージソートを行うための設定を行います。

ヘルプ

Basic Resource Engine

Web Engine | Jms Engine | Ejb Engine

Basic Service Config Connection Factory Persistence Store Message Sort

動的設定 * 必須項目 確認 再設定

Name *	SortByPriority メッセージソート 設定の名前です。キューと永続的サブスクリプションの設定でこの名前を指定すると、当該設定が適用されます。
Key *	JMSPriority メッセージソート のためのキー値を指定します。JMSメッセージヘッダの「JMS」で始まる定義済みプロパティや任意のユーザプロパティを使用できます。
Type	Integer メッセージソート のためのキー値のタイプを指定します。定義済みプロパティに対しては指定する必要がありません。Boolean、Byte、Float、Integer、Double、Stringのみ設定できます。デフォルト値はStringです。
Direction *	Ascending メッセージのソート方向を指定します。Ascending(昇順)またはDescending(降順)を使用できます。

その後、上記基準で作成されたメッセージ・ソート機能をデスティネーションや永続サブスクライバーで使用するよう以下のように設定できます。詳細については、「[3.2.1. デスティネーションの設定](#)」と「[3.2.2. 永続サブスクライバーの設定](#)」。

指定したキー値に該当しないメッセージはソートの対象になりません。この場合、本来の順序が最大限保証されます。つまり、キー値に該当するメッセージと該当しないメッセージが混同してキューに蓄積した場合、キー値に該当するメッセージは設定に基づいてソートされ、キー値に該当しないメッセージはメッセージ間の本来の順序が保証されます。

[図 6.3] デスティネーションにメッセージ・ソートを適用

Message Sort

デスティネーションに適用するメッセージソートを設定します。

6.2.2. クライアントの設定

クライアントでは、対象となるメッセージに、ソートの基準となるキー値をプロパティとして設定します。

サーバーのメッセージ・ソート設定でメッセージ・ソートのキーを「TEST_KEY」と設定したと仮定した場合、以下の例のように設定するとメッセージがソートされます。

```
Message msg = session.createTextMessage("Test");
msg.setIntProperty("TEST_KEY", 1);
```

6.3. JEUS MQのグローバル・オーダー

グローバル・オーダーは、デスティネーションに蓄積されたメッセージが必ず1つずつクライアントに渡されるように提案され、厳格な処理順序を保証する機能です。一般的には、メッセージが複数のクライアントで並列処理される可能性があり、その場合にそれぞれの処理順序を制限する方法がありません。つまり、通常並列で複数のメッセージ・コンシューマをつけると、メッセージの送信順とは関係なくメッセージが処理されます。このような動作は仕様上問題はありませんが、処理順序を保証する必要がある場合にグローバル・オーダー機能が使用されます。

6.3.1. クライアントの設定

グローバル・オーダー機能は、サーバーには特別な設定を行わず、クライアントで専用のAPIを呼び出して使用できます。

参考

グローバル・オーダーは、メッセージ・コンシューマが1つの場合には従来の動作と殆ど変わりません。したがって、グローバル・オーダー機能を使用する際はデスティネーションに複数のメッセージ・コンシューマが接続されるように設定します。

メッセージ・コンシューマでプロデューサーを作成した後、JeusMessageProducerにキャッシングして専用のAPIを呼び出します。

```
JeusMessageProducer producer = (JeusMessageProducer) session.createProducer(queue);
// グローバル・オーダーの指定および名前の割り当て
producer.startGlobalOrder("GLOBAL-ORDER-NAME");
// 名前を指定していない場合
//producer.startGlobalOrder();
```

GLOBAL-ORDER-NAMEは各グローバル・オーダーを区別する名前であり、指定しない場合は任意の名前が割り当てられます。以降のメッセージ送信は従来と同じです。グローバル・オーダーを区別する名前は、複数のクライアントで共有することができます。**同機能をクラスタリングと一緒に使用すると、クラスター全体の処理順序を最大限に保証します。**

6.4. JEUS MQのメッセージ・グループ

メッセージ・グループは、特定の目的を持つグループを設定し、デスティネーションに該当グループに属するメッセージがすべて集まった場合に、そのグループのメッセージを1つのメッセージ・コンシューマにすべて渡す機能です。たとえば、メッセージが合計10個のグループを設定すると、該当グループに10個のメッセージがすべて集まるまでは、そのメッセージは集合状態でサービスがされず、10番目のメッセージが到着すれば、集合メッセージが1つのメッセージ・コンシューマに渡されます。トランザクションの概念と似ており、類似の用途で使用されます。メッセージ・グループはクラスタリングと平行して使用でき、このような場合にも1つのメッセージ・コンシューマに集合メッセージを渡すことができるように処理します。

6.4.1. サーバーの設定

メッセージ・グループ機能を使用するために、**デスティネーション設定**画面でサーバーを追加します。デスティネーション設定画面については、「[3.2.1. デスティネーションの設定](#)」を参照してください。

[図 6.4] メッセージ・グループ設定

Message Group デスティネーションに適用するメッセージグループを設定します。	
Message Handling	<div><div></div><div>[Default: Pass] デスティネーションがメッセージグループを処理する方式を指定します。PassまたはGatherを使用できます。Passを指定すると、一般メッセージと同様に扱います。Gatherを指定すると、メッセージグループを完成させて1つのメッセージとして送信します。</div></div>
Expiration Time	<div><div></div><div>[Default: -1] デスティネーションが未完成メッセージグループを保持する最大時間を秒単位で指定します。デフォルト値は-1で、メッセージグループは完成されるまで消去されません。</div></div>

参考

「Expiration Time」を設定せずに使用した場合、未完成のメッセージ・グループは永遠にサーバーから消えずに保存されます。メモリーの浪費となることがあるため、注意が必要です。

6.4.2. クライアントの設定

クライアントでは、メッセージ・プロデューサーとメッセージ・コンシューマの情報をそれぞれ設定します。

● メッセージ・プロデューサーの設定

メッセージのユーザー・プロパティを設定することにより、機能を有効にします。

```
Message msg = session.createMessage();  
// メッセージ・グループ名を設定します。グループを代表する名前です。  
msg.setStringProperty("JMS_JEUS_MSG_GROUP_NAME", "MESSAGE-GROUP-NAME");  
  
// 該当グループ内での順序を示します。後にクライアントに渡す際の順序でもあります。
```

```

msg.setIntProperty("JMS_JEUS_MSG_GROUP_NUMBERING", 1);
producer.send(msg);

msg = session.createMessage();
msg.setStringProperty("JMS_JEUS_MSG_GROUP_NAME", "MESSAGE-GROUP-NAME");
msg.setIntProperty("JMS_JEUS_MSG_GROUP_NUMBERING", 2);
producer.send(msg);

. . .

msg = session.createMessage();
msg.setStringProperty("JMS_JEUS_MSG_GROUP_NAME", "MESSAGE-GROUP-NAME");
msg.setIntProperty("JMS_JEUS_MSG_GROUP_NUMBERING", 10);
// 該当メッセージが該当グループの最後のメッセージであることを示します。
msg.setBooleanProperty("JMS_JEUS_MSG_GROUP_END", true);
producer.send(msg);

```

- メッセージ・コンシューマの設定

完成したメッセージ・グループを受信すると、すべてのメッセージが1つにまとめられて、一覧で構成されたオブジェクト・メッセージとして受信します。

```

// メッセージ・グループとして受信するメッセージは1つのオブジェクト・メッセージで、このオブジェ
クト・メッセージはメッセージの一覧です。
ObjectMessage result = (ObjectMessage) receiver.receive(TIME_OUT);
List list = (List) result.getObject();

int cnt = 1;
// 以下のように一覧から1つずつ取り出して使用できます。
for(Object obj : list) {
    // メッセージ・プロデューサーで設定した順番どおりメッセージを処理します。
    TextMessage msg = (TextMessage) obj;
    . . .
}

```

6.5. JEUS MQのメッセージ管理機能

メッセージ管理機能は、JEUS MQを利用してメッセージング・サービスを使用中にデスティネーション内に入ってきたメッセージを確認、移動、削除など、メッセージを管理するために提供される機能です。

メッセージ管理機能は、以下のように3つに分けられます。

- メッセージのモニタリング

デスティネーション内のメッセージをモニタリングします。

- メッセージの制御

メッセージの移動、削除、エクスポート、インポートを通じてメッセージを制御します。

- デスティネーションの制御

円滑なメッセージのモニタリングと制御作業のためにデスティネーションを制御します。

この際、メッセージのモニタリングとメッセージの制御機能は現在サービス中のキューや永続的サブスクライバーに残っているメッセージを対象とし、デスティネーションの制御機能はキューとトピック両方を対象にします。

6.5.1. メッセージのモニタリング

キューまたは永続サブスクライバー内に受信されたメッセージをモニタリングおよび各メッセージ情報を照会します。

メッセージ一覧の表示

キューまたは永続サブスクライバー内に残っているすべてのメッセージの一覧を表示する機能です。各メッセージのID、タイプ、作成時刻などの情報を表示し、一覧からメッセージを選択して詳細情報の表示、削除、移動、エクスポートなどの機能を実行できます。

参考

JMSは、メッセージが非常に速く作成されて消費されるため、デスティネーションの消費一時中止機能を使用せずにメッセージ一覧を表示した場合、現在の一覧には表示されたメッセージが既に消費されて存在しない場合があります。

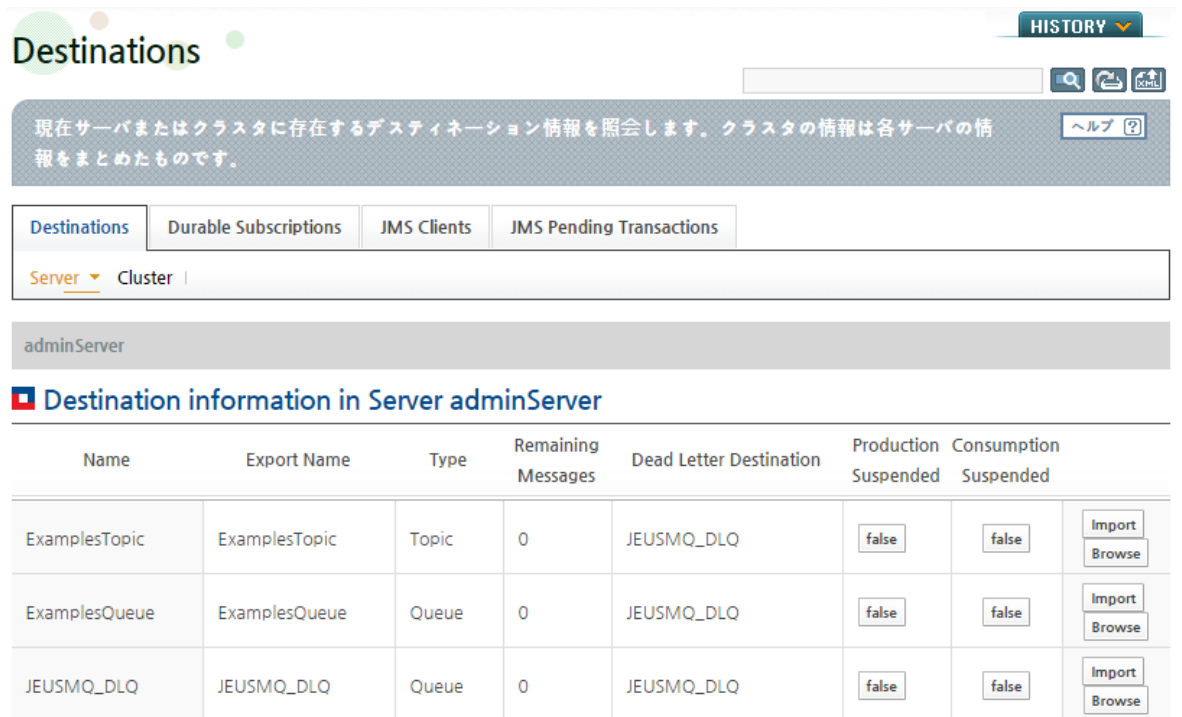
メッセージ一覧を表示する手順は以下のとおりです。

1. 表示するメッセージの位置によって、WebAdminでのメニューが以下のように異なります。

- キュー内のメッセージ

[Monitoring] > **[JMS]**メニューを選択し、**[Destinations]**タブの**[Server]**を選択します。メッセージを表示するサーバーを選択すると、該当サーバー内のデスティネーション一覧が表示されます。

[図 6.5] デスティネーションでのメッセージ一覧の表示



Destinations HISTORY

現在サーバまたはクラスタに存在するデスティネーション情報を照会します。クラスタの情報は各サーバの情報をもとめたものです。 ヘルプ ?

Destinations Durable Subscriptions JMS Clients JMS Pending Transactions

Server Cluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	Import Browse
ExamplesQueue	ExamplesQueue	Queue	0	JEUSMQ_DLQ	false	false	Import Browse
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	Import Browse

– 永続サブスクライバー内のメッセージ

[Monitoring] > [JMS]メニューを選択し、**[Durable Subscriptions]**タブの**[Server]**を選択します。メッセージを表示するサーバーを選択すると、該当サーバー内の永続サブスクリプションの一覧が表示されます。

[図 6.6] 永続サブスクリプションでのメッセージ一覧の表示



Durable Subscriptions HISTORY

現在サーバまたはクラスタに存在する永続的サブスクライバの情報を照会します。クラスタの情報は各サーバの情報をもとめたものです。 ヘルプ ?

Destinations Durable Subscriptions JMS Clients JMS Pending Transactions

Server Cluster

adminServer

Durable Subscription Information

Durable Name	Client ID	Message Selector	Remaining Messages	
durable	client_id		0	Browse

2. キューまたは永続サブスクリプションの**[browse]**ボタンをクリックすると、選択したキューまたは永続サブスライバー内のメッセージ一覧を表示するための**Message List**画面が表示されます。

メッセージ一覧の表示機能を利用する際は、メッセージIDやタイプ、作成時間でフィルタリングすることもできます。また、JMSスペックで指定するMessage Selectorに合わせて、特定のメッセージのみ選択して表示することもできます。キューまたは永続的サブスクリプション内のすべてのメッセージを表示するには、各項目を設定していない状態で**[確認]**ボタンをクリックします。

[図 6.7] 表示するメッセージ一覧の設定

✕

Message List

Selector	<div></div> <div>JMSスペックに定義されているメッセージセレクタです。メッセージセレクタを使って特定のメッセージを選択して閲覧することができます。</div>
ID	<div></div> <div>メッセージIDのパターンを利用して特定のメッセージを選択して閲覧できます。こちらで利用されるパターンはJMSスペックに定義されており、「_」は1つの任意の文字を、「%」は任意の長さの文字を表します。</div>
Type	<div></div> <div>メッセージの特定のタイプを選択して閲覧できます。</div>
From	<div></div> <div>ある時点の後に送信されたメッセージを選択して閲覧できます。形式は、YYYY:MM:DD:HH:MM:SSに従います。</div>
To	<div></div> <div>ある時点の前に送信されたメッセージを選択して閲覧できます。形式は、YYYY:MM:DD:HH:MM:SSに従います。</div>

確認

キャンセル

3. 以下は、デスティネーション内のメッセージ表示一覧です。各メッセージの前のチェックボックスにチェックを入れて、メッセージの削除、移動、エクスポートなどが可能です。「Message ID」列のメッセージIDをクリックしてメッセージの詳細内容を確認できます。

[図 6.8] 表示されたメッセージ一覧

Destinations

HISTORY

現在サーバまたはクラスタに存在するデスティネーション情報を照会します。クラスタの情報は各サーバの情報をまとめたものです。

Destinations

Durable Subscriptions

JMS Clients

JMS Pending Transactions

Server Cluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	<div>Import</div> <div>Browse</div>
ExamplesQueue	ExamplesQueue	Queue	10	JEUSMQ_DLQ	false	false	<div>Import</div> <div>Browse</div>
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	<div>Import</div> <div>Browse</div>

Messages in Destination ExamplesQueue

<input type="checkbox"/>	Message ID	Message Type	Created Time
<input type="checkbox"/>	ID:7627620434500001:1:1	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:2	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:3	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:4	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:5	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:6	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:7	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:8	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:9	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:10	Text	Wed Oct 12 22:32:40 KST 2016

1

[1 - 10 / Total : 10]

Delete

Move

Export

メッセージの詳細情報の表示

メッセージの詳細情報の照会は、キューまたは永続サブスクライバーに受信されたメッセージの詳細情報を表示する機能です。JMS仕様で指定するヘッダーのすべての内容や、メッセージの情報と設定されている属性を確認できます。

84 JEUS MQガイド

参考

JMSは、メッセージが非常に速く作成されて消費されるため、デスティネーションの消費一時中止機能を使用せずにメッセージ詳細情報表示機能を使用した場合、現在の一覧には表示されたメッセージが既に消費されて存在しない場合があります。特に、メッセージ情報を修正する際、一時中止状態でない場合、メッセージ情報の修正については保証できません。

メッセージの詳細情報の表示方法は以下のとおりです。

1. メッセージ一覧表示画面で特定メッセージの「**Message ID**」をクリックすると、メッセージの詳細情報を確認できます。

[図 6.9] 表示されたメッセージ一覧で特定メッセージの詳細情報を表示

Destinations

HISTORY

現在サーバまたはクラスタに存在するデスティネーション情報を照会します。クラスタの情報は各サーバの情報をまとめたものです。

Destinations

Durable Subscriptions

JMS Clients

JMS Pending Transactions

Server Cluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLO	false	false	Import Browse
ExamplesQueue	ExamplesQueue	Queue	10	JEUSMQ_DLO	false	false	Import Browse
JEUSMQ_DLO	JEUSMQ_DLO	Queue	0	JEUSMQ_DLO	false	false	Import Browse

Messages in Destination ExamplesQueue

	Message ID	Message Type	Created Time
<input type="checkbox"/>	ID:7627620434500001:1:1	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:2	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:3	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:4	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:5	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:6	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:7	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:8	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:9	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:10	Text	Wed Oct 12 22:32:40 KST 2016

1 [1 - 10 / Total : 10]

Delete Move Export

2. 以下は、表示されたメッセージの詳細情報画面です。

一部のヘッダーとメッセージのタイプが「Text」の場合、メッセージの内容およびメッセージ・プロパティは修正できます。修正を行って**[確認]**ボタンをクリックすると、メッセージ情報が修正されます。**[再設定]**ボタンをクリックすると、修正された内容が元の状態に戻ります。

[図 6.10] メッセージの詳細情報の表示

Message In destination ExamplesQueue

指定のデスティネーションがconsume-suspended状態でない場合、異常動作する可能性があります。

メッセージの修正が必要な場合、以下の変更可能な値を変更してメッセージを修正できます。

確認

再設定

Message ID	ID:7627620434500001:1:1
Message Type	Text
Destination	ExamplesQueue
Time Created	Wed Oct 12 22:32:40 KST 2016
Priority	4
Time Delivered	Wed Oct 12 22:32:40 KST 2016
Delivery Mode	PERSISTENT
Correlation ID	
Expiration Time	-
Time To Live	0
Redelivered	false
Redelivery Limit	3
Reply To	
Message Body	Test Message
Properties	<div>key=[data_type]value</div>

確認

再設定

6.5.2. メッセージの制御

サーバーに受信されたメッセージに対して、以下のような制御機能を提供します。

- メッセージの移動

メッセージをサーバーやクラスター内の他のデスティネーションに移動させます。

- メッセージの削除

メッセージをキューまたは永続的サブスクライバーから削除します。

- メッセージのエクスポート

サーバーまたはクラスター内の特定のメッセージを互いに異なるサーバーまたはクラスターに移すことができる形式でエクスポートします。

- メッセージのインポート

エクスポートされたメッセージをサーバーまたはクラスターの特定のデスティネーションにインポートします。

メッセージの移動

サーバーまたはクラスター内の他のデスティネーションにメッセージを移動させることができます。誤ったデスティネーションにメッセージを送信した場合などに対処できる機能です。

参考

1. JMSは、メッセージが非常に速く作成されて消費されるため、元のメッセージが存在するデスティネーションの消費一時中止機能を使用しない場合、移動させるメッセージが既に該当デスティネーションに存在しない場合があります。

2. メッセージの移動は、メッセージをそのまま他のデスティネーションに移す機能です。対象デスティネーションに移動させるメッセージと同じIDを持つメッセージが既に存在する場合、移動するメッセージが以前のメッセージを上書きします。

対象デスティネーションで同じIDを持つメッセージが存在して上書きする場合、該当メッセージが既にクライアントに渡されているのであれば、サーバーとクライアント間のメッセージが一致しない状況が発生します。そのため、メッセージを移動させる際は、2つのデスティネーションが両方消費一時中止状態の場合にこの機能を使用することを推奨します。

メッセージの移動は以下の手順で行います。

1. メッセージ一覧表示画面で移動するメッセージをチェックし、[move]ボタンをクリックします。

[図 6.11] メッセージの移動

Destinations

HISTORY

現在サーバまたはクラスタに存在するデスティネーション情報を照会します。クラスタの情報は各サーバの情報をまとめたものです。

ヘルプ

DestinationsDurable SubscriptionsJMS ClientsJMS Pending Transactions

ServerCluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	Import Browse
ExamplesQueue	ExamplesQueue	Queue	10	JEUSMQ_DLQ	false	false	Import Browse
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	Import Browse

Messages in Destination ExamplesQueue

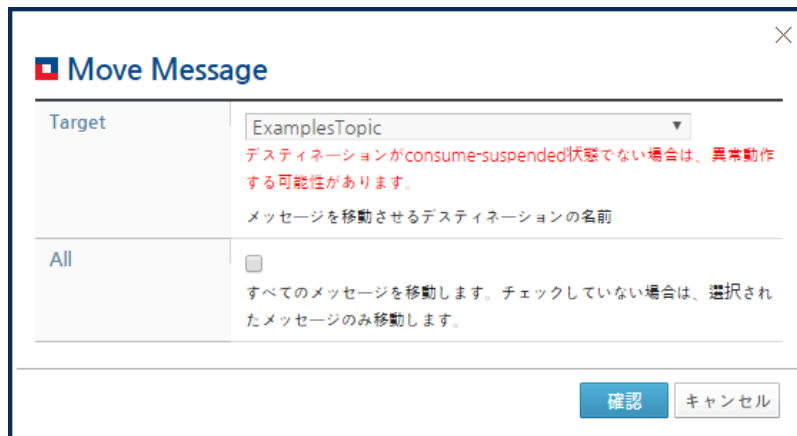
	Message ID	Message Type	Created Time
<input type="checkbox"/>	ID:7627620434500001:1:1	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:2	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:3	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:4	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:5	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:6	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:7	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:8	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:9	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:10	Text	Wed Oct 12 22:32:40 KST 2016

1 [1 - 10 / Total : 10]

DeleteMoveExport

2. **Move Message**画面の「**Target**」で対象となるデスティネーションを選択します。また、「**All**」をチェックすると、現在選択されているキューまたは永続的サブスクライバーのすべてのメッセージを対象デスティネーションに移動させます。各項目を設定し、**[確認]**ボタンをクリックすると、選択したメッセージが設定に従って移動します。

[図 6.12] メッセージの移動設定



The image shows a 'Move Message' dialog box with a title bar and a close button. It contains two main sections: 'Target' and 'All'. The 'Target' section has a dropdown menu showing 'ExamplesTopic' and a warning message in red text: 'デスティネーションがconsume-suspended状態でない場合は、異常動作する可能性があります。' (If the destination is not in a consume-suspended state, abnormal operation is possible). Below this is a label 'メッセージを移動させるデスティネーションの名前' (Name of the destination to move the message to). The 'All' section has a checkbox that is currently unchecked and a message: 'すべてのメッセージを移動します。チェックしていない場合は、選択されたメッセージのみ移動します。' (Move all messages. If not checked, only the selected messages will be moved). At the bottom right, there are two buttons: '確認' (Confirm) and 'キャンセル' (Cancel).

メッセージの削除

メッセージの削除は、サーバー内のメッセージがなくなった場合に使用する機能です。

参考

JMSは、メッセージが非常に速く作成されて消費されるため、デスティネーションの消費一時中止機能を使用しなければ、メッセージ一覧表示画面で確認されたメッセージでも、実際には既に消費されて削除できない場合があります。したがって、正確なメッセージ削除を行うためには、該当デスティネーションが消費一時中止状態でこの機能を使用することを推奨します。

メッセージの削除は以下の手順で行います。

1. メッセージの一覧表示画面で削除するメッセージをチェックし、[delete]ボタンをクリックします。

[図 6.13] メッセージの削除

Destinations

HISTORY

現在サーバまたはクラスタに存在するデスティネーション情報を照会します。クラスタの情報は各サーバの情報をまとめたものです。

ヘルプ

Destinations

Durable Subscriptions

JMS Clients

JMS Pending Transactions

Server

Cluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	<div>Import</div> <div>Browse</div>
ExamplesQueue	ExamplesQueue	Queue	10	JEUSMQ_DLQ	false	false	<div>Import</div> <div>Browse</div>
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	<div>Import</div> <div>Browse</div>

Messages in Destination ExamplesQueue

	Message ID	Message Type	Created Time
<input type="checkbox"/>	ID:7627620434500001:1:1	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:2	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:3	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:4	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:5	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:6	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:7	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:8	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:9	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:10	Text	Wed Oct 12 22:32:40 KST 2016

1

[1 - 10 / Total : 10]

Delete

Move

Export

2. **Delete Message**画面で**[確認]**ボタンをクリックすると、選択されたメッセージをサーバーから削除します。デスティネーション内のすべてのメッセージを削除するには「**All**」にチェックし、**[確認]**ボタンをクリックします。

[図 6.14]



メッセージのエクスポート

キューまたは永続サブスクライバー内のメッセージを他のサーバーやクラスターにエクスポートする機能です。

参考

JMSは、メッセージが非常に速く作成されて消費されるため、一覧では表示されてもメッセージが既に消費された可能性もあります。そのため、メッセージを確実にエクスポートするためには、デスティネーションが消費一時中止状態の際に使用します。

メッセージのエクスポートの手順は以下のとおりです。

1. メッセージ一覧表示画面でエクスポートするメッセージをチェックし、**[export]**ボタンをクリックします。

[図 6.15] メッセージのエクスポート

Destinations

HISTORY

現在サーバまたはクラスタに存在するデスティネーション情報を照会します。クラスタの情報は各サーバの情報をもとめたものです。

ヘルプ

DestinationsDurable SubscriptionsJMS ClientsJMS Pending Transactions

ServerCluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	Import Browse
ExamplesQueue	ExamplesQueue	Queue	10	JEUSMQ_DLQ	false	false	Import Browse
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	Import Browse

Messages in Destination ExamplesQueue

	Message ID	Message Type	Created Time
<input type="checkbox"/>	ID:7627620434500001:1:1	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:2	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:3	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:4	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:5	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:6	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:7	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:8	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:9	Text	Wed Oct 12 22:32:40 KST 2016
<input type="checkbox"/>	ID:7627620434500001:1:10	Text	Wed Oct 12 22:32:40 KST 2016

1 [1 - 10 / Total : 10]

DeleteMoveExport

2. **Message Export**画面で「**All**」をチェックすると、該当デスティネーションや永続的サブスクライバー内のすべてのメッセージをエクスポートできます。**[確認]**ボタンをクリックするとメッセージのエクスポートが実行され、ブラウザを通じて「message.xml」というファイルをダウンロードできます。

[図 6.16] メッセージのエクスポートの設定



メッセージのインポート

メッセージのエクスポート機能を実行すると作成されるXMLファイルを利用して、サーバー内の特定のデスティネーションにメッセージをインポートできます。メッセージの移動とは異なり、全く新しいメッセージとして取り扱われ、メッセージがデスティネーションに入ってきた瞬間に新しいメッセージIDが発給されます。

参考

メッセージのインポートで「**Overwrite**」をチェックして既存のメッセージを置き換える際、メッセージが既にクライアントに渡されている可能性があります。この場合、インポートは行われますが、上書きは正常に実行されません。この機能は該当デスティネーションが消費一時中止状態の際に使用することを推奨します。

メッセージのインポートの手順は以下のとおりです。

1. WebAdminの**[Monitoring]** > **[JMS]**メニューを選択し、**[Destinations]**タブの**[Server]**を選択します。対象となるデスティネーションの**[import]**ボタンをクリックします。

【図 6.17】メッセージのインポート

現在サーバまたはクラスタに存在するデスティネーション情報を照会します。クラスタの情報は各サーバの情報をもとめたものです。

Destinations | Durable Subscriptions | JMS Clients | JMS Pending Transactions

Server | Cluster |

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false
ExamplesQueue	ExamplesQueue	Queue	0	JEUSMQ_DLQ	false	false
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false

2. **Message Import**画面で詳細項目を設定して**【確認】**ボタンをクリックすると、メッセージをインポートします。この際、「**Overwrite**」をチェックすると、以前にあったメッセージのIDをそのまま使用し、対象となるデスティネーション内に同じIDを持つメッセージが既に存在する場合はそのメッセージを上書きします。

【図 6.18】メッセージのインポート設定

Message Import

Path: ファイル選択

メッセージが保存されている。インポートするXMLファイルのパスを指定します。指定したパスのXMLファイルがサーバにアップロードされます。

Overwrite: ☐

当該デスティネーションがConsume suspended状態でない場合は、異常動作する可能性があります。

メッセージをインポートするとき、設定されていた既存のメッセージIDをそのまま使用します。同じメッセージIDがすでにデスティネーションに存在している場合は、新規メッセージに上書きされます。

確認 キャンセル

6.5.3. デスティネーションの制御

サーバー内のデスティネーション別に生産と消費サービスを一時中止あるいは再開できます。

デスティネーションの生産一時中止および再開

該当デスティネーションでのメッセージの生産を一時中止および再開する機能です。デスティネーションが生産一時中止状態になると、該当デスティネーションでメッセージを生産することは不可能になります。

WebAdminの[Monitoring] > [JMS]メニューを選択し、[Destinations]タブの[Server]を選択します。該当サーバーのデスティネーション一覧が表示されます。

[図 6.19] メッセージの生産制御

Destinations

HISTORY

現在サーバーまたはクラスターに存在するデスティネーション情報を照会します。クラスターの情報は各サーバーの情報をまとめたものです。

Destinations

Durable Subscriptions

JMS Clients

JMS Pending Transactions

Server Cluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	Import Browse
ExamplesQueue	ExamplesQueue	Queue	0	JEUSMQ_DLQ	false	false	Import Browse
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	Import Browse

各デスティネーション別に「**Production Suspended**」値によってメッセージ生産に関するデスティネーションの現在の状態を確認でき、各ボタンをクリックしてデスティネーションの状態を変更できます。

設定値	説明
true	デスティネーションの生産が一時中止の状態です
false	デスティネーションが正常にサービスされている状態です

参考

メッセージの生産が一時中止した状態（「**Production Suspended**」の値が「true」の状態）のデスティネーションでメッセージを送信する場合、一定時間待機してからExceptionListenerを通じてjeus.jms.common.destination.InvalidDestinationStateExceptionを通知します。

デスティネーションの消費一時中止および再開

デスティネーションでのメッセージの消費を一時中止および再開する機能です。デスティネーションの消費が一時中止状態になると、該当デスティネーションでメッセージを消費することは不可能になります。

WebAdminの[Monitoring] > [JMS]メニューを選択し、[Destinations]タブの[Server]を選択します。該当サーバーのデスティネーション一覧が表示されます。

[図 6.20] メッセージの消費制御

Destinations

HISTORY

現在サーバーまたはクラスターに存在するデスティネーション情報を照会します。クラスターの情報は各サーバーの情報をまとめたものです。

Destinations

Durable Subscriptions

JMS Clients

JMS Pending Transactions

Server Cluster

adminServer

Destination information in Server adminServer

Name	Export Name	Type	Remaining Messages	Dead Letter Destination	Production Suspended	Consumption Suspended	
ExamplesTopic	ExamplesTopic	Topic	0	JEUSMQ_DLQ	false	false	Import Browse
ExamplesQueue	ExamplesQueue	Queue	0	JEUSMQ_DLQ	false	false	Import Browse
JEUSMQ_DLQ	JEUSMQ_DLQ	Queue	0	JEUSMQ_DLQ	false	false	Import Browse

各デスティネーション別に「Consumption Suspended」値によってメッセージ消費に関するデスティネーションの現在の状態を変更できます。

設定値	説明
true	デスティネーションの消費が一時中止の状態です
false	デスティネーションが正常にサービスされている状態です

参考

メッセージの消費が一時中止した状態（「Consumption Suspended」の値が「true」の状態）のデスティネーションでメッセージを消費する場合、該当デスティネーションにメッセージが存在しない場合と同じように動作します。

6.6. 高信頼性メッセージの送信

JEUS MQを利用してメッセージを送信するとき、永続ストアを使用すると、サーバーの障害が発生したときにも信頼性の高いメッセージの送信を保証できます。しかし、これはJEUS MQサーバーとメッセージ・コンシューマ間のメッセージの送信のみを保証しており、メッセージ・プロデューサーがサーバーに送信するとき、サーバーまたはクライアントの障害が発生したときのメッセージの送信は保証していません。

これを補完するためにJEUS MQでは永続ストアと似たJEUSのLocal Persistent Queue機能を利用してメッセージ・プロデューサーとJEUS MQサーバー間の信頼性を高める機能を提供しています。Local Persistent Queue(以下、LPQ)はデータをローカルのリポジトリに保存した後、保存されたデータは必ず与えられた作業どおり処理されることを保証するサービスです。JEUS MQではLPQを利用して送信しようとするメッセージを送信する前に保存しておき、メッセージがサーバーに送信されるまで送信を試みてメッセージ・プロデューサーからサーバーへの送信の信頼性を高めています。

LPQを利用した高信頼性メッセージの送信は、以下のような特徴を持ちます。

- サーバーの状態に関係なく、メッセージの送信に成功するまでメッセージを再送信
- クライアントの状態の異常のため送信に失敗したメッセージの復旧
- 非同期メッセージ送信

参考

JMSクライアントがメッセージを送信するとき、該当するクライアントがJava EEアプリケーションで、メッセージを受信するJMSサーバーが同じく該当するクライアントがデプロイされたサーバーである場合、性能向上のためネットワークを介さずにメッセージを送信するため、LPQに関する設定は無視されます。

6.6.1. LPQの有効化

LPQを利用して高信頼性メッセージの送信機能を使用するには、まず、LPQを有効にする必要があります。LPQを有効にするプロセスは、メッセージを保存するリポジトリの作成、メッセージを処理するためのキューおよびそれを管理するための管理オブジェクトの作成で構成されています。

参考

LPQを使用するスタンドアロン・クライアントは、LPQライブラリーのjeus-lpq-spi.jar、jeus-lpq.jar、jms-extensions.jarとストア・ライブラリーのjeusstore.jarが必要となります。これらはJEUS_HOME/lib/systemフォルダーに存在しています。

以下は、LPQを有効にする3つの方法です。

- JVMオプションを使用した有効化

JEUS MQクライアントを実行する際、以下のようなJVMオプションを適用してLPQを有効にすることができます。同オプションを使用すると、クライアント内でJMSがコネクションを生成するときにLPQが有効になります。

```
-Djeus.jms.client.send-by-lpq-only=true
```

- LPQ設定ファイルを使用した有効化

JEUS MQクライアントを実行する際、LPQのための設定ファイルが指定された場所に存在していれば、該当するファイルを読み込んでその設定どおりLPQを有効にします。LPQ設定ファイルの場所と設定方法についての詳細内容は、「[6.6.4. LPQの設定](#)」を参照してください。

- JEUS専用のAPIを使用した有効化

クライアント・ソースでJEUS専用のセッション・オブジェクトであるJeusSessionから以下のAPIを使用してLPQを有効にできます。

```
public void startLPQ();
```

参考

1. JVMオプションを使用した場合やJEUS専用のAPIを使用した場合にも、設定ファイルまたはシステム・プロパティを利用してLPQの動作を設定することができます。設定されていない項目にはデフォルト値が適用されます。LPQ設定についての詳細説明は、「[6.6.4. LPQの設定](#)」を参照してください。
2. JEUS専用のAPIを使用した場合以外には、LPQが有効になる時点が明示されません。そのときは、該当するJVMで最初のコネクションが生成されるときにLPQが有効になります。また、LPQが動作を停止する時点は、最後のコネクションがクローズされるときです。ただし、LPQ内に受信されたメッセージが存在する場合には、そのメッセージが処理されるまで待機することになります。

6.6.2. LPQ使用の設定

LPQが有効になると、LPQを介してメッセージを送信するようにし、メッセージ送信の信頼性を高めることができます。以下は、メッセージがLPQを介して送信されるように設定する4つの単位です。

- JVM単位の設定

LPQの有効化方法のうち、JVMオプションを使用して有効にした場合、JVMから送信するすべてのメッセージはLPQを介して送信されます。

- コネクション・ファクトリー単位の設定

JVMオプションに以下とおりLPQを使用するコネクション・ファクトリー名を追加すると、該当するコネクション・ファクトリーを利用して作成されたコネクションから送信されるすべてのメッセージはLPQを介して送信されます。各コネクション・ファクトリーの名前はコンマ(,)で区切ります。

```
jeus.jms.client.connection-factory-for-lpq=<ConnectionFactoryName1,ConnectionFactoryName2,...>
```

参考

JMS仕様上、コネクション・ファクトリー名への制約はありませんが、区切り子の(,)が含まれていると、上記のオプションを正常に適用できないので、LPQ機能を使用することはできません。

- セッション単位の設定

JEUS専用のメッセージ・プロデューサー・オブジェクトの以下のAPIを使用すると、該当のセッションに送信するすべてのメッセージをLPQを介して送信するかどうかを設定することができます。

```
public void setLPQOnly(boolean lpqOnly);
```

- メッセージ単位の設定

各メッセージを、LPQを介して送信するかどうかを指定する方法です。JEUS専用のメッセージ・プロデューサー・オブジェクトの以下のAPIを使用するか、メッセージのユーザー・プロパティを設定する方法があります。

```
// JEUS専用のAPI
public void sendWithLPQ(Message message);
public void sendWithLPQ(Message message, int deliveryMode, int priority,
    timeToLive);
public void sendWithLPQ(Destination destination, Message message);
public void sendWithLPQ(Destination destination, Message message, int
    deliveryMode, int priority, timeToLive);

// JMS      User Property
Message msg = session.createMessage();
msg.setBooleanProperty("JMS_JEUS_USE_LPQ", true);
```

参考

JEUS専用のAPIを使用してLPQを有効にする場合は、該当するセッションでのみLPQを使用できるため、それ以外のセッションから送信されるメッセージにはLPQを設定しても無視されます。

6.6.3. LPQリスナーの設定

メッセージの送信にLPQを使用するとメッセージが非同期的に処理されるため、実際にメッセージが送信されることをクライアントでは把握できません。そのため、LPQではメッセージ送信の結果と正確な時点が把握できるようにリスナーを提供しています。

以下は、リスナーを介してメッセージ送信の結果を取得するため、インターフェースとして提供されるリスナーのjeus.jms.LPQMessageListenerです。

```
package jeus.jms;

public interface LPQForwardListener {

    /**
     * メッセージの送信が完了したときのイベント
     * @param message 送信されたメッセージ
     */
    public void onComplete(Message message);

    /**
     * メッセージの送信中に例外が発生したときのイベント
     * @param message 送信しようとしたメッセージ
     * @param e 発生した例外
     */
    public void onException(Message message, Exception e);

    /**
     * メッセージの送信に失敗したときのイベント
     *
     * @param message 送信に失敗したメッセージ
     * @param cause 送信に失敗した原因
     */
    public void onFailure(Message message, Throwable cause);
}
```

実装されたLPQMessageListenerをJMSセッションのJEUS専用オブジェクトであるJeusSessionに以下のAPIを使用して登録すると、メッセージ送信の結果が提供されます。

```
public void setLPQMessageListener(jeus.jms.LPQMessageListener lpqMessageListener);
```

なお、JEUS専用オブジェクトを使用してLPQを有効にする場合、JeusSessionの以下のAPIを使用してLPQの有効化とリスナーの登録を同時に行うことができます。

```
public void startLPQ(jeus.jms.LPQMessageListener lpqMessageListener);
```

6.6.4. LPQの設定

LPQは、失敗した送信に対する処理、接続が切断されたときの動作、ストアの位置またはサイズに関する設定が必要です。

LPQの設定項目

以下は、LPQ設定項目についての説明です。

- 共通項目

項目	タイプ	説明
jeus.lpq.name	String	使用するLPQの名前を設定します(デフォルト値: JEUS_LPQ)
jeus.lpq.max-message-count	int	LPQが一度に処理できる最大メッセージ数を設定します(デフォルト値: 819200)
jeus.lpq.time-to-live	long	LPQ内のメッセージが存在できる最大時間を設定します(デフォルト値: 43200000ms (12時間))

- 送信関連項目

項目	タイプ	説明
jeus.lpq.retry-limit	int	送信に失敗したメッセージの再送信をトライする回数を設定します。0以下の値に設定された場合、無限に再試行します(デフォルト値: -1)
jeus.lpq.retry-interval	long	メッセージを再送信するとき、毎回再試行の間隔を設定します(デフォルト値: 1000ms)
jeus.lpq.retry-interval-increment	long	メッセージを再送信するとき、毎回再試行の間隔の増加量を設定します(デフォルト値: 0ms)

- 再接続関連項目

項目	タイプ	説明
jeus.lpq.reconnect-retry-interval	long	接続が切断されたとき、再接続をトライする間隔を設定します(デフォルト値: 5000ms)

- ストア関連項目

項目	タイプ	説明
jeus.lpq.store.store-mode	int	使用するストアのタイプを設定します。1はジャーナルストアを使用し、2はメモリーストアを使用します(デフォルト値: 1)

項目	タイプ	説明
jeus.lpq.store.journal.store-base-dir	String	ストアを作成するディレクトリー名を設定します。このディレクトリー名は、各LPQ設定において一意である必要があり、同時に2つ以上のアクセスは許可しません。 相対パスに設定される場合、設定ファイルが見つかった位置またはJVMが実行された位置の下位に設定されます(デフォルト値: JEUS_LPQ_STORE)
jeus.lpq.store.journal.initial-log-file-count	int	ジャーナルストアを作成するとき、初期に作成するログファイルの数を設定します(デフォルト値: 2)
jeus.lpq.store.journal.max-log-file-count	int	作成するログファイルの最大数を設定します(デフォルト値: 10)
jeus.lpq.store.journal.log-file-size	String	ログファイルのサイズを指定します。(デフォルト値: 64M) 整数型の値または数字の後ろに文字を付けて設定できます <ul style="list-style-type: none"> – 'K'(KiloBytes) – 'M'(MegaBytes) – 'G'(GigaBytes)

LPQの設定方法

以下は、LPQ設定方法についての説明です。設定方法の優先順位は、ランタイム・プロパティの設定>設定ファイル> JVMオプションの順です。

● 設定ファイル

LPQ設定ファイルが指定された場所に存在していれば、LPQが有効化される時、そのファイルを読み込んで有効になるLPQに該当する設定を適用します。指定された場所は、DEPLOYED_HOME/myApp/WEB-INF/、DEPLOYED_HOME/myApp/META-INF/、DEPLOYED_HOME/myApp/であり、この順で設定ファイルを検索します。設定ファイルの詳細パスのデフォルト値は「jeus-lpq.properties」であり、以下のオプションを使用して変更することができます。

```
-Djeus.jms.client.lpq-configuration-path=jeus-lpq.properties
```

以下は、LPQ設定ファイルの例です。このサンプル・ファイルは、JEUS_HOME/templates/lpq/にjeus-lpq.propertiesという名前で存在しています。

```
#JEUS Local-Persistent-Queue Configuration
#[ commons ]
jeus.lpq.name=JEUS_LPQ
```

```
jeus.lpq.max-message-count=819200
jeus.lpq.time-to-live=43200000

#[forward]
jeus.lpq.retry-limit=-1
jeus.lpq.retry-interval=1000
jeus.lpq.retry-interval-increment=0

#[reconnect]
jeus.lpq.reconnect-retry-interval=5000

#[store]
jeus.lpq.store.store-mode=1
#[journal-store]
jeus.lpq.store.journal.store-base-dir=JEUS_LPQ
jeus.lpq.store.journal.max-log-file-count=10
jeus.lpq.store.journal.initial-log-file-count=2
jeus.lpq.store.journal.log-file-size=64M
```

- システム・プロパティ

上記の設定項目は、各項目名をキーにして、システム・プロパティとして設定することもできます。たとえば、LPQの名前を設定すると、JEUS MQクライアントを実行するとき、以下のようにJVMオプションを設定することができます。

```
-Djeus.lpq.name=<jeus lpq name>
```

以下のようにクライアント・ソースにコードを追加して、ランタイムの設定を変更することもできます。

```
System.setProperty("jeus.lpq.name", <jeus lpq name>);
```


付録 A. ジャーナルストアの追加属性

本章では、ジャーナルストアに追加で設定できる属性について説明します。これは詳細属性であり、プラットフォーム間で発生し得る性能問題や機能の相違性を解決するために使用することができます。

A.1. プロパティ・リファレンス

- jeus.store.journal.control-file-name

説明	制御ファイルを他の名前に指定する場合に使用します。特別な場合以外はこの設定は使用しません
デフォルト値	control.dat

- jeus.store.journal.log-file-mode

説明	ログ・ファイルを開く際に使用するファイル・モードを指定します。rw、rws、rwdなどを設定でき、rwに設定すると強制的にfile forceを実行します
デフォルト値	rwd

- jeus.store.journal.max-move-count

説明	ログ・ファイルがいっぱいになった場合、必要なスペースを確保するためにオーバーフロー処理を行い、レコードの位置を移します。長期間使用していないレコードがログ・ファイルに残っていると性能が低下するため、第2のストアに移動します。この際、レコードを移動させる最大回数を指定します
デフォルト値	1

- jeus.store.journal.overflow-factor

説明	現在使用中のログ・ファイルに、この値以下の空きスペースがある場合、オーバーフローをチェックします
デフォルト値	0.5

- jeus.store.journal.min-buffer-size

説明	レコードを一度に書き込むための最小のバッファ・サイズを指定します
デフォルト値	4K

- jeus.store.journal.max-buffer-size

説明	レコードを一度に書き込むための最大のバッファ・サイズを指定します
デフォルト値	4MB

- jeus.store.journal.use-direct-buffer

説明	ファイルチャネルを使用して書き込む場合、DirectByteBufferを使用するかどうかを指定します
デフォルト値	false

- jeus.store.journal.max-waiting-thread-count

説明	バッファの書き込み待ちの最大スレッド数を指定します
デフォルト値	32

- jeus.store.journal.sync-forcefully

説明	ファイルチャネルを開くとき、ログファイル・モードに関係なく、強制的にsyncを呼び出すように設定します。一部のOSではファイル・モードが正常に適用されない場合があります
デフォルト値	false

付録 B. JDBC永続ストアの列

JEUS MQで永続ストア(Persistence store)をJDBCに設定したときに生成されるテーブルの列について説明します。

B.1. メタ情報テーブル

JEUS MQの永続ストアについての情報を保存するテーブルです。

項目	タイプ	説明
SERVER_NAME	VARCHAR(255)	JEUS MQのサーバー名
VERSION	BIGINT	JEUS MQのバージョン情報

B.2. デスティネーション・テーブル

デスティネーション情報を保存するテーブルです。

項目	タイプ	説明
DT_ID	BIGINT	デスティネーションのID
DT_NAME	VARCHAR(255)	デスティネーションの名前
DT_QUEUE	BIT	デスティネーションがキューなのかトピックなのかを指定 – キュー : true – トピック : false
DT_VALID	BIT	デスティネーションが有効な状態なのかを判断
DT_LVID	BIGINT	デスティネーションの現在のバージョン
DT_DYNAMIC	BIT	デスティネーションが動的に作成されたのかを判断
DT_OBJECT	BLOB	デスティネーションのバイナリ・データ

B.3. 永続サブスクライバー・テーブル

永続的サブスクライバー情報を保存するテーブルです。

項目	タイプ	説明
DS_ID	BIGINT	永続サブスクライバーのID
DS_CLIENT_ID	VARCHAR(255)	永続サブスクライバーに指定されたクライアントID
DS_NAME	VARCHAR(255)	永続サブスクライバーの名前
DS_SELECTOR	VARCHAR(255)	永続サブスクライバーに指定されたメッセージ・セレクター
DS_VALID	BIT	永続サブスクライバーが有効な状態なのかを判断
DS_LVID	BIGINT	永続サブスクライバーの現在のバージョン
DT_ID	BIGINT	永続サブスクライバーと接続されたトピックのID
DT_LVID	BIGINT	永続サブスクライバーと接続されたトピックのバージョン

B.4. メッセージ・テーブル

メッセージ情報を保存するテーブルです。

項目	タイプ	説明
MG_ID	BIGINT	メッセージのID
MG_TYPE	TINYINT	メッセージのタイプ
MG_LENGTH	INTEGER	メッセージの長さ
MG_OBJECT	TINYINT[]	メッセージのバイナリ・データ
MG_STATUS	SMALLINT	メッセージの状態
MG_GLOBAL_ORDER_CLOCK	SMALLINT	メッセージに設定されたグローバル・オーダーの時刻
MG_PERSISTENT	TINYINT[]	メッセージが永続的に設定されているのかを判断
DT_ID	BIGINT	メッセージが格納されているデスティネーションのID
DT_LVID	BIT	メッセージが格納されているデスティネーションのバージョン
MG_HEADER_LENGTH	INTEGER	メッセージ・ヘッダーの長さ
MG_HEADER_OBJECT	TINYINT[]	メッセージ・ヘッダーのバイナリ・データ

B.5. サブスクリプション・メッセージ・テーブル

永続サブスクライバーのメッセージ情報を保存するテーブルです。

項目	タイプ	説明
DM_ID	BIGINT	永続サブスクライバー・メッセージのID
DM_STATUS	SMALLINT	永続サブスクライバー・メッセージの状態
DM_LVID	BIGINT	永続サブスクライバー・メッセージの現在のバージョン
MG_ID	BIGINT	実際のメッセージのID

項目	タイプ	説明
DS_ID	BIGINT	永続サブスクライバー・メッセージが格納されている永続サブスクライバーのID

B.6. トランザクション・テーブル

トランザクション情報を保存するテーブルです。

項目	タイプ	説明
TR_ID	BIGINT	トランザクションのID
TR_STATUS	TINYINT	トランザクションの状態
TR_OBJECT	TINYINT[]	トランザクションのバイナリ・データ

索引

B

BytesMessage, 20

F

file force, 105

J

JDBCの永続ストア, 63

JEUS MQクラスタリング, 51

jeus.store.journal.control-file-name, 105

jeus.store.journal.log-file-mode, 105

jeus.store.journal.max-buffer-size, 106

jeus.store.journal.max-move-count, 105

jeus.store.journal.max-waiting-thread-count, 106

jeus.store.journal.min-buffer-size, 105

jeus.store.journal.overflow-factor, 105

jeus.store.journal.sync-forcefully, 106

jeus.store.journal.use-direct-buffer, 106

JNDI, 4

M

MapMessage, 20

Message, 5

O

ObjectMessage, 20

R

RequestBlockingTime, 65

S

StreamMessage, 20

T

TextMessage, 5, 20

あ

一時デスティネーション作成メソッド, 68

永続メッセージ・コンシューマ作成メソッド, 68

か

高信頼性メッセージの送信, 97

キューブラウザ作成メソッド, 67

グローバル・オーダー, 78

コネクション, 11

コネクション・コンシューマの復旧, 66

コネクション・ファクトリー・クラスタリング, 51

さ

ジャーナルログの永続ストア, 63

セッションの復旧, 66

た

デスティネーション, 8

デスティネーション・クラスタリング, 52

デスティネーション作成メソッド, 67

は

フェイルバック, 64

ま

メッセージ・グループ, 79

メッセージ・コンシューマ作成メソッド, 68

メッセージ・ソート, 76

メッセージ・ブリッジ, 73

メッセージ・プロデューサー作成メソッド, 68

メッセージ作成メソッド, 67

メッセージ管理機能, 80

