HITACHI
Inspire the Next

Cosminexus V11 アプリケーションサーバ
Cosminexus Reliable Messaging

解説・手引・文法・操作書

3021-3-J19-50

前書き

■ 対象製品

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」の前書きの対象製品の説明を参照してください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

■ 謝辞

Reliable Messaging は，経済産業省が 2003 年度から 3 年間実施した「ビジネスグリッドコンピューティングプロジェクト」の技術開発の成果を含みます。

■ 発行

2024 年 2 月 3021-3-J19-50

■ 著作権

変更内容

変更内容(3021-3-J19-50) uCosminexus Application Server 11-40，uCosminexus Client 11-40，uCosminexus Developer 11-40，uCosminexus Service Architect 11-40，uCosminexus Service Platform 11-40

追加・変更内容	変更個所
記載内容は変更なし（リンク情報だけを変更した）。	-

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルをお読みになる際の前提情報については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」のはじめにの説明を参照してください。

目次

1 概要

Reliable Messaging はアプリケーションサーバシステムを基盤として，アプリケーションをメッセージの非同期通信によって連携させるメッセージングミドルウェアです。

Reliable Messaging では，永続版リソースアダプタと非永続版リソースアダプタの 2 種類のリソースアダプタを提供します。

この章では，Reliable Messaging の概要，必要な前提製品，システムの形態，およびシステム構築・運用の流れとマニュアルの参照先について説明します。

1.1　マニュアルの説明

このマニュアルで使用している表記について説明します。

文法の記号

このマニュアルで使用する各種の記号を説明します。

属性表示記号

ユーザ指定値の範囲などを説明する記号です。

属性表示記号	意味
～	この記号のあとにユーザ指定値の属性を示します。
《　》	ユーザが指定を省略したときの解釈値を示します。
＜　＞	ユーザ指定値の構文要素を示します。
((　))	ユーザ指定値の指定範囲を示します。

構文要素記号

ユーザ指定値の内容を説明する記号です。

構文要素記号	意味
＜英字＞	半角のアルファベット（A～Z，a～z）
＜数字＞	半角の数字（0～9）
＜英数字＞	英字と数字
＜識別子＞	先頭が英字の英数字列と＿（アンダスコア）
＜文字列＞	任意の文字の配列（ただし，2バイト文字は除く）

文法記述記号

記述形式を説明する記号です。

文法記述記号	意味
［　］	この記号で囲まれている項目は省略してもよいことを示します。 （例） 〔-l 最大メッセージ長〕 -l オプションとそのオペランドを指定するか，何も指定しないことを示します。
{　}	この記号で囲まれている複数の項目のうちから一つを選択することを示します。 （例） {local ｜ shr_receive} local と shr_receive のうち，どちらかを指定することを示します。
｜ （ストローク）	この記号で区切られた項目は選択できることを示します。 （例） {local ｜ shr_receive}

文法記述記号	意味
	local と shr_receive のうち，どちらかを指定することを示します。
＜ ＞	この記号で囲まれている項目はユーザの環境によって異なることを表します。 (例) **<HiRDB サーバのホスト名または IP アドレス>** ユーザの環境で使用している HiRDB サーバのホスト名または IP アドレスを指定することを示します。

このマニュアルで使用するディレクトリ名

このマニュアルでは，原則として Windows の表記を使用しています。UNIX の場合は，環境変数の頭に「$」を付加し，ディレクトリセパレータを「/」で読み替えてください。読み替えの例を次の表に示します。

Windows の場合	UNIX の場合
%HRMDIR%¥sql	$HRMDIR/sql

リソースアダプタの違いによる機能相違点の表記

このマニュアルの対象製品は，永続版リソースアダプタと非永続版リソースアダプタの 2 種類のリソースアダプタを提供します。リソースアダプタによって記述を書き分ける場合，次に示す表記を使用して，それぞれの説明にリソースアダプタの種別を明記しています。

表記	意味
永続版リソースアダプタの場合	永続版リソースアダプタに該当する表記です。
非永続版リソースアダプタの場合	非永続版リソースアダプタに該当する表記です。

1.2　Reliable Messaging の概要

Reliable Messaging は，アプリケーションサーバシステム上のアプリケーションがメッセージを使用して非同期に通信するためのミドルウェアです。メッセージ通信機能をアプリケーションに提供します。

送信側アプリケーションがメッセージを送信すると，メッセージはキューに登録されます。受信側アプリケーションがメッセージを受信すると，キューからメッセージが取り出されます。

こうして，キューを介してメッセージを送受信することによって，送信側と受信側のアプリケーションを同時に動作させる必要のない非同期通信ができます。

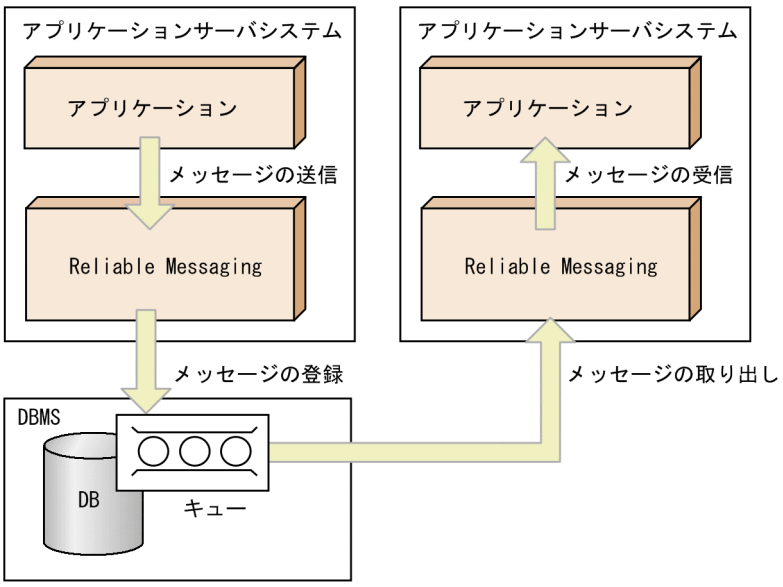
Reliable Messaging では，永続版リソースアダプタと非永続版リソースアダプタを提供します。永続版リソースアダプタと非永続版リソースアダプタの概要について説明します。

1.2.1　永続版リソースアダプタの概要

永続版リソースアダプタでは，メッセージやキューの属性情報，システムステータスなどを DB で管理し**永続化**しています。永続化とは，不揮発性の記憶媒体にデータを保存し，変化しない状態にすることです。

永続版リソースアダプタの概要を次の図に示します。

図 1-1　永続版リソースアダプタの概要



永続版リソースアダプタには，次に示す特長があります。

- 高信頼のメッセージ管理

  Reliable Messaging がメッセージを登録するキューは，DB のテーブル上に作成されます。DB にメッセージを永続化することで，メッセージの再現性を保持し，高信頼のメッセージ管理ができます。

- 高信頼なメッセージ通信

確実に 1 回の配送保証，または順序保証の QoS（通信品質）を，ユーザが通信相手ごとに選択できます。

- JMS インタフェースの採用

  Reliable Messaging はアプリケーションに JMS インタフェースを提供します。JMS インタフェースは，Java でのメッセージング通信のための API です。

- インターネット経由のシステム連携

  通信手段として SOAP および HTTP を利用します。HTTP は幅広く利用されているプロトコルです。多くの企業のファイアウォールは HTTP（80 番ポート）による内外の通信を許可しているので，ファイアウォールを介した接続が比較的容易に実現できます。また，SSL を使用すると，秘密性，完全性などのセキュリティ機能を実現できます。

- WS-Reliability による通信のサポート

  オープンで高信頼な通信プロトコルである WS-Reliability によって，Reliable Messaging 間での通信や，WS-Reliability に準拠した他ベンダのシステムとの通信ができます。WS-Reliability は標準化団体 OASIS で標準化された仕様です。

- DB Connector for Reliable Messaging とのコネクション共有

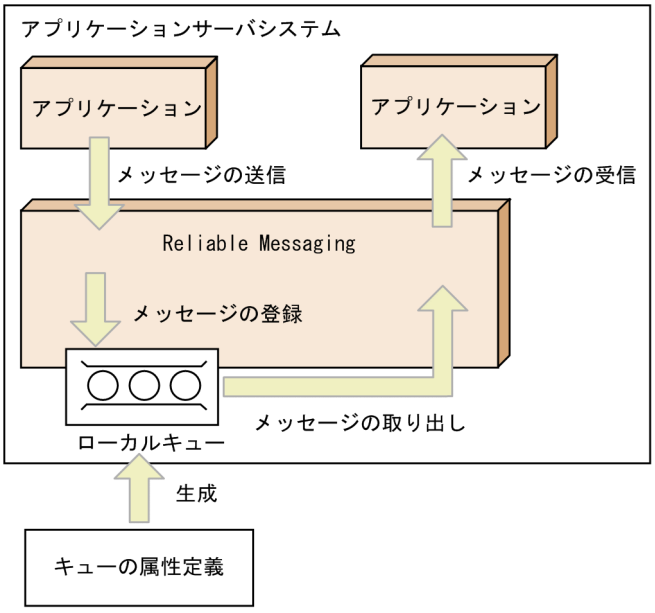
  Reliable Messaging と DB Connector for Reliable Messaging が同じ DB に接続する場合，Reliable Messaging と DB Connector for Reliable Messaging のコネクションを共有することによって，DB のコネクションの有効利用とトランザクションの 1 相コミット化による性能向上の両方を実現できます。

1.2.2　非永続版リソースアダプタの概要

非永続版リソースアダプタでは，キューなどのシステム情報を DB で管理しないで**非永続化**します。非永続化とは，揮発性のメモリだけにデータを保存して，処理を実施することです。

非永続版リソースアダプタの概要を次の図に示します。

図 1-2　非永続版リソースアダプタの概要



非永続版リソースアダプタには，次に示す特長があります。

- DB 接続の不要

  キューなどのシステム情報を DB で管理しないため，DB との接続が不要になります。その結果，レスポンス性能の向上だけでなく，システム導入・構築，運用の簡易化とコストダウンが期待できます。

- JMS インタフェースの採用

  永続版リソースアダプタの場合と同様に，非永続版リソースアダプタでもアプリケーションに JMS インタフェースを提供します。JMS インタフェースは，Java でのメッセージング通信のための API です。

注意事項

非永続版リソースアダプタの場合，使用できるキューはローカルキューだけです。

1.3　ソフトウェア構成

Reliable Messaging が動作するには，幾つかの前提製品が必要です。

Reliable Messaging を使用するシステムのソフトウェア構成について，永続版リソースアダプタの場合を図 1-3 に，非永続版リソースアダプタの場合を図 1-4 に示します。

図 1-3　永続版リソースアダプタを使用する場合のシステムのソフトウェア構成

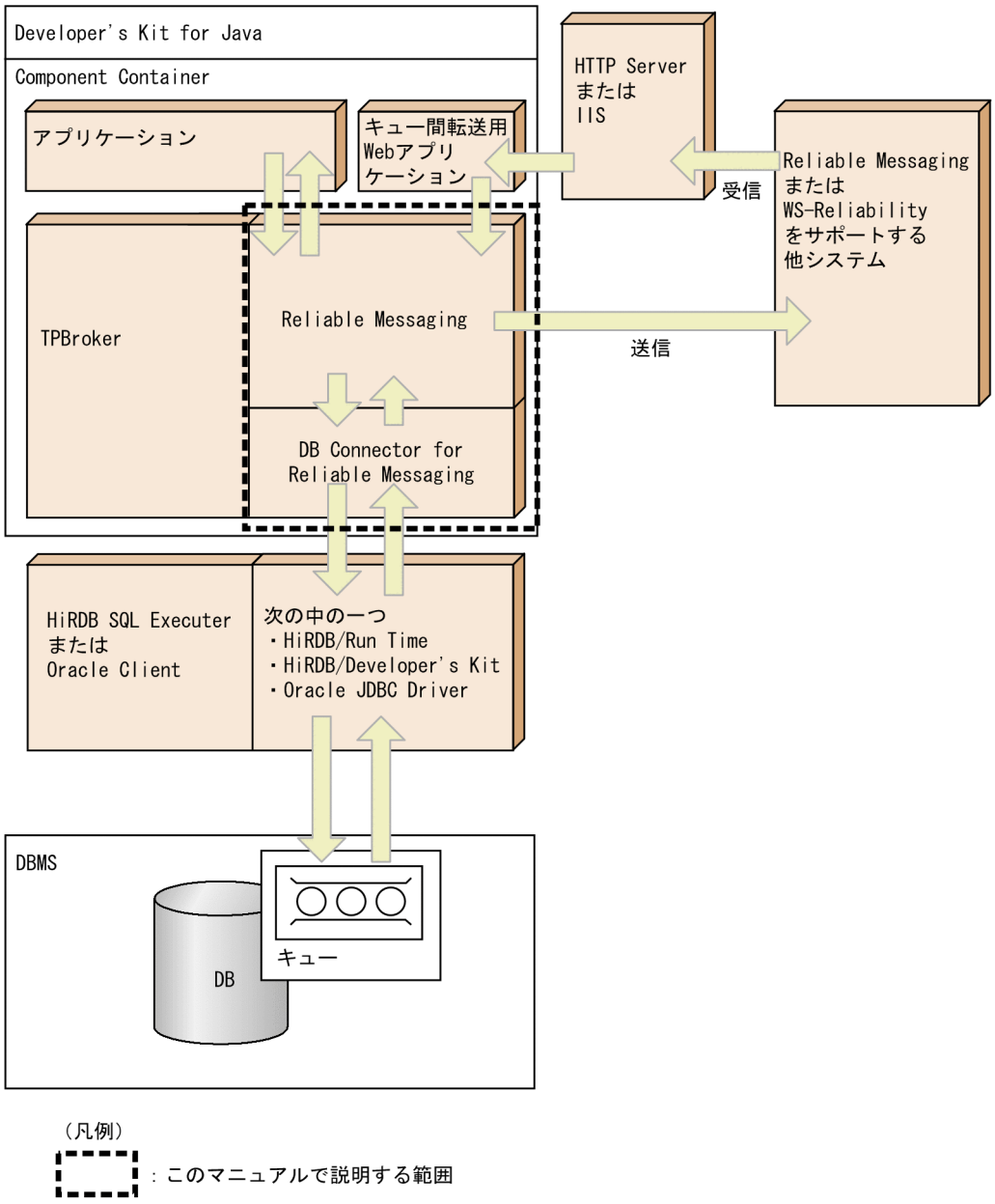
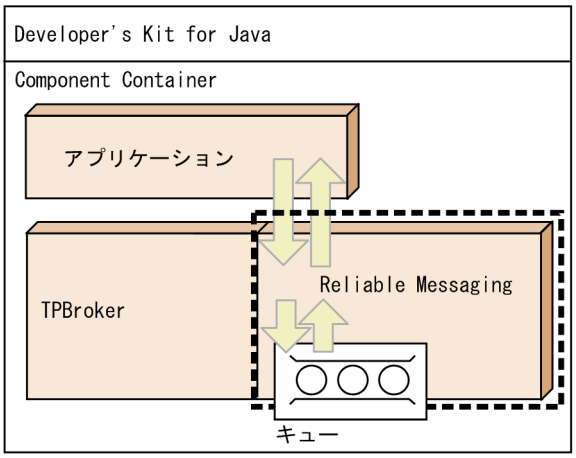


図 1-4　非永続版リソースアダプタを使用する場合のシステムのソフトウェア構成



（凡例）

：このマニュアルで説明する範囲

各要素について説明します。

- Developer's Kit for Java

  Application Server が提供する Java 2 SDK, Standard Edition です。

  詳細については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」を参照してください。

- Component Container

  Reliable Messaging およびアプリケーションが動作する J2EE サーバです。

  Reliable Messaging はリソースアダプタとして動作します。一つの J2EE サーバ上で複数の Reliable Messaging を動作させることができます。

  J2EE サーバについては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」を参照してください。

- アプリケーション

  ユーザが業務に合わせて作成するアプリケーションです。アプリケーションサーバシステム上で動作するアプリケーションとして，Reliable Messaging が提供する JMS インタフェースを使用して作成します。

  サーブレット，JSP または EJB として作成します。

- キュー

  アプリケーションが送信したメッセージを Reliable Messaging が登録するためのキューです。

  - 永続版リソースアダプタの場合

    ユーザは，Reliable Messaging が提供するコマンド（hrmmkque コマンド）を使用して DB のテーブル上にキューを作成します。

    自システムからのメッセージの登録と取り出しだけができるローカルキュー，相手システムとの通信に使用するための共用キューなどの種類があります。詳細については，「2.2　キューの種類」を参照してください。

  - 非永続版リソースアダプタの場合

ユーザは，キュー作成ファイルにキュー属性を定義して Reliable Messaging 内にキューを作成します。

非永続版リソースアダプタでは，ローカルキューだけを使用できます。

- TPBroker

  分散オブジェクト環境でのオブジェクト間通信やトランザクション制御を提供する製品です。

  J2EE サーバは TPBroker を利用してトランザクションマネジャとして動作します。Reliable Messaging はリソースマネジャとして J2EE サーバと連携します。

  詳細については，マニュアル「TPBroker ユーザーズガイド」を参照してください。

- DB Connector for Reliable Messaging

  DB にアクセスするための J2EE リソースアダプタとなる，スタンドアロンモジュールです。Reliable Messaging（永続版リソースアダプタ）が DB にアクセスするときに使用します。

  詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」を参照してください。

- HiRDB/Run Time，HiRDB/Developer's Kit または Oracle JDBC Driver

  他マシン上の DB にアクセスするためのクライアント機能を提供する製品です。Reliable Messaging（永続版リソースアダプタ）と DBMS を同じマシン上で運用する場合は不要です。

  HiRDB/Run Time および HiRDB/Developer's Kit については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。Oracle JDBC Driver については，Oracle のマニュアルを参照してください。

- HiRDB SQL Executer

  SQL を実行するためのクライアントプログラムです。Reliable Messaging（永続版リソースアダプタ）が使用するシステムの管理情報を DB のテーブル上に作成するときに使用します。

  HiRDB SQL Executer については，HiRDB SQL Executer のドキュメントを参照してください。

- DBMS

  Reliable Messaging（永続版リソースアダプタ）では，メッセージを格納するキューや管理情報を DB のテーブル上に作成します。この DB を管理するための DBMS です。

  Reliable Messaging では，DBMS として HiRDB または Oracle※を使用できます。HiRDB のシステム設計については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。Oracle のシステム構築については，Oracle のマニュアルを参照してください。

  注※

  トランザクションのサポートレベルが XATransaction の場合，Oracle RAC 機能は使用できません。

1.4　システム形態

Reliable Messaging のシステム形態は，自システム内だけの通信であるか，システム間の通信があるかによって異なります。各システムの形態と DB サーバを共有する形態について説明します。

1.4.1　単一システムでのアプリケーション連携

自システム内のアプリケーション間で，ローカルキューを介してメッセージを送受信する形態です。この形態は，永続版リソースアダプタ，非永続版リソースアダプタ共に利用できます。

自システムの送信側アプリケーションがメッセージを送信するとき，ローカルキューにメッセージが登録されます。自システムの受信側アプリケーションがメッセージを受信するとき，ローカルキューからメッセージが取り出されます。

単一システムでのアプリケーション連携を次の図に示します。

図 1-5　単一システムでのアプリケーション連携（永続版リソースアダプタの場合）

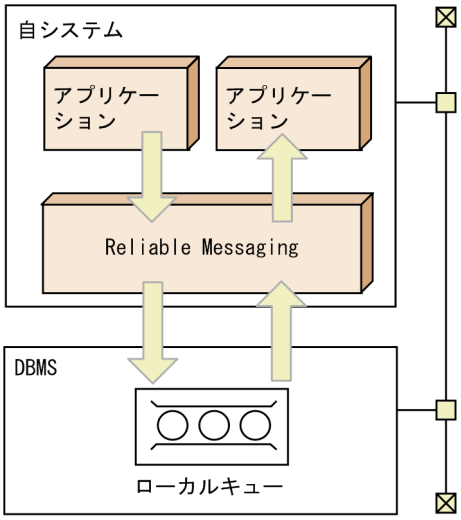
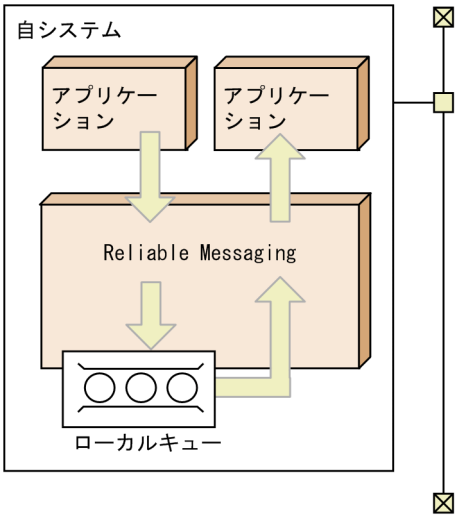


図 1-6　単一システムでのアプリケーション連携（非永続版リソースアダプタ場合）



1.4.2　複数システム間でのアプリケーション連携

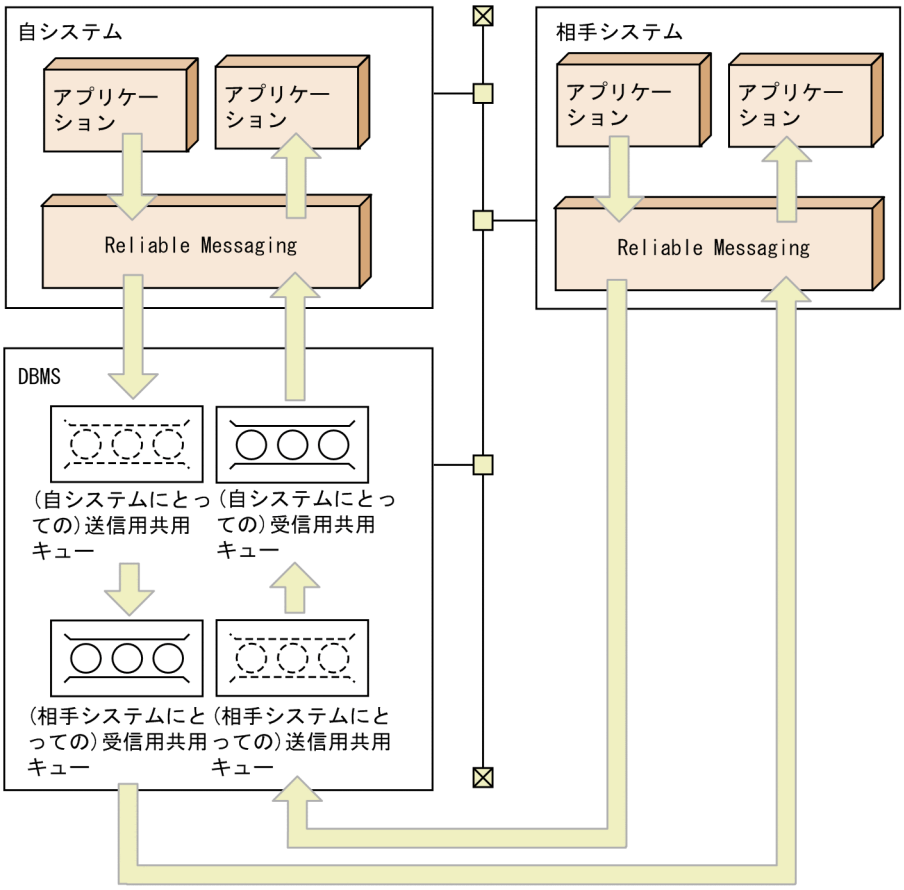
自システムと相手システムが共用キューを共有してメッセージを送受信する形態です。この形態は，永続版リソースアダプタだけが利用できます。

自システムのアプリケーションがメッセージを送信するとき，送信用共用キューにメッセージが登録されます。送信用共用キューは定義だけの仮想的なキューです。メッセージが格納される領域の実体は，相手システムにとっての受信用共用キューにあります。相手システムのアプリケーションがメッセージを受信するとき，（相手システムにとっての）受信用共用キューからメッセージが取り出されます。

自システムおよび相手システムは，Reliable Messaging で構築されたシステムです。

複数システム間でのアプリケーション連携を次の図に示します。

図 1-7　複数システム間でのアプリケーション連携



1.4.3　複数システムのキュー間でメッセージを転送するアプリケーション連携

自システムから，他システムに組み込まれている Reliable Messaging または他ベンダのメッセージングシステムに向けてメッセージを送信する形態です。この形態は，永続版リソースアダプタだけが利用できます。

複数システムのキュー間でメッセージを転送する場合に，自システムで使用するキューを転送キューといいます。転送キューについては，「2.2.2　転送キュー」を参照してください。

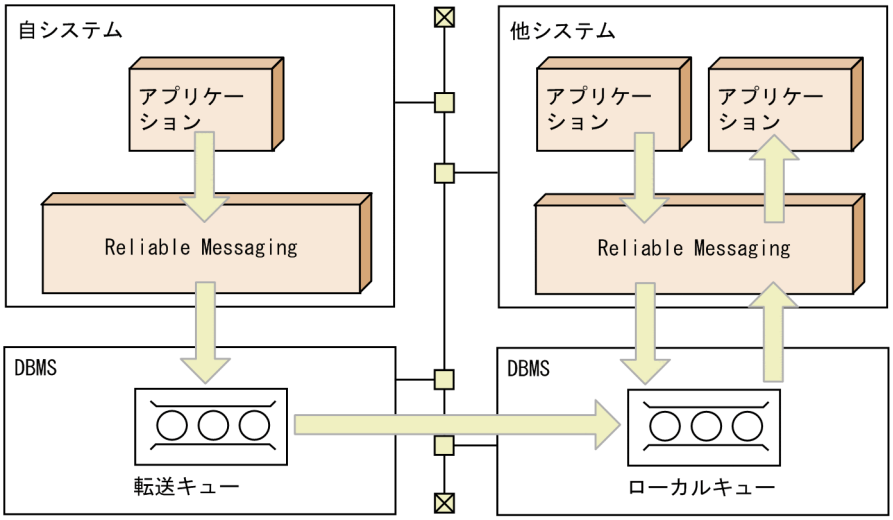
転送キューを使うには，転送先の情報が必要となります。他ベンダに向けたメッセージ転送は，BytesContainer インタフェースを使用することで実現できます。BytesContainer インタフェースについては，「7.8.2　BytesContainer インタフェース」を参照してください。

アプリケーションからメッセージを他システムに転送する手順を次に示します。

1. 転送キューにメッセージを登録します。
2. 転送キューにメッセージが登録された時点で他システムに向けたメッセージの転送が開始されます。ただし，コマンドでサーバ間転送のメッセージ送受信を抑止することもできます。また，転送キューに登録済みのメッセージは，アプリケーションからは取り出せません。

複数システムのキュー間でメッセージを転送するアプリケーション連携を次の図に示します。

図 1-8　複数システムのキュー間でメッセージを転送するアプリケーション連携



1.4.4　DB サーバの共有

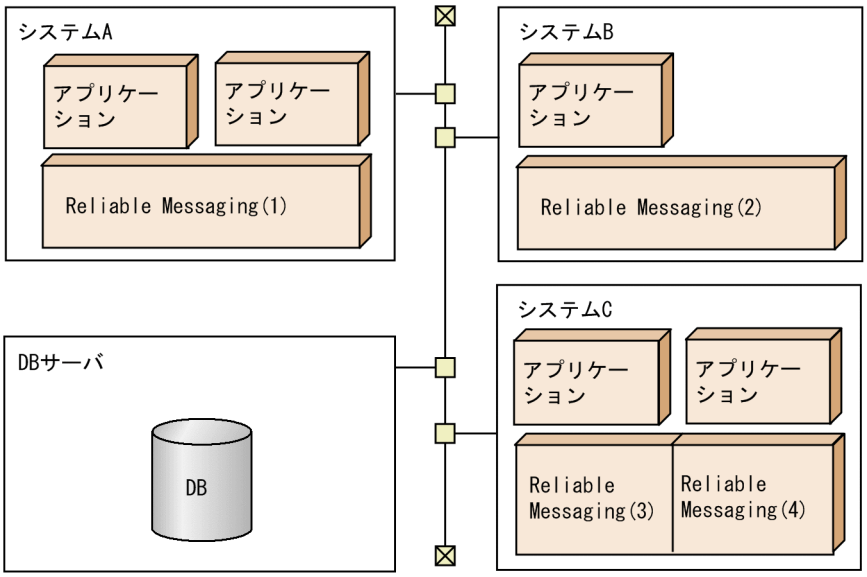
Reliable Messaging は DB サーバが動作するマシンとは異なるマシンで動作できます。この場合，マシン間を LAN 回線などで接続します。DB サーバの共有は，永続版リソースアダプタだけが利用できます。

1 台のマシンには一つ以上の Reliable Messaging を動作させることができます。また，複数の Reliable Messaging で一つの DB サーバを共有できます。

しかし，一つの Reliable Messaging が複数の DB サーバに接続することはできません。

DB サーバの共有を次の図に示します。

図 1-9　DB サーバの共有



(1)　管理情報テーブルとの対応関係

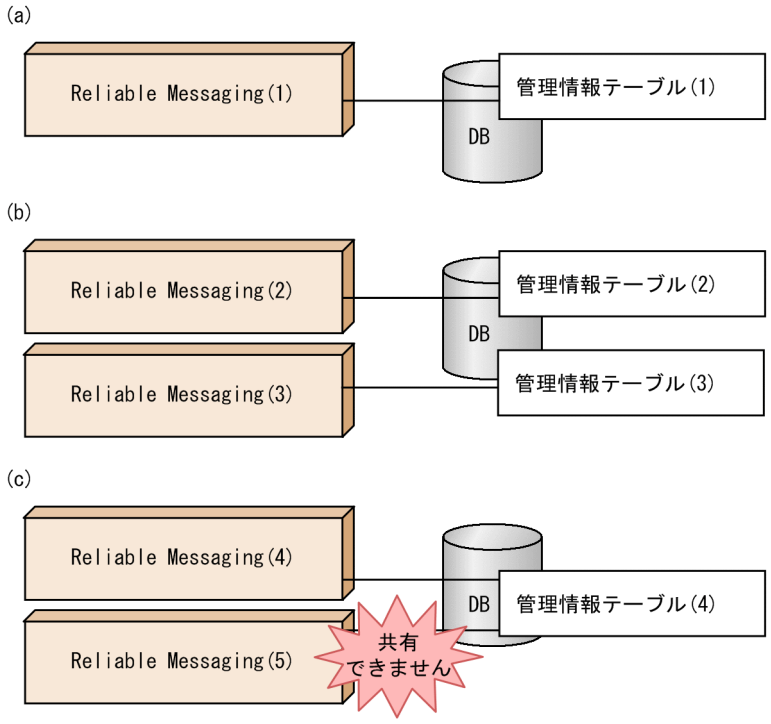
Reliable Messaging は，メッセージを格納するキューや管理情報を DB 上の管理情報テーブルに保存します。管理情報テーブルは，Reliable Messaging に 1：1 に対応する必要があります。複数の Reliable Messaging で一つの管理情報テーブルを共有できません。特に，一つの J2EE サーバ上で複数の Reliable Messaging を動作させる場合はご注意ください。

管理情報テーブルは RMSystemName プロパティ指定値によって特定の Reliable Messaging と対応づけられます。複数の Reliable Messaging がある場合は，システム間で RMSystemName プロパティ指定値を一意にしてください。

管理情報テーブルの作成方法については，HiRDB の場合は「3.4.1(3)　Reliable Messaging の管理情報テーブルの作成」を，Oracle の場合は「3.4.2(3)　Reliable Messaging の管理情報テーブルの作成」を参照してください。

Reliable Messaging と管理情報テーブルの対応を次の図に示します。

図 1-10　Reliable Messaging と管理情報テーブルの対応



図中の番号（a）～（c）について説明します。

(a)

Reliable Messaging（1）が DB 上の管理情報テーブル（1）と 1：1 に対応しています。

(b)

Reliable Messaging（2）が DB 上の管理情報テーブル（2）と 1：1 に対応しています。Reliable Messaging（3）が DB 上の管理情報テーブル（3）と 1：1 に対応しています。

管理情報テーブル（2）と管理情報テーブル（3）は，同じ DB 上に作成されています。

(c)

Reliable Messaging（4）と Reliable Messaging（5）が DB 上の管理情報テーブル（4）を共有しようとしています。これは Reliable Messaging がサポートしない不正な構成です。

(2) XATransaction 利用時のリソースアダプタ間での DB 共有の制限

トランザクションのサポートレベルが XATransaction で，Reliable Messaging のほかにも，DB に関連するリソースアダプタ（DB Connector など）が動作し，各リソースアダプタが同じ DB を使用する場合は，次に示す制限があります。

(a) 一つの J2EE サーバ内の複数のリソースアダプタが共通の DB を利用する場合

接続先 DB が HiRDB のとき

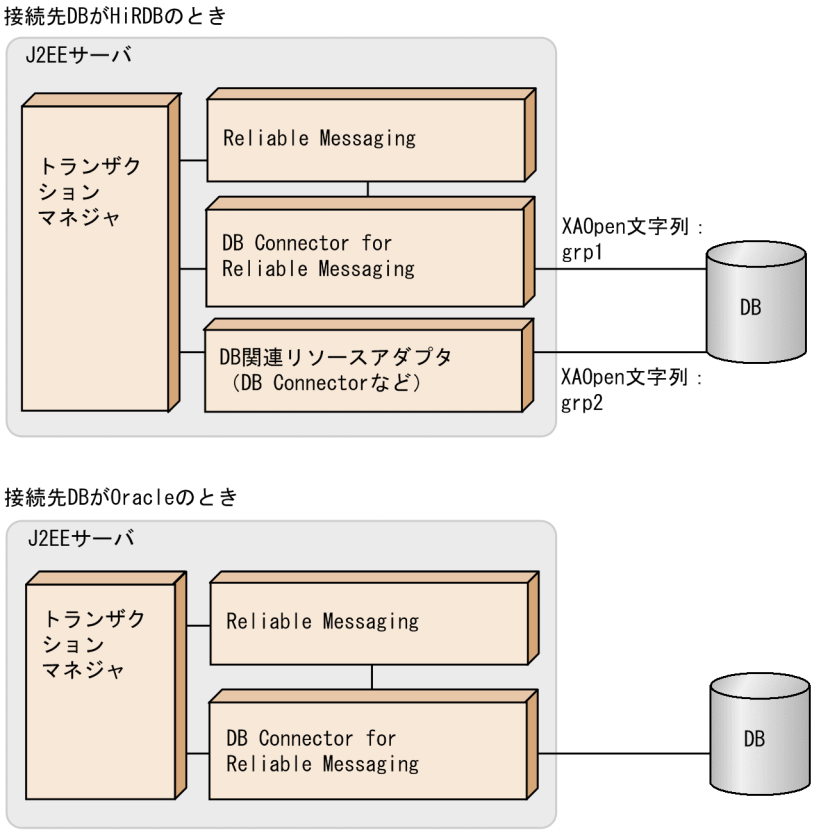
XAOpen 文字列の値は，リソースアダプタごとに一意となるように指定してください。Reliable Messaging では，XAOpen 文字列を連携先の DB Connector に指定します。

接続先 DB が Oracle のとき

連携先の DB Connector との間で DB コネクションを共有するときだけ DB を共有できます。この場合，一つの J2EE サーバ上に複数の Reliable Messaging をデプロイして利用できません。

一つの J2EE サーバ上で複数のリソースアダプタを利用する場合の，リソースアダプタの DB 共有を次の図に示します。

図 1-11 リソースアダプタの DB 共有（J2EE サーバが一つの場合）



(b) 複数の J2EE サーバ内の複数のリソースアダプタが共通の DB を利用する場合

接続先 DB が HiRDB のとき

XAOpen 文字列の値は，リソースアダプタごとに一意となるように指定してください。Reliable Messaging では，XAOpen 文字列を連携先の DB Connector に指定します。

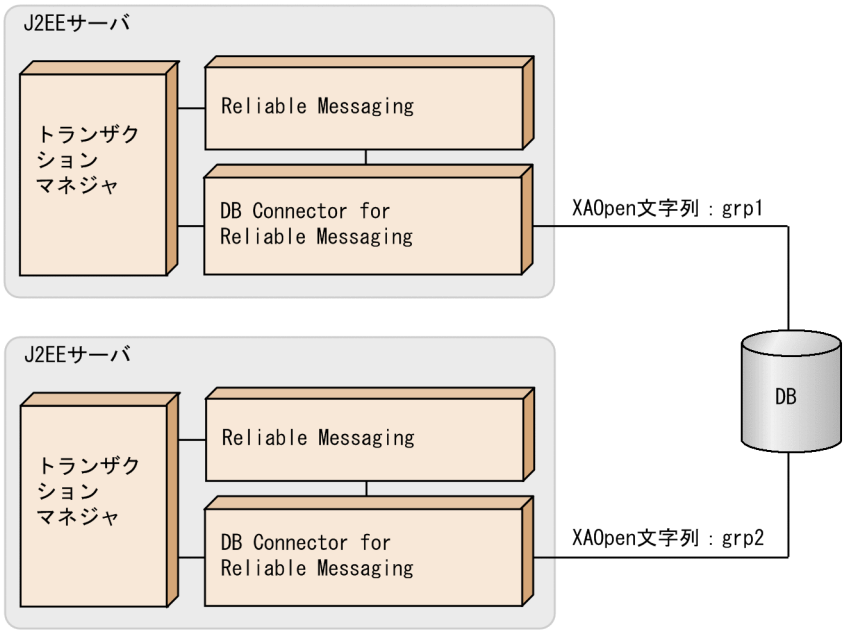
接続先 DB が Oracle のとき

特に制限はありません。

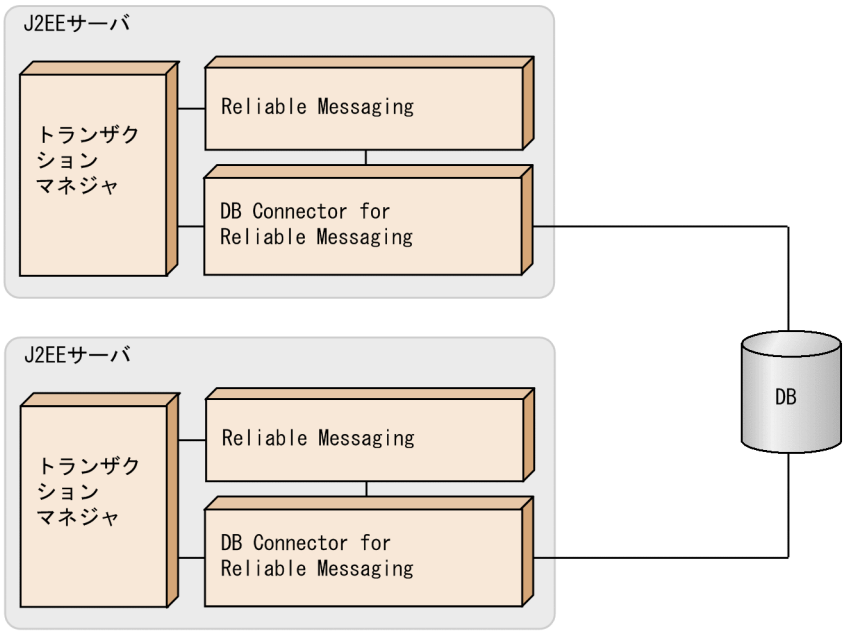
複数の J2EE サーバ上で複数のリソースアダプタを利用する場合の，リソースアダプタの DB 共有を次の図に示します。

図 1-12　リソースアダプタの DB 共有（J2EE サーバが複数の場合）

接続先DBがHiRDBのとき



接続先DBがOracleのとき



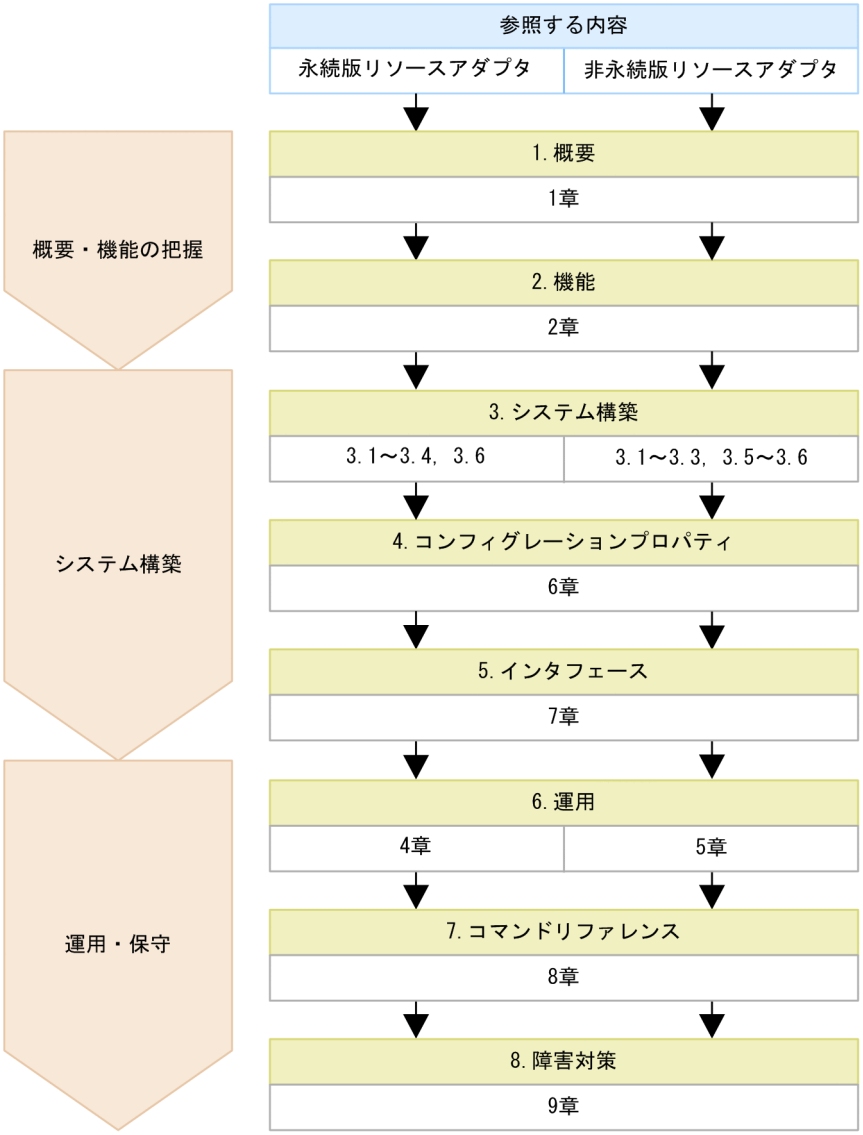
1.5　システム構築・運用の流れとマニュアルの参照先

システム構築・運用の流れとマニュアルでの説明個所の対応について説明します。また，Reliable Messaging の機能とマニュアルでの説明個所の対応についても説明します。

1.5.1　システム構築・運用の流れとマニュアルでの説明個所の対応

Reliable Messaging のシステム構築・運用の流れとマニュアルでの説明個所の対応を次の図に示します。

図 1-13　システム構築・運用の流れとマニュアルでの説明個所の対応



1. 概要

   Reliable Messaging の概要を把握します。

2. 機能

   Reliable Messaging の機能を把握します。

永続版リソースアダプタと非永続版リソースアダプタでは，利用できる機能が異なります。詳細は「1.5.2　Reliable Messaging の機能とマニュアルでの説明個所の対応」を参照してください。

3. システム構築

   Reliable Messaging のシステムを構築します。

   永続版リソースアダプタと非永続版リソースアダプタでは，システムの構築手順が異なります。図 1-13 で示すマニュアルでの説明個所に従ってシステムを構築してください。

4. コンフィグレーションプロパティ

   環境に合わせてコンフィグレーションプロパティを設定します。

   永続版リソースアダプタと非永続版リソースアダプタでは，設定するプロパティが異なります。詳細は「6.1　Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

5. インタフェース

   Reliable Messaging が提供するインタフェースを使用してアプリケーションを実装します。

   非永続版リソースアダプタでは，転送データ相互接続用インタフェースは使用できません。転送データ相互接続用インタフェースについては，永続版リソースアダプタの場合だけ参照してください。

6. 運用

   Reliable Messaging を運用します。

   永続版リソースアダプタの場合は「4.　Reliable Messaging（永続版リソースアダプタ）の運用」を，非永続版リソースアダプタの場合は，「5.　Reliable Messaging（非永続版リソースアダプタ）の運用」を参照してください。

7. コマンドリファレンス

   コマンドを入力することによって Reliable Messaging を運用します。

   永続版リソースアダプタと非永続版リソースアダプタでは，使用できるコマンドが異なります。詳細は「8.2　コマンドの一覧」を参照してください。

8. 障害対策

   Reliable Messaging の障害に対応します。

1.5.2　Reliable Messaging の機能とマニュアルでの説明個所の対応

Reliable Messaging の機能とマニュアルでの説明個所の対応を次の表に示します。

表 1-1　Reliable Messaging の機能とマニュアルでの説明個所の対応

項番	機能	機能詳細	永続版リソースアダプタ	非永続版リソースアダプタ	マニュアルでの説明個所
1	キューの種類	ローカルキュー	○	△※1	2.2.1
		• 転送キュー • 受信用共用キュー	○	×※2	2.2.2，2.2.3，2.2.4，2.2.5

項番	機能	機能詳細	永続版リソースアダプタ	非永続版リソースアダプタ	マニュアルでの説明個所
		• 送信用共用キュー • デッドメッセージキュー			
2	キューの永続性	永続キュー属性	○	×	2.3.1
		非永続キュー属性	○	○	
3	メッセージの受信制御	• プライオリティでの順序 • FIFO での順序 • メッセージセレクタでの順序	○	○	2.6.1，2.6.2
4	メッセージ取り出しモード	• シリアル取り出し属性 • パラレル取り出し属性 • パラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）	○	○	2.3.2
5	メッセージの作成	JMS インタフェース	○	○	7 章
6	メッセージの受け渡し方式	• 値渡し方式 • 参照渡し方式	○	○	2.3.3
7	サーバ間転送	• WS-Reliability（SOAP）によるサーバ間転送 • OpenTP1 システム（TP1/EE）とのサーバ間転送	○	×	2.4
8	Message-driven Bean との連携	Message-driven Bean によるメッセージの配信	○	○	2.6.3
9	コマンド	運用コマンド	○	△※3	8 章
10	DB Connector for Reliable Messaging の機能	• DB Connector for Reliable Messaging とのコネクション共有（トランザクションの 1 相コミット化） • ステートメントキャンセル • 障害調査用 SQL の出力 • 軽微な障害時のコネクション再利用 • コネクション ID の取得	○	×	2.7
11	J2EE インタフェース（JMS, JCA／JTA）	• ローカルトランザクションの利用 • トランザクションマネジャでのトランザクションの利用	○※4	○※4	2.6.6
		• コンテナ管理によるセキュリティ認証（コンテナ認証） • コンポーネント管理によるセキュリティ認証（アプリケーション認証）	○	×※5	2.6.5

項番	機能	機能詳細	永続版リソースアダプタ	非永続版リソースアダプタ	マニュアルでの説明個所
		• シェアリングによるコネクションの共有 • アソシエーションによるコネクションの共有	○	○	2.6.7， マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」
12	転送データ相互接続用 API	転送データ相互接続用インタフェース（BytesContainer）	○	×	7 章
13	キューの抑止制御	キューのメッセージの送受信を抑止	○	△※6	4.2.5，5.2.2，8.3.27
14	キューの障害閉塞	不整合となったキューの部分閉塞	○	×	9.4
15	J2EE サーバ（Application Server）の機能	• リソースへの接続テスト • コネクションの障害検知 • コネクションプーリング	○	○※7	マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」
16	ログとトレース	• 開始停止メッセージログ • Application Server 用メッセージログ • メソッドトレース • PRF トレース	○	○	9.1，9.3
		• 共用キューイベントトレース • 回線トレース	○	×	
17	キューの JNDI ネーミングサービスへの登録	• キュー定義ファイルの使用 • キュー定義ファイルの未使用（キュー作成時の表示名の使用）	○	○※8	3.4.4，3.5.2

（凡例）

○：使用できます。

△：一部使用できます。

×：使用できません。

注※1

非永続キュー属性のローカルキューだけ使用できます。また，キュー間転送によるメッセージの受信はできません。

注※2

永続版リソースアダプタでデッドメッセージキューに登録されるメッセージは，非永続版リソースアダプタでは，即削除されます。

注※3

管理状態だけで実行できるコマンド，キュー間転送で使用するコマンドなどは使用できません。

注※4

トランザクションサポートレベルには，NoTransaction，LocalTransaction，および XATransaction を指定できます。

注※5

認証情報を指定しても無視されます。

注※6

メッセージの送受信抑止だけできます。キュー間転送関連の送受信抑止は使用できません。また，Reliable Messaging の再開始後，キューの抑止状態は引き継がれません。

注※7

リソースへの接続テスト，およびコネクションの障害検知は常に成功します。また，プールされるコネクション中に JDBC リソースは保持されません。

注※8

キュー定義ファイル未使用時のキューの表示名は，キュー作成ファイルで定義します。

2 機能

Reliable Messaging は，JMS の PTP メッセージングモデルに基づく要素でシステムを構成します。要素には，キュー，メッセージおよびアプリケーションがあります。

この章では，PTP メッセージングモデルの概要を示したあとで，各要素の機能について説明します。さらに，DB Connector for Reliable Messaging の機能について説明します。

2.1　メッセージングモデル

メッセージングモデルは，メッセージ通信に関係する構成要素と動作を規定したものです。Reliable Messaging がサポートするメッセージングモデルについて説明します。

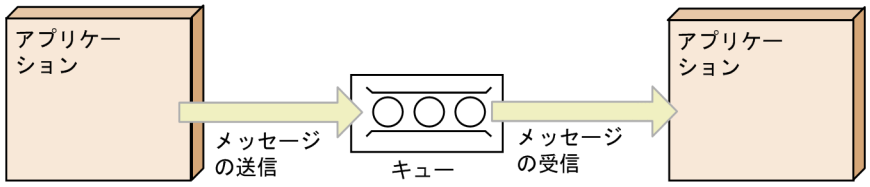
2.1.1　PTP メッセージングモデル

Reliable Messaging が実装する JMS インタフェースは，JMS Version 1.0.2b の PTP（ポイントツーポイント）メッセージングモデルに基づきます。

送信側アプリケーションは特定のキューに対してメッセージを送信します。受信側アプリケーションはキューからメッセージを受信します。

PTP メッセージングモデルの概要を次の図に示します。

図 2–1　PTP メッセージングモデルの概要



2.2　キューの種類

アプリケーションが送受信するメッセージは，Reliable Messaging が管理するキューに登録されます。永続版リソースアダプタの場合と非永続版リソースアダプタの場合とで，キューの作成や削除などの方法が異なります。

- 永続版リソースアダプタの場合

  キューは，hrmmkque コマンドによって DB 上のテーブルに作成されます。キュー作成時に指定する-t オプション指定値によって，目的ごとにキューの種類を作り分けできます。

  hrmmkque のコマンドオプションには，格納メッセージの管理方法や格納できるメッセージ数などの属性を指定します。属性は hrmlsque コマンドで確認でき，hrmchgque コマンドで変更できます。不要なキューは，hrmdelque コマンドによって削除できます。

- 非永続版リソースアダプタの場合

  キュー定義文（定義の先頭に hrmmkque を記述）を指定して，キュー作成ファイルを作成します。キュー作成ファイルに指定したキュー定義文を基に，Reliable Messaging 開始時にキューが作成されます。非永続版リソースアダプタの場合，作成されるキューは非永続キュー属性のローカルキューだけです。また，キューは DB で管理されません。

  キュー定義文のオプションには，格納メッセージの管理方法や格納できるメッセージ数などの属性を指定します。属性は hrmlsque コマンドで確認できます。属性を変更する場合，および不要なキューを削除する場合は，キュー作成ファイルを編集します。

2.2.1　ローカルキュー

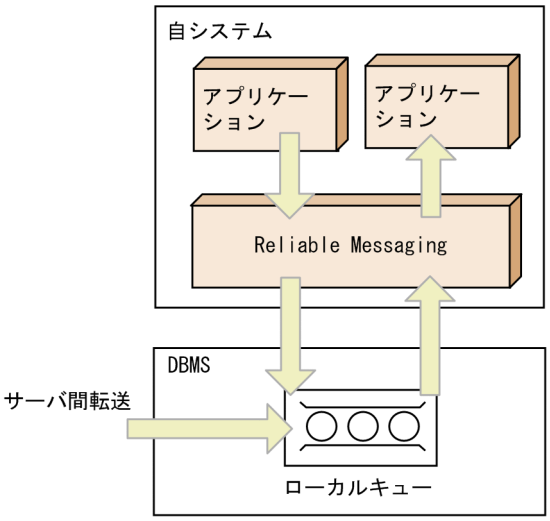
ローカルキューは，自システムのアプリケーション間で送受信するメッセージ，またはサーバ間転送のために受信したメッセージを登録するキューです。自システムのアプリケーションは，ローカルキューを指定してメッセージを送信したり，受信したりします。なお，hrmstopque コマンドを使用して，アプリケーションのメッセージの送受信，およびサーバ間転送のメッセージの送受信を抑止できます。サーバ間転送の詳細は，「2.4　メッセージのキュー間転送」を参照してください。

ローカルキューは，永続版リソースアダプタ，非永続版リソースアダプタのどちらの場合も使用できます。永続版リソースアダプタ，非永続版リソースアダプタのローカルキューの概要をそれぞれ説明します。

(1)　永続版リソースアダプタの場合

永続版リソースアダプタのローカルキューの概要を次の図に示します。

図 2-2　ローカルキューの概要（永続版リソースアダプタの場合）



- 作成方法

  永続版リソースアダプタのローカルキューを作成するには，hrmmkque コマンドの-t オプションに local を指定します。

- 送受信できるメッセージ種別

  永続版リソースアダプタのローカルキューには，アプリケーションは次に示すメッセージインタフェースを使用してメッセージを送受信できます。

  - Message
  - BytesMessage
  - ObjectMessage
  - TextMessage

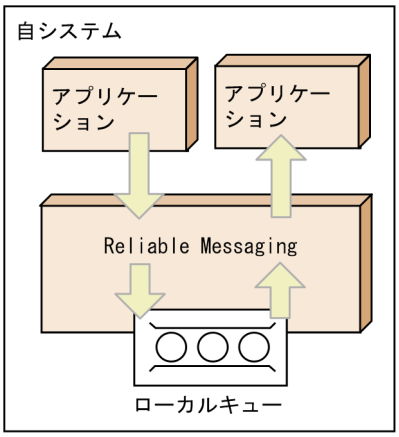
  なお，送受信するメッセージは Reliable Messaging が提供する QueueSession インタフェースのメソッドで生成したオブジェクトです。

(2)　非永続版リソースアダプタの場合

非永続版リソースアダプタの場合，使用できるのは非永続キュー属性のローカルキューだけです。また，サーバ間転送によるメッセージ受信はできません。

非永続版リソースアダプタのローカルキューの概要を次の図に示します。

図 2-3　ローカルキューの概要（非永続版リソースアダプタの場合）



- 作成方法

  キュー定義文（定義の先頭に hrmmkque を記述）を指定して，キュー作成ファイルを作成します。キュー作成ファイルに指定したキュー定義文を基に，Reliable Messaging 開始時に非永続キュー属性のローカルキューが作成されます。

- 送受信できるメッセージ種別

  非永続版リソースアダプタのローカルキューには，アプリケーションは次に示すメッセージインタフェースを使用してメッセージを送受信できます。

  - Message
  - BytesMessage
  - ObjectMessage（ペイロードに BytesContainer は格納できない）
  - TextMessage

  なお，送受信するメッセージは Reliable Messaging が提供する QueueSession インタフェースのメソッドで生成したオブジェクトです。

2.2.2　転送キュー

転送キューは，他システムに存在する Reliable Messaging または他ベンダのメッセージングシステムに向けてメッセージを送信するためのキューです。転送キューを使うには，転送先の情報が必要になります。他ベンダに向けたメッセージ転送は，BytesContainer を使用することで実現できます。

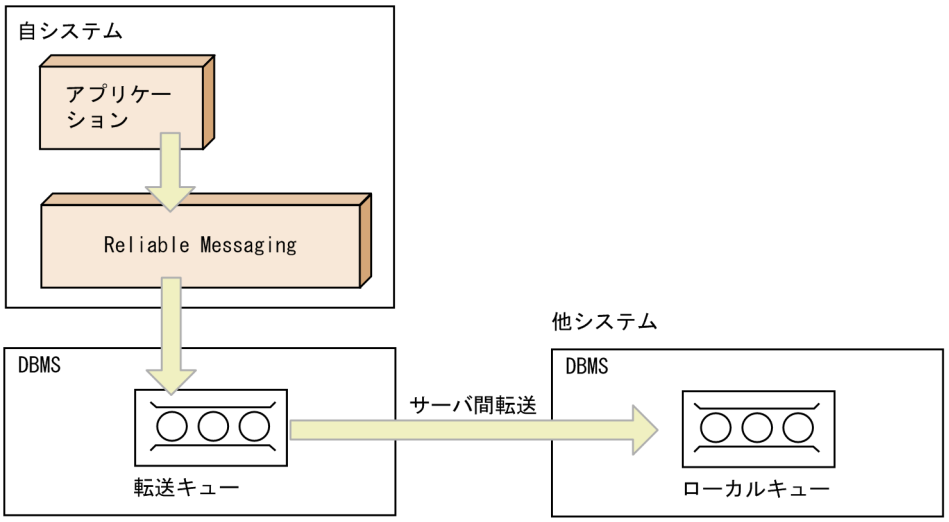
メッセージをアプリケーションから他システムに転送したい場合は，自システムの転送キューにメッセージを登録します。

他システムに向けたメッセージの転送は，転送キューにメッセージが登録された時点で開始します。ただし，hrmstopque コマンドでサーバ間転送のメッセージ送受信を抑止することもできます。また，アプリケーションは転送キューから登録済みのメッセージを取り出すことはできません。

転送キューによるメッセージ転送は，永続版リソースアダプタの場合だけ利用できます。

転送キューの概要を次の図に示します。

図 2-4　転送キューの概要



- 作成方法

  転送キューを作成するには，hrmmkque コマンドの-t オプションに transmit を指定し，事前に hrmmkaddr コマンドで登録しておいたあて先情報を-a オプションに指定します。

- 送信できるメッセージ種別

  転送キューには，アプリケーションは次に示すメッセージインタフェースを使用してメッセージを送信できます。

  - 送信先が Reliable Messaging の場合

    ローカルキューと同じ

  - 送信先が他システムの場合

    ObjectMessage（ボディに BytesContainer を含む）

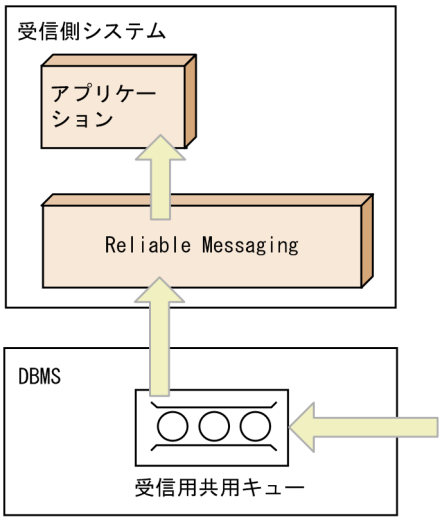
2.2.3　受信用共用キュー

受信用共用キューは，共用キューを使用して複数システム間でのアプリケーション連携をする場合に，メッセージを取り出すためのキューです。受信側アプリケーションは受信用共用キューを指定してメッセージを受信します。また，自システムのアプリケーションから受信用共用キューを指定してメッセージを送信することもできます。さらに，hrmstopque コマンドを使用して，アプリケーションのメッセージの送受信を抑止できます。

受信用共用キューによるメッセージの受信は，永続版リソースアダプタの場合だけ利用できます。

受信用共用キューの概要を次の図に示します。

図 2-5　受信用共用キューの概要



- 作成方法

  受信用共用キューを作成するには，hrmmkque コマンドの-t オプションに shr_receive を指定します。

- 受信できるメッセージ種別

  受信用共用キューから，アプリケーションは次に示すメッセージインタフェースを使用してメッセージを受信できます。

  - BytesMessage

    ただし，メッセージのペイロードだけを取り出しできます。メッセージのペイロードについては，「2.5.1　JMS メッセージの構成」を参照してください。

  なお，受信するメッセージは Reliable Messaging が提供する QueueSession インタフェースのメソッドで生成したオブジェクトです。

注意

- 受信用共用キューは，接続先のデータベースの種別が HiRDB の場合だけ作成できます。接続先のデータベースの種別が Oracle の場合に受信用共用キューを作成するとエラーが発生します。
- 共用キューを作成する場合，同一スキーマ定義内に，送信側システムの管理情報テーブルと受信側システムの管理情報テーブルを定義してください。管理情報テーブルの作成については，「3.4.1(3) Reliable Messaging の管理情報テーブルの作成」を参照してください。

2.2.4　送信用共用キュー

送信用共用キューは，共用キューを使用して複数システム間でのアプリケーション連携をする場合に，メッセージのあて先として指定するキューです。

送信用共用キューは定義だけのキューです。メッセージが格納されるメモリやディスク領域などの実体は，受信側システムの受信用共用キューです。アプリケーションが（送信側システムにとっての）送信用共用

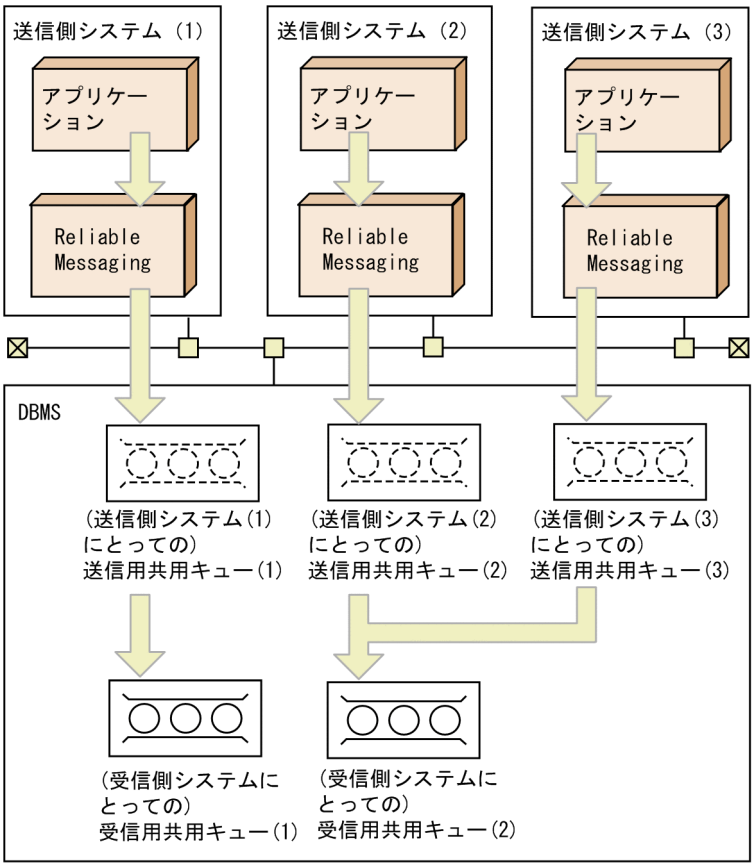
キューにメッセージを登録すると，実体上は（受信側システムにとっての）受信用共用キューに登録されます。

（送信側システムにとっての）送信用共用キューは（受信側システムにとっての）受信用共用キューとｎ：1（ｎ：任意の整数）に対応づけることができます。そのため，受信側システムでは，複数の送信側システムから送信されたメッセージを一つの受信用共用キューに格納するような構成にできます。

送信用共用キューによるメッセージの送信は，永続版リソースアダプタの場合だけ利用できます。

送信用共用キューの概要を次の図に示します。

図 2-6　送信用共用キューの概要



- 作成方法

  送信用共用キューを作成するには，hrmmkque コマンドの-t オプションに shr_send を指定します。-b オプションには，相手システムの受信用共用キューに基づく名前を指定します。キュー属性の多くは，（受信側システムにとっての）受信用共用キューに指定された値に従います。

- 送信できるメッセージ種別

  送信用共用キューには，アプリケーションは次に示すメッセージインタフェースを使用してメッセージを送信できます。

  - BytesMessage

    ただし，メッセージのペイロードだけが保存されます。メッセージのペイロードについては，「2.5.1 JMS メッセージの構成」を参照してください。

なお，送信するメッセージは Reliable Messaging が提供する QueueSession インタフェースのメソッドで生成したオブジェクトです。

注意

送信用共用キューは，接続先のデータベースの種別が HiRDB のときだけ作成できます。接続先のデータベースの種別が Oracle のときに送信用共用キューを作成するとエラーが発生します。

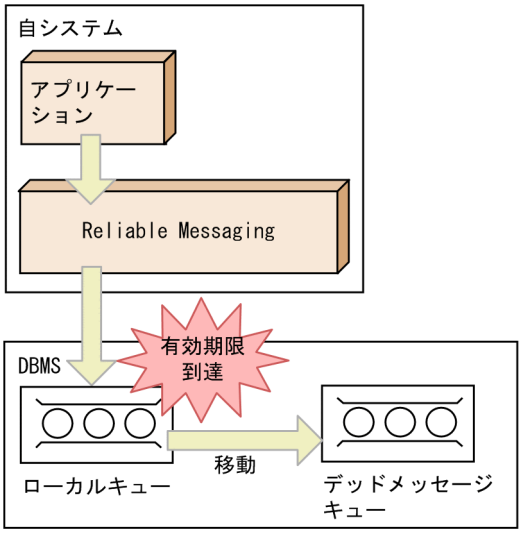
2.2.5 デッドメッセージキュー

デッドメッセージキューは，有効期限に達したメッセージなどが移動される特別なローカルキューです。システム内に一つだけ指定できます。自システムのアプリケーションは，ローカルキューと同様にデッドメッセージキューから該当するメッセージを受信できますが，デッドメッセージキューにメッセージを送信することはできません。また，デッドメッセージキューに移動したメッセージは，移動前のキューに新しいメッセージとして再登録できます。さらに，hrmstopque を使用して，アプリケーションのメッセージの受信を抑止することもできます。

デッドメッセージキューは，永続版リソースアダプタの場合だけ利用できます。永続版リソースアダプタでデッドメッセージキューに移動されるメッセージは，非永続版リソースアダプタでは，即削除されます。

デッドメッセージキューの概要を次の図に示します。

図 2-7 デッドメッセージキューの概要



- 作成方法

  デッドメッセージキューを作成する方法は，ローカルキューと同じです。ただし，RMDeadMessageQueueName プロパティにキュー名を指定する必要があります。

(1) デッドメッセージキューへの登録

サーバ間転送以外の通信では，次に示す場合に，メッセージはデッドメッセージキューに登録されます。サーバ間転送の場合のデッドメッセージキューへの登録については，「2.2.5(2) サーバ間転送でのデッドメッセージキューへの登録」を参照してください。

- メッセージが有効期限に達した場合

  キューに登録されているメッセージが有効期限に達した場合，そのメッセージはデッドメッセージキューに移動されます。ただし，デッドメッセージキューに登録されているメッセージは，有効期限をチェックされないので対象外です。

  また，次に示すときメッセージはデッドメッセージキューに移動されないで削除されます。

  (a)RMDeadMessageQueueName プロパティの指定がないとき

  (b)RMDeadMessageQueueName プロパティに指定されたキューがないとき

  (c)メッセージ数超過や DB 障害などの要因によって，デッドメッセージキューへの登録に失敗したとき

- メッセージの配送回数が最大値に達した場合

  receive()または receiveNoWait()メソッドを発行してメッセージを受信するアプリケーションや Message-driven Bean でメッセージの配信を受けるアプリケーションに，Reliable Messaging がキューから取り出したメッセージを渡すことを配送といいます。

  Reliable Messaging が処理するメッセージの配送回数の最大値は RMMaxDeliveryNum プロパティで指定します。

  キューに登録されているメッセージの配送回数が最大値に達した場合，そのメッセージはデッドメッセージキューに移動されます。ただし，デッドメッセージキューに登録されているメッセージは，配送回数をチェックされないので対象外です。また，次に示すときメッセージはデッドメッセージキューに移動されないで，配送できる状態になります。

  (a)RMDeadMessageQueueName プロパティの指定がないとき

  (b)RMDeadMessageQueueName プロパティに指定されたキューがないとき

  (c)メッセージ数超過や DB 障害などの要因によって，デッドメッセージキューへの登録に失敗したとき

(2) サーバ間転送でのデッドメッセージキューへの登録

サーバ間転送では，次に示す場合に，メッセージはデッドメッセージキューに登録されます。

- メッセージの転送中に通信の有効期限に達した場合

  メッセージの転送中に通信の有効期限に達した場合，そのメッセージはデッドメッセージキューに登録されます。転送中に通信の有効期限に達する場合の例を次に示します。

  (a)転送されてきたメッセージが通信の有効期限に達していたとき

  (b)順序保証をしていて，順序が前のメッセージが転送される前に順序があとのメッセージが転送されて，順序が前のメッセージの到達待ちの間に通信の有効期限に達したとき

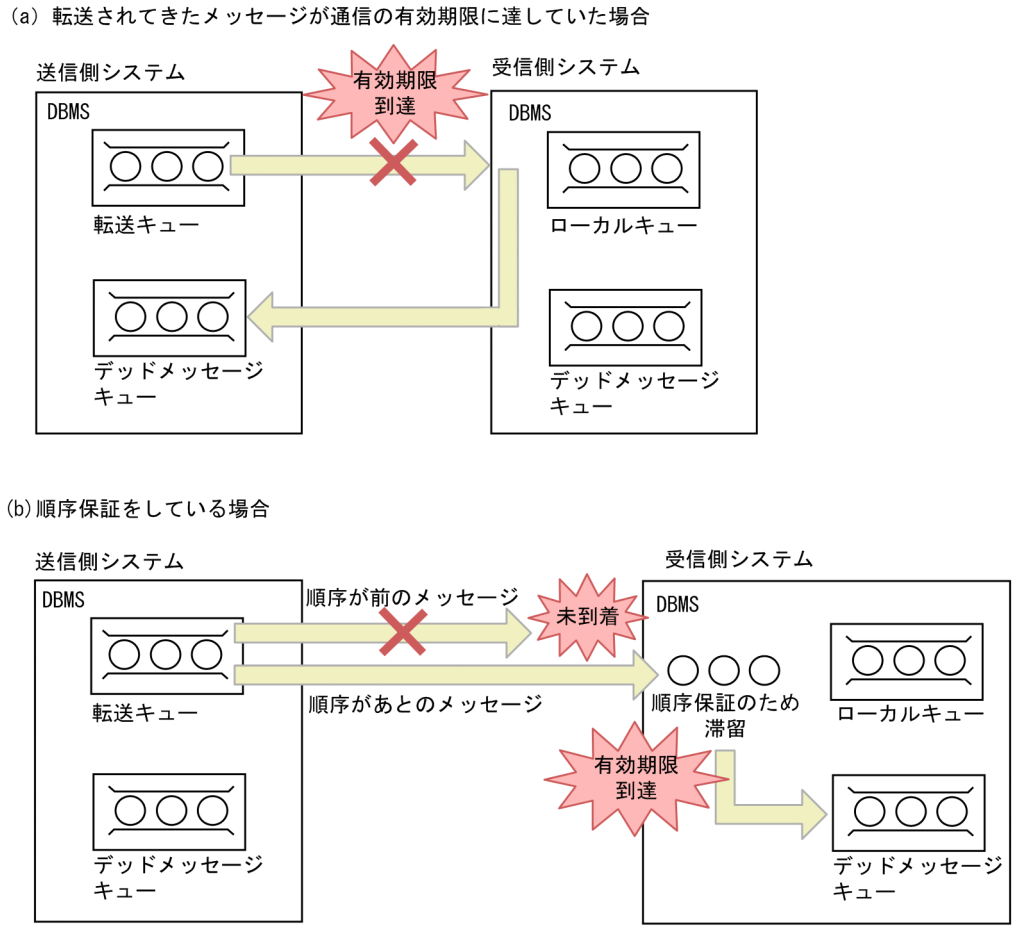
  (a)のときは受信に失敗するため，送信側システムのデッドメッセージキューに登録されます。(b)のとき，順序保証のため滞留している順序があとのメッセージは，ローカルキューには登録されないで，受

信側システムのデッドメッセージキューに登録されます。順序保証については，「2.4.4　キュー間転送の QoS」を参照してください。

メッセージ数の超過や DB 障害などの要因によってデッドメッセージキューへの登録に失敗した場合は，デッドメッセージキューへの登録をリトライし続けます。

通信の有効期限に達した場合の処理を次の図に示します。

図 2-8　メッセージの転送中に通信の有効期限に達した場合の処理



- 電文不正などの障害でメッセージの受信に失敗し，かつ再送もできない場合

  電文不正などの障害でメッセージの受信に失敗し，かつ再送もできない場合は，そのメッセージは送信側システムのデッドメッセージキューに登録されます。

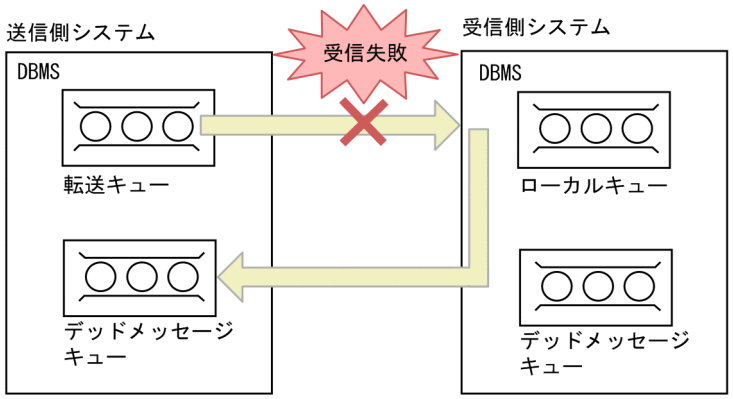
  また，受信に失敗したメッセージが順序保証をしていて，かつすでに受信側システムには順序保証のために滞留しているメッセージがあった場合，滞留しているメッセージはすべて受信側システムのデッドメッセージキューに登録されます。

  メッセージ数の超過や DB 障害などの要因によってデッドメッセージキューへの登録に失敗した場合は，デッドメッセージキューへの登録をリトライし続けます。

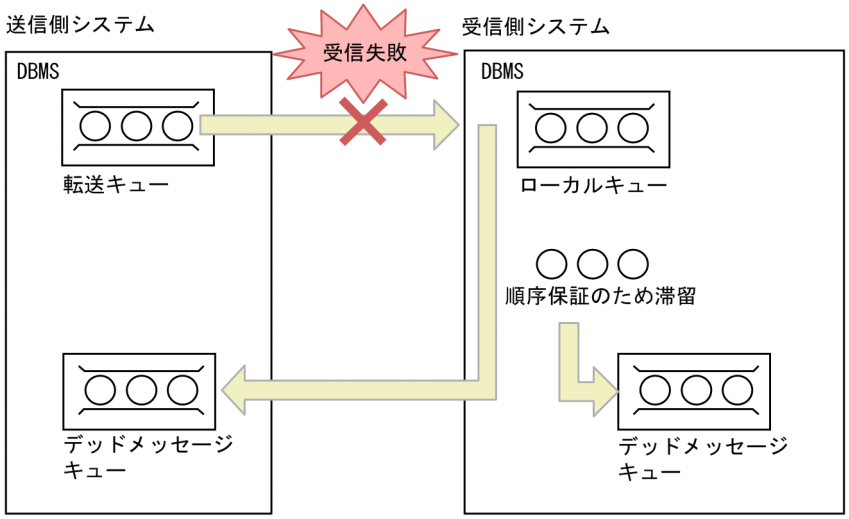
  メッセージの転送ができない場合の処理を，受信側システムに滞留しているメッセージがない場合と滞留しているメッセージがある場合に分けて，次の図に示します。

図 2-9　電文不正などの障害でメッセージの受信に失敗した場合の処理

(a) 受信側システムに滞留しているメッセージがない場合



(b) 受信側システムに滞留しているメッセージがある場合



2.3　メッセージの管理

Reliable Messaging のアプリケーションが送受信するメッセージを管理するときに注意が必要な点について説明します。

2.3.1　キューの永続性

永続版リソースアダプタの場合，キューの作成時には，hrmmkque コマンドの-m オプションにキューの永続性を指定できます。永続性には次に示す属性があり，メッセージの管理が異なります。

- 永続キュー属性（-m オプションに persistent を指定）

  送信側アプリケーションが送信したメッセージは，受信側アプリケーションが受信するまでキューのある DB 上で永続的に管理されます。DB 障害以外によってメッセージが消滅しないため，通常は永続キュー属性を指定することをお勧めします。

  メッセージを送信する際には，DB への書き込みと同時に Reliable Messaging が管理するメモリ上にメッセージがキャッシュされます。同一メッセージの受信の際はメモリ上からメッセージが読み取られるため，取り出し処理のレスポンスが向上します。

- 非永続キュー属性（-m オプションに non_persistent を指定）

  送信側アプリケーションが送信したメッセージは，Reliable Messaging のメモリ上でだけ管理され，DB 上では管理されません。永続キュー属性を指定する場合と比較すると，DB への書き込みが発生しない分だけレスポンスが良くなりますが，障害発生時や Reliable Messaging を再度開始するときにメッセージは消滅します。

キューに指定できる永続性を次の表に示します。

表 2-1　キューに指定できる永続性

項番	キューの種類	永続キュー属性	非永続キュー属性
1	ローカルキュー	○	○
2	転送キュー	○	△※
3	受信用共用キュー	○	×
4	送信用共用キュー	×	×
5	デッドメッセージキュー	○	○

（凡例）

○：指定できます。

△：制限付きで指定できます。

×：指定できません。

注※

順序保証を指定している場合は指定できません。

非永続版リソースアダプタの場合，キューの永続性は非永続キュー属性だけになります。Reliable Messaging の開始時に非永続キュー属性のローカルキューが作成されるため，キューの永続性は指定できません。

2.3.2 メッセージ取り出しモード

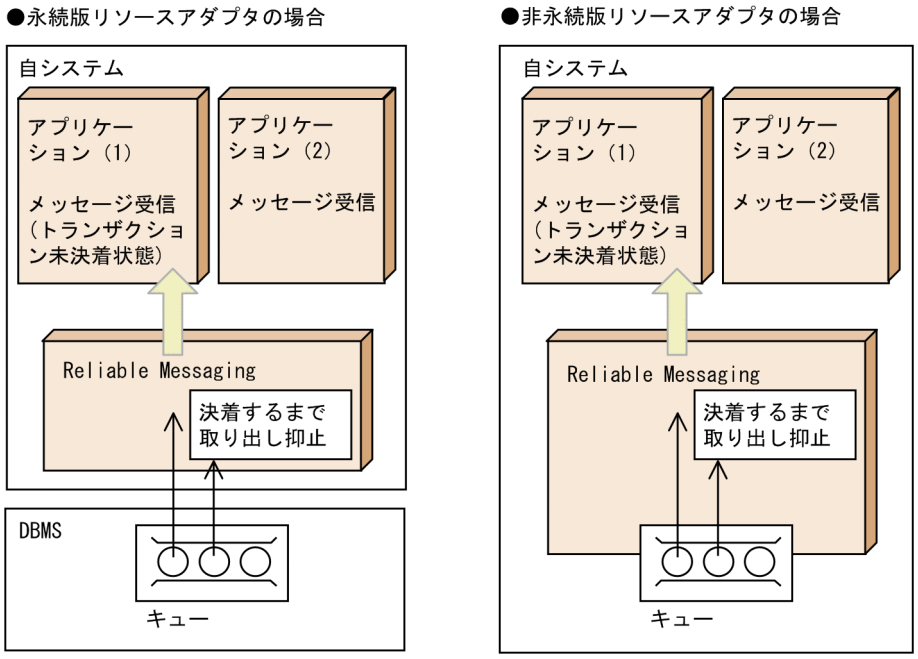
キューの作成時には，次のようにメッセージ取り出しモードを指定できます。

- 永続版リソースアダプタの場合
  hrmmkque コマンドの-d オプションにメッセージ取り出しモードを指定して，コマンドを実行します。
- 非永続版リソースアダプタの場合
  キュー定義文の-d オプションにメッセージ取り出しモードを指定して，キュー作成ファイルを作成します。

メッセージ取り出しモードには次に示す属性があり，アプリケーションがキューからメッセージを受信するときの多重化制御が異なります。

- シリアル取り出し属性（-d オプションに serial を指定）
  アプリケーションがトランザクション内でキューからメッセージを受信したあと，トランザクションが未決着な状態であるとき，異なるアプリケーション（または異なるスレッドのトランザクション）での次メッセージの取り出しが抑止されます。トランザクションが決着してから受信できるようになります。
  つまり，同一トランザクション内でのメッセージの連続的な受信だけができます。
  複数のアプリケーション（または複数のスレッドのトランザクション）でメッセージを受信する場合に，メッセージが送信されたときの順序を保持する必要があるときに指定します。
  シリアル取り出し属性の概要を次の図に示します。

図 2-10　シリアル取り出し属性の概要



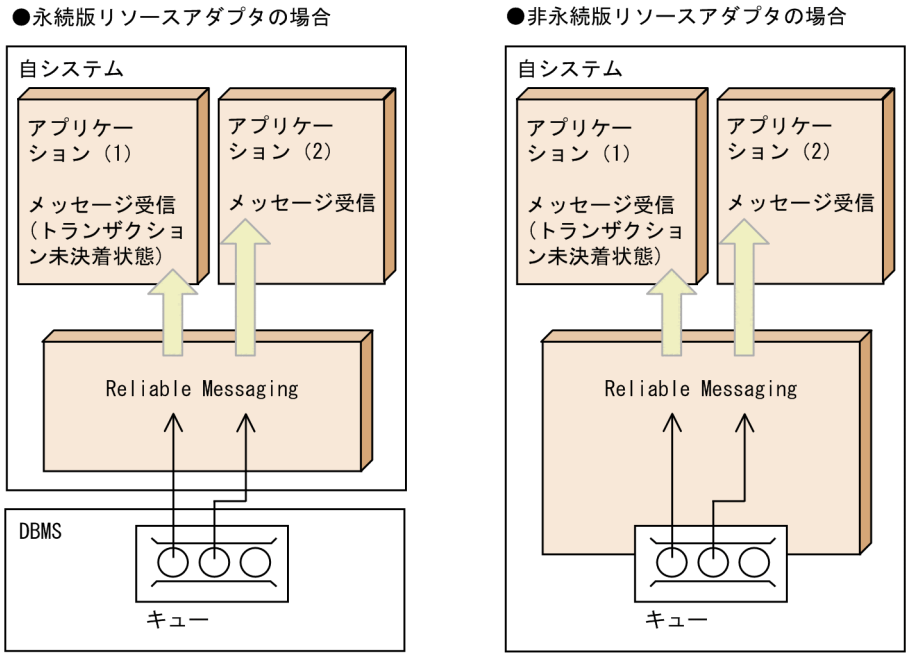
- パラレル取り出し属性（-d オプションに parallel を指定）

  アプリケーションがトランザクション内でキューからメッセージを受信したあと，トランザクションが未決着な状態であるとき，異なるアプリケーション（または異なるスレッドのトランザクション）での次メッセージの受信ができます。

  複数のアプリケーション（または複数のスレッドのトランザクション）でメッセージを受信する場合に，メッセージが送信されたときの順序を保持する必要がないときに指定します。シリアル取り出し属性よりもメッセージ受信時のスループットが優れます。

  パラレル取り出し属性の概要を次の図に示します。

図 2-11　パラレル取り出し属性の概要



- パラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）（-d オプションに parallel_unit_order を指定）

  MDB へのメッセージ配信時に，アプリケーションで設定したメッセージのユニット識別子ごとに順序性を保証します。キューに含まれる特定のメッセージ間では順序性を保証し，それ以外の場合には多重度を確保して MDB を実行できます。

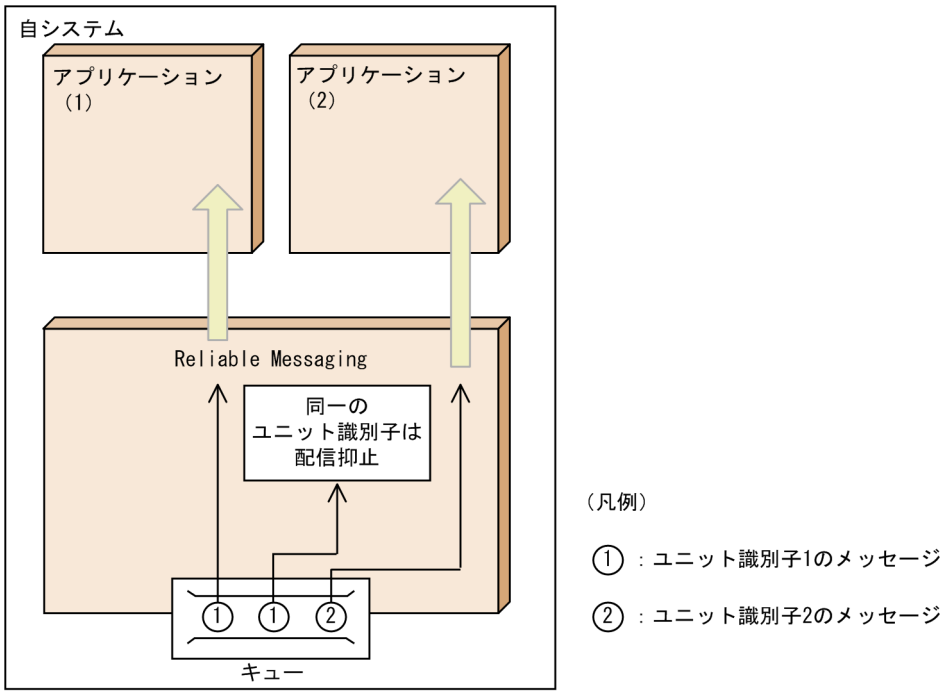
  ユニット識別子での配信順序制御では，MDB へのメッセージ配信時にメッセージのユニット識別子を調べ，配信中のメッセージと同一のユニット識別子の場合はメッセージを配信しないように抑止します。この場合，同一 MDB の他のスレッドに対してもメッセージの配信は抑止されます。配信が抑止されたメッセージは，MDB で行っているメッセージ配信が完了すると配信対象となります。

  配信中のメッセージとは異なるユニット識別子が設定されたメッセージがキューに存在すれば，そのメッセージは配信されます。また，ユニット識別子が設定されていないメッセージも抑止されずに配信されます。

  ユニット識別子の配信順序制御は MDB に対してのみ有効です。アプリケーションからメッセージを送受信する場合は，パラレル取り出し属性と同じ振る舞いとなります。

  ユニット識別子の配信順序制御の概要を次の図に示します。

図 2-12　ユニット識別子の配信順序制御の概要



(1)　ユニット識別子の設定

アプリケーションは送信対象のメッセージに対して，Reliable Messaging 固有のプロパティ JMS_HITACHI_UnitID（String 型）を利用して，メッセージのユニット識別子を設定できます。

ユニット識別子のプロパティ名は大文字小文字を区別します。プロパティ名，型を間違えて設定した場合，アプリケーション指定のプロパティとして扱われます。この場合，ユニット識別子による配信順序制御は行われません。また，空文字や NULL をメッセージ識別子として設定した場合も，メッセージはユニット識別子による配信順序制御の対象とはなりません。

キューに格納されたメッセージのユニット識別子は，メッセージの表示コマンドで参照できます。コマンドの詳細については，「8. コマンドリファレンス」を参照してください。

(2) キューのメッセージ取り出しモードの設定

ユニット識別子による配信順序制御は，キューの作成時にメッセージ取り出しモードとして指定できます。ただし，キューの永続性が非永続キュー属性である必要があります。永続属性のキューに対して，-d オプションとして parallel_unit_order を指定すると例外が発生し，キューの作成に失敗します。

また，キューの属性変更コマンドを利用して，キューの取り出しモードをパラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）に変更することもできます。

(3) 注意事項

- シリアル／パラレルのメッセージ取り出しモードの場合は，ユニット識別子が指定されていても MDB への配信でユニット識別子による配信順序制御は行われません。
- ユニット識別子が設定されない，あるいは，空文字/NULL が指定されている場合，メッセージはパラレル取り出しにより，MDB に配信されます。
- 以下の場合，取り出しモードがユニット識別子による配信順序制御で，メッセージにユニット識別子が付与されていてもユニット識別子による配信順序制御は行われません。
  - メッセージがキュー間転送機能によってキューに格納された場合
  - キューがデッドメッセージキューである場合
  - メッセージがデッドメッセージの再登録によって，キューに格納された場合

  これらの場合，メッセージにユニット識別子が指定されていると，メッセージのプロパティにはユニット識別子は含まれますが，メッセージはパラレル取り出しにより，MDB に配信されます。

  また，メッセージの参照コマンドを実行した場合，デッドメッセージキューおよびデッドメッセージを再登録したメッセージについては，ユニット識別子は表示されません。

  転送機能によってメッセージがキューに格納された場合，メッセージのユニット識別子はメッセージの参照コマンドで参照できますが，ユニット識別子による配信順序制御の対象とはなりません。

2.3.3 メッセージの受け渡し

Reliable Messaging は，アプリケーションとキューとの間でメッセージの受け渡しをします。

(1) メッセージの受け渡し方式

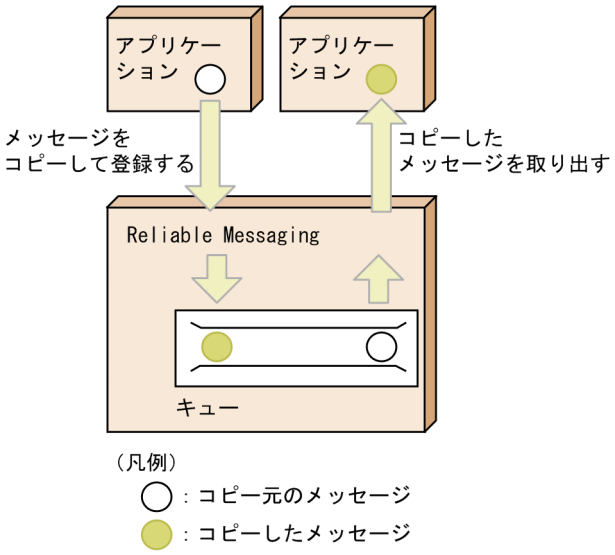
アプリケーションとキューとの間でメッセージを受け渡す方式には，値渡し方式と参照渡し方式の二つ方法があります。それぞれの方法について説明します。

(a) 値渡し方式

値渡し方式は，Reliable Messaging で扱うメッセージとアプリケーションが保持するメッセージを分離する方式です。そのため，アプリケーションが 1 度登録したメッセージを変更したり，取り出したメッセージを再び登録したりできます。Reliable Messaging は，メッセージの登録時および取り出し時にメッセージをコピーします。

値渡し方式を利用したメッセージの受け渡しの概要を次の図に示します。

図 2-13　値渡し方式を利用したメッセージの受け渡しの概要



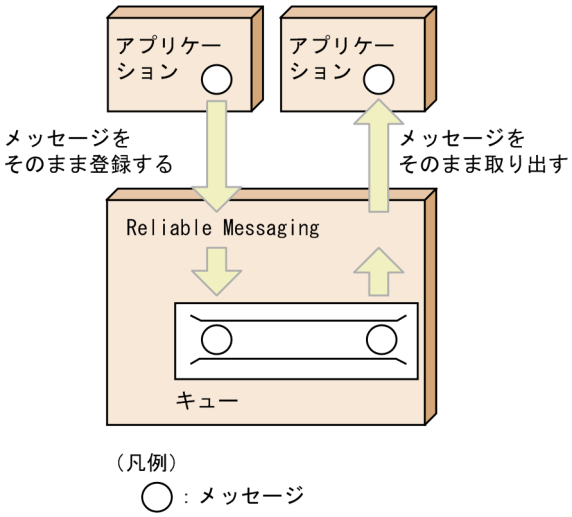
(b) 参照渡し方式

参照渡し方式は，アプリケーションが保持するメッセージをそのまま Reliable Messaging で扱う方式です。アプリケーションが Reliable Messaging と同じメッセージのインスタンスを参照するため，アプリケーションでメッセージを利用するときに，メッセージの再利用ができないなどの制限があります。しかし，値渡し方式よりも高速にメッセージを送受信できます。

参照渡し方式を利用する場合，RMPassByReference プロパティに true を指定します。

参照渡し方式を利用したメッセージの受け渡しの概要を次の図に示します。

図 2-14　参照渡し方式を利用したメッセージの受け渡しの概要



(2)　参照渡し方式を利用した場合の動作

参照渡し方式では，送信側アプリケーションが保持するメッセージと受信側アプリケーション（受信をロールバックした場合，ロールバックする前の受信側アプリケーションを含む）が保持するメッセージが同じインスタンスとなります。そのため，最後の受信側アプリケーションがメッセージを受信すると，送信側アプリケーションと以前の受信側アプリケーションが保持するメッセージは，最後の受信側アプリケーションのメッセージと同じ状態となります。

参照渡し方式を利用した場合の動作を次の図に示します。

図 2-15　参照渡し方式を利用した場合の動作

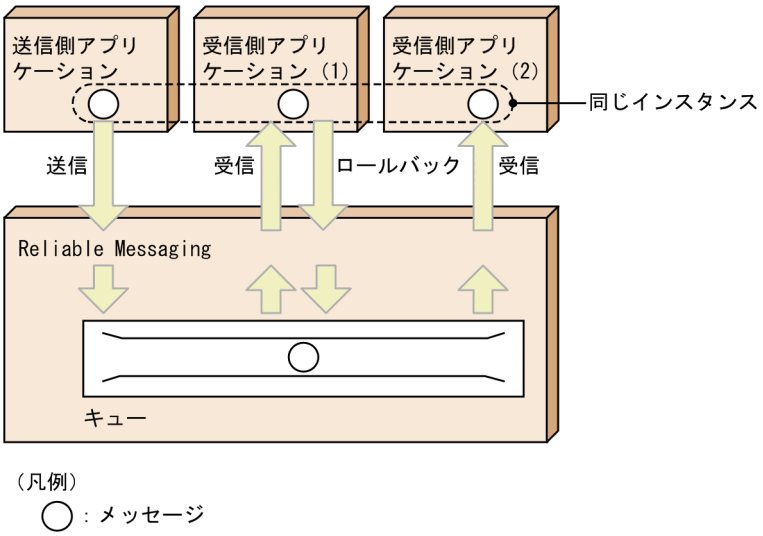


図 2-14 に示すように，送信側アプリケーションと受信側アプリケーションでは次の状態となります。

- 送信側アプリケーションと以前の受信側アプリケーション（受信側アプリケーション（1））が保持し続けているメッセージから取得できる情報は，最後の受信側アプリケーション（受信側アプリケーション（2））が保持するメッセージと等しくなる。

- 送信側アプリケーションと以前の受信側アプリケーション（受信側アプリケーション（1））がMessage.acknowledge()メソッドを呼び出すと，最後の受信アプリケーション（受信側アプリケーション（2））が保持するメッセージの受信を承認したことになる。

(3) 参照渡し方式を利用した場合のメッセージの再利用

参照渡し方式を利用した場合，Reliable Messaging が持つメッセージのインスタンスをアプリケーションも保持していることがあります。そのため，アプリケーションによるメッセージの再利用は行えません。メッセージの再利用とは，Reliable Messaging のキューに登録されたメッセージまたは登録を試みたメッセージを，再び登録することやそのメッセージに設定した情報を変更することです。アプリケーションによってメッセージを再利用した場合はエラーが発生します。再利用できないメッセージを次に示します。

- QueueSender インタフェースを使用して登録したメッセージ（登録失敗およびロールバックしたメッセージを含む）
- QueueReceiver インタフェースを使用して取り出したメッセージ
- MDB によって配信されたメッセージ
- QueueBrowser インタフェースを使用して参照したメッセージ

2.3.4 メッセージの削除

Reliable Messaging は，アプリケーションによるメッセージの取り出しなどで不要になったキュー内のメッセージを削除します。

(1) メッセージの削除方法

メッセージを削除する方法には，遅延削除と即時削除の二つ方法があります。それぞれの方法について説明します。

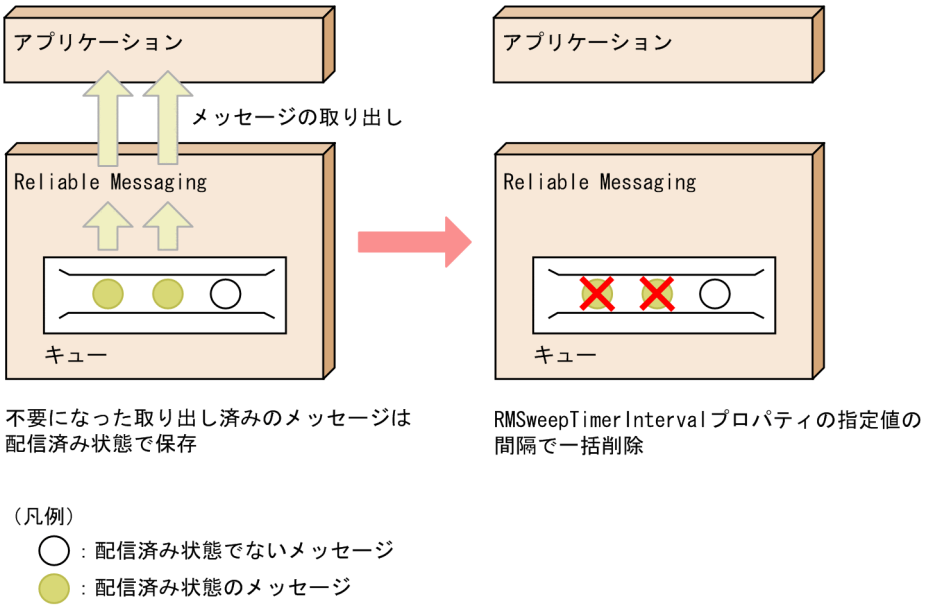
(a) 遅延削除

遅延削除は，メッセージが不要になった時点で削除せず配信済みの状態にしておき，RMSweepTimerInterval プロパティ指定値の間隔で一括に削除する方法です。遅延削除はメッセージの取り出しの延長でメッセージを削除しないため，即時削除よりもスループットが優れる場合があります。

なお，非永続リソースアダプタの場合，遅延削除は利用できません。

遅延削除を利用したメッセージの削除の概要を次の図に示します。

図 2-16　遅延削除を利用したメッセージの削除の概要



永続キューで遅延削除を使用する場合の注意事項

永続キューで遅延削除を使用する場合，以下に注意する必要があります。

- 遅延削除処理中※1 に DB 側に高負荷がかかっていないか
- RMSweepTimerInterval の値
- HiRDB クライアント環境変数「PDCWAITTIME」の値（HiRDB を使用する場合）

これは運用上，以下の問題が発生するのを防止するためです。

【問題】

キュー内の遅延削除対象のメッセージが削除されず，滞留し続ける。滞留が発生した場合，永続キューへのメッセージ送受信処理のスループットが低下する可能性がある※2。

【原因】

- **遅延削除対象のメッセージが滞留し続ける問題について**

  多量のメッセージを一括削除する場合，削除実行時に DB 側で多量の行削除が行われます。

  このとき DB 側に一時的に高負荷がかかり，遅延削除の DB 更新完了に時間がかかる場合があります。DB 更新完了に時間が掛かることにより，タイムアウトし，ロールバックが行われると，メッセージが削除されずキュー内に遅延削除対象メッセージが滞留することとなります※3。

  更に次回以降の遅延削除処理でも，前回の遅延削除で削除できなかったメッセージと新しく削除対象となるメッセージが削除対象となるため，再度タイムアウトが発生し，メッセージの削除ができない可能性が高く，その結果削除対象のメッセージが滞留し続けることとなります。

- **永続キューへのメッセージ送受信処理のスループットが低下する問題について**

  遅延削除対象のメッセージが滞留し続ける場合，遅延削除処理中には DB 側に高負荷がかかります。そのため，遅延削除処理中に永続キューに対するメッセージ送受信を行うと，送受信の DB 更新完了に時間がかかる可能性があり，送受信処理のスループットが低下する可能性があります。

上記の問題を発生させないためには，DB サーバへ高負荷が発生しないように，RMSweepTimerInterval をチューニングし，多量のメッセージが一括削除されないようにする必要があります※4。

また，HiRDB を使用する場合，遅延削除処理でタイムアウトが発生しないように，HiRDB クライアント環境変数「PDCWAITTIME」の値をチューニングする必要があります※5。

ただし RMSweepTimerInterval の値を短くし，プロパティに指定できる最小値を指定しても問題が発生する場合は，即時削除の使用を推奨します。

注※1

遅延削除処理実行中かどうかは DB の SQL トレースより確認できます。

以下に，Reliable Messaging が遅延削除で発行する削除 SQL を示します。

```
DELETE FROM <DBへの接続ユーザ名>.<RMSystemNameプロパティ指定値>_MSG_<キュー名称>
                                        WHERE GROUP_NAME='' AND DELETE_FLAG=?
```

注※2

キュー内の削除対象のメッセージ数，および未配信のメッセージ数が多いほど，またメッセージのサイズが大きいほど発生するリスクは高まります。

注※3

遅延削除処理がロールバックしたかどうかは，キュー内の遅延削除対象のメッセージ数が減少しているかで判断できます。「hrmlsque コマンド」のオプションに-q オプションを指定し実行した後，「total message count」の値から「message count」の値を引いた値が遅延削除対象のメッセージ数となります。遅延削除対象のメッセージ数が減少しない場合は，遅延削除処理がロールバックしている可能性があります。

注※4

チューニングは本番稼動環境と同等の環境で行う必要があります。

更に遅延削除処理にかかる時間が最大となるように，以下の環境でチューニングを行う必要があります。

- システムへの負荷（メッセージの送受信処理など）が最大となるような環境
- キュー内の遅延削除対象となるメッセージ数，配信済み状態ではないメッセージ数，メッセージのサイズが最大となる環境

注※5

PDCWAITTIME のチューニングについては，「3.4.1(2)(b)　HiRDB の環境変数グループの登録」を参照してください。

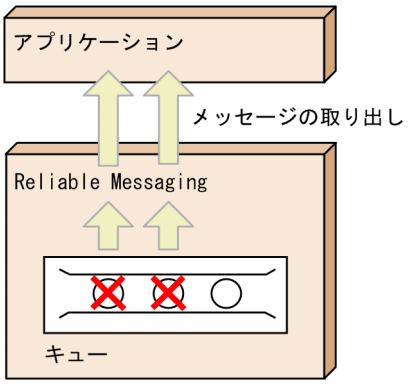
(b)　即時削除

即時削除は，メッセージが不要になった時点で削除する方法です。そのため，遅延削除よりも一時的に使用するメモリの量を削減できます。即時削除は，アプリケーションによるメッセージの取り出しでのコミット時や，hrmdelmsg コマンド実行時のタイミングで実施されます。

ローカルキューまたはデッドメッセージキュー内のメッセージに即時削除を利用する場合，RMDeleteMessageImmediately プロパティに true を指定します。

即時削除を利用したメッセージの削除の概要を次の図に示します。

図 2-17　即時削除を利用したメッセージの削除の概要



不要になった取り出し済みのメッセージを削除して
メモリを解放

（凡例）
◯：配信済み状態でないメッセージ

(2)　キューの種類と利用できる削除方法

キューの種類によって利用できる削除方法が異なります。

キューの種類と利用できる削除方法を次の表に示します。

表 2-2　キューの種類と利用できる削除方法

項番	キューの種類	遅延削除	即時削除
1	ローカルキュー	○	○※1※2
2	転送キュー	○	×
3	受信用共用キュー	×	○
4	送信用共用キュー	－	－
5	デッドメッセージキュー	○	○※1

（凡例）
○：利用できます。
×：利用できません。
－：対象外です。

注※1
RMDeleteMessageImmediately プロパティに true を指定した場合だけ即時削除を行います。

注※2
キュー間転送では，メッセージの順序を保証するため，アプリケーションによって受信されたメッセージでも削除できないことがあります。そのため，キュー間転送で受信したメッセージは即時削除を行いません。

2.3.5　メッセージの有効期間

キューの作成時には，次のようにメッセージの有効期間を指定できます。

- 永続版リソースアダプタの場合
  hrmmkque コマンドの-e オプションにメッセージの有効期間を指定して，コマンドを実行します。
- 非永続版リソースアダプタの場合
  キュー定義文の-e オプションにメッセージの有効期間を指定して，キュー作成ファイルを作成します。

メッセージの有効期間は，キューに登録されたメッセージがシステム内で有効な時間を秒単位で表します。

Reliable Messaging は RMSweepTimerInterval プロパティ指定値の間隔（単位：秒）でメッセージの有効期間を確認し，有効期間に達しているメッセージの削除処理を実行します。有効期間に達している場合，デッドメッセージキューがあるときには，メッセージはデッドメッセージキューに移動されます。デッドメッセージキューがないときには，メッセージは破棄されます。

メッセージの有効期間に 0 が指定されたローカルキューのメッセージの有効期間は無限です。

2.4　メッセージのキュー間転送

メッセージのキュー間転送について説明します。

2.4.1　キュー間転送の概要

転送キューは，メッセージのキュー間転送をするための送信側のキューです。転送キューを使用してキュー間転送をすることで，ネットワークを介した複数システム間でのアプリケーション連携ができるようになります。

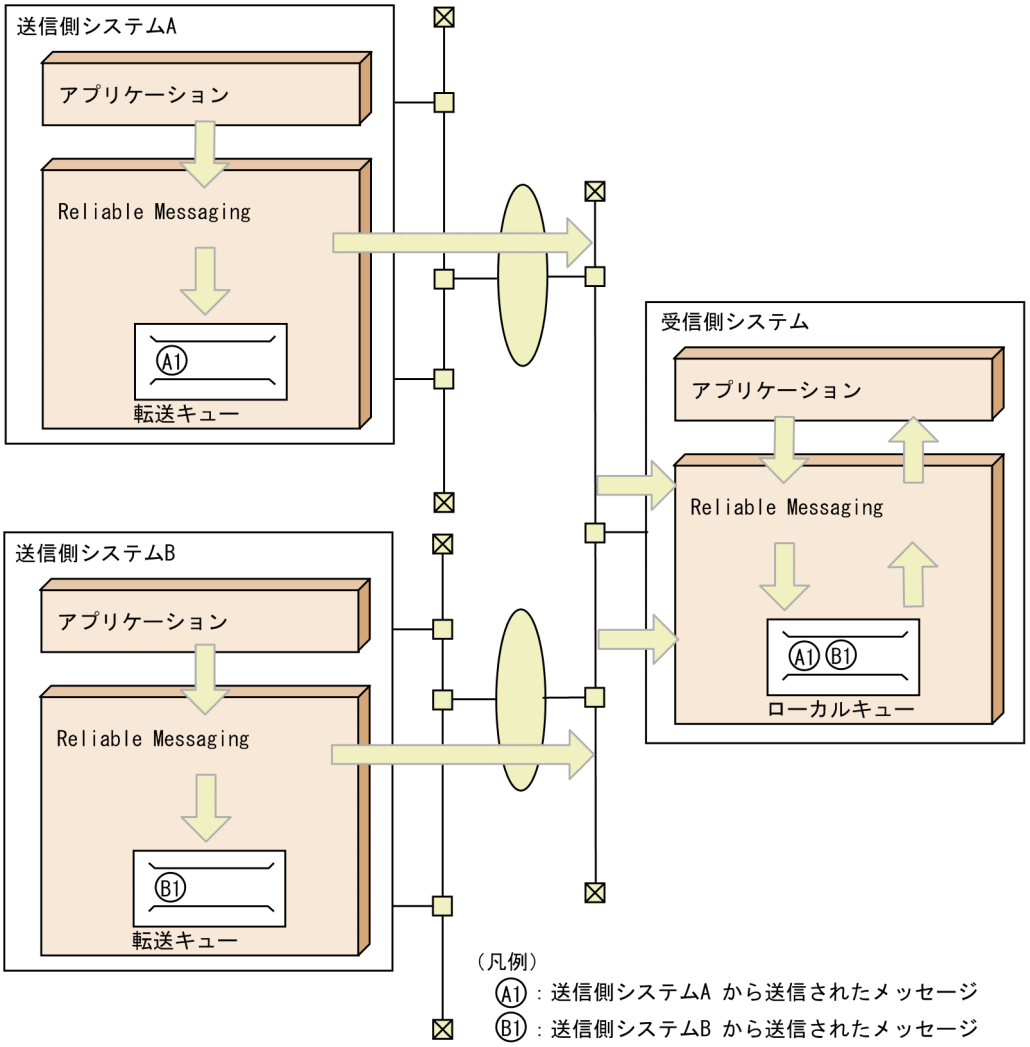
転送キューには，受信側システムのアドレスとローカルキュー名を指定します。転送キューに登録したメッセージは，指定された受信側システムのローカルキューに向けて転送されます。

送信側システムの転送キューは，受信側システムのローカルキューと，n：1（n：任意の整数）に対応づけることができます。このため，受信側では，複数の送信側システムから受信するメッセージを一つのローカルキューに格納する形でシステムを構成できます。

転送キューからメッセージを取り出すことはできません。転送キューに登録されたメッセージは，転送が成功し，転送先のローカルキューにメッセージが登録されたことが確認されたあとに削除されます。

転送キューによるキュー間転送の概要を次の図に示します。

図 2-18　転送キューによるキュー間転送の概要



2.4.2　キュー間転送のあて先指定

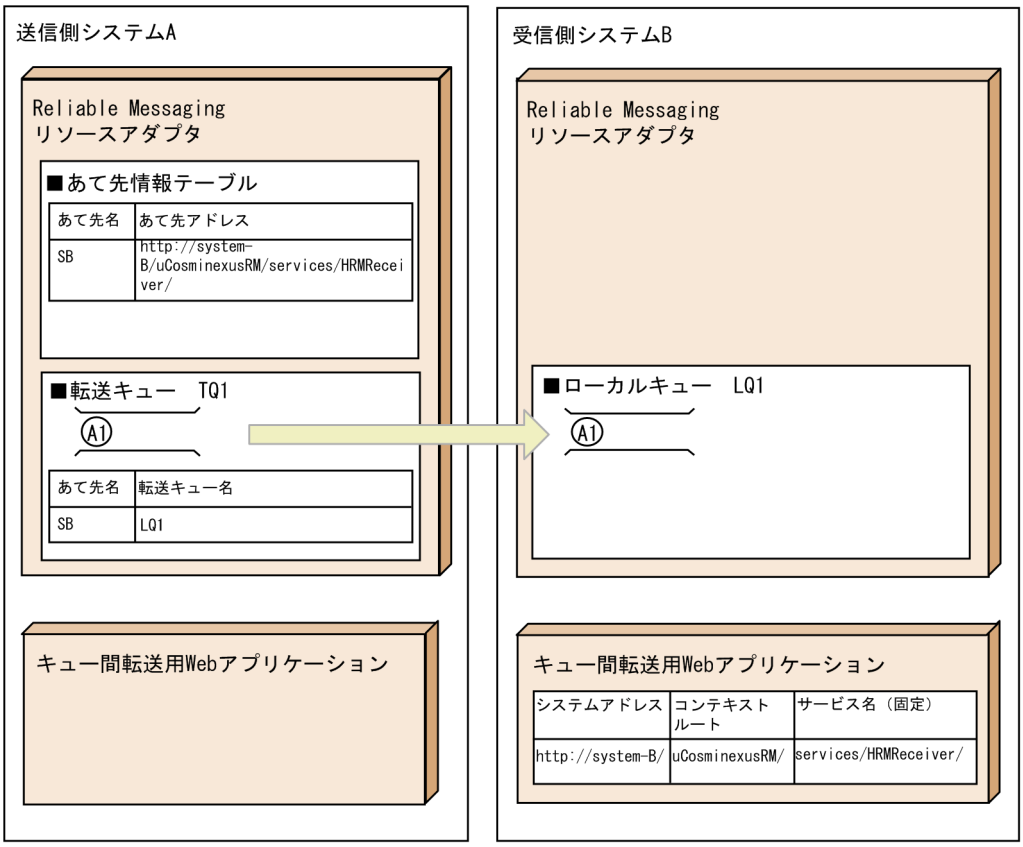
Reliable Messaging は，キュー間転送用の Web アプリケーションを使用してメッセージを受信します。

メッセージを転送するときは，転送先のアドレスとして，受信側システムのキュー間転送用 Web アプリケーションのアドレスを指定します。このアドレス情報は，hrmmkaddr コマンドを使ってあて先情報テーブルに登録します。BASIC 認証使用時には，あて先情報テーブルに BASIC 認証用のユーザ名とパスワードも登録できます。

転送先のローカルキューの識別には，受信側システムのアドレスを表すあて先名と，転送先のキュー名が使用されます。この情報は，転送キューに登録します。

送信側システム A の転送キュー TQ1 から，受信側システム B のローカルキュー LQ1 にメッセージを転送する場合のあて先指定の仕組みを次の図に示します。

図 2-19　転送キューによるキュー間転送のあて先指定の仕組み



（凡例）
Ⓐ1：送信側システムAから受信側システムBに送信されたメッセージ

図に示すように，あて先情報テーブルには，転送先 Reliable Messaging のキュー間転送用 Web アプリケーションのアドレス「http://system-B/uCosminexusRM/services/HRMReceiver/」を登録します。登録したアドレスは，あて先名「SB」で識別されます。

転送キューには，あて先名「SB」と転送先のローカルキュー名「LQ1」を登録します。

この場合の実際の送信アドレスは，「http://system-B/uCosminexusRM/services/HRMReceiver/LQ1」です。

(1)　あて先アドレスの形式

送信側システムに登録するあて先アドレスは，次に示すどちらかの形式で指定してください。

```
http://<ホスト名>:<ポート番号>/<コンテキストルート>/services/HRMReceiver/
https://<ホスト名>:<ポート番号>/<コンテキストルート>/services/HRMReceiver/
```

ホスト名

　受信側システムのホスト名，または IP アドレス

ポート番号

　受信側システムの HTTP(S)通信用のポート番号

コンテキストルート

受信側 Reliable Messaging のキュー間転送用 Web アプリケーションのコンテキストルート

サービス名：services/HRMReceiver（固定）

受信側 Reliable Messaging のキュー間転送用 Web アプリケーションのサービス名

(2) 注意事項

- BASIC 認証を使用する場合は，hrmmkaddr コマンドによるあて先アドレス登録時に，BASIC 認証用のユーザ名とパスワードを設定してください。
- リバースプロキシやリダイレクタ（互換機能）を使用する場合は，キュー間転送用 Web アプリケーションのコンテキストルートに適切にリダイレクトされるように設定してください。

2.4.3 キュー間転送の通信モデル

Reliable Messaging がサポートする通信モデルには，次に示す二つの層があります。

- アプリケーションが扱うキューの層
- キューのメッセージを取り出して通信を行う通信層

ここでは通信層の詳細を説明します。

(1) 通信層のメッセージとグループ

通信層では「グループ」をメッセージ管理の単位とします。グループは，グローバルに一意の識別子を持ち，複数のメッセージが属する集合です。また，順序保証などの QoS（通信品質）を保証する範囲でもあります。

各メッセージは，グループの識別子とグループ内の通し番号によって一意に識別されます。送信側のグループは，転送キューの作成時に Reliable Messaging が生成します。受信側のグループは，グループ内で最初のメッセージの受信時に Reliable Messaging が生成します。

(2) メッセージとグループの有効期間

通信層のメッセージとグループに対して有効期間を設定できます。どちらも有効期間を超過した場合の通信は失敗します。

- メッセージの有効期間

  メッセージの有効期間は，メッセージの送信時刻から，転送キューに指定した通信層のメッセージ有効期限に達するまでの間です。
- グループの有効期間

通信で使用するグループは転送キューの作成時に生成されます。有効期間は，グループの生成時刻から，転送キューに指定したグループの有効期限に達するまでの間です。グループの有効期間に達すると，自動的に次のグループが作成されて次のグループに切り替わります。

2.4.4　キュー間転送の QoS

Reliable Messaging で保証する QoS には，配送保証と順序保証の 2 種類があります。

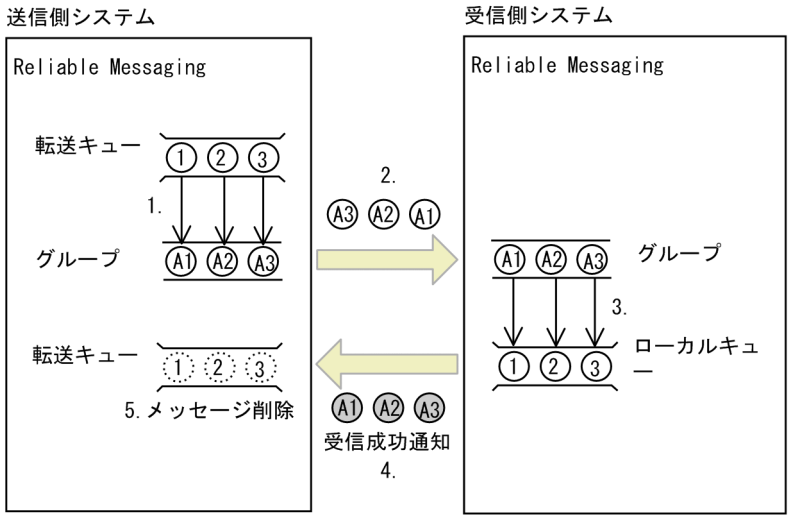
(1)　配送保証

Reliable Messaging は，受信側システムから受信の成功が通知されるまでメッセージを再送し続けることで，高信頼なキュー間転送を実現します。また，再送によって同じメッセージを複数回受信した場合も，識別子を用いてメッセージの重複を判断し，重複してキューに登録されることを防止します。

次の図は，送信側システムが A1，A2，A3 のメッセージを送信し，受信側システムでキューに正常に登録できたため，A1，A2，A3 の受信成功が通知された場合を示しています。1.～5.は，処理の順序を示します。

受信成功が通知されると，送信側システムのキューのメッセージは削除されます。受信成功が一定の期間通知されない場合は，A1，A2，A3 を再送します。

図 2-20　配送保証の仕組み



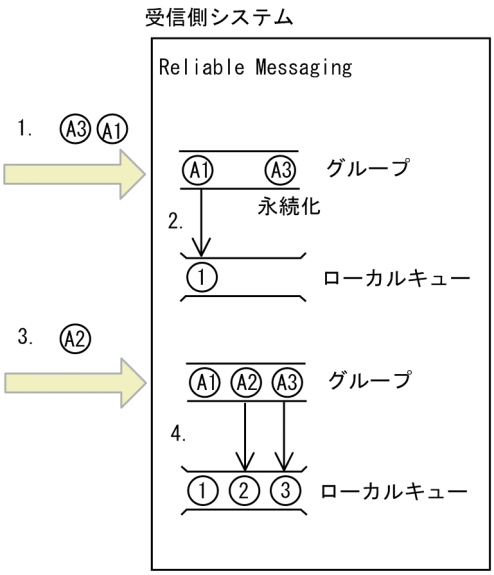
(2)　順序保証

順序保証が指定された転送キューから送信されたメッセージは，送信された順序で受信側システムのキューに登録します。順序どおりでないメッセージを受信した場合はメッセージを永続化して保持しておき，順序が回復した時点でキューに登録します。

次の図は，A1→A2→A3 の順序で順序保証が指定されている場合の処理の仕組みを示しています。1.～4.は，処理の順序を示します。

A1，A3 の順序でメッセージを受信しましたが，A3 は順序不正のためローカルキューに登録されません。その後，A2 を受信した時点で，A2，A3 がローカルキューに登録されます。

図 2-21　順序保証の仕組み



順序保証を行う場合の滞留メッセージの監視と，メッセージ障害時の動作について説明します。

(a)　滞留メッセージの監視

例えば図 2-21 で，障害などが原因で A2 のメッセージが受信されない場合，A3 以降のメッセージはアプリケーションに配送されないメッセージとして滞留します。この場合，A3 以降のメッセージは，通信の有効期限に達してデッドメッセージキューに移動されるまで，滞留したままとなります。

この対処として，Reliable Messaging は，定期的に滞留メッセージを監視します。滞留メッセージの監視時間は，RMTRPendingNotifyInterval プロパティに指定します。メッセージの監視中に，このプロパティに指定した時間より長く滞留しているメッセージが存在する場合は，ログを出力して通知します。

滞留したメッセージを待ち合わせる必要がない場合は，hrmskipmsg コマンドによって A2 のメッセージをスキップし，A3 以降のメッセージをアプリケーションに配送できます。また，スキップしたメッセージをスキップしたあとに受信すると，そのメッセージの受信は失敗します。

(b)　メッセージ障害時の動作

受信側システムから受信失敗が通知されると，送信側システムでは新しいグループで，以降のメッセージを送信します。そのため，受信側システムでは，新しいグループで送信されたメッセージがキューに登録されます。また，受信側システムで，障害が発生したグループ内のキューに登録しないで保持していたメッセージは，デッドメッセージキューに登録されます。

2.4.5　メッセージの再送

キュー間転送でメッセージを再送するときの間隔は，3 種類のプロパティで指定します。Reliable Messaging は，2 種類のプロパティに指定した再送間隔と，別のプロパティに指定したそれらを切り替えるタイミング（再送回数）を使用してメッセージの再送間隔を調整します。

メッセージの再送の間隔を調整するためのプロパティを次の表に示します。

表 2-3　メッセージの再送の間隔を指定するプロパティ

項番	プロパティ	意味
1	RMTRResendInterval1	再送間隔 1（軽微なネットワーク障害などの短期的な障害を想定）
2	RMTRResendInterval2	再送間隔 2（受信側システムのサーバダウンなどの長期的な障害を想定）
3	RMTRResendInterval1Num	再送間隔を切り替えるタイミング（RMTRResendInterval1 プロパティの指定値で再送する回数）

Reliable Messaging がメッセージを再送する処理の流れは次のとおりです。

1. RMTRResendInterval1 プロパティに指定した間隔でメッセージの再送を行います。
2. RMTRResendInterval1 プロパティの指定値での再送を，RMTRResendInterval1Num プロパティに指定した回数分繰り返します。初回の送信は，RMTRResendInterval1Num プロパティで指定した再送回数に，1 回分として含まれます。
3. RMTRResendInterval2 プロパティに指定した間隔での再送に切り替えて，メッセージの再送を続けます。

2.4.6　キュー間転送の互換通信

送信側システムと受信側システムの Reliable Messaging のバージョンの違いによって，次のようにメッセージを送受信します。

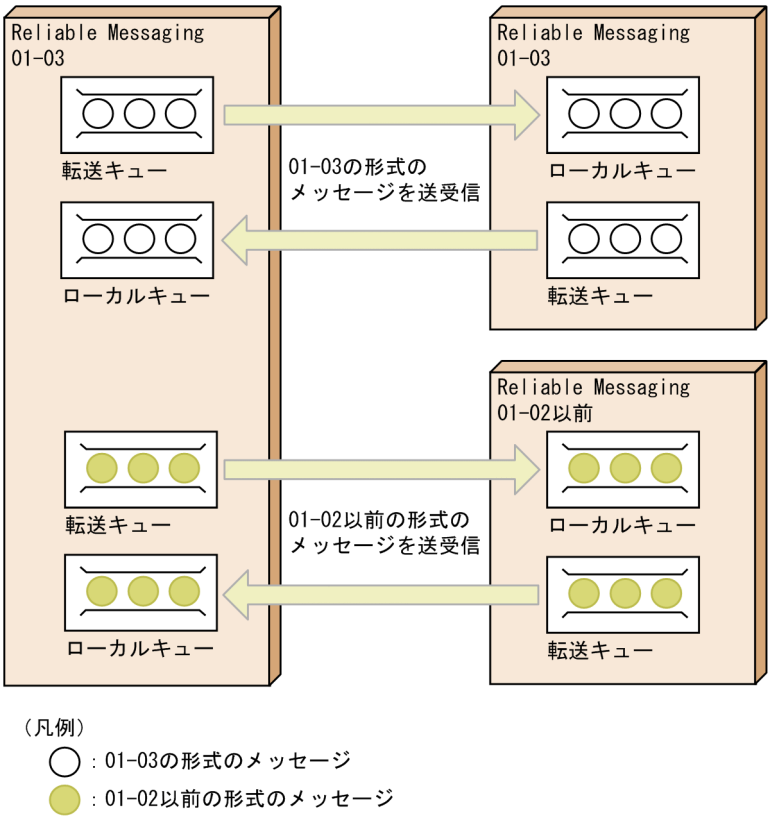
- メッセージを送信する場合

  受信側システムの Reliable Messaging のバージョンに合わせた形式のメッセージを送信します。ユーザは受信側システムの Reliable Messaging のバージョンに合わせて，送信するメッセージの形式を選択する必要があります。送信するメッセージの形式の選択については，「2.4.6(1)　送信するメッセージの形式の選択」を参照してください。
- メッセージを受信する場合

  送信側システムの Reliable Messaging のバージョンの形式でメッセージを受信します。

Reliable Messaging のバージョンの違いによるメッセージの送受信の概要を次の図に示します。

図 2-22　Reliable Messaging のバージョンの違いによるメッセージの送受信の概要



(1)　送信するメッセージの形式の選択

送信側システムの Reliable Messaging では，受信側システムの Reliable Messaging のバージョンのメッセージ形式に合わせて，次のように転送モードを選択します。

- 受信側システムの Reliable Messaging のバージョンが 01-03 以降の場合

  hrmmkque コマンドで転送キューを作成する，または hrmchgque コマンドで転送キューの属性を変更するときに，-i オプションに normal（通常モード）を指定します。
- 受信側システムの Reliable Messaging のバージョンが 01-02 以前の場合

  hrmmkque コマンドで転送キューを作成する，または hrmchgque コマンドで転送キューの属性を変更するときに，-i オプションに compatible（互換モード）を指定します。

  なお，01-02 以前のバージョンの Reliable Messaging に対して通常モードで ObjectMessage を送信した場合，ペイロードに null が指定された ObjectMessage として受信されます。

(2)　注意事項

(a)　01-02 以前のバージョンの Reliable Messaging と通信する場合

01-03 以降のバージョンの Reliable Messaging が 01-02 以前のバージョンの Reliable Messaging とメッセージ送受信する場合の注意事項を次に示します。

- 送信側システムの Reliable Messaging のバージョンが 01-03 以降の場合

互換モードの転送キューを使用してユーザ定義クラスを設定した ObjectMessage を送信する場合は，コンテナ拡張ライブラリとしてユーザ定義クラスを指定する必要があります。

- 受信側システムの Reliable Messaging のバージョンが 01-03 以降の場合

  01-02 以前のバージョンの Reliable Messaging からユーザ定義クラスを設定した ObjectMessage を受信する場合は，コンテナ拡張ライブラリとしてユーザ定義クラスを指定する必要があります。

(b)　01-03 以降のバージョンの Reliable Messaging と通信する場合

互換モードの転送キューを使用してユーザ定義クラスを設定した ObjectMessage を送信する場合は，コンテナ拡張ライブラリとしてユーザ定義クラスを指定する必要があります。

(c)　送受信できる Reliable Messaging のバージョン

送受信できる Reliable Messaging のバージョンを転送モードごとに次の表に示します。

表 2-4　送受信できる Reliable Messaging のバージョン

送信側システムの Reliable Messaging のバージョン	転送モード	受信側システムの Reliable Messaging のバージョン	
		01-02 以前	01-03 以降
01-02 以前	－	△	△
01-03 以降	互換モード	△	△
	通常モード	×	○

（凡例）

○：送受信できます。

△：送受信できます。ただし，ユーザ定義クラスを設定した ObjectMessage を送信する場合は，送信側システムおよび受信側システムでコンテナ拡張ライブラリとしてユーザ定義クラスを指定する必要があります。

×：送受信できません。

－：該当しません。

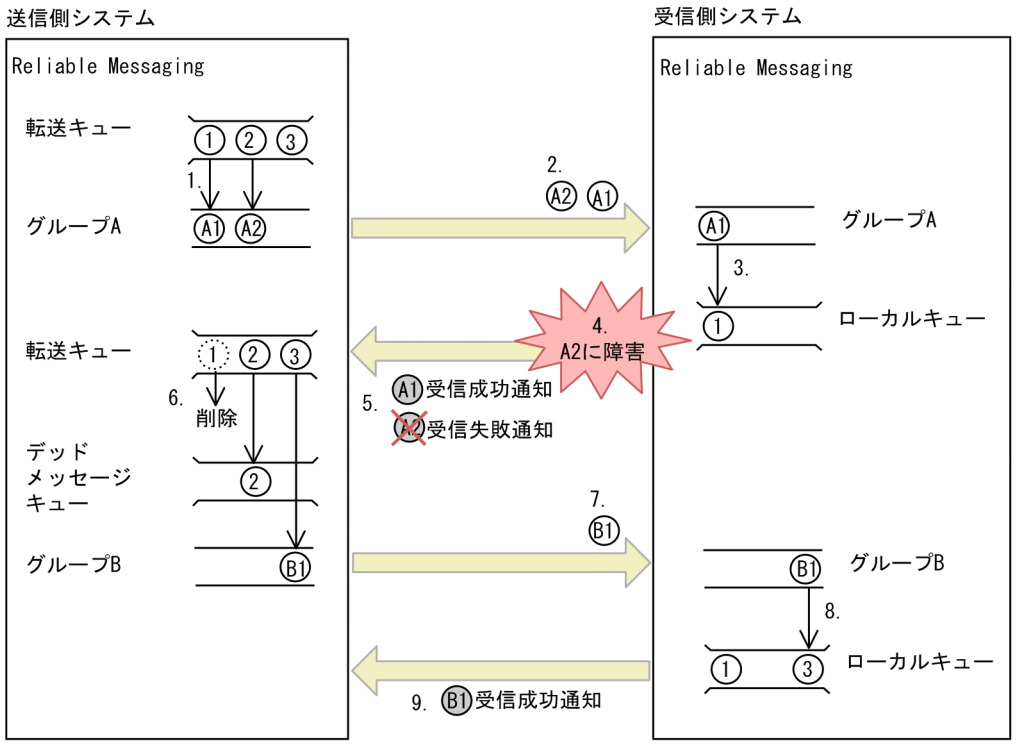
2.4.7　キュー間転送の障害時の動作

受信側システムで受信したメッセージに，致命的な障害がある場合やメッセージの有効期間を超過していた場合，Reliable Messaging は，送信側システムに受信の失敗を通知します。受信の失敗が通知されたメッセージは，送信側システムのデッドメッセージキューに登録されます。さらに，送信側システムでは新しいグループを作成し，障害以降に送信されたメッセージを新しいグループで送信します。

次の図は，送信側システムが A1，A2 のメッセージを送信し，受信側システムで A2 に障害が発生したため，A2 の受信失敗が通知された場合を示しています。1.～9.は，処理の順序を示します。

受信失敗が通知されると，送信側システムの A2 はデッドメッセージキューに登録され，次のメッセージは新しいグループの B1 として送信されます。

図 2-23　キュー間転送の障害時の動作



2.4.8　キュー間転送の注意事項

- デッドメッセージキューに関する注意事項

  送信側システムが送信したメッセージが，受信側システムに受信されたにもかかわらず，何らかの原因で受信成功が通知されない場合，または通知されるのが遅れた場合，そのあとに再送されたメッセージが有効期限に達していて受信失敗が送信側システムに通知されると，そのメッセージは次のように処理されることがあります。

  - 受信側システムに受信されたメッセージが滞留していたときは，送信側システムと受信側システムの両方のデッドメッセージキューに登録される。
  - 受信側システムに受信されたメッセージがキューに登録されていたときは，送信側システムのデッドメッセージキューに登録されているメッセージが，受信側システムで配送される。

- 非永続キューに関する注意事項

  受信側システムのキューが非永続キューの場合，受信側システムのキューに作成されたグループは再起動時に無効になります。つまり，受信側システムの Reliable Messaging が，一つでもメッセージを受信するとグループが作成されます。その後，受信側システムの Reliable Messaging を再起動すると，以降再起動前に作られたグループによるメッセージの受信は失敗します。受信に失敗したメッセージは，受信の失敗を通知することによって，送信側システムのデッドメッセージキューに登録されます。登録先のデッドメッセージキューは，RMDeadMessageQueueName プロパティで指定します。

2.4.9 SOAP プロトコル

転送キューを使用した複数システム間でのメッセージの送受信には，SOAP プロトコルをベースとする WS-Reliability を採用しています。メッセージの送受信には SOAP メッセージを使います。Reliable Messaging がメッセージ送信時に設定する SOAP メッセージの構成要素およびメッセージ受信時の使用方法を次の表に示します。

表 2-5 Reliable Messaging の SOAP メッセージ使用方法

項番	構成要素名		SOAP メッセージの使用方法	
			送信	受信
1	SOAP エンベロープ	SOAP ヘッダ	WS-Reliability1.1 プロトコルのヘッダ情報を設定	WS-Reliability1.1 プロトコルのヘッダ情報を取得※1
2		SOAP ボディ	設定しない	無視する
3	SOAP アタッチメント		メッセージ本体※2 を設定	メッセージ本体を取得

注※1

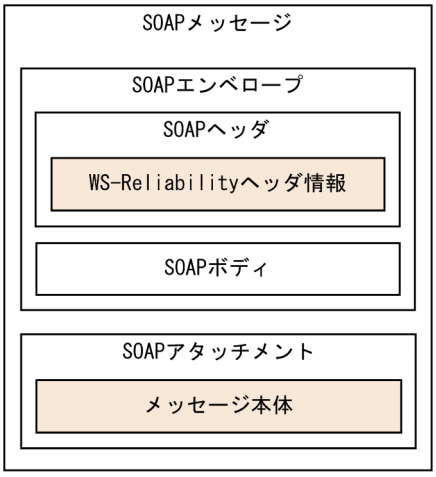
WS-Reliability 以外のプロトコルのヘッダ情報が設定されていた場合，Reliable Messaging はそのプロトコルのヘッダ情報を無視します。

注※2

メッセージ本体は，Reliable Messaging 固有のメッセージオブジェクトをシリアライズしたデータ，またはプリミティブなバイト配列で設定されています。

Reliable Messaging で扱う SOAP メッセージの構成を次の図に示します。

図 2-24 SOAP メッセージの構成



2.5　メッセージの構成

Reliable Messaging のアプリケーションが送受信するメッセージは，JMS メッセージの形式に従います。JMS メッセージは Message インタフェースをルートインタフェースとするオブジェクトです。

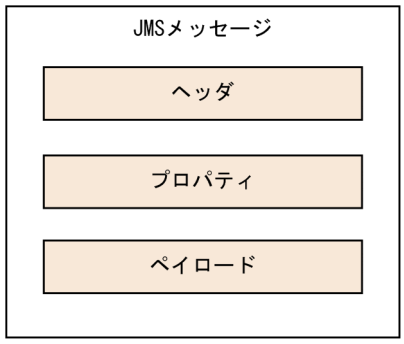
Reliable Messaging で使用するメッセージの構成について説明します。

2.5.1　JMS メッセージの構成

Reliable Messaging は，JMS メッセージを送受信することでアプリケーション連携します。JMS メッセージは，ヘッダ，プロパティおよびペイロードの三つの要素で構成されます。

JMS メッセージの構成を次の図に示します。

図 2-25　JMS メッセージの構成



(1)　JMS メッセージのヘッダ

JMS メッセージのヘッダは，アプリケーションやシステムに，メッセージのあて先や識別子などの制御情報を提供するためのフィールドです。

ヘッダの一覧を次の表に示します。

表 2-6　ヘッダの一覧

項番	ヘッダ	型	説明
1	JMSDestination	javax.jms.Destination	メッセージが送信されるあて先です。
2	JMSDeliveryMode	int	メッセージの永続性を示す値です。
3	JMSMessageID	java.lang.String	プロバイダによって送信される各メッセージを一意に識別するメッセージ識別子です。このヘッダの値は RMSystemName プロパティ指定値と現在時刻および通番によって構成されます。 RMSystemName プロパティ指定値によって，JMSMessageID の一意性は，Reliable Messaging をわたります。

項番	ヘッダ	型	説明
4	JMSTimestamp	long	メッセージの送信時刻を表す値（単位：ミリ秒）です。値はミリ秒で測定した現在時刻と協定世界時のUTC1970年1月1日午前0時との差です。 なお，メッセージのトランザクションやそのほかの受信側のキューイングによって実際の送信はあとから行われるため，メッセージが実際に送信された時刻ではありません。
5	JMSExpiration	long	メッセージの有効期間となる時刻を表す値（単位：ミリ秒）です。値はミリ秒で測定した現在時刻と協定世界時のUTC1970年1月1日午前0時との差です。 このヘッダの値はhrmmkqueコマンドの-eオプション指定値を基に決定されます。
6	JMSRedelivered	boolean	メッセージが再配送中であるかどうかを示す値です。値がtrueのとき，メッセージが再配送中であることを示します。
7	JMSPriority	int	メッセージのプライオリティです。JMSはいちばん低い優先度の0からいちばん高い優先度の9まで，10のレベルのプライオリティを定義しています。
8	JMSReplyTo	javax.jms.Destination	応答先のあて先です。このヘッダが設定されたメッセージを受信したアプリケーションが，該当するあて先へメッセージを送信するかどうかは任意です。
9	JMSCorrelationID	java.lang.String	あるメッセージをほかのメッセージと関連づけるための相関識別子です。一般的には，応答メッセージをその要求メッセージと関連づけるために参照されます。
10	JMSType	java.lang.String	メッセージが送信されるときに，アプリケーションによって提供されるメッセージタイプです。

ヘッダの設定タイミングと取得タイミングを次の表に示します。

表 2-7　ヘッダの設定タイミングと取得タイミング

項番	ヘッダ	設定タイミング	取得タイミング※
1	JMSDestination	QueueSession.createSender(Queue queue)メソッド発行時に，指定したキューの名前が設定されます。 Message.setJMSDestination()メソッドでこのヘッダを設定できません。	メッセージ送信完了後に取得できます。
2	JMSDeliveryMode	メッセージ送信時にReliable Messagingがhrmmkqueコマンドの-mオプション指定値に従って設定します。 Message.setJMSDeliveryMode()メソッドでこのヘッダを設定できません。	メッセージ受信後に取得できます。
3	JMSMessageID	メッセージ送信時にReliable Messagingが設定します。また，デッドメッセージ再登録時にReliable Messagingが再設定します。	メッセージ送信完了後に取得できます。

項番	ヘッダ	設定タイミング	取得タイミング※
		Message.setJMSMessageID()メソッドでこのヘッダを設定できません。	
4	JMSTimestamp	メッセージ送信時に Reliable Messaging が設定します。また，デッドメッセージ再登録時に Reliable Messaging が再設定します。 Message.setJMSTimestamp()メソッドでこのヘッダを設定できません。	メッセージ送信完了後に取得できます。
5	JMSExpiration	送信側と受信側で，設定のタイミングが異なります。 送信側の場合，メッセージ送信時に Reliable Messaging がキュー属性（メッセージ有効期間）に従って設定します。また，デッドメッセージの再登録時に Reliable Messaging が登録するキューの属性（メッセージ有効期間）に従って再設定します。 キュー間転送を実施する場合，受信側では，受信側ローカルキューの属性の設定によって次のように異なります。 • メッセージ有効期間の選択が受信側となっている場合は，受信側のキュー属性（メッセージ有効期間）に従って再設定されます。 • メッセージ有効期間の選択が送信側となっている場合は，送信側で設定された値となり，受信側では再設定しません。 なお，Message インタフェースの setJMSExpiration メソッドでは，このヘッダの値を設定できません。setJMSExpiration メソッドの詳細については「7.4.4 Message インタフェース」を参照してください。	メッセージ送信完了後に取得できます。
6	JMSRedelivered	メッセージをアプリケーションに配送したが，リカバーされたなどでもう一度配送する必要があるときに true に設定されます。 Message.setJMSRedelivered()メソッドでこのヘッダを設定できません。 Reliable Messaging を再度開始すると，false に設定されます。	メッセージ受信後に取得できます。
7	JMSPriority	QueueSender.setPriority()メソッドまたは QueueSender.send()メソッドで指定した priority の値が設定されます。両方のメソッドで priority が指定されている場合は，QueueSender.send()メソッドで指定した値が優先されます。 Message.setJMSPriority()メソッドでこのヘッダを設定できません。	メッセージ送信完了後に取得できます。
8	JMSReplyTo	Message.setJMSReplyTo()メソッド発行時に，指定値が設定されます。設定は任意です。 受信側からのメッセージ返信を必要とする場合，送信側のユーザがこのヘッダにあて先を設定します。	設定タイミング後に取得できます。

項番	ヘッダ	設定タイミング	取得タイミング※
9	JMSCorrelationID	Message.setJMSCorrelationID()メソッド発行時に，指定値が設定されます。設定は任意です。	設定タイミング後に取得できます。
10	JMSType	Message.setJMSType()メソッド発行時に，指定値が設定されます。設定は任意です。	設定タイミング後に取得できます。

注※

表中に示したタイミング以外でヘッダの値を取得した場合，ヘッダが設定されていないときの値が返されます。詳細については，「7.4.4　Message インタフェース」を参照してください。

ObjectMessage（ペイロードに BytesContainer を含む）を送受信した場合のヘッダの設定タイミングと取得タイミングを次の表に示します。

表 2-8　ObjectMessage（ペイロードに BytesContainer を含む）を送受信した場合のヘッダの設定タイミングと取得タイミング

項番	ヘッダ	設定タイミング	取得タイミング※
1	JMSDestination	QueueSession.createSender(Queue queue)メソッド発行時に，指定したキューの名前が設定されます。 Message.setJMSDestination()メソッドでこのヘッダを設定できません。	送信側ではメッセージ送信完了後に取得できます。受信側では取得できません。
2	JMSDeliveryMode	メッセージ送信時に Reliable Messaging が hrmmkque コマンドの-m オプション指定値に従って設定します。 Message.setJMSDeliveryMode()メソッドでこのヘッダを設定できません。	メッセージ受信後に取得できます。
3	JMSMessageID	メッセージ送信時に Reliable Messaging が設定します。また，デッドメッセージ再登録時に Reliable Messaging が再設定します。 Message.setJMSMessageID()メソッドでこのヘッダを設定できません。	送信側ではメッセージ送信完了後に取得できます。受信側では取得できません。
4	JMSTimestamp	メッセージ送信時に Reliable Messaging が設定します。また，デッドメッセージ再登録時に Reliable Messaging が再設定します。 Message.setJMSTimestamp()メソッドでこのヘッダを設定できません。	送信側ではメッセージ送信完了後に取得できます。受信側では取得できません。
5	JMSExpiration	送信側ではメッセージ送信時に Reliable Messaging がキュー属性（メッセージ有効期間）に従って設定します。ローカルキュー送信の場合も同様です。また，デッドメッセージの再登録時に Reliable Messaging が登録するキューの属性（メッセージ有効期間）に従って再設定します。 キュー間転送を実施した場合，受信側では，受信側ローカルキューの属性の設定によって次のように異なります。	キュー間転送の場合，送信側ではメッセージ送信完了後に取得できます。受信側では別の値が取得できます。

項番	ヘッダ	設定タイミング	取得タイミング※
		• メッセージ有効期間の選択が受信側となっている場合は，受信側のキュー属性（メッセージ有効期間）に従って設定されます。 • メッセージ有効期間の選択が送信側となっている場合は，WS-Reliability 電文の ExpiryTime タグと同じ値が設定されます。	
6	JMSRedelivered	メッセージをアプリケーションに配送したが，リカバーされたなどでもう一度配送する必要があるときに true に設定されます。 Message.setJMSRedelivered()メソッドでこのヘッダを設定できません。 Reliable Messaging を再度開始すると，false に設定されます。	メッセージ受信後に取得できます。
7	JMSPriority	送信側では QueueSender.setPriority()メソッドまたは QueueSender.send()メソッドで指定した priority の値が設定されます。両方のメソッドで priority が指定されている場合は，QueueSender.send()メソッドで指定した値が優先されます。 キュー間転送を実施した場合，受信側で受信メッセージが永続される前に，デフォルト値が設定されます。 Message.setJMSPriority()メソッドでこのヘッダを設定できません。	キュー間転送の場合，送信側ではメッセージ送信完了後に取得できます。受信側では別の値が取得できます。
8	JMSReplyTo	Message.setJMSReplyTo()メソッド発行時に，指定値が設定されます。設定は任意です。 受信側からのメッセージ返信を必要とする場合，送信側のユーザがこのヘッダにあて先を設定します。	送信側では設定タイミング後に取得できます。受信側では取得できません。
9	JMSCorrelationID	Message.setJMSCorrelationID()メソッド発行時に，指定値が設定されます。設定は任意です。	送信側では設定タイミング後に取得できます。受信側では取得できません。
10	JMSType	Message.setJMSType()メソッド発行時に，指定値が設定されます。設定は任意です。	送信側では設定タイミング後に取得できます。受信側では取得できません。

注※

表中に示したタイミング以外でヘッダの値を取得した場合，ヘッダが設定されていないときの値が返されます。詳細については，「7.4.4　Message インタフェース」を参照してください。

(2)　JMS メッセージのプロパティ

JMS メッセージのプロパティは，ヘッダに追加する制御情報が格納されたフィールドです。次に示す種類があります。

1. JMS 定義のプロパティ

JMS が定義するプロパティです。これらのプロパティの完全なセットは，JMS 仕様で定義されています。

半角英字だけで定義され，大文字と小文字は区別されます。

2. Reliable Messaging 固有のプロパティ
   Reliable Messaging が独自に提供するプロパティです。
   半角英字だけで定義され，大文字と小文字は区別されます。
3. アプリケーション指定のプロパティ
   ユーザが必要に応じて定義するプロパティです。ただし，使用できるプロパティ名には制限があります。
   プロパティ名には半角および全角を使用できます。また，大文字と小文字は区別されます。

(a) JMS 定義のプロパティ

JMS 定義のプロパティの一覧を次の表に示します。

表 2-9 JMS 定義のプロパティの一覧

項番	プロパティ	型	説明
1	JMSXUserID	java.lang.String	メッセージを送信するユーザのユーザ識別子です。 このプロパティは未サポートです。
2	JMSXAppID	java.lang.String	メッセージを送信するアプリケーションのアプリケーション識別子です。 このプロパティは未サポートです。
3	JMSXConsumerTXID	java.lang.String	メッセージが受信および承認されたトランザクションのトランザクション識別子です。 このプロパティは未サポートです。
4	JMSXProducerTXID	java.lang.String	メッセージが生成されたトランザクションのトランザクション識別子です。 このプロパティは未サポートです。
5	JMSXRcvTimestamp	long	メッセージがコンシューマに配送された配送時刻です。 時刻の値はミリ秒で測定した現在時刻と協定世界時の UTC1970 年 1 月 1 日午前 0 時との差です。
6	JMSXDeliveryCount	int	メッセージ配送回数です。
7	JMSXState	int	コンシューマから送信されたメッセージの各コピーの状態を示す値です。 このプロパティは未サポートです。
8	JMSXGroupID	java.lang.String	メッセージが属しているグループのグループ識別子です。
9	JMSXGroupSeq	int	メッセージが属しているグループのシーケンス番号です。

JMS 定義のプロパティの設定タイミングと取得タイミングを次の表に示します。

表 2-10　JMS 定義のプロパティの設定タイミングと取得タイミング

項番	プロパティ	設定タイミング	取得タイミング
1	JMSXUserID	−	−
2	JMSXAppID	−	−
3	JMSXConsumerTXID	−	−
4	JMSXProducerTXID	−	−
5	JMSXRcvTimestamp	メッセージをコンシューマに配送した際に，Reliable Messaging が設定します。	メッセージ受信後に取得できます。
6	JMSXDeliveryCount	メッセージ承認時に Reliable Messaging が設定します。Reliable Messaging を再度開始すると，この値は 0 に設定されます。	メッセージ受信後に取得できます。
7	JMSXState	−	−
8	JMSXGroupID	Message.setStringProperty("JMSXGroupID", groupID)または setObjectProperty("JMSXGroupID", groupID)メソッド発行時に，指定した groupID の値が設定されます。設定は任意です。ユーザが設定できる文字数は半角全角を問わないで 512 文字以下です。	設定タイミング後に取得できます。
9	JMSXGroupSeq	Message.setIntProperty("JMSXGroupSeq", seq)メソッド発行時に，指定した seq の値が設定されます。設定は任意です。 ユーザが設定できる値は 0 以上です。	設定タイミング後に取得できます。

（凡例）

−：未サポートのため該当しません。

また，ObjectMessage（ペイロードに BytesContainer を含む）を送受信した場合の JMS 定義のプロパティの設定タイミングと取得タイミングを次の表に示します。

表 2-11　ObjectMessage（ペイロードに BytesContainer を含む）を送受信した場合の JMS 定義のプロパティの設定タイミングと取得タイミング

項番	プロパティ	設定タイミング	取得タイミング
1	JMSXUserID	−	−
2	JMSXAppID	−	−
3	JMSXConsumerTXID	−	−
4	JMSXProducerTXID	−	−

項番	プロパティ	設定タイミング	取得タイミング
5	JMSXRcvTimestamp	メッセージをコンシューマに配送した際に，Reliable Messaging が設定します。	メッセージ受信後に取得できます。
6	JMSXDeliveryCount	メッセージ承認時に Reliable Messaging が設定します。Reliable Messaging を再度開始すると，この値は 0 に設定されます。	メッセージ受信後に取得できます。
7	JMSXState	−	−
8	JMSXGroupID	Message.setStringProperty("JMSXGroupID", groupID)または setObjectProperty("JMSXGroupID", groupID)メソッド発行時に，指定した groupID の値が設定されます。設定は任意です。ユーザが設定できる文字数は半角全角を問わないで 512 文字以下です。	送信側では設定タイミング後に取得できます。受信側では取得できません。
9	JMSXGroupSeq	Message.setIntProperty("JMSXGroupSeq", seq)メソッド発行時に，指定した seq の値が設定されます。設定は任意です。 ユーザが設定できる値は 0 以上です。	送信側では設定タイミング後に取得できます。受信側では取得できません。

（凡例）

−：未サポートのため該当しません。

(b) Reliable Messaging 固有のプロパティ

Reliable Messaging 固有のプロパティの一覧を次の表に示します。

表 2-12 Reliable Messaging 固有のプロパティの一覧

項番	プロパティ	型	説明
1	JMS_HITACHI_DeadMessageTimestamp	long	メッセージがデッドメッセージキューに移動された時刻を示す値（単位：ミリ秒）です。 時刻の値はミリ秒で測定した現在時刻と協定世界時の UTC1970 年 1 月 1 日午前 0 時との差です。
2	JMS_HITACHI_DeadMessageCause	java.lang.String	メッセージがデッドメッセージキューに移動された原因を示す文字列※です。
3	JMS_HITACHI_DeadMessageOriginalQueueName	java.lang.String	メッセージがデッドメッセージキューに移動される前に保存されていたキュー名です。
4	JMS_HITACHI_DeadMessageID	java.lang.String	デッドメッセージを一意に識別するための値です。このプロパティ

項番	プロパティ	型	説明
			の値は，コンフィグレーションプロパティの RMSystemName プロパティの値と現在時刻および通番によって決定されます。
5	JMS_HITACHI_UnitID	java.lang.String	パラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）で利用するユニット識別子の値です。このプロパティの値は，ユーザがアプリケーションで送信するメッセージに対して設定します。

注※

次の表に示す文字列のどれかが設定されます。

表 2-13　デッドメッセージキューに移動された原因を示す文字列

項番	文字列	説明
1	DMQCAUSE-001	メッセージが有効期限に達しました。
2	DMQCAUSE-002	メッセージの配送回数が最大値に達しました。
3	DMQCAUSE-003	順序制御のために滞留しているメッセージが有効期限に達しました。または，滞留しているメッセージの属しているグループが閉鎖しました。
4	DMQCAUSE-004	メッセージの受信に失敗した通知を受け取りました。

Reliable Messaging 固有のプロパティ固有のプロパティの設定タイミングと取得タイミングを次の表に示します。

表 2-14　Reliable Messaging 固有のプロパティの設定タイミングと取得タイミング

項番	プロパティ	設定タイミング	取得タイミング
1	JMS_HITACHI_DeadMessageTimestamp	メッセージがデッドメッセージキューに保存される際に，Reliable Messaging が設定します。 なお，デッドメッセージが再登録される際に削除します。	デッドメッセージキューからメッセージを受信したあとで取得できます。
2	JMS_HITACHI_DeadMessageCause	メッセージがデッドメッセージキューに保存される際に，Reliable Messaging が設定します。 なお，デッドメッセージが再登録される際に削除します。	デッドメッセージキューからメッセージを受信したあとで取得できます。

項番	プロパティ	設定タイミング	取得タイミング
3	JMS_HITACHI_DeadMessageOriginalQueueName	メッセージがデッドメッセージキューに保存される際に，Reliable Messaging が設定します。 なお，デッドメッセージが再登録される際に削除します。	デッドメッセージキューからメッセージを受信したあとで取得できます。
4	JMS_HITACHI_DeadMessageID	メッセージがデッドメッセージキューに保存される際に，Reliable Messaging が設定します。なお，デッドメッセージが再登録される際に削除します。	デッドメッセージキューからメッセージを受信したあとで取得できます。
5	JMS_HITACHI_UnitID	ユーザが Message.setStringProperty("JMS_HITACHI_UnitID", unitID)メソッド発行時に，指定した unitID の値が設定されます。設定は任意です。ユーザが設定できる文字数は半角全角を問わないで 512 文字以下です。	設定タイミング後に取得できます。

(c) アプリケーション指定のプロパティ

ユーザが必要に応じて定義するプロパティです。

ユーザは，メッセージインタフェースの set<型名>Property()メソッドの name 引数に任意のプロパティ名を指定してプロパティに値を設定します。get<型名>Property()メソッドの name 引数に set メソッドと同じ名前を指定してプロパティから値を取得します。

プロパティ名はユーザの実装に依存します。ただし，次に示す名前は使用できません。

- JMS 定義のプロパティが使用する接頭語（JMSX）で始まるプロパティ名
- Reliable Messaging 固有のプロパティが使用する接頭語（JMS_）で始まるプロパティ名
- メッセージセレクタで使用する予約語（NOT，AND，OR，BETWEEN，LIKE，IN，IS，NULL，TRUE および FALSE）と同じプロパティ名

(d) プロパティの型変換

Reliable Messaging 固有のプロパティとアプリケーション指定のプロパティに設定できる型は，boolean，byte，short，int，long，float，double および String です。書き込まれたときと異なる型でプロパティを読み取る場合，型は変換されます。

プロパティの型変換を次の表に示します。

表 2-15　プロパティの型変換

書き込み時の型	読み取り時の型							
	boolean	byte	short	int	long	float	double	String
boolean	○							○
byte		○	○	○	○			○
short			○	○	○			○
int				○	○			○
long					○			○
float						○	○	○
double							○	○
String	○	○	○	○	○	○	○	○

（凡例）

○：型が変換されます。

空白：型が変換されません。読み取り時に JMSException（MessageFormatException）が発生します。

注

String 型で書き込まれた文字列を String 以外の型に変換して読み取る場合，書き込まれた文字列を，変換後の型でラッパークラスの valueOf メソッドによって解釈できないとき，java.lang.NumberFormatException が発生します。

(3)　JMS メッセージのペイロード

JMS メッセージのペイロードは，メッセージ本体のフィールドです。

Reliable Messaging は，次に示すメッセージインタフェースを提供します。各メッセージインタフェースのペイロードを次の表に示します。

表 2-16　メッセージインタフェースのペイロード

項番	インタフェース名	ペイロードの説明
1	Message	ペイロードがありません。
2	BytesMessage	ペイロードがプリミティブなバイトの配列です。
3	ObjectMessage※	ペイロードがシリアライズできる Java オブジェクトです（BytesContainer およびユーザが定義したクラスのオブジェクトを格納できます）。
4	TextMessage	ペイロードが java.lang.String 型です。

注※

非永続版リソースアダプタの場合，ObjectMessage のペイロードに BytesContainer は格納できません。

(a) ObjectMessage のペイロードに BytesContainer を格納する場合

永続版リソースアダプタの場合，ObjectMessage オブジェクトの setObject メソッドの引数に BytesContainer オブジェクトを指定することで，BytesContainer を含む ObjectMessage を送信できます。詳細は「7.7　転送データ相互接続用インタフェースの使い方」を参照してください。

(b) ObjectMessage のペイロードにユーザが定義したクラスのオブジェクトを格納する場合

ObjectMessage のペイロードにユーザが定義したクラスのオブジェクトを格納する場合は，送受信するアプリケーションにユーザが定義したクラスを含める必要があります。アプリケーションにユーザが定義したクラスを含めない場合は，Application Server の J2EE サーバ用オプション定義ファイルにユーザが定義したクラスまたはクラスが含まれる JAR のクラスパスを，コンテナ拡張ライブラリとして指定する必要があります。

これらの設定や指定をしていない場合，取り出したメッセージからペイロードを取得するときに例外が発生します。ペイロードの取得で例外が発生した場合でも，取り出したメッセージのトランザクションが決着すると，メッセージが配信済みになるので注意してください。

また，Reliable Messaging 01-02 以前でキューに登録したメッセージを移行した場合や，Reliable Messaging 01-02 以前とキュー間転送する場合も，Application Server の J2EE サーバ用オプション定義ファイルにユーザが定義したクラスまたはクラスが含まれる JAR のクラスパスを，コンテナ拡張ライブラリとして指定する必要があります。

Application Server の J2EE サーバ用オプション定義ファイル（usrconf.cfg）は，次に示す場所に格納されています。

```
<Application Serverのインストールディレクトリ>¥CC¥server¥usrconf¥ejb¥<サーバ名>¥usrconf.cfg
```

usrconf.cfg ファイルをテキストエディタで開き，次の行を追加します。

```
add.class.path=<コンテナ拡張ライブラリ用のclassまたはJARのクラスパス>
```

J2EE サーバ用オプション定義ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

2.5.2　メッセージ要素のアクセスモード

JMS メッセージのメッセージ要素（ヘッダ，プロパティ，およびペイロード）の値を取得したり，設定したりするには，メッセージインタフェースが提供するメソッドを使用します。各要素には，次に示すアクセスモードがあります。

- 読み取り書き込み両用モード（Read／Write mode）
  読み取りも書き込みもできるモードです。ヘッダ，プロパティおよびペイロードで発生するモードです。

- 読み取り専用モード（Read Only mode）
  情報の読み取りだけができるモードです。このモードのときにメッセージ要素に情報を書き込もうとすると MessageNotWriteableException が発生します。プロパティとペイロードで発生するモードです。
- 書き込み専用モード（Write Only mode）
  情報の書き込みだけができるモードです。このモードのときにメッセージ要素から情報を読み取ろうとすると MessageNotReadableException が発生します。ペイロードで発生するモードです。

メッセージ要素とアクセスモードの関係を次の表に示します。

表 2-17　メッセージ要素とアクセスモードの関係

項番	メッセージ要素	読み取り書き込み両用モード	読み取り専用モード	書き込み専用モード
1	ヘッダ	○	×	×
2	プロパティ	○	○	×
3	ペイロード	○	○	○

（凡例）
○：あります。
×：ありません。

(1)　メッセージ生成時のアクセスモード

メッセージインタフェースのインスタンスは，次に示すメソッドの発行によって生成します。

- QueueSession.createMessage()メソッド：Message インタフェースの場合
- QueueSession.createBytesMessage()メソッド：BytesMessage インタフェースの場合
- QueueSession.createObjectMessage()メソッド：ObjectMessage インタフェースの場合
- QueueSession.createTextMessage()メソッド：TextMessage インタフェースの場合

メッセージ生成時のメッセージ要素のアクセスモードを次の表に示します。

表 2-18　メッセージ生成時のメッセージ要素のアクセスモード

項番	インタフェース	メッセージ要素	モード
1	Message	ヘッダ	読み取り書き込み両用モード
2		プロパティ	
3	BytesMessage	ヘッダ	読み取り書き込み両用モード
4		プロパティ	
5		ペイロード	書き込み専用モード
6	ObjectMessage	ヘッダ	読み取り書き込み両用モード

項番	インタフェース	メッセージ要素	モード
7		プロパティ	
8		ペイロード	
9	TextMessage	ヘッダ	
10		プロパティ	
11		ペイロード	

(2) メッセージ受信時のアクセスモード

受信側アプリケーションは，QueueReceiver.receive()または receiveNoWait()メソッドの戻り値として，メッセージを受信します。

メッセージ受信時のメッセージ要素のアクセスモードを次の表に示します。

表 2-19 メッセージ受信時のメッセージ要素のアクセスモード

項番	インタフェース	メッセージ要素	モード
1	Message	ヘッダ	読み取り書き込み両用モード
2		プロパティ	読み取り専用モード
3	BytesMessage	ヘッダ	読み取り書き込み両用モード
4		プロパティ	読み取り専用モード
5		ペイロード	
6	ObjectMessage	ヘッダ	読み取り書き込み両用モード
7		プロパティ	読み取り専用モード
8		ペイロード	
9	TextMessage	ヘッダ	読み取り書き込み両用モード
10		プロパティ	読み取り専用モード
11		ペイロード	

また，メッセージ要素のアクセスモードは，メッセージ受信後に特定のタイミングによって移行することがあります。アクセスモードが移行するタイミングを次の表に示します。

表 2-20 アクセスモードが移行するタイミング

項番	タイミング	メッセージ要素	移行前	移行後
1	Message.clearProperties()メソッドの発行	プロパティ	読み取り専用モード	読み取り書き込み両用モード

項番	タイミング	メッセージ要素	移行前	移行後
2	ObjectMessage.clearBody()メソッドの発行 TextMessage.clearBody()メソッドの発行	ペイロード		
3	BytesMessage.clearBody()メソッドの発行			書き込み専用モード
4	BytesMessage.reset()メソッドの発行		書き込み専用モード	読み取り専用モード

2.5.3 メッセージとキューの関係

Reliable Messaging のキューは，種類によって格納できるメッセージインタフェースが異なります。また，キュー作成時の指定値によってメッセージの管理方法が異なります。

(1) メッセージインタフェースとキューの種類

使用するメッセージインタフェースによって，そのメッセージを格納できるキューの種類は異なります。

メッセージインタフェースと格納できるキューの種類を次の表に示します。

表 2-21 メッセージインタフェースと格納できるキューの種類

項番	インタフェース	格納できるキューの種類
1	Message	ローカルキュー 転送キュー
2	BytesMessage	ローカルキュー 転送キュー 送信用共用キュー（ただし，ペイロードだけ） 受信用共用キュー（ただし，ペイロードだけ）
3	ObjectMessage	ローカルキュー※ 転送キュー
4	TextMessage	ローカルキュー 転送キュー

注※

BytesContainer を ObjectMessage のペイロードに設定して送信する処理では，ペイロードに設定した BytesContainer のデータだけを送信します。

(2)　キュー作成時の指定値とメッセージのヘッダ

キューの作成は，永続版リソースアダプタの場合と非永続版リソースアダプタの場合とで作成方法が異なります。

- 永続版リソースアダプタの場合

  hrmmkque コマンドによって作成されます。

- 非永続版リソースアダプタの場合

  キュー定義文（定義の先頭に hrmmkque を記述）を指定して，キュー作成ファイルを作成します。キュー作成ファイルに指定されたキュー定義文を基に，Reliable Messaging 開始時にローカルキューが作成されます。

作成時の指定値によって，そのキューに格納されるメッセージの管理方法は異なります。

特に，次に示すオプション指定値はメッセージのヘッダに影響します。

- キューの永続性（永続版リソースアダプタの場合，-m オプション指定値）

  送信されたメッセージが永続キュー属性のキューに登録されるとき，そのメッセージは永続メッセージになります。アプリケーションが永続メッセージを受信するとき，メッセージの JMSDeliveryMode ヘッダは DeliveryMode.PERSISTENT に設定されています。

  送信されたメッセージが非永続キュー属性のキューに登録されるとき，そのメッセージは非永続メッセージになります。アプリケーションが非永続メッセージを受信するとき，メッセージの JMSDeliveryMode ヘッダは DeliveryMode.NON_PERSISTENT に設定されています。

  なお，非永続版リソースアダプタの場合は，非永続メッセージしか扱いません。そのため，作成されるキューは非永続キュー属性のキューだけになります。

  永続メッセージと非永続メッセージが，各属性のキューでどのように管理されるかについては，「2.3.1 キューの永続性」を参照してください。

- メッセージの有効期間（-e オプション指定値）

  送信されたメッセージがキューに登録されるとき，メッセージの有効期間が秒単位で設定されます。アプリケーションがメッセージを受信するとき，メッセージの JMSExpiration ヘッダに有効期間が設定されています。

  有効期間が指定されたメッセージが，どのように管理されるかについては，「2.3.5　メッセージの有効期間」を参照してください。

2.5.4　メッセージサイズを見積もる方法

Reliable Messaging で送受信するメッセージのメッセージサイズを見積もるための計算式は次のとおりです。

ローカルキューまたは転送キューを使用する場合

　7500 ＋ N ＋ M ＋ L ＋ S（単位：バイト）

- N ＝ユーザが設定できる JMS ヘッダの設定サイズ
  設定するヘッダによって，次の値を加算します。
  JMSReplyTo：シリアライズしたオブジェクトのサイズ
  JMSCorrelationID：半角文字数＋全角文字数×3
  JMSType：半角文字数＋全角文字数×3
- M ＝ JMS 定義プロパティの設定サイズ
  設定するプロパティによって，次の値を加算します。
  JMSXGroupID：半角文字数＋全角文字数×3
  JMSXGroupSeq：100
- L ＝ l ＋ m
  l：ユーザ定義プロパティ名のサイズ（(半角文字数＋全角文字数×3）×プロパティ数)
  m：ユーザ定義プロパティ値のサイズ
  プロパティの型によって，次の値を加算します。
  java.lang.String：(半角文字数＋全角文字数×3）×プロパティ数
  java.lang.String 以外：100×プロパティ数
- S ＝ JMS メッセージのペイロード設定サイズ
  メッセージインタフェースによって，次の値を加算します。
  TextMessage：半角文字数＋全角文字数×3
  BytesMessage：半角文字数＋全角文字数×2
  ObjectMessage：シリアライズしたオブジェクトのサイズ

共用キューを使用する場合

1500 ＋半角文字数＋全角文字数×2（単位：バイト）

2.6　アプリケーションからのメッセージ操作

Reliable Messaging のアプリケーションは，JMS インタフェースまたは転送データ相互接続用インタフェースを使用してメッセージを操作します。ここでは，アプリケーション作成時に留意しなくてはならない点について説明します。

2.6.1　メッセージの受信制御

アプリケーションが複数のメッセージを受信するとき，メッセージの受信順序はプライオリティ，FIFO およびメッセージセレクタによって決定されます。

- プライオリティでの順序

  送信側アプリケーションが QueueSender.setPriority()または send()メソッドで設定したプライオリティの高い順で，受信側アプリケーションはメッセージを受信します。同じプライオリティのメッセージは FIFO での順序で受信します。

- FIFO での順序

  送信側アプリケーションがすべてのメッセージにプライオリティを設定しなかった場合，またはすべてのメッセージに同じプライオリティを設定した場合，受信側アプリケーションは，メッセージが送信またはコミットされた順序で受信します。

  ただし，送信側アプリケーションが複数のスレッドから同時に同じキューにメッセージを送信した場合，メッセージを受信する順序は保証されません。

- メッセージセレクタでの順序

  受信側アプリケーションがメッセージセレクタを指定すると，キューに登録されたメッセージの中から，特定の条件に合うメッセージだけを受信できます。複数のメッセージが条件に合う場合は，プライオリティでの順序または FIFO での順序でメッセージを受信できます。

  メッセージセレクタについては，「2.6.2　メッセージセレクタ」を参照してください。

2.6.2　メッセージセレクタ

QueueSession.createReceiver()または createBrowser()メソッドの引数に，メッセージセレクタを指定できます。メッセージセレクタは，受信するメッセージの条件を指定するための java.lang.String 型の構文です。受信側アプリケーションはメッセージセレクタを使用することによって，構文の条件に合うヘッダおよびプロパティの値を持つメッセージをキューから受信できます。

注意

　受信用共用キューからメッセージを受信または閲覧する場合は，メッセージセレクタは使用できません。

メッセージセレクタには，識別子，比較演算子，リテラルおよび論理演算子を組み合わせた構文を指定します。

(1) 識別子

次に示す識別子を指定できます。識別子は半角英字であり，大文字と小文字が区別されます。

不正な識別子を指定した場合は InvalidSelectorException が発生します。

- ヘッダの JMSCorrelationID
- プロパティの JMSXGroupID
- プロパティの JMSXGroupSeq

(2) 比較演算子

等号（=）だけを指定できます。比較演算子は半角記号です。

不正な比較演算子を指定した場合は InvalidSelectorException が発生します。

(3) リテラル

指定できるリテラルは識別子によって異なります。各識別子に指定できるリテラルを次に示します。

不正なリテラルを指定した場合は InvalidSelectorException が発生します。

- ヘッダの JMSCorrelationID またはプロパティの JMSXGroupID の場合
  文字列リテラルを指定できます。
  文字列リテラルはアポストロフィ（'）で囲んでください。文字列リテラルにアポストロフィ（'）を含む場合はアポストロフィ（'）を二つ記述します。例えば"literal's"を指定する場合は，次に示すとおり記述します。

```
'literal''s'
```

- プロパティの JMSXGroupSeq の場合
  厳密な数値リテラルを指定できます。
  数値リテラルは半角数字と半角記号です。int の範囲の値を指定できます。0 以上の整数値（57，＋ 62 など）を指定してください。負数は指定しないでください。

(4) 論理演算子

AND だけを指定できます。論理演算子は半角英字であり，大文字だけを指定できます。

JMSXGroupID についての式と JMSXGroupSeq についての式の組み合わせだけが有効で，論理演算子の前後には空白が必要です。

OR や NOT を使用したり，JMSCorrelationID についての式が含まれる場合は InvalidSelectorException が発生します。

(5) 指定できる構文の例

メッセージセレクタに指定できる構文の例を次に示します。

```
JMSCorrelationID = 'aaa'
JMSXGroupID = 'bbb'
JMSXGroupSeq = 1
JMSXGroupID = 'bbb' AND JMSXGroupSeq = 2
JMSXGroupSeq = 3 AND JMSXGroupID = 'ccc'
```

2.6.3 Message-driven Bean との連携

受信用のキューにメッセージが登録されたことを契機に，Message-driven Bean を実装したアプリケーションはメッセージの配信を受けることができます。

メッセージの配信を受けるには，アプリケーションで次に示すインタフェースを実装するクラスを定義し，MessageListener の onMessage()メソッドをオーバライドしてください。

- javax.ejb.MessageDrivenBean
- javax.jms.MessageListener

キューにメッセージが登録されると，それを契機に Reliable Messaging は onMessage()メソッドを呼び出し，その引数としてメッセージを渡します。ユーザは onMessage()メソッドの内部を実装し，業務に合わせてメッセージを処理するコードを作成してください。

Message-driven Bean 実装時の注意事項については，マニュアル「アプリケーションサーバ アプリケーション開発ガイド」を参照してください。また，キューのリファレンスの解決方法については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」を参照してください。

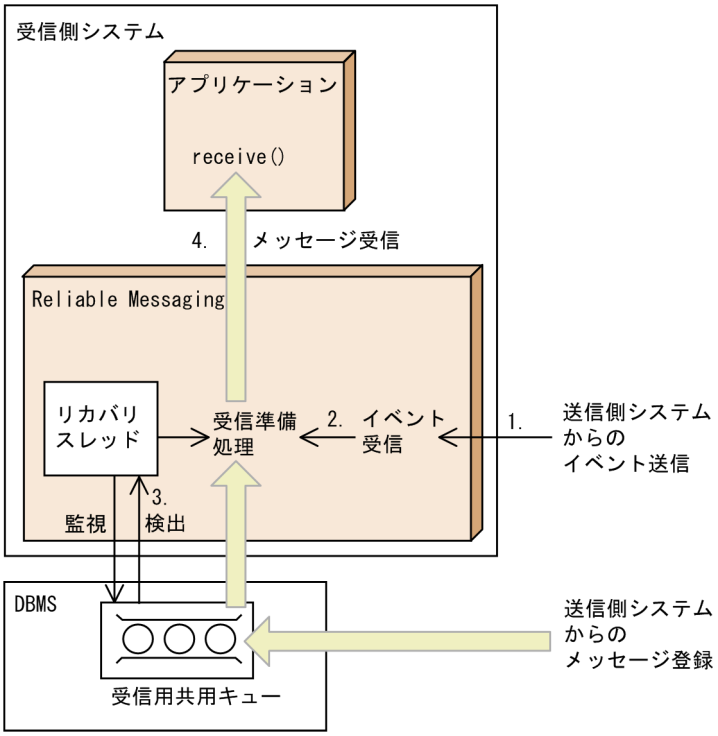
2.6.4 共用キューでのメッセージ受信時の処理の流れ

共用キューを使用して複数システム間でのアプリケーション連携をする場合について，メッセージ受信の処理の流れを説明します。

共用キューを使用して複数システム間でのアプリケーション連携をする場合に受信側アプリケーションが QueueReceiver.receive()または receiveNoWait()メソッドを発行してメッセージを受信するには，Reliable Messaging 内部で受信準備を完了しておく必要があります。受信準備はイベントの受信，またはリカバリスレッドでのメッセージ登録の検出を契機として開始されます。

共用キューを使用して複数システム間でのアプリケーション連携をする場合の受信処理の流れを次の図に示します。

図 2-26　共用キューを使用して複数システム間でのアプリケーション連携をする場合の受信処理の流れ



図中の番号について，説明します。

1. イベントの送信

   送信側システムは送信用共用キューへのメッセージ登録が完了したあと，受信側システムにイベントを送信します。

   送信時のあて先として受信用共用キューに定義されているホスト名とポート番号が使用されます。

2. イベントの受信

   イベントを受信した受信側システムでは，Reliable Messaging が受信準備を開始します。受信準備では，DB にアクセスしてメッセージオブジェクトを生成したり，トランザクションの制御元にメッセージの有効化を連絡したりします。

   受信用共用キューを使用する場合には RMSHConnectFlag プロパティに true を指定してください。また，RMSHPort プロパティにイベントを受信するためのポート番号を指定してください。

3. リカバリスレッドでのメッセージの検出

   イベントの受信に失敗したときのために，Reliable Messaging は内部にリカバリスレッドを動作させています。リカバリスレッドは RMSHRecoveryTimerInterval プロパティ指定値（単位：秒）で自システムの受信用共用キューを監視します。メッセージが登録されていても受信準備が開始されていないキューを検出すると，準備処理が開始されます。

4. アプリケーションでのメッセージの受信

   受信準備が完了したあと，アプリケーションが QueueReceiver.receive()または receiveNoWait()メソッドを発行すると，Reliable Messaging はキューから取り出したメッセージをアプリケーションに配送します。

2.6.5　DB アクセス時の認証

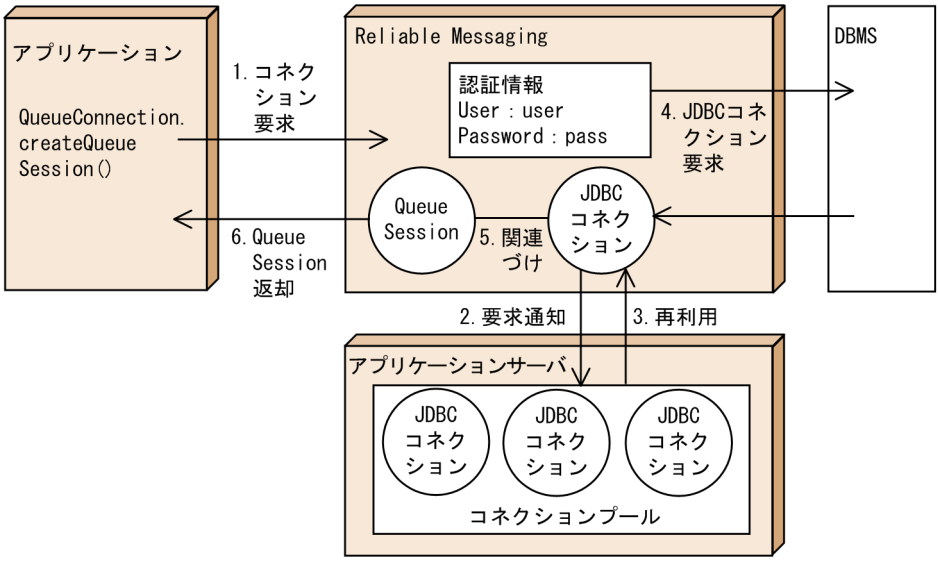
アプリケーションが取得するコネクション（QueueSession オブジェクト）には，JDBC コネクションが関連づけられます。

アプリケーションがコネクションを要求すると，Reliable Messaging は Application Server のコネクションプーリング機能と連携し，プーリングされている JDBC コネクションを再利用します。JDBC コネクションがプールにないときは，DBMS から新しい JDBC コネクションを取得します。このとき DBMS の接続ユーザの認証にはアプリケーションで指定した認証方法（コンテナ認証またはアプリケーション認証）が使用されます。認証情報の設定については，「3.4.5　Reliable Messaging のプロパティ定義（永続版リソースアダプタの場合）」を参照してください。

また，アプリケーションがコネクションを解放するときは，Application Server のコネクションプーリング機能を経由して処理されます。コネクションプーリング数（Application Server での定義値）に空きがあると JDBC コネクションは解放されないでプールされます。

アプリケーションからのコネクション取得の概要を次の図に示します。

図 2-27　アプリケーションからのコネクション取得の概要



図中の番号について，説明します。

1. コネクションの要求

   アプリケーションが QueueConnection.createQueueSession()メソッドを発行し，コネクションを要求します。

2. コネクションの要求通知

   Reliable Messaging が Application Server にコネクションの要求通知をします。

3. JDBC コネクションの再利用

   Application Server 内にプーリング中のコネクションがある場合はプール中の JDBC コネクションが再利用されます。

4. JDBC コネクションの要求

   プーリング中のコネクションがない場合，Reliable Messaging は DBMS に要求して JDBC コネクションを取得します。コンテナ認証の場合は，リソースアダプタのプロパティで指定した認証情報が利用されます。また，アプリケーション認証の場合は，アプリケーションで指定した認証情報が利用されます。

5. QueueSession オブジェクトとの関連づけ

   Reliable Messaging はアプリケーションに返却する QueueSession オブジェクトを生成し，3.または 4.で取得した JDBC コネクションと関連づけます。

6. QueueSession オブジェクトの返却

   Reliable Messaging はアプリケーションに QueueSession オブジェクトを返します。

2.6.6 トランザクション制御

Reliable Messaging のアプリケーションはトランザクションマネジャでのトランザクション，およびローカルトランザクションを利用できます。ただし，トランザクションマネジャでのトランザクションとローカルトランザクションとを並行して利用することはできません。

例えば，トランザクションマネジャが制御しているときに次に示す状態の QueueSession オブジェクトを，トランザクション属性が NotSupported のメソッド内で利用しようとした場合，ローカルトランザクションを並行して利用することになるため例外が発生します。

- QueueConnection.createQueueSession()メソッドの transacted 引数に true を指定した QueueSession オブジェクト
- QueueConnection.createQueueSession()メソッドの acknowledgeMode 引数に CLIENT_ACKNOWLEDGE を指定した QueueSession オブジェクト

(1) ローカルトランザクションの利用

キューセッションでのローカルトランザクションを利用する場合，
QueueConnection.createQueueSession()メソッドの transacted 引数に true を指定します。

ローカルトランザクションの場合，Reliable Messaging のメッセージ送受信だけがトランザクションの対象になります。他製品が提供するリソースに対する操作は対象になりません。

(2) トランザクションマネジャでのトランザクションの利用

Reliable Messaging では，トランザクションマネジャでのトランザクションを利用できます。次に示す場合，トランザクションマネジャでのトランザクションが発生します。

- Application Server の BMT 使用時：javax.transaction.UserTransaction.begin()発行
- Application Server の CMT 使用時：Enterprise Bean のビジネスメソッド呼び出し

トランザクションマネジャでのトランザクションの場合，Reliable Messaging のメッセージ送受信だけでなく，他製品が提供するリソースに対する操作を含めたトランザクションが使用できます。

トランザクションマネジャでのトランザクションを使用する場合，Connector 属性ファイルの<transaction-support>タグに LocalTransaction または XATransaction を指定してください。ただし，LocalTransaction を指定した場合，他製品が提供するリソースに対する操作を含めたトランザクションを使用できません。NoTransaction を指定した場合，トランザクションマネジャでのトランザクションを使用できません。Connector 属性ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

(3) トランザクションとメッセージ送信

メッセージ送信時の QueueSender.send()メソッドの動作について，トランザクションマネジャによってトランザクションが制御されていない場合と制御されている場合に分けて説明します。

- トランザクションマネジャによって制御されていない場合

  トランザクションマネジャによってトランザクションが制御されていない場合，QueueSender.send()メソッドの動作は QueueConnection.createQueueSession()メソッドの transacted 引数指定値によって異なります。

  トランザクションマネジャによってトランザクションが制御されていない場合の send()メソッドの動作を次の表に示します。

表 2-22 トランザクションマネジャによってトランザクションが制御されていない場合の send()メソッドの動作

項番	transacted 引数	動作内容
1	true	• QueueSender.send()メソッドを発行するとキューセッションでのローカルトランザクションが開始されます。QueueSession.commit()メソッドを発行したときにメッセージ送信がコミットされます。 • ローカルトランザクションが未決着の状態で，トランザクションマネジャでのトランザクションを開始すると例外が発生します。ローカルトランザクションが開始されていない状態であるなら，トランザクションを開始できます。
2	false	QueueSender.send()メソッドを発行すると，メッセージを正常に送信完了したときに自動的にコミットされます。

注

コミット済みのメッセージを再度送信する場合，プロパティが再設定されて送信されます。

- トランザクションマネジャによって制御されている場合

  トランザクションマネジャによってトランザクションが制御されている場合，QueueSender.send()メソッドでのメッセージ送信はトランザクションマネジャでのトランザクションに含まれます。QueueSession.commit()または rollback()メソッドを発行すると例外が発生します。

(4) トランザクションとメッセージ受信

メッセージの受信方式には，同期受信および非同期受信があります。

- 同期受信

  アプリケーションが QueueReceiver.receive()または receiveNoWait()メソッドを発行したタイミングでメッセージを受信します。

- 非同期受信

  Message-driven Bean がキューを監視し，キューにメッセージが到着したタイミングでメッセージを受信します。

  onMessage()メソッドの message 引数によってメッセージを受信できます。この場合は，QueueConnection.createQueueSession()メソッドの acknowledgeMode 引数に Session.AUTO_ACKNOWLEDGE を指定したときの QueueReceiver.receive()または receiveNoWait()メソッドの動作と同じ動作内容になります。Message-driven Bean との連携については，「2.6.3　Message-driven Bean との連携」を参照してください。

以降では，同期受信についてトランザクションマネジャによってトランザクションが制御されていない場合と制御されている場合に分けて説明します。

- トランザクションマネジャによって制御されていない場合

  トランザクションマネジャによってトランザクションが制御されていない場合，receive()および receiveNoWait()メソッドの動作は QueueConnection.createQueueSession()メソッドの transacted および acknowledgeMode 引数指定値によって異なります。

  トランザクションマネジャによってトランザクションが制御されていない場合の receive()および receiveNoWait()メソッドの動作を次の表に示します。

表 2-23　トランザクションマネジャによってトランザクションが制御されていない場合の receive()および receiveNoWait()メソッドの動作

項番	transacted 引数	acknowledgeMode 引数	動作内容
1	true	−	• QueueReceiver.receive()または receiveNoWait()メソッドを発行するとキューセッションでのローカルトランザクションが開始されます。QueueSession.commit()メソッドを発行したときにメッセージ受信が承認されます。 • ローカルトランザクションが未決着の状態で，トランザクションマネジャでのトランザクションを開始すると例外が発生します。ローカルトランザクションが開始されていない状態であるなら，トランザクションマネジャでのトランザクションを開始できます。
2	false	AUTO_ACKNOWLEDGE	メッセージを正常に受信完了したとき，メッセージは自動的に承認されます。
3		DUPS_OK_ACKNOWLEDGE	
4		CLIENT_ACKNOWLEDGE	Queue.Receiver()および receiveNoWait()メソッドでのメッセージの受信はローカルトランザクションに含まれます。

項番	transacted 引数	acknowledgeMode 引数	動作内容
			Message.acknowledge()メソッドを発行すると，同一のセッションで以前に受信して承認していないすべてのメッセージおよび現在のメッセージが承認されます。 QueueSession.recover()メソッドを発行すると，配送済みのメッセージが再度配送できるようになります。 Message.acknowledge()または QueueSession.recover()メソッドを発行しないかぎり，トランザクションマネジャでのトランザクションを開始できません。

（凡例）

－：該当しません。

- トランザクションマネジャによって制御されている場合

  トランザクションマネジャによってトランザクションが制御されている場合，QueueReceiver.receive()および receiveNoWait()メソッドの動作は QueueConnection.createQueueSession()メソッドの transacted および acknowledgeMode 引数指定値によって異なります。

  トランザクションマネジャによってトランザクションが制御されている場合の receive()および receiveNoWait()メソッドの動作を次の表に示します。

表 2-24　トランザクションマネジャによってトランザクションが制御されている場合の receive()および receiveNoWait()メソッドの動作

項番	transacted 引数	acknowledgeMode 引数	動作内容
1	true	－	QueueReceiver.receive()または receiveNoWait()メソッドでのメッセージの受信はトランザクションマネジャでのトランザクションに含まれます。 QueueSession.commit()または rollback()メソッドを発行すると例外が発生します。
2	false	AUTO_ACKNOWLEDGE	
3		DUPS_OK_ACKNOWLEDGE	
4		CLIENT_ACKNOWLEDGE	QueueReceiver.receive()または receiveNoWait()メソッドでのメッセージの受信はトランザクションマネジャでのトランザクションに含まれます。 Message.acknowledge()または QueueSession.recover()メソッドを発行すると例外が発生します。

（凡例）

－：該当しません。

(5)　Application Server のアウトプロセストランザクションサービスの利用

Application Server のアウトプロセストランザクションサービスを使用し，XID 再利用で最適化を行った場合，グローバルトランザクションで異なる二つのリソースアダプタ（別々にデプロイされた二つの

Reliable Messaging，Reliable Messaging と DB Connector など）から同一の DB に接続するコネクションを使用しないでください。

この構成にすると，リソースマネジャでエラーが発生する，または応答が返らないおそれがあります。XID 再利用による最適化については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

2.6.7　アプリケーション作成時の注意事項

Reliable Messaging のアプリケーションを作成するときの注意事項を次に示します。

なお，非永続版リソースアダプタの場合は，(2)，(6)，(8)，および(9)の注意事項には該当しません。

(1)　スレッド間でのオブジェクトの共有

QueueSession オブジェクトおよび QueueSession オブジェクトを使用して生成したオブジェクトを複数のスレッドやアプリケーションで共有できません。共有した場合の動作は保証されません。

(2)　JDBC コネクション

(a)　JDBC コネクションの消費

アプリケーションのリソースアダプタリファレンスの解決時に，サーブレットおよび JSP や Enterprise Bean の属性ファイルの<res-sharing-scope>タグに"Shareable"が指定されていない場合，Reliable Messaging では QueueSession オブジェクトが一つ生成されるたびに，JDBC コネクションが一つ消費されます。このため，アプリケーションで使用しなくなった QueueSession オブジェクトについては，必ず QueueSession.close()メソッドを発行してリソースを解放するようにしてください。属性ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

ただし，Application Server のコネクションプーリング機能を使用する場合は QueueSession オブジェクト生成時に JDBC コネクションは再利用されます。QueueSession.close()メソッドで解放しても JDBC コネクションは継続して使用されます。

Message-driven Bean を使用する場合は，メッセージが配信されるとき，Message-driven Bean のインスタンスごとに一つの JDBC コネクションが消費されます。メッセージの配信が終了すると，リソースは解放されます。コネクションプーリング機能を使用している場合は，JDBC コネクションは解放されないでプールされます。

(b)　トランザクションマネジャ内での JDBC コネクションの利用

トランザクションマネジャでのトランザクション内では，最初にトランザクションに参加した JDBC コネクションが常に利用されます。つまり，2 番目以降の JDBC コネクションに対する処理も，最初の JDBC コネクションを利用して行われるので注意が必要です。

(3) コネクションシェアリング時の指定

アプリケーションのリソースアダプタリファレンスの解決時に，サーブレットおよびJSPやEnterprise Beanの属性ファイルの<res-sharing-scope>タグに"Shareable"を指定し，かつ論理コネクション（java.sql.Connection，QueueSessionオブジェクト）を複数生成する場合は，QueueSessionオブジェクト生成時のtransacted引数にfalseを，acknowledgeMode引数にAUTO_ACKNOWLEDGEを指定してください。属性ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

異なる引数を指定してQueueSessionオブジェクトを生成した場合，コネクションシェアリングが動作する契機で例外が発生します。

注意事項

アプリケーションがMessage-driven Beanの場合，アプリケーション内で1つだけ論理コネクションを生成する場合でも，コネクションシェアリングが動作する可能性があります。

Message-driven Beanではメッセージを配信する契機で論理コネクション（配信用コネクション）を1つ生成します。

この配信用コネクションとMessage-driven Bean内で生成される論理コネクションでシェアリングが行なわれる可能性があります。

配信用コネクションとコネクションシェアリングが行なわれる条件については，「付録 G.1(1)　MDBの場合」を参照してください。

(4) Message-driven Beanの監視対象のキュー

Message-driven Beanで監視するキューがシリアル取り出し属性の場合は，Message-driven Beanの多重度を2以上にしても性能面での効果はありません。多重度を上げてメッセージを配送する場合は監視するキューをパラレル取り出し属性にすることをお勧めします。

また，Message-driven Beanで監視するキューがシリアル取り出し属性の場合は，同一のキューに対するQueueReceiver.receive()メソッドを発行しないでください。発行するとデッドロックが発生します。

(5) Message-driven Bean使用時のトランザクションの指定

Message-driven Beanを使用する場合，Connector属性ファイルの<transaction-support>タグにXATransactionまたはLocalTransactionを指定してください。XATransactionまたはLocalTransaction以外を指定するとMessage-driven Beanアプリケーションのデプロイに失敗します。Connector属性ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

(6) Message-driven Bean を BMT で使用するときの DB Connector のステートメントプールの指定

Message-driven Bean を BMT で使用する場合，接続先が HiRDB で，かつ XATransaction モードのときは，DB Connector のステートメントプールを使用しないでください。DB Connector のステートメントプール使用した場合，エラーメッセージとともに SQLException 例外が発生することがあります。

(7) Message-driven Bean 使用時のコネクションプーリング数の指定

Message-driven Bean を使用する場合，コネクションプーリング数が Message-driven Bean の多重度よりも少ないと，性能が劣化することがあります。コネクションプーリング数は，Message-driven Bean の多重度よりも多くすることをお勧めします。

(8) 共用キュー

(a) 共用キューのアクセス順序

一つのトランザクション内で複数の共用キューにアクセスする場合，各アプリケーションでキューにアクセスする順序を統一する必要があります。異なる順序で共用キューにアクセスしようとすると，デッドロックが発生するおそれがあります。

(b) 受信用共用キューの利用

一つのトランザクション内で，同一の受信用共用キューに対する受信と送信を混在させないようにしてください。

(9) メッセージのサイズ制限

キュー間転送を行う場合，SOAP 通信基盤の SOAP アタッチメントに関する仕様によって，転送するメッセージのサイズに制限が掛かる条件があります。制限の内容および制限が掛かる条件については SOAP 通信基盤のドキュメントを参照してください。

(10) 参照渡し方式利用時の JMS メッセージの再利用

参照渡し方式を利用する場合，1 度でも登録，取り出し，または参照した JMS メッセージに対しては，次の操作を実行しないでください。

- メッセージの登録
  - 登録したメッセージの再登録
  - 登録したメッセージの更新メソッドおよび acknowledge()メソッドの呼び出し
  - 登録したメッセージが BytesMessage の場合，ペイロードの参照メソッドの呼び出し
- メッセージの取り出し
  - 取り出したメッセージの登録

- 取り出したメッセージの更新メソッドの呼び出し
- 取り出しをコミットしたメッセージの acknowledge()メソッドの呼び出し
- 取り出しをロールバックしたメッセージのすべてのメソッドの呼び出し

- メッセージの参照
  - 参照したメッセージの登録
  - 参照したメッセージの更新メソッドおよび acknowledge()メソッドの呼び出し

参照渡し方式利用時の JMS メッセージのメソッドの呼び出し可否を次の表に示します。

表 2-25　JMS メッセージのメソッドの呼び出し可否

JMS メッセージのメソッドを呼び出すタイミング		JMS メッセージのメソッドの呼び出し可否		
		参照メソッド	更新メソッド	acknowledge()メソッド
登録後	send()メソッドの発行後	○※	×	×
	commit()メソッドの発行後	○※	×	×
	rollback()メソッドの発行後	○※	×	×
取り出し後	receive()メソッドの発行または MDB 配信後	○	×	○
	commit()メソッドまたは acknowledge()メソッドの発行後	○	×	×
	rollback()メソッドまたは recover()メソッドの発行後	×	×	×
参照後		○	×	×

（凡例）

○：呼び出せます。

×：呼び出せません。

注※

BytesMessage の場合，ペイロードの参照メソッドは呼び出せません。

登録，取り出し，または参照した JMS メッセージに対して，呼び出せないメソッドを呼び出した場合の動作は保証できません。呼び出せないメソッドを呼び出した場合，ほかのセッションで取り出したメッセージの受信を承認（acknowledge）してしまう，メッセージデータが不正になる，例外が発生するなどの状態になることがあります。

登録に失敗したメッセージを再び登録したい場合は， 新たに JMS メッセージを作成し，作成したメッセージに，登録に失敗したメッセージに設定していた情報を設定し直して，登録してください。

(11) 参照渡し方式利用時の JMS メッセージの排他

参照渡し方式を利用する場合，JMS メッセージに対して synchronized による排他を取らないでください。JMS メッセージで排他を取得した場合，デッドロックが発生するおそれがあります。

(12) JMS インターフェースの範囲

Reliable Messaging は JMS1.0.2 で提供されているインタフェースの範囲で利用してください。JMS1.1 で追加されたインタフェースを使用すると予期しない例外が発生するおそれがあります。

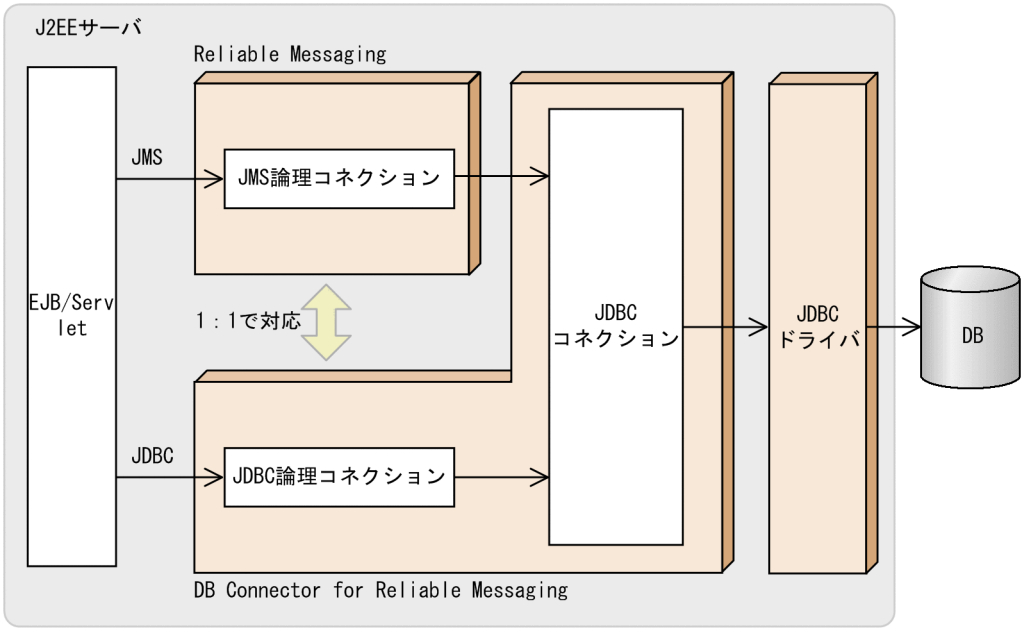
2.7 DB Connector for Reliable Messaging の機能

Reliable Messaging は，システム状態やユーザから送られてきたメッセージを DB 上のテーブルに永続化します。その場合，DB へのアクセスには DB Connector for Reliable Messaging を利用します。

2.7.1 Reliable Messaging と DB Connector for Reliable Messaging の関係

Reliable Messaging と DB Connector for Reliable Messaging の関係を次の図に示します。

図 2-28 Reliable Messaging と DB Connector for Reliable Messaging の関係



DB Connector for Reliable Messaging はデータベースにアクセスするためのリソースアダプタです。Reliable Messaging は，DB Connector for Reliable Messaging の JDBC コネクションを取得し，JDBC インタフェースを利用してデータベースにアクセスします。

Reliable Messaging と DB Connector for Reliable Messaging は 1：1 に対応づける必要があります。そのため，Reliable Messaging の RMLinkedDBConnectorName プロパティに DB Connector for Reliable Messaging の表示名を指定します。同様に，DB Connector for Reliable Messaging から Reliable Messaging への対応づけも必要です。

なお，DB Connector for Reliable Messaging と連携できるのは，永続版リソースアダプタの Reliable Messaging だけです。

(1) リソースアダプタ間の依存関係

Reliable Messaging は，DB Connector for Reliable Messaging の JDBC コネクションを利用して DB にアクセスします。したがって，Reliable Messaging の稼働時には，DB Connector for Reliable Messaging も稼働している必要があります。

Component Container は，Reliable Messaging の開始時に，連携する DB Connector for Reliable Messaging が稼働中かどうかをチェックし，稼働していない場合はエラーを返します。また，DB Connector for Reliable Messaging の停止時には，連携する Reliable Messaging が停止済みかどうかをチェックし，停止していない場合はエラーを返します。

リソースアダプタの状態と開始または停止の依存関係を次の表に示します。

表 2-26 DB Connector for Reliable Messaging の状態と Reliable Messaging の開始または停止の関係

DB Connector for Reliable Messaging の状態	連携する Reliable Messaging の動作
稼働中	開始できる
	停止できる
停止中	開始できない

表 2-27 Reliable Messaging の状態と DB Connector for Reliable Messaging の開始または停止の関係

Reliable Messaging の状態	連携する DB Connector for Reliable Messaging の動作
稼働中	停止できない
停止中	開始できる
	停止できる

リソースアダプタの開始時には，Component Container が Reliable Messaging と DB Connector for Reliable Messaging の対応づけをチェックします。Reliable Messaging の開始時に DB Connector for Reliable Messaging と正しく対応づけられなかった場合は，Reliable Messaging は開始できません。

(2) リソースアダプタの設定

リソースアダプタの設定情報のうち，コネクションプールに関する情報や認証情報は，Reliable Messaging のプロパティに設定します。また，DB への接続情報は DB Connector for Reliable Messaging のプロパティに設定します。

リソースアダプタの設定情報の分類を次の表に示します。

表 2-28　リソースアダプタの設定情報の分類

設定情報		DB Connector for Reliable Messaging	Reliable Messaging
コンフィグレーションプロパティ	Reliable Messaging 固有の情報	−	○
	DB Connector for Reliable Messaging 固有の情報（ステートメントプールなど）	○	−
	ログ取得情報	○	○
	連携先リソースアダプタ名	○	○
認証情報（ユーザ名，パスワード）※1※2		○	○
コネクションプールサイズ※2		○	○
コネクション障害検知※2		○	○
コネクション取得リトライ※2		○	○
トランザクションサポートレベル※2		○	○
データベース接続定義		○	−

（凡例）

○：設定が必要です。

−：設定項目がありません。

注※1

コンテナ認証時の DB への接続ユーザ名およびパスワードとして使用されます。

注※2

DB Connector for Reliable Messaging と Reliable Messaging で同じ値を設定してください。

2.7.2　DB Connector for Reliable Messaging の機能一覧

DB Connector の機能の中には，DB Connector for Reliable Messaging で使用できる機能と使用できない機能があります。DB Connector for Reliable Messaging で使用できる DB Connector の機能を次の表に示します。なお，各機能の詳細については，「2.7.3　DB Connector for Reliable Messaging 連携時のコネクションと SQL の運用」，およびマニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」を参照してください。

表 2-29　DB Connector for Reliable Messaging で使用できる DB Connector の機能

項番	DB Connector の機能	DB Connector for Reliable Messaging での使用可否
1	コネクションプーリング	○
2	コネクションプールのウォーミングアップ	○
3	コネクションシェアリング・アソシエーション	○

項番	DB Connector の機能	DB Connector for Reliable Messaging での使用可否
4	ステートメントプーリング	○
5	ステートメントキャンセル	○
6	DataSource オブジェクトのキャッシング	○
7	DB Connector のコンテナ管理でのサインオンの最適化	○
8	コネクションの障害検知	○
9	コネクションの障害検知のタイムアウト	○
10	コネクション枯渇時のコネクション取得待ち	○
11	コネクションの取得リトライ	○
12	コネクションプールの情報表示	○
13	コネクションプールのクリア	○
14	コネクションの自動クローズ	○
15	コネクションスイーパ	○
16	接続テスト	○
17	コネクション数調節機能	○
18	コネクションプールの情報表示	○
19	コネクションプールの一時停止	×
20	コネクションプールの再開	×
21	J2EE リソースのオプショナル名機能	○
22	コネクション ID の取得	○
23	障害調査用 SQL の出力	◎
24	軽微な障害時のコネクション再利用	○

(凡例)

◎：必ず有効になります。

○：使用できます。

×：使用できません。

2.7.3 DB Connector for Reliable Messaging 連携時のコネクションと SQL の運用

Reliable Messaging と DB Connector for Reliable Messaging が連携するときのコネクションと SQL の運用について説明します。

(1) コネクションの共有

DB Connector for Reliable Messaging と Reliable Messaging は，共に JDBC コネクションを利用します。同一トランザクション内で JDBC と JMS のアクセスを行う場合は，JDBC コネクションを共有できます。

DB Connector for Reliable Messaging と Reliable Messaging の間でコネクションを共有するには，次の条件を満たす必要があります。

- DB Connector for Reliable Messaging と Reliable Messaging が使用する DB が同一であること。
- Reliable Messaging の RMAssociateJDBCFlag プロパティに true が指定されていること。
- アプリケーションのリソースアダプタリファレンスの解決時に，データソースとキューコネクションファクトリの Session Bean 属性ファイル，Entity Bean 属性ファイル，WAR 属性ファイル，または MessageDrivenBean 属性ファイルの<res-sharing-scope>タグに"Shareable"を指定していること（各属性ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください）。
- 認証情報（ユーザ名，パスワード）が同一であること。
- 認証方法（コンテナ認証またはアプリケーション認証）が同一であること。
- Reliable Messaging の QueueSession が，すべて AutoAcknowledge モードであること（AutoAcknowledge モード以外の QueueSession との間でコネクションを共有しようとすると，論理コネクション（java.sql.Connection, QueueSession）の取得に失敗します）。

コネクション共有の条件を満たさない場合は，別の JDBC コネクションが取得されます。この場合も，トランザクションマネジャでのトランザクション内では最初にトランザクションに参加した JDBC コネクションが常に利用されます。つまり，2 番目以降の JDBC コネクションに対する処理も，最初の JDBC コネクションを利用して行われるので注意が必要です。

なお，コネクション共有を利用するかどうかに関係なく，JDBC コネクションは Reliable Messaging のコネクションとしてプール管理されます。一方，DB Connector for Reliable Messaging のコネクションはプール管理されません。コネクション取得時には Reliable Messaging のコネクションプールから JDBC コネクションを取得します。

(2) ステートメントキャンセル

DB Connector for Reliable Messaging と連携する場合，実行中の SQL 処理が返ってこない状態でトランザクションのタイムアウトが発生すると，ステートメントがキャンセルされます。

Reliable Messaging のステートメントキャンセルで注意する点を次に示します。

- ステートメントキャンセルは，ユーザが JDBC を利用して行う処理に対してだけ有効です。ユーザが JMS，または JDBC と JMS との間でコネクションシェアリングを利用する場合，ステートメントキャンセルは JDBC の処理に対してだけ有効となります。このとき，JMS を通して行われる処理に対しては，ステートメントがキャンセルされません。

(3) 障害調査用 SQL の出力

デッドロックやスローダウンなどの障害が発生した場合，発行した SQL が障害の要因となった可能性があります。そこで，発行した SQL をログに出力することによって，障害要因の解析が容易になります。ログに出力される SQL の情報を障害調査用 SQL と呼びます。障害調査用 SQL の詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」を参照してください。

Reliable Messaging の障害調査用 SQL の出力で注意する点を次に示します。

- 障害調査用 SQL の出力は，ユーザが JDBC を利用して行う処理に対してだけ有効です。ただし，ユーザがリソースアダプタ管理のトランザクション（リソース固有の API によって，ユーザが直接トランザクションを管理する方法）で JDBC と JMS との間でコネクションシェアリングを利用する場合，JMS を通して行われる処理に対しての障害調査用 SQL が出力されることがあります。
- ステートメントキャンセルと障害調査用 SQL の出力は，まず障害調査用 SQL の出力が実行され，そのあとにステートメントキャンセルが実行されます。
- 障害調査用 SQL は，DB Connector for Reliable Messaging の稼働ログ，および性能解析トレースに出力されます。

  DB Connector for Reliable Messaging の稼働ログ

  <Application Server のログディレクトリ>¥connectors

  性能解析トレース

  <Application Server のインストールディレクトリ>¥PRF¥spool¥utt¥prf¥PRF_ID¥dcopltrc

(4) 軽微な障害時のコネクション再利用

コネクション利用時に発生したエラーが，テーブルの一意性に違反した SQL の発行など軽微な障害の場合，コネクションを再利用します。

Reliable Messaging の軽微な障害時のコネクション再利用で注意する点を次に示します。

- 軽微な障害時のコネクション再利用は，ローカルトランザクションの場合だけ有効です。トランザクションを使用していない，またはトランザクションのサポートレベルが XATransaction の場合，コネクションは破棄されます。

(5) コネクション ID の取得

DB Connector for Reliable Messaging と連携した場合，接続先 DB のコネクション ID を取得できます。コネクション ID とは，DB との接続に使用しているコネクションを一意に識別するための接続情報です。コネクション ID の取得の詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」を参照してください。

Reliable Messaging のコネクション ID の取得で注意する点を次に示します。

- Reliable Messaging は，一つのトランザクション内の場合，アプリケーションは同じ DB Connector for Reliable Messaging のコネクションを利用します。そのため，一つのトランザクション内で複数のコネクションを利用しているときに cjlistpool コマンドを実行すると，同じコネクション ID が二つあるように見えます。

(6) コネクションの接続テスト

Component Container が提供する Reliable Messaging の接続テストを行うには，連携する DB Connector for Reliable Messaging が稼働中であることが前提となります。また，DB Connector for Reliable Messaging の接続テストでは，Reliable Messaging と DB Connector for Reliable Messaging の両方が稼働中であることを前提とします。

接続テストに失敗した場合は，メッセージの内容に応じて要因を取り除いてください。

2.7.4 DB Connector for Reliable Messaging 連携時の注意事項

- Reliable Messaging が管理状態または閉塞状態のときには，DB Connector for Reliable Messaging を利用して，トランザクションマネジャでのトランザクションの利用はできません。
- DB だけを利用する場合は，通常の DB Connector を利用してください。DB Connector for Reliable Messaging は，コネクション共有機能のオーバヘッドがあるため，性能が劣化します。
- DB Connector for Reliable Messaging を利用している場合，java.sql.Connection の setAutoCommit()メソッドは利用できません。したがって，JDBC 固有のトランザクション制御はできません。なお，コネクション共有を行い，トランザクションマネジャのトランザクションを利用していない場合，JDBC のコネクションは AutoCommit=true となります。また，JMS の QueueSession は AUTO_ACKNOWLEDGE モードとなります。
- Oracle を利用して JDBC と JMS のアクセスを行う場合は，コネクション共有機能を使うことが前提となります。Reliable Messaging では，同一 DB に対して，異なるリソースアダプタとの間の 2 相コミットはできません。
- java.sql.Connection からの生成物（例：java.sql.Statement）を，トランザクションの範囲を超えて使用することはできません。

3 システム構築

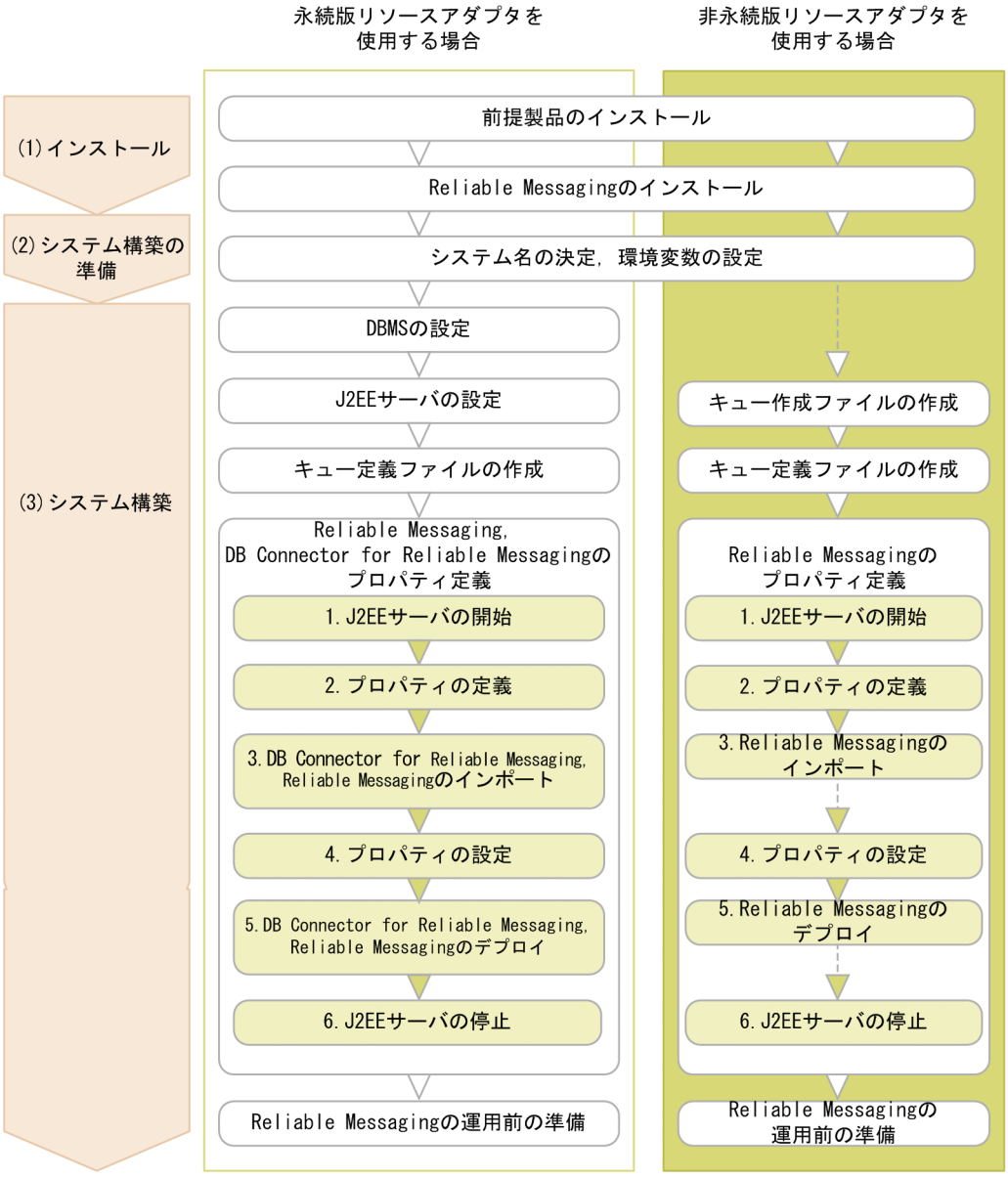
この章では Reliable Messaging のシステムを構築する手順について説明します。

なお，Reliable Messaging のシステム構築は，永続版リソースアダプタを使用する場合と非永続版リソースアダプタを使用する場合とで手順が異なります。

3.1.1　Reliable Messaging のシステム構築の流れ

Reliable Messaging のシステム構築の流れを次の図に示します。

図 3-1　Reliable Messaging のシステム構築の流れ



図中の番号について説明します。

(1)　インストール

Application Server や DBMS などの前提製品をインストールしたあと，Reliable Messaging をアプリケーションサーバシステムが動作するマシンにインストールします。Windows の場合はインストーラを，

UNIX の場合は PP インストーラを使用してインストールしてください。また，インストール後に格納されるファイルについては，「3.2.2　Reliable Messaging をインストールすると格納されるファイル」を参照してください。

(2)　システム構築の準備

システム名として，RMSystemName プロパティに指定する値（先頭が英字の 1～3 文字の大文字英字または数字）を決定します。また，Reliable Messaging が使用する環境変数を設定します。

システム名の決定については，「3.3.1　Reliable Messaging のシステム名の決定」を，環境変数の設定については，「3.3.2　環境変数の設定」を参照してください。

(3)　システム構築

システム構築の手順は，永続版リソースアダプタを使用する場合と，非永続版リソースアダプタを使用する場合で異なります。

永続版リソースアダプタを使用する場合は，「3.4　Reliable Messaging のシステム構築（永続版リソースアダプタの場合）」を，非永続版リソースアダプタを使用する場合は，「3.5　Reliable Messaging のシステム構築（非永続版リソースアダプタの場合）」を参照してください。

なお，キュー定義ファイルの作成以降の作業については，コマンドでシステムを構築しています。コマンド以外にも，GUI（プラグイン）や Management Server を利用してシステムを構築することもできます。GUI（プラグイン）または Management Server を利用したシステム構築方法については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」およびマニュアル「アプリケーションサーバ アプリケーション開発ガイド」を参照してください。

また，Reliable Messaging および DB Connector for Reliable Messaging のプロパティは，ユーザが環境に合わせて定義します。プロパティ定義の流れを次に示します。

1. J2EE サーバの開始

   Application Server を開始します。また，アプリケーションの要件に合わせて，必要な Application Server のサービスを開始してください。

2. プロパティの定義

   DB Connector for Reliable Messaging および Reliable Messaging のプロパティを定義します。Reliable Messaging で提供する Connector 属性ファイルのテンプレートを任意のディレクトリにコピーして，コピーしたファイルを編集してください。

   認証情報（User および Password）には Reliable Messaging が HiRDB または Oracle にアクセスするために利用する接続ユーザのユーザ名，パスワードを指定してください。

   なお，非永続版リソースアダプタでは認証情報（User および Password）は，指定しても無視されます。

3. DB Connector for Reliable Messaging および Reliable Messaging のインポート

   サーバ管理コマンドを使用して，リソースアダプタとして DB Connector for Reliable Messaging および Reliable Messaging をインポートします。

4. プロパティの設定

   サーバ管理コマンドを使用して，DB Connector for Reliable Messaging および Reliable Messaging のプロパティを設定します。

5. DB Connector for Reliable Messaging および Reliable Messaging のデプロイ

   サーバ管理コマンドを使用して，DB Connector for Reliable Messaging および Reliable Messaging をデプロイします。

6. J2EE サーバの停止

   必要に応じて J2EE サーバ（Component Container）を停止します。引き続き Reliable Messaging を開始する場合は，J2EE サーバを停止しないで，永続版リソースアダプタを使用するときは「4.1 Reliable Messaging とアプリケーションの開始と停止（永続版リソースアダプタの場合）」を，非永続版リソースアダプタを使用するときは「5.1　Reliable Messaging とアプリケーションの開始と停止（非永続版リソースアダプタの場合）」を参照してください。

なお，Reliable Messaging および DB Connector for Reliable Messaging のシステムを構築するサンプル手順については，「付録 C.1　環境構築のサンプル手順」を参照してください。

3.2　Reliable Messaging のインストール

Reliable Messaging は，Windows の場合はインストーラを，UNIX の場合は PP インストーラを使用してインストールします。インストールは，インストーラ，または PP インストーラの指示に従ってください。なお，Reliable Messaging のインストールは，前提製品をインストールしたあとに実施します。前提製品のインストールについては，「3.2.1　前提製品のインストール」を参照してください。

3.2.1　前提製品のインストール

Reliable Messaging をインストールする前に，Application Server や DBMS（永続版リソースアダプタを使用する場合だけ）などの前提製品をインストールする必要があります。前提製品のインストールの方法については，この章で説明する内容に加えて各製品が提供するマニュアルを参照してください。

(1)　Application Server のインストール

Reliable Messaging は J2EE サーバ上で動作します。Reliable Messaging が動作するマシンに，次に示す製品をインストールしてください。

- Component Container
- Developer's Kit for Java
- TPBroker
- Performance Tracer
  PRF トレースを取得する場合だけ必要です。

インストールしたあと，アプリケーションで実行する業務に合わせてアプリケーションサーバシステムを構築してください。構築方法については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

(2)　DBMS のインストール

永続版リソースアダプタを使用する場合，DBMS として HiRDB または Oracle をインストールしてください。

(3)　DB クライアントのインストール

永続版リソースアダプタを使用する場合で，DBMS として HiRDB を使用するとき，次に示す製品をインストールしてください。

- HiRDB/Run Time または HiRDB/Developer's Kit
  詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
- HiRDB SQL Executer

詳細については，HiRDB SQL Executer のドキュメントを参照してください。

3.2.2 Reliable Messaging をインストールすると格納されるファイル

Reliable Messaging をインストールすると，次に示すファイルが格納されます。Reliable Messaging のインストール内容を次の表に示します。

表 3-1 Reliable Messaging のインストール内容

項番	名称	ディレクトリ	ファイル名
1	永続版リソースアダプタ	%HRMDIR%¥lib	reliablemessaging.rar
2	非永続版リソースアダプタ	%HRMDIR%¥lib	reliablemessagingNP.rar
3	ライブラリ	%HRMDIR%¥lib	reliablemessaging-api.jar
4	クラスライブラリ	%HRMDIR%¥lib	hrmcommand.jar
5	EAR	%HRMDIR%¥lib	hrmqueue-transmit.ear
6	コマンド	%HRMDIR%¥bin	Windows の場合 hrm*.bat UNIX の場合 hrm*
7	クライアント定義ファイル	%HRMDIR%¥conf	c4webcl.properties
8	テーブル作成用 SQL ファイル	%HRMDIR%¥sql	createtableshirdb.sql createtablesoracle.sql
9	テーブル削除用 SQL ファイル	%HRMDIR%¥sql	deletetableshirdb.sql deletetablesoracle.sql
10	共用キューバージョンアップ用 SQL	%HRMDIR%¥sql	shqupdate_V1toV2.sql
11	Connector 属性ファイルのテンプレート	%HRMDIR%¥conf	rm_prop.xml rmnp_prop.xml
12	クライアントログファイル	%HRMDIR%¥logs¥cmd¥clt	－
13	サンプルアプリケーション	%HRMDIR%¥samples¥SessionBean1	JMSSample1.java JMSSample1Client.java JMSSample1EJB.java JMSSample1Home.java QueueConfig.properties QueueConfigNP.properties QueueMake.properties config.xml

項番	名称	ディレクトリ	ファイル名
			Windows の場合 compileBean.bat compileClient.bat testClient.bat deployApp.bat unDeployApp.bat UNIX の場合 compileBean compileClient testClient deployApp unDeployApp
14		%HRMDIR%¥samples¥SessionBean1¥DD_EJB¥META-INF	ejb-jar.xml
15		%HRMDIR%¥samples¥SessionBean1¥DD_APP¥META-INF	application.xml
16		%HRMDIR%¥samples¥SessionBean2¥Receive	JMSSample2Receive.java JMSSample2ReceiveClient.java JMSSample2ReceiveEJB.java JMSSample2ReceiveHome.java QueueConfig.properties config.xml Windows の場合 compileBean.bat compileClient.bat testReceiveClient.bat deployApp.bat unDeployApp.bat UNIX の場合 compileBean compileClient testReceiveClient deployApp unDeployApp
17		%HRMDIR%¥samples¥SessionBean2¥Receive¥DD_EJB¥META-INF	ejb-jar.xml
18		%HRMDIR%¥samples¥SessionBean2¥Receive¥DD_APP¥META-INF	application.xml

項番	名称	ディレクトリ	ファイル名
19		%HRMDIR%¥samples¥ SessionBean2¥Send	JMSSample2Send.java JMSSample2SendClient.java JMSSample2SendEJB.java JMSSample2SendHome.java QueueConfig.properties config.xml Windows の場合 compileBean.bat compileClient.bat testSendClient.bat deployApp.bat unDeployApp.bat UNIX の場合 compileBean compileClient testSendClient deployApp unDeployApp
20		%HRMDIR%¥samples¥ SessionBean2¥Send¥DD_EJB¥MET A-INF	ejb-jar.xml
21		%HRMDIR%¥samples¥ SessionBean2¥Send¥DD_APP¥MET A-INF	application.xml

(凡例)

－：インストール直後はファイルがありません。

注

%HRMDIR%は Reliable Messaging がインストールされたディレクトリを表します。

3.3　システム構築の準備

システムを構築する前の準備について説明します。

3.3.1　Reliable Messaging のシステム名の決定

システム名として，システム間で一意となる値（先頭が英字の 1～3 文字の大文字英字または数字）を決定します。決定した値は環境変数，RMSystemName プロパティなどに指定します。

3.3.2　環境変数の設定

Reliable Messaging が動作するマシンで，次に示す環境変数を設定してください。

- HRMDIR
  Reliable Messaging がインストールされたディレクトリのパスを指定します。
- HRM_SYSTEM_NAME
  Reliable Messaging が提供するコマンドを実行する際に操作の対象となるシステム名を指定する環境変数です。「3.3.1　Reliable Messaging のシステム名の決定」で決定した値を指定します。
- HRM_CMD_HOST
  CORBA ネーミングサービスのホスト名です。Reliable Messaging のコマンドが CORBA ネーミングサービスを使用するときのホスト名または IP アドレスを指定します。
  CORBA ネーミングサービスは JNDI の機能を実現します。ユーザが Reliable Messaging のコマンドを使用したり，アプリケーションがキューを lookup するときに動作している必要があります。
  IP アドレスを二つ以上持つマシン上で明示的にネーミングサービスのアドレスを指定したい場合などには，このプロパティを指定してください。
  CORBA ネーミングサービスを自動起動モード（J2EE サーバ用ユーザプロパティファイルの ejbserver.naming.startupMode に automatic または inprocess を指定）で使用し，EJB コンテナの IP アドレスを固定（J2EE サーバ用ユーザプロパティファイルの vbroker.se.iiop_tp.host を指定）した場合は，このプロパティに固定した IP アドレスを指定してください。
- HRM_CMD_PORT
  CORBA ネーミングサービス（nameserv）のポート番号です。Reliable Messaging のコマンドが CORBA ネーミングサービス（nameserv）を使用するときのポート番号を指定します。
  ユーザが特に変更していない場合は，デフォルト値である 900 を指定してください。
- PATH
  Reliable Messaging が提供するコマンドが格納されたディレクトリを指定します。%HRMDIR%¥bin を追加してください。

3.4 Reliable Messaging のシステム構築（永続版リソースアダプタの場合）

永続版リソースアダプタを使用する場合の Reliable Messaging のシステム構築について説明します。

3.4.1 DBMS の設定（HiRDB を使用する場合）

HiRDB を使用する場合の DBMS の設定について説明します。説明する以外の設定内容や手順については，HiRDB のマニュアルを参照してください。

(1) DBMS の初期設定（HiRDB を使用する場合）

(a) HiRDB の初期設定

HiRDB をインストールしたら，環境設定をします。Reliable Messaging の管理情報テーブルを格納するための RD エリアを作成してください。HiRDB の環境設定については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(b) HiRDB のユーザ権限の付与

HiRDB では"ユーザの作成"という操作がなく，ユーザ権限の一つである CONNECT 権限をユーザに付与することによってユーザが作成されます。ユーザ権限については，マニュアル「HiRDB システム運用ガイド」を参照してください。

Reliable Messaging は HiRDB の接続ユーザを使用して HiRDB にアクセスします。Reliable Messaging で使用する接続ユーザには，次に示すユーザ権限を付与してください。

- CONNECT 権限
- スキーマ定義権限

ユーザ権限を付与するには，データベース定義ユティリティ（pddef）または HiRDB SQL Executer を使用します。ここでは，データベース定義ユティリティ（pddef）を使用してユーザ権限を付与する場合の手順を説明します。Reliable Messaging が動作するマシンまたは HiRDB が動作するマシンで，次に示す手順を実行してください。データベース定義ユティリティ（pddef）の使用方法については，マニュアル「HiRDB コマンドリファレンス」を参照してください。

1. 環境変数の設定

   次に示す環境変数を設定します。

   Windows の場合

```
SET PDHOST = <HiRDBサーバのホスト名またはIPアドレス>
SET PDNAMEPORT = <HiRDBサーバのポート番号>
SET PDUSER = <DBA権限を持つユーザ名>/<パスワード>
```

UNIX（csh）の場合

```
setenv PDHOST <HiRDBサーバのホスト名またはIPアドレス>
setenv PDNAMEPORT <HiRDBサーバのポート番号>
setenv PDUSER <DBA権限を持つユーザ名>/<パスワード>
```

UNIX（sh）の場合

```
PDHOST=<HiRDBサーバのホスト名またはIPアドレス>
PDNAMEPORT=<HiRDBサーバのポート番号>
PDUSER=<DBA権限を持つユーザ名>/<パスワード>
export PDHOST
export PDNAMEPORT
export PDUSER
```

2. データベース定義ユティリティ（pddef）の開始

次に示すとおりコマンドを入力して，データベース定義ユティリティ（pddef）を開始します。

```
pddef
```

3. SQL 文の実行

次に示す SQL 文を実行します。

```
GRANT  CONNECT  TO  <権限を付与する接続ユーザ名>※
             IDENTIFIED  BY  <パスワード>;
GRANT  SCHEMA   TO  <権限を付与する接続ユーザ名>※;
```

注※

Reliable Messaging が使用する接続ユーザの名前です。HiRDB のクライアント環境変数グループに登録した PDUSER 環境変数と同じ値になります。PDUSER 環境変数については，「3.4.1(2)(b)　HiRDB の環境変数グループの登録」を参照してください。

4. データベース定義ユティリティ（pddef）の終了

データベース定義ユティリティ（pddef）を終了するには，

Windows の場合

Ctrl キーと Z キーを同時に押したあと，Enter キーを押してください。

UNIX の場合

Ctrl キーと D キーを同時に押してください。

(c)　HiRDB のスキーマの定義

Reliable Messaging の管理情報テーブル用のスキーマを定義します。

スキーマを定義するには，データベース定義ユティリティ（pddef）または HiRDB SQL Executer を使用します。ここでは，データベース定義ユティリティ（pddef）を使用してスキーマ定義する場合の手順を説明します。Reliable Messaging が動作するマシンまたは HiRDB が動作するマシンで，次に示す手順を実行してください。データベース定義ユティリティ（pddef）の使用方法については，マニュアル「HiRDB コマンドリファレンス」を参照してください。

1. 環境変数の設定

次に示す環境変数を設定します。

Windows の場合

```
SET PDHOST = <HiRDBサーバのホスト名またはIPアドレス>
SET PDNAMEPORT = <HiRDBサーバのポート番号>
SET PDUSER = <接続ユーザ名>/<パスワード>※
```

UNIX（csh）の場合

```
setenv PDHOST <HiRDBサーバのホスト名またはIPアドレス>
setenv PDNAMEPORT <HiRDBサーバのポート番号>
setenv PDUSER <接続ユーザ名>/<パスワード>※
```

UNIX（sh）の場合

```
PDHOST=<HiRDBサーバのホスト名またはIPアドレス>
PDNAMEPORT=<HiRDBサーバのポート番号>
PDUSER=<接続ユーザ名>/<パスワード>※
export PDHOST
export PDNAMEPORT
export PDUSER
```

注※

接続ユーザ名には，権限を付与した接続ユーザ名を指定します。詳細については，「3.4.1(1)(b) HiRDB のユーザ権限の付与」を参照してください。

2. データベース定義ユティリティ（pddef）の開始

次に示すとおりコマンドを入力して，データベース定義ユティリティ（pddef）を開始します。

```
pddef
```

3. SQL 文の実行

次に示す SQL 文を実行します。

```
CREATE SCHEMA;
```

4. データベース定義ユティリティ（pddef）の終了

データベース定義ユティリティ（pddef）を終了するには，

Windows の場合

Ctrl キーと Z キーを同時に押したあと，Enter キーを押してください。

UNIX の場合

Ctrl キーと D キーを同時に押してください。

(d) HiRDB の RD エリアの準備

Reliable Messaging の管理情報テーブルを格納するために，必要に応じて RD エリアを作成します。RD エリアの作成方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

複数のユーザで Reliable Messaging にアクセスする際にアプリケーション認証を使用する場合は，公用 RD エリアを作成し，その公用 RD エリアに Reliable Messaging の管理情報テーブルを作成します。

(e)　HiRDB の排他資源と同時アクセス可能実表数の見積もり

HiRDB の排他資源および同時アクセス可能実表数の見積もりについては，「付録 F　HiRDB の見積もり」を参照してください。

(2)　DB クライアントの設定

次に示す製品をインストールしたら，Reliable Messaging が動作するマシンで設定をします。

- HiRDB/Run Time または HiRDB/Developer's Kit
- HiRDB SQL Executer

(a)　HiRDB の環境変数の設定

次に示す環境変数を設定します。

- PDXAMODE
  1 を指定します。
- PDTXACANUM
  Reliable Messaging が使用する DB コネクション数の最大値を指定します。この場合の最大値は，Reliable Messaging 本体およびアプリケーションが使用する DB コネクションの合計です。Reliable Messaging が DB コネクションを使用するタイミングと，使用するコネクションの数を次の表に示します。

表 3-2　Reliable Messaging が DB コネクションを使用するタイミングと使用するコネクションの数

項番	Reliable Messaging の機能	DB コネクションを使用するタイミング	コネクション使用数
1	コマンド	コマンド実行時に取得し，コマンド完了後に解放します。	1
2	共用キューによるシステム間連携	リソースアダプタの開始時に 1 コネクションを取得し，リソースアダプタの停止時に解放します。	1
3		イベント受信時に 1 コネクションを取得し，イベント受信完了後に解放します。	1
4	メモリ復元処理	メモリ復元時に取得し，復元完了時に解放します。復元処理の詳細については，「4.1.4(1)　状態遷移」の開始中状態を参照してください。	1
5	メッセージ削除処理	リソースアダプタの開始時に取得し，リソースアダプタ停止時に解放します。	1
6	XA リカバリ	リソースアダプタの開始時に 1 コネクションを取得し，リソースアダプタの停止時に解放します。	1

項番	Reliable Messaging の機能	DB コネクションを使用するタイミング		コネクション使用数
7	キュー間転送	送信	1 メッセージの送信でキャッシュにメッセージがない場合，またはグループを切り替える場合に，1 コネクションを取得し，メッセージの送信終了後に解放します。 コネクション取得数の最大値は，送信スレッドの起動数です。	1 以上
8		受信	1 メッセージの受信で 1 コネクション取得し，メッセージの受信完了後に解放します。ただし，同時リクエスト数に比例してコネクション取得数も増加します。	1 以上

上記の表に示したとおり，Reliable Messaging の DB コネクション使用数は次の値になります。

6 ＋送信スレッドの起動数＋メッセージ受信リクエスト数

なお，Application Server の J2EE コンポーネントから HiRDB にアクセスする場合は，J2EE コンポーネントが利用するコネクション数に，Reliable Messaging の DB コネクション使用数を加えた値を指定します。

- LD_LIBRARY_PATH（Linux の場合），LIBPATH（AIX の場合）
  下記のパスを指定してください。
  /opt/Cosminexus/TPB/lib
  /opt/Cosminexus/PRF/lib
  /opt/Cosminexus/CTM/lib（CTM 利用時だけ）
  /opt/HiRDB/client/lib
  /opt/HiRDB/client/utl

注意事項

HiRDB のクライアント環境定義 PDDBLOG には，NO を指定しないで ALL を指定してください。

(b)　HiRDB の環境変数グループの登録

HiRDB の環境変数グループは，HiRDB のクライアント環境変数グループの設定ファイルに登録します。

登録方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

ここで登録した環境変数グループ名は，DB Connector for Reliable Messaging のコンフィグレーションプロパティで，XAOpen 文字列として設定する必要があります。DB Connector for Reliable Messaging のコンフィグレーションプロパティについては，「6.3　DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

HiRDB のクライアント環境変数グループに登録する内容を次の表に示します。

表 3-3　HiRDB のクライアント環境変数グループに登録する内容

項番	環境変数名	設定内容
1	PDHOST	HiRDB サーバのホスト名または IP アドレスを指定します。
2	PDUSER	HiRDB サーバのユーザ名※1 およびパスワードを指定します。
3	PDNAMEPORT	HiRDB サーバのポート番号を指定します。
4	PDSWAITTIME※2	Component Container のトランザクションタイムアウトの値※3 よりも大きな値を指定します。
5	PDCWAITTIME※2	Component Container のトランザクションタイムアウトの値※3 よりも大きな値を指定します。 また，メッセージの遅延削除にかかる時間よりも大きな値を指定します※4。
6	PDSWATCHTIME	0 を指定します。

注※1

Reliable Messaging が HiRDB の管理する DB にアクセスするために使用する接続ユーザの名前です。

共用キューを使用して複数システム間でのアプリケーション連携をする場合は，送信側システムと受信側システムで同じ接続ユーザ名を指定します。

注※2

HiRDB を使用する場合，Reliable Messaging のトランザクション決着中に，HiRDB の設定値である PDSWAITTIME および PDCWAITTIME に指定した値によってタイムアウトが発生し DB がロールバックすると，Reliable Messaging は閉塞することがあります。このため，PDSWAITTIME および PDCWAITTIME には，適切な値を設定してください。PDSWAITTIME および PDCWAITTIME については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。なお，Reliable Messaging が閉塞した場合は，Application Server を強制停止したあと，再起動してください。

注※3

Component Container のトランザクションタイムアウト値は J2EE サーバ用ユーザプロパティファイルまたは UserTransaction.setTransactionTimeout()メソッドで指定します。J2EE サーバ用ユーザプロパティファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。UserTransaction.setTransactionTimeout()メソッドについては，Java のドキュメントを参照してください。

注※4

チューニングは本番稼動環境と同等の環境で行う必要があります。更に遅延削除処理にかかる時間が最大となるように，以下の環境でチューニングを行う必要があります。

- システムへの負荷(メッセージの送受信処理など)が最大となるような環境
- キュー内の遅延削除対象となるメッセージ数，配信済み状態ではないメッセージ数，メッセージのサイズが最大となる環境

メッセージの遅延削除にかかる時間は，Reliable Messaging が HiRDB へ発行する，削除 SQL にかかる時間から判断します。削除 SQL にかかる時間は HiRDB の SQL トレースから判断します。

以下に，Reliable Messaging が遅延削除で発行する削除 SQL を示します。

削除 SQL は永続キュー毎に発行されるため，発行する SQL の中で一番時間のかかった SQL を元にチューニングを行ってください。

```
DELETE FROM <DBへの接続ユーザ名>.<RMSystemNameプロパティ指定値>_MSG_<キュー名称>
                                        WHERE GROUP_NAME=’’ AND DELETE_FLAG=?”
```

(3) Reliable Messaging の管理情報テーブルの作成

DBMS として HiRDB を使用する場合，次に示す手順で管理情報テーブルを作成します。HiRDB SQL Executer の使用方法については，HiRDB SQL Executer のドキュメントを参照してください。

1. コンフィグレーションプロパティの RMSystemName に指定する値（3 文字以内の英数字）の決定

   RMSystemName については，「6.2 Reliable Messaging のコンフィグレーションプロパティの詳細説明」を参照してください。

2. SQL ファイルの編集

   テーブル作成用 SQL ファイル（%HRMDIR%¥sql¥createtableshirdb.sql）を任意の場所にコピーしてから，このファイルをテキストエディタで開き，ファイル中の 5 か所の"<RMSystemName>"の部分を手順 1.で決定した RMSystemName の指定値で置き換えて，ファイルを保存します。

   また，必要に応じて"<RMAREA>"の部分を表の行を格納する RD エリア名に変更します。または"IN <RMAREA>"の部分を削除します。"IN <RMAREA>"の部分を削除した場合は，格納する RD エリアを HiRDB が決定します。HiRDB が決定する RD エリアについては，マニュアル「HiRDB SQL リファレンス」を参照してください。

3. SQL ファイルの実行

   Windows の場合

   次に示すコマンドで HiRDB SQL Executer を開始します。

   ```
   pdsqlw -u <接続ユーザ名>/<パスワード>※
          -h <HiRDBサーバのホスト名またはIPアドレス>
          -n <HiRDBサーバのポート番号>
   ```

   HiRDB SQL Executer の［ファイル］メニューから［ファイルから実行］を選択し，手順 2.で編集した SQL ファイルを指定して実行します。

   UNIX の場合

   環境変数 PDUSER※,PDHOST,PDNAMEPORT を設定し，次に示すコマンドで HiRDB SQL Executer から SQL ファイルを実行します。

   ```
   pdsql ＜ <手順2.で編集したSQLスクリプトファイルのパス>
   ```

   注※

   接続ユーザ名には，権限を付与した接続ユーザ名を指定します。詳細については，「3.4.1(1)(b) HiRDB のユーザ権限の付与」を参照してください。

共用キューを使用して複数システム間でのアプリケーション連携をする場合には，送信側システムと受信側システムで同じ接続ユーザ名を使用します。

3.4.2 DBMS の設定（Oracle を使用する場合）

Oracle を使用する場合の DBMS の設定について説明します。説明する以外の設定内容や手順については，Oracle のマニュアルを参照してください。

なお，Oracle RAC の機能は，Oracle 10g Release2 以降に対してだけ使用できます。Oracle RAC の機能を利用する場合は，Oracle Clusterware をインストールしてください。

(1) DBMS の初期設定（Oracle を使用する場合）

DBMS として Oracle を使用する場合は，Oracle に管理情報を格納するために，Oracle に接続ユーザを作成して，接続ユーザに権限を付与する必要があります。ユーザ作成や権限付与の方法の詳細については，Oracle のマニュアルを参照してください。

(a) 付与する権限

システム権限

- CREATE ANY INDEX システム権限
- CREATE SESSION システム権限
- CREATE TABLE システム権限
- FORCE ANY TRANSACTION システム権限

オブジェクト権限

- SYS.DBA_PENDING_TRANSACTIONS の SELECT 権限
- SYS.DBMS_SYSTEM の EXECUTE 権限

ロール権限

- SELECT_CATALOG_ROLE 権限

(b) ユーザ作成と権限付与の方法

次のどちらかの方法でユーザ作成と権限付与を行います。

- Oracle Enterprise Manager コンソールを使用する
- sqlplus を使用する

(2)　DB クライアントの設定

Oracle に接続するためには，Reliable Messaging が動作するマシンで，ネット・サービス名を作成します。ネット・サービス名を作成する方法については，Oracle のマニュアルを参照してください。

(3)　Reliable Messaging の管理情報テーブルの作成

DBMS として Oracle を使用する場合，次に示す手順で管理情報テーブルを作成します。SQL*Plus の使用方法については，Oracle のマニュアルを参照してください。

1. コンフィグレーションプロパティの RMSystemName に指定する値（3 文字以内の英数字）の決定

   RMSystemName については，「6.2　Reliable Messaging のコンフィグレーションプロパティの詳細説明」を参照してください。

2. SQL ファイルの編集

   テーブル作成用 SQL ファイル（%HRMDIR%¥sql¥createtablesoracle.sql）を任意の場所にコピーしてから，このファイルをテキストエディタで開き，ファイル中の 5 か所の"<RMSystemName>"の部分を手順 1.で決定した RMSystemName の指定値で置き換えて，ファイルを保存します。

3. GUI 版 SQL*Plus またはコマンドライン版 sqlplus の起動

   「3.4.2(1)　DBMS の初期設定（Oracle を使用する場合）」の手順で作成したユーザを使用してデータベースに接続します。

4. コマンドの実行

   次の形式で実行してください。

```
start <手順2. で編集したSQLファイルの絶対パス>
```

3.4.3　J2EE サーバ（Application Server）の設定

J2EE サーバ（Application Server）の設定をして，J2EE サーバを構築します。J2EE サーバの構築については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

3.4.4　キュー定義ファイルの作成（永続版リソースアダプタの場合）

キュー定義ファイルは，JNDI ネーミングサービスに登録するキュー表示名と実際のキュー名を関連づけるファイルです。テキストファイルで作成して，任意のディレクトリに格納してください。キュー定義ファイルのフルパスを QueueConfigFileName プロパティに指定すると，Reliable Messaging の開始時に，定義されたキュー表示名が JNDI ネーミングサービスに登録されます。

なお，キュー作成時の表示名を利用する場合，キュー定義ファイルの作成は不要です。

(1) キュー定義ファイルの記述形式

記述形式

```
QueueImplClass = jp.co.Hitachi.soft.reliablemessaging.ra.jms.QueueImpl※1
Queue.通番※2.DisplayName = キュー表示名※3
Queue.通番.QueueName = キュー名※4
```

注※1

ユーザの環境に依存しない固定の値です。

注※2

通番には 1～20480 の範囲の値を指定してください。

注※3

キュー表示名として任意の名称を指定します。

キュー表示名は JNDI ネーミングサービスに"<Reliable Messaging の表示名>__<キュー表示名>__que"の形式で登録されます。なお，"<Reliable Messaging の表示名>__<キュー表示名>__que"の形式の文字列の長さが 256 文字を超えないようにしてください。

<Reliable Messaging の表示名>は，Connector 属性ファイルの<display-name>タグの指定値をサニタイズ（半角英数字以外をアンダスコア（_）に変更）した値です。<display-name>タグの初期値は"Cosminexus_Reliable_Messaging"です。Connector 属性ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

<キュー表示名>には，アンダスコア二つ（__）を指定しないでください。

注※4

キュー名は hrmmkque コマンド引数指定値を指定します。

(2) キュー定義ファイルの記述例

```
QueueImplClass = jp.co.Hitachi.soft.reliablemessaging.ra.jms.QueueImpl
Queue.1.DisplayName = Q1
Queue.1.QueueName = Queue1
Queue.2.DisplayName = Q2
Queue.2.QueueName = Queue2
```

(3) キュー定義ファイルの作成規則

- java.util.Properties.load()メソッドで読み取りできる形式にしてください。コメントや継続行の扱いも同様です。
- QueueImplClass の指定がない場合および値が指定されていない場合はデプロイ時にエラーが発生します。
- 一つのキューについて DisplayName および QueueName の両方を指定してください。
- 同一のプロパティを複数記述した場合は，どの値が有効になるか保証しません。

- 通番は 1 から指定してください。通番 1 のキュー定義がない場合はデプロイ時にエラーが発生します。
- Queue.n.DisplayName の指定がない場合および値が指定されていない場合，通番が n-1 番目のキューまでが有効になります。通番 n 以降のキュー定義は無視されます。
- QueueName の指定がない，または値が指定されていないキューは無視されます。
- キュー定義が一つもない場合はフォーマット不正によってデプロイ時にエラーが発生します。
- 表示名が重複した場合，通番の大きい方が有効になります。
- 通番の最大値より大きい値のキュー定義は無視されます。
- DisplayName に指定できる文字は，半角英字（A～Z，a～z）半角数字（0～9）および半角のアンダスコア（_）です。これら以外の文字を指定したキューは無視されます。
- QueueName に指定する値は，大文字と小文字を hrmmkque コマンド引数指定値と一致させてください。

3.4.5 Reliable Messaging のプロパティ定義（永続版リソースアダプタの場合）

Reliable Messaging のプロパティを定義します。Reliable Messaging をデプロイしたあとでも実行できます。なお，設定済みの Reliable Messaging のプロパティを変更する場合は，該当する Reliable Messaging を停止した状態で実行してください。

Reliable Messaging がデータベースと接続するには，DB Connector for Reliable Messaging のコンフィグレーションプロパティを設定する必要があります。DB Connector for Reliable Messaging のコンフィグレーションプロパティの設定については，「3.4.8 DB Connector for Reliable Messaging のプロパティ定義」を参照してください。

(1) 編集する属性ファイル

Reliable Messaging で提供する Connector 属性ファイルのテンプレートを任意のディレクトリにコピーして，コピーしたファイルを編集します。

Connector 属性ファイルのテンプレートは，次のディレクトリに格納されています。

```
%HRMDIR%¥conf¥rm_prop.xml
```

なお，cjgetresprop コマンドによって取得した Connector 属性ファイルを使用することもできます。

注意事項

テンプレートからコピーした Connector 属性ファイルに ASCII 文字以外を使用する場合は，次のどちらかの対処が必要です。

- UTF-8 形式で保存する。
- 編集したファイルの文字エンコーディング形式に従って，ファイルの先頭行に次に示す encoding 宣言を追加する。

  <?xml version="1.0" encoding="＜文字エンコーディング＞"?>

(2) 編集する属性設定項目

Reliable Messaging のプロパティ設定項目を次に示します。

- リソースアダプタの一般情報
- コンフィグレーションプロパティ
- 実行時プロパティ

(a) リソースアダプタの一般情報

設定できる Reliable Messaging の一般情報属性（<outbound-resourceadapter>タグ）の設定項目を次に示します。

項目	必須	対応するタグ
トランザクションサポートのレベル	○	<transaction-support>
再認証のサポート有無※	○	<reauthentication-support>

（凡例） ○：必須

注※ 必ず false を指定してください。

プロパティの設定項目の説明については，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

(b) コンフィグレーションプロパティ

Reliable Messaging のコンフィグレーションプロパティ（<outbound-resourceadapter> - <connection-definition> - <config-property>タグ）の設定項目を次に示します。

項目	対応するタグ
コンフィグレーションプロパティ名	<config-property-name>
コンフィグレーションプロパティのデータ型	<config-property-type>
コンフィグレーションプロパティの値	<config-property-value>※

注※ コンフィグレーションプロパティの値をクリアする場合

<config-property-value>タグだけを指定して，値を指定しないでください。

<config-property-value>タグ自体が指定されていない場合は，そのプロパティの値は変更されないで，すでに設定されている値がそのまま有効になります。

定義するコンフィグレーションプロパティの数だけ上記の設定を繰り返してください。

Reliable Messaging のコンフィグレーションプロパティの設定内容については，「6.　コンフィグレーションプロパティ」を参照してください。

(c)　実行時プロパティ

Reliable Messaging の実行時プロパティ（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <property>タグ）の設定項目を次に示します。

項目	対応するタグ
プロパティ名	<property-name>
プロパティのデータ型	<property-type>
プロパティの値	<property-value>

定義するプロパティの数だけ，上記の設定を繰り返してください。

実行時プロパティ名（<property-name>）およびコネクションプールの動作と注意事項については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の DB Connector のプロパティ定義に関する内容を参照してください。

注意事項

ユーザ名およびコネクションプールに関する設定値は，Reliable Messaging と DB Connector for Reliable Messaging で同じ値を設定してください。

(3)　コンフィグレーションプロパティの設定例

Reliable Messaging のコンフィグレーションプロパティの設定例を示します。コンフィグレーションプロパティの各項目の詳細については，「6.2　Reliable Messaging のコンフィグレーションプロパティの詳細説明」を参照してください。

表 3-4　Reliable Messaging のコンフィグレーションプロパティの設定例（永続版リソースアダプタの場合）

項番	プロパティ名	設定例
1	RMSystemName	＜システム名＞
2	RMLinkedDBConnectorName	＜連携する DB Connector の表示名＞
3	QueueConfigFileName	指定しない
4	RMDeadMessageQueueName	＜デッドメッセージキュー名＞
5	RMWaitRestoration	true

項番	プロパティ名	設定例
6	RMStartTimeout	60
7	RMAssociateJDBCFlag	true
8	RMSweepTimerInterval	600
9	RMDeleteMessageImmediately	false
10	RMPassByReference	false
11	RMMaxDeliveryNum	10
12	RMMethodTraceLevel	1
13	RMLineTraceLevel	3
14	RMLogTraceFileNum	2
15	RMLogTraceFileSize	2097152
16	RMSHConnectFlag	false
17	RMSHPort※1	20351
18	RMSHRecoveryTimerInterval※1	60
19	RMTRConnectFlag	false
20	RMTRSendThreadNum※2	1
21	RMTRResendInterval1Num※2	6
22	RMTRResendInterval1※2	10
23	RMTRResendInterval2※2	600
24	RMTRResendTimerInterval※2	10
25	RMTRPendingNotifyInterval※2	600
26	RMTRTransferControlDir※2	＜クライアント定義ファイルが格納されているディレクトリのパス＞
27	RMAutoDeleteMessage	false

注※1

RMSHConnectFlag プロパティが false の場合，設定値は無視されます。

注※2

RMTRConnectFlag プロパティが false の場合，設定値は無視されます。

3.4.6 DB Connector for Reliable Messaging の選択

インポートする DB Connector for Reliable Messaging を選択します。使用するデータベース，およびトランザクションの管理方法によって，次の表に示すどれかの RAR ファイルを指定します。

表 3-5　指定する RAR ファイルの種類

項番	RAR ファイル	説明
1	DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar	HiRDB Type4 JDBC Driver 用の DB Connector for Reliable Messaging です。HiRDB を使用する場合で，ローカルトランザクションを使用するとき，またはトランザクション管理なしで使用するとき（トランザクションのサポートレベルに LocalTransaction または NoTransaction を指定するとき）に選択します。 HiRDB Type4 JDBC Driver の ConnectionPoolDataSource を使用して HiRDB に接続します。
2	DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar	HiRDB Type4 JDBC Driver 用の DB Connector for Reliable Messaging です。HiRDB を使用する場合で，グローバルトランザクションを使用するとき（トランザクションのサポートレベルに XATransaction を指定するとき）に選択します。 HiRDB Type4 JDBC Driver の XADataSource を使用して，HiRDB に接続します。
3	DBConnector_Oracle_CP_Cosminexus_RM.rar	Oracle JDBC Thin Driver 用の DB Connector for Reliable Messaging です。Oracle を使用する場合で，ローカルトランザクションを使用するとき，またはトランザクション管理なしで使用するとき（トランザクションのサポートレベルに LocalTransaction または NoTransaction を指定するとき）に選択します。 Oracle JDBC Thin Driver の ConnectionPoolDataSource を使用して Oracle に接続します。
4	DBConnector_Oracle_XA_Cosminexus_RM.rar	Oracle JDBC Thin Driver 用の DB Connector for Reliable Messaging です。Oracle を使用する場合で，グローバルトランザクションを使用するとき（トランザクションのサポートレベルに XATransaction を指定するとき）に選択します。 Oracle JDBC Thin Driver の XADataSource を使用して Oracle に接続します。

3.4.7　DB Connector for Reliable Messaging の前提製品の設定

(1)　Oracle JDBC Thin Driver の設定

Oracle JDBC Thin Driver を使用して Oracle に接続する場合，J2EE サーバ側で次に示す設定をしてください。

(a) Oracle JDBC Thin Driver の JAR ファイルの入手

Application Server がサポートするバージョンの，Oracle JDBC Thin Driver の JAR ファイルを入手してください。

(b) J2EE サーバのユーザクラスパスの指定

J2EE サーバのユーザクラスパスに Oracle JDBC Thin Driver の JAR ファイルを指定してください。J2EE サーバ用の usrconf.cfg の add.class.path キーに「add.class.path=<Oracle JDBC Thin Driver の JAR ファイルのパス>」の形式で指定してください。

Application Server がサポートするバージョンの Oracle JDBC Thin Driver の JAR ファイル，および Oracle JDBC Thin Driver の設定方法の詳細については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の Oracle の設定（Oracle JDBC Thin Driver の場合）に関する内容を参照してください。

(2) HiRDB Type4 JDBC Driver の設定

HiRDB Type4 JDBC Driver を使用して HiRDB に接続する場合，J2EE サーバ側で次に示す設定をしてください。

(a) HiRDB Type4 JDBC Driver の JAR ファイルの入手

HiRDB Type4 JDBC Driver の JAR ファイルを入手してください。

JAR ファイルの格納場所については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(b) J2EE サーバのユーザクラスパスの指定

J2EE サーバのユーザクラスパスに HiRDB Type4 JDBC Driver の JAR ファイルを指定してください。J2EE サーバ用の usrconf.cfg の add.class.path キーに「add.class.path=<HiRDB Type4 JDBC Driver の JAR のパス>」の形式で指定してください。

HiRDB Type4 JDBC Driver の設定方法の詳細については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の HiRDB の設定（HiRDB Type4 JDBC Driver の場合）に関する内容を参照してください。

3.4.8 DB Connector for Reliable Messaging のプロパティ定義

Reliable Messaging がデータベースと接続するには，DB Connector for Reliable Messaging のプロパティを定義する必要があります。「3.4.6 DB Connector for Reliable Messaging の選択」で選択した DB Connector for Reliable Messaging のプロパティを定義します。

DB Connector for Reliable Messaging のプロパティの定義は，DB Connector for Reliable Messaging をデプロイしたあとでも実行できます。なお，設定済みの DB Connector for Reliable Messaging のプ

ロパティを変更する場合は，該当する DB Connector for Reliable Messaging を停止した状態で実行してください。

注意事項

DB Connector for Reliable Messaging では，認証機能情報およびコネクションプーリング情報の設定は無効になります。このため，リソースアダプタの認証機能情報のユーザ名とパスワードの設定は，Reliable Messaging のプロパティ設定で実施してください。

(1) 編集する属性ファイル

Application Server で提供する DB Connector for Reliable Messaging の Connector 属性ファイルのテンプレートを任意のディレクトリにコピーして，コピーしたファイルを編集します。

DB Connector for Reliable Messaging の Connector 属性ファイルのテンプレートは，次のディレクトリに格納されています。

```
＜Application Serverのインストールディレクトリ＞¥CC¥admin¥templates
```

DB Connector for Reliable Messaging の Connector 属性ファイルのテンプレートについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」の Connector 属性ファイルのテンプレートファイルに関する内容を参照してください。

注意事項

テンプレートからコピーした Connector 属性ファイルに ASCII 文字以外を使用する場合は，次のどちらかの対処が必要です。

- UTF-8 形式で保存する。
- 編集したファイルの文字エンコーディング形式に従って，ファイルの先頭行に次に示す encoding 宣言を追加する。

  <?xml version="1.0" encoding="＜文字エンコーディング＞"?>

(2) 編集する属性設定項目

DB Connector for Reliable Messaging のプロパティ設定項目を次に示します。

- リソースアダプタの一般情報
- コンフィグレーションプロパティ
- 実行時プロパティ

(a) リソースアダプタの一般情報

設定できる DB Connector for Reliable Messaging の一般情報属性（<outbound-resourceadapter>タグ）の設定項目を次に示します。

項目	必須	対応するタグ
トランザクションサポートのレベル	○	<transaction-support>
再認証のサポート有無	○	<reauthentication-support>

（凡例）　○：必須

プロパティの設定項目の説明については，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

(b)　コンフィグレーションプロパティ

リソースアダプタのコンフィグレーションプロパティ（<outbound-resourceadapter> - <connection-definition> - <config-property>タグ）の設定項目を次に示します。

項目	対応するタグ
コンフィグレーションプロパティ名	<config-property-name>
コンフィグレーションプロパティのデータ型	<config-property-type>
コンフィグレーションプロパティの値	<config-property-value>※

注※　コンフィグレーションプロパティの値をクリアする場合

<config-property-value>タグだけを指定して，値を指定しないでください。

<config-property-value>タグ自体が指定されていない場合は，そのプロパティの値は変更されないで，すでに設定されている値がそのまま有効になります。

定義するコンフィグレーションプロパティの数だけ上記の設定を繰り返してください。

設定する必要がある項目は，DB Connector for Reliable Messaging の種類によって一部異なります。DB Connector for Reliable Messaging の種類によって，設定できるコンフィグレーションプロパティの違いについては，「3.4.8(3)　DB Connector for Reliable Messaging の種類によるコンフィグレーションプロパティの違い」を参照してください。

なお，PreparedStatementPoolSize，CallableStatementPoolSize，LogLevel，および CancelStatement 以外のプロパティは，すべて HiRDB Type4 JDBC Driver，または Oracle JDBC Thin Driver に設定する項目です。詳細は，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

DB Connector for Reliable Messaging のコンフィグレーションプロパティの設定内容については，「6.3 DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

(c)　実行時プロパティ

DB Connector for Reliable Messaging の実行時プロパティ（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <property>タグ）の設定項目を次に示します。

項目	対応するタグ
プロパティ名	<property-name>
プロパティのデータ型	<property-type>
プロパティの値	<property-value>

定義するプロパティの数だけ，上記の設定を繰り返してください。

実行時プロパティ名（<property-name>）およびコネクションプールの動作と注意事項については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の DB Connector のプロパティ定義に関する内容を参照してください。

なお，DB Connector for Reliable Messaging の場合は，次の実行時プロパティの設定だけが有効となります。

- ログを出力するかどうかの選択（LogEnabled）

注意事項

ユーザ名およびコネクションプールに関する設定値は，Reliable Messaging と DB Connector for Reliable Messaging で同じ値を設定してください。

(3) DB Connector for Reliable Messaging の種類によるコンフィグレーションプロパティの違い

DB Connector for Reliable Messaging の種類によって，設定するコンフィグレーションプロパティが異なります。DB Connector for Reliable Messaging の種類を次に示します。

- HiRDB Type4 JDBC Driver 対応の DB Connector for Reliable Messaging
- Oracle JDBC Thin ドライバ対応の DB Connector for Reliable Messaging

(a) 各 DB Connector for Reliable Messaging 共通のコンフィグレーションプロパティ

HiRDB Type4 JDBC Driver 対応の DB Connector for Reliable Messaging，Oracle JDBC Thin ドライバ対応の DB Connector for Reliable Messaging 共通で設定が必要なコンフィグレーションプロパティを次の表に示します。

表 3-6 各 DB Connector for Reliable Messaging 共通のコンフィグレーションプロパティ

プロパティ名	設定値
linkedResourceAdapterName	Reliable Messaging の表示名

(b) HiRDB Type4 JDBC Driver 対応の DB Connector for Reliable Messaging のコンフィグレーションプロパティ

HiRDB Type4 JDBC Driver 対応の DB Connector for Reliable Messaging を使用する場合，Reliable Messaging として設定が必要な DB Connector for Reliable Messaging のコンフィグレーションプロパティを次の表に示します。

表 3-7 HiRDB Type4 JDBC Driver 対応の DB Connector for Reliable Messaging のコンフィグレーションプロパティ

項番	プロパティ名	設定値
1	description	接続先 DB に必要な接続付加情報
2	DBHostName	接続先 HiRDB のホスト名
3	XAOpenString※	XA_OPEN 文字列
4	SQLWarningLevel	SQL 実行時に発生した警告保持レベル
5	maxBinarySize	JDBC SQL タイプ LONGVARBINARY 型データ取得時のデータサイズの上限

注※ XATransaction で使用する場合だけ設定が必要です。

(c) Oracle JDBC Thin ドライバ対応の DB Connector for Reliable Messaging のコンフィグレーションプロパティ

Oracle JDBC Thin ドライバ対応の DB Connector for Reliable Messaging を使用する場合，Reliable Messaging として設定が必要な DB Connector for Reliable Messaging のコンフィグレーションプロパティを次の表に示します。

表 3-8 Oracle JDBC Thin ドライバ対応の DB Connector for Reliable Messaging のコンフィグレーションプロパティ

項番	プロパティ名	設定値
1	databaseName	Oracle サーバ上の特定のデータベース名（SID）
2	serverName	Oracle サーバのホスト名または IP アドレス

(4) コンフィグレーションプロパティの設定例

次に示す場合の DB Connector for Reliable Messaging のコンフィグレーションプロパティの設定例を示します。

- DBConnector_DABJ_CP_Cosminexus_RM.rar で，HiRDB を使用する場合
- DBConnector_DABJ_XA_Cosminexus_RM.rar で，HiRDB を使用する場合
- DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar で，HiRDB を使用する場合

- DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar で，HiRDB を使用する場合
- DBConnector_Oracle_CP_Cosminexus_RM.rar で，Oracle を使用する場合
- DBConnector_Oracle_XA_Cosminexus_RM.rar で，Oracle を使用する場合

コンフィグレーションプロパティの各項目の詳細については，「6.3　DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

(a)　DBConnector_DABJ_CP_Cosminexus_RM.rar で，HiRDB を使用する場合

DBConnector_DABJ_CP_Cosminexus_RM.rar で，HiRDB を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 3-9　データベースとして HiRDB を使用する場合
(DBConnector_DABJ_CP_Cosminexus_RM.rar の場合)

項番	プロパティ名	HiRDB の場合の設定例
1	linkedResourceAdapterName	＜連携する Reliable Messaging のリソースアダプタの表示名＞
2	networkProtocol	lib
3	description	＜ HiRDB ポート番号＞※
4	DBHostName	＜ HiRDB ホスト名＞
5	loginTimeout	0
6	serverName	−
7	portNumber	40179
8	databaseName	HIRDB
9	DBEnv	−
10	encodLang	−
11	JDBC_IF_TRC	false
12	SV_EVENT_TRC	false
13	TRC_NO	500
14	uapName	−
15	bufSize	64
16	rowSize	16
17	OSAuthorize	false
18	HiRDBCursorMode	false
19	blockUpdate	false
20	executeDirectMode	false

項番	プロパティ名	HiRDB の場合の設定例
21	SQLWarningIgnore	false
22	LONGVARBINARY_Access	REAL
23	bufferPoolSize	0
24	PreparedStatementPoolSize	10
25	CallableStatementPoolSize	10
26	ConnectionIDUpdate	false
27	logLevel	ERROR

（凡例）　－：設定は不要です。

注※　HiRDB クライアントの環境変数グループ名（Windows の場合）または環境変数グループの設定ファイルのパス（UNIX の場合）を指定することもできます。

(b)　DBConnector_DABJ_XA_Cosminexus_RM.rar で，HiRDB を使用する場合

DBConnector_DABJ_XA_Cosminexus_RM.rar で，HiRDB を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 3-10　データベースとして HiRDB を使用する場合 (DBConnector_DABJ_XA_Cosminexus_RM.rar の場合)

項番	プロパティ名	HiRDB の場合の設定例
1	linkedResourceAdapterName	＜連携する Reliable Messaging のリソースアダプタの表示名＞
2	networkProtocol	lib
3	description	＜環境変数グループ識別子＞※1
4	DBHostName	＜ HiRDB ホスト名＞
5	XAOpenString	Windows の場合 ＜環境変数グループ識別子※2 ＞＋＜環境変数グループ名※3 ＞ UNIX の場合 ＜環境変数グループ識別子※2 ＞＋＜環境変数グループの設定ファイルのパス※3 ＞
6	loginTimeout	0
7	serverName	－
8	portNumber	40179
9	databaseName	HIRDB
10	DBEnv	－
11	encodLang	－
12	JDBC_IF_TRC	false

項番	プロパティ名	HiRDB の場合の設定例
13	SV_EVENT_TRC	false
14	TRC_NO	500
15	uapName	－
16	bufSize	64
17	rowSize	16
18	OSAuthorize	false
19	HiRDBCursorMode	false
20	blockUpdate	false
21	executeDirectMode	false
22	SQLWarningIgnore	false
23	LONGVARBINARY_Access	REAL
24	bufferPoolSize	0
25	XACloseString	－
26	RMID	1※4
27	XAThreadMode	true
28	XALocalCommitMode	true
29	PreparedStatementPoolSize	10
30	CallableStatementPoolSize	10
31	ConnectionIDUpdate	false
32	logLevel	ERROR

（凡例）　－：設定は不要です。

注※1　J2EE サーバ内でユニークな 4 バイトの文字列を指定します。

注※2　［Description］フィールドに入力した値を指定します。

注※3　HiRDB の環境変数グループ名（Windows の場合）または環境変数グループの設定ファイルのパス（UNIX の場合）を指定します。詳細については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

注※4　リソースマネジャの識別子を指定します。J2EE サーバ内でユニークな 1～2147483647 の数値で指定します。

(c)　DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar で，HiRDB を使用する場合

DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar で，HiRDB を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 3-11　データベースとして HiRDB を使用する場合
(DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar の場合)

項番	プロパティ名	HiRDB の場合の設定例
1	linkedResourceAdapterName	＜連携する Reliable Messaging のリソースアダプタの表示名＞
2	description	＜ HiRDB ポート番号＞※
3	DBHostName	＜ HiRDB ホスト名＞
4	environmentVariables	＜ HiRDB クライアント環境変数名＞
5	loginTimeout	8
6	encodeLang	－
7	JDBC_IF_TRC	false
8	TRC_NO	500
9	uapName	－
10	LONGVARBINARY_Access	REAL
11	SQLInNum	300
12	SQLOutNum	300
13	SQLWarningLevel	SQLWARN
14	SQLWarningIgnore	false
15	HiRDBCursorMode	false
16	maxBinarySize	64000
17	LONGVARBINARY_AccessSize	0
18	PreparedStatementPoolSize	10
19	CallableStatementPoolSize	10
20	CancelStatement	true
21	logLevel	ERROR

(凡例)　－：設定は不要です。

注※　HiRDB クライアントの環境変数グループ名（Windows の場合）または環境変数グループの設定ファイルのパス（UNIX の場合）も指定できます。

(d)　DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar で，HiRDB を使用する場合

DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar で，HiRDB を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 3-12　データベースとして HiRDB を使用する場合 (DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar の場合)

項番	プロパティ名	HiRDB の場合の設定例
1	linkedResourceAdapterName	＜連携する Reliable Messaging のリソースアダプタの表示名＞
2	description	＜環境変数グループ識別子＞
3	DBHostName	＜ HiRDB ホスト名＞
4	environmentVariables	＜ HiRDB クライアント環境変数名＞
5	XAOpenString	Windows の場合 ＜環境変数グループ識別子※1 ＞＋＜環境変数グループ名※2 ＞ UNIX の場合 ＜環境変数グループ識別子※1 ＞＋＜環境変数グループの設定ファイルのパス※2 ＞
6	loginTimeout	8
7	encodeLang	－
8	JDBC_IF_TRC	false
9	TRC_NO	500
10	uapName	－
11	LONGVARBINARY_Access	REAL
12	SQLInNum	300
13	SQLOutNum	300
14	SQLWarningLevel	SQLWARN
15	SQLWarningIgnore	false
16	HiRDBCursorMode	false
17	maxBinarySize	64000
18	LONGVARBINARY_AccessSize	0
19	XACloseString	－
20	XALocalCommitMode	true
21	PreparedStatementPoolSize	10
22	CallableStatementPoolSize	10
23	CancelStatement	true
24	logLevel	ERROR

（凡例）　－：設定は不要です。

注※1　［Description］フィールドに入力した値を指定します。

注※2　HiRDB の環境変数グループ名（Windows の場合）または環境変数グループの設定ファイルのパス（UNIX の場合）を指定します。詳細については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

(e)　DBConnector_Oracle_CP_Cosminexus_RM.rar で，Oracle を使用する場合

DBConnector_Oracle_CP_Cosminexus_RM.rar で，Oracle を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 3-13　データベースとして Oracle を使用する場合 (DBConnector_Oracle_CP_Cosminexus_RM.rar の場合)

項番	プロパティ名	Oracle の場合の設定例
1	linkedResourceAdapterName	＜連携する Reliable Messaging のリソースアダプタの表示名＞
2	databaseName	＜ Oracle SID ＞
3	serverName	＜接続先 Oracle のホスト名称，または IP アドレス＞
4	portNumber	1521
5	url	－
6	loginTimeout	8000
7	PreparedStatementPoolSize	10
8	CallableStatementPoolSize	10
9	ConnectionIDUpdate	false
10	logLevel	ERROR

（凡例）　－：設定は不要です。

(f)　DBConnector_Oracle_XA_Cosminexus_RM.rar で，Oracle を使用する場合

DBConnector_Oracle_XA_Cosminexus_RM.rar で，Oracle を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 3-14　データベースとして Oracle を使用する場合 (DBConnector_Oracle_XA_Cosminexus_RM.rar の場合)

項番	プロパティ名	Oracle の場合の設定例
1	linkedResourceAdapterName	＜連携する Reliable Messaging のリソースアダプタの表示名＞
2	databaseName	＜ Oracle SID ＞
3	serverName	＜接続先 Oracle のホスト名称，または IP アドレス＞
4	portNumber	1521
5	url	－
6	loginTimeout	8000

項番	プロパティ名	Oracle の場合の設定例
7	sessionTimeout	300
8	PreparedStatementPoolSize	10
9	CallableStatementPoolSize	10
10	ConnectionIDUpdate	false
11	logLevel	ERROR

（凡例） －：設定は不要です。

3.4.9 DB Connector for Reliable Messaging と Reliable Messaging のインポート

DB Connector for Reliable Messaging および Reliable Messaging をインポートします。

(1) DB Connector for Reliable Messaging のインポート

次に示すコマンドを実行して DB Connector for Reliable Messaging をインポートします。

cjimportres コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

実行形式

```
cjimportres ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -type rar -f ＜ファイルパス＞
```

実行例

```
cjimportres MyServer -type rar -f ”C:¥Program Files¥Hitachi¥Cosminexus¥CC¥DBConnector¥ReliableMessaging¥DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar”
```

＜ファイルパス＞には，「3.4.6 DB Connector for Reliable Messaging の選択」で選択した RAR ファイル指定してください。RAR ファイルは，次のディレクトリに格納されています。

- Windows の場合

  ＜Application Serverのインストールディレクトリ＞¥CC¥DBConnector¥ReliableMessaging¥
- UNIX の場合

  /opt/Cosminexus/CC/DBConnector/ReliableMessaging/

(2) Reliable Messaging のインポート

DB Connector for Reliable Messaging と同様に，cjimportres コマンドを実行して，Reliable Messaging をインポートします。

実行例

```
cjimportres MyServer -type rar -f "C:¥Program Files¥Hitachi¥Cosminexus¥RM¥lib¥reliablemes
saging.rar"
```

＜ファイルパス＞には，Reliable Messaging の RAR ファイル（reliablemessaging.rar）を指定してください。RAR ファイルは，次のディレクトリに格納されています。

%HRMDIR%¥lib

3.4.10 DB Connector for Reliable Messaging と Reliable Messaging のプロパティ設定

次に示すコマンドを実行して，DB Connector for Reliable Messaging および Reliable Messaging のプロパティを設定します。

実行形式

```
cjsetresprop ［＜サーバ名称＞］ -type rar -resname ＜DB Connector for Reliable Messaging
の表示名，またはReliable Messagingの表示名＞ -c ＜Connector属性ファイルパス＞
```

実行例

```
cjsetresprop MyServer -type rar -resname Reliable_Messaging -c rm_prop.xml
```

注意事項

DB Connector for Reliable Messaging および Reliable Messaging をデプロイしたあとでプロパティを定義する場合は，cjgetrarprop コマンドと cjsetrarprop コマンドを使用してください。コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

3.4.11 DB Connector for Reliable Messaging と Reliable Messaging のデプロイ

DB Connector for Reliable Messaging および Reliable Messaging は，デプロイすると J2EE リソースアダプタとして使用できます。

サーバ管理コマンドでインポートした DB Connector for Reliable Messaging および Reliable Messaging をデプロイすると，その J2EE サーバ上で動作するすべての J2EE アプリケーションから使用できるようになります。なお，デプロイしたあとで，プロパティを定義することもできます。デプロイ後に定義する場合は，該当する DB Connector for Reliable Messaging および Reliable Messaging を停止した状態で実行してください。プロパティを定義する方法については，「3.4.5 Reliable Messaging のプロパティ定義（永続版リソースアダプタの場合）」および「3.4.8 DB Connector for Reliable Messaging のプロパティ定義」を参照してください。

次に示すコマンドを実行して，DB Connector for Reliable Messaging および Reliable Messaging をデプロイします。

実行形式

```
cjdeployrar ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -resname ＜DB Connector for Reliable Messagingの表示名＞ -resname ＜Reliable Messagingの表示名＞
```

実行例

```
cjdeployrar MyServer -resname DB_Connector_for_HiRDB_Type4_Cosminexus_RM -resname Cosminexus_Reliable_Messaging
```

cjdeployrar コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

3.4.12 Reliable Messaging の運用前の準備（永続版リソースアダプタの場合）

Reliable Messaging の運用前の準備について説明します。

(1) DB Connector for Reliable Messaging と Reliable Messaging の接続テスト

DB Connector for Reliable Messaging および Reliable Messaging に設定した内容が正しいかどうか，接続テストによって検証します。

(a) 接続テストで確認できる項目

接続テストの成功によって，次に示す項目を確認できます。

- 接続先 DB の設定（ホスト名，ポート番号，ユーザ名，パスワードなど）が正しいこと。
- 接続先 DB が正常に開始していること。
- RMLinkedDBConnectorName プロパティが正しく設定されていて，かつ指定した Reliable Messaging と連携する DB Connector for Reliable Messaging が開始済みであること。
- RMSystemName プロパティのフォーマットが正しいこと。

(b) 接続テストの手順

1. DB Connector for Reliable Messaging を開始します。
   DB Connector for Reliable Messaging の開始方法については，「3.4.12(2) DB Connector for Reliable Messaging と Reliable Messaging の開始」を参照してください。
2. 次に示すコマンドを実行して Reliable Messaging の接続テストを実施します。

実行形式

```
cjtestres ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -type rar -resname ＜Reliable Messagingの表示名＞
```

実行例

```
cjtestres MyServer -type rar -resname Cosminexus_Reliable_Messaging
```

なお，接続に失敗した場合には，出力されるメッセージを基にエラーに対処し，再度接続テストを実施してください。

cjtestres コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

注意事項

一度接続テストをした DB Connector for Reliable Messaging および Reliable Messaging は，J2EE サーバを再起動するまで削除できません。DB Connector for Reliable Messaging および Reliable Messaging を削除する場合は，DB Connector for Reliable Messaging および Reliable Messaging を停止してから，J2EE サーバを再起動してください。

(c) 接続テストの失敗事例

接続テストに失敗する事例を次に示します。

- DB の管理情報テーブルが正常に作成されていない場合

  接続テストでは DB へのアクセスを試みます。Reliable Messaging が前提とする DB の管理情報テーブルにアクセスするため，DB の管理情報テーブルが正常に作成されていないときは接続テストに失敗します。

- Component Container 側でエラーを検知した場合

  接続テストでは Reliable Messaging と Component Container の両方でチェックを実行します。そのため，Component Container 側でエラーを検知すると接続テストに失敗する場合もあります。

接続テストに成功しても Reliable Messaging の起動時に接続に失敗する場合があります。事例を次に示します。

- QueueConfigFileName プロパティを指定した場合のファイルパスが不正のとき，または指定したキュー定義ファイルの内容が不正のとき
- ロガーの初期化に失敗したとき，またはコマンドのネーミングサービスの登録に失敗したとき

(2) DB Connector for Reliable Messaging と Reliable Messaging の開始

DB Connector for Reliable Messaging および Reliable Messaging を開始します。なお，DB Connector for Reliable Messaging および Reliable Messaging が連携するために，J2EE リソースアダプタは次の順序で開始してください。

1. DB Connector for Reliable Messaging
2. Reliable Messaging

この順序で開始しないとエラーになります。

なお，Reliable Messaging の接続テストを実施して，すでに DB Connector for Reliable Messaging が開始している場合は，手順 1.は不要です。

1. 次に示すコマンドを実行して DB Connector for Reliable Messaging を開始します。

   実行形式

   ```
   cjstartrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <DB Connecto
   r for Reliable Messagingの表示名>
   ```

   実行例

   ```
   cjstartrar MyServer -resname DB_Connector_for_HiRDB_Type4_Cosminexus_RM
   ```

2. 手順 1.と同様に，cjstartrar コマンドを実行して Reliable Messaging を開始します。

   実行形式

   ```
   cjstartrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <Reliable Me
   ssagingの表示名>
   ```

   実行例

   ```
   cjstartrar MyServer -resname Cosminexus_Reliable_Messaging
   ```

cjstartrar コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

注意事項

- J2EE アプリケーション中の J2EE リソースが DB Connector for Reliable Messaging または Reliable Messaging を参照している場合は，DB Connector for Reliable Messaging または Reliable Messaging を開始してから，J2EE アプリケーションを開始してください。
- 一度開始したリソースアダプタは，J2EE サーバを再起動するまで削除できません。リソースアダプタを削除する場合は，リソースアダプタを停止してから，J2EE サーバを再起動してください。

(3) キューの作成

DB Connector for Reliable Messaging および Reliable Messaging を開始すると，Reliable Messaging が管理状態になります。管理状態になったことを確認したら，hrmmkque コマンドを入力して，キューを作成してください。そのあと，hrmstart コマンドを入力して，Reliable Messaging を実行状態にします。

詳細については，「4.2 キューの運用（永続版リソースアダプタの場合）」を参照してください。

3.4.13　キュー間転送を使用する場合の設定

キュー間転送を使用する場合の設定について説明します。

(1)　SOAP 通信基盤の設定

キュー間転送をする場合，SOAP 通信基盤を使用します。

SOAP 通信基盤は，クライアント定義ファイルおよびサーバ定義ファイルに定義された内容に従って送受信処理をします。

(a)　クライアント定義ファイルの使用方法

SOAP 通信基盤が出力するメッセージログ，トレースファイルの内容を指定する場合，または Reliable Messaging を複数デプロイする場合は次に示す順序に従ってください。

1. クライアント定義ファイルを任意のディレクトリにコピーします。
2. Reliable Messaging の RMTRTransferControlDir プロパティに，コピーしたクライアント定義ファイルが格納されているディレクトリのパスを指定します。

注意事項

- クライアント定義ファイルは削除しないでください。
- クライアント定義ファイルのファイル名は変更しないでください。
- Reliable Messaging を複数デプロイする場合，RMTRTransferControlDir プロパティには，クライアント定義ファイルが格納されているディレクトリのパスを Reliable Messaging ごとに指定してください。なお，クライアント定義ファイルで設定する，c4web.logger.log_file_prefix キーは Reliable Messaging ごとに異なる文字列を設定してください。
- uCosminexus Reliable Messaging 01-00 から Reliable Messaging 01-02 以降にバージョンアップする場合は，J2EE サーバを起動する前に J2EE サーバ用オプション定義ファイル（usrconf.cfg）をテキストエディタで開き，次に示す行を削除してください。

  add.class.path=<uCosminexus Reliable Messaging 01-00 のインストールディレクトリ>¥conf
- J2EE サーバ用オプションファイル（usrconf.cfg）をテキストエディタで開き，add.class.path=<Application Server のインストールディレクトリ>¥c4web¥lib¥hitsaaj.jar が指定されていて，add.class.path=<Application Server のインストールディレクトリ>¥jaxws¥lib¥cjjaxws.jar が指定されていないことを確認してください。

クライアント定義ファイルで設定できるキー名称と値の一覧を次の表に示します。次の表に示したキーだけを設定してください。ほかのキーを設定した場合，動作は保証されません。

表 3-15　クライアント定義ファイルで設定できるキー名称と値の一覧

キー名称	値
c4web.logger.log_file_prefix	トレースファイル，アプリケーションログのプレフィクス
c4web.common.prf_trace_level	性能解析トレースの出力オプション
c4web.common.send_max_soap_envelope_size	送信できる SOAPEnvelope の最大サイズ
c4web.common.receive_max_soap_envelope_size	受信できる SOAPEnvelope の最大サイズ
c4web.attachment.send_max_attachment_count	送信できる添付データの最大個数
c4web.attachment.receive_max_attachment_count	受信できる添付データの最大個数
c4web.attachment.send_max_attachment_size	送信できる添付データの最大サイズ
c4web.attachment.receive_max_attachment_size	受信できる添付データの最大サイズ

クライアント定義ファイルで設定できるキーの詳細については，マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」を参照してください。

(b)　サーバ定義ファイルの設定

キュー間転送をする場合，サーバ定義ファイルは，キュー間転送用 Web アプリケーションを対象に設定する必要があります。

サーバ定義ファイルで設定できるキー名称と値の一覧を次の表に示します。次の表に示したキーだけを設定してください。ほかのキーを設定した場合，動作は保証されません。

表 3-16　サーバ定義ファイルで設定できるキー名称と値の一覧

キー名称	値
c4web.logger.<識別子>.log_file_prefix	トレースファイル，アプリケーションログのプレフィクス
c4web.common.<識別子>.prf_trace_level	性能解析トレースの出力オプション
c4web.common.<識別子>.send_max_soap_envelope_size	送信できる SOAPEnvelope の最大サイズ
c4web.common.<識別子>.receive_max_soap_envelope_size	受信できる SOAPEnvelope の最大サイズ
c4web.attachment.<識別子>.send_max_attachment_count	送信できる添付データの最大個数
c4web.attachment.<識別子>.receive_max_attachment_count	受信できる添付データの最大個数
c4web.attachment.<識別子>.send_max_attachment_size	送信できる添付データの最大サイズ
c4web.attachment.<識別子>.receive_max_attachment_size	受信できる添付データの最大サイズ

キー名称の<識別子>には，キュー間転送用 Web アプリケーションのコンテキストルートを指定します。

サーバ定義ファイルで設定できるキーの詳細については，マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」を参照してください。

(2)　キュー間転送を使用する場合の Reliable Messaging の設定

キュー間転送を使用する場合は，送信側と受信側の Reliable Messaging に対して次に示す設定をしてください。

1. コンフィグレーションプロパティの RMTRConnectFlag に true を指定
2. サーバ管理コマンドで Reliable Messaging の ear ファイル（キュー間転送用 Web アプリケーション）をインポート
3. サーバ管理コマンドで Reliable Messaging のキュー間転送用 Web アプリケーションと Reliable Messaging のリソースアダプタを関連づけ
4. hrmstart コマンドを入力して Reliable Messaging を実行状態に移行

(3)　転送データ相互接続用インタフェースを利用する場合の設定

転送データ相互接続用インタフェースを利用したアプリケーションを作成する場合，転送データ相互接続用インタフェースのライブラリを開発環境のクラスパスに含める必要があります。詳細は「7.5　転送データ相互接続用インタフェースの一覧」を参照してください。

3.5　Reliable Messaging のシステム構築（非永続版リソースアダプタの場合）

非永続版リソースアダプタの場合の Reliable Messaging のシステム構築について説明します。

3.5.1　キュー作成ファイルの作成

非永続版リソースアダプタを使用する場合，キュー作成ファイルに記述したキュー定義文を基にキューを作成します。キュー作成ファイルはテキストファイルで作成して，任意のディレクトリに格納してください。キュー作成ファイルのフルパスを QueueMakeFileName プロパティに指定すると，キュー作成ファイルに指定されたキュー定義文を基に，Reliable Messaging 開始時に非永続キュー属性のローカルキューが作成されます。

(1)　キュー作成ファイルの記述形式

記述形式

```
hrmmkque　〔-d {serial｜parallel｜parallel_unit_order}〕
　　〔-n 最大メッセージ数〕〔-x 表示名〕
　　〔-e メッセージ有効期間〕
　　キュー名
```

- オプション

-d {serial ｜ parallel ｜ parallel_unit_order}　～《parallel》

作成するキューのメッセージ取り出しモードを指定します。

serial：シリアル取り出し属性

parallel：パラレル取り出し属性

parallel_unit_order：パラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）

各属性を指定したときのメッセージの処理については，「2.3.2　メッセージ取り出しモード」のメッセージ取り出しモードの説明を参照してください。

-n 最大メッセージ数　～＜数字＞((1～65535))《1024》

キューに格納するメッセージの最大数を指定します。

-e メッセージ有効期間　～＜数字＞((0～2592000))《0》

キューに格納するメッセージの有効期間を秒単位で指定します。

0 を指定する場合，メッセージの有効期間は無限です。

有効期間を指定するときのメッセージの処理については，「2.3.5　メッセージの有効期間」を参照してください。有効期間に達すると，そのメッセージは破棄されます。

-x 表示名　～＜ 1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はキュー名と同じ名称を指定したものと見なされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。

オプションを指定するときの注意事項を次に示します。

- オプションは，キュー名より前に指定してください。
- オプションは順序不同です。
- "hrmmkque"，オプション，およびキュー名の間は一つ以上の半角スペース，タブ，または改行のどれかを入力してください。
- 同じオプションフラグを 2 回以上指定すると最後に指定したものが有効となります。例えば，次のように「-n 1」のあとに「-n 2」を指定すると，「-n 2」が有効になります。

  ```
  hrmmkque -n 1 -n 2
  ```

- 引数

  キュー名　～＜ 1～20 文字の識別子＞

  作成するキューの名前を指定します。

  指定できる文字は，先頭文字が英数字で，2 文字目以降が英数字または_（アンダスコア）となります。

  キュー作成ファイルに対して同一のキュー名称を指定した場合は，後ろに記述された同一のキュー名称を含むキュー定義文が無視され，処理が続行されます。

(2)　キュー作成ファイルの記述例

記述例 1

```
hrmmkque queue1;
```

記述例 2

```
hrmmkque -e 1000    # 有効期限は1000秒（約16分）を指定
-n 65535            # 最大メッセージ数は-nオプションの最大値を指定
queue2;
```

(3)　キュー作成ファイルの作成規則

- 定義の先頭には，hrmmkque を入力してください。行の先頭でも途中でも差し支えありません。
- 定義の終端には，セミコロン（;）を指定してください。
- hrmmkque からセミコロン（;）までが定義と解釈されます。
- 定義の中で改行やコメントを挿入できます。ただし，文字列の途中で改行はできません。
- シャープ（#）で始まる行はコメントと見なされます。行の途中にシャープ（#）がある場合は，シャープ（#）以降から行の最後までがコメントと見なされます。
- コメントに相当しない位置で，かつキュー定義文に含まれない位置で入力された，"hrmmkque"，シャープ（#），半角スペース，タブ，および改行以外の文字は，すべて不正と見なされます。

参考

キューを削除する場合，削除するキューのキュー定義文を削除するか，または先頭にシャープ（#）を付けてコメントにします。

(4) キュー作成ファイルの作成時の注意事項

- 次の場合は，Reliable Messaging が閉塞します。
  - キュー作成ファイルの読み込みに失敗した場合
  - 指定されたパスにキュー作成ファイルがない場合
  - キュー作成ファイル読み込み時にメモリ不足が発生した場合
  - キュー作成ファイルに不正なキュー定義文がある場合
  - キュー作成ファイルに"hrmmkque"と対になるセミコロンがない場合
  - キュー作成ファイルに有効なキュー定義文が存在しない場合
  - キュー作成ファイルの読み込み処理中に何らかの入出力例外が発生した場合
- オプションフラグ，フラグ引数，またはキュー名に不正がある場合，該当の定義文を無視して処理が続行されます。
- -x オプションで指定したキューの表示名は，キュー定義ファイルを使用しない場合に，指定した表示名で JNDI ネーミングサービスに登録されます。キュー定義ファイルを使用する場合は，指定した表示名で JNDI ネーミングサービスに登録されません。

3.5.2 キュー定義ファイルの作成（非永続版リソースアダプタの場合）

キュー定義ファイルは，JNDI ネーミングサービスに登録するキュー表示名と実際のキュー名を関連づけるファイルです。テキストファイルで作成して，任意のディレクトリに格納してください。キュー定義ファイルのフルパスを QueueConfigFileName プロパティに指定すると，Reliable Messaging の開始時に，定義されたキュー表示名が JNDI ネーミングサービスに登録されます。

なお，キュー作成時の表示名を利用する場合，キュー定義ファイルの作成は不要です。

(1) キュー定義ファイルの記述形式

キュー定義ファイルの記述形式について，次に示します。

記述形式

```
QueueImplClass = jp.co.Hitachi.soft.reliablemessaging_np.ra.jms.QueueImpl※1
Queue.通番※2.DisplayName = キュー表示名※3
Queue.通番.QueueName = キュー名※4
```

注※1

ユーザの環境に依存しない固定の値です。

永続版リソースアダプタの場合と指定値が異なるので注意してください。

注※2

通番には 1～20480 の範囲の値を指定してください。

注※3

キュー表示名として任意の名称を指定します。

キュー表示名は JNDI ネーミングサービスに"<Reliable Messaging の表示名>__<キュー表示名>__que"の形式で登録されます。なお，"<Reliable Messaging の表示名>__<キュー表示名>__que"の形式の文字列の長さが 256 文字を超えないようにしてください。

Reliable Messaging の表示名は，Connector 属性ファイルの<display-name>タグの指定値をサニタイズ（半角英数字以外をアンダスコア（_）に変更）した値です。<display-name>タグの初期値は"Cosminexus_Reliable_Messaging"です。Connector 属性ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

<キュー表示名>には，アンダスコア二つ（__）を指定しないでください。

注※4

キュー名はキュー作成ファイルの hrmmkque 定義の引数指定値を指定します。

(2) キュー定義ファイルの記述例

```
QueueImplClass = jp.co.Hitachi.soft.reliablemessaging_np.ra.jms.QueueImpl
Queue.1.DisplayName = Q1
Queue.1.QueueName = Queue1
Queue.2.DisplayName = Q2
Queue.2.QueueName = Queue2
```

(3) キュー定義ファイルの作成規則

- java.util.Properties.load()メソッドで読み取りできる形式にしてください。コメントや継続行の扱いも同様です。
- QueueImplClass の指定がない場合および値が指定されていない場合はデプロイ時にエラーが発生します。
- 一つのキューについて DisplayName および QueueName の両方を指定してください。
- 同一のプロパティを複数記述した場合は，どの値が有効になるか保証しません。
- 通番は 1 から指定してください。通番 1 のキュー定義がない場合はデプロイ時にエラーが発生します。
- Queue.n.DisplayName の指定がない場合および値が指定されていない場合，通番が n-1 番目のキューまでが有効になります。通番 n 以降のキュー定義は無視されます。
- QueueName の指定がない，または値が指定されていないキューは無視されます。
- キュー定義が一つもない場合はフォーマット不正によってデプロイ時にエラーが発生します。

- 表示名が重複した場合，通番の大きい方が有効になります。
- 通番の最大値より大きい値のキュー定義は無視されます。
- DisplayName に指定できる文字は，半角英字（A～Z，a～z）半角数字（0～9）および半角のアンダスコア（_）です。これら以外の文字を指定したキューは無視されます。
- QueueName に指定する値は，大文字と小文字を hrmmkque 定義の引数指定値と一致させてください。

3.5.3 Reliable Messaging のプロパティ定義（非永続版リソースアダプタの場合）

Reliable Messaging のプロパティを定義します。Reliable Messaging をデプロイしたあとでも実行できます。なお，設定済みの Reliable Messaging のプロパティを変更する場合は，該当する Reliable Messaging を停止した状態で実行してください。

(1) 編集する属性ファイル

Reliable Messaging で提供する Connector 属性ファイルのテンプレートを任意のディレクトリにコピーして，コピーしたファイルを編集します。

Connector 属性ファイルのテンプレートは，次のディレクトリに格納されています。

```
%HRMDIR%¥conf¥rmnp_prop.xml
```

なお，cjgetresprop コマンドによって取得した Connector 属性ファイルを使用することもできます。

注意事項

テンプレートからコピーした Connector 属性ファイルに ASCII 文字以外を使用する場合は，次のどちらかの対処が必要です。

- UTF-8 形式で保存する。
- 編集したファイルの文字エンコーディング形式に従って，ファイルの先頭行に次に示す encoding 宣言を追加する。

  <?xml version="1.0" encoding="＜文字エンコーディング＞"?>

(2) 編集する属性設定項目

Reliable Messaging のプロパティ設定項目を次に示します。

- リソースアダプタの一般情報
- コンフィグレーションプロパティ
- 実行時プロパティ

(a)　リソースアダプタの一般情報

設定できる Reliable Messaging の一般情報属性（<outbound-resourceadapter>タグ）の設定項目を次に示します。

項目	必須	対応するタグ
トランザクションサポートのレベル	○	<transaction-support>
再認証のサポート有無※	○	<reauthentication-support>

（凡例）　○：必須

注※　必ず false を指定してください。

プロパティの設定項目の説明については，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

(b)　コンフィグレーションプロパティ

Reliable Messaging のコンフィグレーションプロパティ（<outbound-resourceadapter> - <connection-definition> - <config-property>タグ）の設定項目を次に示します。

項目	対応するタグ
コンフィグレーションプロパティ名	<config-property-name>
コンフィグレーションプロパティのデータ型	<config-property-type>
コンフィグレーションプロパティの値	<config-property-value>※

注※　コンフィグレーションプロパティの値をクリアする場合

<config-property-value>タグだけを指定して，値を指定しないでください。

<config-property-value>タグ自体が指定されていない場合は，そのプロパティの値は変更されないで，すでに設定されている値がそのまま有効になります。

定義するコンフィグレーションプロパティの数だけ上記の設定を繰り返してください。

Reliable Messaging のコンフィグレーションプロパティの設定内容については，「6.　コンフィグレーションプロパティ」を参照してください。

(c)　実行時プロパティ

Reliable Messaging の実行時プロパティ（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <property>タグ）の設定項目を次に示します。

項目	対応するタグ
プロパティ名	<property-name>
プロパティのデータ型	<property-type>
プロパティの値	<property-value>

定義するプロパティの数だけ，上記の設定を繰り返してください。

実行時プロパティ名（<property-name>）およびコネクションプールの動作と注意事項については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の DB Connector のプロパティ定義に関する内容を参照してください。

(3) コンフィグレーションプロパティの設定例

Reliable Messaging のコンフィグレーションプロパティの設定例を示します。コンフィグレーションプロパティの各項目の詳細については，「6.2 Reliable Messaging のコンフィグレーションプロパティの詳細説明」を参照してください。

表 3-17 Reliable Messaging のコンフィグレーションプロパティの設定例（非永続版リソースアダプタの場合）

項番	プロパティ名	設定例
1	RMSystemName	＜システム名＞
2	QueueMakeFileName	＜キュー作成ファイルの場所＞
3	QueueConfigFileName	指定しない
4	RMSweepTimerInterval	600
5	RMPassByReference	false
6	RMMaxDeliveryNum	10
7	RMMethodTraceLevel	1
8	RMLogTraceFileNum	2
9	RMLogTraceFileSize	2097152

3.5.4 Reliable Messaging のインポート

次に示すコマンドを実行して Reliable Messaging をインポートします。

cjimportres コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

実行形式

```
cjimportres ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -type rar -f ＜ファイルパス＞
```

実行例

```
cjimportres MyServer -type rar -f ”C:¥Program Files¥Hitachi¥Cosminexus¥RM¥lib¥reliablemessagingNP.rar”
```

＜ファイルパス＞には，Reliable Messaging の RAR ファイル（reliablemessagingNP.rar）を指定してください。RAR ファイルは，次のディレクトリに格納されています。

```
%HRMDIR%¥lib
```

3.5.5　Reliable Messaging のプロパティ設定

次に示すコマンドを実行して，Reliable Messaging のプロパティを設定します。

実行形式

```
cjsetresprop ［＜サーバ名称＞］ -type rar -resname ＜Reliable Messagingの表示名＞ -c ＜Connector属性ファイルパス＞
```

実行例

```
cjsetresprop MyServer -type rar -resname Cosminexus_Reliable_Messaging -c rmnp_prop.xml
```

注意事項

Reliable Messaging をデプロイしたあとでプロパティを定義する場合は，cjgetrarprop コマンドと cjsetrarprop コマンドを使用してください。コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

3.5.6　Reliable Messaging のデプロイ

Reliable Messaging は，デプロイすると J2EE リソースアダプタとして使用できます。

サーバ管理コマンドでインポートした Reliable Messaging をデプロイすると，その J2EE サーバ上で動作するすべての J2EE アプリケーションから使用できるようになります。なお，デプロイしたあとで，プロパティを定義することもできます。デプロイ後に定義する場合は，該当する Reliable Messaging を停止した状態で実行してください。プロパティを定義する方法については，「3.5.3　Reliable Messaging のプロパティ定義（非永続版リソースアダプタの場合）」を参照してください。

次に示すコマンドを実行して，Reliable Messaging をデプロイします。

実行形式

```
cjdeployrar ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -resname ＜Reliable Messagingの表示名＞
```

実行例

```
cjdeployrar MyServer -resname Cosminexus_Reliable_Messaging
```

cjdeployrar コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

3.5.7 Reliable Messaging の運用前の準備（非永続版リソースアダプタの場合）

Reliable Messaging の運用前の準備について説明します。

(1) Reliable Messaging の接続テスト

Reliable Messaging に設定した内容が正しいかどうか，接続テストによって検証します。

(a) 接続テストで確認できる項目

接続テストの成功によって，次に示す項目を確認できます。

- RMSystemName プロパティのフォーマットが正しいこと。

(b) 接続テストの手順

1. 次に示すコマンドを実行して Reliable Messaging の接続テストを実施します。

   実行形式

   ```
   cjtestres ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -type rar -resname ＜Reliable Messagingの表示名＞
   ```

   実行例

   ```
   cjtestres MyServer -type rar -resname Cosminexus_Reliable_Messaging
   ```

   なお，接続に失敗した場合には，出力されるメッセージを基にエラーに対処し，再度接続テストを実施してください。

cjtestres コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

注意事項

一度接続テストをした Reliable Messaging は，J2EE サーバを再起動するまで削除できません。Reliable Messaging を削除する場合は，Reliable Messaging を停止してから，J2EE サーバを再起動してください。

(c) 接続テストの失敗事例

接続テストに失敗する事例を次に示します。

- Component Container 側でエラーを検知した場合

  接続テストでは Reliable Messaging と Component Container の両方でチェックを実行します。そのため，Component Container 側でエラーを検知すると接続テストに失敗する場合もあります。

接続テストに成功しても Reliable Messaging の起動時に接続に失敗する場合があります。事例を次に示します。

- QueueConfigFileName プロパティや QueueMakeFileName プロパティに指定したファイルパスが不正のとき，または指定したファイルの内容が不正のとき
- ロガーの初期化に失敗したとき，またはコマンドのネーミングサービスの登録に失敗したとき

(2) Reliable Messaging の開始

cjstartrar コマンドを実行して，Reliable Messaging を開始します。

実行形式

```
cjstartrar ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -resname ＜Reliable Messagingの表示名＞
```

実行例

```
cjstartrar MyServer -resname Cosminexus_Reliable_Messaging
```

cjstartrar コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

注意事項

- J2EE アプリケーション中の J2EE リソースが Reliable Messaging を参照している場合は，Reliable Messaging を開始してから，J2EE アプリケーションを開始してください。
- 一度開始したリソースアダプタは，J2EE サーバを再起動するまで削除できません。リソースアダプタを削除する場合は，リソースアダプタを停止してから，J2EE サーバを再起動してください。

3.6　システム構築時の注意事項

システム構築時の注意事項について説明します。

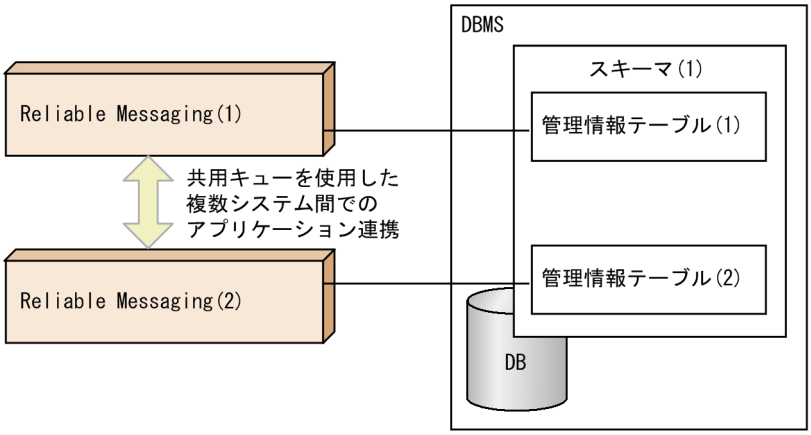
3.6.1　DBMS の設定をする場合

DBMS の初期設定時の注意事項について説明します。

- 共用キューを使用して複数システム間でのアプリケーション連携をする場合には，同一スキーマ定義内に送信側システムの管理情報テーブルと受信側システムの管理情報テーブルを定義する必要があります。

  共用キューを使用して複数システム間でのアプリケーション連携をする場合の管理情報テーブルを次の図に示します。

  図 3-2　共用キューを使用して複数システム間でのアプリケーション連携をする場合の管理情報テーブル



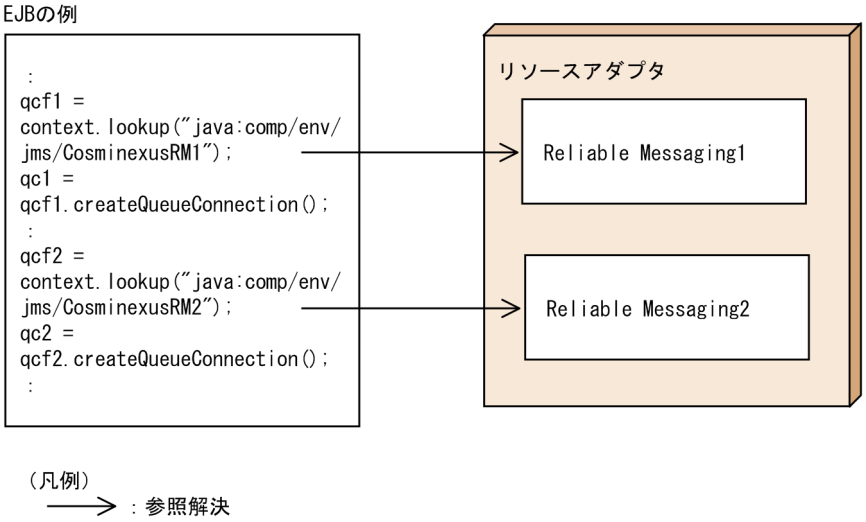
- 共用キューの場合，送信側（または受信側）システムの管理情報テーブルを定義したあとで受信側（または送信側）システムの管理情報テーブルを定義するときは，DBMS の初期設定は不要です。受信側（または送信側）システムでは，送信側（または受信側）システムと同じ接続ユーザ名を使用します。
- 各 DB で使用する言語モードは，DB Connector for Reliable Messaging などの上位のソフトウェアの言語モードと合わせてください。

3.6.2　複数デプロイをする場合

Reliable Messaging はリソース使用量が多いため，複数デプロイは推奨しません。複数デプロイをする場合の注意事項を次に示します。

- 複数の Reliable Messaging 間で DB 上の管理情報テーブルを共有しないようにしてください。
- アプリケーションがネーミングサービスから lookup メソッドでオブジェクトを取得するときのオブジェクト名称によって，稼働する Reliable Messaging を切り替えます。J2EE 環境での複数デプロイの例を次の図に示します。

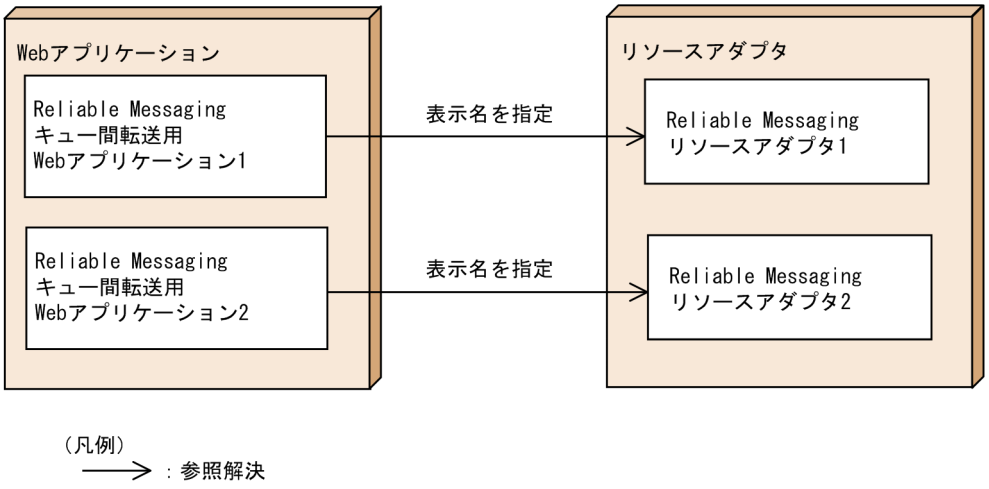
図 3-3　J2EE 環境での複数デプロイの例



- Reliable Messaging を複数デプロイした場合，Reliable Messaging ごとに別々のクライアント定義ファイルを指定してください。クライアント定義ファイルの詳細については，「3.4.13(1)　SOAP 通信基盤の設定」を参照して確認してください。
- Reliable Messaging を複数デプロイする場合，キュー間転送用 Web アプリケーションのコンテキストルートを一意にするように設定してください。
- キュー間転送用 Web アプリケーションとリソースアダプタの関連づけは，1 対 1 にしてください。つまり，複数のキュー間転送用 Web アプリケーションを一つのリソースアダプタに関連づけることはできません。

  Component Container では，キュー間転送用 Web アプリケーションのカスタマイズ時に，関連するリソースアダプタの表示名を指定することで関連づけます。J2EE 環境でのキュー間転送用 Web アプリケーションとリソースアダプタの関連づけを次の図に示します。

図 3-4　J2EE 環境でのキュー間転送用 Web アプリケーションとリソースアダプタの関連づけ



- J2EE サーバ内に異なるバージョンの Reliable Messaging を同時にインポートすることはできません。異なるバージョンを同時にインポートした場合，不正な動作をする可能性があります。

3.6.3　OutOfMemory 発生時の動作に関する設定を行う場合

Reliable Messaging を使用する場合，Developer's Kit for Java の OutOfMemory 発生時強制終了機能を有効にしてください。

具体的には J2EE サーバ用オプション定義ファイル(usrconf.cfg)で add.jvm.arg=-XX:+HitachiOutOfMemoryAbort を設定してください。

また，OutOfMemory ハンドリング機能（-XX:+HitachiOutOfMemoryHandling）と組み合わせて使用した場合でも，Reliable Messaging 内で OutOfMemory が発生すると，J2EE サーバは強制終了します。

3.6.4　ObjectMessage のペイロードにユーザが定義したクラスのオブジェクトを格納する場合

ObjectMessage のペイロードにユーザが定義したクラスのオブジェクトを格納する場合，アプリケーションにユーザが定義したクラスを含める必要があります。アプリケーションにユーザ定義クラスを含めない場合，Application Server の J2EE サーバ用オプション定義ファイルにユーザが定義したクラスのクラスパスを指定する必要があります。「2.5.1(3)　JMS メッセージのペイロード」を参照してください。

3.6.5　同一サーバ上で Reliable Messaging を複数デプロイする場合

同一サーバ上で Reliable Messaging を複数デプロイする場合，またはキュー間転送の通信設定を変更する場合は，Reliable Messaging ごとに別々のクライアント定義ファイルを指定する必要があります。詳細については，「3.4.13(1)(a)　クライアント定義ファイルの使用方法」を参照してください。

3.6.6　PRF トレースファイルを利用する場合

PRF トレースファイルを利用するに場合に必要な設定については，永続版リソースアダプタの場合は「4.4 PRF トレースファイルの運用（永続版リソースアダプタの場合）」を，非永続版リソースアダプタの場合は「5.3　PRF トレースファイルの運用（非永続版リソースアダプタの場合）」を参照してください。

4 Reliable Messaging（永続版リソースアダプタ）の運用

Reliable Messaging のアプリケーションを開始するには，あらかじめ Reliable Messaging を開始しておく必要があります。Reliable Messaging を停止するには，あらかじめアプリケーションを停止しておく必要があります。

この章では，永続版リソースアダプタを使用する場合の，Reliable Messaging およびアプリケーションの開始と停止，管理情報テーブルを保存した DB の運用などについて説明します。

4.1 Reliable Messaging とアプリケーションの開始と停止（永続版リソースアダプタの場合）

永続版リソースアダプタを使用する場合の，Reliable Messaging とアプリケーションの開始手順および停止手順について説明します。

4.1.1 Reliable Messaging の開始（永続版リソースアダプタの場合）

Reliable Messaging を開始するには，あらかじめプロパティのカスタマイズとデプロイを完了させている必要があります。詳細については，「3.4.5 Reliable Messaging のプロパティ定義（永続版リソースアダプタの場合）」を参照してください。また，Reliable Messaging の開始時に使用するサーバ管理コマンドについては，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」を参照してください。

Reliable Messaging を開始する手順を次に示します。

1. DBMS の開始

   HiRDB または Oracle を開始します。

2. J2EE サーバの開始

   Application Server を開始します。また，アプリケーションの要件に合わせて，必要な Application Server のサービスを開始してください。

3. DB Connector for Reliable Messaging の開始

   サーバ管理コマンドを使用して，リソースアダプタとして DB Connector for Reliable Messaging を開始します。

4. Reliable Messaging の開始

   サーバ管理コマンドを使用して，リソースアダプタとして Reliable Messaging を開始します。

注意

- サーバ管理コマンドで Reliable Messaging が開始状態になっても，実際には開始中状態のことがあります。このときにアプリケーションを開始すると Reliable Messaging がコネクションを取得する契機でエラーが発生します。ただし，Message-driven Bean の場合は，Reliable Messaging が開始するまでアプリケーションの開始処理が完了しません。開始処理が完了し KFRM01009-I（メッセージログの出力レベルに関係なく出力します）が出力されるのを待ってからアプリケーションを開始してください。開始中状態の詳細については，「4.1.4 Reliable Messaging の状態遷移（永続版リソースアダプタの場合）」を参照してください。
- Connector 属性ファイルの<transaction-support>タグに LocalTransaction または NoTransaction を指定した場合，サーバ管理コマンドで Reliable Messaging が開始状態になって

も，実際には開始処理に失敗していることがあります。このときにアプリケーションを開始すると Reliable Messaging がコネクションを取得する契機でエラーが発生します。

- Reliable Messaging の開始中は，Reliable Messaging が使用しているキュー定義ファイルを削除したり，読み込みができなくなるようなアクセス権限の変更は行わないでください。Reliable Messaging の再開始時にキュー定義ファイルが読み込めない状態であった場合，データベースに未決着のトランザクションが滞留してしまい，Reliable Messaging が正常に開始できなくなります。
- Reliable Messaging のリソースアダプタを開始するときに，OutOfMemoryError が発生し KDJE48571-E が出力された場合，Reliable Messaging の内部状態が不正になる可能性があります。この場合 Application Server を再起動してください。

4.1.2　アプリケーションの開始（永続版リソースアダプタの場合）

アプリケーションを開始するには，あらかじめ Reliable Messaging が開始されている必要があります。各手順で使用するサーバ管理コマンドおよび各属性ファイルの詳細については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」およびマニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

アプリケーションを開始する手順を次に示します。

1. キューの作成

   アプリケーションで使用するキューを作成していない場合は，hrmmkque コマンドを使用してキューを作成します。

   キューの作成の詳細については，「4.2　キューの運用（永続版リソースアダプタの場合）」を参照してください。

2. アプリケーションのインポート

   サーバ管理コマンドを使用して，Reliable Messaging のアプリケーションをインポートします。

3. J2EE アプリケーションのプロパティ設定

   テキストエディタを使って，Session Bean 属性ファイル，Entity Bean 属性ファイル，WAR 属性ファイル，または MessageDrivenBean 属性ファイルを編集して，リソースアダプタのリファレンス定義，リソース環境のリファレンス定義およびメッセージ参照の設定項目を設定します。

4. 実行状態への移行

   次に示す場合は，hrmstart コマンドを入力して Reliable Messaging を管理状態から実行状態に移行してください。

   - 初めて Reliable Messaging を開始した場合
   - 前回のオンラインで管理状態のときに Reliable Messaging を停止した場合

   Reliable Messaging の状態については，hrmlsstat コマンドによって確認できます。状態遷移については，「4.1.4　Reliable Messaging の状態遷移（永続版リソースアダプタの場合）」を参照してください。

5. アプリケーションの開始

サーバ管理コマンドを使用して，アプリケーションを開始します。

4.1.3 Reliable Messaging の停止（永続版リソースアダプタの場合）

サーバ管理コマンドを使用して，アプリケーションを停止させてから，Reliable Messaging を停止させます。サーバ管理コマンドについては，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」を参照してください。

Reliable Messaging を停止する手順を次に示します。

1. アプリケーションの停止

サーバ管理コマンドを使用して，アプリケーションを停止します。一つの Reliable Messaging を利用する複数のアプリケーションがある場合はすべて停止します。キュー間転送用 Web アプリケーションも含みます。

2. Reliable Messaging の停止

サーバ管理コマンドを使用して，Reliable Messaging を停止します。

3. DB Connector for Reliable Messaging の停止

サーバ管理コマンドを使用して，DB Connector for Reliable Messaging を停止します。

4. J2EE サーバの停止

アプリケーションサーバシステムを使用しない場合は，J2EE サーバと Application Server のサービスを停止します。

5. DBMS の終了

DBMS を使用しない場合は，HiRDB または Oracle を終了します。

注意

- Reliable Messaging のリソースアダプタを停止するときに，OutOfMemoryError が発生し KDJE42093-E が出力された場合，Reliable Messaging の内部状態が不正になる可能性があります。この場合 Application Server を再起動してください。

4.1.4 Reliable Messaging の状態遷移（永続版リソースアダプタの場合）

アプリケーションサーバシステムにとって，Reliable Messaging には未デプロイ状態，デプロイ済み状態，およびリソースアダプタ開始状態の三つの状態があります。また，リソースアダプタ開始状態の Reliable Messaging の内部状態として，実行状態，管理状態，閉塞状態および開始中状態があります。

Reliable Messaging の状態について次に説明します。

実行状態

アプリケーションにサービスを提供する状態です。管理状態から hrmstart コマンドの入力によって移行します。

管理状態

Reliable Messaging が提供するコマンドを入力して，キューを作成したり，メッセージを削除したりするためのシステム管理用の状態です。実行状態から hrmstop コマンドの入力によって移行します。

閉塞状態

Reliable Messaging 内部のメモリ状態不正などによって，処理の続行ができなくなった状態です。実行状態，管理状態，または開始中状態から移行します。この状態では，アプリケーションにサービスは提供されません。

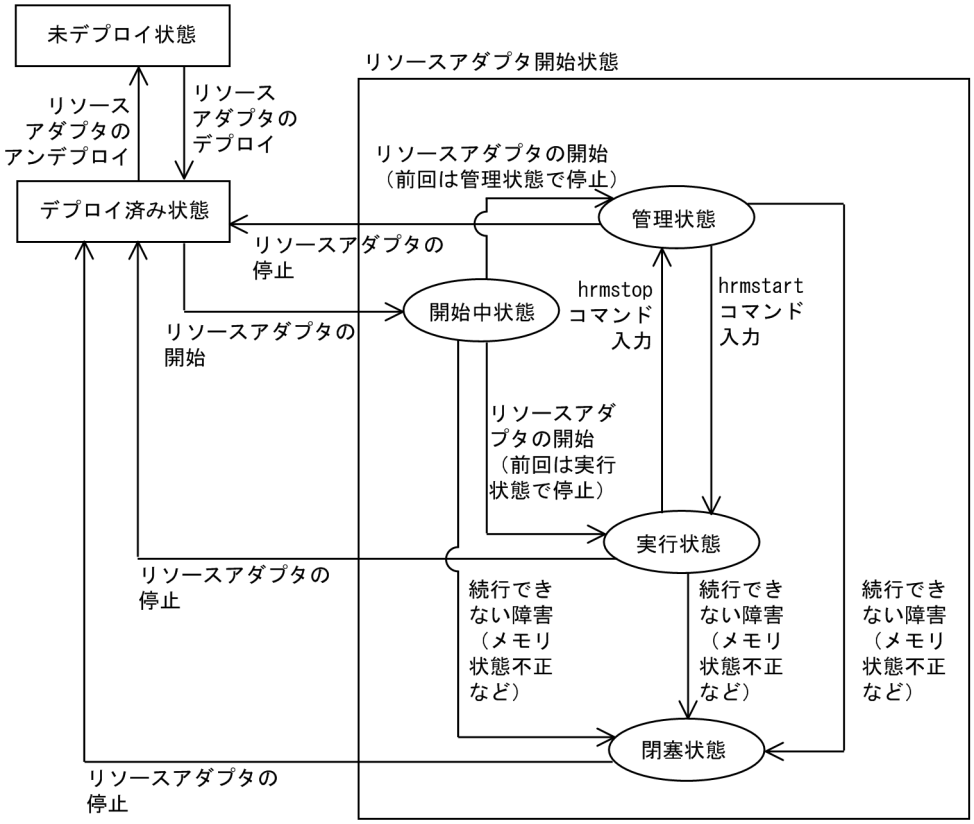
開始中状態

リソースアダプタを開始して，前回の状態によって管理状態または実行状態に移行するまでの一時的な状態です。この状態では，DB 上に格納されているキューやメッセージ情報をメモリ上に復元します。復元に掛かる時間は，DB に格納されているメッセージの数に比例します。

(1) 状態遷移

Reliable Messaging の状態遷移を次の図に示します。

図 4–1 Reliable Messaging の状態遷移



リソースアダプタを開始した直後は，前回のオンラインで Reliable Messaging を停止したときの内部状態に移行します。また，コマンドによっては特定の内部状態のときだけに入力できます。詳細については，「8.2　コマンドの一覧」を参照してください。

(2)　管理状態への移行処理

管理状態に移行するときは，あらかじめアプリケーションを停止させてください。

hrmstop コマンドを入力すると，Reliable Messaging は実行状態から管理状態へ移行します。移行時には，仕掛かり中のメッセージ処理に影響を与えないように，次に示す処理が実行されます。

- 新しいメッセージ受信を停止し，仕掛かり中のメッセージ受信処理が完了するまで待ちます。
- 新しいメッセージ送信を停止し，仕掛かり中のメッセージ送信処理が完了するまで待ちます。
- キュー間転送での新しい SOAP メッセージの受信を停止し，仕掛かり中の SOAP メッセージの受信処理が完了するまで待ちます。
- キュー間転送での新しい SOAP メッセージの送信を停止し，仕掛かり中の SOAP メッセージの送信処理で応答が返されるのを待ちます。
- 共用キュー連携時は新しいイベントの受信を停止し，仕掛かり中のメッセージ受信処理が完了するまで待ちます。
- アプリケーションからの新しい API 呼び出しに対して例外を返し，仕掛かり中の API 処理が完了するまで待ちます。なお，仕掛かり中のトランザクションがあっても，Reliable Messaging はトランザクションをロールバックしません。

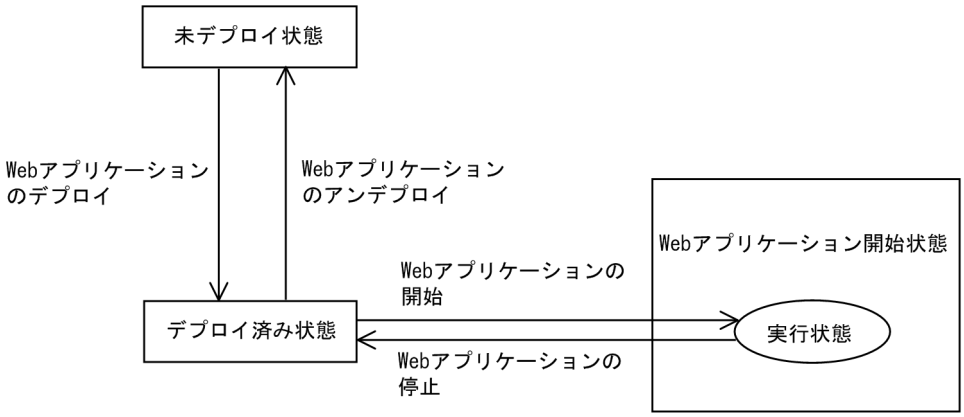
(3)　キュー間転送用 Web アプリケーションの状態遷移

Reliable Messaging のキュー間転送用 Web アプリケーションを J2EE アプリケーションとしてデプロイします。J2EE コンテナから見たキュー間転送用 Web アプリケーションの状態として，未デプロイ状態，デプロイ状態，および開始状態の三つの状態があります。また，開始状態のキュー間転送用 Web アプリケーションの内部状態として，実行状態があります。

キュー間転送用 Web アプリケーションが開始状態に遷移できるのは，Reliable Messaging がリソースアダプタ開始状態であることが前提となります。

キュー間転送用 Web アプリケーションの状態遷移を次の図に示します。

図 4-2　キュー間転送用 Web アプリケーションの状態遷移



4.2　キューの運用（永続版リソースアダプタの場合）

ローカルキュー，共用キュー，転送キュー，およびデッドメッセージキューの運用方法について説明します。

4.2.1　ローカルキューによるシステム内アプリケーション間連携

自システム内のアプリケーション間で，ローカルキューを介してメッセージを送受信する手順を次に説明します。

1. Reliable Messaging のリソースアダプタの開始

2. 管理状態への移行

   Reliable Messaging を管理状態にします。hrmlsstat コマンドで Reliable Messaging の状態を確認してください。実行状態の場合は hrmstop コマンドを入力すると管理状態に移行します。

3. ローカルキューの作成

   hrmmkque コマンドの-t オプションに local を指定して，メッセージを受信するためのローカルキューを作成します。hrmmkque コマンドについては，「8.3.17　hrmmkque（ローカルキューの作成）」を参照してください。

4. 実行状態への移行

   hrmstart コマンドを入力して，Reliable Messaging を実行状態にします。

4.2.2　共用キューによるシステム間連携

(1)　共用キューによるシステム間連携の手順

共用キューによるシステム間連携の手順を次に説明します。共用キューを使用する場合，共用キューのバージョンについてあらかじめ確認してください。共用キューのバージョンについては，「4.2.2(2)　Reliable Messaging のバージョンと共用キューのバージョン」を参照してください。

受信側システムの手順

1. Reliable Messaging のリソースアダプタの開始

2. 管理状態への移行

   Reliable Messaging を管理状態にします。hrmlsstat コマンドで Reliable Messaging の状態を確認してください。実行状態の場合は hrmstop コマンドを入力すると管理状態に移行します。

3. 受信用共用キューの作成

hrmmkque コマンドの-t オプションに shr_receive を指定して，メッセージを受信するための受信用共用キューを作成します。hrmmkque コマンドについては，「8.3.18　hrmmkque（受信用共用キューの作成）」を参照してください。

4. 実行状態への移行
   hrmstart コマンドを入力して，Reliable Messaging を実行状態にします。

送信側システムの手順

1. Reliable Messaging のリソースアダプタの開始

2. 管理状態への移行
   Reliable Messaging を管理状態にします。hrmlsstat コマンドで Reliable Messaging の状態を確認してください。実行状態の場合は hrmstop コマンドを入力すると管理状態に移行します。

3. 送信用共用キューの作成
   hrmmkque コマンドの-t オプションに shr_send を指定して，送信用共用キューを作成します。hrmmkque コマンドについては，「8.3.19　hrmmkque（送信用共用キューの作成）」を参照してください。

4. 実行状態への移行
   hrmstart コマンドを入力して，Reliable Messaging を実行状態にします。

5. ユーザアプリケーションの開始
   ユーザアプリケーションを開始して，共用キューによるシステム間連携を行います。

(2)　Reliable Messaging のバージョンと共用キューのバージョン

Reliable Messaging のバージョンによって共用キューに格納する情報が異なるため，Reliable Messaging のバージョンに対応して共用キューにもバージョンが存在します。

また，Reliable Messaging のバージョンによって使用できる共用キューのバージョンが決められています。したがって，共用キューを使用する場合は，送信側システムと受信側システムの Reliable Messaging のバージョンを統一し，必要に応じて共用キューをバージョンアップする必要があります。

Reliable Messaging のバージョンと，それに対応する共用キューのバージョンを次の表に示します。

表 4-1　Reliable Messaging のバージョンと共用キューのバージョン

Reliable Messaging のバージョン	共用キューのバージョン
01-00	0100
01-01 以降	0200

なお，共用キューのバージョン 0200 を使用して TP1/EE とシステム間連携する場合は，TP1/EE の DB キュー機能オプションは TYPE1 に指定してください。共用キューのバージョン 0100 を使用する場合は，TP1/EE の DB キュー機能オプションは TYPE0 に指定してください。

4.2.3　キュー間転送によるシステム間連携

キュー間転送を行う場合は，事前に受信側システムのあて先アドレスを確認しておいてください。

キュー間転送の手順を次に説明します。

受信側システムの手順

1. Reliable Messaging のリソースアダプタの開始
   このとき，キュー間転送用 Web アプリケーションは停止しておきます。
2. 管理状態への移行
   Reliable Messaging を管理状態にします。hrmlsstat コマンドで Reliable Messaging の状態を確認してください。実行状態の場合は hrmstop コマンドを入力すると管理状態に移行します。
3. ローカルキューの作成
   hrmmkque コマンドの-t オプションに local を指定して，メッセージを受信するためのローカルキューを作成します。hrmmkque コマンドについては，「8.3.17　hrmmkque（ローカルキューの作成）」を参照してください。
4. 実行状態への移行
   hrmstart コマンドを入力して，Reliable Messaging を実行状態にします。
5. キュー間転送用 Web アプリケーションの開始

送信側システムの手順

1. Reliable Messaging のリソースアダプタの開始
   このとき，キュー間転送用 Web アプリケーションは停止しておきます。
2. 管理状態への移行
   Reliable Messaging を管理状態にします。hrmlsstat コマンドで Reliable Messaging の状態を確認してください。実行状態の場合は hrmstop コマンドを入力すると管理状態に移行します。
3. 受信側システムのあて先アドレスの登録
   hrmmkaddr コマンドを入力して，受信側システムのあて先アドレスをあて先情報テーブルに登録します。
4. 転送キューの作成

hrmmkque コマンドの-t オプションに transmit を指定して，転送キューを作成します。受信側システムの Reliable Messaging のバージョンが 01-02 以前の場合は，hrmmkque コマンドの-i オプションに compatible を指定して，転送キューを作成します。hrmmkque コマンドについては，「8.3.20 hrmmkque（転送キューの作成）」を参照してください。

5. 実行状態への移行

hrmstart コマンドを入力して，Reliable Messaging を実行状態にします。

6. キュー間転送用 Web アプリケーションの開始

7. ユーザアプリケーションの開始

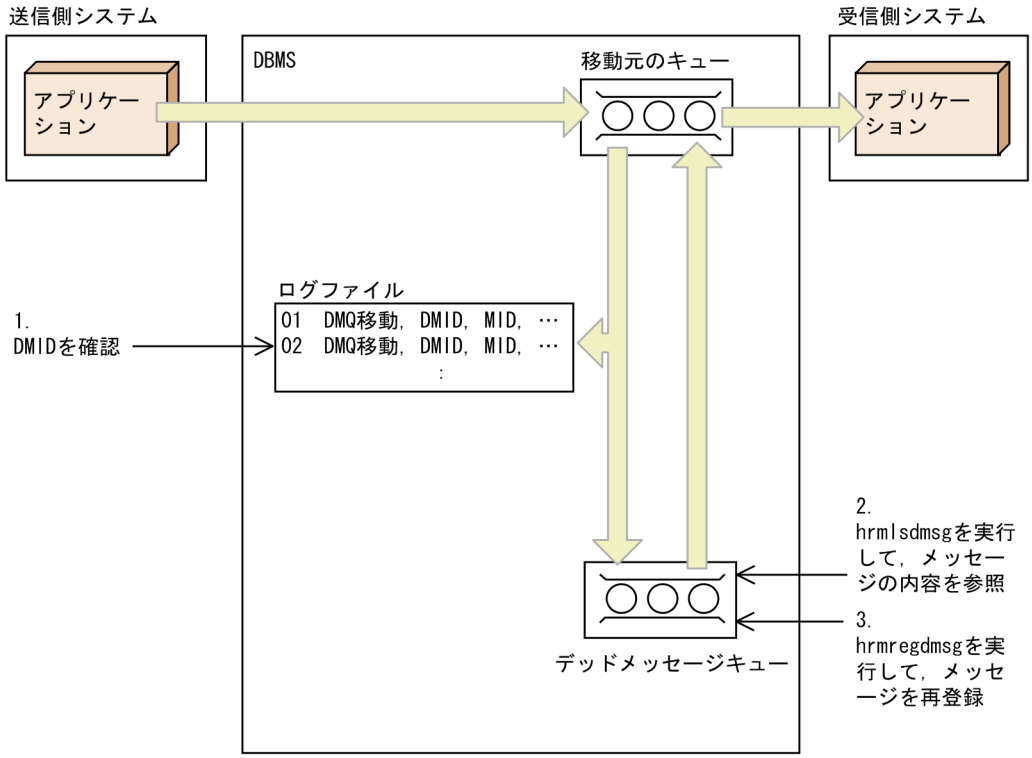
ユーザアプリケーションを開始して，キュー間転送を行います。

4.2.4　デッドメッセージキューによるデッドメッセージの再登録

デッドメッセージキューを作成する方法は，ローカルキューと同じです。ただし，RMDeadMessageQueueName プロパティにキュー名を指定する必要があります。ローカルキューの作成方法については，「4.2.1　ローカルキューによるシステム内アプリケーション間連携」を参照してください。

ここでは，デッドメッセージを再登録する場合の手順について説明します。有効期限切れ，配信回数超過，通信失敗などで，デッドメッセージキューに移動したメッセージを，移動前のキュー（ローカルキュー，受信用共用キュー，または転送キュー）に新しいメッセージとして再登録できます。デッドメッセージを再登録する場合の手順を次の図に示します。

図 4-3　デッドメッセージの再登録



1. デッドメッセージキュー移動時にログファイルに出力されるデッドメッセージ ID（DMID）を参照します。

   ログファイルは，Application Server 用メッセージログに出力されます。また，DMID については，「4.2.4(1)　DMID」を参照してください。

2. DMID を基に，hrmlsdmsg コマンドでデッドメッセージキューに移動したデッドメッセージの内容（JMS ヘッダ，プロパティ，ペイロード，デッドメッセージ固有の属性情報など）を確認します。

   hrmlsdmsg コマンドについては，「8.3.10　hrmlsdmsg（デッドメッセージの参照）」を参照してください。

3. 該当メッセージをデッドメッセージキューに移動する前のキューに再登録したい場合は，hrmregdmsg コマンドを使用します。

   該当メッセージはデッドメッセージキューから削除され，デッドメッセージキューに移動する前のキューに再登録されます。このとき，新しいメッセージとして登録されます。※hrmregdmsg コマンドについては，「8.3.21　hrmregdmsg（デッドメッセージの再登録）」を参照してください。

注※

デッドメッセージ移動前のキューに再登録する場合，ユーザが指定したメッセージの情報は引き継がれます。ただし，メッセージ ID やメッセージ有効期限などの Reliable Messaging がメッセージ登録時に設定する情報は再設定されます。また，デッドメッセージ固有の属性情報（DMID，移動前キュー名，デッドメッセージキュー移動原因，デッドメッセージキュー移動時刻）は失われます。

(1) DMID

デッドメッセージ ID（DMID）とは，デッドメッセージを識別するために設定される一意の ID です。hrmlsdmsg コマンドでデッドメッセージを参照したり，hrmregdmsg コマンドでデッドメッセージを再登録したりするときの引数として使用します。

DMID は，デッドメッセージキュー移動時のログメッセージに出力されます。また，JMS メッセージの Reliable Messaging 固有のプロパティや hrmlsdmsg コマンドの実行結果として参照できます。

(2) サーバ間転送での注意事項

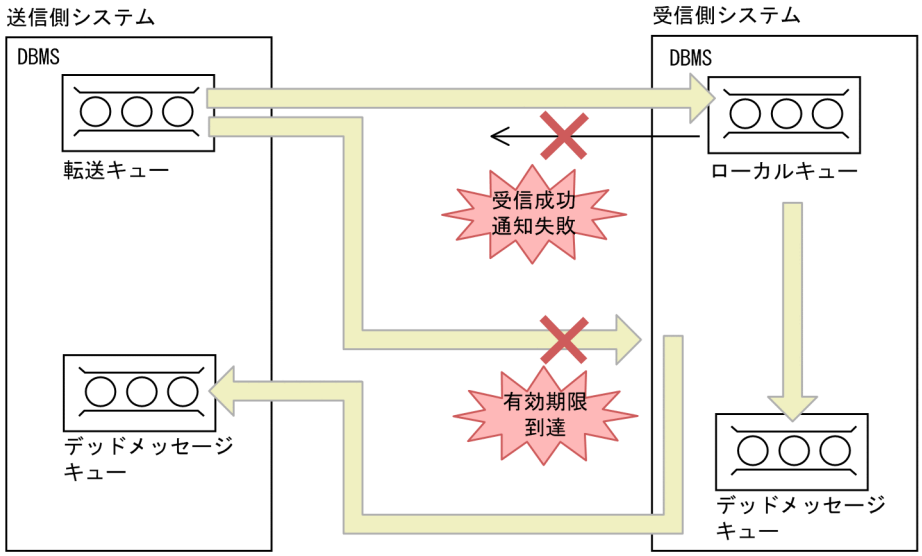
サーバ間転送時の送信側システムのデッドメッセージキューには，受信側システムに到達しているメッセージや受信側デッドメッセージキューに格納されているメッセージが存在する場合があります。

この場合，送信側システムの転送キューからデッドメッセージキュー移動したデッドメッセージを転送キューに再登録する前に，メッセージ ID などで受信側システムのメッセージ到達状態を確認して，受信側システムに到達済みのメッセージを再登録しないようにしてください。送信側システムで転送キューに再登録したメッセージは，最初に転送キューに登録したメッセージとは別のメッセージとして転送されます。そのため，送信側システムで再登録すると，受信側システムで同じ内容のメッセージが重複して受信されることになります。

受信側システムのメッセージ到達状態を確認するには，送信側システムで該当するデッドメッセージのデッドメッセージキュー移動時のメッセージログ（KFRM40012-I）に出力される DMID，グループ ID，グループ内シーケンス番号と，受信側システムで回線トレースに出力されるグループ ID，グループ内シーケンス番号とを照合してください。回線トレースについては，「9.3.5　回線トレース」を参照してください。

送信側・受信側システム両方のデッドメッセージキューに，同じメッセージが格納される場合を次の図に示します。

図 4-4　送信側・受信側システムのデッドメッセージキューに，同じメッセージが格納される場合



1. 転送キューからローカルキューへの送信は成功します。

2. 受信システムからの受信成功通知に失敗します。
3. ローカルキューに格納されたメッセージは有効期限に到達するなどして，受信側システムのデッドメッセージキューに格納されます。
4. 転送キューからメッセージが再送されますが，有効期限に到達して，失敗通知が送信側システムに通知されます。これによって，送信側システムのデッドメッセージキューに同じメッセージが格納されます。

4.2.5 キューの状態遷移（永続版リソースアダプタの場合）

キューには，通常状態，抑止状態（一部抑止状態および完全抑止状態），閉塞状態があります。キューの抑止状態は，システムを管理状態に移行して，すべてのキューに対してメッセージの送受信を抑止するのではなく，特定のキューに対してメッセージの送受信を抑止する場合などに利用します。

キューの状態について次に説明します。

通常状態
: 正常に使用できる状態です。

一部抑止状態
: メッセージの送受信が一部抑止されている状態です。指定するキューの種類によって，抑止される形態が異なります。キューの種類と抑止される形態については，「8.3.27(3) オプション」を参照してください。

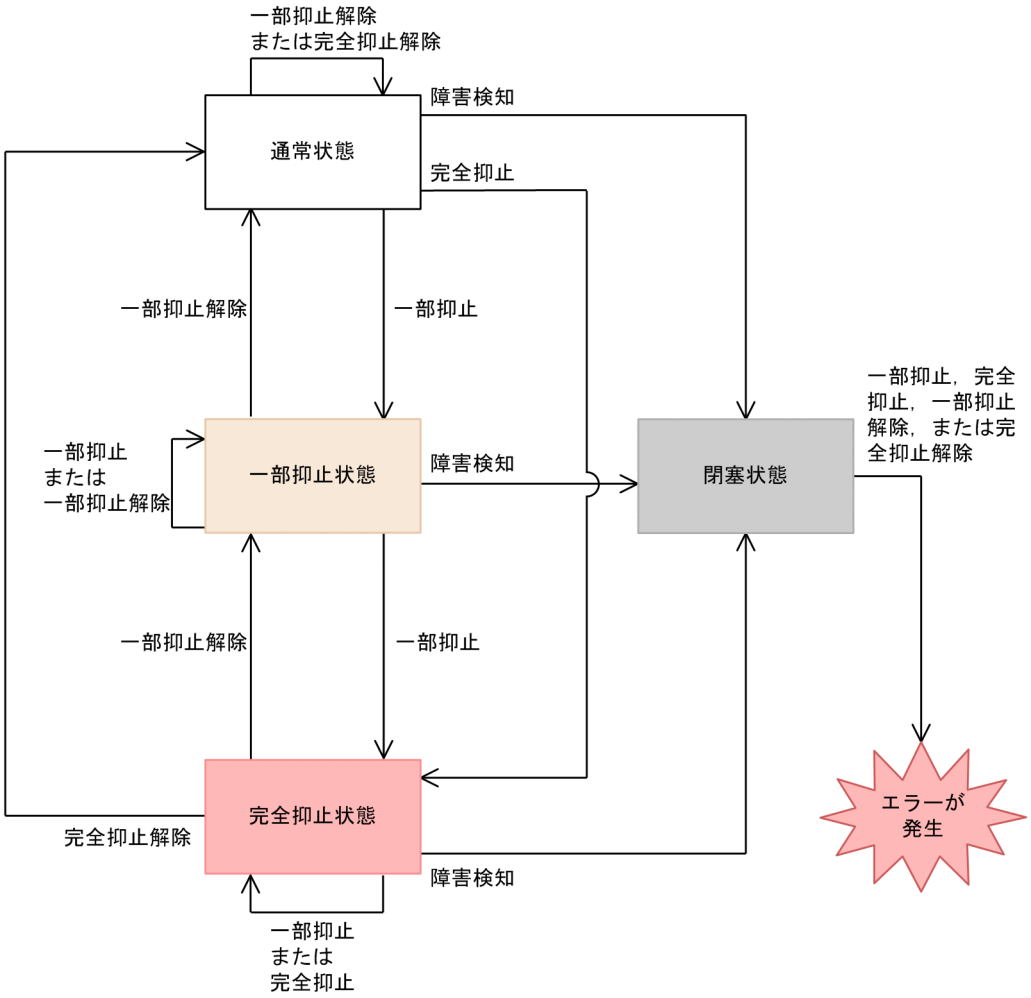
完全抑止状態
: メッセージの送受信が完全に抑止されている状態です。指定するキューの種類に応じて，抑止できる形態をすべて抑止します。

閉塞状態
: キューに障害が発生した状態です。hrmlsque コマンドおよび hrmdelque コマンドだけを受け付け，それ以外のコマンドおよびメッセージの送受信を実行できません。

キューの状態遷移を，キューの種類別に次の図に示します。

図 4-5　キューの状態遷移

●ローカルキュー，転送キューおよび受信用共用キューの場合

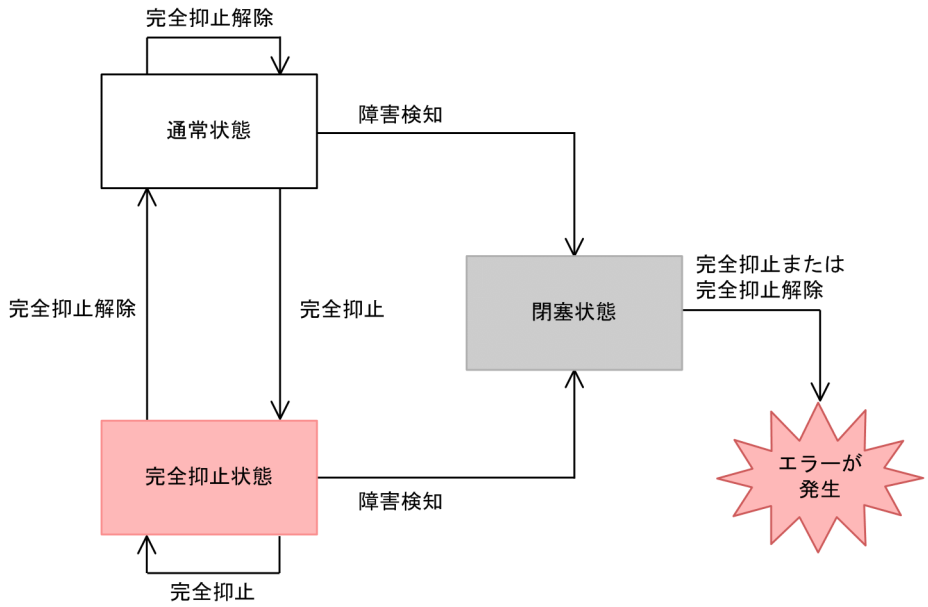


●送信用共用キューの場合

注

送信用共用キューは抑止および抑止解除状態はありません。

●デッドメッセージキューの場合



注 1

デッドメッセージキューは，抑止形態が一つであるため一部抑止および一部抑止解除状態がありません。

注 2

デッドメッセージキューが閉塞状態に移行すると，システムが閉塞します。

図中で使用されるイベントの意味は次のとおりです。

一部抑止

hrmstopque コマンドの-y オプションに ap_send，ap_receive，trs_send，trs_receive，または ap_all を指定して，実行します。

完全抑止

hrmstopque コマンドの-y オプションに all を指定して，実行します。

一部抑止解除

hrmstartque コマンドの-y オプションに ap_send，ap_receive，trs_send，trs_receive，または ap_all を指定して，実行します。

完全抑止解除

hrmstartque コマンドの-y オプションに all を指定して，実行します。

障害検知

Reliable Messaging がキューに関連する障害を検知したことを示します。

4.3　DB の運用

Reliable Messaging は，メッセージを格納するキューや管理情報を DB 上の管理情報テーブルに保存します。テーブルの移行手順や接続ユーザ名の変更手順などについて説明します。

DBMS として HiRDB または Oracle を使用できます。それぞれの DBMS で必要な設定について説明します。ここで説明していない設定内容や手順については，HiRDB または Oracle のマニュアルを参照してください。

4.3.1　管理情報テーブルの移行

DBMS の動作マシンの入れ替えなどによって，管理情報テーブルを異なる DB に移行するときの手順について，次に示します。

(1)　管理情報を格納する DB が HiRDB の場合

1. Reliable Messaging の停止

   Reliable Messaging を停止します。詳細については，「4.1.3　Reliable Messaging の停止（永続版リソースアダプタの場合）」を参照してください。

2. テーブルの移行

   HiRDB のディクショナリ搬出入ユティリティ（pdexp コマンド）とデータベース再編成ユティリティ（pdrorg コマンド）を使用して管理情報テーブルを移行します。詳細については，マニュアル「HiRDB システム運用ガイド」および「HiRDB コマンドリファレンス」を参照してください。

3. 環境設定の変更

   次のどちらかの方法で環境設定を変更します。

   - HiRDB クライアント環境変数登録ツールを使用して，移行先マシンに合わせて環境変数を変更します（Windows の場合）。または，HiRDB のクライアント環境変数グループの設定ファイルに記載している環境変数を，移行先マシンに合わせて変更します（UNIX の場合）。該当する環境変数については，「3.4.1(2)(b)　HiRDB の環境変数グループの登録」を参照してください。
   - HiRDB クライアント環境変数登録ツールを使用して，新しい環境変数グループを登録し，DB Connector for Reliable Messaging のコンフィグレーションプロパティを変更します（Windows の場合）。または，HiRDB のクライアント環境変数グループの設定ファイルを新しく作成し，DB Connector for Reliable Messaging のコンフィグレーションプロパティを変更します（UNIX の場合）。詳細については，「6.3　DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

(2)　管理情報を格納する DB が Oracle の場合

1. Reliable Messaging の停止

Reliable Messaging を停止します。詳細については，「4.1.3　Reliable Messaging の停止（永続版リソースアダプタの場合）」を参照してください。

2. テーブルの移行

エクスポート・ユーティリティおよびインポート・ユーティリティを使用して管理情報テーブルを移行します。詳細については，Oracle のマニュアルを参照してください。

3. 環境設定の変更

次のどちらかの方法で環境設定を変更します。

- DB クライアントの環境設定時に作成したネット・サービスのネットワーク情報を，移行先のマシンに合わせて変更します。DB クライアントの環境設定については，「3.4.2(2)　DB クライアントの設定」を参照してください。
- Oracle Client の環境設定時に作成したネット・サービスのネットワーク情報に従って新たにネット・サービス名を作成し，DB Connector for Reliable Messaging のコンフィグレーションプロパティを変更します。詳細については，「6.3　DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

4.3.2　管理情報テーブルの削除

DBMS の動作マシンの入れ替えや Reliable Messaging のアンインストールなどによって，管理情報テーブルを DB から削除するときの手順について，次に示します。

1. アプリケーションの停止

Reliable Messaging を使用しているアプリケーションをすべて停止します。詳細については，「4.1.3 Reliable Messaging の停止（永続版リソースアダプタの場合）」を参照してください。

2. 管理状態への移行

hrmstop コマンドを使用して Reliable Messaging を管理状態に移行します。

3. Reliable Messaging の再開始

コネクションプーリングで使用中の DB のコネクションや未決着のトランザクションが残らないように Reliable Messaging を一度停止したあと，再度開始します。

4. 全キューの削除

hrmdelque コマンドを使用して，作成済みのキューをすべて削除します。

5. Reliable Messaging の停止

Reliable Messaging を停止します。詳細については，「4.1.3　Reliable Messaging の停止（永続版リソースアダプタの場合）」を参照してください。

6. テーブルの削除

SQL ファイルを使用して，Reliable Messaging の管理情報テーブルを削除します。管理情報テーブルの削除方法を「4.3.2(1)　テーブル削除用 SQL ファイルの使用」で説明します。

なお，キューが削除されている場合も，手順 1.～手順 3.，手順 5.を実行してから手順 6.を実行してください。

(1)　テーブル削除用 SQL ファイルの使用

(a)　HiRDB の場合

DBMS として HiRDB を使用する場合，次に示す手順で管理情報テーブルを削除します。HiRDB SQL Executer の使用方法については，HiRDB SQL Executer のドキュメントを参照してください。

1. SQL ファイルの編集

テーブル削除用 SQL ファイル（%HRMDIR%¥sql¥deletetableshirdb.sql）を任意の場所にコピーしたあと，"<RMSystemName>"の部分を「3.3.1　Reliable Messaging のシステム名の決定」で決定した値に変更します。

2. SQL ファイルの実行

Windows の場合

次に示すコマンドで HiRDB SQL Executer を開始したあと，HiRDB SQL Executer の［ファイル］メニューから［ファイルから実行］を選択し，手順 1.で編集した SQL ファイルを指定して実行します。

```
pdsqlw -u 〈接続ユーザ名〉/〈パスワード〉※
       -h 〈HiRDBサーバのホスト名またはIPアドレス〉
       -n 〈HiRDBサーバのポート番号〉
```

UNIX の場合

環境変数 PDUSER※，PDHOST，PDNAMEPORT を設定したあと，次に示すコマンドで HiRDB SQL Executer から SQL ファイルを実行します。

```
pdsql　＜　〈手順1.で編集したSQLファイルのパス〉
```

注※

接続ユーザ名には，権限を付与した接続ユーザ名を指定します。詳細については，「3.4.1(1)(b) HiRDB のユーザ権限の付与」を参照してください。

(b)　Oracle の場合

DBMS として Oracle を使用する場合，次に示す手順で管理情報テーブルを削除します。SQL*Plus の使用方法については，Oracle のマニュアルを参照してください。

1. SQL ファイルの編集

テーブル削除用 SQL ファイル（%HRMDIR%¥sql¥deletetablesoracle.sql）を任意の場所にコピーしたあと，"<RMSystemName>"の部分を「3.3.1　Reliable Messaging のシステム名の決定」で決定した値に変更します。

2. GUI 版 SQL*Plus またはコマンドライン版 sqlplus の起動

システム構築時のユーザ名を使用してデータベースに接続します。

3. コマンドの実行

次の形式で実行してください。

```
start <手順1. で任意の場所にコピーしたあと編集したSQLファイルの絶対パス>
```

4.3.3　接続ユーザ名の変更

DBMS の接続ユーザ名を変更する場合は，管理情報テーブルを作り直す必要があります。作り直しの対象となるテーブルについては，「4.3.6　管理情報テーブルの一覧」を参照してください。テーブルを移行したあと，環境設定を変更します。

(1)　テーブルの移行

(a)　HiRDB の場合

Reliable Messaging を停止したあと，次に示す手順でテーブルを作り直します。各手順の詳細については，マニュアル「HiRDB システム運用ガイド」および「HiRDB コマンドリファレンス」を参照してください。

1. 定義系 SQL の作成

変更前ユーザが所有するテーブルの定義系 SQL を HiRDB の pddefrev コマンドによって生成したあと，生成した定義系 SQL ファイルから CREATE TABLE 文の認可識別子の指定を削除します。

2. RD エリアの閉塞

移行対象のデータを格納している RD エリアを HiRDB の pdhold コマンドによって閉塞します。

3. テーブルデータのアンロード

変更前ユーザが所有するテーブルのテーブルデータを HiRDB のデータベース再編成ユティリティ（pdrorg）によってアンロードします。

4. RD エリアの閉塞解除

手順 2.で閉塞した RD エリアを HiRDB の pdrels コマンドによって解除します。

5. 変更後ユーザの作成とスキーマ定義

変更後ユーザにユーザ権限を付与し，管理情報テーブル用のスキーマを定義します。詳細については，「3.4.1(1)(b)　HiRDB のユーザ権限の付与」および「3.4.1(1)(c)　HiRDB のスキーマの定義」を参照してください。

6. バックアップの取得

データベース移行中の障害発生に備えて HiRDB の pdcopy コマンドによってバックアップを取得します。バックアップを取得する RD エリアを次に示します。

- マスタディレクトリ用 RD エリア
- データディレクトリ用 RD エリア
- データディクショナリ用 RD エリア
- 移行するテーブルを格納するユーザ用 RD エリア

7. テーブルの定義

Windows の場合

次に示すコマンドで HiRDB SQL Executer を開始したあと，HiRDB SQL Executer の［ファイル］メニューから［ファイルから実行］を選択し，手順 1.で編集した定義系 SQL ファイルを指定して実行します。

```
pdsqlw -u <変更後ユーザの接続ユーザ名>/<パスワード>
       -h <HiRDBサーバのホスト名またはIPアドレス>
       -n <HiRDBサーバのポート番号>
```

UNIX の場合

変更後ユーザに合わせて環境変数 PDUSER，PDHOST，PDNAMEPORT を設定したあと，HiRDB SQL Executer から手順 1.で編集した定義系 SQL ファイルを実行します。

```
pdsql　<　<手順1.で編集した定義系SQLファイルのパス>
```

8. RD エリアの閉塞

変更後ユーザが所有するテーブルを格納している RD エリアを HiRDB の pdhold コマンドによって閉塞します。

9. テーブルデータのリロード

変更後ユーザが所有するテーブルにテーブルデータを HiRDB のデータベース再編成ユティリティ（pdrorg）によってリロードします。

10. RD エリアの閉塞解除

手順 8.で閉塞した RD エリアを HiRDB の pdrels コマンドによって解除します。

11. テーブルの削除

Windows の場合

次に示すコマンドで HiRDB SQL Executer を開始したあと，DROP TABLE 文によって変更前ユーザが所有するテーブルを削除します。

```
pdsqlw -u <変更前ユーザの接続ユーザ名>/<パスワード>
       -h <HiRDBサーバのホスト名またはIPアドレス>
       -n <HiRDBサーバのポート番号>
```

UNIX の場合

変更前ユーザに合わせて環境変数 PDUSER, PDHOST, PDNAMEPORT を設定したあと，HiRDB SQL Executer を起動させて，DROP TABLE 文により変更前ユーザが所有する表を削除します。

12. バックアップの取得

データベース移行後のバックアップを HiRDB の pdcopy コマンドによって取得します。バックアップを取得する RD エリアを次に示します。

- マスタディレクトリ用 RD エリア
- データディレクトリ用 RD エリア
- データディクショナリ用 RD エリア
- 移行したテーブルを格納するユーザ用 RD エリア

(b) Oracle の場合

各手順の詳細については，Oracle のマニュアルを参照してください。

1. Reliable Messaging の停止

Reliable Messaging を停止します。詳細については，「4.1.3 Reliable Messaging の停止（永続版リソースアダプタの場合）」を参照してください。

2. テーブルの移行

エクスポート・ユーティリティおよびインポート・ユーティリティを使用して管理情報テーブルを移行します。詳細については，Oracle のマニュアルを参照してください。

(2) 環境設定の変更

(a) HiRDB の場合

変更後ユーザに合わせて環境変数を変更します。変更する環境変数については，「3.4.1(2)(b) HiRDB の環境変数グループの登録」を参照してください。

または，新しい環境変数グループ名を登録してから，DB Connector for Reliable Messaging のコンフィグレーションプロパティの指定値を変更します。詳細については，「6.3 DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

(b) Oracle の場合

DB Connector for Reliable Messaging のコンフィグレーションプロパティの指定値を変更します。詳細については，「6.3 DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧」を参照してください。

4.3.4　DB のバックアップ

管理情報テーブルを保存する DB をバックアップする場合は，DBMS が提供するバックアップ取得機能を使用して実行します。バックアップの対象となるテーブルについては，「4.3.6　管理情報テーブルの一覧」を参照してください。

DBMS として HiRDB を使用する場合は，マニュアル「HiRDB システム運用ガイド」を参照してください。DBMS として Oracle を使用する場合は，Oracle のマニュアルを参照してください。

4.3.5　RD エリアの容量の確保

HiRDB を使用する場合，メッセージの送受信を長時間繰り返すと，HiRDB で使用できる RD エリアの容量が徐々に減少します。RD エリアの容量が減少した場合は，再編成や使用中空きページの解放を行ない，容量を確保してください。再編成や使用中空きページの解放の方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

4.3.6　管理情報テーブルの一覧

Reliable Messaging の管理情報テーブルは，メッセージを格納するキューを保存するテーブルと管理情報を保存するテーブルによって構成されます。

管理情報テーブルを使用するのに必要な，構成情報（テーブルの個数，列数，行数，名称規則，可変長文字列数，データ長，データ型，キー名，キー長）を次の表に示します。

表 4-2　管理情報テーブルの一覧（テーブルの個数，列数，行数，名称規則）

項番	種類	個数	列数※	行数	名称規則
1	システム管理情報テーブル	1	10	1	次に示す文字列を順に連結した名称です。 • <RMSystemName プロパティ指定値> • "_SYSTEMINFORMATION"
2	キュー情報テーブル	1	23	最大キュー数	次に示す文字列を順に連結した名称です。 • <RMSystemName プロパティ指定値> • "_QUEUEINFORMATION"
3	FIFO 情報テーブル	1	21	最大キュー数と最大通信用グループ数の総数	次に示す文字列を順に連結した名称です。

項番	種類	個数	列数※	行数	名称規則
					• <RMSystemName プロパティ指定値> • "_FIFOINFORMATION"
4	あて先情報テーブル	1	7	最大あて先数	次に示す文字列を順に連結した名称です。 • <RMSystemName プロパティ指定値> • "_ADDRESS_MAPPING"
5	メッセージ情報テーブル	永続キュー属性のローカルキューまたは永続キュー属性の転送キューの最大キュー数	21	永続キュー属性のローカルキューの最大メッセージ数とスイーパ実行間隔内で登録される最大メッセージ数または永続キュー属性の転送キューの最大メッセージ数の総数	次に示す文字列を順に連結した名称です。 • <RMSystemName プロパティ指定値> • "_MSG_" • <キュー名>
6	共用キュー受信用メッセージ情報テーブル	受信用共用キューの最大キュー数	3	受信用共用キューの最大メッセージ数	次に示す文字列を順に連結した名称です。 • <RMSystemName プロパティ指定値> • "_SHR_" • <キュー名> • "MG"
7	共用キュー受信用ライト管理テーブル	受信用共用キューの最大キュー数	4	1	次に示す文字列を順に連結した名称です。 • <RMSystemName プロパティ指定値> • "_SHR_" • <キュー名> • "WT"
8	共用キュー受信用リード管理テーブル	受信用共用キューの最大キュー数	4	1	次に示す文字列を順に連結した名称です。 • <RMSystemName プロパティ指定値> • "_SHR_" • <キュー名> • "RD"

注※

列数の詳細については，表 4-4 を参照してください。

表 4-3　管理情報テーブルの一覧（テーブルの可変長文字列数，データ長，データ型，キー名，キー長）

項番	種類	可変長文字列数※1 (VARCHAR≧256)	データ長 (バイト)	データ型※2	キー名※3	キー長 (バイト)
1	システム管理情報テーブル	5	1024	既定義型	－	－
2	キュー情報テーブル	3	1024		QUEUE_NAME	32
3	FIFO 情報テーブル	1 または 3	256 または 1024		FIFO_ID	256
4	あて先情報テーブル	1 または 3	512 または 1024		LOGICAL_ADDRESS	32
5	メッセージ情報テーブル	2 または 5	256 または 1024		FIFO_ID	256
					SEQUENCE_NO	20
					GROUP_NAME	256
					GROUP_MESSAGE_NUMBER	20
6	共用キュー受信用メッセージ情報テーブル	0	0		MSG#	4
7	共用キュー受信用ライト管理テーブル	0	0		－	－
8	共用キュー受信用リード管理テーブル	0	0		－	－

（凡例）

－：該当しません。

注※1

HiRDB では VARCHAR 型，Oracle では VARCHAR2 型となります。

注※2

データ型の詳細については，表 4-4 を参照してください。

注※3

キー名は一意にしてください。

データ型および列数の詳細を次に示します。

表 4-4　管理情報テーブルの一覧（データ型および列数の詳細）

項番	テーブル	HiRDB でのデータ型	Oracle でのデータ型	列数
1	システム管理情報テーブル	TIMESTAMP	DATE	2
		INTEGER	NUMBER(10)	2
		VARCHAR(128)	VARCHAR2(128)	1
		VARCHAR(1024)	VARCHAR2(1024)	5
2	キュー情報テーブル	TIMESTAMP	DATE	2
		INTEGER	NUMBER(10)	6
		VARCHAR(20)	VARCHAR2(20)	6
		VARCHAR(32)	VARCHAR2(32)	4
		VARCHAR(64)	VARCHAR2(64)	1
		VARCHAR(128)	VARCHAR2(128)	1
		VARCHAR(1024)	VARCHAR2(1024)	3
3	FIFO 情報テーブル	TIMESTAMP	DATE	2
		INTEGER	NUMBER(10)	4
		VARCHAR(20)	VARCHAR2(20)	8
		VARCHAR(32)	VARCHAR2(32)	1
		VARCHAR(256)	VARCHAR2(256)	1
		VARCHAR(1024)	VARCHAR2(1024)	3
		BINARY(10000000)	BLOB	2
4	あて先情報テーブル	VARCHAR(16)	VARCHAR2(16)	2
		VARCHAR(32)	VARCHAR2(32)	1
		VARCHAR(512)	VARCHAR2(512)	1
		VARCHAR(1024)	VARCHAR2(1024)	3
5	メッセージ情報テーブル	INTEGER	NUMBER(10)	4
		VARCHAR(20)	VARCHAR2(20)	8
		VARCHAR(128)	VARCHAR2(128)	1
		VARCHAR(256)	VARCHAR2(256)	2
		VARCHAR(1024)	VARCHAR2(1024)	5
		BINARY(2147483647)	LONG RAW	1
6	共用キュー受信用メッセージ情報テーブル	INTEGER	－	1
		BINARY(n)※	－	1

項番	テーブル	HiRDB でのデータ型	Oracle でのデータ型	列数
		BINARY(512)	−	1
7	共用キュー受信用ライト管理テーブル	INTEGER	−	2
		SMALLINT	−	1
		BINARY(256)	−	1
8	共用キュー受信用リード管理テーブル	INTEGER	−	1
		SMALLINT	−	1
		BINARY(256)	−	2

（凡例）

−：該当しません。

注※

n は，ユーザが受信用共用キュー作成時に指定したメッセージ長です。

(1)　管理情報テーブルの作成方法

管理情報テーブルを作成するには，SQL ファイルまたはコマンドを実行します。管理情報テーブルの作成方法を次の表に示します。

表 4-5　管理情報テーブルの作成方法

項番	種類	作成方法
1	システム管理情報テーブル	テーブル作成用 SQL ファイルを実行して作成します。管理情報テーブルの作成については，HiRDB の場合は「3.4.1(3)　Reliable Messaging の管理情報テーブルの作成」を，Oracle の場合は「3.4.2(3)　Reliable Messaging の管理情報テーブルの作成」を参照してください。
2	キュー情報テーブル	
3	FIFO 情報テーブル	
4	あて先情報テーブル	
5	メッセージ情報テーブル	hrmmkque コマンド（-t オプションに local または transmit を指定し，かつ-m オプションで persistent を指定する場合）を実行して作成します。hrmmkque コマンドについては，「8.3.17　hrmmkque（ローカルキューの作成）」および「8.3.20　hrmmkque（転送キューの作成）」を参照してください。
6	共用キュー受信用メッセージ情報テーブル	hrmmkque コマンド（-t オプションに shr_receive を指定する場合）を実行して作成します。hrmmkque コマンドについては，「8.3.18　hrmmkque（受信用共用キューの作成）」を参照してください。
7	共用キュー受信用ライト管理テーブル	
8	共用キュー受信用リード管理テーブル	

(2)　管理情報テーブルの削除方法

管理情報テーブルを削除するには，SQL ファイルまたはコマンドを実行します。管理情報テーブルの削除方法を次の表に示します。

表 4-6　管理情報テーブルの削除方法

項番	種類	削除方法
1	システム管理情報テーブル	テーブル削除用 SQL ファイルを実行して削除します。管理情報テーブルの削除については，「4.3.2(1)　テーブル削除用 SQL ファイルの使用」を参照してください。
2	キュー情報テーブル	
3	FIFO 情報テーブル	
4	あて先情報テーブル	
5	メッセージ情報テーブル	hrmdelque コマンドを実行して削除します。hrmdelque コマンドについては，「8.3.8　hrmdelque（キューの削除）」を参照してください。
6	共用キュー受信用メッセージ情報テーブル	
7	共用キュー受信用ライト管理テーブル	
8	共用キュー受信用リード管理テーブル	

4.3.7　アプリケーション認証による DB アクセス設定

アプリケーション認証を使うと，Reliable Messaging の初期設定を行った DBA 権限を持つユーザ（DBA ユーザ）以外のユーザ（一般ユーザ）も，Reliable Messaging にアクセスできるようになります。

アプリケーション認証を利用するには，DB に格納した Reliable Messaging の管理情報テーブルごとに，一般ユーザに適切なアクセス権限を与える必要があります。

ユーザに与えるアクセス権限の種類を次の表に示します。

表 4-7　アクセス権限の種類

項番	アクセス権限の種類	説明
1	SELECT 権限	テーブルの行を参照できる
2	INSERT 権限	テーブルに行を追加できる
3	DELETE 権限	テーブルから行を削除できる
4	UPDATE 権限	テーブルの行を更新できる

(1)　HiRDB の場合

DBA ユーザ権限を持つユーザ（DBA ユーザ）が，アプリケーション認証によって Reliable Messaging のテーブルにアクセスするユーザ（一般ユーザ）に権限を与えるには，次の SQL 文を実行します。

```
GRANT CONNECT TO <一般ユーザ> IDENTIFIED BY <一般ユーザのパスワード>;
GRANT SELECT,INSERT,DELETE,UPDATE
ON <DBAユーザ>.<管理情報テーブル名> TO <一般ユーザ>;
```

ユーザ権限の設定に関する詳細は，マニュアル「HiRDB システム運用ガイド」を参照してください。

(2) Oracle の場合

DBA ユーザ権限を持つユーザ（DBA ユーザ）が，アプリケーション認証によって Reliable Messaging のテーブルにアクセスするユーザ（一般ユーザ）に権限を与えるには，次の SQL 文を実行します。

```
CREATE USER <一般ユーザ> IDENTIFIED BY <一般ユーザのパスワード>;
GRANT SELECT,INSERT,DELETE,UPDATE
ON <DBAユーザ>.<管理情報テーブル名> TO <一般ユーザ>;
```

ユーザ権限の設定に関する詳細は，Oracle のマニュアルを参照してください。

4.4　PRF トレースファイルの運用（永続版リソースアダプタの場合）

Application Server の PRF トレースファイルを利用して，Reliable Messaging の性能を解析できます。

PRF トレースファイルの運用方法，取得レベル，および編集について説明します。

4.4.1　PRF トレースファイルの運用方法

PRF トレースを取得するには，Performance Tracer（以降，パフォーマンストレーサと呼びます）をインストールしておく必要があります。パフォーマンストレーサは，J2EE サーバの起動に合わせて起動するため，運用管理ドメインや J2EE サーバでの設定が必要となります。これらの設定は Management Server を利用して実施します。Management Server での設定手順を次に示します。なお，設定方法の詳細については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」を参照してください。

1. パフォーマンストレーサの追加
   運用管理ドメインで，パフォーマンストレーサを追加します。
2. パフォーマンストレーサの設定
   論理サーバの環境設定で，パフォーマンストレーサの設定をします。
3. J2EE サーバで利用するパフォーマンストレーサの設定
   論理サーバの環境設定で，J2EE サーバで利用するパフォーマンストレーサを選択します。

4.4.2　PRF トレースファイルの取得レベル

PRF トレースファイルの取得レベルには標準レベルと詳細レベルがあり，パフォーマンストレーサに PRF トレース取得レベルを設定できます。Reliable Messaging では，標準レベルの場合，主にアプリケーションインタフェースでのトレース情報を取得し，詳細レベルの場合，さらに内部インタフェースのトレース情報を取得します。

PRF トレース取得レベルは，パフォーマンストレーサの起動中に任意のタイミングで変更できます。PRF トレース取得レベルの変更方法については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

4.4.3　PRF トレースファイルの編集

PRF トレースはバイナリ形式で出力されるため，cprfed コマンドで編集する必要があります。cprfed コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。また，PRF トレースの出力については，「9.3.6　PRF トレース」を参照してください。

4.5 Reliable Messaging 運用時の注意事項（永続版リソースアダプタの場合）

永続版リソースアダプタを使用する場合の，Reliable Messaging の運用時の注意事項について説明します。

4.5.1 Windows 使用時の注意事項

Windows で，Reliable Messaging のシステムを構築，運用する場合の注意事項について説明します。

(1) UAC(ユーザーアカウント制御)機能が有効な環境で Reliable Messaging が提供するコマンドを使用する時の注意事項

Reliable Messaging が提供するコマンドは，管理者特権で実行する必要があります。Reliable Messaging が提供するコマンドは，「管理者：コマンドプロンプト」で実行してください。

「管理者：コマンドプロンプト」は，Windows で提供されている機能を使用して起動してください。

(2) JIS X0213:2004 に含まれる Unicode の補助文字を使用する場合の注意事項

JIS X0213:2004 の第三水準および第四水準の文字の一部には，Unicode の補助文字が含まれます。Unicode の補助文字とは，基本多言語面以外の文字（Unicode のコードポイントが U+10000～U+10FFFF の範囲の文字）のことです。UTF-16 エンコーディングでは，サロゲートペアとして表されます。

Unicode の補助文字を使用する場合の注意事項を次に示します。

(a) リクエストで使用する場合の注意事項

Reliable Messaging に対して，クライアントから Unicode の補助文字を含むリクエストを送信した場合，Unicode の補助文字は，ログや PRF トレースに正しく出力されません。ただし，その場合も，Unicode の補助文字以外の文字は，ログや PRF トレースに正しく出力されます。

また，リクエストに Unicode の補助文字が含まれる場合も，アプリケーションは正しく動作します。

リクエストでの Unicode の補助文字の使用を制限したい場合には，アプリケーションでの対応などを検討してください。

(b) 環境構築／運用時の注意事項

Reliable Messaging のシステムを構築，運用する場合，およびアプリケーションやリソースをデプロイする場合に使用する定義に，Unicode の補助文字は使用できません。

Unicode の補助文字を使用できない定義の例を示します。

- EAR，WAR，JAR，EJB-JAR，サーブレット，JSP，クラス，メソッド，引数，または変数の名称
- DD 内の各種定義
- システムのインストール先の指定値
- そのほか，各種定義ファイルの設定値

4.5.2 永続版リソースアダプタと非永続版リソースアダプタを切り替える場合の注意事項

- 永続版リソースアダプタから非永続版リソースアダプタに切り替える場合，または非永続版リソースアダプタから永続版リソースアダプタに切り替える場合，Application Server の提供するリソースアダプタバージョンアップコマンドによる更新はできません。永続版リソースアダプタと非永続版リソースアダプタを切り替えるには，切り替え前のリソースアダプタのアンデプロイ（cjundeployrar）とリソース削除（cjdeleteres）を実行し，新しく切り替え後のリソースアダプタのインポート（cjimportres）とデプロイ（cjdeployrar）を実行してください。
- 切り替え前のリソースアダプタで作成したキューや定義情報は，切り替え後のリソースアダプタで使用できません。切り替え後のリソースアダプタの仕様に従って再定義する必要があります。リソースアダプタの属性ファイル（Connector 属性ファイル）は，切り替え後のリソースアダプタを J2EE サーバにインポート（またはデプロイ）したあと，Application Server の属性ファイル取得／設定コマンド cjgetresprop／cjsetresprop（または cjgetrarprop／cjsetrarprop）で設定し直してください。

4.5.3 Reliable Messaging が動作するマシンの時刻設定の注意事項

NTP プログラムでマシン時刻の補正をする場合は slew モードを使用し，なおかつマシン時刻が戻らないようにしてください。

大幅な時刻補正が必要な場合は，製品のプロセスを停止してから時刻補正を行なうようにしてください。その場合，補正後の時刻がプロセス停止時刻以前の時刻にならないように注意してください。

製品のプロセスを停止せずにマシン時刻を戻すと，時刻補正前に登録したメッセージと補正後に登録したメッセージの受信順序が入れ替わる場合があります。

5 Reliable Messaging（非永続版リソースアダプタ）の運用

Reliable Messaging のアプリケーションを開始するには，あらかじめ Reliable Messaging を開始しておく必要があります。Reliable Messaging を停止するには，あらかじめアプリケーションを停止しておく必要があります。

この章では，非永続版リソースアダプタを使用する場合の，Reliable Messaging およびアプリケーションの開始と停止，ローカルキューの運用などについて説明します。

5.1 Reliable Messaging とアプリケーションの開始と停止（非永続版リソースアダプタの場合）

非永続版リソースアダプタを使用する場合の，Reliable Messaging とアプリケーションの開始手順および停止手順について説明します。

5.1.1 Reliable Messaging の開始（非永続版リソースアダプタの場合）

Reliable Messaging を開始するには，あらかじめプロパティのカスタマイズとデプロイを完了させている必要があります。詳細については，「3.5.3 Reliable Messaging のプロパティ定義（非永続版リソースアダプタの場合）」を参照してください。また，Reliable Messaging の開始時に使用するサーバ管理コマンドについては，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」を参照してください。

Reliable Messaging を開始する手順を次に示します。

1. J2EE サーバの開始

   Application Server を開始します。また，アプリケーションの要件に合わせて，必要な Application Server のサービスを開始してください。

2. Reliable Messaging の開始

   サーバ管理コマンドを使用して，リソースアダプタとして Reliable Messaging を開始します。

注意

- 非永続版リソースアダプタの場合，Reliable Messaging が正常に開始すると，実行状態になります。
- 非永続版リソースアダプタの場合，起動時にキュー作成ファイルの読み込みや解析で続行できないエラーが発生すると，閉塞状態で起動します。
- Connector 属性ファイルの<transaction-support>タグに LocalTransaction または NoTransaction を指定した場合，Reliable Messaging が開始状態になっても，実際には開始処理に失敗していることがあります。このときにアプリケーションを開始すると Reliable Messaging がコネクションを取得する契機でエラーとなります。このため，Reliable Messaging が正常に開始していることを Application Server のログメッセージで確認してください。Reliable Messaging が正常に開始している場合，ログメッセージとして KFRM01009-I が表示されます。
- Reliable Messaging の開始中は，Reliable Messaging が使用しているキュー定義ファイルおよびキュー作成ファイルを削除したり，読み込みができなくなったりするようなアクセス権限の変更は行わないでください。
- Reliable Messaging のリソースアダプタを開始するときに，OutOfMemoryError が発生し KDJE48571-E が出力された場合，Reliable Messaging の内部状態が不正になる可能性があります。この場合 Application Server を再起動してください。

5.1.2　アプリケーションの開始（非永続版リソースアダプタの場合）

アプリケーションを開始するには，あらかじめ Reliable Messaging が開始されている必要があります。各手順で使用するサーバ管理コマンドおよび各属性ファイルの詳細については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」およびマニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」を参照してください。

アプリケーションを開始する手順を次に示します。

1. アプリケーションのインポート

   サーバ管理コマンドを使用して，Reliable Messaging のアプリケーションをインポートします。

2. J2EE アプリケーションのプロパティ設定

   テキストエディタを使って，Session Bean 属性ファイル，Entity Bean 属性ファイル，WAR 属性ファイル，または MessageDrivenBean 属性ファイルを編集して，リソースアダプタのリファレンス定義，リソース環境のリファレンス定義およびメッセージ参照の設定項目を設定します。

3. アプリケーションの開始

   サーバ管理コマンドを使用して，アプリケーションを開始します。

5.1.3　Reliable Messaging の停止（非永続版リソースアダプタの場合）

サーバ管理コマンドを使用して，アプリケーションを停止させてから，Reliable Messaging を停止させます。サーバ管理コマンドについては，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」を参照してください。

Reliable Messaging を停止する手順を次に示します。

1. アプリケーションの停止

   サーバ管理コマンドを使用して，アプリケーションを停止します。一つの Reliable Messaging を利用する複数のアプリケーションがある場合はすべて停止します。

2. Reliable Messaging の停止

   サーバ管理コマンドを使用して，Reliable Messaging を停止します。

3. J2EE サーバの停止

   アプリケーションサーバシステムを使用しない場合は，J2EE サーバと Application Server のサービスを停止します。

注意

- Reliable Messaging のリソースアダプタを停止するときに，OutOfMemoryError が発生し KDJE42093-E が出力された場合，Reliable Messaging の内部状態が不正になる可能性があります。この場合 Application Server を再起動してください。

5.1.4　Reliable Messaging の状態遷移（非永続版リソースアダプタの場合）

アプリケーションサーバシステムにとって，Reliable Messaging には未デプロイ状態，デプロイ済み状態，およびリソースアダプタ開始状態の三つの状態があります。また，非永続版リソースアダプタを使用する場合，リソースアダプタ開始状態の Reliable Messaging の内部状態として，実行状態，閉塞状態および開始中状態があります。なお，非永続版リソースアダプタには，管理状態はありません。

Reliable Messaging の状態について次に説明します。

実行状態

アプリケーションにサービスを提供する状態です。

閉塞状態

Reliable Messaging 内部のメモリ状態不正などによって，処理の続行ができなくなった状態です。アプリケーションにサービスは提供されなくなります。

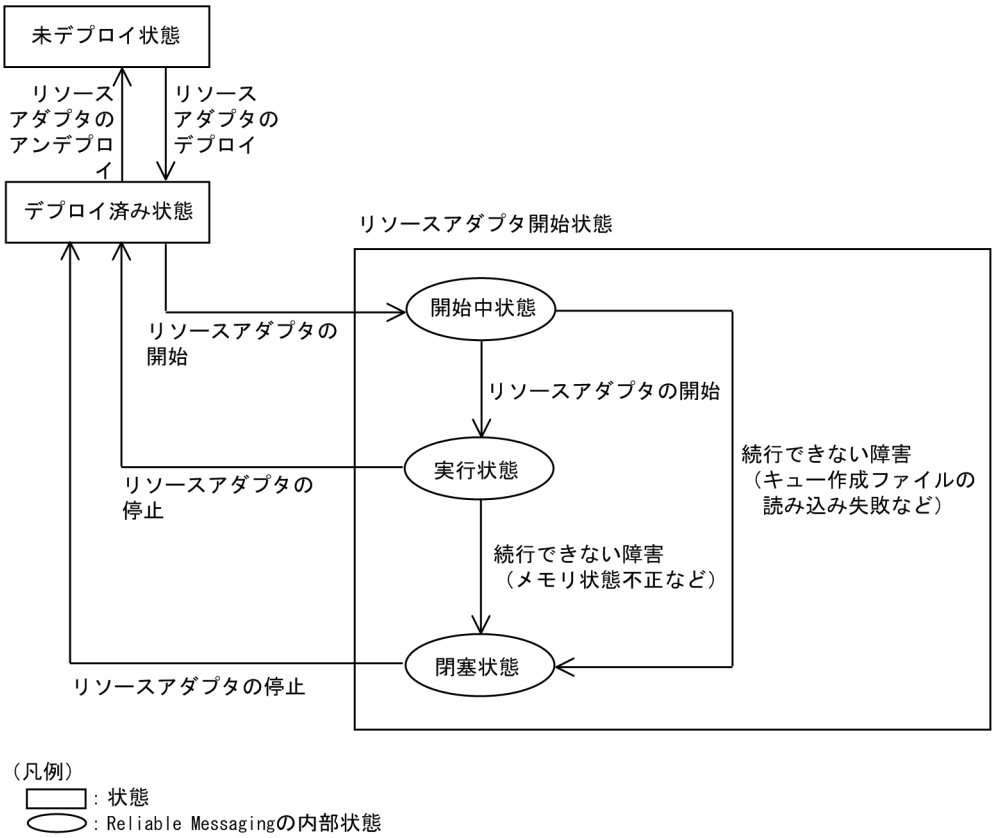
開始中状態

リソースアダプタを開始して，実行状態に移行するまでの一時的な状態です。

(1)　状態遷移

非永続版リソースアダプタを使用する場合の Reliable Messaging の状態遷移を次の図に示します。

図 5-1　非永続版リソースアダプタを使用する場合の Reliable Messaging の状態遷移



5.2　ローカルキューの運用（非永続版リソースアダプタの場合）

5.2.1　ローカルキューの操作方法

Reliable Messaging の非永続版リソースアダプタを開始すると，キュー作成ファイルを基にローカルキューが作成されます。Reliable Messaging が実行状態に移行すると，メッセージの送受信ができます。

非永続版リソースアダプタを使用する場合のローカルキューの操作について次に説明します。

- キューの追加，削除，および属性変更をする場合

  キュー作成ファイルを編集し，Reliable Messaging を再起動してください。非永続版リソースアダプタでは非永続キューだけを提供するため，Reliable Messaging を再起動するとすべてのメッセージがなくなります。キュー作成ファイルの詳細については，「3.5.1　キュー作成ファイルの作成」を参照してください。
- キュー情報の表示，メッセージの表示，メッセージの削除をする場合

  次に示すコマンドを実行してください。コマンドの詳細については，「8.　コマンドリファレンス」を参照してください。
  - hrmlsque（キュー情報の表示）
  - hrmlsmsg（メッセージの表示）
  - hrmdelmsg（メッセージの削除）

5.2.2　ローカルキューの状態遷移（非永続版リソースアダプタの場合）

キューには，通常状態，抑止状態（一部抑止状態および完全抑止状態）があります。キューの抑止状態は，特定のキューに対してメッセージの送受信を抑止する場合などに利用します。

キューの状態について次に説明します。

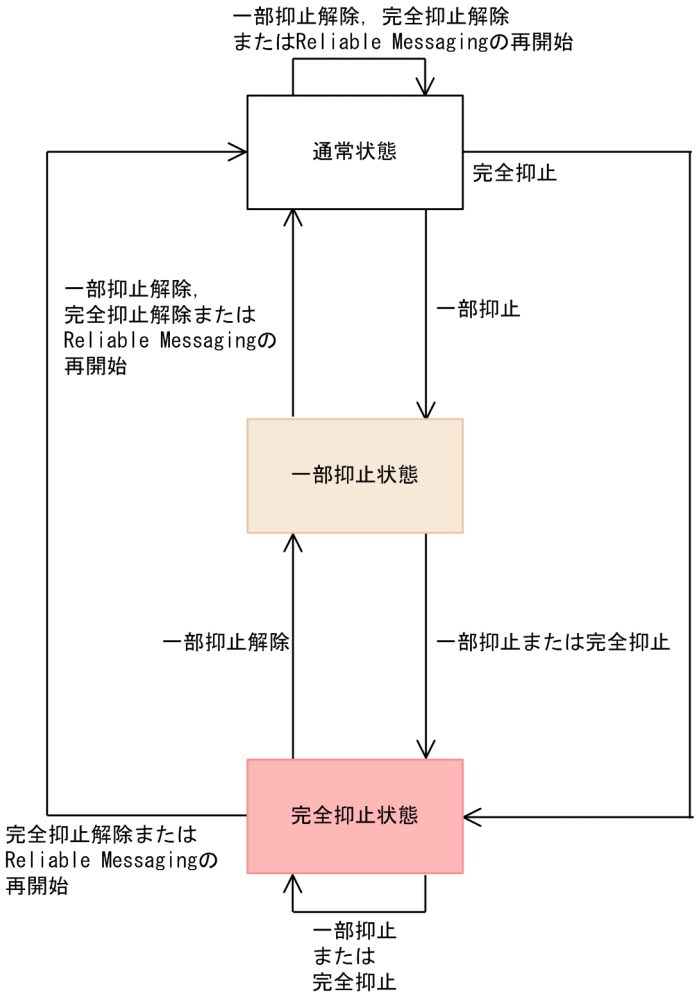
通常状態
: 正常に使用できる状態です。

一部抑止状態
: メッセージの送受信が一部抑止されている状態です。指定するキューの種類によって，抑止される形態が異なります。キューの種類と抑止される形態については，「8.3.27(3)　オプション」を参照してください。

完全抑止状態
: メッセージの送受信が完全に抑止されている状態です。指定するキューの種類に応じて，抑止できる形態をすべて抑止します。

キューの状態遷移を次の図に示します。

図 5-2　キューの状態遷移



5.3　PRF トレースファイルの運用（非永続版リソースアダプタの場合）

Application Server の PRF トレースファイルを利用して，Reliable Messaging の性能を解析できます。

PRF トレースを取得するには，Performance Tracer（以降，パフォーマンストレーサと呼びます）をインストールしておく必要があります。パフォーマンストレーサは，J2EE サーバの起動に合わせて起動するため，運用管理ドメインや J2EE サーバでの設定が必要となります。これらの設定は Management Server を利用して実施します。設定手順，および設定内容の詳細については，「4.4　PRF トレースファイルの運用（永続版リソースアダプタの場合）」を参照してください。

5.4 Reliable Messaging 運用時の注意事項（非永続版リソースアダプタの場合）

非永続版リソースアダプタを使用する場合の，Reliable Messaging の運用時の注意事項について説明します。

5.4.1 Windows 使用時の注意事項

Windows で，Reliable Messaging のシステムを構築，運用する場合の注意事項について説明します。

(1) UAC（ユーザーアカウント制御）機能が有効な環境で Reliable Messaging が提供するコマンドを使用する時の注意事項

Reliable Messaging が提供するコマンドは，管理者特権で実行する必要があります。Reliable Messaging が提供するコマンドは，「管理者：コマンドプロンプト」で実行してください。

「管理者：コマンドプロンプト」は，Windows で提供されている機能を使用して起動してください。起動方法の例を次に示します。

1. [スタート] ボタンをクリックします。
2. [アクセサリ] を選択します。
3. [コマンド プロンプト] を右クリックして，[管理者として実行] をクリックします。
   管理者のパスワードまたは確認を求められた場合は，画面の指示に従って，パスワードを入力するか，または確認情報を提供してください。

(2) JIS X0213:2004 に含まれる Unicode の補助文字を使用する場合の注意事項

JIS X0213:2004 の第三水準および第四水準の文字の一部には，Unicode の補助文字が含まれます。Unicode の補助文字とは，基本多言語面以外の文字（Unicode のコードポイントが U+10000～U+10FFFF の範囲の文字）のことです。UTF-16 エンコーディングでは，サロゲートペアとして表されます。

Unicode の補助文字を使用する場合の注意事項を次に示します。

(a) リクエストで使用する場合の注意事項

Reliable Messaging に対して，クライアントから Unicode の補助文字を含むリクエストを送信した場合，Unicode の補助文字は，ログや PRF トレースに正しく出力されません。ただし，その場合も，Unicode の補助文字以外の文字は，ログや PRF トレースに正しく出力されます。

また，リクエストに Unicode の補助文字が含まれる場合も，アプリケーションは正しく動作します。

リクエストでの Unicode の補助文字の使用を制限したい場合には，アプリケーションでの対応などを検討してください。

(b) 環境構築／運用時の注意事項

Reliable Messaging のシステムを構築，運用する場合，およびアプリケーションやリソースをデプロイする場合に使用する定義に，Unicode の補助文字は使用できません。

Unicode の補助文字を使用できない定義の例を示します。

- EAR，WAR，JAR，EJB-JAR，サーブレット，JSP，クラス，メソッド，引数，または変数の名称
- DD 内の各種定義
- システムのインストール先の指定値
- そのほか，各種定義ファイルの設定値

5.4.2 永続版リソースアダプタと非永続版リソースアダプタを切り替える場合の注意事項

- 永続版リソースアダプタから非永続版リソースアダプタに切り替える場合，または非永続版リソースアダプタから永続版リソースアダプタに切り替える場合，Application Server の提供するリソースアダプタバージョンアップコマンドによる更新はできません。永続版リソースアダプタと非永続版リソースアダプタを切り替えるには，切り替え前のリソースアダプタのアンデプロイ（cjundeployrar）とリソース削除（cjdeleteres）を実行し，新しく切り替え後のリソースアダプタのインポート（cjimportres）とデプロイ（cjdeployrar）を実行してください。
- 切り替え前のリソースアダプタで作成したキューや定義情報は，切り替え後のリソースアダプタで使用できません。切り替え後のリソースアダプタの仕様に従って再定義する必要があります。リソースアダプタの属性ファイル（Connector 属性ファイル）は，切り替え後のリソースアダプタを J2EE サーバにインポート（またはデプロイ）したあと，Application Server の属性ファイル取得／設定コマンド cjgetresprop／cjsetresprop（または cjgetrarprop／cjsetrarprop）で設定し直してください。

5.4.3 Reliable Messaging が動作するマシンの時刻設定の注意事項

NTP プログラムでマシン時刻の補正をする場合は slew モードを使用し，なおかつマシン時刻が戻らないようにしてください。

大幅な時刻補正が必要な場合は，製品のプロセスを停止してから時刻補正を行なうようにしてください。その場合，補正後の時刻がプロセス停止時刻以前の時刻にならないように注意してください。

製品のプロセスを停止せずにマシン時刻を戻すと，時刻補正前に登録したメッセージと補正後に登録したメッセージの受信順序が入れ替わる場合があります。

6 コンフィグレーションプロパティ

Reliable Messaging および DB Connector for Reliable Messaging を開始するには，ユーザの環境に合わせてコンフィグレーションプロパティを設定する必要があります。

この章では，Reliable Messaging および DB Connector for Reliable Messaging のコンフィグレーションプロパティの詳細について説明します。

6.1 Reliable Messaging のコンフィグレーションプロパティの一覧

コンフィグレーションプロパティは Reliable Messaging の動作内容を指定するプロパティです。プロパティを定義したら，サーバ管理コマンドを使用して，Reliable Messaging をインポートします。そのあと，プロパティを設定します。プロパティ定義の詳細については，永続版リソースアダプタの場合は「3.4.5 Reliable Messaging のプロパティ定義（永続版リソースアダプタの場合）」を，非永続版リソースアダプタの場合は「3.5.3 Reliable Messaging のプロパティ定義（非永続版リソースアダプタの場合）」を参照してください。

Reliable Messaging が提供するコンフィグレーションプロパティの一覧を次の表に示します。

表 6-1 Reliable Messaging のコンフィグレーションプロパティの一覧

項番	プロパティ名	設定内容	永続版リソースアダプタでの指定	非永続版リソースアダプタでの指定
1	RMSystemName	システム名	◎	◎
2	RMLinkedDBConnectorName	連携する DB Connector の表示名	◎	×
3	QueueMakeFileName	キュー作成ファイルの場所	×	◎
4	QueueConfigFileName	キュー定義ファイルの場所	○※1	○※1
5	RMDeadMessageQueueName	デッドメッセージキュー名	○	×
6	RMWaitRestoration	キュー定義ファイル未使用時の Reliable Messaging 開始時の復元完了待ち合わせの有無	○※2	×
7	RMStartTimeout	Reliable Messaging 開始処理のタイムアウト時間（無限待ちの回避）	○※3	×
8	RMAssociateJDBCFlag	DB Connector とのコネクション共有の利用有無	○	×
9	RMSweepTimerInterval	メッセージ削除処理の実行間隔	○	○
10	RMDeleteMessageImmediately	メッセージ即時削除の利用有無	○	×
11	RMPassByReference	メッセージ送受信時の参照渡し方式の利用の有無	○	○
12	RMMaxDeliveryNum	配送回数の最大値	○	○
13	RMMethodTraceLevel	メソッドトレースの出力レベル	○	○
14	RMLineTraceLevel	回線トレースの出力レベル	○	×
15	RMLogTraceFileNum	トレースファイルの最大面数	○	○
16	RMLogTraceFileSize	トレースファイルのファイルサイズ	○	○
17	RMSHConnectFlag	共用キューを使用して複数システム間でのアプリケーション連携をする場合の受信用共用キューの有無	○	×

項番	プロパティ名	設定内容	永続版リソースアダプタでの指定	非永続版リソースアダプタでの指定
18	RMSHPort	共用キューを使用して複数システム間でのアプリケーション連携をする場合のイベント受信用ポート番号	○※4	×
19	RMSHRecoveryTimerInterval	共用キューを使用して複数システム間でのアプリケーション連携をする場合のリカバリスレッド監視間隔	○※4	×
20	RMTRConnectFlag	キュー間転送の使用有無	○	×
21	RMTRSendThreadNum	送信スレッドの起動数	○※5	×
22	RMTRResendInterval1Num	再送間隔 1 での再送回数	○※5	×
23	RMTRResendInterval1	再送間隔 1	○※5	×
24	RMTRResendInterval2	再送間隔 2	○※5	×
25	RMTRResendTimerInterval	再送タイマ監視間隔	○※5	×
26	RMTRPendingNotifyInterval	滞留メッセージの監視時間	○※5	×
27	RMTRTransferControlDir	クライアント定義ファイルが格納されているディレクトリのパス	○※5	×
28	RMAutoDeleteMessage	デッドメッセージキュー未使用時の無効メッセージ自動削除の有無	○	×

（凡例）

◎：必ず指定します。

○：必要に応じて指定します。

×：指定不要です。

注※1

キュー定義ファイルを使用する場合，QueueConfigFileName プロパティは必須です。キュー定義ファイルを使用しない場合は，QueueConfigFileName プロパティを省略するか，プロパティの値を空にしてください。

注※2

キュー定義ファイルを使用する場合，RMWaitRestoration プロパティは無視されます。

注※3

キュー定義ファイルを使用する場合，または RMWaitRestoration プロパティに false を指定する場合，RMStartTimeout プロパティは無視されます。

注※4

RMSHConnectFlag プロパティに false を指定する場合，これらのプロパティは無視されます。

注※5

RMTRConnectFlag プロパティに false を指定する場合，これらのプロパティは無視されます。

6.2　Reliable Messaging のコンフィグレーションプロパティの詳細説明

各コンフィグレーションプロパティの詳細について説明します。

6.2.1　RMSystemName ＝ システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字：java.lang.String ＞

Reliable Messaging が連携するシステム全体で一意なシステム名を指定します。このプロパティは必須です。

このプロパティの指定がない場合や空白を指定している場合，Reliable Messaging の開始時にエラーが発生します。

6.2.2　RMLinkedDBConnectorName ＝ 連携する DB Connector の表示名

～＜文字列：java.lang.String ＞((1～128))

連携する DB Connector の表示名を指定します。このプロパティは必須です。

プロパティが省略された場合や，プロパティに指定した表示名の DB Connector が起動していない場合，起動に失敗します。

6.2.3　QueueMakeFileName ＝ キュー作成ファイルの場所

～＜文字列：java.lang.String ＞

キュー作成ファイル名を絶対パスで指定します。非永続版リソースアダプタではこのプロパティは必須です。永続版リソースアダプタではこのプロパティは指定不要です。

非永続版リソースアダプタでこのプロパティの指定がない場合，または指定したキュー作成ファイルの読み取りに失敗した場合，Reliable Messaging の開始時にエラーが発生します。

キュー作成ファイルの詳細については，「3.5.1　キュー作成ファイルの作成」を参照してください。

6.2.4　QueueConfigFileName ＝ キュー定義ファイルの場所

～＜文字列：java.lang.String ＞

キュー定義ファイル名を絶対パスで指定します。

このプロパティは，キュー定義ファイルを使用する場合は必須です。キュー定義ファイルを使用しない場合はこのプロパティを省略するか，プロパティの値を空にしてください。

キュー定義ファイルを使用しない場合，キュー作成時に指定した表示名に従って生成されたキューの識別名が JNDI ネーミングサービスに登録されます。

指定したキュー定義ファイルの読み取りに失敗した場合，Reliable Messaging の開始時にエラーが発生します。

キュー定義ファイルの詳細については，永続版リソースアダプタの場合，「3.4.4　キュー定義ファイルの作成（永続版リソースアダプタの場合）」を参照してください。非永続版リソースアダプタの場合，「3.5.2 キュー定義ファイルの作成（非永続版リソースアダプタの場合）」を参照してください。

6.2.5　RMDeadMessageQueueName ＝ デッドメッセージキュー名

～＜ 1～20 文字の識別子：java.lang.String ＞

デッドメッセージキューを使用する場合に，デッドメッセージキューとして使用するローカルキュー名を指定します。ローカルキュー以外のキュー名を指定する場合，Reliable Messaging の開始時にエラーが発生します。このプロパティの指定がない場合，または存在しないキュー名を指定した場合，デッドメッセージキューを使用できません。デッドメッセージキューとして使用するキューは，ユーザが hrmmkque コマンドでローカルキューとして作成してください。

なお，このプロパティを変更して別のローカルキューをデッドメッセージキューにする場合，変更前のデッドメッセージキューに格納されているメッセージをすべて削除してください。

キュー間転送では，転送できなかったメッセージは転送キューに蓄積されます。転送キュー内のメッセージが格納できる最大メッセージ数に達すると，転送キュー内のメッセージが減らないかぎり，新たにメッセージを転送キューに格納し送信できなくなります。

デッドメッセージキューを使用すると，転送できなかったメッセージを削除処理のタイミングで転送キューからデッドメッセージキューに移動できます。これによって，転送キューがいっぱいになるのを防止できます。

6.2.6　RMWaitRestoration ＝ Reliable Messaging 開始時の復元完了待ち合わせの有無

～＜ java.lang.Boolean ＞((true ｜ false))《true》

キュー定義ファイルを使用しない場合に，Reliable Messaging を開始するときに非同期に実行する復元処理（DB に格納されているキュー情報を読み込みキューを復元すること）が完了するまで，Reliable Messaging の開始を待ち合わせるかどうかを指定します。

キュー定義ファイルを使用する場合，このプロパティは無視されます。

非永続版リソースアダプタの場合，Reliable Messaging の開始処理と復元処理を同期で実行しているためこのプロパティは使用しません。

このプロパティに true を指定した場合，復元処理の完了を待ち合わせて Reliable Messaging を開始します。false を指定した場合，復元処理の完了を待ち合わせないで Reliable Messaging を開始します。

このプロパティに true を指定した場合，Reliable Messaging の開始を待ち合わせることにより，次のような影響があります。

- Reliable Messaging とアプリケーションが開始している状態で Application Server を再起動したときに，DB からの復元処理に時間が掛かるとアプリケーションの開始処理の完了が遅くなります。
- Application Server が開始している状態で，Reliable Messaging を開始するときに，Reliable Messaging の復元処理に時間が掛かると，リソースアダプタの開始コマンドがタイムアウトすることがあります。
- DB ほか関連リソースにアクセスできないなど，Reliable Messaging が正常に開始完了（非閉塞状態で開始し，KFRM01009-I メッセージが出力される）できない環境で Reliable Messaging を開始すると，次のような現象が発生します。

表 6-2　Reliable Messaging を正常に開始完了できない環境で発生する現象

トランザクションサポートレベル	Reliable Messaging の開始時の環境	
	Application Server 再起動の延長の場合	Application Server 起動状態の場合
XATransaction	関連リソースに正常アクセスできるまで，または RMStartTimeout で指定した値に到達し開始処理のリトライがタイムアウトするまで，Application Server の開始処理が完了しません。	Reliable Messaging の開始処理が失敗します。
LocalTransaction，NoTransaction	関連リソースに正常アクセスできるまで，または RMStartTimeout で指定した値に到達し開始処理のリトライがタイムアウトするまで，Application Server の開始処理が完了しません。	関連リソースに正常アクセスできるまで，または RMStartTimeout で指定した値に到達し開始処理のリトライがタイムアウトするまで，Reliable Messaging の開始処理が完了しません。

Application Server または Reliable Messaging の開始を待っている状態で，開始処理を中断する場合は，cjstopsv コマンドに-f オプションを指定して J2EE サーバを強制停止してください。cjstopsv コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

このプロパティに false を指定した場合，タイミングによって次の現象が発生します。

- Reliable Messaging を利用するアプリケーションが JNDI ネーミングサービスからキューをルックアップするときに，ルックアップに失敗します（ただし，Message-driven Bean（MDB）を除きます）。なお，次の二つの条件を満たしている場合，アプリケーション開始時にキューのルックアップを実行します。

- Stateless Session Bean でインスタンスプールの最小値が 1 以上である
- ejbCreate()メソッド内でキューのルックアップ処理を実装，または EJB のアノテーション機能によってキューを定義している

これらの条件を満たしている場合にキューのルックアップに失敗したときは，アプリケーションの開始が失敗しますので注意してください。

ルックアップについては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の JNDI 名前空間とルックアップについての記述を参照してください。

Stateless Session Bean については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(EJB コンテナ)」を参照してください。

アノテーション機能については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」を参照してください。

- Reliable Messaging を利用するアプリケーションを開始するときに，表示名に対応したキュー名を正しく取得できずにエラーとなり，アプリケーションの開始に失敗します。

このプロパティは省略できます。

6.2.7 RMStartTimeout = Reliable Messaging 開始処理のタイムアウト時間

～＜数字：java.lang.Integer ＞((0～600))《60》（単位：秒）

このプロパティを設定することで，Reliable Messaging の開始処理を待ち合わせる場合，タイムアウトが発生するまで待つことを避けることができます。

非永続版リソースアダプタの場合，開始処理でリトライするエラー要因がないため，このプロパティは使用しません。

Reliable Messaging の開始時に非同期に実行する復元処理（DB に格納されているキュー情報を読み込みキューを復元すること）が完了するまで Reliable Messaging の開始を待ち合わせる場合，開始処理失敗時のリトライをタイムアウトする時間を指定します。このプロパティは，RMWaitRestoration プロパティが true で，かつキュー定義ファイルを使用しないときにだけ有効となります。それ以外のとき，このプロパティの設定値は無視されタイムアウトしません。

このプロパティの値は，Reliable Messaging や Reliable Messaging を利用するアプリケーションを開始する場合，実行するコマンドなどがタイムアウトする時間よりも小さい値を指定することをお勧めします。値に 0 を指定したとき，Reliable Messaging の開始処理はタイムアウトしないため，開始処理が正常に完了する（非閉塞状態で開始し，KFRM01009-I メッセージが出力される）まで待ち合わせます。

なお，実際にタイムアウトする時間は，指定した値よりも数秒遅れることがあります。

タイムアウト後，Reliable Messaging は開始処理のリトライを中止し，システム閉塞状態で開始するため，Reliable Messaging 開始の待ち合わせは解除されます。この場合，Application Server または Reliable Messaging を停止し，正しい設定・環境で再度 Reliable Messaging を開始してください。

Reliable Messaging が開始処理に失敗しリトライするエラー要因を次の表に示します。

表 6-3　Reliable Messaging が開始処理に失敗しリトライするエラー要因

トランザクションサポートレベル	Reliable Messaging の開始時の環境	
	Application Server 再起動の延長の場合	Application Server 起動状態の場合
XATransaction	• DB 停止しているなど一時的な障害 • Application Server 再起動前と再起動後で接続先 DB の IP アドレスが変更されるような場合	リトライするエラー要因はありません
LocalTransaction, NoTransaction	• DB 停止しているなど一時的な障害 • DB 接続に関する設定の誤り（ユーザ名・パスワードの不正など）	• DB 停止しているなど一時的な障害 • DB 接続に関する設定の誤り（ユーザ名・パスワードの不正など）

このプロパティは省略できます。

6.2.8　RMAssociateJDBCFlag=DB Connector とのコネクション共有機能の使用有無

～＜java.lang.Boolean＞((true ｜ false))《true》

DB Connector とのコネクション共有機能を使用するかどうかを指定します。

DB Connector とのコネクション共有機能を使用する場合は true，使用しない場合は false を指定します。このプロパティが true の場合，非永続キューだけを利用していても，DB に対してトランザクション命令を発行します。非永続キューだけを利用する場合は false を指定します。ただし，このプロパティを false に設定すると，連携する DB Connector のコネクションは取得できないため，注意してください。

また，このプロパティを false に設定して HiRDB を XATransaction モードで利用する場合は，DB Connector のステートメントプールを利用しないでください。

6.2.9　RMSweepTimerInterval ＝ メッセージ削除処理の実行間隔

～＜数字：java.lang.Integer＞((60～86400))《600》（単位：秒）

メッセージの削除処理を実行する間隔を指定します。

キューに登録されたメッセージをアプリケーションが読み出したときや，キュー上のメッセージが有効期間に達したとき，その時点ではメッセージは削除されません。RMSweepTimerInterval プロパティ指定値の間隔で実行されるメッセージ削除処理によって削除されます。

- 永続版リソースアダプタの場合

  指定値が大き過ぎるとメモリ使用量や DB のエリア使用量が増加し，小さ過ぎると DB アクセスが増加するので注意してください。また，転送キューを利用する場合，転送に成功したメッセージはメッセージ削除処理によって削除されるまでキューに格納されたままなので，指定値を大きくし過ぎないように注意してください。

  RMSweepTimerInterval の値を大きくすると，削除対象となるメッセージ数が増える可能性があります。削除対象メッセージ数が増えると，遅延削除の DB 更新処理にかかる時間が増加するため，HiRDB を使用する場合，HiRDB のクライアント環境変数 PDCWAITTIME の値をチューニングしなおす必要があります。

  PDCWAITTIME のチューニング方法については，「3.4.1(2)(b)　HiRDB の環境変数グループの登録」を参照してください。

- 非永続版リソースアダプタの場合

  指定値が大き過ぎるとメモリ使用量が増加し，小さ過ぎると DB アクセス頻度に関係なく CPU に負荷の掛かる頻度が高くなります。

このプロパティは省略できます。

6.2.10　RMDeleteMessageImmediately＝メッセージ即時削除の利用有無

～＜java.lang.Boolean＞((true ｜ false))《false》

メッセージの取り出しで，キュー内のメッセージを配信済みの状態にしないで，すぐに削除するかどうかを指定します。

非永続版リソースアダプタの場合，即時削除だけを使用できます。遅延削除は使用できないためこのプロパティは使用しません。

メッセージをすぐに削除する場合は true を指定します。RMSweepTimerInterval プロパティで指定した間隔で定期的に削除する場合は false を指定します。

このプロパティが true の場合，取り出し時にメッセージを削除するため，永続キューのスループットが低下するおそれがありますが，一時的なメモリ使用量を削減できます。なお，有効期限切れのメッセージは RMSweepTimerInterval プロパティで指定した間隔で定期的に削除されます。

次のメッセージがこのプロパティの対象になります。

- アプリケーションからローカルキューに登録されたメッセージ
- デッドメッセージキューに登録されたメッセージ

即時削除・遅延削除の詳細については，「2.3.4　メッセージの削除」を参照してください。

6.2.11　RMPassByReference ＝ メッセージ送受信時の参照渡し方式の利用の有無

～＜java.lang.Boolean＞((true ｜ false))《false》

メッセージの送受信時に，参照渡し方式を利用するかどうかを設定します。

メッセージの参照渡し方式を利用する場合は true を指定します。利用しない場合は false を指定します。このプロパティが true の場合，アプリケーションが Reliable Messaging と同じメッセージのインスタンスを参照するため，高速にメッセージを送受信できます。

ただし，このプロパティを true に指定するとき，アプリケーション側でメッセージ送受信後にメッセージオブジェクトに対する変更や再利用はできません。これらの動作を行った場合，メッセージオブジェクトに対するメソッドの処理結果は保証できないため，注意してください。詳細は「2.6.7(10)　参照渡し方式利用時の JMS メッセージの再利用」を参照してください。

6.2.12　RMMaxDeliveryNum ＝ 配送回数の最大値

～＜数字：java.lang.Integer＞((0～512))《10》

Reliable Messaging から受信側アプリケーションへのメッセージの配送回数の最大値を指定します。0 を指定したときメッセージは無制限に再配送されます。

メッセージ配送中に QueueSession.recover()メソッドが発行されるかトランザクションがロールバックされると，このプロパティの回数内でメッセージは再配送の対象となります。

- 永続版リソースアダプタの場合

  配送回数の最大値に達したとき，デッドメッセージキューがあればメッセージはデッドメッセージキューに移動されます。ただし，次に示すとき，メッセージは元のキューに戻されます。

  - デッドメッセージキューがないとき
  - デッドメッセージキューの最大メッセージ数を超えるとき

  なお，メッセージの配送回数は DB によって永続化されないため，Reliable Messaging を再度開始したときには 0 に戻ります。

- 非永続版リソースアダプタの場合

  配送回数の最大値を超えて配送されたメッセージは，デッドメッセージキューに移動されないで削除されます。

6.2.13　RMMethodTraceLevel = メソッドトレースの出力レベル

～＜数字：java.lang.Integer＞((1～5))《1》

Reliable Messaging が出力するメソッドトレースの出力レベルを指定します。範囲外の値を指定した場合はデフォルト値が設定されます。

各レベルの出力情報については，「9.3.3　メソッドトレース」を参照してください。

6.2.14　RMLineTraceLevel = 回線トレースの出力レベル

～＜数字：java.lang.Integer＞((1～5))《3》

Reliable Messaging が出力する回線トレースのトレースの出力レベルを指定します。範囲外の値を指定した場合はデフォルト値が設定されます。

各レベルの出力情報については，「9.3.5　回線トレース」を参照してください。

6.2.15　RMLogTraceFileNum = トレースファイルの最大面数

～＜数字：java.lang.Integer＞((2～16))《2》

Reliable Messaging が出力するトレースファイル共通の出力ファイルの最大面数を指定します。範囲外の値を指定した場合はデフォルト値が設定されます。

6.2.16　RMLogTraceFileSize = トレースファイルのファイルサイズ

～＜数字：java.lang.Integer＞((4096～2147483647))《2097152》（単位：バイト）

Reliable Messaging が出力するトレースファイル共通のファイル 1 面当たりの最大のファイルサイズを指定します。範囲外の値を指定した場合はデフォルト値が設定されます。

6.2.17　RMSHConnectFlag = 共用キューを使用して複数システム間でのアプリケーション連携をする場合の受信用共用キューの有無

～＜ java.lang.Boolean ＞((true ｜ false))《false》

共用キューを使用して複数システム間でのアプリケーション連携をする場合の受信用共用キューの有無を指定します。

共用キューを使用して複数システム間でのアプリケーション連携をする場合に自システムで受信用共用キューを使用するときは true，使用しないときは false を指定します。

Reliable Messaging が接続する DB の種別が Oracle の場合に true を指定すると，開始処理に失敗します。

このプロパティに false を設定した場合，次に示す機能は使用できません。

- イベント受信機能
- リカバリスレッド機能
- 受信用共用キューに対するメッセージの登録および取り出し
- 受信用共用キューに対するメッセージの参照および削除
- 受信用共用キューからデッドメッセージキューに移動したメッセージの再登録

6.2.18 RMSHPort = 共用キューを使用して複数システム間でのアプリケーション連携をする場合のイベント受信用ポート番号

～＜数字：java.lang.Integer＞((1024～65535))《20351》

共用キューを使用して複数システム間でのアプリケーション連携をする場合のイベント受信用のポート番号を指定します。

6.2.19 RMSHRecoveryTimerInterval = 共用キューを使用して複数システム間でのアプリケーション連携をする場合のリカバリスレッド監視間隔

～＜数字：java.lang.Integer＞((5～300))《60》（単位：秒）

共用キューを使用して複数システム間でのアプリケーション連携をする場合に動作する，リカバリスレッドの監視間隔を指定します。リカバリスレッドについては，「2.6.4　共用キューでのメッセージ受信時の処理の流れ」を参照してください。

このプロパティの指定値が大き過ぎると未処理のメッセージが滞留することがあり，小さ過ぎると DB アクセスが増加するので注意してください。

6.2.20 RMTRConnectFlag=キュー間転送の使用有無

～＜ java.lang.Boolean ＞((true ｜ false))《false》

キュー間転送を使用するかどうかを指定します。キュー間転送を使用する場合は true を，使用しない場合は false を指定します。キュー間転送を使用したメッセージの送受信を行う場合，true を指定してください。

このプロパティを true に設定して HiRDB を XATransaction モードで利用する場合は，DB Connector のステートメントプールを利用しないでください。

このプロパティに false を設定した場合，次に示す内容は実行できません。

- キュー間転送およびキュー間転送によるメッセージの受信
- 転送キューへのメッセージの登録
- このプロパティが true のときに転送キューから受信したメッセージの削除
- 転送キューからデッドメッセージキューに移動したメッセージの再登録
- 次に示すコマンドの実行（引数に転送キューを指定）
  - hrmmkque（キューの作成）
  - hrmdelque（キューの削除）
  - hrmlsque（キュー情報の表示）
  - hrmchgque（キューの属性変更）
  - hrmlsmsg（メッセージの表示）
  - hrmdelmsg（メッセージの削除）
  - hrmstopque（キューの抑止）
  - hrmstoptrs（送受信抑止）
- 次に示すコマンドの実行
  - hrmmkaddr（あて先登録）
  - hrmdeladdr（あて先削除）
  - hrmlsaddr（あて先表示）
  - hrmskipmsg（受信待ちメッセージのスキップ）
  - hrmlstrs（通信状態表示）

6.2.21　RMTRSendThreadNum=送信スレッドの起動数

～＜数字：java.lang.Integer＞((1～128))《1》（単位：個）

キュー間転送で，メッセージの送信を行うために利用するスレッドの最大起動数を指定します。このプロパティを省略した場合，何も指定しなかった場合，および範囲外の値を指定した場合は，1 が指定されたものとして動作します。

一つのメッセージを送信する処理を一つのスレッドで行うため，このプロパティの値を増やすことで同時に送信できるメッセージ数を増やすことができます。しかし，マシンへの負荷が増加しスループットが低下する場合があります。

6.2.22　RMTRResendInterval1Num=再送間隔 1 での再送回数

～＜数字：java.lang.Integer ＞((1～100))《6》（単位：送信回数）

キュー間転送で，再送間隔 1 で再送する回数を指定します。メッセージの再送間隔をこのプロパティで切り替えることができます。切り替える前の再送間隔を再送間隔 1 に指定し，切り替えたあとの再送間隔を再送間隔 2 に指定します。

このプロパティを省略した場合や，何も指定しなかった場合，範囲外の値を指定した場合は，デフォルト値を指定したものとして動作します。

6.2.23　RMTRResendInterval1=再送間隔 1

～＜数字：java.lang.Integer ＞((1～86400))《10》（単位：秒）

キュー間転送で，転送先からの応答が返ってこないメッセージを再び送信するまでの時間を指定します。このプロパティは RMTRResendInterval1Num によって切り替える前の再送間隔です。このプロパティを省略した場合や範囲外の値を指定した場合は，デフォルト値が指定されたものとして動作します。

6.2.24　RMTRResendInterval2 = 再送間隔 2

～＜数字：java.lang.Integer ＞((1～86400))《600》（単位：秒）

キュー間転送で，転送先からの応答が返ってこないメッセージを再び送信するまでの時間を指定します。このプロパティは RMTRResendInterval1Num によって切り替えたあとの再送間隔です。このプロパティを省略した場合や範囲外の値を指定した場合は，デフォルト値が指定されたものとして動作します。

6.2.25　RMTRResendTimerInterval = 再送タイマ監視間隔

～＜数字：java.lang.Integer ＞((1～3600))《10》（単位：秒）

再送タイマ監視間隔を指定します。再送タイマ監視間隔とは，あるメッセージの前回の送信時から経過した時間が，再送間隔を超えているかどうか監視を行う間隔です。

このプロパティを省略した場合や範囲外の値を指定した場合は，デフォルト値が指定されたものとして動作します。

6.2.26　RMTRPendingNotifyInterval = 滞留メッセージの監視時間

～＜数字：java.lang.Integer＞((1～86400))《600》（単位：秒）

滞留メッセージを監視する時間を指定します。滞留メッセージを監視するタイミングは，メッセージ削除処理と同じです。メッセージの監視中に，このプロパティに指定した時間より長く滞留しているメッセージが存在する場合は，メッセージの滞留を通知する KFRM13011-W（メッセージログの出力レベルに関係なく出力します）が出力されます。

このプロパティを省略した場合や範囲外の値を指定した場合は，デフォルト値が指定されたものとして動作します。

6.2.27　RMTRTransferControlDir = クライアント定義ファイルが格納されているディレクトリのパス

～＜文字列：java.lang.String＞

キュー間転送の動作制御内容を定義する，クライアント定義ファイルが格納されているディレクトリのパスを指定します。

同じサーバ上で Reliable Messaging を複数デプロイする場合は，このプロパティに，コピーしたクライアント定義ファイルが格納されているディレクトリのパスを指定してください。クライアント定義ファイルの詳細については，「3.4.13(1)　SOAP 通信基盤の設定」を参照してください。

このプロパティを省略した場合や空文字を指定した場合は，デフォルトのクライアント定義ファイルのあるディレクトリのパス（%HRMDIR%¥conf）が指定されたものとして動作します。

6.2.28　RMAutoDeleteMessage = デッドメッセージキュー未使用時の無効メッセージ自動削除の有無

～＜ java.lang.Boolean＞((true ｜ false))《false》

キュー間転送でデッドメッセージキューを使用しない場合，次の条件を満たすメッセージを自動的に削除するかどうかを指定します。

1. 電文不正などによって転送できなかったメッセージ
2. 順序保証のグループが閉鎖した時点で，対象グループに属していた滞留メッセージ

true を指定すると 1.，2.のメッセージとも自動的に削除します。false を指定すると，1.のメッセージはキュー内に残るため，削除する場合は手作業で hrmdelmsg（メッセージの削除）コマンドを起動します。2.のメッセージは滞留メッセージとして残り，通信層のメッセージ有効期限切れになった時点で削除されます。

このプロパティの指定がない場合は false が指定されたものとして動作します。また，デッドメッセージキューを使用する場合，このプロパティの指定は無視されます。

自動削除を行うと，転送できなかったメッセージは削除処理のタイミングで削除されます。

自動削除を行わない場合は，メッセージの削除コマンドを手作業で実行し，転送できなかったメッセージを削除してください。転送できなかったメッセージを削除しなかったときは，転送キューに蓄積されます。転送キュー内のメッセージが格納できる最大数に達すると，キュー間転送が行えなくなります。

このプロパティは省略できます。

6.3 DB Connector for Reliable Messaging のコンフィグレーションプロパティの一覧

DB Connector for Reliable Messaging のコンフィグレーションプロパティは，リソースアダプタの属性を取得・編集するときに指定するプロパティです。プロパティを定義したら，サーバ管理コマンドを使用して，DB Connector for Reliable Messaging をインポートします。そのあと，プロパティを設定します。プロパティ定義の詳細については，「3.4.8 DB Connector for Reliable Messaging のプロパティ定義」を参照してください。

DB Connector for Reliable Messaging が提供するコンフィグレーションプロパティの一覧について，データベースへの接続方法ごとに説明します。

6.3.1 HiRDB Type4 JDBC Driver を使用して HiRDB に接続する場合

- DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar
  トランザクション管理をしない場合，またはローカルトランザクションを使用する場合に使用します。
  指定できるプロパティについては，表 6-4 を参照してください。
- DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar
  グローバルトランザクションを使用する場合に使用します。
  指定できるプロパティについては，表 6-5 を参照してください。

表 6-4 DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar を使用する場合に指定できるプロパティ

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
linkedResourceAdapterName	java.lang.String	連携する Reliable Messaging リソースアダプタの表示名を指定します。 DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar を使用する場合に指定できます。
description	java.lang.String	データベースへの接続に必要な接続付加情報を設定します。設定された値は，HiRDB Type4 JDBC Driver の setDescription メソッドに渡されます。
DBHostName	java.lang.String	接続する HiRDB のホスト名を設定します。設定された値は，HiRDB Type4 JDBC Driver の setDBHostName メソッドに渡されます。
environmentVariables	java.lang.String	HiRDB クライアント環境変数を指定します。設定された値は，HiRDB Type4 JDBC Driver の setEnvironmentVariables メソッドに渡されます。

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
loginTimeout	java.lang.Integer	getConnection メソッドで Connection オブジェクトを取得する際の，HiRDB サーバとの物理接続確立の最大待ち時間（秒）を指定します。 デフォルト値は 8 です。
encodeLang	java.lang.String	データ変換時の文字セット名称を設定します。設定された値は，HiRDB Type4 JDBC Driver の setEncodeLang メソッドに渡されます。
JDBC_IF_TRC	java.lang.Boolean	JDBC インタフェースメソッドトレースの取得の有無を設定します。設定された値は，HiRDB Type4 JDBC Driver の setJDBC_IF_TRACE メソッドに渡されます。 デフォルト値は false です。
TRC_NO	java.lang.Integer	JDBC インタフェースメソッドトレースのエントリ数を設定します。設定された値は，HiRDB Type4 JDBC Driver の setTRC_NO メソッドに渡されます。 デフォルト値は 500 です。
uapName	java.lang.String	アプリケーション名称を設定します。設定された値は，HiRDB Type4 JDBC Driver の setUapName メソッドに渡されます。
LONGVARBINARY_Access	java.lang.String	JDBC SQL タイプ LONGVARBINARY（HiRDB データ型である列属性 BLOB，列属性 BINARY）のデータベースアクセス方法を指定します。設定された値は，HiRDB Type4 JDBC Driver の setLONGVARBINARY_Access メソッドに渡されます。 デフォルト値は REAL です。
SQLInNum	java.lang.Integer	実行する SQL の入力パラメタの最大数を指定します。設定された値は，HiRDB Type4 JDBC Driver の setSQLInNum メソッドに渡されます。 デフォルト値は 300 です。
SQLOutNum	java.lang.Integer	実行する SQL の検索項目の最大数を指定します。設定された値は，HiRDB Type4 JDBC Driver の setSQLOutNum メソッドに渡されます。 デフォルト値は 300 です。
SQLWarningLevel	java.lang.String	SQL 実行時に発生した警告保持レベルを指定します。設定された値は，HiRDB Type4 JDBC Driver の setSQLWarningLevel メソッドに渡されます。 デフォルト値は SQLWARN です。 DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar を使用する場合，IGNORE は指定しないでください。指定した場合，Reliable Messaging のキューから取り出すメッセージが破損するおそれがあります。
SQLWarningIgnore	java.lang.Boolean	データベースから返される警告を Connection クラスで保持しないかどうかの情報を設定します。設定された値は，HiRDB

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
		Type4 JDBC Driver の setSQLWarningIgnore メソッドに渡されます。 デフォルト値は false です。
HiRDBCursorMode	java.lang.Boolean	HiRDB がコミットを行った場合に ResultSet クラスのオブジェクトを有効とするかを指定します。設定された値は，HiRDB Type4 JDBC Driver の setHiRDBCursorMode メソッドに渡されます。 デフォルト値は false です。
maxBinarySize※	java.lang.Integer	JDBC SQL タイプ LONGVARBINARY 型データ取得時のデータサイズの上限を設定します。設定された値は，HiRDB Type4 JDBC Driver の setMaxBinarySize メソッドに渡されます。DBConnector_HiRDB_Type4_CP.rar を使用する場合，デフォルト値は 0 です。 DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar を使用する場合，デフォルト値は 64000 です。 DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar を使用する場合，0 は指定しないでください。
LONGVARBINARY_AccessSize	java.lang.Integer	HiRDB サーバに対して一度に要求する JDBC SQL タイプ LONGVARBIANRY 型データの長さを指定します。設定された値は，HiRDB Type4 JDBC Driver の setLONGVARBINARY_AccessSize メソッドに渡されます。 デフォルト値は 0 です。
PreparedStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとの PreparedStatement のプールサイズを設定します。有効範囲は 0～4095 です。 デフォルト値は 10 です。
CallableStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとの CallableStatement のプールサイズを設定します。有効範囲は 0～4095 です。 デフォルト値は 10 です。
CancelStatement	java.lang.Boolean	トランザクションタイムアウトや UAP 強制停止時にステートメントキャンセルを実行するかどうかを設定します。 • true を指定した場合 実行中の SQL をキャンセルします。 • false を指定した場合 実行中の SQL をキャンセルしません。 デフォルト値は true です。 DBConnector_HiRDB_Type4_CP.rar を使用する場合に指定できます。
logLevel	java.lang.String	DB Connector が出力するログトレースのレベルを指定します。 • 0 または ERROR • 10 または WARNING

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
		• 20 または INFORMATION デフォルト値は 0 または ERROR です。

表 6-5　DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar を使用する場合に指定できるプロパティ

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
linkedResourceAdapterName	java.lang.String	連携する Reliable Messaging リソースアダプタの表示名を指定します。 DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar を使用する場合に指定できます。
description	java.lang.String	データベースへの接続に必要な接続付加情報を設定します。設定された値は，HiRDB Type4 JDBC Driver の setDescription メソッドに渡されます。
DBHostName	java.lang.String	接続する HiRDB のホスト名を設定します。設定された値は，HiRDB Type4 JDBC Driver の setDBHostName メソッドに渡されます。
environmentVariables	java.lang.String	HiRDB クライアント環境変数を指定します。設定された値は，HiRDB Type4 JDBC Driver の setEnvironmentVariables メソッドに渡されます。
XAOpenString	java.lang.String	XA_OPEN 文字列を設定します。設定された値は，HiRDB Type4 JDBC Driver の setXAOpenString メソッドに渡されます。
loginTimeout	java.lang.Integer	getConnection メソッドで Connection オブジェクトを取得する際の，HiRDB サーバとの物理接続確立の最大待ち時間（秒）を指定します。設定された値は，HiRDB Type4 JDBC Driver の setLoginTimeout メソッドに渡されます。 デフォルト値は 8 です。
encodeLang	java.lang.String	データ変換時の文字セット名称を設定します。設定された値は，HiRDB Type4 JDBC Driver の setEncodeLang メソッドに渡されます。
JDBC_IF_TRC	java.lang.Boolean	JDBC インタフェースメソッドトレースの取得の有無を設定します。設定された値は，HiRDB Type4 JDBC Driver の setJDBC_IF_TRACE メソッドに渡されます。
TRC_NO	java.lang.Integer	JDBC インタフェースメソッドトレースのエントリ数を設定します。設定された値は，HiRDB Type4 JDBC Driver の setTRC_NO メソッドに渡されます。 デフォルト値は 500 です。
uapName	java.lang.String	アプリケーション名称を設定します。設定された値は，HiRDB Type4 JDBC Driver の setUapName メソッドに渡されます。

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
LONGVARBINARY_Access	java.lang.String	JDBC SQL タイプ LONGVARBINARY（HiRDB データ型である列属性 BLOB，列属性 BINARY）のデータベースアクセス方法を指定します。設定された値は，HiRDB Type4 JDBC Driver の setLONGVARBINARY_Access メソッドに渡されます。 デフォルト値は REAL です。
SQLInNum	java.lang.Integer	実行する SQL の入力パラメタの最大数を指定します。設定された値は，HiRDB Type4 JDBC Driver の setSQLInNum メソッドに渡されます。 デフォルト値は 300 です。
SQLOutNum	java.lang.Integer	実行する SQL の検索項目の最大数を指定します。設定された値は，HiRDB Type4 JDBC Driver の setSQLOutNum メソッドに渡されます。 デフォルト値は 300 です。
SQLWarningLevel	java.lang.String	SQL 実行時に発生した警告保持レベルを指定します。設定された値は，HiRDB Type4 JDBC Driver の setSQLWarningLevel メソッドに渡されます。 デフォルト値は SQLWARN です。 DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar を使用する場合，IGNORE は指定しないでください。指定した場合，Reliable Messaging のキューから取り出すメッセージが破損するおそれがあります。
SQLWarningIgnore	java.lang.Boolean	データベースから返される警告を Connection クラスで保持しないかどうかの情報を設定します。設定された値は，HiRDB Type4 JDBC Driver の setSQLWarningIgnore メソッドに渡されます。 デフォルト値は false です。
HiRDBCursorMode	java.lang.Boolean	HiRDB がコミットを行った場合に ResultSet クラスのオブジェクトを有効とするかを指定します。設定された値は，HiRDB Type4 JDBC Driver の setHiRDBCursorMode メソッドに渡されます。 デフォルト値は false です。
maxBinarySize※	java.lang.Integer	JDBC SQL タイプ LONGVARBINARY 型データ取得時のデータサイズの上限を設定します。設定された値は，HiRDB Type4 JDBC Driver の setMaxBinarySize メソッドに渡されます。DBConnector_HiRDB_Type4_XA.rar を使用する場合，デフォルト値は 0 です。 DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar を使用する場合，デフォルト値は 64000 です。 DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar を使用する場合，0 は指定しないでください。
LONGVARBINARY_AccessSize	java.lang.Integer	HiRDB サーバに対して一度に要求する JDBC SQL タイプ LONGVARBIANRY 型データの長さを指定します。設定され

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
		た値は，HiRDB Type4 JDBC Driver の setLONGVARBINARY_AccessSize メソッドに渡されます。 デフォルト値は 0 です。
XACloseString	java.lang.String	XA クローズ文字列を設定します。設定された値は，HiRDB Type4 JDBC Driver の setXACloseString メソッドに渡されます。
XALocalCommitMode	java.lang.Boolean	トランザクションがグローバルトランザクションでない場合にオートコミット機能を有効にするかを設定します。設定された値は HiRDB Type4 JDBC Driver の setXALocalCommitMode メソッドに渡されます。 デフォルト値は true です。 このプロパティはデフォルト値から変更しないでください。
PreparedStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとの PreparedStatement のプールサイズを設定します。有効範囲は 0～4095 です。 デフォルト値は 10 です。
CallableStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとの CallableStatement のプールサイズを設定します。有効範囲は 0～4095 です。 デフォルト値は 10 です。
CancelStatement	java.lang.Boolean	トランザクションタイムアウトや UAP 強制停止時にステートメントキャンセルを実行するかどうかを設定します。 • true の場合 実行中の SQL をキャンセルします。 • false の場合 実行中の SQL をキャンセルしません。 デフォルト値は true です。 DBConnector_HiRDB_Type4_XA.rar を使用する場合に指定できます。
logLevel	java.lang.String	DB Connector が出力するログトレースのレベルを指定します。 • 0 または ERROR • 10 または WARNING • 20 または INFORMATION デフォルト値は 0 または ERROR です。

注※

maxBinarySize プロパティに設定するバッファサイズを見積もるための計算式は次のとおりです。

ローカルキューまたは転送キューを使用する場合

（5000 ＋ N ＋ M ＋ L ＋ S）（単位：バイト）

- N ＝ユーザが設定できる JMS ヘッダの設定サイズ

設定するヘッダによって次の値を加算します。
JMSReplyTo：シリアライズしたオブジェクトのサイズ
JMSCorrelationID：半角文字数＋全角文字数×3
JMSType：半角文字数＋全角文字数×3

- M ＝ JMS 定義プロパティの設定サイズ
  設定するプロパティによって次の値を加算します。
  JMSXGroupID：半角文字数＋全角文字数×3
  JMSXGroupSeq：100
- L ＝ l ＋ m
  l：ユーザ定義プロパティ名のサイズ（(半角文字数＋全角文字数×3）×プロパティ数）
  m：ユーザ定義プロパティ値のサイズ
  プロパティの型によって次の値を加算します。
  java.lang.String：（半角文字数＋全角文字数×3）×プロパティ数
  java.lang.String 以外：100×プロパティ数
- S ＝ JMS メッセージのペイロード設定サイズ
  メッセージのインタフェースによって次の値を加算します。
  TextMessage：半角文字数＋全角文字数×3
  BytesMessage：半角文字数＋全角文字数×2
  ObjectMessage：シリアライズしたオブジェクトのサイズ

共用キューを使用する場合
（1000 ＋ S）（単位：バイト）

- S ＝ JMS メッセージのペイロード設定サイズ
  メッセージのインタフェースによって次の値を加算します。
  TextMessage：半角文字数＋全角文字数×3
  BytesMessage：半角文字数＋全角文字数×2
  ObjectMessage：シリアライズしたオブジェクトのサイズ

データサイズには，0 を指定しないでください。0 を指定した場合，メモリ不足が発生し，J2EE サーバがダウンするおそれがあります。

HiRDB を使用する場合で，サイズの大きいメッセージを送受信するときには，連携先の DB Connector の maxBinarySize プロパティをメッセージのサイズより十分大きく設定してください。maxBinarySize プロパティの値が小さいと，メッセージの受信に失敗する場合があります。

6.3.2 Oracle JDBC Thin Driver を使用して Oracle に接続する場合

- DBConnector_Oracle_CP_Cosminexus_RM.rar

トランザクション管理をしない場合，またはローカルトランザクションを使用する場合に使用します。
指定できるプロパティについては，表 6-6 を参照してください。

- DBConnector_Oracle_XA_Cosminexus_RM.rar
  グローバルトランザクションを使用する場合に使用します。
  指定できるプロパティについては，表 6-7 を参照してください。

なお，プロパティで設定可能な値については，Oracle のマニュアルを参照してください。

表 6-6　DBConnector_Oracle_CP_Cosminexus_RM.rar を使用する場合に指定できるプロパティ

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
linkedResourceAdapterName	java.lang.String	連携する Reliable Messaging リソースアダプタの表示名を指定します。 DBConnector_Oracle_CP_Cosminexus_RM.rar を使用する場合に指定できます。
databaseName	java.lang.String	Oracle サーバ上の特定のデータベース名（SID）を指定します。設定された値は，Oracle JDBC Thin Driver の setDatabaseName メソッドに渡されます
serverName	java.lang.String	Oracle サーバのホスト名または IP アドレスを指定します。設定された値は，Oracle JDBC Thin Driver の setServerName メソッドに渡されます。
portNumber	java.lang.Integer	Oracle のサーバが要求をリスニングするポート番号を指定します。デフォルトは 1521 番ポートです。設定された値は，Oracle JDBC Thin Driver の setPortNumber メソッドに渡されます。
url	java.lang.String	Oracle JDBC Thin Driver がデータベースに接続するために必要な JDBC URL を指定します。 このプロパティに値が設定された場合，databaseName，portNumber，serverName で指定された値は無視されます。また，ユーザが url で指定を行う場合は JDBC URL に thin ドライバを指定します。 （例） jdbc:oracle:thin:@ServerA:1521:service1
loginTimeout	java.lang.Integer	データベースへの接続試行のタイムアウト（単位：ミリ秒）を指定します。0 を指定するとタイムアウトは無限となり，接続が確立されるかエラーが発生するまでブロックされます。デフォルト値は 8000 です。設定された値は Oracle JDBC Thin Driver の setLoginTimeout メソッドに渡されます。Oracle JDBC Thin Driver 9.2.0.8 以降，10.1.0.5 以降および 10.2 以降の場合は秒単位に切り上げて setLoginTimeout メソッドに渡されます。

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
PreparedStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとのPreparedStatement のプールサイズを設定します。デフォルトは 10 です。
CallableStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとのCallableStatement のプールサイズを設定します。デフォルトは 10 です。
CancelStatement	java.lang.Boolean	トランザクションタイムアウトや業務アプリケーション強制停止時に，Statement クラス，CallableStatement クラスおよび PreparedStatement クラスで実行中の SQL をキャンセルするかどうかを指定します。 • true を指定した場合 実行中の SQL をキャンセルします。 • false を指定した場合 実行中の SQL をキャンセルしません。 デフォルト値は true です。 専用サーバ接続をする場合は，false を指定してください。 DBConnector_Oracle_CP.rar を使用する場合に，指定できます。
ConnectionIDUpdate	java.lang.Boolean	コネクション ID を DataSource#getConnection メソッドごとに更新するかどうかを指定します。 • true を指定した場合 DataSource#getConnection メソッドのたびにコネクション ID を生成します。 • false を指定した場合 DataSource#getConnection メソッドで新規の物理コネクションを確立したときにコネクション ID を生成し，そのあとは更新しません。 デフォルト値は false です。
logLevel	java.lang.String	DB Connector が出力するログ・トレースのレベルを指定します。 次の値が指定できます。 • 0 または ERROR • 10 または WARNING • 20 または INFORMATION デフォルトは，0 または ERROR です。

表 6-7　DBConnector_Oracle_XA_Cosminexus_RM.rar を使用する場合に指定できるプロパティ

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
linkedResourceAdapterName	java.lang.String	連携する Reliable Messaging リソースアダプタの表示名を指定します。 DBConnector_Oracle_CP_Cosminexus_RM.rar を使用する場合に指定できます。
databaseName	java.lang.String	Oracle サーバ上の特定のデータベース名（SID）を指定します。設定された値は，Oracle JDBC Thin Driver の setDatabaseName メソッドに渡されます。
serverName	java.lang.String	Oracle サーバのホスト名または IP アドレスを指定します。設定された値は，Oracle JDBC Thin Driver の setServerName メソッドに渡されます。
portNumber	java.lang.Integer	Oracle のサーバが要求をリスニングするポート番号を指定します。デフォルトは 1521 番ポート。設定された値は，Oracle JDBC Thin Driver の setPortNumber メソッドに渡されます。
url	java.lang.String	Oracle JDBC Thin Driver がデータベースに接続するために必要な JDBC URL を指定します。 このプロパティに値が設定された場合，databaseName,portNumber,serverName で指定された値は無視されます。また，ユーザが url で指定を行う場合は JDBC URL に thin ドライバを指定します。
loginTimeout	java.lang.Integer	データベースへの接続試行のタイムアウト（単位：ミリ秒）を指定します。0 を指定するとタイムアウトは無限となり，接続が確立されるかエラーが発生するまでブロックされます。デフォルト値は 8000 です。設定された値は Oracle JDBC Thin Driver の setLoginTimeout メソッドに渡されます。Oracle JDBC Thin Driver 9.2.0.8 以降，10.1.0.5 以降および 10.2 以降の場合は秒単位に切り上げて setLoginTimeout メソッドに渡されます。
sessionTimeout	java.lang.Integer	Oracle サーバでのセッションタイムアウト（トランザクションブランチがアクティブでない状態でいられる最大時間）を秒単位で指定します。J2EE サーバのトランザクションタイムアウトよりも長い時間を指定する必要があります。デフォルト値は 300 秒です。設定された値は Oracle JDBC Thin Driver の XAResource.setTransactionTimeout メソッドに渡されます。
PreparedStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとの PreparedStatement のプールサイズを設定します。デフォルト値は 10 です。
CallableStatementPoolSize	java.lang.Integer	コネクションプールに割り当てられるコネクションごとの CallableStatement のプールサイズを設定します。デフォルト値は 10 です。

プロパティ名（config-property-name）	データ型（config-property-type）	プロパティの値（config-property-value）
CancelStatement	java.lang.Boolean	トランザクションタイムアウトや業務アプリケーション強制停止時に，Statement クラス，CallableStatement クラスおよび PreparedStatement クラスで実行中の SQL をキャンセルするかどうかを指定します。 • true を指定した場合 実行中の SQL をキャンセルします。 • false を指定した場合 実行中の SQL をキャンセルしません。 デフォルト値は true です。 専用サーバ接続をする場合は，false を指定してください。DBConnector_Oracle_XA.rar を使用する場合に指定できます。
ConnectionIDUpdate	java.lang.Boolean	コネクション ID を DataSource#getConnection メソッドごとに更新するかどうかを指定します。 • true を指定した場合 DataSource#getConnection メソッドのたびにコネクション ID を生成します。 • false を指定した場合 DataSource#getConnection メソッドで新規の物理コネクションを確立したときにコネクション ID を生成し，そのあとは更新しません。 デフォルト値は false です。
logLevel	java.lang.String	DB Connector が出力するログ・トレースのレベルを指定します。 次の値が指定できます。 • 0 または ERROR • 10 または WARNING • 20 または INFORMATION デフォルト値は，0 または ERROR です。

7 インタフェース

Reliable Messaging のアプリケーションは，Reliable Messaging が提供するインタフェースを使用して実装します。アプリケーションの実装については，「2.6　アプリケーションからのメッセージ操作」を参照してください。

この章では，Reliable Messaging が提供するインタフェースの種類，インタフェースの種類別のクラス，メソッドの詳細およびアプリケーションの障害時に設定される障害コードについて説明します。

7.1　インタフェースの種類

7.1.1　Reliable Messaging が提供するインタフェースの種類

Reliable Messaging が提供するインタフェース用のパッケージには，次の 2 種類があります。

- JMS インタフェース

  Reliable Messaging がメッセージを送受信する際に使用するインタフェースです。永続版リソースアダプタ，非永続版リソースアダプタの両方で使用します。

  JMS インタフェースの詳細については，「7.2　JMS インタフェースの一覧」，「7.3　JMS インタフェース継承図」，および「7.4　JMS インタフェースの詳細」を参照してください。

- 転送データ相互接続用インタフェース

  BytesContainer を使用して，Reliable Messaging 同士，または Reliable Messaging と他ベンダのメッセージングシステムなどとの間でメッセージを送受信する際に使用するインタフェースです。非永続版リソースアダプタでは使用できません。

  転送データ相互接続用インタフェースの詳細については，「7.5　転送データ相互接続用インタフェースの一覧」，「7.6　転送データ相互接続用インタフェース継承図」，「7.7　転送データ相互接続用インタフェースの使い方」，および「7.8　転送データ相互接続用インタフェースの詳細」を参照してください。

7.1.2　非永続版リソースアダプタ使用時の注意事項

非永続版リソースアダプタの場合，永続版リソースアダプタと同様に JMS インタフェースを使用できます。ただし，転送データ相互接続用インタフェースは使用できません。

また，JMS インタフェース使用時には次の点に注意してください。

- BytesContainer を使用した処理はできません。
- ローカルキューに関するインタフェースだけを使用できます。そのほかの種類のキューに関するインタフェースは使用できません。
- 非永続版リソースアダプタの場合，DB を使用しないため，DB や DB Connector に関係するインタフェースは使用できません。

7.2 JMS インタフェースの一覧

Reliable Messaging の JMS インタフェースは，永続版リソースアダプタ，非永続版リソースアダプタともに使用できます。

JMS インタフェースは，次に示すパッケージによって提供されます。

```
javax.jms
```

Reliable Messaging が提供する JMS インタフェースの一覧を次の表に示します。

表 7-1 JMS インタフェースの一覧

項番	インタフェース名※	機能
1	BytesMessage	設定したデータをバイトストリームで保持します。提供する機能は，ペイロードの設定と取得およびアクセスモードの変更です。 なお，Message インタフェースが提供する機能も含みます。
2	ConnectionMetaData	JMS の基本情報を提供します。提供する基本情報は，JMS API のバージョン，JMS プロバイダのバージョン，JMS プロバイダ名および JMSX プロパティ名の列挙などです。
3	DeliveryMode	メッセージの永続化と非永続化を示す定数を提供します。
4	Message	BytesMessage，ObjectMessage および TextMessage のルートインタフェースです。提供する機能は，ヘッダの設定と取得，プロパティの設定，取得と初期化，ペイロードの初期化およびメッセージ受信の承認です。
5	ObjectMessage	設定したデータを java.io.Serializable 型で保持します。提供する機能は，ペイロードの設定と取得です。 なお，Message インタフェースが提供する機能も含みます。
6	Queue （Destination）	JMS プロバイダ特有のキュー名を保持します。提供する機能は，キュー名の取得です。
7	QueueBrowser	JMS クライアントがキューからメッセージを削除しないで参照するための機能を提供します。提供する機能は，現在のキューにあるメッセージ一覧の取得，メッセージセレクタの取得，QueueBrowser オブジェクトのクローズおよびキューの取得です。
8	QueueConnection （Connection）	アプリケーションが Reliable Messaging にアクセスするために使用するアプリケーションレベルのハンドルを提供します。提供する機能は，QueueSession オブジェクトの生成，配送の開始と停止，ConnectionMetaData オブジェクトの取得，コネクションのクローズです。
9	QueueConnectionFactory （ConnectionFactory）	コネクションを生成する機能を提供します。
10	QueueReceiver （MessageConsumer）	JMS クライアントがキューからメッセージを取得するための機能を提供します。提供する機能は，メッセージの受信，メッセージセレクタの取得，QueueReceiver オブジェクトのクローズおよびキューの取得です。

項番	インタフェース名※	機能
11	QueueSender (MessageProducer)	JMS クライアントがあて先にメッセージを送信するための機能を提供します。提供する機能は，メッセージの送信，QueueSender オブジェクトのクローズです。
12	QueueSession (Session)	アプリケーションが論理的なコネクションハンドルとして利用するインタフェースです。提供する機能は，Message，BytesMessage，ObjectMessage，TextMessage，QueueSender，QueueReceiver および QueueBrowser オブジェクトの生成，ローカルトランザクションのコミットとロールバック，セッションのクローズならびに配送するメッセージのリカバーです。
13	TextMessage	設定したデータを java.lang.String 型で保持します。提供する機能は，ペイロードの設定と取得です。 なお，Message インタフェースが提供する機能も含みます。

注

Oracle Corporation が提供する JMS 1.0.2b と Reliable Messaging が提供する JMS インタフェースとの機能差については，「付録 A　JMS 仕様との差異」を参照してください。

注※

丸括弧内は継承元となる親インタフェース名です。

JMS インタフェース使用時の例外クラスは，次に示すパッケージによって提供されます。

```
javax.jms
```

JMS インタフェース使用時の例外クラスの一覧を次の表に示します。

表 7-2　JMS インタフェースの例外クラスの一覧

項番	例外クラス名	例外が発生する状況
1	IllegalStateException	メソッドが不正なタイミングや不適切なタイミングで発行された場合，または JMS プロバイダが要求された操作に対して適切な状態にない場合
2	InvalidClientIDException	JMS クライアントがコネクションの JMS クライアント識別子を JMS プロバイダに拒否された値に設定しようとした場合
3	InvalidDestinationException	あて先が JMS プロバイダによって理解されない，つまりあて先が有効でない場合
4	InvalidSelectorException	JMS クライアントが不正な構文のメッセージセレクタを JMS プロバイダに指定した場合
5	JMSException	JMS API 例外が発生した場合。すべての JMS API 例外のルートクラスであるため，特定のタイミングに該当しません。
6	JMSSecurityException	JMS クライアントが送信したユーザ名またはパスワードが JMS プロバイダに拒否された場合
7	MessageEOFException	BytesMessage インタフェースの読み取り中に予期しないストリーム終端に達した場合

項番	例外クラス名	例外が発生する状況
8	MessageFormatException	JMS クライアントがメッセージによってサポートされていないデータタイプを使おうとした場合，またはメッセージデータを間違った型で読み取ろうとした場合
9	MessageNotReadableException	JMS クライアントが書き込み専用のメッセージを読み取ろうとした場合
10	MessageNotWriteableException	JMS クライアントが読み取り専用のメッセージに書き込もうとした場合
11	ResourceAllocationException	JMS プロバイダがメソッドによって要求されたリソースを割り当てられない場合
12	TransactionInProgressException	トランザクションが進行中であるために操作が無効である場合
13	TransactionRolledBackException	Session.commit()メソッドの発行の結果が現在のトランザクションのロールバックに帰着する場合

注意事項

Reliable Messaging を一つの J2EE サーバに複数デプロイして，一つのアプリケーションから複数の Reliable Messaging を利用する場合，一つの Reliable Messaging 側で作成したオブジェクトを他方の Reliable Messaging 側に渡すような操作はしないでください。

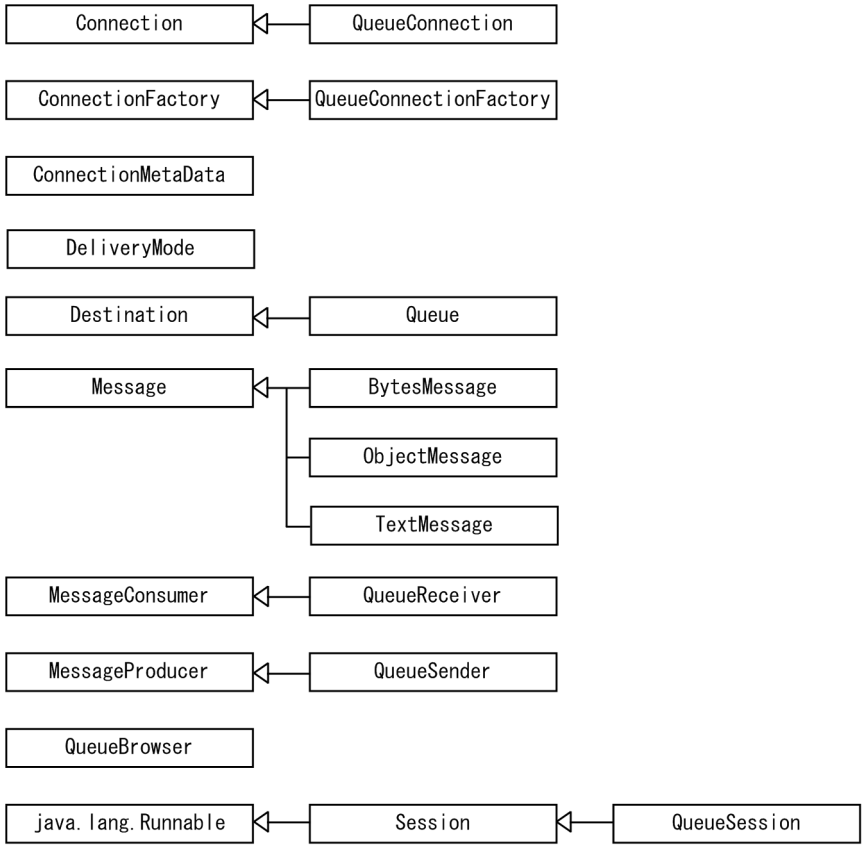
例えば，Reliable Messaging1 で作成したメッセージを Reliable Messaging2 の javax.jms.QueueSender によって送信するなどの操作はしないでください。

7.3　JMS インタフェース継承図

Reliable Messaging が提供する JMS インタフェースおよび例外クラスには，それぞれ継承関係があります。

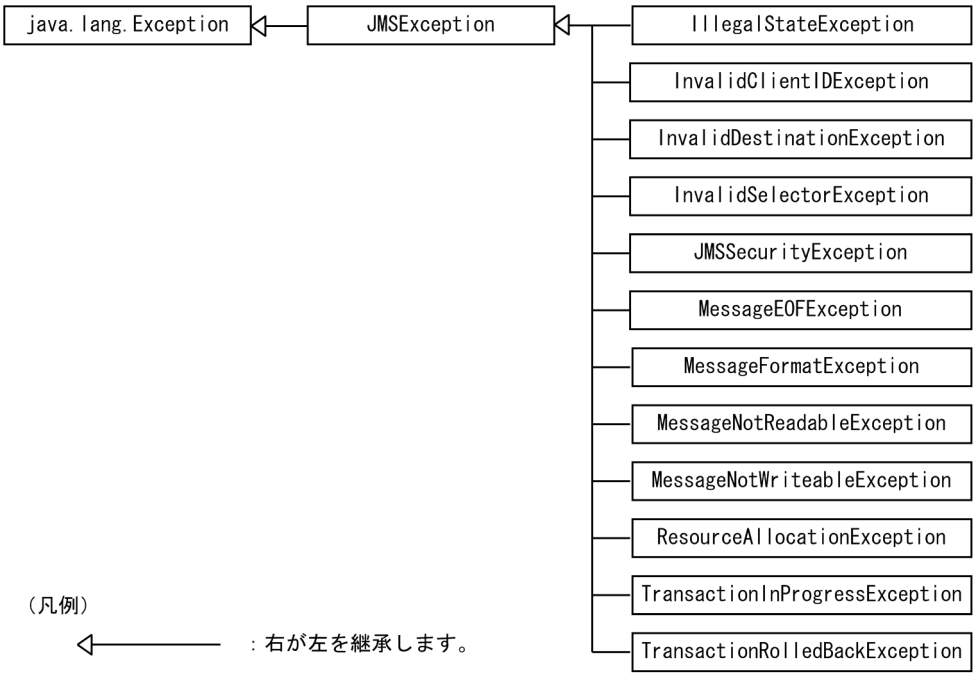
JMS インタフェースの継承を次の図に示します。

図 7-1　JMS インタフェースの継承



例外クラスの継承を次の図に示します。

図 7-2　例外クラスの継承



7.4　JMS インタフェースの詳細

JMS インタフェースを名前のアルファベット順に説明します。

7.4.1　BytesMessage インタフェース

BytesMessage インタフェースはバイトストリームを含むメッセージを送受信するために使用します。BytesMessage インタフェースは Message インタフェースを継承しているため，Message インタフェースの機能を持っています。

(1)　ペイロードの設定と取得

BytesMessage インタフェースを使用することによってペイロードを設定および取得できます。このペイロードはバイトストリームです。JMS メッセージの各要素については，「2.5.1　JMS メッセージの構成」を参照してください。

(2)　形式

```
public interface BytesMessage extends Message
{
  public boolean readBoolean() throws JMSException;
  public byte    readByte() throws JMSException;
  public int     readBytes(byte[] value) throws JMSException;
  public int     readBytes(byte[] value, int length)
                   throws JMSException;
  public char    readChar() throws JMSException;
  public double  readDouble() throws JMSException;
  public float   readFloat() throws JMSException;
  public int     readInt() throws JMSException;
  public long    readLong() throws JMSException;
  public short   readShort() throws JMSException;
  public int     readUnsignedByte() throws JMSException;
  public int     readUnsignedShort() throws JMSException;
  public java.lang.String
                 readUTF() throws JMSException;
  public void    reset() throws JMSException;
  public void    writeBoolean(boolean value) throws JMSException;
  public void    writeByte(byte value) throws JMSException;
  public void    writeBytes(byte[] value) throws JMSException;
  public void    writeBytes(byte[] value, int offset, int length)
                   throws JMSException;
  public void    writeChar(char value) throws JMSException;
  public void    writeDouble(double value) throws JMSException;
  public void    writeFloat(float value) throws JMSException;
  public void    writeInt(int value) throws JMSException;
  public void    writeLong(long value) throws JMSException;
  public void    writeObject(java.lang.Object value)
                   throws JMSException;
  public void    writeShort(short value) throws JMSException;
```

```
  public void    writeUTF(java.lang.String value) throws JMSException;
}
```

(3)　フィールド

ありません。

(4)　メソッド

「(2)　形式」に記載した順序で各メソッドを説明します。

(a)　readBoolean メソッド

```
public boolean readBoolean() throws JMSException
```

バイトストリームから boolean 型の値を読み取ります。

- 引数
  ありません。
- 戻り値
  読み取られた boolean の値です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　readByte メソッド

```
public byte readByte() throws JMSException
```

バイトストリームから符号付き 8 ビット値を読み取ります。

- 引数
  ありません。
- 戻り値
  符号付き 8 ビットとみなされるバイトストリームの次のバイト。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(c)　readBytes メソッド

```
public int readBytes(byte[] value) throws JMSException
```

バイトストリームからバイト配列を読み取ります。バイトストリームの長さより配列 value の長さが短い場合，配列 value の長さまでのバイトストリームを読み取ることができます。残りのバイトストリームも再度メソッドを発行することによって配列 value の長さまで読み取ることができます。バイトストリームの長さより配列 value の長さが長い場合，バイトストリームをすべて読み取ることができます。

このとき，読み取られる総バイト数を示す戻り値は，配列 value の長さよりも短くなり，バイトストリームから読み取るバイトが残っていないことを表します。バイトストリームからさらに読み取りを実行すると，-1 を返します。

- 引数

引数名	説明
value	データの読み取り先のバッファ

- 戻り値

  バッファに読み取られるバイトの総数。ストリームの終端に達してデータがなくなった場合は-1 です。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(d)　readBytes メソッド

```
public int readBytes(byte[] value, int length) throws JMSException
```

バイトストリームの一部を読み取ります。

- 引数

引数名	説明
value	データの読み取り先のバッファ
length	読み取るバイト数（value.length 以下）

- 戻り値

  バッファに読み取られるバイトの総数。ストリームの終端に達してデータがなくなった場合は-1 です。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(e)　readChar メソッド

```
public char readChar() throws JMSException
```

バイトストリームから Unicode 文字値を読み取ります。

- 引数

  ありません。

- 戻り値

  バイトストリームの次の 2 バイトで表される Unicode 文字。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(f)　readDouble メソッド

```
public double readDouble() throws JMSException
```

バイトストリームから double 値を読み取ります。

- 引数

ありません。

- 戻り値
  バイトストリームの次の 8 バイトを double と解釈した値です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(g)　readFloat メソッド

```
public float readFloat() throws JMSException
```

バイトストリームから float 値を読み取ります。

- 引数
  ありません。
- 戻り値
  バイトストリームの次の 4 バイトを float と解釈した値です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(h)　readInt メソッド

```
public int readInt() throws JMSException
```

バイトストリームから符号付き 32 ビット整数を読み取ります。

- 引数
  ありません。
- 戻り値

バイトストリームの次の 4 バイトを int と解釈した値です。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(i)　readLong メソッド

```
public long readLong() throws JMSException
```

バイトストリームから符号付き 64 ビット整数を読み取ります。

- 引数
  ありません。
- 戻り値
  バイトストリームの次の 8 バイトを long と解釈した値です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(j)　readShort メソッド

```
public short readShort() throws JMSException
```

バイトストリームから符号付き 16 ビット数を読み取ります。

- 引数
  ありません。
- 戻り値
  バイトストリームの次の 2 バイトを符号付き 16 ビットと解釈した値です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(k)　readUnsignedByte メソッド

```
public int readUnsignedByte() throws JMSException
```

バイトストリームから符号なし 8 ビット数を読み取ります。

- 引数
  ありません。
- 戻り値
  バイトストリームの次のバイトを符号なし 8 ビットと解釈した値です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(l)　readUnsignedShort メソッド

```
public int readUnsignedShort() throws JMSException
```

バイトストリームから符号なし 16 ビット数を読み取ります。

- 引数
  ありません。
- 戻り値
  バイトストリームの次の 2 バイトを符号なし 16 ビット整数と解釈した値です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(m)　readUTF メソッド

```
public java.lang.String readUTF() throws JMSException
```

修正 UTF-8 形式を使用してエンコードされた文字列をバイトストリームから読み取ります。一度の発行で読み取られる文字列は 65535 バイトまでです。

- 引数
  ありません。
- 戻り値
  バイトストリームから読み取られた Unicode 文字列。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの読み取りに失敗しました。
MessageEOFException	バイトストリームの予期しない終端に達しました。
MessageNotReadableException	書き込み専用モードのペイロードから情報を読み取ろうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(n)　reset メソッド

```
public void reset() throws JMSException
```

ペイロードを書き込み専用モードから読み取り専用モードに移行してから，バイトストリームを先頭に再配置します。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージのリセットに失敗しました。
MessageFormatException	メッセージ形式不正が原因でメッセージのリセットに失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(o)　writeBoolean メソッド

```
public void writeBoolean(boolean value) throws JMSException
```

boolean をバイトストリームに 1 バイト値として書き込みます。

- 引数

引数名	説明
value	書き込まれる boolean 値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(p)　writeByte メソッド

```
public void writeByte(byte value) throws JMSException
```

byte をバイトストリームに 1 バイト値として書き込みます。

- 引数

引数名	説明
value	書き込まれる byte 値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(q)　writeBytes メソッド

```
public void writeBytes(byte[] value) throws JMSException
```

バイト配列をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれるバイト配列

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(r)　writeBytes メソッド

```
public void writeBytes(byte[] value, int offset, int length)
  throws JMSException
```

バイト配列の一部をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれるバイト配列値
offset	バイト配列内の初期オフセット
length	書き込まれるバイト数

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(s)　writeChar メソッド

```
public void writeChar(char value) throws JMSException
```

上位バイトを先頭とする 2 バイト値として，char 型引数をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれる char 値

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(t)　writeDouble メソッド

```
public void writeDouble(double value) throws JMSException
```

Double.doubleToLongBits()メソッドを使用して double 型引数を long に変換し，上位バイトを先頭とする 8 バイトとして，long 値をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれる double 値

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(u)　writeFloat メソッド

```
public void writeFloat(float value) throws JMSException
```

Float.floatToIntBits()メソッドを使用して float 型引数を int に変換し，上位バイトを先頭とする 4 バイトとして，int 値をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれる float 値

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(v)　writeInt メソッド

```
public void writeInt(int value) throws JMSException
```

上位バイトを先頭とする 4 バイト値として，int 型引数をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれる int 値

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(w)　writeLong メソッド

```
public void writeLong(long value) throws JMSException
```

上位バイトを先頭とする 8 バイト値として，long 型引数をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれる long 値

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(x)　writeObject メソッド

```
public void writeObject(java.lang.Object value) throws JMSException
```

オブジェクトをバイトストリームに書き込みます。引数に設定できる型はオブジェクト化されたプリミティブオブジェクト型（Integer，Double および Long など），String オブジェクトおよびバイト配列だけです。

- 引数

引数名	説明
value	書き込まれる Java プログラミング言語のオブジェクト（Java オブジェクト）

- 戻り値

ありません。

- 例外

例外クラス	説明
java.lang.NullPointerException	引数の値が null なので，ペイロードの書き込みに失敗しました。
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageFormatException	引数の型が無効なためペイロードの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(y)　writeShort メソッド

```
public void writeShort(short value) throws JMSException
```

上位バイトを先頭とする 2 バイト値として，short 型引数をバイトストリームに書き込みます。

- 引数

引数名	説明
value	書き込まれる short 値

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(z)　writeUTF メソッド

```
public void writeUTF(java.lang.String value) throws JMSException
```

マシンに依存しない UTF-8 エンコーディング形式を使用して，文字列をバイトストリームに書き込みます。一度の発行で書き込まれる文字列は 65535 バイトまでです。

- 引数

引数名	説明
value	書き込まれる String 型の値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの書き込みに失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.2　ConnectionMetaData インタフェース

ConnectionMetaData インタフェースは JMS の基本情報を提供します。

(1)　JMS 基本情報の取得

次に示す JMS 基本情報を取得できます。

- JMS API のメジャーバージョン番号
- JMS API のマイナーバージョン番号
- JMS API のバージョン
- JMS プロバイダのメジャーバージョン番号
- JMS プロバイダのマイナーバージョン番号
- JMS プロバイダのバージョン
- JMS プロバイダ名
- JMSX プロパティ名の列挙

(2)　形式

```
public interface ConnectionMetaData
{
  public int      getJMSMajorVersion() throws JMSException;
  public int      getJMSMinorVersion() throws JMSException;
  public java.lang.String
                  getJMSProviderName() throws JMSException;
  public java.lang.String
                  getJMSVersion() throws JMSException;
  public java.util.Enumeration
```

```
                  getJMSXPropertyNames() throws JMSException;
  public int      getProviderMajorVersion() throws JMSException;
  public int      getProviderMinorVersion() throws JMSException;
  public java.lang.String
                  getProviderVersion() throws JMSException;
}
```

(3)　フィールド

ありません。

(4)　メソッド

「(2)　形式」に記載した順序で各メソッドを説明します。

(a)　getJMSMajorVersion メソッド

```
public int getJMSMajorVersion() throws JMSException
```

JMS API のメジャーバージョン番号を示す整数である 1 を返します。

- 引数
  ありません。
- 戻り値
  JMS API のメジャーバージョン番号。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　getJMSMinorVersion メソッド

```
public int getJMSMinorVersion() throws JMSException
```

JMS API のマイナーバージョン番号を示す整数である 0 を返します。

- 引数
  ありません。
- 戻り値
  JMS のマイナーバージョン番号。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(c)　getJMSProviderName メソッド

```
public java.lang.String getJMSProviderName() throws JMSException
```

JMS プロバイダ名を示す文字列である"Cosminexus Reliable Messaging"を返します。

- 引数
  ありません。
- 戻り値
  JMS プロバイダ名。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(d)　getJMSVersion メソッド

```
public java.lang.String getJMSVersion() throws JMSException
```

JMS API のバージョンを示す文字列である"1.0"を返します。

- 引数
  ありません。
- 戻り値
  JMS API のバージョン。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(e) getJMSXPropertyNames メソッド

```
public java.util.Enumeration getJMSXPropertyNames() throws JMSException
```

Reliable Messaging で使用できる JMS 定義 JMSX プロパティ名を示す文字列の列挙（"JMSXRcvTimestamp"，"JMSXDeliveryCount"，"JMSXGroupID"および"JMSXGroupSeq"）を返します。

- 引数
  ありません。
- 戻り値
  JMSX プロパティ名の列挙。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(f) getProviderMajorVersion メソッド

```
public int getProviderMajorVersion() throws JMSException
```

JMS プロバイダのメジャーバージョン番号を返します。この値は，Reliable Messaging のバージョンに対応します。例えば，Reliable Messaging 01-02 の場合，1 を返します。

- 引数
  ありません。
- 戻り値
  JMS プロバイダのメジャーバージョン番号。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(g) getProviderMinorVersion メソッド

```
public int getProviderMinorVersion() throws JMSException
```

JMS プロバイダのマイナーバージョン番号を返します。この値は，Reliable Messaging のリビジョンに対応します。例えば，Reliable Messaging 01-02 の場合，2 を返します。

- 引数
  ありません。
- 戻り値
  JMS プロバイダのマイナーバージョン番号。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(h)　getProviderVersion メソッド

```
public java.lang.String getProviderVersion() throws JMSException
```

JMS プロバイダのバージョンを示す文字列を返します。この文字列は，Reliable Messaging のバージョンおよびリビジョンに対応します。ただし，限定コードは対象外です。例えば，Reliable Messaging 01-02-01 の場合は，"1.2"を返します。

- 引数
  ありません。
- 戻り値
  JMS プロバイダのバージョン。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメタデータの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.3　DeliveryMode インタフェース

DeliveryMode インタフェースは，JMS でサポートしているメッセージの管理形態を表す定数を定義するインタフェースです。

(1) 形式

```
public interface DeliveryMode
{
  public static final int  NON_PERSISTENT;
  public static final int  PERSISTENT;
}
```

(2) フィールド

「(1) 形式」に記載した順序で各フィールドを説明します。

(a) NON_PERSISTENT フィールド

```
public static final int NON_PERSISTENT
```

非永続メッセージを表す定数です。メッセージの送受信で，メッセージを補助記憶装置に記録しないで処理することを意味します。その代わり，PERSISTENT と比べてメッセージ送受信のオーバヘッドが小さくなります。非永続メッセージの管理形態については，「2.5.3(2) キュー作成時の指定値とメッセージのヘッダ」を参照してください。

(b) PERSISTENT フィールド

```
public static final int PERSISTENT
```

永続メッセージを表す定数です。メッセージの送受信で，メッセージを補助記憶装置に記録して処理することを意味します。永続メッセージの管理形態については，「2.5.3(2) キュー作成時の指定値とメッセージのヘッダ」を参照してください。

(3) メソッド

ありません。

7.4.4 Message インタフェース

Message インタフェースはペイロードを持たないメッセージを送受信するために使用できます。Message インタフェースは，すべての JMS メッセージのルートインタフェースです。

JMS メッセージはヘッダとプロパティとペイロード（メッセージ本体）の 3 要素から構成されます。JMS メッセージ，構成要素およびアクセスモードについては，「2.5 メッセージの構成」を参照してください。

(1) ヘッダの設定と取得

Message インタフェースを使用することでヘッダを設定および取得できます。

(2) プロパティの設定，取得および初期化

Message インタフェースを使用することでプロパティを設定，取得および初期化できます。また，プロパティの値は上書きできます。プロパティの初期化には clearProperties()メソッドを使用します。

(3) ペイロードの初期化

clearBody()メソッドを発行してペイロードを初期化します。

(4) メッセージ受信の承認

acknowledge()メソッドを発行して，メッセージの受信を承認します。

(5) 形式

```
public interface Message
{
  public static final int   DEFAULT_DELIVERY_MODE;
  public static final int   DEFAULT_PRIORITY;
  public static final long  DEFAULT_TIME_TO_LIVE;
  public void     acknowledge() throws JMSException;
  public void     clearBody() throws JMSException;
  public void     clearProperties() throws JMSException;
  public boolean  getBooleanProperty(java.lang.String name)
                    throws JMSException;
  public byte     getByteProperty(java.lang.String name)
                    throws JMSException;
  public double   getDoubleProperty(java.lang.String name)
                    throws JMSException;
  public float    getFloatProperty(java.lang.String name)
                    throws JMSException;
  public int      getIntProperty(java.lang.String name)
                    throws JMSException;
  public java.lang.String
                  getJMSCorrelationID() throws JMSException;
  public byte[]   getJMSCorrelationIDAsBytes() throws JMSException;
  public int      getJMSDeliveryMode() throws JMSException;
  public Destination
                  getJMSDestination() throws JMSException;
  public long     getJMSExpiration() throws JMSException;
  public java.lang.String
                  getJMSMessageID() throws JMSException;
  public int      getJMSPriority() throws JMSException;
  public boolean  getJMSRedelivered() throws JMSException;
  public Destination
                  getJMSReplyTo() throws JMSException;
  public long     getJMSTimestamp() throws JMSException;
  public java.lang.String
                  getJMSType() throws JMSException;
  public long     getLongProperty(java.lang.String name)
                    throws JMSException;
  public java.lang.Object
                  getObjectProperty(java.lang.String name)
```

```
                    throws JMSException;
  public java.util.Enumeration
                  getPropertyNames() throws JMSException;
  public short    getShortProperty(java.lang.String name)
                    throws JMSException;
  public java.lang.String
                  getStringProperty(java.lang.String name)
                    throws JMSException;
  public boolean  propertyExists(java.lang.String name)
                    throws JMSException;
  public void     setBooleanProperty(java.lang.String name,
                    boolean value) throws JMSException;
  public void     setByteProperty(java.lang.String name, byte value)
                    throws JMSException;
  public void     setDoubleProperty(java.lang.String name,
                    double value) throws JMSException;
  public void     setFloatProperty(java.lang.String name, float value)
                    throws JMSException;
  public void     setIntProperty(java.lang.String name, int value)
                    throws JMSException;
  public void     setJMSCorrelationID(java.lang.String correlationID)
                    throws JMSException;
  public void     setJMSCorrelationIDAsBytes(byte[] correlationID)
                    throws JMSException;
  public void     setJMSDeliveryMode(int deliveryMode)
                    throws JMSException;
  public void     setJMSDestination(Destination destination)
                    throws JMSException;
  public void     setJMSExpiration(long expiration)
                    throws JMSException;
  public void     setJMSMessageID(java.lang.String id)
                    throws JMSException;
  public void     setJMSPriority(int priority) throws JMSException;
  public void     setJMSRedelivered(boolean redelivered)
                    throws JMSException;
  public void     setJMSReplyTo(Destination replyTo)
                    throws JMSException;
  public void     setJMSTimestamp(long timestamp) throws JMSException;
  public void     setJMSType(java.lang.String type)
                    throws JMSException;
  public void     setLongProperty(java.lang.String name, long value)
                    throws JMSException;
  public void     setObjectProperty(java.lang.String name,
                    java.lang.Object value) throws JMSException;
  public void     setShortProperty(java.lang.String name, short value)
                    throws JMSException;
  public void     setStringProperty(java.lang.String name,
                    java.lang.String value) throws JMSException;
}
```

(6) フィールド

「(5) 形式」に記載した順序で各フィールドを説明します。

(a) DEFAULT_DELIVERY_MODE フィールド

```
public static final int DEFAULT_DELIVERY_MODE
```

メッセージの属性を示す値のデフォルト値を保持しています。デフォルト値は DeliveryMode.PERSISTENT（永続）です。

(b) DEFAULT_PRIORITY フィールド

```
public static final int DEFAULT_PRIORITY
```

メッセージの優先度を示す値のデフォルト値を保持しています。デフォルト値は 4 です。

(c) DEFAULT_TIME_TO_LIVE フィールド

```
public static final long DEFAULT_TIME_TO_LIVE
```

メッセージの保持時間を示す値のデフォルト値を保持しています。デフォルト値は 0 です。

(7) メソッド

「(5) 形式」に記載した順序で各メソッドを説明します。

(a) acknowledge メソッド

```
public void acknowledge() throws JMSException
```

現在のメッセージおよび同じセッションで以前に受信し，かつ以前に承認していないすべてのメッセージに対して，メッセージの受信を承認します。メソッドが正常に終了した場合，受信側は承認したメッセージを何度も受信しないで済みます。

メソッドを発行できる条件を次に示します。条件外でメソッドを発行した場合は JMSException が発生します。

- メッセージを同期受信で取得しました。
- メッセージは，トランザクション設定が false かつメッセージ承認モードが CLIENT_ACKNOWLEDGE の QueueSession オブジェクトによって生成されました。
- メッセージが関連づけられているコネクションおよびセッションがクローズされていません。
- トランザクションがトランザクションマネジャによって制御されていません。

- 引数
  ありません。
- 戻り値
  ありません。

- 例外

例外クラス	説明
IllegalStateException	トランザクションマネジャでのトランザクションが開始しているときに，メソッドが発行されました。
	クローズされたセッションに対して，メソッドが発行されました。
JMSException	トランザクション設定が true のときに，メソッドが発行されました。
	トランザクション設定が false かつメッセージ承認モードが AUTO_ACKNOWLEDGE のときに，メソッドが発行されました。
	トランザクション設定が false かつメッセージ承認モードが DUPS_OK_ACKNOWLEDGE のときに，メソッドが発行されました。
	非同期受信でメッセージを受信したときに，メソッドが発行されました。
	キューブラウザで取得した閲覧用のメッセージに対して，メソッドが発行されました。
	内部エラーのためにメソッドの実行に失敗し，Session.recover()メソッドが正常に終了した際と同じ状態になりました。
	内部エラーのために JMS プロバイダがメッセージの承認に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　clearBody メソッド

```
public void clearBody() throws JMSException
```

BytesMessage，ObjectMessage および TextMessage オブジェクトでメソッドを発行した場合，JMS メッセージのペイロード（メッセージ本体）が初期化されます。初期化されたペイロードの値は，メッセージ生成直後のペイロードの値と同じです。Message オブジェクトでメソッドを発行した場合は無視されます。なお，メソッドを発行してもヘッダとプロパティは初期化されません。

ペイロードが読み取り専用モードの際にメソッドを発行した場合，ペイロードが初期化されるとともにペイロードの書き込みができるようになります。

- 引数

  ありません。
- 戻り値

  ありません。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージ本体の初期化に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(c)　clearProperties メソッド

```
public void clearProperties() throws JMSException
```

メッセージのプロパティを初期化します。初期化されたプロパティの値はメッセージ生成直後のプロパティの値と同じです。なお，メソッドを実行してもヘッダとペイロードは初期化されません。

プロパティが読み取り専用モードの際にメソッドを発行した場合，プロパティが初期化されるとともにプロパティの読み取りと書き込みの両方ができるようになります。

- 引数

  ありません。

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがプロパティの初期化に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(d)　getBooleanProperty メソッド

```
public boolean getBooleanProperty(java.lang.String name)
  throws JMSException
```

boolean 型のプロパティの値を返します。

- 引数

引数名	説明
name	boolean 型のプロパティの名前

- 戻り値

  boolean 型のプロパティの値です。指定された name 引数に対応するプロパティ値がない場合，false を返します。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。

例外クラス	説明
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(e)　getByteProperty メソッド

```
public byte getByteProperty(java.lang.String name) throws JMSException
```

byte 型のプロパティの値を返します。

- 引数

引数名	説明
name	byte 型のプロパティの名前

- 戻り値

  byte 型のプロパティの値です。

- 例外

例外クラス	説明
java.lang.NumberFormatException	指定された name 引数に対応するプロパティ値がありません。
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(f)　getDoubleProperty メソッド

```
public double getDoubleProperty(java.lang.String name)
  throws JMSException
```

double 型のプロパティの値を返します。

- 引数

引数名	説明
name	double 型のプロパティの名前

- 戻り値
  double 型のプロパティの値です。
- 例外

例外クラス	説明
java.lang.NumberFormatException	指定された name 引数に対応するプロパティ値がありません。
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(g)　getFloatProperty メソッド

```
public float getFloatProperty(java.lang.String name)
  throws JMSException
```

float 型のプロパティの値を返します。

- 引数

引数名	説明
name	float 型のプロパティの名前

- 戻り値
  float 型のプロパティの値です。
- 例外

例外クラス	説明
java.lang.NumberFormatException	指定された name 引数に対応するプロパティ値がありません。
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(h)　getIntProperty メソッド

```
public int getIntProperty(java.lang.String name) throws JMSException
```

int 型のプロパティの値を返します。

- 引数

引数名	説明
name	int 型のプロパティの名前

- 戻り値

  int 型のプロパティの値です。

- 例外

例外クラス	説明
java.lang.NumberFormatException	指定された name 引数に対応するプロパティ値がありません。
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(i)　getJMSCorrelationID メソッド

```
public java.lang.String getJMSCorrelationID() throws JMSException
```

メッセージに設定された JMSCorrelationID ヘッダの値を String 型で取得します。JMSCorrelationID ヘッダが設定されていない場合，null を返します。

- 引数

  ありません。

- 戻り値

  メッセージの相関識別子。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダが相関識別子の取得に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(j)　getJMSCorrelationIDAsBytes メソッド

```
public byte[] getJMSCorrelationIDAsBytes() throws JMSException
```

メッセージに設定された JMSCorrelationID ヘッダの値を byte[]で取得します。JMSCorrelationID ヘッダが設定されていない場合，null を返します。

注意

setJMSCorrelationIDAsBytes(byte[])実行時の J2EE サーバで有効となる Java VM のデフォルトエンコーディングと，getJMSCorrelationIDAsBytes()実行時の J2EE サーバで有効となる Java VM のデフォルトエンコーディングは，同じエンコーディングを使用してください。

異なるエンコーディングを使用する場合，setJMSCorrelationIDAsBytes(byte[])の引数の値と，getJMSCorrelationIDAsBytes()で取得される値は異なる場合があります。

キュー間転送を用い，異なるシステム間で JMSCorrelationID を設定したメッセージの送受信を行う場合に注意してください。

- 引数

  ありません。

- 戻り値

  メッセージの相関識別子。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダが相関識別子の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(k)　getJMSDeliveryMode メソッド

```
public int getJMSDeliveryMode() throws JMSException
```

メッセージに設定された JMSDeliveryMode ヘッダの値を返します。JMSDeliveryMode ヘッダが設定されていない場合，DeliveryMode.PERSISTENT を返します。

- 引数

  ありません。

- 戻り値

  メッセージの管理形態として，次に示すどちらかの値を返します。

  - DeliveryMode.NON_PERSISTENT
  - DeliveryMode.PERSISTENT

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの管理形態の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(l)　getJMSDestination メソッド

```
public Destination getJMSDestination() throws JMSException
```

メッセージに設定された JMSDestination ヘッダの値を返します。JMSDestination ヘッダが設定されていない場合，null を返します。

- 引数

  ありません。

- 戻り値

  メッセージに設定された JMSDestination ヘッダの値を返します。JMSDestination ヘッダが設定されていない場合，null を返します。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがあて先の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(m)　getJMSExpiration メソッド

```
public long getJMSExpiration() throws JMSException
```

メッセージに設定された JMSExpiration ヘッダの値を返します。JMSExpiration ヘッダが設定されていない場合，0 を返します。

- 引数

  ありません。

- 戻り値

  メッセージの有効期間値。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージの有効期間の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(n) getJMSMessageID メソッド

```
public java.lang.String getJMSMessageID() throws JMSException
```

メッセージに設定された JMSMessageID ヘッダの値を返します。JMSMessageID ヘッダが設定されていない場合，null を返します。

- 引数
  ありません。
- 戻り値
  メッセージ識別子。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージ識別子の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(o) getJMSPriority メソッド

```
public int getJMSPriority() throws JMSException
```

メッセージに設定された JMSPriority ヘッダの値を返します。JMSPriority ヘッダが設定されていない場合，javax.jms.Messge.DEFAULT_PRIORITY を返します。

- 引数
  ありません。
- 戻り値
  メッセージの優先度。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージ優先度の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(p) getJMSRedelivered メソッド

```
public boolean getJMSRedelivered() throws JMSException
```

メッセージに設定された JMSRedelivered ヘッダの値を返します。JMSRedelivered ヘッダが設定されていない場合，false を返します。

- 引数
  ありません。
- 戻り値
  メッセージが再配送中の場合は true です。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダが再配送状態の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(q)　getJMSReplyTo メソッド

```
public Destination getJMSReplyTo() throws JMSException
```

メッセージに設定された JMSReplyTo ヘッダの値を返します。JMSReplyTo ヘッダが設定されていない場合，null を返します。

- 引数
  ありません。
- 戻り値
  メッセージへの応答の送信先である Destination オブジェクト。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダが JMSReplyTo ヘッダの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(r)　getJMSTimestamp メソッド

```
public long getJMSTimestamp() throws JMSException
```

メッセージに設定された JMSTimestamp ヘッダの値を返します。JMSTimestamp ヘッダが設定されていない場合，0 を返します。

- 引数
  ありません。

- 戻り値
  メッセージのタイムスタンプ。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがタイムスタンプの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(s)　getJMSType メソッド

```
public java.lang.String getJMSType() throws JMSException
```

メッセージに設定された JMSType ヘッダの値を返します。JMSType ヘッダが設定されていない場合，null を返します。

- 引数
  ありません。
- 戻り値
  メッセージ型。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージ型の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(t)　getLongProperty メソッド

```
public long getLongProperty(java.lang.String name) throws JMSException
```

long 型のプロパティの値を返します。

- 引数

引数名	説明
name	long 型のプロパティの名前

- 戻り値
  指定された名前の long 型のプロパティ値です。

- 例外

例外クラス	説明
java.lang.NumberFormatException	指定された name 引数に対応するプロパティ値がありません。
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(u)　getObjectProperty メソッド

```
public java.lang.Object getObjectProperty(java.lang.String name)
  throws JMSException
```

Object 型のプロパティの値を返します。指定された名前に対応するプロパティ値がない場合，null を返します。

メソッドは Boolean，Byte，Short，Integer，Long，Float，Double および String クラスの値だけを返します。

- 引数

引数名	説明
name	Java オブジェクトプロパティの名前

- 戻り値

  指定された名前の Java オブジェクトプロパティ値のオブジェクト化形式。例えば，プロパティが int として設定された場合は Integer を返します。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(v)　getPropertyNames メソッド

```
public java.util.Enumeration getPropertyNames() throws JMSException
```

すべてのプロパティの名前を返します。プロパティがない場合，null を返します。

メソッドで取得するプロパティ名の順序は保証されません。

- 引数
  ありません。
- 戻り値
  プロパティ値のすべての名前についての列挙。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(w)　getShortProperty メソッド

```
public short getShortProperty(java.lang.String name)
  throws JMSException
```

short 型のプロパティの値を返します。

- 引数

引数名	説明
name	short 型のプロパティの名前

- 戻り値
  指定された名前の short 型のプロパティ値です。
- 例外

例外クラス	説明
java.lang.NumberFormatException	指定された name 引数に対応するプロパティ値がありません。
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(x) getStringProperty メソッド

```
public java.lang.String getStringProperty(java.lang.String name)
  throws JMSException
```

String 型のプロパティの値を返します。指定された名前に対応するプロパティ値がない場合，null を返します。

- 引数

引数名	説明
name	String 型のプロパティの名前

- 戻り値

  指定された名前の String 型のプロパティ値。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageFormatException	メッセージ形式不正が原因でプロパティの型変換に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(y) propertyExists メソッド

```
public boolean propertyExists(java.lang.String name)
  throws JMSException
```

指定されたプロパティ名に対応するプロパティがあれば true を返し，なければ false を返します。

- 引数

引数名	説明
name	存在を判定するプロパティの名前

- 戻り値

  プロパティがある場合は true，ない場合は false を返します。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。

例外クラス	説明
	内部エラーのために JMS プロバイダがプロパティの存在確認に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(z)　setBooleanProperty メソッド

```
public void setBooleanProperty(java.lang.String name, boolean value)
  throws JMSException
```

boolean 型のプロパティの値を設定します。

プロパティ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
name	boolean 型のプロパティの名前（64 文字以下）
value	設定する boolean 型のプロパティ値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(aa)　setByteProperty メソッド

```
public void setByteProperty(java.lang.String name, byte value)
  throws JMSException
```

byte 型のプロパティの値を設定します。

プロパティ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
name	byte 型のプロパティの名前（64 文字以下）
value	設定する byte 型のプロパティ値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(ab)　setDoubleProperty メソッド

```
public void setDoubleProperty(java.lang.String name, double value)
  throws JMSException
```

double 型のプロパティの値を設定します。

プロパティ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
name	double 型のプロパティの名前（64 文字以下）
value	設定する double 型のプロパティ値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。

例外クラス	説明
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(ac)　setFloatProperty メソッド

```
public void setFloatProperty(java.lang.String name, float value)
  throws JMSException
```

float 型のプロパティの値を設定します。

プロパティ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
name	float 型のプロパティの名前（64 文字以下）
value	設定する float 型のプロパティ値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(ad)　setIntProperty メソッド

```
public void setIntProperty(java.lang.String name, int value)
  throws JMSException
```

int 型のプロパティの値を設定します。

プロパティ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
name	int 型のプロパティの名前（64 文字以下）
value	設定する int 型のプロパティ値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティ値の取得に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(ae)　setJMSCorrelationID メソッド

```
public void setJMSCorrelationID(java.lang.String correlationID)
  throws JMSException
```

メッセージの JMSCorrelationID ヘッダを String 型で設定します。

ヘッダ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
correlationID	相関識別子（512 文字以下）

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	correlationID 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダが相関識別子の設定に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(af)　setJMSCorrelationIDAsBytes メソッド

```
public void setJMSCorrelationIDAsBytes(byte[] correlationID)
  throws JMSException
```

メッセージの JMSCorrelationID ヘッダを byte[]で設定します。

ヘッダ値は上書きできます。引数の文字数制限は半角全角を問いません。

注意

- 引数には，J2EE サーバで有効となる Java VM のデフォルトエンコーディングで正しく String に変換ができる byte[]を指定してください。正しく変換できない引数を指定した場合，JMSCorrelationID に不正なデータが設定されます。
- setJMSCorrelationIDAsBytes(byte[])実行時の J2EE サーバで有効となる Java VM のデフォルトエンコーディングと，getJMSCorrelationIDAsBytes()実行時の J2EE サーバで有効となる Java VM のデフォルトエンコーディングは，同じエンコーディングを使用してください。

  異なるエンコーディングを使用する場合，setJMSCorrelationIDAsBytes(byte[])の引数の値と，getJMSCorrelationIDAsBytes()で取得される値は異なる場合があります。

  キュー間転送を用い，異なるシステム間で JMSCorrelationID を設定したメッセージの送受信を行う場合に注意してください。

- 引数

引数名	説明
correlationID	相関識別子（バイト配列）（512 文字以下）

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	correlationID 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダが相関識別子の設定に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(ag)　setJMSDeliveryMode メソッド

```
public void setJMSDeliveryMode(int deliveryMode) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
deliveryMode	メッセージの管理形態として，次に示すどちらかの値を指定できます。 • DeliveryMode.NON_PERSISTENT：非永続 • DeliveryMode.PERSISTENT：永続

- 戻り値

  ありません。

- 例外

  ありません。

(ah) setJMSDestination メソッド

```
public void setJMSDestination(Destination destination)
  throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
destination	メッセージの送信先

- 戻り値

  ありません。

- 例外

  ありません。

(ai) setJMSExpiration メソッド

```
public void setJMSExpiration(long expiration) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
expiration	メッセージの有効期間

- 戻り値
  ありません。
- 例外
  ありません。

(aj) setJMSMessageID メソッド

```
public void setJMSMessageID(java.lang.String id) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
id	メッセージ識別子

- 戻り値
  ありません。
- 例外
  ありません。

(ak) setJMSPriority メソッド

```
public void setJMSPriority(int priority) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
priority	メッセージの優先度

- 戻り値
  ありません。
- 例外
  ありません。

(al)　setJMSRedelivered メソッド

```
public void setJMSRedelivered(boolean redelivered) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
redelivered	メッセージが再配送中であるかどうかを示す値です。値が true の場合，再配送中であることを示します。

- 戻り値

  ありません。

- 例外

  ありません。

(am)　setJMSReplyTo メソッド

```
public void setJMSReplyTo(Destination replyTo) throws JMSException
```

メッセージの JMSReplyTo ヘッダを Destination 型で設定します。

ヘッダ値は上書きできます。

- 引数

引数名	説明
replyTo	メッセージへの応答の送信先

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダが JMSReplyTo ヘッダの設定に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(an)　setJMSTimestamp メソッド

```
public void setJMSTimestamp(long timestamp) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
timestamp	メッセージのタイムスタンプ

- 戻り値
  ありません。
- 例外
  ありません。

(ao) setJMSType メソッド

```
public void setJMSType(java.lang.String type) throws JMSException
```

メッセージの JMSType ヘッダを String で設定します。

ヘッダ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
type	メッセージ型（512 文字以下）

- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	type 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがメッセージ型の設定に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(ap) setLongProperty メソッド

```
public void setLongProperty(java.lang.String name, long value)
  throws JMSException
```

long 型のプロパティの値を設定します。

プロパティ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
name	long 型のプロパティの名前（64 文字以下）
value	設定する long 型のプロパティ値

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティ値の設定に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(aq)　setObjectProperty メソッド

```
public void setObjectProperty(java.lang.String name,
  java.lang.Object value) throws JMSException
```

Object 型のプロパティの値を設定します。

- メソッドは Boolean，Byte，Short，Integer，Long，Float，Double および String クラスの値を受け付けます。そのほかのクラスを使用すると，JMSException が発生します。
- プロパティ値には null を指定できます。
- 引数の文字数制限は半角全角を問いません。
- プロパティ値は上書きできます。

- 引数

引数名	説明
name	Java オブジェクトプロパティの名前（64 文字以下）
value	設定する Java オブジェクトプロパティ値（String 型のプロパティ値の場合は 512 文字以下）

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	value 引数に String 型のクラスを指定し，その文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティ値の設定に失敗しました。
MessageFormatException	引数 value の形式不正が原因でプロパティの型変換に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(ar)　setShortProperty メソッド

```
public void setShortProperty(java.lang.String name, short value)
  throws JMSException
```

short 型のプロパティの値を設定します。

プロパティ値は上書きできます。引数の文字数制限は半角全角を問いません。

- 引数

引数名	説明
name	short 型のプロパティの名前（64 文字以下）
value	設定する short 型のプロパティ値

- 戻り値

ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティの設定に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(as)　setStringProperty メソッド

```
public void setStringProperty(java.lang.String name,
  java.lang.String value) throws JMSException
```

String 型のプロパティの値を設定します。

- プロパティ値には null を指定できます。
- 引数の文字数制限は半角全角を問いません。
- プロパティ値は上書きできます。

- 引数

引数名	説明
name	String 型のプロパティの名前（64 文字以下）
value	設定する String 型のプロパティ値（512 文字以下）

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	name 引数に null が指定されました。
	name 引数の文字数が制限値を超えました。
	value 引数の文字数が制限値を超えました。
	内部エラーのために JMS プロバイダがプロパティ値の設定に失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.5　ObjectMessage インタフェース

ObjectMessage インタフェースはシリアライズできる Java オブジェクトを含むメッセージを送受信するために使用します。ObjectMessage インタフェースは Message インタフェースを継承しているため，Message インタフェースの機能を持っています。

(1)　ペイロードの設定と取得

ObjectMessage インタフェースを使用することでペイロードを設定および取得できます。このペイロードはシリアライズできる Java オブジェクトです。JMS メッセージの各要素については，「2.5.1　JMS メッセージの構成」を参照してください。

(2)　形式

```
public interface ObjectMessage extends Message
{
  public java.io.Serializable
               getObject() throws JMSException;
  public void  setObject(java.io.Serializable object)
                 throws JMSException;
}
```

(3)　フィールド

ありません。

(4)　メソッド

「(2)　形式」に記載した順序で各メソッドを説明します。

(a)　getObject メソッド

```
public java.io.Serializable getObject() throws JMSException
```

直列化できるメッセージ本体データを返します。メッセージ本体が設定されていない場合はデフォルト値である null を返します。

- 引数
  ありません。
- 戻り値
  メッセージのデータを含む直列化できるオブジェクト。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがオブジェクトの取得に失敗しました。
MessageFormatException	ペイロードの型が無効なため直列化されたオブジェクトの復元が失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b) setObject メソッド

```
public void setObject(java.io.Serializable object) throws JMSException
```

メッセージのデータを含む直列化できるオブジェクトを設定します。

setObject メソッドの内部では引数オブジェクトのコピーを生成し，そのコピーを ObjectMessage オブジェクトのペイロードに設定します。そのため，そのあと引数オブジェクトを変更しても ObjectMessage オブジェクトのペイロードに影響しなくなります。

注意

ユーザが定義したクラスのオブジェクトを ObjectMessage オブジェクトのペイロードに設定する場合は，ユーザが定義したクラスまたはクラスが含まれる JAR ファイルのクラスパスを，コンテナ拡張ライブラリとして Application Server の J2EE サーバ用オプション定義ファイルに指定してください。J2EE サーバ用オプション定義ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

クラスパスが指定されていない状態で，ObjectMessage オブジェクトのペイロードにユーザが定義したクラスのオブジェクトを設定しようとすると，JMSException が発生します。

- 引数

引数名	説明
object	メッセージのデータ

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	object 引数に null が指定されました。
	内部エラーのために JMS プロバイダがオブジェクトの設定に失敗しました。
MessageFormatException	引数の型が無効なためオブジェクトの直列化が失敗しました。
MessageNotWriteableException	読み取り専用モードのプロパティに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.6 Queue（Destination）インタフェース

Queue（Destination）インタフェースは，送受信するメッセージのあて先を指定するための JMS 管理オブジェクトです。Queue（Destination）インタフェースは，メッセージのあて先となるキューをカプセル化したものです。

(1) キュー名の取得

キュー名の取得を行います。

(2) 形式

```
public interface Destination
{ }

public interface Queue extends Destination
{
  public java.lang.String
               getQueueName() throws JMSException;
  public java.lang.String
               toString();
}
```

(3) フィールド

ありません。

(4) メソッド

「(2) 形式」に記載した順序で各メソッドを説明します。

(a) getQueueName メソッド

```
public java.lang.String getQueueName() throws JMSException
```

キュー名を返します。名前に依存した JMS クライアントにはポータビリティが保証されません。

- 引数
  ありません。
- 戻り値
  キュー名。
- 例外

例外クラス	説明
JMSException	内部エラーのために Queue オブジェクトの JMS プロバイダ実装がキュー名を返すことに失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　toString メソッド

```
public java.lang.String toString()
```

キュー名を特定するための文字列を返します。java.lang.Object の toString()メソッドをオーバライドします。

- 引数
  ありません。
- 戻り値
  JMS プロバイダがキューを特定する識別用の値です。
- 例外
  ありません。

7.4.7　QueueBrowser インタフェース

QueueBrowser インタフェースは，JMS クライアントがキューからメッセージを削除しないで参照するために使用します。

(1)　現在のキューにあるメッセージ一覧の取得

現在のキューにあるメッセージの一覧を取得します。メッセージセレクタを設定して QueueBrowser オブジェクトが生成されている場合は，メッセージセレクタに応じたメッセージだけを参照できます。

(2)　メッセージセレクタの取得

QueueBrowser オブジェクトを生成する際に設定されたメッセージセレクタを取得します。メッセージセレクタについては，「2.6.2　メッセージセレクタ」を参照してください。

(3)　キューブラウザのクローズ

QueueBrowser オブジェクトをクローズできます。クローズによって，QueueBrowser オブジェクトに関連づけられたすべてのリソースを解放します。

QueueBrowser インタフェースのメソッドは，複数スレッドからの同時発行を JMS で禁止されています。ただし，close()メソッドだけは複数スレッドから同時発行してパラレルに実行できます。

(4)　キューの取得

キューブラウザを生成する際に設定されたキューを取得します。

(5)　形式

```
public interface QueueBrowser
{
  public void     close() throws JMSException;
  public java.util.Enumeration
                  getEnumeration() throws JMSException;
  public java.lang.String
                  getMessageSelector() throws JMSException;
  public Queue    getQueue() throws JMSException;
}
```

(6)　フィールド

ありません。

(7)　メソッド

「(5)　形式」に記載した順序で各メソッドを説明します。

(a)　close メソッド

```
public void close() throws JMSException
```

キューブラウザをクローズします。キューブラウザには幾つかのリソースが割り当てられている場合があるため，リソースが不要になった場合はキューブラウザをクローズしてください。

- クローズされているキューブラウザをクローズしても例外は発生しません。
- このメソッドは QueueBrowser インタフェースでは唯一，複数スレッドから同時に発行できます。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがキューブラウザをクローズする処理に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b) getEnumeration メソッド

```
public java.util.Enumeration getEnumeration() throws JMSException
```

現在のキューにあるメッセージの列挙を取得します。

- 一覧を取得したあとにメッセージを閲覧する場合，メッセージがすでに削除または消失していることがあります。
- キューブラウザが close 状態の場合，Enumeration.hasMoreElements()メソッドの戻り値は false です。
- キューブラウザが close 状態の場合，Enumeration.nextElement()メソッドを発行すると NoSuchElementException が発生します。
- 送信用共用キューおよび転送キューに getEnumeration()メソッドを発行した場合は空のリストを返します。

- 引数

  ありません。

- 戻り値

  表示用メッセージの一覧。

- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはキューブラウザをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが QueueBrowser オブジェクトのメッセージ一覧の取得に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9 障害コードの詳細」を参照してください。

(c) getMessageSelector メソッド

```
public java.lang.String getMessageSelector() throws JMSException
```

キューブラウザのメッセージセレクタ構文を返します。

- 引数

  ありません。

- 戻り値

  キューブラウザのメッセージセレクタ構文。

- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはキューブラウザをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが QueueBrowser オブジェクトのメッセージセレクタの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(d)　getQueue メソッド

```
public Queue getQueue() throws JMSException
```

キューブラウザに関連づけられたキューを返します。

- 引数
  ありません。
- 戻り値
  キューブラウザの Queue オブジェクト。
- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはキューブラウザをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが QueueBrowser オブジェクトに関連づけられたキューの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.8　QueueConnection（Connection）インタフェース

QueueConnection インタフェースは，アプリケーションが Reliable Messaging にアクセスするために使用するアプリケーションレベルのハンドルを提供します。

QueueConnection オブジェクトを取得するには，
QueueConnectionFactory.createQueueConnection()メソッドを使用します。

(1)　キューセッションの生成

createQueueSession()メソッドを発行することで QueueSession オブジェクトを生成します。また，createQueueSession()メソッドの transacted 引数によってトランザクション設定を指定し，acknowledgeMode 引数を指定することでメッセージ承認モードを指定できます。

各引数を指定したときのトランザクションとメッセージ送受信の関係については，「2.6.6　トランザクション制御」を参照してください。

(2)　配送の開始と停止

start()および stop()メソッドを発行することで配送の開始と停止を制御できます。QueueConnection オブジェクトは生成時に配送停止状態に設定されています。

(3)　コネクションメタデータの取得

Reliable Messaging の製品名やバージョン，JMS API のバージョンを保持する ConnectionMetaData オブジェクトを取得できます。

(4)　キューコネクションのクローズ

QueueConnection オブジェクトをクローズできます。クローズによって，QueueConnection オブジェクトに関連づけられたすべてのセッションをクローズし，QueueConnection オブジェクトに関連づけられたすべてのリソースを解放します。

(5)　形式

```
public interface Connection
{
  public void     close() throws JMSException;
  public java.lang.String
                  getClientID() throws JMSException;
  public ExceptionListener
                  getExceptionListener() throws JMSException;
  public ConnectionMetaData
                  getMetaData() throws JMSException;
  public void     setClientID(java.lang.String clientID)
                    throws JMSException;
  public void     setExceptionListener(ExceptionListener listener)
                    throws JMSException;
  public void     start() throws JMSException;
  public void     stop() throws JMSException;
}

public interface QueueConnection extends javax.jms.Connection
{
  public ConnectionConsumer
                  createConnectionConsumer(Queue queue,
                    java.lang.String messageSelector,
                    ServerSessionPool sessionPool, int maxMessages)
                    throws JMSException;
  public QueueSession
                  createQueueSession(boolean transacted,
                    int acknowledgeMode) throws JMSException;
}
```

(6)　フィールド

ありません。

(7)　メソッド

「(5)　形式」に記載した順序で各メソッドを説明します。

(a)　close メソッド

```
public void close() throws JMSException
```

コネクションをクローズします。メソッドを発行すると，コネクションに関連づけられているすべてのセッションがクローズされます。コネクションに代わって JavaVM 外部リソースが割り当てられているので，これらのリソースが不要になった場合はコネクションをクローズしてください。メソッドを発行すると次に示すとおり動作します。

- メッセージ処理が適切な方法で終了されるまで，メソッドは復帰しません。
- コネクションに関連づけられているすべてのセッションのトランザクションはロールバックされます。
- コネクションに関連づけられているすべてのセッションがクローズします。
- コネクションに関連づけられているすべてのセッションでのすべての未承認メッセージは，QueueSession.recover()メソッドが正常終了した場合と同じ状態になります。
- クローズされているコネクションをクローズしても例外は発生しません。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがコネクションのクローズに失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　getClientID メソッド

```
public java.lang.String getClientID() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，null を返します。

- 引数
  ありません。
- 戻り値
  一意の JMS クライアント識別子。
- 例外
  ありません。

(c) getExceptionListener メソッド

```
public ExceptionListener getExceptionListener() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，null を返します。

- 引数
  ありません。
- 戻り値
  例外リスナ。
- 例外
  ありません。

(d) getMetaData メソッド

```
public ConnectionMetaData getMetaData() throws JMSException
```

コネクションのメタデータを返します。

- 引数
  ありません。
- 戻り値
  コネクションメタデータ。
- 例外

例外クラス	説明
JMSException	コネクションがクローズしている状態で，メソッドを発行しました。
	内部エラーのために JMS プロバイダがコネクション用のメタデータ取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(e)　setClientID メソッド

```
public void setClientID(java.lang.String clientID) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
java.lang.String	一意の JMS クライアント識別子

- 戻り値

  ありません。

- 例外

  ありません。

(f)　setExceptionListener メソッド

```
public void setExceptionListener(ExceptionListener listener)
  throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
listener	例外リスナ

- 戻り値

  ありません。

- 例外

  ありません。

(g)　start メソッド

```
public void start() throws JMSException
```

コネクションを開始することによって，コネクションに関連づけられているすべてのセッションで，メッセージの配送を開始（または再度開始）します。stop()メソッド発行後や生成後のコネクションは配送が禁止された状態であり，start()メソッドを発行するまでコネクションでは配送が開始されません。開始しているコネクションに対する start()メソッドの発行は無視されます。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	コネクションがクローズしている状態で，メソッドを発行しました。
	内部エラーのために JMS プロバイダがメッセージ配送の開始に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(h)　stop メソッド

```
public void stop() throws JMSException
```

コネクションでのメッセージの配送を一時的に停止します。コネクションを停止すると，コネクションに関連づけられているすべてのセッションで配送が禁止されます。停止しているコネクションに対する stop()メソッドの発行は無視されます。

メッセージの配送が停止するまで，stop()メソッドの発行は復帰しません。これは，コネクションが再度開始されるまで，receive()メソッドの復帰を待つすべてのスレッドがメッセージを保持して復帰しないことを意味します。停止したコネクションの QueueReceiver.receive()メソッドのタイマは進行し続けるため，コネクション停止中に receive()メソッドの処理がタイムアウトすることがあります。

なお，コネクションが停止している間もメッセージは送信できます。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	コネクションがクローズしている状態で，メソッドを発行しました。

例外クラス	説明
	内部エラーのために JMS プロバイダがメッセージ配送の停止に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(i)　createConnectionConsumer メソッド

```
public ConnectionConsumer createConnectionConsumer(Queue queue,
  java.lang.String messageSelector, ServerSessionPool sessionPool,
  int maxMessages) throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，JMSException が発生します。

- 引数

引数名	説明
queue	アクセス対象のキュー
messageSelector	メッセージセレクタ構文に一致するプロパティを保持するメッセージだけが配送されます。
sessionPool	接続コンシューマに関連づけるサーバセッションプール
maxMessages	一度にサーバセッションに割り当てできる最大メッセージ数

- 戻り値

  コネクションコンシューマ。

- 例外

例外クラス	説明
JMSException	メソッドが未サポートであることを示します。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(j)　createQueueSession メソッド

```
public QueueSession createQueueSession(boolean transacted,
  int acknowledgeMode) throws JMSException
```

QueueSession オブジェクトを生成します。

- 引数

引数名	説明
transacted	トランザクション設定を指定する引数です。セッションがトランザクション処理されるかどうかを示します。有効な値を次に示します。 • true：トランザクション管理されます。 • false：トランザクション管理されません。
acknowledgeMode	メッセージ承認モードを指定する引数です。JMS クライアントが受信するメッセージをどのように承認するかを示します。transacted 引数が true の場合は無視されます。有効な値を次に示します。 • Session.AUTO_ACKNOWLEDGE • Session.DUPS_OK_ACKNOWLEDGE • Session.CLIENT_ACKNOWLEDGE

- 戻り値

  新しく作成されたキューセッション。

- 例外

例外クラス	説明
JMSException	コネクションがクローズしている状態で，メソッドを発行しました。
	内部エラーまたはトランザクションおよびメッセージ承認モードのサポートが不足しているために，QueueConnection オブジェクトがセッションの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.9　QueueConnectionFactory（ConnectionFactory）インタフェース

QueueConnectionFactory（ConnectionFactory）インタフェースは，QueueConnection オブジェクトを生成するためのファクトリです。JNDI に登録されている QueueConnectionFactory の JNDI 登録名をアプリケーションが lookup することによって QueueConnectionFactory オブジェクトを取得します。

(1)　キューコネクションの生成

createQueueConnection()メソッドを発行することによって QueueConnection オブジェクトを生成します。

(2)　形式

```
public interface ConnectionFactory
{ }

public interface QueueConnectionFactory
  extends ConnectionFactory
{
```

```
  public QueueConnection
                  createQueueConnection() throws JMSException;
  public QueueConnection
                  createQueueConnection(java.lang.String userName,
                    java.lang.String password) throws JMSException;
}
```

(3)　フィールド

ありません。

(4)　メソッド

「(2)　形式」に記載した順序で各メソッドを説明します。

(a)　createQueueConnection メソッド

```
public QueueConnection createQueueConnection()
  throws JMSException
```

キューコネクションを作成します。QueueConnection.start()メソッドが明示的に発行されるまで，メッセージは配送されません。

- 引数

  ありません。

- 戻り値

  新規に作成されたキューコネクション。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがキューコネクションの作成に失敗しました。
JMSSecurityException	無効なユーザ名またはパスワードが指定されたために，JMS クライアント認証が失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　createQueueConnection メソッド

```
public QueueConnection createQueueConnection(
  java.lang.String userName, java.lang.String password)
  throws JMSException
```

デフォルトのユーザ識別情報を使用してキューコネクションを作成します。QueueConnection.start()メソッドが明示的に発行されるまで，メッセージは配送されません。

- 引数

引数名	説明
userName	呼び出し側のユーザ名
password	呼び出し側のパスワード

- 戻り値

  新規に作成されたキューコネクション。

- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがキューコネクションの作成に失敗しました。
JMSSecurityException	無効なユーザ名またはパスワードが指定されたために，JMS クライアント認証が失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.10　QueueReceiver（MessageConsumer）インタフェース

QueueReceiver（MessageConsumer）インタフェースは，JMS クライアントがキューからメッセージを受信するために使用します。

(1)　メッセージの受信

メッセージを受信できます。メッセージセレクタを設定して QueueReceiver オブジェクトが生成されている場合は，メッセージセレクタに応じたメッセージだけを受信できます。

また，メッセージの受信処理には，次に示す種類があります。

- 通常のメッセージ受信処理
- 次メッセージがすぐに利用できるならそのメッセージを受信する処理
- タイムアウト間隔内に到着する次メッセージを受信する処理

(2)　メッセージセレクタの取得

QueueReceiver オブジェクトを生成する際に設定されたメッセージセレクタを取得します。メッセージセレクタについては，「2.6.2　メッセージセレクタ」を参照してください。

(3) キューレシーバのクローズ

QueueReceiver オブジェクトをクローズできます。クローズによって，QueueReceiver オブジェクトに関連づけられたすべてのリソースを解放します。QueueReceiver（MessageConsumer）インタフェースのメソッドは，複数スレッドからの同時発行を JMS で禁止されています。ただし，close()メソッドだけは複数スレッドから同時発行してパラレルに実行できます。

(4) キューの取得

QueueReceiver オブジェクトを生成する際に設定されたキューを取得します。

(5) 形式

```
public interface MessageConsumer
{
  public void     close() throws JMSException;
  public MessageListener
                  getMessageListener() throws JMSException;
  public java.lang.String
                  getMessageSelector() throws JMSException;
  public Message  receive() throws JMSException;
  public Message  receive(long timeout) throws JMSException;
  public Message  receiveNoWait() throws JMSException;
  public void     setMessageListener(MessageListener listener)
                    throws JMSException;
}

public interface QueueReceiver extends MessageConsumer
{
  public Queue    getQueue() throws JMSException;
}
```

(6) フィールド

ありません。

(7) メソッド

「(5) 形式」に記載した順序で各メソッドを説明します。

(a) close メソッド

```
public void close() throws JMSException
```

メッセージコンシューマをクローズします。メッセージコンシューマには幾つかのリソースが割り当てられている場合があるため，リソースが不要になった場合はメッセージコンシューマをクローズしてください。

- クローズされているメッセージコンシューマをクローズしても例外は発生しません。

- このメソッドは MessageConsumer インタフェースでは唯一，複数スレッドから同時に発行できます。
- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージコンシューマをクローズする処理に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　getMessageListener メソッド

```
public MessageListener getMessageListener() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，null を返します。

- 引数
  ありません。
- 戻り値
  メッセージコンシューマのリスナまたは null（リスナが設定されていない場合）。
- 例外
  ありません。

(c)　getMessageSelector メソッド

```
public java.lang.String getMessageSelector() throws JMSException
```

メッセージコンシューマのメッセージセレクタ構文を返します。

- 引数
  ありません。
- 戻り値
  メッセージコンシューマのメッセージセレクタ。
- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージコンシューマをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダがメッセージセレクタの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(d)　receive メソッド

```
public Message receive() throws JMSException
```

メッセージコンシューマ用に生成された次のメッセージを受信します。

メソッドの発行時に，メッセージコンシューマの生成元であるコネクションが stop 状態の場合は null を返します。メソッドの実行中にコネクションが stop 状態になった場合は受信待ち状態でメソッドの処理を続行します。

- 引数
  ありません。
- 戻り値
  メッセージコンシューマ用に生成される次のメッセージ，または null（メッセージコンシューマが同時にクローズされる場合）。
- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージコンシューマをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが次のメッセージの受信に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(e)　receive メソッド

```
public Message receive(long timeout) throws JMSException
```

指定されたタイムアウト間隔内に到着する次のメッセージを受信します。メッセージが到着するか，タイムアウトになるか，またはメッセージコンシューマがクローズされる場合，このメソッドは終了します。timeout 引数に 0 を指定するとタイムアウトになりません。

メッセージの発行時にメッセージコンシューマの生成元のコネクションが stop 状態の場合は null を返します。メソッドの実行中にコネクションが stop 状態になった場合は受信を一時的に中断した状態でタイムアウトするまでメソッドの処理を続行します。

なお，receive()発行時に該当キュー内にメッセージがなかった場合は，その後 1 秒ごとのポーリングによってキュー内のメッセージの有無をチェックします。

- 引数

引数名	説明
timeout	タイムアウト値（ミリ秒）

- 戻り値

  メッセージコンシューマ用に生成される次のメッセージ，または null（タイムアウトになるか，メッセージコンシューマが同時にクローズされる場合）。

- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージコンシューマをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが次のメッセージの受信に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(f)　receiveNoWait メソッド

```
public Message receiveNoWait() throws JMSException
```

次のメッセージがすぐに利用できる場合，そのメッセージを受信します。

メッセージの発行時にメッセージコンシューマの生成元のコネクションが stop 状態の場合は null を返します。

- 引数

  ありません。

- 戻り値

  メッセージコンシューマ用に生成された次のメッセージ，または null（受信できる次のメッセージがない場合）。

- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージコンシューマをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが次のメッセージの受信に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(g)　setMessageListener メソッド

```
public void setMessageListener(MessageListener listener)
  throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
listener	メッセージ配送先のリスナ

- 戻り値

  ありません。

- 例外

  ありません。

(h)　getQueue メソッド

```
public Queue getQueue() throws JMSException
```

キューレシーバに関連した Queue オブジェクトを返します。

- 引数

  ありません。

- 戻り値

  レシーバの Queue オブジェクト。

- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージコンシューマをクローズしている状態でメソッドを発行しました。

例外クラス	説明
	内部エラーのために JMS プロバイダが QueueReceiver オブジェクトに関連づけられたキューの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.11　QueueSender（MessageProducer）インタフェース

QueueSender（MessageProducer）インタフェースは，JMS クライアントがあて先にメッセージを送信するために使用します。

(1)　メッセージの送信

指定したあて先にメッセージを送信できます。send()メソッドの実行中に，JMS クライアント内の他スレッドがメッセージを変更してはいけません。メッセージが変更された場合の send()メソッドの結果は保証されません。JMSPriority ヘッダの設定は，メッセージの送信処理の一部として実行されます。

(2)　キューセンダのクローズ

QueueSender オブジェクトをクローズできます。クローズによって，QueueSender オブジェクトに関連づけられたすべてのリソースを解放します。QueueSender（MessageProducer）インタフェースのメソッドは，複数スレッドからの同時呼び出しを JMS で禁止されています。ただし，close()メソッドだけは，複数スレッドからの同時発行に対してパラレルに実行できます。

(3)　形式

```
public interface MessageProducer
{
  public void     close() throws JMSException;
  public int      getDeliveryMode() throws JMSException;
  public boolean  getDisableMessageID() throws JMSException;
  public boolean  getDisableMessageTimestamp() throws JMSException;
  public int      getPriority() throws JMSException;
  public long     getTimeToLive() throws JMSException;
  public void     setDeliveryMode(int deliveryMode)
                    throws JMSException;
  public void     setDisableMessageID(boolean value)
                    throws JMSException;
  public void     setDisableMessageTimestamp(boolean value)
                    throws JMSException;
  public void     setPriority(int defaultPriority)
                    throws JMSException;
  public void     setTimeToLive(long timeToLive) throws JMSException;
}

public interface QueueSender extends MessageProducer
{
```

```
  public Queue    getQueue() throws JMSException;
  public void     send(Message message) throws JMSException;
  public void     send(Message message, int deliveryMode, int priority,
                    long timeToLive) throws JMSException;
  public void     send(Queue queue, Message message)
                    throws JMSException;
  public void     send(Queue queue, Message message, int deliveryMode,
                   int priority, long timeToLive) throws JMSException;
}
```

(4)　フィールド

ありません。

(5)　メソッド

「(3)　形式」に記載した順序で各メソッドを説明します。

(a)　close メソッド

```
public void close() throws JMSException
```

メッセージプロデューサをクローズします。メッセージプロデューサには幾つかのリソースが割り当てられている場合があるため，リソースが不要になった場合はメッセージプロデューサをクローズしてください。

- クローズされているメッセージプロデューサをクローズしても例外は発生しません。
- このメソッドは MessageProducer インタフェースでは唯一，複数スレッドから同時に発行できます。

- 引数

  ありません。
- 戻り値

  ありません。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがメッセージプロデューサをクローズする処理に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　getDeliveryMode メソッド

```
public int getDeliveryMode() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，0 を返します。

- 引数
  ありません。
- 戻り値
  メッセージプロデューサ用のメッセージ管理形態。次に示すどちらかの値を返します。
  - DeliveryMode.NON_PERSISTENT：非永続
  - DeliveryMode.PERSISTENT：永続
- 例外
  ありません。

(c) getDisableMessageID メソッド

```
public boolean getDisableMessageID() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，false を返します。

- 引数
  ありません。
- 戻り値
  メッセージ識別子が無効かどうかを示す値です。
- 例外
  ありません。

(d) getDisableMessageTimestamp メソッド

```
public boolean getDisableMessageTimestamp() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，false を返します。

- 引数
  ありません。
- 戻り値
  メッセージのタイムスタンプが無効かどうかを示す値です。次に示すどちらかを返します。
  - true：メッセージ識別子は使用できません。
  - false：メッセージ識別子は使用できます。

- 例外
  ありません。

(e) getPriority メソッド

```
public int getPriority() throws JMSException
```

メッセージプロデューサに指定された優先度を返します。なお，この値はメッセージに設定された JMSPriority ヘッダの値と同じです。

- 引数
  ありません。
- 戻り値
  メッセージプロデューサ用のメッセージ優先度。
- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージプロデューサをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが優先度の取得に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(f) getTimeToLive メソッド

```
public long getTimeToLive() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，0 を返します。

- 引数
  ありません。
- 戻り値
  メッセージの有効期間（ミリ秒単位）。ゼロは無制限を示します。
- 例外
  ありません。

(g) setDeliveryMode メソッド

```
public void setDeliveryMode(int deliveryMode) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
deliveryMode	メッセージプロデューサのメッセージの管理形態。次に示すどちらかの値を指定します。 • DeliveryMode.NON_PERSISTENT：非永続 • DeliveryMode.PERSISTENT：永続

- 戻り値

  ありません。

- 例外

  ありません。

(h) setDisableMessageID メソッド

```
public void setDisableMessageID(boolean value) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
value	メッセージ識別子が無効かどうかを示す値

- 戻り値

  ありません。

- 例外

  ありません。

(i) setDisableMessageTimestamp メソッド

```
public void setDisableMessageTimestamp(boolean value)
  throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
value	メッセージ識別子が無効かどうかを示す値

- 戻り値

  ありません。

- 例外

  ありません。

(j) setPriority メソッド

```
public void setPriority(int defaultPriority) throws JMSException
```

プロデューサのデフォルトの優先度を指定します。0（最低）～9（最高）の範囲の値を指定します。デフォルト値は javax.jms.Message.DEFAULT_PRIORITY です。範囲以外の値を指定した場合，JMSException が発生します。

- 引数

引数名	説明
defaultPriority	メッセージプロデューサ用のメッセージ優先度。0～9 の範囲で値を指定します。

- 戻り値

  ありません。

- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージプロデューサをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが優先度の設定に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(k) setTimeToLive メソッド

```
public void setTimeToLive(long timeToLive) throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
timeToLive	メッセージの有効期間（ミリ秒単位）。ゼロは無制限を示します。

- 戻り値
  ありません。
- 例外
  ありません。

(l) getQueue メソッド

```
public Queue getQueue() throws JMSException
```

キューセンダに関連づけられたキューを返します。

- 引数
  ありません。
- 戻り値
  送信側のキュー。
- 例外

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージプロデューサをクローズしている状態でメソッドを発行しました。
	内部エラーのために JMS プロバイダが QueueSender オブジェクトに関連づけられたキューの取得に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(m) send メソッド

```
public void send(Message message) throws JMSException
```

メッセージをキューに送信します。キューセンダで設定した優先度が使用されます。

送信メソッドの実行時にメッセージを変更した場合，送信の結果は保証されません。

- 引数

引数名	説明
message	送信するメッセージ

- 戻り値
  ありません。

- 例外

例外クラス	説明
InvalidDestinationException	JMS クライアントがメソッドを使用する際，キューセンダに無効なキューを指定しました。
JMSException	セッション，コネクションまたはメッセージプロデューサをクローズしている状態でメソッドを発行しました。
	message 引数に null が指定されました。
	内部エラーのために JMS プロバイダがメッセージの送信に失敗しました。
MessageFormatException	無効なメッセージが指定されました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(n)　send メソッド

```
public void send(Message message, int deliveryMode, int priority,
  long timeToLive) throws JMSException
```

メッセージをキューに送信します。

- 送信メソッドの実行時にメッセージを変更した場合，送信の結果は保証されません。
- このメソッドを発行した場合，deliveryMode 引数と timeToLive 引数を無視して実行します。

- 引数

引数名	説明
message	送信するメッセージ
deliveryMode	使用するメッセージの管理形態。次に示すどちらかの値を指定します。 • DeliveryMode.NON_PERSISTENT：非永続 • DeliveryMode.PERSISTENT：永続
priority	メッセージの優先度。0（最低）～9（最高）の範囲で値を指定します。
timeToLive	メッセージの有効期間（ミリ秒）

- 戻り値

  ありません。

- 例外

例外クラス	説明
InvalidDestinationException	JMS クライアントがメソッドを使用する際，キューセンダに無効なキューを指定しました。

例外クラス	説明
JMSException	セッション，コネクションまたはメッセージプロデューサをクローズしている状態でメソッドを発行しました。
	message 引数に null が指定されました。
	内部エラーのために JMS プロバイダがメッセージの送信に失敗しました。
MessageFormatException	無効なメッセージが指定されました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(o)　send メソッド

```
public void send(Queue queue, Message message) throws JMSException
```

メッセージをキューに送信します。キューセンダで設定した優先度が使用されます。

queue 引数に null が指定された場合，Session.createSender()メソッドの queue 引数に指定されたキューにメッセージを送信します。

送信メソッドの実行時にメッセージを変更した場合，送信の結果は保証されません。

- 引数

引数名	説明
queue	メッセージのあて先キュー
message	送信するメッセージ

- 戻り値

  ありません。

- 例外

例外クラス	説明
InvalidDestinationException	JMS クライアントがメソッドを使用する際，キューセンダに無効なキューを指定しました。
JMSException	セッション，コネクションまたはメッセージプロデューサをクローズしている状態でメソッドを発行しました。
	message 引数に null が指定されました。
	内部エラーのために JMS プロバイダがメッセージの送信に失敗しました。
MessageFormatException	無効なメッセージが指定されました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(p) send メソッド

```
public void send(Queue queue, Message message, int deliveryMode,
  int priority, long timeToLive) throws JMSException
```

メッセージをキューに送信します。

queue 引数に null が指定された場合，Session.createSender()メソッドの queue 引数に指定されたキューにメッセージを送信します。

- 送信メソッドの実行時にメッセージを変更した場合，送信の結果は保証されません。
- このメソッドを発行した場合，deliveryMode 引数と timeToLive 引数を無視して実行します。

- 引数

引数名	説明
queue	メッセージのあて先キュー
message	送信するメッセージ
deliveryMode	使用するメッセージの管理形態。次に示すどちらかの値を指定します。 • DeliveryMode.NON_PERSISTENT：非永続 • DeliveryMode.PERSISTENT：永続
priority	メッセージの優先度。0（最低）～9（最高）の範囲で値を指定します。
timeToLive	メッセージの有効期間（ミリ秒）

- 戻り値

  ありません。

- 例外

例外クラス	説明
InvalidDestinationException	JMS クライアントがメソッドを使用する際，キューセンダに無効なキューを指定しました。
JMSException	セッション，コネクションまたはメッセージプロデューサをクローズしている状態でメソッドを発行しました。
	message 引数に null が指定されました。
	priority 引数に 0～9 の範囲外の値が指定されました。
	内部エラーのために JMS プロバイダがメッセージの送信に失敗しました。
MessageFormatException	無効なメッセージが指定されました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.12　QueueSession（Session）インタフェース

QueueSession（Session）インタフェースはアプリケーションが使用する論理的なコネクションハンドルです。

(1)　メッセージオブジェクトの生成

メッセージオブジェクト（Message，BytesMessage，ObjectMessage または TextMessage オブジェクト）を生成します。

(2)　キューセンダの生成

QueueSender オブジェクトを生成します。QueueSender オブジェクトを生成する際，メッセージのあて先を createSender()メソッドの引数に指定します。

(3)　キューレシーバの生成

QueueReceiver オブジェクトを生成します。QueueReceiver オブジェクトを生成する際，メッセージセレクタを設定できます。

(4)　キューブラウザの生成

QueueBrowser オブジェクトを生成します。QueueBrowser オブジェクトを生成する際，メッセージセレクタを設定できます。

(5)　ローカルトランザクションのコミットとロールバック

キューセンダとキューレシーバにわたるローカルトランザクションの開始，コミットおよびロールバックを実行します。ローカルトランザクションについては，「2.6.6(1)　ローカルトランザクションの利用」を参照してください。

(6)　キューセッションのクローズ

QueueSession オブジェクトをクローズできます。クローズによって，QueueSession オブジェクトに関連づけられたすべての QueueSender，QueueReceiver および QueueBrowser オブジェクトをクローズし，QueueSession オブジェクトに関連づけられたすべてのリソースを解放します。

QueueSession（Session）インタフェースのメソッドは，複数スレッドからの同時発行を JMS で禁止されています。ただし，close()メソッドだけは複数スレッドから同時発行してパラレルに実行できます。また，クローズされた QueueSession オブジェクトに関連するインタフェースのメソッドを実行すると IllegalStateException が発生しますが，close()メソッドでは発生しません。

(7) 配送メッセージのリカバー

一度配送されたメッセージを再び配送できるようにします。recover()メソッドの動作については，「2.6.6(4) トランザクションとメッセージ受信」を参照してください。

(8) 形式

```
public interface Session extends java.lang.Runnable
{
  public static final int  AUTO_ACKNOWLEDGE;
  public static final int  CLIENT_ACKNOWLEDGE;
  public static final int  DUPS_OK_ACKNOWLEDGE;
  public void     close() throws JMSException;
  public void     commit() throws JMSException;
  public BytesMessage
                  createBytesMessage() throws JMSException;
  public MapMessage
                  createMapMessage() throws JMSException;
  public Message  createMessage() throws JMSException;
  public ObjectMessage
                  createObjectMessage() throws JMSException;
  public ObjectMessage
                  createObjectMessage(java.io.Serializable object)
                    throws JMSException;
  public StreamMessage
                  createStreamMessage() throws JMSException;
  public TextMessage
                  createTextMessage() throws JMSException;
  public TextMessage
                  createTextMessage(java.lang.String text)
                    throws JMSException;
  public MessageListener
                  getMessageListener() throws JMSException;
  public boolean  getTransacted() throws JMSException;
  public void     recover() throws JMSException;
  public void     rollback() throws JMSException;
  public void     run();
  public void     setMessageListener(MessageListener listener)
                    throws JMSException;
}

public interface QueueSession extends Session
{
  public QueueBrowser
                  createBrowser(Queue queue) throws JMSException;
  public QueueBrowser
                  createBrowser(Queue queue,
                    java.lang.String messageSelector)
                    throws JMSException;
  public Queue    createQueue(java.lang.String queueName)
                    throws JMSException;
  public QueueReceiver
                  createReceiver(Queue queue) throws JMSException;
  public QueueReceiver
                  createReceiver(Queue queue,
                    java.lang.String messageSelector)
```

```
                    throws JMSException;
  public QueueSender
                  createSender(Queue queue) throws JMSException;
  public TemporaryQueue
                  createTemporaryQueue() throws JMSException;
}
```

(9)　フィールド

「(8)　形式」に記載した順序で各フィールドを説明します。

(a)　AUTO_ACKNOWLEDGE フィールド

```
public static final int AUTO_ACKNOWLEDGE
```

このメッセージ承認モードでは，次に示す場合にセッションが JMS クライアントのメッセージ受信を自動的に確認応答します。

- セッションが receive()メソッドの発行からの復帰に成功した場合
- メッセージを処理するためにセッションが発行したメッセージリスナが復帰に成功した場合

(b)　CLIENT_ACKNOWLEDGE フィールド

```
public static final int CLIENT_ACKNOWLEDGE
```

このメッセージ承認モードでは，JMS クライアントがメッセージの acknowledge()メソッドを発行して処理済みのメッセージを確認応答します。

(c)　DUPS_OK_ACKNOWLEDGE フィールド

```
public static final int DUPS_OK_ACKNOWLEDGE
```

このメッセージ承認モードは AUTO_ACKNOWLEDGE と同じ動作です。

(10)　メソッド

「(8)　形式」に記載した順序で各メソッドを説明します。

(a)　close メソッド

```
public void close() throws JMSException
```

セッションをクローズします。セッションには幾つかのリソースが割り当てられている場合があるため，リソースが不要になった場合はセッションをクローズしてください。メソッドを発行すると次に示すとおり動作します。

- セッションのトランザクションはロールバックされます。

- セッションに関連づけられているすべての QueueSender，QueueReceiver および QueueBrowser オブジェクトがクローズされます。
- メッセージ処理が適切な方法で終了されるまで，メソッドは復帰しません。
- セッションのすべての未承認メッセージは QueueSession.recover()メソッドが正常終了したときと同じ状態になります。

なお，クローズされているセッションをクローズしても例外は発生しません。

複数スレッドから同時に発行できるメソッドは，QueueSession インタフェースではこのメソッドだけです。

- 引数

  ありません。
- 戻り値

  ありません。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがセッションをクローズする処理に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　commit メソッド

```
public void commit() throws JMSException
```

トランザクションで実行されたメッセージをすべてコミットし，設定中のロックをすべて解放します。

- 引数

  ありません。
- 戻り値

  ありません。
- 例外

例外クラス	説明
IllegalStateException	セッションのトランザクションが開始していない状態で，メソッドが発行されました。
	トランザクションマネジャでのトランザクションが進行中（例えば，分散トランザクションが JTA によって制御されている）であるため，コミットに失敗しました。

例外クラス	説明
	トランザクションマネジャでのトランザクションがサスペンド中であるため，コミットに失敗しました。
	セッションまたはコネクションがクローズしている状態で，メソッドが発行されました。
JMSException	内部エラーのために JMS プロバイダがトランザクションのコミットに失敗しました。
TransactionRolledBackException	コミット時に内部エラーのためにトランザクションがロールバックされました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(c)　createBytesMessage メソッド

```
public BytesMessage createBytesMessage() throws JMSException
```

BytesMessage オブジェクトを作成します。

- 引数
  ありません。
- 戻り値
  BytesMessage オブジェクト。
- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがメッセージの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(d)　createMapMessage メソッド

```
public MapMessage createMapMessage() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，JMSException が発生します。

- 引数
  ありません。

- 戻り値
  MapMessage オブジェクト。
- 例外

例外クラス	説明
JMSException	メソッドが未サポートであることを示します。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(e)　createMessage メソッド

```
public Message createMessage() throws JMSException
```

Message オブジェクトを作成します。

- 引数
  ありません。
- 戻り値
  Message オブジェクト。
- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがメッセージの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(f)　createObjectMessage メソッド

```
public ObjectMessage createObjectMessage() throws JMSException
```

ObjectMessage オブジェクトを作成します。

- 引数
  ありません。
- 戻り値
  ObjectMessage オブジェクト。
- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがメッセージの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(g)　createObjectMessage メソッド

```
public ObjectMessage createObjectMessage(java.io.Serializable object)
  throws JMSException
```

ペイロードを設定済みの ObjectMessage オブジェクトを，object 引数を基に作成します。

- 引数

引数名	説明
object	メッセージの初期化に使用するオブジェクト

- 戻り値

  ペイロードを設定済みの ObjectMessage オブジェクトを，object 引数を基に作成します。

- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがメッセージの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(h)　createStreamMessage メソッド

```
public StreamMessage createStreamMessage() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，JMSException が発生します。

- 引数

  ありません。

- 戻り値

  StreamMessage オブジェクト。

- 例外

例外クラス	説明
JMSException	メソッドが未サポートであることを示します。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(i)　createTextMessage メソッド

```
public TextMessage createTextMessage() throws JMSException
```

TextMessage オブジェクトを作成します。

- 引数

  ありません。

- 戻り値

  TextMessage オブジェクト。

- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがメッセージの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(j)　createTextMessage メソッド

```
public TextMessage createTextMessage(java.lang.String text)
  throws JMSException
```

ペイロードを設定済みの TextMessage オブジェクトを，text 引数を基に作成します。

- 引数

引数名	説明
text	メッセージの初期化に使用する文字列

- 戻り値

  ペイロードを設定済みの TextMessage オブジェクトを，text 引数を基に作成します。

- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがメッセージの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(k)　getMessageListener メソッド

```
public MessageListener getMessageListener() throws JMSException
```

このメソッドは未サポートです。

メソッドを発行した場合，null を返します。

- 引数
  ありません。
- 戻り値
  セッションに関連づけられたメッセージリスナ。
- 例外
  ありません。

(l)　getTransacted メソッド

```
public boolean getTransacted() throws JMSException
```

セッションがトランザクションモードであるかどうかを示します。

- 引数
  ありません。
- 戻り値
  セッションがトランザクションモードである場合は true，そうでない場合は false を返します。
- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがトランザクションモード値の取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(m) recover メソッド

```
public void recover() throws JMSException
```

メソッドを発行すると次に示すとおり動作します。

- メッセージ配送を停止します。
- 今までセッションを介して配送されたメッセージに再配送マークを付けて，再配送できるようにします。
- すべての配送済みの未承認メッセージを含む配送シーケンスを再開します。ただし，元の配送順序と同じ順序でメッセージが再配送されることは保証しません。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
IllegalStateException	トランザクションマネジャでのトランザクションが開始しているセッションで，メソッドが発行されました。
	トランザクションマネジャでのトランザクションが開始していることによって，メソッドが失敗しました。
	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがメッセージ配送の停止および再開に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(n) rollback メソッド

```
public void rollback() throws JMSException
```

トランザクションで実行されたメッセージをすべてロールバックし，設定中のロックをすべて解放します。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外

例外クラス	説明
IllegalStateException	セッションのトランザクションが開始していない状態で，メソッドが発行されました。
	トランザクションマネジャでのトランザクションが進行中（例えば，分散トランザクションが JTA によって制御されている）であるため，ロールバックに失敗しました。
	トランザクションマネジャでのトランザクションがサスペンド中であるため，ロールバックに失敗しました。
	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのために JMS プロバイダがトランザクションのロールバックに失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(o)　run メソッド

```
public void run()
```

このメソッドは未サポートです。

このメソッドを発行した場合，実質的な処理は行われません。

- 引数
  ありません。
- 戻り値
  ありません。
- 例外
  ありません。

(p)　setMessageListener メソッド

```
public void setMessageListener(MessageListener listener)
  throws JMSException
```

このメソッドは未サポートです。

このメソッドで設定した値は実際には使用されません。

- 引数

引数名	説明
listener	セッションに関連づけるメッセージリスナ

- 戻り値
  ありません。
- 例外
  ありません。

(q) createBrowser メソッド

```
public QueueBrowser createBrowser(Queue queue) throws JMSException
```

QueueBrowser オブジェクトを作成して，指定されたキューのメッセージを調べます。

- 引数

引数名	説明
queue	アクセス対象のキュー

- 戻り値
  QueueBrowser オブジェクト。
- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
InvalidDestinationException	無効なキューが指定されたために QueueBrowser オブジェクトの生成に失敗しました。
JMSException	内部エラーのためにセッションが QueueBrowser オブジェクトの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(r) createBrowser メソッド

```
public QueueBrowser createBrowser(Queue queue,
  java.lang.String messageSelector) throws JMSException
```

QueueBrowser オブジェクトを作成し，メッセージセレクタを使用して指定されたキューのメッセージを調べます。

- 引数

引数名	説明
queue	アクセス対象のキュー
messageSelector	メッセージセレクタ構文

- 戻り値
  QueueBrowser オブジェクト。
- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
InvalidDestinationException	無効なキューが指定されたために QueueBrowser オブジェクトの生成に失敗しました。
InvalidSelectorException	無効なメッセージセレクタを設定しました。
JMSException	内部エラーのためにセッションが QueueBrowser オブジェクトの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(s)　createQueue メソッド

```
public Queue createQueue(java.lang.String queueName)
  throws JMSException
```

キューの名前を指定して，キューの識別情報を作成します。

queueName 引数には hrmmkque コマンドに指定したキュー名を指定します。

- 引数

引数名	説明
queueName	キューの名前

- 戻り値
  指定された名前の Queue オブジェクト。
- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
JMSException	内部エラーのためにセッションがキューの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(t) createReceiver メソッド

```
public QueueReceiver createReceiver(Queue queue) throws JMSException
```

QueueReceiver オブジェクトを作成して，指定されたキューからメッセージを受信します。

- 引数

引数名	説明
queue	アクセス対象のキュー

- 戻り値

  QueueReceiver オブジェクト。

- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
InvalidDestinationException	無効なキューが指定されたために QueueReceiver オブジェクトの生成に失敗しました。
JMSException	内部エラーのためにセッションが QueueReceiver オブジェクトの作成に失敗しました。

  例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(u) createReceiver メソッド

```
public QueueReceiver createReceiver(Queue queue,
  java.lang.String messageSelector) throws JMSException
```

QueueReceiver オブジェクトを作成し，メッセージセレクタを使用して指定されたキューからメッセージを受信します。

- 引数

引数名	説明
queue	アクセス対象のキュー。
messageSelector	メッセージセレクタ。メッセージセレクタ構文に一致するプロパティを保持するメッセージだけが配送されます。

- 戻り値

  QueueReceiver オブジェクト。

- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
InvalidDestinationException	無効なキューが指定されたために QueueReceiver オブジェクトの生成に失敗しました。
InvalidSelectorException	無効なメッセージセレクタを設定しました。
JMSException	内部エラーのためにセッションが QueueReceiver オブジェクトの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(v)　createSender メソッド

```
public QueueSender createSender(Queue queue) throws JMSException
```

QueueSender オブジェクトを作成して，指定されたキューにメッセージを送信します。

- 引数

引数名	説明
queue	アクセス対象のキュー

- 戻り値

  QueueSender オブジェクト。

- 例外

例外クラス	説明
IllegalStateException	セッションまたはコネクションがクローズしている状態で，メソッドを発行しました。
InvalidDestinationException	無効なキューが指定されたために QueueSender オブジェクトの生成に失敗しました。
JMSException	内部エラーのためにセッションが QueueSender オブジェクトの作成に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(w)　createTemporaryQueue メソッド

```
public TemporaryQueue createTemporaryQueue() throws JMSException
```

このメソッドは未サポートです。

このメソッドを発行した場合，JMSException が発生します。

- 引数
  ありません。
- 戻り値
  TemporaryQueue オブジェクト。
- 例外

例外クラス	説明
JMSException	メソッドが未サポートであることを示します。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.4.13　TextMessage インタフェース

TextMessage インタフェースは java.lang.String 型のデータを含むメッセージを送受信するために使用します。TextMessage インタフェースは Message インタフェースを継承しているため，Message インタフェースの機能を持っています。

(1)　ペイロードの設定と取得

TextMessage インタフェースを使用することでペイロードを設定および取得できます。このペイロードは java.lang.String 型のデータです。JMS メッセージの各要素については，「2.5.1　JMS メッセージの構成」を参照してください。

(2)　形式

```
public interface TextMessage extends Message
{
  public java.lang.String
                 getText() throws JMSException;
  public void    setText(java.lang.String string) throws JMSException;
}
```

(3)　フィールド

ありません。

(4)　メソッド

「(2)　形式」に記載した順序で各メソッドを説明します。

(a) getText メソッド

```
public java.lang.String getText() throws JMSException
```

メッセージのデータを含む文字列を返します。メッセージ本体が設定されていない場合はデフォルト値である null を返します。

- 引数
  ありません。
- 戻り値
  メッセージのデータを含む String オブジェクト。
- 例外

例外クラス	説明
JMSException	内部エラーのために JMS プロバイダがテキストの取得に失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b) setText メソッド

```
public void setText(java.lang.String string) throws JMSException
```

メッセージのデータを含む文字列を設定します。

- 引数

引数名	説明
string	メッセージのデータを含む String オブジェクト

- 戻り値
  ありません。
- 例外

例外クラス	説明
JMSException	string 引数に null が指定されました。
	内部エラーのために JMS プロバイダがテキストの設定に失敗しました。
MessageNotWriteableException	読み取り専用モードのペイロードに情報を書き込もうとしました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

7.5　転送データ相互接続用インタフェースの一覧

Reliable Messaging の転送データ相互接続用インタフェースは永続版リソースアダプタだけが使用できます。非永続版リソースアダプタでは使用できません。

転送データ相互接続用インタフェースは，次に示すパッケージによって提供されます。

```
jp.co.Hitachi.soft.reliablemessaging.api
```

Reliable Messaging が提供する転送データ相互接続用インタフェースの一覧を次の表に示します。

表 7-3　転送データ相互接続用インタフェースの一覧

項番	インタフェース名	機能
1	BytesContainerFactory	BytesContainer を生成します。
2	BytesContainer	設定したデータをバイト配列で保持します。提供する機能は，次のとおりです。 • バイト配列およびバイト配列のデータ型の設定 • バイト配列およびバイト配列のデータ型の取得 • 保持するバイト配列数の取得 • 保持する全データの消去

転送データ相互接続用インタフェース使用時の例外クラスは，次に示すパッケージによって提供されます。

```
jp.co.Hitachi.soft.reliablemessaging.api
```

転送データ相互接続用インタフェース使用時の例外クラスの一覧を次の表に示します。

表 7-4　転送データ相互接続用インタフェースの例外クラスの一覧

項番	例外クラス名	例外が発生する状況
1	HRMException	すべての転送データ相互接続用 API 例外のルートクラスであるため，特定のタイミングに該当しません。
2	HRMIllegalArgumentException	メソッドの呼び出しで不正な引数を指定した場合

転送データ相互接続用インタフェースおよび例外のクラスは，次に示すライブラリで提供されます。

```
%HRMDIR%¥lib¥reliablemessaging-api.jar
```

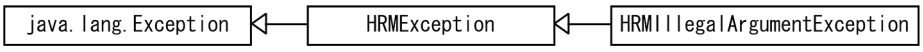
転送データ相互接続用インタフェースを使用するアプリケーションのコンパイル時には，上に示すライブラリをクラスパスに含めてください。

7.6　転送データ相互接続用インタフェース継承図

Reliable Messaging の転送データ相互接続用インタフェース BytesContainerFactory および BytesContainer には，継承関係はありません。

一方，例外クラスには継承関係があります。例外クラスの継承を次の図に示します。

図 7-3　転送データ相互接続用例外クラスの継承



（凡例）

：右が左を継承します。

7.7　転送データ相互接続用インタフェースの使い方

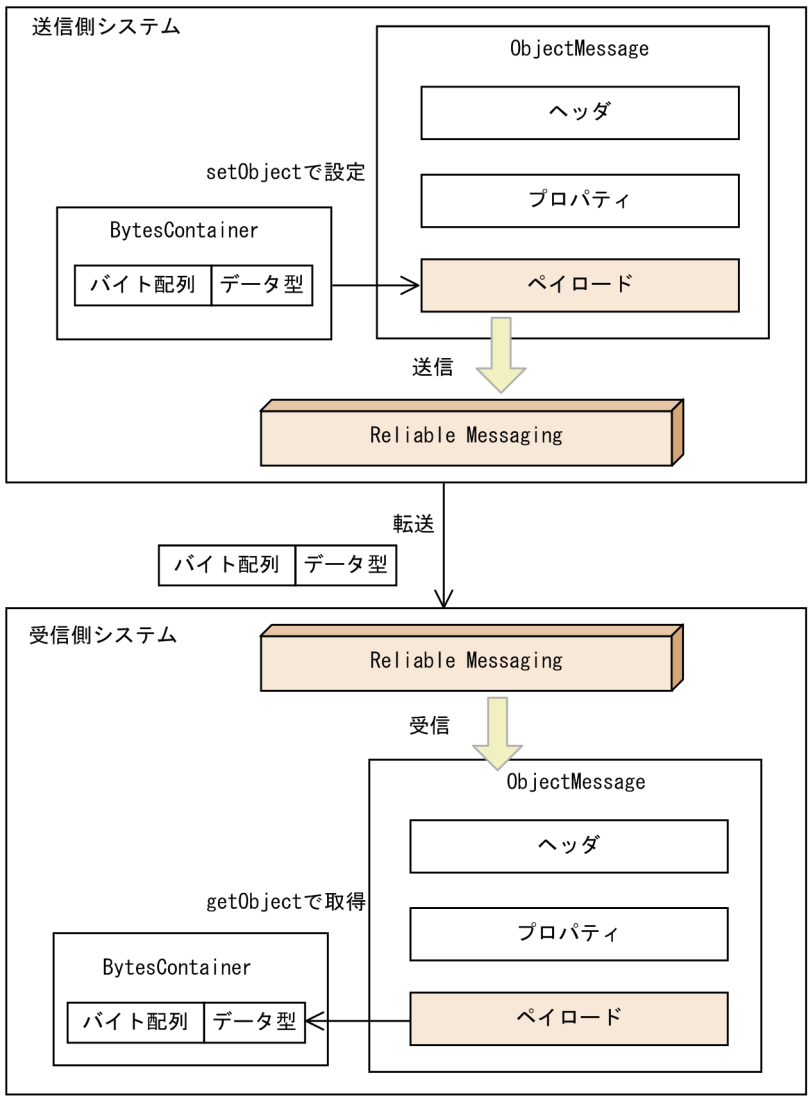
JMS インタフェースを使用してメッセージを送受信する際，送信メッセージに BytesContainer を設定する方法と，受信したメッセージから BytesContainer を取得する方法を次の表に示します。

表 7-5　BytesContainer の使用方法

項番	処理内容	BytesContainer の使用方法
1	送信処理	javax.jms.ObjectMessage オブジェクトの setObject メソッドで BytesContainer オブジェクトを設定してメッセージを送信します。
2	受信処理	受信処理で取得した javax.jms.ObjectMessage オブジェクトの getObject メソッドを使用して BytesContainer オブジェクトを取得します。

Reliable Messaging 同士の通信での BytesContainer の使用方法の概要を次の図に示します。

図 7-4　Reliable Messaging 同士の通信での BytesContainer 使用方法の概要



他システムとの通信での BytesContainer 使用方法の概要を次の図に示します。

図 7-5　Reliable Messaging が送信側の場合の他システムとの通信

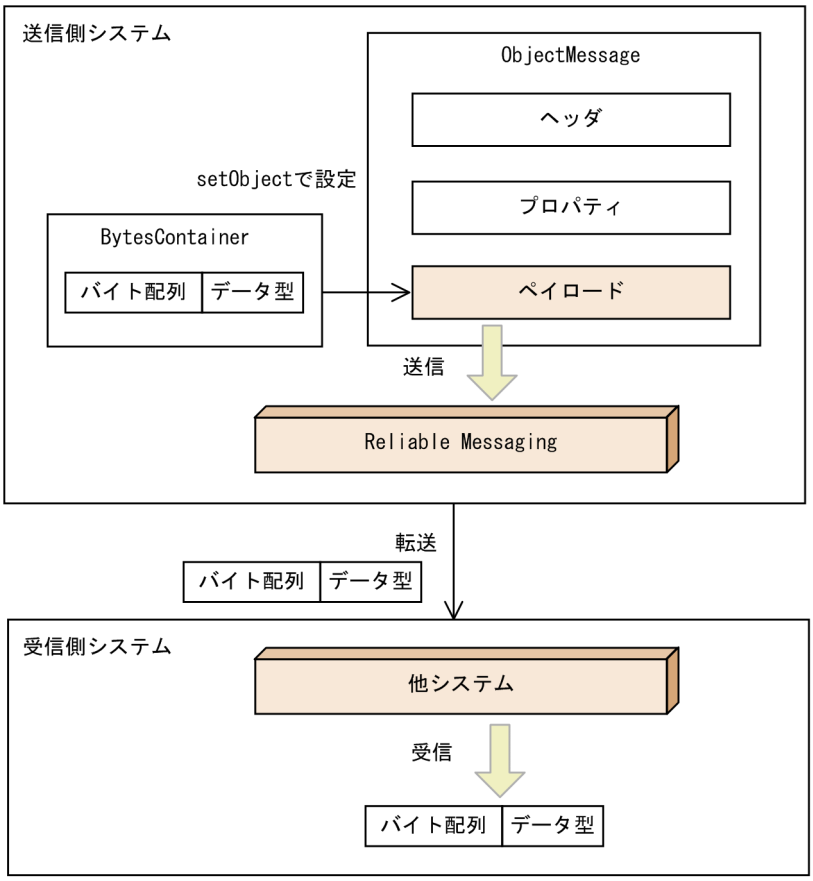
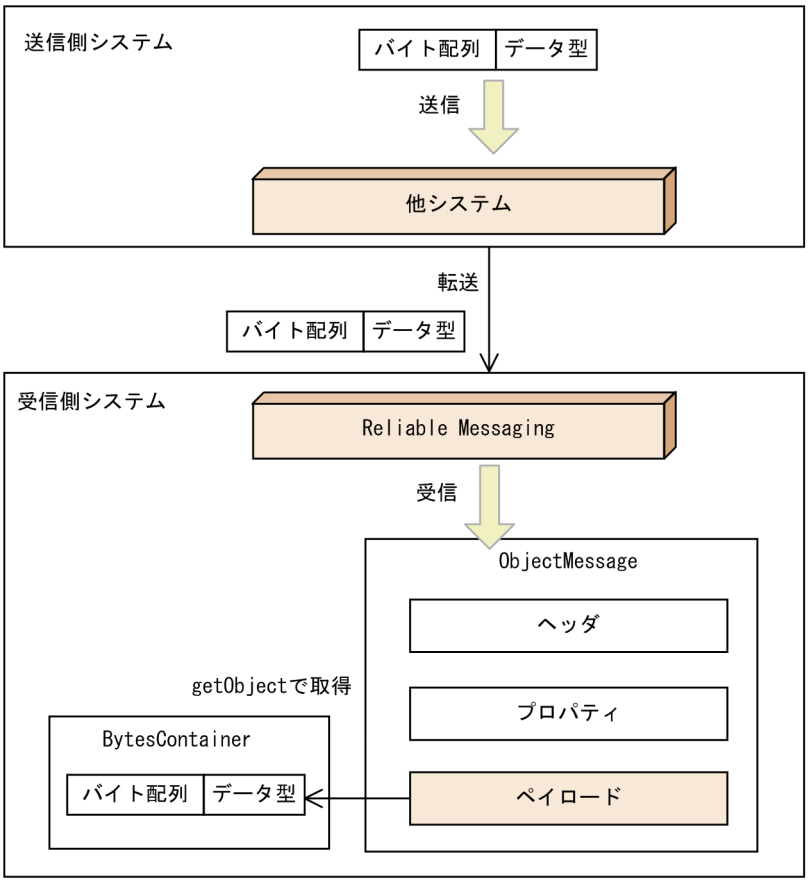


図 7-6　Reliable Messaging が受信側の場合の他システムとの通信



7.8　転送データ相互接続用インタフェースの詳細

転送データ相互接続用インタフェースを，BytesContainerFactory インタフェース，BytesContainer インタフェース，HRMException クラス，HRMIllegalArgumentException クラスの順に説明します。

7.8.1　BytesContainerFactory インタフェース

BytesContainerFactory インタフェースは BytesContainer を生成します。

アプリケーションでは，QueueConnection オブジェクトの createQueueSession メソッドで取得した QueueSession オブジェクトに対して BytesContainerFactory をキャストすることによって，BytesContainerFactory オブジェクトを取得できます。

(1)　BytesContainer の生成

createBytesContainer メソッドを呼び出すことで BytesContainer を生成します。

(2)　形式

```
public interface BytesContainerFactory
{
  public BytesContainer createBytesContainer();
}
```

(3)　フィールド

ありません。

(4)　メソッド

「(2)　形式」に記載した順序で各メソッドを説明します。

(a)　createBytesContainer メソッド

```
public BytesContainer createBytesContainer()
```

BytesContainer を作成します。

- 引数
  ありません。
- 戻り値
  新たに作成された BytesContainer。

7.8.2 BytesContainer インタフェース

BytesContainer インタフェースはバイト配列を含むメッセージを送受信するために使用します。Reliable Messaging 同士だけでなく，WS-Reliability1.1 に準拠する他システムとデータをやり取りする際に使用できます。

(1) BytesContainer インタフェースの仕様

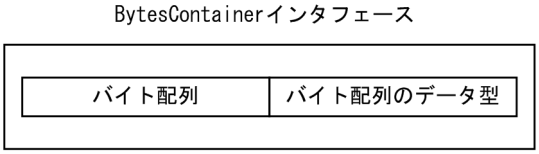
BytesContainer インタフェースは，バイト配列およびバイト配列のデータ型を示す値の組を 1 個だけ含むインタフェースです。

BytesContainer インタフェースは，次の二つの要素で構成されます。

- バイト配列
  プリミティブなバイト配列です。
- バイト配列のデータ型
  バイト配列のデータ型を示す値が入ります。

BytesContainer インタフェースの構成を次の図に示します。

図 7-7 BytesContainer インタフェースの構成



(2) BytesContainer インタフェースのデータ型

BytesContainer インタフェースに含まれるバイト配列のデータ型の一覧を次の表に示します。

表 7-6 バイト配列のデータ型一覧

項番	データ型の種類	バイト配列のデータ型の内容
1	HTML 型	拡張子が html，htm のファイルの内容をバイト配列化したデータ
2	TEXT 型	拡張子が txt，text のファイルの内容をバイト配列化したデータ※
3	IEF 型	拡張子が ief のファイルの内容をバイト配列化したデータ
4	TIFF 型	拡張子が tiff, tif のファイルの内容をバイト配列化したデータ
5	XWD 型	拡張子が xwd のファイルの内容をバイト配列化したデータ
6	POSTSCRIPT 型	拡張子が ai, eps, ps のファイルの内容をバイト配列化したデータ
7	RTF 型	拡張子が rtf のファイルの内容をバイト配列化したデータ
8	TEX 型	拡張子が tex のファイルの内容をバイト配列化したデータ

項番	データ型の種類	バイト配列のデータ型の内容
9	TEXINFO 型	拡張子が texinfo，texi のファイルの内容をバイト配列化したデータ
10	ROFF 型	拡張子が t，tr，roff のファイルの内容をバイト配列化したデータ
11	AU 型	拡張子が au のファイルの内容をバイト配列化したデータ
12	MIDI 型	拡張子が midi，mid のファイルの内容をバイト配列化したデータ
13	AIFC 型	拡張子が aifc のファイルの内容をバイト配列化したデータ
14	AIF 型	拡張子が aif，aiff のファイルの内容をバイト配列化したデータ
15	WAV 型	拡張子が wav のファイルの内容をバイト配列化したデータ
16	MPEG 型	拡張子が mpeg，mpg，mpe のファイルの内容をバイト配列化したデータ
17	QT 型	拡張子が qt，mov のファイルの内容をバイト配列化したデータ
18	AVI 型	拡張子が avi のファイルの内容をバイト配列化したデータ
19	上に示す以外の型	上に示す以外の内容をバイト配列化したデータ

注※

データ型が TEXT 型の場合，文字コード（charset）を UTF-8 に指定して転送します。

(3) バイト配列数の取得

BytesContainer インタフェースが保持するバイト配列の数を取得する機能を提供します。

(4) 保持する全データの消去

BytesContainer インタフェースが保持するデータ（バイト配列およびバイト配列のデータ型）をすべて消去する機能を提供します。

(5) 形式

```
public interface BytesContainer
{
  public static final String TEXT_HTML;
  public static final String TEXT_PLAIN;
  public static final String IMAGE_IEF;
  public static final String IMAGE_TIFF;
  public static final String IMAGE_X_XWINDOWDUMP;
  public static final String APPLICATION_POSTSCRIPT;
  public static final String APPLICATION_RTF;
  public static final String APPLICATION_X_TEX;
  public static final String APPLICATION_X_TEXINFO;
  public static final String APPLICATION_X_TROFF;
  public static final String AUDIO_BASIC;
  public static final String AUDIO_MIDI;
  public static final String AUDIO_X_AIFC;
  public static final String AUDIO_X_AIFF;
  public static final String AUDIO_X_WAV;
  public static final String VIDEO_MPEG ;
```

```
  public static final String VIDEO_QUICKTIME;
  public static final String VIDEO_X_MSVIDEO;
  public static final String APPLICATION_OCTET_STREAM;

  public void addPayload(byte[] payload) throws HRMException;
  public void addPayload(byte[] payload, String contentType) throws HRMException;
  public int size();
  public byte[] getPayload(int number) throws HRMException;
  public String getContentType(int number) throws HRMException;
  public void clear();
}
```

(6) フィールド

「(5) 形式」に記載した順序で各フィールドを説明します。

(a) TEXT_HTML フィールド

```
public static final String TEXT_HTML
```

バイト配列のデータが HTML 型であることを示す値です。

(b) TEXT_PLAIN フィールド

```
public static final String TEXT_PLAIN
```

バイト配列のデータが TEXT 型であることを示す値です。

(c) IMAGE_IEF フィールド

```
public static final String IMAGE_IEF
```

バイト配列のデータが IEF 型であることを示す値です。

(d) IMAGE_TIFF フィールド

```
public static final String IMAGE_TIFF
```

バイト配列のデータが TIFF 型であることを示す値です。

(e) IMAGE_X_XWINDOWDUMP フィールド

```
public static final String IMAGE_X_XWINDOWDUMP
```

バイト配列のデータが XWD 型であることを示す値です。

(f)　APPLICATION_POSTSCRIPT フィールド

```
public static final String APPLICATION_POSTSCRIPT
```

バイト配列のデータが POSTSCRIPT 型であることを示す値です。

(g)　APPLICATION_RTF フィールド

```
public static final String APPLICATION_RTF
```

バイト配列のデータが RTF 型であることを示す値です。

(h)　APPLICATION_X_TEX フィールド

```
public static final String APPLICATION_X_TEX
```

バイト配列のデータが TEX 型であることを示す値です。

(i)　APPLICATION_X_TEXINFO フィールド

```
public static final String APPLICATION_X_TEXINFO
```

バイト配列のデータが TEXINFO 型であることを示す値です。

(j)　APPLICATION_X_TROFF フィールド

```
public static final String APPLICATION_X_TROFF
```

バイト配列のデータが ROFF 型であることを示す値です。

(k)　AUDIO_BASIC フィールド

```
public static final String AUDIO_BASIC
```

バイト配列のデータが AU 型であることを示す値です。

(l)　AUDIO_MIDI フィールド

```
public static final String AUDIO_MIDI
```

バイト配列のデータが MIDI 型であることを示す値です。

(m)　AUDIO_X_AIFC フィールド

```
public static final String AUDIO_X_AIFC
```

バイト配列のデータが AIFC 型であることを示す値です。

(n)　AUDIO_X_AIFF フィールド

```
public static final String AUDIO_X_AIFF
```

バイト配列のデータが AIF 型であることを示す値です。

(o)　AUDIO_X_WAV フィールド

```
public static final String AUDIO_X_WAV
```

バイト配列のデータが WAV 型であることを示す値です。

(p)　VIDEO_MPEG フィールド

```
public static final String VIDEO_MPEG
```

バイト配列のデータが MPEG 型であることを示す値です。

(q)　VIDEO_QUICKTIME フィールド

```
public static final String VIDEO_QUICKTIME
```

バイト配列のデータが QT 型であることを示す値です。

(r)　VIDEO_X_MSVIDEO フィールド

```
public static final String VIDEO_X_MSVIDEO
```

バイト配列のデータが AVI 型であることを示す値です。

(s)　APPLICATION_OCTET_STREAM フィールド

```
public static final String APPLICATION_OCTET_STREAM
```

バイト配列のデータが，ほかのフィールドで示したデータ型以外であることを示す値です。

(7)　メソッド

「(5)　形式」に記載した順序で各メソッドを説明します。

(a)　addPayload メソッド

```
public void addPayload(byte[] payload) throws HRMException
```

メッセージのデータを含む byte[]を設定します。payload のデータ型を示す値は APPLICATION_OCTET_STREAM になります。

BytesContainer に設定できるバイト配列の制限数は 1 です。

- 引数

引数名	説明
payload	メッセージのデータを含む byte[]

- 戻り値

  ありません。

- 例外

例外クラス	説明
HRMIllegalArgumentException	引数 payload に null を指定しました。
HRMException	制限数より多くのバイト配列を設定しました。または，内部エラーのために メソッドの処理が失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(b)　addPayload メソッド

```
public void addPayload(byte[] payload, String contentType) throws HRMException
```

メッセージのデータを含む byte[]を設定します。

BytesContainer に設定できるバイト配列の制限数は 1 です。

引数 contentType に TEXT_PLAIN を指定した場合，引数 payload に指定する値は，UTF-8 の文字コードの値として扱います。文字コードが UTF-8 以外の値を指定してキュー間転送をした場合，受信側で同じ値を取得できません。

- 引数

引数名	説明
payload	メッセージのデータを含む byte[]
contentType	payload のデータ型を示す値

- 戻り値

  ありません。

- 例外

例外クラス	説明
HRMIllegalArgumentException	次のうちのどれかの状態になりました。 • 引数 payload に null を指定した • 引数 contentType に null を指定した • 引数 contentType に BytesContainer のフィールドで定義した値以外を設定した
HRMException	制限数より多くのバイト配列を設定しました。または，内部エラーのためにメソッドの処理が失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(c)　size メソッド

```
public int size()
```

BytesContainer インタフェースが保持するバイト配列の数を返します。

- 引数
  ありません。
- 戻り値
  BytesContainer が保持するバイト配列の数を示す値です。
- 例外
  ありません。

(d)　getPayload メソッド

```
public byte[] getPayload(int number) throws HRMException
```

引数 number で指定されたバイト配列を返します。この引数 number と同じ値を getContentType メソッドの引数に指定して発行した場合の値が，TEXT_PLAIN だったとき，このメソッドで取得するデータは UTF-8 の文字コードで変換された値です。

- 引数

引数名	説明
number	取得するバイト配列の番号を示す値です。制限数は 0 です。

- 戻り値
  メッセージのデータを含む byte[]。
- 例外

例外クラス	説明
HRMIllegalArgumentException	引数 number が制限数の範囲外です。または，BytesContainer が保持するバイト配列数を超える値を引数 number に指定しました。
HRMException	内部エラーのためにメソッドの処理が失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(e)　getContentType メソッド

```
public String getContentType(int number) throws HRMException
```

引数 number で指定されたバイト配列のデータ型を返します。

- 引数

引数名	説明
number	取得するバイト配列の番号を示す値です。制限数は 0 です。

- 戻り値

  payload のデータ型を示す値です。

- 例外

例外クラス	説明
HRMIllegalArgumentException	引数 number が制限数の範囲外です。または，BytesContainer が保持するバイト配列数を超える値を引数 number に指定しました。
HRMException	内部エラーのためにメソッドの処理が失敗しました。

例外クラスには障害コードと障害情報が設定されます。障害コードと対処については，「7.9　障害コードの詳細」を参照してください。

(f)　clear メソッド

```
public void clear()
```

BytesContainer インタフェースが保持するデータ（バイト配列およびバイト配列のデータ型）をすべて消去します。

- 引数

  ありません。

- 戻り値

  ありません。

- 例外

ありません。

7.8.3 HRMException クラス

このクラスは転送データ相互接続用 API 例外のルートクラスです。なお，java.lang.Exception の親クラスである java.lang.Throwable クラスから継承したメソッドに関する説明（形式，コンストラクタ，メソッド）は省略します。

(1) 形式

```
public class HRMException extends java.lang.Exception
{
  public HRMException(String errorMessage, String errorCode){}
  public String getErrorCode(){}
  public String getErrorMessage(){}
  public Exception getLinkedException() {}
}
```

(2) コンストラクタ

「(1) 形式」に記載した順序でコンストラクタを説明します。

(a) HRMException コンストラクタ

```
public HRMException(String errorMessage, String errorCode)
```

HRMException を生成する際に，障害文字列と障害コードを設定します。

- 引数

引数名	説明
errorMessage	障害発生要因を示す障害文字列
errorCode	障害発生要因を示す障害コード

- 戻り値

  ありません。

- 例外

  ありません。

(3) メソッド

「(1) 形式」に記載した順序で各メソッドを説明します。

(a) getErrorCode メソッド

```
public java.lang.String getErrorCode()
```

障害発生要因を示す障害コードを返します。

- 引数
  ありません。
- 戻り値
  障害発生要因を示す障害コードです。
- 例外
  ありません。

(b) getErrorMessage メソッド

```
public java.lang.String getErrorMessage()
```

障害発生要因を示す障害文字列を返します。

- 引数
  ありません。
- 戻り値
  障害発生要因を示す障害文字列です。
- 例外
  ありません。

(c) getLinkedException メソッド

```
public java.lang.Exception getLinkedException()
```

このクラスにリンクされた例外を取得します。

- 引数
  ありません。
- 戻り値
  リンクされた例外です。
- 例外
  ありません。

7.8.4　HRMIllegalArgumentException クラス

このクラスは，メソッドの呼び出しで不正な引数を指定したことを示す例外クラスです。なお，java.lang.Exception の親クラスである java.lang.Throwable クラスから継承したメソッドに関する説明（形式，コンストラクタ，メソッド）は省略します。

(1)　形式

```
public class HRMIllegalArgumentException extends HRMException
{
  public HRMIllegalArgumentException(
      String errorMessage, String errorCode) {}
  public String getErrorCode() {}
  public String getErrorMessage() {}
  public Exception getLinkedException() {}
}
```

(2)　コンストラクタ

HRMException クラスと同じ内容です。「7.8.3　HRMException クラス」を参照してください。

(3)　メソッド

HRMException クラスと同じ内容です。「7.8.3　HRMException クラス」を参照してください。

7.9 障害コードの詳細

アプリケーションでは，例外クラスの getErrorCode()メソッドを発行することによって障害コードを取得できます。また，getMessage()メソッドを発行することによって障害文字列を取得できます。障害コードおよび障害文字列を参照して，障害に対処してください。

障害コードの一覧を次の表に示します。

表 7-7 障害コードの一覧

項番	障害コード	障害文字列	発生要因	対処
1	HRM-00001	Null has been specified for the argument.	引数に null を指定しています。	引数には null 以外を指定してください。
2	HRM-00002	The value set for the argument (number) is outside the valid range.	引数の number に制限範囲外の値を設定しました。	引数の number には制限範囲内の値を設定してください。
3	HRM-00003	More bytes than the limit have been set for BytesContainer.	BytesContainer に対して制限より多くのバイト配列を設定しました。	BytesContainer に設定するバイト配列の数が制限を超えないようにしてください。
4	HRM-00004	A value other than that defined in the BytesContainer field has been set for the contentType argument.	引数の contentType に BytesContainer のフィールドで定義した値以外を設定しました。	引数の contentType には BytesContainer のフィールドで定義した値を設定してください。
5	HRM-00005	An element corresponding to the value for the argument (number) was not found.	引数の number の値に対応する要素が見つかりません。	対応する要素が存在する値を引数の number に設定してください。
6	HRM-00006	The destination is invalid. There is no message in this destination.	デスティネーションは不正です。このデスティネーションにはメッセージはありません。	指定したキューを見直してください。送信用共用キューからメッセージ数を取得しようとしていると考えられます。
7	HRM-00103	The destination does not exist.	デスティネーションがありません。	指定したキュー名を見直してください。
8	HRM-00203	The destination is invalid. It is not possible to send to this destination.	デスティネーションは，送信できるあて先ではありません。	指定したキュー名を見直してください。次の場合が考えられます。 デッドメッセージキューに送信しています。
9	HRM-00204	The destination is invalid. It is not possible	デスティネーションは，受信できるあて先ではありません。	受信できないキューを見直してください。次の場合が考えられます。

項番	障害コード	障害文字列	発生要因	対処
		to receive from this destination.		送信用共用キューから受信しています。
10	HRM-00205	This message cannot be sent to a shared queue.	このメッセージは共用キューに送信できません。	共用キューに送信する場合は，BytesMessage インタフェースを使用して送信してください。
11	HRM-00206	The size of this message exceeds the maximum length defined for the queue.	このメッセージの本体のサイズはキューで定義された最大メッセージ長をオーバしています。	メッセージの本体の長さを見直してください。
12	HRM-00207	The system definition information conflicts with the queue attribute information.	システムの定義情報とキューの属性情報が不一致です。	キューの属性情報，またはシステムの定義情報を見直してください。次に示す状況が考えられます。 RMSHConnectFlag プロパティ指定値が false のときに受信用共用キューを利用しようしています。
13	HRM-00208	Registration of messages into the queue is suppressed.	キューへのメッセージの登録は抑止されています。	キューへのメッセージの登録の抑止を解除してください。
14	HRM-00209	Retrieval of messages from the queue is suppressed.	キューからのメッセージの取り出しは抑止されています。	キューからのメッセージの取り出しの抑止を解除してください。
15	HRM-00210	The queue is being deleted.	キューの削除中のため，キューに対する処理が失敗しました。	別のキューを利用するか，またはこのキューが削除されてないか確認して再度処理を実行してください。
16	HRM-00301	uCosminexus Reliable Messaging is not in an executable state.	Reliable Messaging が実行状態ではありません。	Reliable Messaging を実行状態に移行してください。
17	HRM-00302	uCosminexus Reliable Messaging has not been initialized.	Reliable Messaging が初期化されていません。	Reliable Messaging を管理状態または実行状態に移行してください。
18	HRM-00303	The queue is in a blocked state.	キューは閉塞状態です。	キューを使用できないので，異なるキューを使用してください。
19	HRM-00304	RMConnection has already closed.	RMConnection は close 済みです。	発行手順を見直してください。
20	HRM-00306	RMConnection is already in a transaction-executing state.	RMConnection はすでにトランザクション実行状態です。	トランザクションを決着してから処理してください。

項番	障害コード	障害文字列	発生要因	対処
21	HRM-00307	RMConnection is in a transaction-not-executing state.	RMConnection はトランザクションが実行されていない状態です。	トランザクションを開始してから処理してください。
22	HRM-00308	The table management information of the shared queue is invalid. The shared queue status changed to a blocked status.	共用キューのテーブル管理情報が不正です。共用キューは閉塞状態になりました。	次に示す項目を確認してください。 • キュー名 • ライト通番 • ライト通番のラップカウンタ • ライト通番の最大値 • リード通番 • リード通番のラップカウンタ
23	HRM-00401	The number of messages or queue capacity has exceeded the maximum.	キューの容量やメッセージ数の上限値を超えました。	キューの容量を増やすか，メッセージを削除してください。
24	HRM-00407	A memory shortage occurred.	メモリ不足が発生しました。	利用できるメモリを増やし，処理を再度実行してください。
25	HRM-00408	The maximum value of the message length who can register with queue was exceeded.	キューに登録できるメッセージ長の上限値を超過しました。	登録先のキューがローカルキューの場合，メッセージのペイロード，プロパティおよびヘッダの指定値を見直し，メッセージのサイズを小さくしてください。 送信用共用キューの場合，メッセージのペイロードのサイズを小さくするか，または登録先キューとなる受信用共用キューの最大メッセージ長を増やしてください。
26	HRM-00409	The end of the stream was unexpectedly reached while reading the message.	メッセージ読み取り中に予期しないストリームの終端に達しました。	HiRDB Type4 JDBC Driver 対応の DB Connector を使用している場合は，DB Connector の maxBinarySize プロパティに指定値より大きな値を設定して，Reliable Messaging と DB Connector を再開始してください。
27	HRM-00410	The destination queue has been deleted.	送信先の受信用共用キューが削除されています。	送信用共用キューは閉塞するので，削除してください。

項番	障害コード	障害文字列	発生要因	対処
28	HRM-00411	Registration in the destination queue is suppressed.	送信先の受信用共用キューは登録を抑止されています。	送信先の受信用共用キューの抑止が解除されるのを待ってください。
29	HRM-00502	An attempt to access the database has failed.	DB アクセスに失敗しました。	メッセージの詳細から DB の障害を回復してください。
30	HRM-00503	An attempt to access the queue has failed.	キューアクセスに失敗しました。	メッセージの詳細から DB の障害を回復してください。
31	HRM-00504	The number of database connections exceeded the maximum.	DB コネクション数が最大値を超過しました。	Reliable Messaging の DB コネクション数を増やす場合は，DB のコネクション関連定義を見直してください。
32	HRM-00505	An error occurred during a begin or commit or rollback of the local transaction.	ローカルトランザクションのコミットまたはロールバックに失敗しました。	メッセージの詳細から DB の障害を回復してください。
33	HRM-00506	An attempt to acquire a connection to the database has failed.	DB へのコネクションの取得に失敗しました。	メッセージの詳細から DB の障害を回復してください。
34	HRM-00601	uCosminexus Reliable Messaging is in a blocked state.	Reliable Messaging は閉塞状態です。	Reliable Messaging を再度開始してください。
35	HRM-00701	ManagedConnection has already been destroyed.	ManagedConnection はすでにデストロイされています。	メソッドの発行手順を見直してください。
36	HRM-00702	ManagedConnection has already been cleaned up.	ManagedConnection はすでにクリーンアップされています。	メソッドの発行手順を見直してください。
37	HRM-00703	The session transaction is already running.	セッションのトランザクションがすでに開始しています。	メソッドの発行手順を見直してください。
38	HRM-00704	The session transaction is not running.	セッションのトランザクションは開始していません。	メソッドの発行手順を見直してください。
39	HRM-00705	An attempt was made to generate QueueSession without the attribute that enables sharing of QueueSession.	生成しようとした QueueSession オブジェクトはシェアリングできる属性ではありません。	指定した QueueSession オブジェクトの属性を見直してください。
40	HRM-00706	The already generated QueueSession does not have the attribute that	生成済みの QueueSession オブジェクトはシェアリン	シェアリングする際は trasacted 引数が false で，かつ AUTO_ACKNOWLEDGE

項番	障害コード	障害文字列	発生要因	対処
		enables sharing of QueueSession.	グできる属性ではありません。	モードの QueueSession オブジェクトを使用してください。
41	HRM-00801	An unsupported method was executed.	サポートされていないメソッドを実行しました。	発行するメソッドを見直してください。
42	HRM-00802	The method execution failed because the QueueConnection was in a closed state.	QueueConnection オブジェクトが close 状態だったため，メソッドの実行に失敗しました。	メソッドの発行手順を見直してください。
43	HRM-00803	The method execution failed because the QueueSession was in a closed state.	QueueSession オブジェクトが close 状態だったため，メソッドの実行に失敗しました。	メソッドの発行手順を見直してください。
44	HRM-00804	The method execution failed because the QueueSender was in a closed state.	QueueSender オブジェクトが close 状態だったため，メソッドの実行に失敗しました。	メソッドの発行手順を見直してください。
45	HRM-00805	The method execution failed because the QueueReceiver was in a closed state.	QueueReceiver オブジェクトが close 状態だったため，メソッドの実行に失敗しました。	メソッドの発行手順を見直してください。
46	HRM-00806	The method execution failed because the QueueBrowser was in a closed state.	QueueBrowser オブジェクトが close 状態だったため，メソッドの実行に失敗しました。	メソッドの発行手順を見直してください。
47	HRM-00807	The method's parameter value is null.	メソッドの引数の値が null です。	引数を見直してください。
48	HRM-00808	The method's parameter value is invalid.	メソッドの引数の値が不正です。	引数を見直してください。
49	HRM-00809	The length of the method's parameter exceeds the restriction value.	メソッドの引数の長さが制限値を超えていました。	引数の長さを見直してください。
50	HRM-00810	An attempt to read the message has failed.	メッセージの読み取りに失敗しました。	引数を見直してください。
51	HRM-00811	An attempt to write the message has failed.	メッセージの書き込みに失敗しました。	引数を見直してください。

項番	障害コード	障害文字列	発生要因	対処
52	HRM-00812	The method execution failed because the message has been in the acknowledge impossible state.	メッセージが acknowledge できない状態だったため，メソッドの実行に失敗しました。	発行手順を見直してください。
53	HRM-00813	The method execution failed because the end of an unexpected bytes stream has been reached.	予期しないバイトストリームの終端に達しました。	メッセージの読み取り手順を見直してください。
54	HRM-00814	The method execution failed because the type conversion was invalid.	型変換が無効であったため，メソッドの実行に失敗しました。	メッセージに格納したプロパティまたはペイロードの型を見直してください。 ObjectMessage オブジェクトに格納するオブジェクトがユーザの定義したクラスの場合，次の内容を確認してください。 • クラスがシリアライズできるかどうか • アプリケーションがクラスを保持しているかどうか • クラスパスが Application Server の J2EE サーバ用オプション定義ファイルに指定されているかどうか BytesMessage オブジェクト以外の種類のメッセージを共用キューに送信しようとしていないか見直してください。
55	HRM-00816	The writing of property failed because the property has been read-only mode.	プロパティが読み取り専用モードであったため，書き込みに失敗しました。	メッセージの書き込み手順を見直してください。
56	HRM-00817	The reading of payload failed because the payload has been write-only mode.	ペイロードが書き込み専用モードであったため，読み取りに失敗しました。	メッセージの読み取り手順を見直してください。
57	HRM-00818	The writing of payload failed because the payload has been read-only mode.	ペイロードが読み取り専用モードであったため，書き込みに失敗しました。	メッセージの書き込み手順を見直してください。
58	HRM-00819	An invalid queue was specified.	無効なキューが指定されました。	引数を見直してください。

項番	障害コード	障害文字列	発生要因	対処
59	HRM-00820	An invalid message selector was specified.	無効なメッセージセレクタが指定されました。	メッセージセレクタの値を見直し，正常なメッセージセレクタを設定し直してください。
60	HRM-00822	An internal error occurred.	内部エラーが発生しました。	この障害の発生前後に出力されたメッセージを参照し，該当する障害要因を取り除いてください。障害要因を取り除けない場合は，メッセージログおよびトレースを採取し，保守員に連絡してください。
61	HRM-00823	The method was executed in an inappropriate state.	不適切な状態でメソッドを実行しました。	メソッドの発行手順を見直してください。
62	HRM-00824	The session transaction is in a non-running state.	セッションのトランザクションが開始していない状態です。	メソッドの発行手順を見直してください。
63	HRM-00825	The session transaction was in a running state.	セッションのトランザクションが開始している状態です。	メソッドの発行手順を見直してください。
64	HRM-00826	The transaction was in a running state, due to the TM.	トランザクションマネジャでのトランザクションが開始している状態です。	メソッドの発行手順を見直してください。
65	HRM-00827	The transaction was in a suspended state, due to the TM.	トランザクションマネジャでのトランザクションがサスペンドしている状態です。	メソッドの発行手順を見直してください。
66	HRM-00828	A memory shortage occurred.	メモリ不足が発生しました。	利用できるメモリを増やし，処理を再度実行してください。
67	HRM-00829	An invalid message selector was specified.	無効なメッセージセレクタが指定されました。	メッセージセレクタの値を見直し，正常なメッセージセレクタを設定し直してください。
68	HRM-00830	Invalid ContentType is set to BytesContainer.	BytesContainer が保持する ContentType の値が不適切だったため，メソッドの実行に失敗しました。	BytesContainer が保持する ContentType の値が，この製品でサポートしている値かどうかを見直してください。
69	HRM-00831	The combination of payload and ContentType set to BytesContainer is illegal.	BytesContainer が保持する ContentType とペイロード内容の組み合わせが不正だったため，メソッドの実行に失敗しました。	BytesContainer が保持する ContentType とペイロード内容を適切な組み合わせにしてください。

項番	障害コード	障害文字列	発生要因	対処
70	HRM-00832	The method failed because an attempt was made to re-use a message that was passed by reference.	参照渡しのメッセージを再利用しようとしたため，メソッドの実行に失敗しました。	メソッドの発行手順を見直してください。
71	HRM-00833	An attempt to acquire a connection has failed.	コネクションの取得に失敗しました。	この障害の発生前後に出力されたメッセージやコネクション取得に関する設定情報を参照し，障害要因を取り除いてください。
72	HRM-00901	An I/O exception occurred.	IOException が発生しました。	入出力エラーの障害要因を取り除いてください。
73	HRM-00902	The value of the configuration property for database connection is invalid.	DB に接続するためのコンフィグレーションプロパティの値が不正です。	コンフィグレーションプロパティを修正してください。
74	HRM-00903	An attempt to register queue information has failed.	キュー情報の登録に失敗しました。	メッセージの詳細から DB の障害を回復してください。
75	HRM-00904	Table creation failed.	テーブルの作成に失敗しました。	メッセージの詳細から DB の障害を回復してください。
76	HRM-00905	An attempt to register FIFO information has failed.	FIFO 情報の登録に失敗しました。	メッセージの詳細から DB の障害を回復してください。
77	HRM-00906	Table initialization failed.	テーブルの初期化に失敗しました。	メッセージの詳細から DB の障害を回復してください。
78	HRM-00907	An attempt to delete queue information has failed.	キュー情報の削除に失敗しました。	メッセージの詳細から DB の障害を回復してください。
79	HRM-00908	An attempt to delete FIFO information has failed.	FIFO 情報の削除に失敗しました。	メッセージの詳細から DB の障害を回復してください。
80	HRM-00909	An attempt to delete a table has failed.	テーブルの削除に失敗しました。	メッセージの詳細から DB の障害を回復してください。
81	HRM-00910	An attempt to lock a table has failed.	テーブルのロックに失敗しました。	メッセージの詳細から DB の障害を回復してください。
82	HRM-00912	You do not have access permission.	アクセス権限がありません。	policy ファイルを見直してください。次に示す状況が考えられます。 スレッドまたはソケットに関するアクセス権限を設定していません。

項番	障害コード	障害文字列	発生要因	対処
83	HRM-00913	An attempt to acquire system management information has failed.	システム管理情報の取得に失敗しました。	SQL ファイルの実行が失敗していないかどうかを確認し，必要に応じて SQL ファイルを再度実行してください。
84	HRM-00914	An attempt to acquire the IP address used for the group ID has failed.	グループ ID に使用する IP アドレスの取得に失敗しました。	マシンに IP アドレスが指定されているか確認します。
85	HRM-01001	An attempt was made to create a shared queue in Oracle.	共用キューを Oracle に作成しようとしました。	接続先の DB を HiRDB に変更してください。または，ローカルキューを作成してください。
86	HRM-01101	Failed to check content in BytesContainer.	BytesContainer が保持する ContentType と Content の組み合わせチェック処理に失敗しました。	メッセージログおよびトレースを採取し，保守員に連絡してください。

8 コマンドリファレンス

ユーザはコマンドを入力することによって Reliable Messaging を運用します。

この章では，Reliable Messaging が提供するコマンドについて説明します。

8.1　コマンドの概要

Reliable Messaging はシステムを運用するためにコマンドを提供します。

コマンドを実行するための前提条件や複数の Reliable Messaging を動作させる場合のシステム名の設定などについて説明します。

8.1.1　コマンド実行の前提条件

Reliable Messaging のコマンドを実行する場合の前提条件を次に示します。

- HRMDIR，HRM_SYSTEM_NAME，HRM_CMD_HOST，HRM_CMD_PORT および PATH 環境変数を設定する必要があります。各環境変数については，「3.3.2　環境変数の設定」を参照してください。
- Reliable Messaging のコマンドを実行するコンソールでは，PRFSPOOL 環境変数を無効にする必要があります。
- Reliable Messaging の開始が完了している必要があります。Reliable Messaging の開始については，「4.1.1　Reliable Messaging の開始（永続版リソースアダプタの場合）」，または「5.1.1　Reliable Messaging の開始（非永続版リソースアダプタの場合）」を参照してください。
- UNIX の場合は，Reliable Messaging のコマンドを実行するには，root または Component Container 管理ユーザの権限が必要となります。

8.1.2　コマンドの有効範囲

Reliable Messaging のコマンドは，自システムの Reliable Messaging にだけ有効です。対象となる Reliable Messaging と異なるマシンでコマンドを入力する運用はできません。

8.1.3　コマンドの同時実行

複数のコマンドを同時に実行することはできません。同時に実行した場合は，先に実行したコマンドが完了するまで待ち合わせます。

8.1.4　システム名の設定

自システムで複数の Reliable Messaging を動作させる場合は，コマンドの対象となる Reliable Messaging のシステム名を次に示すどちらかの方法で指定します。

●コマンド引数での指定

コマンドを入力するときに，コマンド引数にシステム名を指定する方法です。

●環境変数での指定

コマンドの入力画面で HRM_SYSTEM_NAME 環境変数にシステム名を指定する方法です。指定例を次に示します。

Windows の場合

```
prompt>set HRM_SYSTEM_NAME=HRM
```

UNIX（csh）の場合

```
prompt>setenv HRM_SYSTEM_NAME HRM
```

UNIX（sh）の場合

```
prompt>HRM_SYSTEM_NAME=HRM
prompt>export HRM_SYSTEM_NAME
```

コマンド引数と環境変数の両方にシステム名を指定する場合は，コマンド引数に指定したシステム名が有効になります。システム名は RMSystemName プロパティ指定値です。

8.1.5 コマンドの記述形式

Reliable Messaging が提供するコマンドの記述形式について，次に示します。

```
コマンド名　オプション　コマンド引数
```

- コマンド名

  コマンド名は，実行するコマンドのファイル名称です。

  Reliable Messaging が提供するコマンドは%HRMDIR%¥bin にありますので，PATH 環境変数に%HRMDIR%¥bin を追加してください。

- オプション

  次の説明中に使用する「>」はコマンドプロンプト，「cmd」はコマンド名称です。

1. オプションはマイナス記号（-）で始まる文字列で，フラグ引数を取らないか，または 1 個のフラグ引数を取ります。

   オプションの記述形式を次に示します。

   ```
   -オプションフラグ
   または
   -オプションフラグ　フラグ引数
   ```

   オプションフラグは，1 文字の英数字（英大文字と英小文字は区別されます）です。

   フラグ引数はオプションフラグに対する引数です。

2. フラグ引数を取らないオプションフラグは，一つのマイナス記号（-）の後ろにまとめて指定できます。
（例）次の二つは同じ意味です。

```
> cmd -a -b -c
> cmd -abc
```

3. フラグ引数を必要とするオプションフラグのフラグ引数は，省略できません。
4. オプションの指定順序はありません。
5. コマンド名，オプション，コマンド引数の間には一つ以上のスペースを挿入してください。
6. 同じオプションフラグを 2 回以上指定すると，最後に指定したものが有効になります。
（例）
次に示すとおり入力すると，フラグ引数として 2 が有効になります。

```
> cmd -a 1 -a 2
      無効 有効
```

7. オプションは，コマンド引数より前に指定しなければなりません。
（例）
オプションフラグ a がフラグ引数を取らない場合，次に示すとおり入力すると，file と-b は，コマンド引数とみなされます。

```
> cmd -a file -b
```

8. 二つのマイナス記号（-）は，オプションの終わりを示します。
（例）
次に示すとおり入力すると，-b はコマンド引数とみなされます。

```
> cmd -a -- -b
```

9. マイナス記号（-）だけのオプションは入力できません。
（例）
次に示すとおり入力すると，-はコマンド引数とみなされます。

```
> cmd -
```

- コマンド引数
コマンド引数は，コマンド操作の対象になるものを指定します。複数のコマンド引数がある場合は，各コマンドの指定形式に示される順序に従ってください。

8.1.6 リクエストタイムアウト値の変更

ユーザによって実行されたコマンドは，Reliable Messaging にリクエストを送信し，応答を待ちます。コマンドが応答を受け取れないときのリクエストタイムアウト値（単位：秒）について，ユーザは独自の値を設定できます。

Reliable Messaging のコマンドごとに，%HRMDIR%¥bin ディレクトリにバッチファイル（Windows の場合）またはシェルスクリプト（UNIX の場合）が提供されています。設定を変更する場合は，各コマンドのバッチファイルまたはシェルスクリプトを編集して次に示す行の右辺の数値を変更してください。

Windowsの場合

```
set PROPS=%PROPS% -Dejbserver.rmi.request.timeout=0
```

UNIXの場合

```
set PROPS="${PROPS} -Dejbserver.rmi.request.timeout=0"
```

0～86400（単位：秒）の値を指定できます。デフォルトでは 0 が設定されています。

値を設定するときは，次に示す点に注意してください。

- 値が小さいとコマンド入力直後にリクエストタイムアウトとなり，コマンドを正常に実行できないことがあります。
- 0 を指定するとタイムアウトしなくなります。コマンド実行時に長時間レスポンスがない場合は，強制的にコマンド処理を中断してください。Ctrl キーと C キーを同時に押します。
- Reliable Messaging のバージョンアップや修正パッチを適用した場合，リクエストタイムアウト値は 0 で上書きされます。リクエストタイムアウト値を変更している場合は，値を設定し直してください。

8.1.7　コマンド実行時に出力されるログ

Reliable Messaging のコマンドは，Application Server の JNDI を利用するため，コマンドを実行すると，%HRMDIR%¥logs¥cmd 以下に Application Server のクライアントのログと TPBroker の通信トレースファイルを出力します。

8.1.8　UNIX でコマンドを実行する場合の注意事項

UNIX の場合は，次のどちらかの作業を実施してから Reliable Messaging のコマンドを実行するようにしてください。

- umask を 0 に設定する
- $HRMDIR/logs 以下のファイルおよびディレクトリのアクセス権を，グループやほかのすべてのユーザに対して書き込みを許可するように設定する（root でコマンドを実行したあとに，Component Container 管理ユーザでコマンドを実行する場合）

どちらの作業も実施しないまま，別のユーザがコマンドを実行すると，コマンドの実行がエラーになったり，コマンドのログが不正に残ったりする場合があります。

8.2 コマンドの一覧

Reliable Messaging が提供するコマンドの一覧を次の表に示します。

表 8-1 コマンドの一覧

項番	コマンド名称	機能	開始中状態で実行		実行状態で実行		管理状態で実行	閉塞状態で実行
			永続版	非永続版	永続版	非永続版	永続版	
1	hrmchgaddr	あて先変更	×	×	×	×	○	×
2	hrmchgque	• ローカルキューの属性変更 • 受信用共用キューの属性変更 • 送信用共用キューの属性変更 • 転送キューの属性変更	×	×	×	×	○※	×
3	hrmdeladdr	あて先削除	×	×	×	×	○	×
4	hrmdelmsg	メッセージの削除	×	×	○ (-f オプション指定時)	○	○	×
5	hrmdelque	キューの削除	×	×	×	×	○※	×
6	hrmlsaddr	あて先表示	×	×	○	×	○	×
7	hrmlsdmsg	デッドメッセージの参照	×	×	○	×	○	×
8	hrmlsmsg	メッセージの表示	×	×	○	○	○	×
9	hrmlsque	キュー情報の表示	×	×	○	○	○	×
10	hrmlsstat	システム状態の表示	○	○	○	○	○	○
11	hrmlstrn	トランザクション状態の表示	×	×	○	○	×	×
12	hrmlstrs	通信状態表示	×	×	○	×	○	×
13	hrmmkaddr	あて先登録	×	×	×	×	○	×
14	hrmmkque	• ローカルキューの作成 • 受信用共用キューの作成	×	×	×	×	○	×

項番	コマンド名称	機能	開始中状態で実行		実行状態で実行		管理状態で実行	閉塞状態で実行
			永続版	非永続版	永続版	非永続版	永続版	
		• 送信用共用キューの作成 • 転送キューの作成						
15	hrmregdmsg	デッドメッセージの再登録	×	×	○	×	○	×
16	hrmskipmsg	受信待ちメッセージのスキップ	×	×	○	×	○	×
17	hrmstart	実行状態への移行	×	×	×	×	○	×
18	hrmstartque	キューの抑止解除	×	×	○	○	○	×
19	hrmstarttrs	送受信抑止解除	×	×	○	×	○	×
20	hrmstop	管理状態への移行	×	×	○	×	×	×
21	hrmstopque	キューの抑止	×	×	○	○	○	×
22	hrmstoptrs	送受信抑止	×	×	○	×	○	×

（凡例）

○：実行できます。

×：実行できません。

注※

初回開始時の場合，または管理状態からの再開始後に実行状態に移行していない管理状態の場合だけ実行できます。

8.3　コマンドの詳細

Reliable Messaging が提供するコマンドの詳細を名前のアルファベット順に説明します。

8.3.1　hrmchgaddr（あて先変更）

(1)　形式

```
hrmchgaddr 〔-u　あて先アドレス〕　〔-i ユーザID -p パスワード〕
           〔-S　システム名〕　あて先名
```

(2)　機能

あて先情報テーブルに登録したあて先情報を変更します。

(3)　オプション

-u あて先アドレス

～＜1～512 文字の文字列＞

メッセージを転送するあて先アドレスを指定します。

-i ユーザ ID

～＜1～16 文字の英数字＞

BASIC 認証のためのユーザ ID を指定します。

このオプションは-p オプションと同時に指定してください。このオプションを指定して-p オプションを指定していない場合は，エラーが発生します。

-p パスワード

～＜1～16 文字の英数字＞

BASIC 認証のためのパスワードを指定します。

このオプションは-i オプションと同時に指定してください。このオプションを指定して-i オプションを指定していない場合は，エラーが発生します。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンド操作対象となるシステムのシステム名（プロパティの RMSystemName）を指定します。

指定を省略した場合は，環境変数に設定されたシステム名を指定したものとみなします。

環境変数にもシステム名が設定されていない場合はエラーが発生します。

(4) コマンド引数

あて先名

～＜1～32 文字の文字列＞

あて先情報に対応する論理名を指定します。

指定したあて先名が存在しない場合はエラーが発生します。また，キューにあて先名が指定されている場合は，そのキュー内のメッセージが転送される前だけ変更できます。

(5) 注意事項

- 各オプションを省略した場合，その情報は変更しないものとみなされます。すべてのオプションを省略した場合，情報の変更は行わないで，そのままコマンド処理が終了します。
- あて先アドレスは，URL の記載ミスなどで存在しないあて先アドレスを指定した場合にかぎり，変更できます。実在するあて先アドレスを指定していてほかのあて先アドレスに変更した場合，メッセージが消滅するおそれがあります。
- ユーザ ID およびパスワードを削除する変更はできません。ユーザ ID およびパスワードを削除したい場合は，hrmdeladdr コマンドであて先を削除し，hrmmkaddr コマンドでユーザ ID およびパスワードを指定しないで，再度あて先を登録してください。
- -u オプションで，自システムのキュー間転送用 Web アプリケーションの URL を指定しないでください。
- -p オプションにパスワードを指定した場合，このコマンドの実行中に，プロセスの引数を確認できる OS 機能などでパスワードが観測されるおそれがあります。-p オプションを使用する際は，他ユーザがプロセスの引数を確認できる OS 機能などを使用できない状況下で，このコマンドを実施してください。

8.3.2 hrmchgque（ローカルキューの属性変更）

(1) 形式

```
hrmchgque 〔-d {serial | parallel | parallel_unit_order}〕
          〔-n 最大メッセージ数〕
          〔-c キャッシュメッセージ数〕 〔-e メッセージ有効期間〕
          〔-w {sender | receiver}〕 〔-x 表示名〕
          〔-S システム名〕 キュー名
```

(2) 機能

作成済みのローカルキューの属性を変更します。

非永続版リソースアダプタの場合，キューの属性はキュー作成ファイルの定義によって変更されます。詳細については，「3.5.1 キュー作成ファイルの作成」を参照してください。

(3) オプション

-d {serial | parallel | parallel_unit_order}

ローカルキューのメッセージ取り出しモードを指定します。

- serial：シリアル取り出し属性
- parallel：パラレル取り出し属性
- parallel_unit_order：パラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）

キューの永続属性を persistent に設定したローカルキューに対して，-d オプションで parallel_unit_order を指定した場合，エラーが発生します。

各属性を指定したときのメッセージの処理については，「2.3.2　メッセージ取り出しモード」を参照してください。

-n 最大メッセージ数

～＜数字＞((1～65535))

ローカルキューに格納するメッセージの最大数を指定します。

-c キャッシュメッセージ数

～＜数字＞((0～65535))

キャッシュに格納するメッセージの数を指定します。

-e メッセージ有効期間

～＜数字＞((0～2592000))（単位：秒）

ローカルキューに格納するメッセージの有効期間を指定します。

0 を指定する場合，メッセージの有効期間は無限です。

有効期間を指定するときのメッセージの処理については，「2.3.5　メッセージの有効期間」を参照してください。

-w {sender | receiver}

転送キューからメッセージを受信する場合，メッセージの有効期間については，送信側の有効期間を使用するのか，または受信側で更新するのかを選択します。転送キューの詳細については，「8.3.20 hrmmkque（転送キューの作成）」を参照してください。

- sender：送信側の有効期間
- receiver：受信側の有効期間

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

-x 表示名

～＜ 1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションがJNDIネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はコマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。詳細は「8.3.2(5)　注意事項」を参照してください。

(4)　コマンド引数

キュー名

～＜1～20文字の識別子＞

属性を変更するローカルキューの名前を指定します。

指定したローカルキューがない場合はエラーが発生します。

(5)　注意事項

- 各オプションを省略する場合，その属性は変更されません。キュー名だけを指定した場合，属性は変更されないでそのままコマンド処理が終了します。
- 最大メッセージ数，キャッシュメッセージ数，メッセージ有効期間を変更できるのは，ローカルキューにメッセージがない場合だけです。メッセージがある場合はエラーが発生します。
- Reliable Messaging初回開始時の管理状態の場合，または管理状態からの再開始後に実行状態に移行していない管理状態の場合だけ実行できます。それ以外の場合はエラーが発生します。ただし，Reliable Messaging起動直後の管理状態でも，相手システムが起動中の場合，エラーが発生するおそれがあります。
- キュー定義ファイルを使用しない場合，変更前の表示名でJNDIネーミングサービスに登録されていたオブジェクトは削除され，-xオプションで指定した変更後の表示名でJNDIネーミングサービスに登録されます。このコマンド実行時に，アプリケーションがJNDIネーミングサービスから該当するキューに対応する値（javax.jms.Queue型のオブジェクト）を取得しているかどうかは，このコマンドの実行結果に影響しません。

8.3.3　hrmchgque（受信用共用キューの属性変更）

(1)　形式

```
hrmchgque 〔-c キャッシュメッセージ数〕 〔-x 表示名〕
          〔-S システム名〕 キュー名
```

(2)　機能

作成済みの受信用共用キューの属性を変更します。

(3) オプション

-c キャッシュメッセージ数

～＜数字＞((0～65535))

キャッシュに格納するメッセージの数を指定します。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

-x 表示名

～＜ 1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はコマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。詳細は「8.3.3(5)　注意事項」を参照してください。

(4) コマンド引数

キュー名

～＜ 1～20 文字の識別子＞

属性を変更する受信用共用キューの名前を指定します。

指定した受信用共用キューがない場合はエラーが発生します。

(5) 注意事項

- 各オプションを省略する場合，その属性は変更されません。キュー名だけを指定した場合，属性は変更されないでそのままコマンド処理が終了します。
- キャッシュメッセージ数を変更できるのは，受信用共用キューにメッセージがない場合だけです。メッセージがある場合はエラーが発生します。
- 受信用共用キューの最大メッセージ長または最大メッセージ数を変更する場合は，hrmlsque コマンドでキュー情報を確認してからキューを削除して作成し直してください。または，異なるキューを作成してください。
- Reliable Messaging 初回開始時の管理状態の場合，または管理状態からの再開始後に実行状態に移行していない管理状態の場合だけ実行できます。それ以外の場合はエラーが発生します。ただし，Reliable Messaging 起動直後の管理状態でも，相手システムが起動中の場合，エラーが発生するおそれがあります。

- キュー定義ファイルを使用していない場合，変更前の表示名で JNDI ネーミングサービスに登録されていたオブジェクトは削除され，-x オプションで指定した変更後の表示名で JNDI ネーミングサービスに登録されます。このコマンド実行時に，アプリケーションが JNDI ネーミングサービスから該当するキューに対応する値（javax.jms.Queue 型のオブジェクト）を取得しているかどうかは，このコマンドの実行結果に影響しません。

8.3.4 hrmchgque（送信用共用キューの属性変更）

(1) 形式

```
hrmchgque 〔-x 表示名〕 〔-S システム名〕 キュー名
```

(2) 機能

作成済みの送信用共用キューの属性を変更します。

(3) オプション

-x 表示名

～＜1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はコマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

属性を変更する送信用共用キューの名前を指定します。

指定した送信用共用キューがない場合はエラーが発生します。

(5) 注意事項

- 各オプションを省略する場合，その属性は変更されません。キュー名だけを指定した場合，属性は変更されないでそのままコマンド処理が終了します。
- Reliable Messaging 初回開始時の管理状態の場合，または管理状態からの再開始後に実行状態に移行していない管理状態の場合だけ実行できます。それ以外のときはエラーが発生します。ただし，Reliable Messaging 起動直後の管理状態でも，相手システムが起動中の場合，エラーが発生するおそれがあります。
- キュー定義ファイルを使用していない場合，変更前の表示名で JNDI ネーミングサービスに登録されていたオブジェクトは削除され，-x オプションで指定した変更後の表示名で JNDI ネーミングサービスに登録されます。このコマンド実行時に，アプリケーションが JNDI ネーミングサービスから該当するキューに対応する値（javax.jms.Queue 型のオブジェクト）を取得しているかどうかは，このコマンドの実行結果に影響しません。

8.3.5 hrmchgque（転送キューの属性変更）

(1) 形式

```
hrmchgque 〔-n 最大メッセージ数〕 〔-c キャッシュメッセージ数〕
          〔-e メッセージ有効期間〕 〔-a あて先名〕
          〔-v 転送先キュー名 | -y〕 〔-i {normal | compatible}〕
          〔-j {exactly_once | in_order}〕
          〔-g 通信層のグループ有効期間〕 〔-s 通信層のメッセージ有効期間〕
          〔-x 表示名〕 〔-S システム名〕 キュー名
```

(2) 機能

作成済みの転送キューの属性を変更します。

(3) オプション

-n 最大メッセージ数

～＜数字＞((1～65535))

転送キューに格納するメッセージの最大数を指定します。

-c キャッシュメッセージ数

～＜数字＞((0～65535))

キャッシュに格納するメッセージの数を指定します。

-e メッセージ有効期間

～＜数字＞((1～2592000))（単位：秒）

転送キューに格納するメッセージの有効期間を指定します。

有効期間を指定するときのメッセージの処理については，「2.3.5　メッセージの有効期間」を参照してください。

-a あて先名

～＜1～32 文字の文字列＞

メッセージを転送するあて先のあて先名を指定します。hrmmkaddr コマンドで，事前にあて先名を登録する必要があります。登録されていないあて先名を指定するとエラーが発生します。

-v 転送先キュー名

～＜1～20 文字の識別子＞

メッセージを転送する場合，転送したメッセージを登録する転送キューのキュー名を指定します。転送キューの詳細については，「8.3.20　hrmmkque（転送キューの作成）」を参照してください。

-y

-v オプションを省略して，-a オプションで指定したあて先名に対応するあて先アドレスだけを送信アドレスとする場合に指定します。転送キューの詳細については，「8.3.20　hrmmkque（転送キューの作成）」を参照してください。

-i {normal ｜ compatible}

作成するキューの転送モードを指定します。

- normal：通常モード
- compatible：互換モード

転送先の Reliable Messaging のバージョンが 01-03 以降の場合は normal を指定します。バージョンが 01-02 以前の場合は compatible を指定します。

normal を指定して作成した転送キューを使用して，01-02 以前の Reliable Messaging にメッセージを転送した場合，ペイロードが空のメッセージとして受信されることがありますので，注意してください。

-j {exactly_once ｜ in_order}

転送に使用する QoS（通信品質）の種別を指定します。

- exactly_once：配送保証および重複防止
- in_order：順序保証

このオプションを in_order に変更する場合，キューモードは永続キューだけ有効となります。それ以外を指定した場合はエラーが発生します。

-g 通信層のグループ有効期間

～＜数字＞((10～2592000))（単位：秒）

通信層のグループの有効期間を指定します。

転送キューの詳細については，「8.3.20　hrmmkque（転送キューの作成）」を参照してください。

-s 通信層のメッセージ有効期間

～＜数字＞((10～2592000))（単位：秒）

通信層のメッセージの有効期間を指定します。

転送キューの詳細については，「8.3.20　hrmmkque（転送キューの作成）」を参照してください。

-x 表示名

～＜1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はコマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。詳細は「8.3.5(5)　注意事項」を参照してください。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4)　コマンド引数

キュー名

～＜1～20 文字の識別子＞

属性を変更する転送キューの名前を指定します。

指定した転送キューがない場合はエラーが発生します。

(5)　注意事項

- 各オプションを省略する場合，その属性は変更されません。キュー名だけを指定した場合，属性は変更されないでそのままコマンド処理が終了します。
- 最大メッセージ数，キャッシュメッセージ数，メッセージ有効期間，あて先名，キューの転送モード，QoS，通信層のグループ有効期間，および通信層のメッセージ有効期間を変更できるのは，転送キューにメッセージがない場合だけです。メッセージがある場合はエラーが発生します。
- 転送先キュー名または-y オプションの指定内容は，次のうちどちらかの状態のときだけ変更できます。
  - キューにメッセージがない状態
  - キューに現存するメッセージが転送される前の状態

  転送済みのメッセージが存在する場合はエラーが発生します。
- Reliable Messaging 初回開始時の管理状態の場合，または管理状態からの再開始後に実行状態に移行していない管理状態の場合だけ実行できます。それ以外の場合はエラーが発生します。ただし，Reliable Messaging 起動直後の管理状態でも，相手システムが起動中の場合，エラーが発生するおそれがあります。
- キュー定義ファイルを使用していない場合，変更前の表示名で JNDI ネーミングサービスに登録されていたオブジェクトは削除され，-x オプションで指定した変更後の表示名で JNDI ネーミングサービスに

登録されます。このコマンド実行時に，アプリケーションがJNDIネーミングサービスから該当するキューに対応する値（javax.jms.Queue型のオブジェクト）を取得しているかどうかは，このコマンドの実行結果に影響しません。

8.3.6 hrmdeladdr（あて先削除）

(1) 形式

```
hrmdeladdr 〔-S システム名〕 あて先名
```

(2) 機能

あて先情報テーブルに登録したあて先情報を削除します。

(3) オプション

-S システム名

～＜先頭が英字の1～3文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemNameプロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

あて先名

～＜1～32文字の文字列＞

削除したいあて先情報のあて先名を指定します。

キューに指定されていないあて先だけ削除できます。

指定したあて先名が存在しない，またはキューに指定されている場合はエラーが発生します。

8.3.7 hrmdelmsg（メッセージの削除）

(1) 形式

```
hrmdelmsg 〔-f〕 {-a | -n メッセージ通番}〔-S システム名〕 キュー名
```

(2) 機能

キュー内のメッセージを削除します。

永続版リソースアダプタの場合，実行状態でこのコマンドを使用するときは-f オプションを指定してください。

非永続版リソースアダプタの場合，実行状態でこのコマンドを使用するときも-f オプションを指定する必要はありません。

(3) オプション

-f

- 永続版リソースアダプタの場合
  Reliable Messaging が実行状態でも強制的にメッセージを削除する場合に指定します。
  実行状態でこのコマンドを使用する場合は，このオプションを指定しないと，エラーが発生します。
- 非永続版リソースアダプタの場合
  非永続版リソースアダプタでは指定しても無視されます。

-a

コマンド引数に指定したキューに格納されている全メッセージを削除する場合に指定します。

-n メッセージ通番

～<数字>((1～65535))

削除するメッセージのメッセージ通番を指定します。

指定するメッセージ通番の上限値は，コマンド引数に指定したキューに定義された最大メッセージ数です。指定したメッセージ通番のメッセージがない場合はエラーが発生します。

- 永続版リソースアダプタの場合
  コマンド引数に指定したキューの種類によって，指定できるメッセージ通番は異なります。詳細については，「8.3.7(5) 注意事項」を参照してください。
- 非永続版リソースアダプタの場合
  非永続版リソースアダプタの場合，キューの種類による通番の違いはありません。

-S システム名

～<先頭が英字の 1～3 文字の大文字英字または数字>

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。
HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

メッセージを削除するキューの名前を指定します。

指定したキューがない場合はエラーが発生します。

(5) 注意事項

- 実行状態でコマンドを実行する場合は，-f オプションを指定してください。ただし，タイミングによってはエラーが発生するおそれがあります。そのため，管理状態での実行をお勧めします。
- 設定できるオプションはキューの種類によって異なります。

  hrmdelmsg コマンドで各キューに設定できるオプションを次の表に示します。

表 8–2 hrmdelmsg コマンドで各キューに設定できるオプション

キューの種類	-f オプション		-a オプション		-n オプション	
	永続版	非永続版	永続版	非永続版	永続版	非永続版
ローカルキュー	○	△※1	○	○	○	○
受信用共用キュー	○	×	○	×	△※2	×
送信用共用キュー※3	×	×	×	×	×	×
転送キュー※4	○	×	○	×	×	×

（凡例）

○：設定できます。

△：設定できます（制限があります）。

×：設定できません（エラーが発生します）。

注※1

非永続版リソースアダプタでは-f オプションを指定しても無視されます（エラーは発生しません）。

注※2

コマンド引数に受信用共用キューを指定する場合，-n オプションで指定できるメッセージ通番は 1 だけです。1 以外を指定したときはエラーが発生します。

注※3

送信用共用キューからはメッセージを削除できません。コマンド引数に送信用共用キューを指定する場合はエラーが発生します。

注※4

キューの種類が転送キューのキュー名を指定し，かつ QoS（通信品質）で順序保証を指定した場合，受信側に未配信メッセージが滞留するときがあります。

8.3.8 hrmdelque（キューの削除）

(1) 形式

```
hrmdelque 〔-f〕 〔-S システム名〕 キュー名
```

(2) 機能

作成済みのキューを削除します。

非永続版リソースアダプタの場合，キュー作成ファイルの定義を削除することでキューが削除されます。キュー作成ファイルの詳細については，「3.5.1 キュー作成ファイルの作成」を参照してください。

(3) オプション

-f

コマンド引数に指定したキューにメッセージがあっても強制的にキューを削除するときに指定します。この場合，キューにある全メッセージを削除してからキューが削除されます。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜ 1～20 文字の識別子＞

削除するキューの名前を指定します。

指定したキューがない場合はエラーが発生します。

(5) 注意事項

- -f オプションを指定しない場合，コマンド引数に指定したキューにメッセージがあるとエラーが発生します。
- 受信用共用キューを削除する場合は送信処理を中断してください。送信処理を中断しないとき，送受信処理またはリカバリ処理でエラーが発生するおそれがあります。
- キュー間転送で使用しているキューを削除する場合，送信側と受信側のキューを両方削除してください。片方のキューだけ削除すると，メッセージが滞留する場合があります。

- 初回開始時の場合，または管理状態からの再開始後に実行状態に移行していない管理状態の場合だけ実行できます。それ以外の場合はエラーが発生します。ただし，Reliable Messaging 起動直後の管理状態でも，相手システムが起動中の場合，エラーが発生するおそれがあります。
- 削除するキューを Message-driven Bean が監視している場合，このコマンドを実行するとエラーが発生します。
- 閉塞状態のキューは，-f オプションを指定しなくてもメッセージの有無に関係なく，削除されます。また，閉塞した原因によって，キューを削除できない場合があります。対処方法については，キューの閉塞時に出力されたメッセージログを参照してください。
- 削除対象の共用キューが他システムから参照されている場合は，該当の共用キューを参照しているすべてのコネクションを切断しておいてください。
- キュー定義ファイルを使用していない場合，このコマンド実行時に，アプリケーションが JNDI ネーミングサービスから該当するキューに対応する値（javax.jms.Queue 型のオブジェクト）を取得しているかどうかに関係なく，JNDI ネーミングサービスに登録されているオブジェクトは削除されます。

8.3.9 hrmlsaddr（あて先表示）

(1) 形式

```
hrmlsaddr {-n | -a あて先名} 〔-S システム名〕
```

(2) 機能

あて先情報テーブルに登録したあて先名一覧，またはあて先情報を表示します。

(3) オプション

-n

登録済みのあて先名一覧と登録済みのあて先数を表示します。あて先情報は表示しません。

-a あて先名

～＜1～32 文字の文字列＞

表示するあて先情報に対応する論理名を指定します。

指定したあて先名が存在しない場合はエラーが発生します。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) 出力形式

- -n オプション指定の場合

```
address name
  aa....aa
     :
  aa....aa
all address count  ＝  bb....bb
```

- -a オプション指定の場合

```
address name  ＝  aa....aa
url  ＝  cc....cc
user id  ＝  dd....dd
password  ＝  ee....ee
```

aa....aa

あて先名（文字列）

bb....bb

全あて先の個数（10 進数）

cc....cc

あて先アドレス（文字列）

dd....dd

BASIC 認証のためのユーザ ID（英数字）

BASIC 認証を使用しない場合，空白で表示されます。

ee....ee

BASIC 認証のためのパスワード（"********"）

BASIC 認証を使用しない場合，空白で表示されます。

8.3.10 hrmlsdmsg（デッドメッセージの参照）

(1) 形式

```
hrmlsdmsg {-i デッドメッセージID | -n メッセージ通番 〔-e 出力メッセージ数〕}
          〔-o 出力メッセージバイト数〕 〔-p〕 〔-S システム名〕
```

(2) 機能

デッドメッセージキュー内のデッドメッセージの情報を表示します。デッドメッセージ情報やメッセージ内容が表示できます。

(3) オプション

-i デッドメッセージ ID

～＜1～39 文字の識別子＞

表示するデッドメッセージのデッドメッセージ ID（DMID）を指定します。

指定した DMID のメッセージがデッドメッセージキュー内にない場合はエラーが発生します。

-n メッセージ通番

～＜数字＞((1～65535))

表示するデッドメッセージのメッセージ通番を指定します。

指定できるメッセージ通番の上限値は，デッドメッセージキューに定義された最大メッセージ数です。

指定したメッセージ通番以降にデッドメッセージがない場合は，エラーが発生します。

-e 出力メッセージ数

～＜数字＞((1～100))《1》

表示するメッセージ数を指定します。-n オプションで指定したメッセージ通番から，このオプションに指定した数のデッドメッセージ情報が表示されます。

指定を省略した場合，一つのデッドメッセージ情報が表示されます。キュー内のメッセージ数よりも大きい値を指定した場合は，キュー内のメッセージの範囲で表示されます。

-o 出力メッセージバイト数

～＜数字＞((0～1024))《0》（単位：バイト）

メッセージの内容を表示する場合に，表示するバイト数を指定します。

指定を省略した場合，または 0 を指定した場合は，メッセージの内容は表示されません。

-p

アプリケーション指定のプロパティを表示する場合に指定します。

JMS 定義のプロパティ，および Reliable Messaging 固有のプロパティは表示されません。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。

HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) 出力形式

```
message number  =  aa....aa
dead message id  =  bb....bb
original queue name  =  cc....cc
original queue type  =  dd....dd
cause  =  ee....ee
dead message timestamp  =  ff....ff
create time  =  gg....gg
correlation id  =  hh....hh※1
```

```
group id  =  ii....ii※1
group seq  =  jj....jj※1
message id  =  kk....kk※1
property※2
 ll....ll  =  mm....mm※2※3
    :
 ll....ll  =  mm....mm※2※3
message type  =  nn....nn
message length  =  oo....oo
message
 00000000[0x pp....pp  pp....pp  pp....pp  pp....pp] qq....qq
                              :
 00000000[0x pp....pp  pp....pp  pp....pp  pp....pp] qq....qq
```

注※1

次の場合は表示されません。

- デッドメッセージ移動元が受信用共用キューの場合
- 転送された BytesContainer メッセージの場合
- メッセージ登録時に設定していない場合

注※2

-p オプションで，プロパティ表示を指定した場合に表示されます。

注※3

次の場合は表示されません。

- デッドメッセージ移動元が受信用共用キューの場合
- BytesContainer メッセージの場合
- アプリケーション指定のプロパティが未設定の場合
- メッセージのユニット識別子のプロパティ（JMS_HITACHI_UnitID）は表示されません。

aa....aa

メッセージ通番（10 進数）

bb....bb

デッドメッセージ ID（文字列）

cc....cc

デッドメッセージキュー移動前のキュー名（文字列）

dd....dd

デッドメッセージキュー移動前のキューの種類

- LOCAL：ローカルキュー
- SHARE_RECEIVE：受信用共用キュー
- TRANSMIT：転送キュー

ee....ee

デッドメッセージキュー移動原因（文字列）

表示文字列は，API 仕様の Reliable Messaging 固有のプロパティでの「JMS_HITACHI_DeadMessageCause」に設定される文字列と同じです。

ff....ff

デッドメッセージキュー移動時間（文字列）

gg....gg

メッセージ生成時刻（文字列）

デッドメッセージキュー移動元が受信用共用キューの場合，または転送された BytesContainer メッセージの場合，****/**/** **:**:**.*** (**********)が表示されます。

hh....hh

JMSCorrelationID（文字列）

ii....ii

JMSXGroupID（文字列）

jj....jj

JMSXGroupSeq（10 進数）

kk....kk

JMSMessageID（文字列）

ll....ll

プロパティ名（文字列）

mm....mm

プロパティ値（文字列）

プロパティの型の文字列表現で表示されます。プロパティ値に null を設定した場合は表示されません。

nn....nn

メッセージ種別

- TEXT_MESSAGE：テキストメッセージ
- BYTE_MESSAGE：バイトメッセージ
- OBJECT_MESSAGE：オブジェクトメッセージ
- OBJECT_MESSAGE(BYTES_CONTAINER)：BytesContainer メッセージ
- MESSAGE：メッセージ

BytesContainer については，「7.8.2　BytesContainer インタフェース」を参照してください。

oo....oo

メッセージ長（10 進数）

メッセージ種別が TEXT_MESSAGE，または BYTE_MESSAGE の場合だけ表示されます。

pp....pp

メッセージ内容（16 進数）

qq....qq

メッセージ内容（文字列）

次に示す文字以外の文字データがメッセージ中にある場合，その文字データはピリオド（.）に変換して出力されます。

- 半角文字の英数字
- 半角文字のパーセント（%），スラント（/），ピリオド（.），アンダースコア（_）および空白（ ）

(5) 注意事項

- 対象のデッドメッセージキューは，RMDeadMessageQueueName プロパティで指定したキューです。
- 次のメッセージは表示されません。
  - アプリケーションで登録したメッセージ
  - Reliable Messaging 01-01 以前にデッドメッセージキューに移動したメッセージ
- 表示されるメッセージ通番は，デッドメッセージキュー内で先頭メッセージから順番に通し番号を振られています。表示されないメッセージ（アプリケーションで登録したメッセージ，Reliable Messaging 01-01 以前にデッドメッセージキューに移動したメッセージ）が混在している場合，表示されるメッセージ通番は連続しないことがあります。
- 表示できるメッセージの内容は，バイトおよびテキスト型のメッセージだけです。

8.3.11 hrmlsmsg（メッセージの表示）

(1) 形式

```
hrmlsmsg -n メッセージ通番 〔-e 出力メッセージ数〕
         〔-o 出力メッセージバイト数〕
         〔-S システム名〕 キュー名
```

(2) 機能

キュー内のメッセージの情報を表示します。メッセージ属性やメッセージ内容が表示されます。また，アプリケーションが取り出せるメッセージの一覧が表示されます。

転送キューの場合は，転送前または転送処理中のメッセージ一覧が表示されます。

(3) オプション

-n メッセージ通番

～<数字>((1～65535))

表示するメッセージのメッセージ通番を指定します。

指定できるメッセージ通番の上限値は，コマンド引数に指定したキューに定義された最大メッセージ数です。指定したメッセージ通番のメッセージがない場合はエラーが発生します。

-e 出力メッセージ数

～<数字>((1～100))《1》

表示するメッセージ数を指定します。-n オプションで指定したメッセージ通番のメッセージを先頭にして，このオプションに指定した数のメッセージが表示されます。

キューにあるメッセージ数よりも大きい値を指定した場合は，キューにある範囲で表示されます。

-o 出力メッセージバイト数

～<数字>((0～1024))《0》（単位：バイト）

メッセージ内容を表示する場合に，表示するバイト数を指定します。

指定を省略した場合，または 0 を指定した場合は，メッセージ内容は表示されません。

-S システム名

～<先頭が英字の 1～3 文字の大文字英字または数字>

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～< 1～20 文字の識別子>

表示するメッセージがあるキューの名前を指定します。

指定したキューがない場合または送信用共用キューを指定した場合は，エラーが発生します。

(5) 出力形式

- ローカルキューを指定する場合（永続版リソースアダプタの場合）

```
message number  =  aa....aa
queue name = bb....bb
original queue name  =  cc....cc※
cause  =  dd....dd※
fifo id = ee....ee
sequence number  =  ff....ff
create time  =  gg....gg
modify time  =  hh....hh
expiry time  =  ii....ii
```

```
entry time  =  jj....jj
delete flag  =  kk....kk
priority  =  ll....ll
correlation id  =  mm....mm
group id  =  nn....nn
group seq  =  oo....oo
message id  =  pp....pp
redelivery flag  =  qq....qq
delivery count  =  rr....rr
message unit id  =  xx....xx
message type  =  ss....ss
message length  =  tt....tt
message
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
                              :
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
```

注※

デッドメッセージキューを指定した場合だけ表示します。

- ローカルキューを指定する場合（非永続版リソースアダプタの場合）

```
message number  =  aa....aa
queue name = bb....bb
fifo id = ee....ee
sequence number  =  ff....ff
create time  =  gg....gg
modify time  =  hh....hh
expiry time  =  ii....ii
entry time  =  jj....jj
delete flag  =  kk....kk
priority  =  ll....ll
correlation id  =  mm....mm
group id  =  nn....nn
group seq  =  oo....oo
message id  =  pp....pp
redelivery flag  =  qq....qq
delivery count  =  rr....rr
message unit id  =  xx....xx
message type  =  ss....ss
message length  =  tt....tt
message
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
                              :
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
```

- 受信用共用キューを指定する場合

```
message number  =  aa....aa
queue name  =  bb....bb
internal message number  =  ww....ww
message length  =  ss....ss
message
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
                              :
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
```

- 転送キューを指定する場合

```
message number  =  aa....aa
queue name  =  bb....bb
fifo id = ee....ee
sequence number  =  ff....ff
create time  =  gg....gg
modify time  =  hh....hh
expiry time  =  ii....ii
entry time  =  jj....jj
delete flag  =  kk....kk
priority  =  ll....ll
correlation id  =  mm....mm
group id  =  nn....nn
group seq  =  oo....oo
message id  =  pp....pp
message type  =  ss....ss
message length  =  tt....tt
message
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
                              :
 00000000[0x uu....uu  uu....uu  uu....uu  uu....uu] vv....vv
```

aa....aa

メッセージ通番（10 進数）

bb....bb

キュー名（文字列）

cc....cc

デッドメッセージキュー移動前キュー名（文字列）

dd....dd

デッドメッセージキュー移動原因（文字列）

表示文字列は，API 仕様の Reliable Messaging 固有のプロパティで，「JMS_HITACHI_DeadMessageCause」に設定される文字列と同じです。

ee....ee

FIFO ID（文字列）

ff....ff

シーケンス番号（10 進数）

gg....gg

メッセージ生成時刻（文字列）

永続版リソースアダプタの場合，受信用共用キューからデッドメッセージキューへ移動されたメッセージのとき，または転送キューからローカルキューへ転送された BytesContainer メッセージのときは，****/**/** **:**:**.*** (**********)が表示されます。

hh....hh

メッセージ更新時刻（文字列）

永続版リソースアダプタの場合，受信用共用キューからデッドメッセージキューへ移動されたメッセージのとき，または転送キューからローカルキューへ転送された BytesContainer メッセージのときは，****/**/** **:**:**.*** (**********)が表示されます。

ii....ii

メッセージ有効期間（文字列）

無限の場合，0 が表示されます。

jj....jj

メッセージ登録時刻（文字列）

kk....kk

メッセージ削除（可否）フラグ

次に示すどちらかが表示されます。

- DELETE：削除できます。
- NO_DELETE：削除不可です。

ll....ll

メッセージ優先度（10 進数）

mm....mm

JMSCorrelationID（文字列）

メッセージ登録時に設定されていない場合，空白が表示されます。

nn....nn

JMSXGroupID（文字列）

メッセージ登録時に設定されていない場合，空白が表示されます。

oo....oo

JMSXGroupSeq（10 進数）

メッセージ登録時に設定されていない場合，空白が表示されます。

pp....pp

JMSMessageID（文字列）

メッセージ登録時に設定されていない場合，空白が表示されます。

qq....qq

メッセージの再配送フラグ

次に示すどちらかが表示されます。

- NON_REDELIVERY：再配送していません。
- REDELIVERY：再配送しました。

rr....rr

メッセージの配送回数（10 進数）

ss....ss

メッセージ種別

- 永続版リソースアダプタの場合

  次に示すどれかが表示されます。

  ・TEXT_MESSAGE：テキストメッセージ

  ・BYTE_MESSAGE：バイトメッセージ

  ・OBJECT_MESSAGE：オブジェクトメッセージ

  ・OBJECT_MESSAGE(BYTES_CONTAINER)：BytesContainer メッセージ

  ・MESSAGE：メッセージ

  BytesContainer については，「7.8.2　BytesContainer インタフェース」を参照してください。

- 非永続版リソースアダプタの場合

  次に示すどれかが表示されます。

  ・TEXT_MESSAGE：テキストメッセージ

  ・BYTE_MESSAGE：バイトメッセージ

  ・OBJECT_MESSAGE：オブジェクトメッセージ

  ・MESSAGE：メッセージ

tt....tt

メッセージ長（10 進数）

メッセージ種別が TEXT_MESSAGE または BYTE_MESSAGE の場合だけ表示されます。

uu....uu

メッセージ内容（16 進数）

vv....vv

メッセージ内容（文字列）

次に示す文字以外の文字データがメッセージ中にある場合，その文字データはピリオド（.）に変換して出力されます。

- 半角文字の英数字
- 半角文字のパーセント（%），スラント（/），ピリオド（.），アンダースコア（_）および空白（ ）

ww....ww

内部メッセージ通番（10 進数）

xx....xx

メッセージのユニット識別子（文字列）

メッセージ識別子を設定しないまたは NULL と設定した場合，空白で表示されます。

デッドメッセージキューに移動したメッセージでは，空白で表示されます。

キューモードが永続の場合や転送機能によってキューに格納されたメッセージでは，メッセージに設定したユニット識別子は表示されます。

(6) 注意事項

- 表示できるメッセージの内容は，バイトおよびテキスト型のメッセージだけです。
- BytesContainer メッセージをローカルキューに登録した場合，ローカルキューに登録したときの情報が表示されます。
- 非永続版リソースアダプタの場合，ローカルキュー内のメッセージ情報だけが表示されます。

8.3.12 hrmlsque（キュー情報の表示）

(1) 形式

```
hrmlsque {-n | -q キュー名} 〔-S システム名〕
```

(2) 機能

作成済みのキュー名一覧および作成済みのキュー情報を表示します。

(3) オプション

-n

作成済みのキュー名一覧とキューの数が出力されます。キュー情報は出力されません。

-q キュー名

～＜1～20 文字の識別子＞

キュー情報を表示するキューの名前を指定します。

非永続版リソースアダプタの場合は，ローカルキューだけ指定できます。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) 出力形式

- -n オプションを指定する場合

```
queue name
  aa....aa
     :
  aa....aa
all queue count  =  yy....yy
```

- -q オプションにキュー名を指定する場合

ローカルキューのとき

```
queue name  =  aa....aa
display name  =  NN....NN
queue mode  =  bb....bb
queue type  =  cc....cc
create date  =  dd....dd
modify date  =  ee....ee
ap delivery mode  =  ff....ff
max message number  =  hh....hh
max cache message number  =  ii....ii
expiry time  =  jj....jj
local fifo id  =  kk....kk
queue state  =  ll....ll
message count  =  uu....uu
delivered message count  =  vv....vv
total message count  =  ww....ww
cache message count  =  xx....xx
expiry time pattern  =  BB....BB
```

受信用共用キューのとき

```
queue name  =  aa....aa
display name  =  NN....NN
queue mode  =  bb....bb
queue type  =  cc....cc
share queue version  =  MM....MM
create date  =  dd....dd
modify date  =  ee....ee
ap delivery mode  =  ff....ff
max message size  =  gg....gg
max message number  =  hh....hh
max cache message number  =  ii....ii
local fifo id  =  kk....kk
queue state  =  ll....ll
message count  =  uu....uu
cache message count  =  xx....xx
```

送信用共用キューのとき

```
queue name  =  aa....aa
display name  =  NN....NN
queue mode  =  bb....bb
queue type  =  cc....cc
share queue version  =  MM....MM
create date  =  dd....dd
modify date  =  ee....ee
ap delivery mode  =  ff....ff
max message size  =  gg....gg
local fifo id  =  kk....kk
queue state  =  ll....ll
share queue host  =  mm....mm
share queue port  =  nn....nn
share queue name  =  oo....oo
```

転送キューのとき

```
queue name  =  aa....aa
display name  =  NN....NN
queue mode  =  bb....bb
queue type  =  cc....cc
create date  =  dd....dd
modify date  =  ee....ee
max message number  =  hh....hh
max cache message number  =  ii....ii
expiry time  =  jj....jj
local fifo id  =  kk....kk
queue state  =  ll....ll
message count  =  uu....uu
delivered message count  =  vv....vv
total message count  =  ww....ww
cache message count  =  xx....xx
address name  =  CC....CC
url  =  DD....DD
address queue name  =  EE....EE
user id  =  FF....FF
password  =  GG....GG
transmission mode  =  HH....HH
qos  =  II....II
transmission group expiry time  =  KK....KK
transmission expiry time  =  LL....LL
```

aa....aa

キュー名（文字列）

bb....bb

キューモード

- 永続版リソースアダプタの場合

  次に示すどちらかが表示されます。

  ・PERSISTENT：永続キュー

  ・NON_PERSISTENT：非永続キュー

  受信用共用キューおよび送信用共用キューの場合は PERSISTENT が表示されます。

- 非永続版リソースアダプタの場合

  ・NON_PERSISTENT：非永続キュー

cc....cc

キュータイプ

- 永続版リソースアダプタの場合

  次に示すどれかが表示されます。

  ・LOCAL：ローカルキュー

  ・SHARE_RECEIVE：受信用共用キュー

  ・SHARE_SEND：送信用共用キュー

  ・TRANSMIT：転送キュー

- 非永続版リソースアダプタの場合
  ・LOCAL：ローカルキュー

dd....dd

- 永続版リソースアダプタの場合
  キューを作成した日付（文字列）
- 非永続版リソースアダプタの場合
  キュー作成ファイルを読み込んだときの日付が表示されます。キュー作成ファイルについては，「3.5.1　キュー作成ファイルの作成」を参照してください。

ee....ee

- 永続版リソースアダプタの場合
  キューを更新した日付（文字列）
- 非永続版リソースアダプタの場合
  キューを作成した日付（dd....dd）と同じ値が表示されます。

ff....ff

メッセージ取り出しモード
次に示すどちらかが表示されます。

- SERIAL：シリアル取り出し属性
- PARALLEL：パラレル取り出し属性
- PARALLEL_UNIT_ORDER：パラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）

永続版リソースアダプタの受信用共用キューおよび送信用共用キューの場合は SERIAL が表示されます。

gg....gg

最大メッセージ長（10 進数）

hh....hh

最大メッセージ数（10 進数）

ii....ii

- 永続版リソースアダプタの場合
  最大キャッシュメッセージ数（10 進数）
- 非永続版リソースアダプタの場合
  最大メッセージ数（hh....hh）と同じ値が表示されます。

jj....jj

メッセージの有効期間（10 進数）

kk....kk

ローカル書き込み用の FIFO ID（文字列）
FIFO ID は管理情報としてシステム内部で使用しているキュー情報です。

ll....ll

キューの状態（文字列）

- 永続版リソースアダプタの場合

  次に示すどれかが表示されます。

  ・NORMAL：正常状態

  ・AP_SEND_DISABLED：メッセージの送信抑止状態

  ・AP_RECEIVE_DISABLED：メッセージの受信抑止状態

  ・TRS_SEND_DISABLED：キュー間転送メッセージの送信抑止状態

  ・TRS_RECEIVE_DISABLED：キュー間転送メッセージの受信抑止状態

  ・BLOCKED：閉塞状態

  複数の抑止をしている場合は，これらがすべて表示されます。閉塞状態のキューは，hrmlsque コマンドと hrmdelque コマンド以外は受け付けません。また，キューが閉塞状態にある場合に出力される形式は異なります。詳細は「8.3.12(5)　注意事項」を参照してください。

- 非永続版リソースアダプタの場合

  次に示すどれかが表示されます。

  ・NORMAL：正常状態

  ・AP_SEND_DISABLED：メッセージの送信抑止状態

  ・AP_RECEIVE_DISABLED：メッセージの受信抑止状態

  複数の抑止をしている場合は，これらがすべて表示されます。

mm....mm

共用キューを使用して複数システム間でのアプリケーション連携をする場合のイベント送信先ホスト名または IP アドレス（文字列）

nn....nn

共用キューを使用して複数システム間でのアプリケーション連携をする場合のイベント送信先ポート番号（10 進数）

oo....oo

共用キューを使用して複数システム間でのアプリケーション連携をする場合の登録先キュー名（文字列）

uu....uu

仕掛かり中メッセージ数（10 進数）

未配信メッセージの数です。有効期限切れメッセージおよび受信処理中メッセージの数を含みます。

転送キューの場合，転送待ちメッセージの数および転送処理中メッセージの数も含みます。

vv....vv

配送済みメッセージ数（10 進数）

ローカルキューの場合，hrmdelmsg コマンドによって削除されたメッセージなども数に含みます。

転送キューの場合，Acknowledgment 受信済みメッセージの数が表示されます。

ww....ww

総メッセージ数（10 進数）

仕掛かり中メッセージの数，配信済みメッセージの数および滞留メッセージの数の和です。

転送キューの場合，滞留メッセージの数は常に 0 となります。

xx....xx

キャッシュに格納されているメッセージ数（10 進数）

ローカルキューの場合，キャッシュに格納されている滞留メッセージの数も含みます。

yy....yy

全キューの個数（10 進数）

BB....BB

メッセージ有効期間の選択

- 永続版リソースアダプタの場合
  次に示すどちらかが表示されます。
  ・SENDER：送信側の有効期間
  ・RECEIVER：受信側の有効期間
- 非永続版リソースアダプタの場合
  「***」が表示されます。

CC....CC

あて先名（文字列）

DD....DD

あて先アドレス（文字列）

EE....EE

転送先キュー名 （識別子）

転送先キューを使用しない場合，空白で表示されます。

FF....FF

BASIC 認証のためのユーザ ID（英数字）

BASIC 認証を使用しない場合，空白で表示されます。

GG....GG

BASIC 認証のためのパスワード（"********"）

BASIC 認証を使用しない場合，空白で表示されます。

HH....HH

転送モード

永続版リソースアダプタの転送キューの場合，次に示すどちらかが表示されます。

- NORMAL：通常モード
- COMPATIBLE：互換モード

II....II

QoS（通信品質）の種別

次に示すどちらかが表示されます。

- EXACTLY_ONCE：配送保証および重複防止
- IN_ORDER：順序保証

KK....KK

通信層のグループ有効期間（10 進数）

LL....LL

通信層のメッセージ有効期間（10 進数）

MM....MM

共用キューのバージョン

送信用共用キューの場合は，対応する受信用共用キューのバージョンを表示します。

NN....NN

キューに対応する表示名（文字列）

キュー定義ファイルの使用有無に関係なく出力します。表示名が指定されていない場合は，キュー名称が表示名として出力されます。

(5) 注意事項

- キューが閉塞状態であった場合，次の形式で出力されます。

```
queue name  =  aa....aa
queue type  =  cc....cc
queue state  =  BLOCKED
```

Reliable Messaging 起動時に，キューの種類の値不正によってキューが閉塞した場合は，queue type には"UNKNOWN"が出力されます。キューの閉塞状態および閉塞した場合の対処方法については「9.4　キューの障害」，またはキューが閉塞するときに出力されるメッセージログを参照してください。

非永続版リソースアダプタではキューは閉塞しないため，"BLOCKED"は表示されません。

- 送信用共用キューの参照時に，メッセージの送信先のシステムが停止中などでイベントが送信できない場合，share queue host と share queue port の値は空白で表示されることがあります。
- 送信用共用キューの参照時に，対応する受信用共用キューが接続先の DB 上に存在しない場合，エラーが発生します。
- 非永続版リソースアダプタの場合，ローカルキューについてのキュー情報だけが表示されます。

8.3.13　hrmlsstat（システム状態の表示）

(1)　形式

```
hrmlsstat 〔-S システム名〕
```

(2)　機能

Reliable Messaging の内部状態を表示します。Reliable Messaging の内部状態については，次の個所を参照してください。

- 永続版リソースアダプタの場合「4.1.4　Reliable Messaging の状態遷移（永続版リソースアダプタの場合）」
- 非永続版リソースアダプタの場合「5.1.4　Reliable Messaging の状態遷移（非永続版リソースアダプタの場合）」

(3)　オプション

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4)　出力形式

```
system status　＝　aa....aa
```

aa....aa

システム状態が表示されます。

次に示すどれかが表示されます。

- 永続版リソースアダプタの場合
  - ・EXECUTED_STATE：実行状態
  - ・MANAGED_STATE：管理状態
  - ・BLOCKADE_STATE：閉塞状態
  - ・CREATED_STATE：開始中状態
- 非永続版リソースアダプタの場合
  - ・EXECUTED_STATE：実行状態
  - ・BLOCKADE_STATE：閉塞状態
  - ・CREATED_STATE：開始中状態

8.3.14　hrmlstrn（トランザクション状態の表示）

(1)　形式

```
hrmlstrn 〔-S システム名〕
```

(2)　機能

トランザクションの状態を表示します。

(3)　オプション

-S システム名

～<先頭が英字の 1～3 文字の大文字英字または数字>

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。

HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4)　出力形式

- ローカルトランザクションの場合

```
timestamp ＝ aa....aa
transaction id
  bb....bb
transaction state ＝ cc....cc
```

- トランザクションマネジャでのグローバルトランザクションの場合

```
timestamp ＝ aa....aa
transaction id
   XID(dd....dd,ee....ee)
transaction state ＝ cc....cc
```

aa....aa

トランザクション開始時刻（文字列）

YYYY/MM/DD hh:mm:ss の形式で表示されます。不明の場合はアスタリスク（*）が表示されます。

- YYYY：西暦
- MM：月
- DD：日
- hh：時間
- mm：分
- ss：秒

bb....bb

ローカルトランザクション識別子（文字列）

cc....cc

トランザクションの状態（文字列）

次に示すどれかが表示されます。

- 永続版リソースアダプタの場合
  - ・NON_EXISTENT：初期状態
  - ・ACTIVE：トランザクション処理中
  - ・IDLE：トランザクション指示待ち
  - ・PREPARED：トランザクション決着指示待ち
  - ・ROLLBACK_ONLY：ロールバック指示待ち
  - ・HEURISTIC_COMPLETED（詳細情報）：ヒューリスティック決着済み

    詳細情報として，次に示すどちらかが表示されます。
  - ・HEURCOM：ヒューリスティックコミット
  - ・HEURRB：ヒューリスティックロールバック
  - ・PREPARED_OR_HEURISTIC：トランザクション決着指示待ちまたは

    ヒューリスティック決着済み
  - ・LOCAL_ACTIVE：ローカルトランザクション処理中
- 非永続版リソースアダプタの場合
  - ・LOCAL_ACTIVE：ローカルトランザクション処理中

次に示すオプションは永続版リソースアダプタでだけ使用できます。

dd....dd

グローバルトランザクション識別子（文字列）

ee....ee

ブランチ識別子（文字列）

(5) 注意事項

- アプリケーションが使用するトランザクションの状態だけが表示されます。Reliable Messaging が内部で使用するトランザクションの状態は表示されません。
- 非永続版リソースアダプタの場合，ローカルトランザクションの状態だけ表示されます。

8.3.15 hrmlstrs（通信状態表示）

(1) 形式

```
hrmlstrs {-n | -g グループID} 〔-m 通信層グループ内メッセージ通番〕
         〔-e 出力メッセージ数〕〔-S システム名〕 キュー名
```

(2) 機能

メッセージ転送中に通信層のグループの状態やグループ内のメッセージ通信状態を表示します。

(3) オプション

-n

キューに存在する通信層のグループ ID 一覧と送信用/受信用の種別，グループの有効/無効を表示します。

-g グループ ID

～＜1～256 文字の文字列＞

通信状態を参照するグループのグループ ID を指定します。

グループが無効の場合はグループの参照はできません。グループが無効の場合にこのオプションを指定するとエラーが発生します。

-m 通信層グループ内メッセージ通番

～＜数字＞((1～65535))

出力するメッセージのメッセージ通番を指定します。指定するメッセージ通番の上限値は，指定したキューに定義された最大メッセージ数となります。

指定したメッセージ通番のメッセージが指定したキュー内に存在しない場合はエラーが発生します。-g オプションを指定していて，さらに指定したグループが有効の場合だけ指定できます。-g オプションを指定しない，または指定したグループが無効であった場合，このオプションを指定するとエラーが発生します。

-e 出力メッセージ数

～＜数字＞((1～100))《1》

メッセージ情報を表示するメッセージ数を指定します。-m オプションで指定したメッセージ通番から，このオプションで指定された出力メッセージ数のメッセージ情報を表示します。指定を省略した場合，1 メッセージ情報を表示します。

組み込まれているメッセージ数より大きい値を指定した場合は，組み込まれているメッセージ数までを表示します。

-g と-m オプションを両方指定したときだけ，このオプションは有効です。それ以外の場合はエラーが発生します。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

状態を参照するキューの名前を指定します。

指定したキューが存在しない，またはローカルキュー，転送キュー以外のキューを指定した場合はエラーが発生します。

(5) 注意事項

表示されるメッセージ通番は通信層のグループ内の通番です。hrmlsmsg コマンドで表示される message number と異なります。また，通信層グループ内メッセージ有効期限に到達したメッセージは出力対象になりません。

(6) 出力形式

- -n オプション指定の場合

```
transmission group id : group type : group state
  aa....aa: bb....bb: cc....cc
     :         :         :
  aa....aa: bb....bb: cc....cc
all group count = dd....dd
```

- -g オプション グループ ID 指定の場合

  受信用グループのとき

```
transmission group id  =  aa....aa
qos  =  ee....ee
non received messages  =  ff....ff....ff....ff  (FF....FF)
skipped messages  =  gg....gg....gg....gg  (GG....GG)
transmission group expiry time  =  hh....hh
```

  送信用グループのとき

```
transmission group id  =  aa....aa
qos  =  ee....ee
delivering messages  =  ii....ii....ii....ii (II....II)
transmission group expiry time  =  hh....hh
```

- -g オプション グループ ID　-m オプション 1 指定の場合

```
transmission group message number  =  jj....jj
transmission sequence number = kk....kk
message state  =  ll....ll
correlation id  =  mm....mm
group id  =  nn....nn
group seq  =  oo....oo
message id  =  pp....pp
transmission expiry time  =  qq....qq
```

aa....aa

通信層のグループ ID（文字列）

bb....bb

グループの種別

次に示すどちらかが表示されます。

- SEND：送信用グループ
- RECEIVE：受信用グループ

cc....cc

グループの状態

次に示すどちらかが表示されます。

- NOT_CLOSED：有効
- CLOSED：無効

dd....dd

グループの総数（10 進数）

ee....ee

QoS（通信品質）の種別

次に示すどちらかが表示されます。

- EXACTLY_ONCE：配送保証および重複防止
- IN_ORDER：順序保証

ff....ff (FF....FF)

受信待ち（未受信）シーケンス番号（受信待ちメッセージ総数）（10 進数）

受信済みシーケンス番号の最大値未満のシーケンス番号で最大 10 メッセージが表示されます。

gg....gg (GG....GG)

スキップ済みシーケンス番号（スキップ済みメッセージ総数）（10 進数）

受信待ちスキップコマンドでスキップされたシーケンス番号で最大 10 メッセージが表示されます。

hh....hh

通信層グループ有効期限（文字列）

ii....ii (II....II)

転送中シーケンス番号（転送中メッセージ総数）（10 進数）

最大 10 メッセージが表示されます。

jj....jj

通信層グループ内メッセージ通番（10 進数）

kk....kk

通信層グループ内シーケンス番号（10 進数）

ll....ll

メッセージ状態

次に示すどちらかが表示されます。

- NON_DELIVERED：配信待ちまたは転送待ち
- DELIVERING：転送中

  再起動後の未送信メッセージも含みます。

mm....mm

JMSCorrelationID（文字列）

メッセージ登録時に設定されていない場合，空白が表示されます。

nn....nn

JMSXGroupID（文字列）

メッセージ登録時に設定されていない場合，空白が表示されます。

oo....oo

JMSXGroupSeq（10 進数）

メッセージ登録時に設定されていない場合，空白が表示されます。

pp....pp

JMSMessageID（文字列）

メッセージ登録時に設定されていない場合，空白が表示されます。

qq....qq

通信層グループ内メッセージ有効期限（文字列）

8.3.16 hrmmkaddr（あて先登録）

(1) 形式

```
hrmmkaddr -u あて先アドレス 〔-i ユーザID -p パスワード〕
        〔-S システム名〕 あて先名
```

(2) 機能

サーバ間転送に使用するあて先情報を，あて先情報テーブルに登録します。

(3) オプション

-u あて先アドレス

～＜1～512 文字の文字列＞

メッセージを転送するあて先アドレスを指定します。

転送先システムのキュー間転送用 Web アプリケーションの URL を指定してください。なお，指定する文字列は，RFC2396 で規定された文字とします。

-i ユーザ ID

～＜1～16 文字の英数字＞

BASIC 認証のためのユーザ ID を指定します。

このオプションは-p オプションと同時に指定してください。このオプションを指定して-p オプションを指定していない場合は，エラーが発生します。

-p パスワード

～＜1～16 文字の英数字＞

BASIC 認証のためのパスワードを指定します。

このオプションは-i オプションと同時に指定してください。このオプションを指定して-i オプションを指定していない場合は，エラーが発生します。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンド操作対象となるシステムのシステム名（プロパティの RMSystemName）を指定します。

指定を省略した場合は，環境変数に設定されたシステム名を指定したものとみなされます。

環境変数にもシステム名が設定されていない場合はエラーが発生します。

(4) コマンド引数

あて先名

～＜1～32 文字の識別子＞

あて先情報に対応する論理名を指定します。

指定したあて先名がすでに実在する場合はエラーが発生します。

(5) 注意事項

- 転送キューはこのコマンドのあて先名に対応するあて先情報を基にメッセージを転送します。
- -u オプションで，自システムのキュー間転送用 Web アプリケーションの URL を指定しないでください。

- -p オプションにパスワードを指定した場合，このコマンドの実行中に，プロセスの引数を確認できる OS 機能などでパスワードが観測されるおそれがあります。-p オプションを使用する際は，他ユーザがプロセスの引数を確認できる OS 機能などを使用できない状況下で，このコマンドを実施してください。

8.3.17　hrmmkque（ローカルキューの作成）

(1)　形式

```
hrmmkque -t local 〔-m {persistent | non_persistent}〕
         〔-d {serial | parallel | parallel_unit_order}〕
         〔-n 最大メッセージ数〕
         〔-c キャッシュメッセージ数〕 〔-e メッセージ有効期間〕
         〔-w {sender | receiver}〕 〔-x 表示名〕
         〔-r RDエリア名〕 〔-S システム名〕 キュー名
```

(2)　機能

指定されたキュー属性で，ローカルキューを作成します。

非永続版リソースアダプタの場合，キューはキュー作成ファイルの定義によって作成されます。キュー作成ファイルの詳細については，「3.5.1　キュー作成ファイルの作成」を参照してください。

(3)　オプション

-t local

作成するキューの種類を指定します。ローカルキューが指定されます。

-m {persistent ｜ non_persistent}

～《persistent》

作成するキューの永続性を指定します。

- persistent：永続キュー属性
- non_persistent：非永続キュー属性

各属性を指定したときのメッセージの処理については，「2.3.1　キューの永続性」を参照してください。

-d {serial ｜ parallel ｜ parallel_unit_order}

～《parallel》

作成するキューのメッセージ取り出しモードを指定します。

- serial：シリアル取り出し属性
- parallel：パラレル取り出し属性
- parallel_unit_order：パラレル取り出し属性（ただし，同一ユニット識別子の配信順序制御）

parallel_unit_order を指定する場合は，-m オプションで non_persistent を指定してください。指定しない場合は，エラーとなります。

各属性を指定したときのメッセージの処理については，「2.3.2　メッセージ取り出しモード」を参照してください。

-n 最大メッセージ数

～＜数字＞((1～65535))《1024》

キューに格納するメッセージの最大数を指定します。

-c キャッシュメッセージ数

～＜数字＞((0～65535))《最大メッセージ数》

キャッシュに格納するメッセージの数を指定します。

指定を省略した場合，-n オプションに指定する最大メッセージ数が設定されます。0 を指定する場合，キャッシュにメッセージは格納されません。

このオプションに大きな値を指定する場合，メモリを消費しますがメッセージ受信の性能が向上します。小さい値を指定する場合，メモリの消費は抑えられますが，DB アクセス回数が増えることによってメッセージ受信の性能が低下します。

-e メッセージ有効期間

～＜数字＞((0～2592000))《0》（単位：秒）

キューに格納するメッセージの有効期間を指定します。

0 を指定する場合，メッセージの有効期間は無限です。

有効期間を指定するときのメッセージの処理については，「2.3.5　メッセージの有効期間」を参照してください。有効期間に達すると，そのメッセージはデッドメッセージとして扱われます。

-w {sender ｜ receiver}

～《sender》

転送キューからメッセージを受信する場合，メッセージの有効期間については，送信側の有効期間を使用するのか，または受信側で更新するのかを選択します。転送キューの詳細については，「8.3.20 hrmmkque（転送キューの作成）」を参照してください。

- sender：送信側の有効期間
- receiver：受信側の有効期間

receiver を指定した場合，-e オプションで指定した値がメッセージの有効期間となります。

-x 表示名

～＜ 1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はコマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。詳細は「8.3.17(5)　注意事項」を参照してください。

-r RD エリア名

～＜1～30 文字の識別子および空白＞

メッセージ情報テーブルを格納する RD エリアの名前を指定します。

このオプションを指定する場合，メッセージ情報テーブルを格納する RD エリアをあらかじめ用意してください。HiRDB の RD エリアの作成については，マニュアル「HiRDB コマンドリファレンス」を参照してください。

このオプションを省略する場合，格納する RD エリアを HiRDB が決定します。HiRDB が決定する RD エリアについては，マニュアル「HiRDB SQL リファレンス」を参照してください。

なお，-m オプションで"non_persistent"を指定した場合または DB に Oracle を使用している場合，このオプションの指定値は無効になります。

注意

- RD エリア名に空白を含む場合は，次のように指定してください。

  Windows の場合

  引用符（"）で囲んでください。

  UNIX の場合

  アポストロフィ（’）で囲み，さらにその外側を引用符（"）で囲んでください。
- RD エリア名は大文字と小文字が区別されます。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

作成するキューの名前を指定します。

既存のキューの名前を指定した場合はエラーが発生します。

(5) 注意事項

- コマンド引数に指定するキュー名は，英字の大文字と小文字が区別されません。英字の大文字と小文字の違いだけのキュー名がすでにある場合，エラーが発生します。

  キュー定義ファイルを使用していない場合，-x オプションで指定したキューの表示名で JNDI ネーミングサービスに登録されます。

  キュー定義ファイルを使用している場合は，キュー定義ファイルの指定が優先され，-x オプションで指定した表示名での JNDI ネーミングサービスへの登録はされません。

キュー定義ファイルを使用して Reliable Messaging を運用している場合，キュー定義ファイルを使用しないで Reliable Messaging を再開始したときは，-x オプションで指定した表示名が JNDI ネーミングサービスに登録されます。

キュー定義ファイルの使用有無の詳細については，「3.4.4　キュー定義ファイルの作成（永続版リソースアダプタの場合）」，または「3.5.2　キュー定義ファイルの作成（非永続版リソースアダプタの場合）」を参照してください。

- -x オプションで表示名を指定しなくても，作成するキューのキュー名が，作成済みキューの表示名と重複した場合，キューの作成は失敗します。

8.3.18　hrmmkque（受信用共用キューの作成）

(1)　形式

```
hrmmkque -t shr_receive 〔-l 最大メッセージ長〕
        〔-n 最大メッセージ数〕 〔-c キャッシュメッセージ数〕
        〔-r RDエリア名〕 〔-x 表示名〕
        〔-S システム名〕 キュー名
```

(2)　機能

指定されたキュー属性で，受信用共用キューを作成します。

(3)　オプション

-t shr_receive

作成するキューの種類を指定します。受信用共用キューが指定されます。

-l 最大メッセージ長

～＜数字＞((1000～1048576))《33000》（単位：バイト）

キューに格納するメッセージの最大長（ペイロードの最大長）を指定します。

-n 最大メッセージ数

～＜数字＞((1～65535))《1024》

キューに格納するメッセージの最大数を指定します。

-c キャッシュメッセージ数

～＜数字＞((0～65535))《最大メッセージ数》

キャッシュに格納するメッセージの数を指定します。

指定を省略した場合，-n オプションに指定する最大メッセージ数が設定されます。0 を指定する場合，キャッシュにメッセージは格納されません。

このオプションに大きな値を指定する場合，メモリを消費しますがメッセージ受信の性能が向上します。小さい値を指定する場合，メモリの消費は抑えられますが，DB アクセス回数が増えることによってメッセージ受信の性能が低下します。

-r RD エリア名

～＜1～30 文字の識別子および空白＞

次に示す管理情報テーブルを格納する RD エリアの名前を指定します。

- 共用キュー受信用メッセージ情報テーブル
- 共用キュー受信用ライト管理テーブル
- 共用キュー受信用リード管理テーブル

このオプションを指定する場合，これらの管理情報テーブルを格納する RD エリアをあらかじめ用意してください。また，これらの管理情報テーブルは，受信用共用キューごとに同じ RD エリアに格納する必要があります。HiRDB の RD エリアの作成については，マニュアル「HiRDB コマンドリファレンス」を参照してください。

このオプションを省略する場合，格納する RD エリアを HiRDB が決定します。HiRDB が決定する RD エリアについては，マニュアル「HiRDB SQL リファレンス」を参照してください。

注意

- RD エリア名に空白を含む場合は，次のように指定してください。
  Windows の場合
  引用符（"）で囲んでください。
  UNIX の場合
  アポストロフィ（’）で囲み，さらにその外側を引用符（"）で囲んでください。
- RD エリア名は大文字と小文字が区別されます。

-x 表示名

～＜1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はコマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。詳細は「8.3.18(5)　注意事項」を参照してください。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

作成するキューの名前を指定します。

既存のキューの名前を指定した場合はエラーが発生します。

TP1/EE とシステム間連携をする場合は 1～19 文字のキュー名を指定してください。キュー名が 20 文字の場合は TP1/EE とシステム間連携ができません。

(5) 注意事項

- 受信用共用キューは，接続先のデータベースの種別が HiRDB の場合だけ作成できます。接続先のデータベースの種別が Oracle の場合に受信用共用キューを作成するとエラーが発生します。
- コマンド引数に指定するキュー名は，英字の大文字と小文字が区別されません。英字の大文字と小文字の違いだけのキュー名がすでにある場合，エラーが発生します。
- Reliable Messaging のバージョンによって使用できる共用キューのバージョンが決められています。したがって，共用キューを使用する場合は，送信側システムと受信側システムの Reliable Messaging のバージョンを統一し，必要に応じて共用キューをバージョンアップする必要があります。

  共用キューのバージョンアップの詳細については，「付録 H.3　共用キューのバージョンアップ」を参照してください。
- キュー定義ファイルを使用していない場合，-x オプションで指定したキューの表示名で JNDI ネーミングサービスに登録されます。

  キュー定義ファイルを使用している場合は，キュー定義ファイルの指定が優先され，-x オプションで指定した表示名での JNDI ネーミングサービスへの登録はされません。

  キュー定義ファイルを使用して Reliable Messaging を運用している場合，キュー定義ファイルを使用しないで Reliable Messaging を再開始したときは，-x オプションで指定した表示名が JNDI ネーミングサービスに登録されます。

  キュー定義ファイルの使用有無の詳細については，「3.4.4　キュー定義ファイルの作成（永続版リソースアダプタの場合）」を参照してください。
- -x オプションで表示名を指定しなくても，作成するキューのキュー名が，作成済みキューの表示名と重複した場合，キューの作成は失敗します。

8.3.19 hrmmkque（送信用共用キューの作成）

(1) 形式

```
hrmmkque -t shr_send 〔-l 最大メッセージ長〕
         -b 共用キューを使用する場合の登録先キュー名
         〔-x 表示名〕 〔-S システム名〕 キュー名
```

(2) 機能

指定されたキュー属性で，送信用共用キューを作成します。

(3) オプション

-t shr_send

作成するキューの種類を指定します。送信用共用キューが指定されます。

-l 最大メッセージ長

～＜数字＞((0 または 1～1048576))《0》（単位：バイト）

キューに格納するメッセージの最大長（ペイロードの最大長）を指定します。

メッセージを送信する際のメッセージ長チェックの基準値となります。指定する値は，受信側システムの受信用共用キューに指定された最大メッセージ長と同じにしてください。

メッセージ長をチェックしない場合は，指定を省略するか 0 を指定してください。

-b 共用キューを使用する場合の登録先キュー名

～＜文字列＞

共用キューを使用して複数システム間でのアプリケーション連携をする場合にメッセージを登録する，登録先キュー名を指定します。

自システムの送信用共用キューは定義だけのキューです。メッセージが格納されるメモリやディスク領域などの実体は，相手システムの受信用共用キューです。このオプションでは，自システムの送信したメッセージが登録される登録先キュー名を，相手システムの受信用共用キュー名を基に次に示す形式で指定します。

Reliable Messaging 間で連携する場合

```
〈システム名〉_SHR_〈受信用共用キュー名〉
```

システム名は相手システムの RMSystemName プロパティ指定値です。受信用共用キュー名は，相手システムで定義している受信用共用キュー名です。

他システム間で連携する場合

接続先のシステムで定義しているキュー名を指定してください。

-x 表示名

～＜ 1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合はコマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。詳細は「8.3.19(5)　注意事項」を参照してください。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

作成するキューの名前を指定します。

既存のキューの名前を指定した場合はエラーが発生します。

(5) 注意事項

- 送信用共用キューは，接続先のデータベースの種別が HiRDB のときだけ作成できます。接続先のデータベースの種別が Oracle のときに送信用共用キューを作成するとエラーが発生します。
- -b オプションに対して，Reliable Messaging 間の連携の場合に指定する受信用共用キュー名は，英字の大文字と小文字が区別されません。
- コマンド引数に指定するキュー名は，英字の大文字と小文字が区別されません。英字の大文字と小文字の違いだけのキュー名がすでにある場合，エラーが発生します。
- -b オプションで指定したキュー名に対応する受信用共用キューが接続先のデータベースに存在しない場合，エラーが発生します。
- -b オプションで指定したキュー名に対応する受信用共用キューが連携できる共用キューのバージョンではない場合，エラーが発生します。
- Reliable Messaging のバージョンによって使用できる共用キューのバージョンが決められています。したがって，共用キューを使用する場合は，送信側システムと受信側システムの Reliable Messaging のバージョンを統一し，必要に応じて共用キューをバージョンアップする必要があります。

  共用キューのバージョンアップの詳細については，「付録 H.3　共用キューのバージョンアップ」を参照してください。
- キュー定義ファイルを使用していない場合，-x オプションで指定したキューの表示名で JNDI ネーミングサービスに登録されます。

  キュー定義ファイルを使用している場合は，キュー定義ファイルの指定が優先され，-x オプションで指定した表示名での JNDI ネーミングサービスへの登録はされません。

  キュー定義ファイルを使用して Reliable Messaging を運用している場合，キュー定義ファイルを使用しないで Reliable Messaging を再開始したときは，-x オプションで指定した表示名が JNDI ネーミングサービスに登録されます。

  キュー定義ファイルの使用有無の詳細については，「3.4.4　キュー定義ファイルの作成（永続版リソースアダプタの場合）」を参照してください。
- -x オプションで表示名を指定しなくても，作成するキューのキュー名が，作成済みキューの表示名と重複した場合，キューの作成は失敗します。

8.3.20　hrmmkque（転送キューの作成）

(1)　形式

```
hrmmkque -t transmit 〔-m {persistent | non_persistent}〕
         〔-n 最大メッセージ数〕 〔-c キャッシュメッセージ数〕
         〔-e メッセージ有効期間〕
         -a あて先名 {-v 転送先キュー名 | -y} 〔-i {normal | compatible}〕
         〔-j {exactly_once | in_order}〕 〔-g 通信層のグループ有効期間〕
         〔-s 通信層のメッセージ有効期間〕 〔-x 表示名〕
         〔-r RDエリア名〕 〔-S システム名〕 キュー名
```

(2)　機能

指定されたキュー属性で，転送キューを作成します。

(3)　オプション

-t transmit

作成するキューの種類を指定します。転送キューが指定されます。

-m {persistent ｜ non_persistent}

～《persistent》

作成するキューの永続性を指定します。

- persistent：永続キュー属性
- non_persistent：非永続キュー属性

各属性を指定したときのメッセージの処理については，「2.3.1　キューの永続性」を参照してください。

-j オプションに in_order を指定した場合，このオプションの指定は persistent（永続キュー）だけとなります。-j オプションに in_order を指定して，同時にキューモードに non_persistent（非永続キュー）を指定した場合はエラーが発生します。

-n 最大メッセージ数

～＜数字＞((1～65535))《1024》

キューに格納するメッセージの最大数を指定します。

-c キャッシュメッセージ数

～＜数字＞((0～65535))《1024》

キャッシュに格納するメッセージの数を指定します。

指定を省略した場合，-n オプションに指定した最大メッセージ数が設定されます。0 を指定した場合，キャッシュにメッセージは格納されません。

このオプションに大きな値を指定する場合，メモリを消費しますがメッセージ転送の性能が向上します。小さい値を指定する場合，メモリの消費は抑えられますが，DB アクセス回数が増えることによってメッセージ転送の性能が低下します。

-e メッセージ有効期間

～＜数字＞((1～2592000))《2592000》（単位：秒）

キューに格納するメッセージの有効期間を指定します。

有効期間を指定するときのメッセージの処理については，「2.3.5　メッセージの有効期間」を参照してください。

有効期間に達すると，そのメッセージはデッドメッセージとして扱われます。

-s オプションで指定した値以下の値を指定してください。-s オプションで指定した値より大きい値を指定するとエラーが発生します。

-a あて先名

～＜1～32 文字の文字列＞

メッセージを転送するあて先のあて先名を指定します。hrmmkaddr コマンドで，事前にあて先名を登録する必要があります。登録されていないあて先名を指定するとエラーが発生します。

-v 転送先キュー名

～＜1～20 文字の識別子＞

メッセージを転送する場合，転送したメッセージを登録するキューの名前を指定します。キュー間転送のあて先の詳細については，「2.4.2　キュー間転送のあて先指定」を参照してください。

-y

-v オプションを省略して，-a オプションで指定したあて先名に対応するあて先アドレスだけを送信アドレスとする場合に指定します。キュー間転送のあて先の詳細については，「2.4.2　キュー間転送のあて先指定」を参照してください。

-i {normal ｜ compatible}

～《normal》

作成するキューの転送モードを指定します。

- normal：通常モード
- compatible：互換モード

転送先の Reliable Messaging のバージョンが 01-03 以降の場合は normal を指定します。バージョンが 01-02 以前の場合は compatible を指定します。

normal を指定して作成した転送キューを使用して，01-02 以前の Reliable Messaging にメッセージを転送した場合，ペイロードが空のメッセージとして受信されることがありますので，注意してください。

-j {exactly_once ｜ in_order}

～《exactly_once》

転送に使用する QoS（通信品質）の種別を指定します。

- exactly_once：配送保証および重複防止
- in_order：順序保証

-g 通信層のグループ有効期間

～＜数字＞((10～2592000))（単位：秒）《2592000》

通信層のグループの有効期間を指定します。

このオプションの値は，通信層でメッセージを送受信するために使用するグループの有効期間を示します。このオプションで指定した有効期間に達すると，新たにグループが作成され，メッセージは新しいグループに格納されます。

-s 通信層のメッセージ有効期間

～＜数字＞((10～2592000))（単位：秒）《-e オプションの値》

通信層のグループに格納するメッセージの有効期間を指定します。

このオプションの値は，メッセージがグループ内で管理される期間を示します。このオプションで指定した有効期間に達すると，そのメッセージはデッドメッセージとして扱われます。

-e オプションで指定した値以上の値を指定してください。-e オプションで指定した値未満の値を指定した場合，エラーが発生します。-e オプションで指定した値が 1～9 の場合，このオプションを省略すると値は 10 を指定したとみなされます。

-x 表示名

～＜ 1～64 文字の英数字および_（アンダースコア）＞

キューの表示名を指定します。表示名とは，アプリケーションが JNDI ネーミングサービスからキューを取得するときの，キューの論理名のことです。

指定を省略した場合は，コマンド引数で指定したキュー名と同じ名称を指定したものとみなされます。

指定した表示名と同じ名称を持つキューがすでに存在している場合，エラーとなります。詳細は「8.3.20(5)　注意事項」を参照してください。

-r RD エリア名

～＜ 1～30 文字の識別子および空白＞

メッセージ情報テーブルを格納する RD エリアの名前を指定します。メッセージ情報テーブルを格納する RD エリアをあらかじめ用意してください。HiRDB の RD エリアの作成については，マニュアル「HiRDB コマンドリファレンス」を参照してください。

-m オプションに non_persistent を指定した場合，または DB に Oracle を使用している場合，このオプションの指定値は無効になります。

このオプションを省略した場合，格納する RD エリアを HiRDB が決定します。HiRDB が決定する RD エリアについては，マニュアル「HiRDB SQL リファレンス」を参照してください。

注意

- RD エリア名に空白を含む場合は，次のように指定してください。
  Windows の場合
  引用符（"）で囲んでください。
  UNIX の場合
  アポストロフィ（’）で囲み，さらにその外側を引用符（"）で囲んでください。
- RD エリア名は大文字と小文字が区別されます。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

作成するキューの名前を指定します。

既存のキューの名前を指定した場合はエラーが発生します。

(5) 注意事項

- コマンド引数に指定するキュー名は，英字の大文字と小文字が区別されません。英字の大文字と小文字の違いだけのキュー名がすでにある場合，エラーが発生します。
- -g オプションおよび-s オプションを使うと，受信側のリソースの使用量に影響します。運用方法に合わせた値を指定してください。例えば Web サービスなど，不特定多数が通信をする運用の場合は，-g オプションおよび-s オプションの値を大きくすると，受信側にリソースを蓄積させてしまうので，小さい値を指定することをお勧めします。
- キュー定義ファイルを使用していない場合，-x オプションで指定したキューの表示名で JNDI ネーミングサービスに登録されます。

  キュー定義ファイルを使用している場合は，キュー定義ファイルの指定が優先され，-x オプションで指定した表示名での JNDI ネーミングサービスへの登録はされません。

  キュー定義ファイルを使用して Reliable Messaging を運用している場合，キュー定義ファイルを使用しないで Reliable Messaging を再開始したときは，-x オプションで指定した表示名が JNDI ネーミングサービスに登録されます。

  キュー定義ファイルの使用有無の詳細については，「3.4.4　キュー定義ファイルの作成（永続版リソースアダプタの場合）」を参照してください。
- -x オプションで表示名を指定しなくても，作成するキューのキュー名が，作成済みキューの表示名と重複した場合，キューの作成は失敗します。
- Reliable Messaging を 09-00 にバージョンアップした場合，Reliable Messaging 01-02 以前で作成した転送キューは，転送モードが compatible（互換モード）の転送キューとして扱われます。

8.3.21　hrmregdmsg（デッドメッセージの再登録）

(1)　形式

```
hrmregdmsg {-i デッドメッセージID | -q 移動前キュー名} 〔-S システム名〕
```

(2)　機能

デッドメッセージを，デッドメッセージキューに移動される前のキューに，新しいメッセージとして再登録します。また，デッドメッセージキュー内の該当のデッドメッセージを削除します。

(3)　オプション

-i デッドメッセージ ID

～＜1～39 文字の識別子＞

再登録するデッドメッセージのデッドメッセージ ID（DMID）を指定します。

指定した DMID のメッセージがデッドメッセージキュー内にない場合はエラーが発生します。

-q 移動前キュー名

～＜1～20 文字の識別子＞

デッドメッセージキューに移動する前にメッセージが存在したキュー名を指定します。

指定したキュー名から移動されたすべてのデッドメッセージが，再登録されます。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4)　注意事項

- 対象のデッドメッセージキューは，RMDeadMessageQueueName プロパティで指定したキューです。
- キュー間転送時の送信側デッドメッセージキューには，受信側システムに到達しているメッセージが存在する場合があります。このデッドメッセージを再登録すると，受信側で別のメッセージとして受信されます。そのため，受信側システムのメッセージ到達状態を確認してから再登録してください。詳細は，「4.2.4(2)　サーバ間転送での注意事項」を参照してください。
- 次に示すデッドメッセージは，再登録できません。また，-q オプションで移動前キュー名を指定した場合，次に示すメッセージは再登録しないで，そのほかのメッセージを再登録します。
  - 配信中のメッセージ
  - 配信済みのメッセージ
  - 移動前キューの最大メッセージ長より大きいメッセージ

- デッドメッセージキュー移動時と移動前キューの種類が変わっているメッセージ
- Reliable Messaging 01-01 以前にデッドメッセージキューに移動したメッセージ

- 移動前キューがメッセージの登録を抑止されている場合，デッドメッセージは再登録できません。メッセージの登録抑止については，「8.3.27　hrmstopque（キューの抑止）」を参照してください。
- -q オプションで移動前キュー名を指定した場合で，再登録対象のメッセージが複数あるときは，1 メッセージごとに再登録処理が繰り返されます。再登録処理の途中で次に示すような処理を続行できない障害が発生した場合，処理が中止され，以降のメッセージは再登録されません。
  - 移動前のキューの最大メッセージ数を超えた場合
  - 移動前のキューで，DB 障害が発生した場合
- デッドメッセージの再登録では，メッセージの再登録処理の順番は保証されません。
- Reliable Messaging が管理状態の場合，共用キューにデッドメッセージを再登録したとき，共用キューの状態は更新されません（メッセージが登録された状態になりません）。共用キューの状態は次のタイミングで更新されます。
  - Reliable Messaging を実行状態にする
  - Reliable Messaging を再起動する
- デッドメッセージの再登録により追加されたメッセージは，移動前キューのメッセージ取り出しモードが parallel_unit_order であっても，同一ユニット識別子の配信順序制御の対象とはなりません。また，キューに格納されている時に，メッセージ参照コマンドでユニット識別子は表示されません。ただし，そのメッセージを受信／配信した場合，メッセージには JMS_HITACHI_UnitID プロパティは設定されています。

8.3.22　hrmskipmsg（受信待ちメッセージのスキップ）

(1)　形式

```
hrmskipmsg -n 受信待ちグループ内シーケンス番号 -g グループID
          〔-S システム名〕 キュー名
```

(2)　機能

受信待ちになっているメッセージをスキップします。このコマンドは QoS（通信品質）が順序保証の転送メッセージを受信しているグループに対して，順序が前のメッセージが障害などで受信されないで，あとのメッセージがグループに滞留した場合に使用します。

(3)　オプション

-n 受信待ちグループ内シーケンス番号

～＜数字＞((0～18446744073709551615))

スキップしたい受信待ちメッセージの，通信層グループ内シーケンス番号を指定します。

最小の受信待ちメッセージ番号だけ指定できます。それ以外の受信待ちメッセージ番号を指定した場合はエラーが発生します。

-g グループ ID

～＜1～256 文字の文字列＞

コマンド操作対象となる通信層のグループ ID を指定します。QoS が in-order の場合だけ指定できます。QoS が exactly-once のグループ ID を指定した場合はエラーが発生します。

グループが無効の場合は，メッセージのスキップはできません。グループが無効の場合にこのオプションを指定するとエラーが発生します。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

スキップしたいメッセージが送信側から転送される予定のキューの名前を指定します。

指定したキューが存在しない，またはローカルキュー以外のキューを指定した場合はエラーが発生します。

(5) 注意事項

- 受信済みであるメッセージはスキップできません。受信済みであるメッセージ通番を指定するとエラーが発生します。メッセージが受信か未受信かは，hrmlstrs コマンドで確認してください。
- メッセージのスキップ後にそのメッセージが再送された場合は，エラーが発生します。メッセージは送信側のデッドメッセージキューに格納されます。
- スキップしたメッセージ以降は再び順序保証が保たれます。

8.3.23 hrmstart（実行状態への移行）

(1) 形式

```
hrmstart 〔-S システム名〕
```

(2) 機能

Reliable Messaging を管理状態から実行状態に移行します。管理状態および実行状態については，「4.1.4 Reliable Messaging の状態遷移（永続版リソースアダプタの場合）」を参照してください。

(3) オプション

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。

HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) 注意事項

Reliable Messaging が実行状態のときにこのコマンドを実行するとエラーが発生します。

8.3.24 hrmstartque（キューの抑止解除）

(1) 形式

永続版リソースアダプタの場合

```
hrmstartque -y {ap_send | ap_receive | trs_send | trs_receive | ap_all | all}
          〔-S  システム名〕 キュー名
```

非永続版リソースアダプタの場合

```
hrmstartque -y {ap_send | ap_receive | ap_all | all}
          〔-S  システム名〕 キュー名
```

(2) 機能

キューのメッセージの送受信を抑止解除します。

非永続版リソースアダプタの場合，ローカルキューの抑止解除だけできます。

(3) オプション

- 永続版リソースアダプタの場合

-y {ap_send | ap_receive | trs_send | trs_receive | ap_all | all}

抑止解除の形態を指定します。

- ap_send：メッセージの送信抑止の解除

- ap_receive：メッセージの受信抑止の解除
- trs_send：転送キューのキュー間転送メッセージの送信抑止の解除
- trs_receive：ローカルキューの転送メッセージの受信抑止の解除
- ap_all：メッセージの送受信抑止の解除
- all：指定するキューの種類に応じて，抑止状態をすべて解除

- 非永続版リソースアダプタの場合

-y {ap_send | ap_receive | ap_all | all}

抑止解除の形態を指定します。

- ap_send：メッセージの送信抑止の解除
- ap_receive：メッセージの受信抑止の解除
- ap_all：メッセージの送受信抑止の解除
- all：ap_all と同じ意味になります。

キューの種類によって，指定できる抑止解除の形態が異なります。キューと抑止解除の形態との対応を次の表に示します。

表 8-3　キューと抑止解除できる形態

抑止解除の形態	ローカルキュー		転送キュー	送信用共用キュー	受信用共用キュー	デッドメッセージキュー
	永続版	非永続版				
ap_send	○	○	○	×	○	×
ap_receive	○	○	×	×	○	○
trs_send	×	×	○	×	×	×
trs_receive	○	×	×	×	×	×
ap_all	○	○	×	×	○	×
all	○	○※	○	×	○	○

（凡例）

○：指定できます。

×：指定できません。指定した場合，エラーが発生します。

注※

非永続版リソースアダプタで all 指定すると，ap_all を指定したときと同じになります。

-S システム名

～<先頭が英字の 1～3 文字の大文字英字または数字>

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。
HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

抑止状態を解除するキューの名前を指定します。

- 永続版リソースアダプタの場合
  指定したキューが存在しない，または送信用共用キューを指定した場合はエラーが発生します。
- 非永続版リソースアダプタの場合
  指定したキューが存在しない場合はエラーが発生します。

8.3.25 hrmstarttrs（送受信抑止解除）

(1) 形式

```
hrmstarttrs -y {send | receive} 〔-S システム名〕 キュー名
```

(2) 機能

転送メッセージの送受信抑止状態を元の状態に戻します。

(3) オプション

-y {send | receive}

抑止を解除する形式を指定します。

- send：送信抑止解除
- receive：受信抑止解除

ローカルキューの場合，receive を指定してください。転送キューの場合，send を指定してください。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。
HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～＜1～20 文字の識別子＞

送受信抑止を解除するキューの名前を指定します。

指定したキューが存在しない，またはローカルキュー，転送キュー以外のキューを指定した場合はエラーが発生します。

(5) 注意事項

Reliable Messaging 01-01 以降での，このコマンドの使用は推奨しません。キュー間転送の抑止を解除する場合は，hrmstartque コマンドの-y オプションに trs_send または trs_receive を指定してください。hrmstartque コマンドについては，「8.3.24　hrmstartque（キューの抑止解除）」を参照してください。

8.3.26　hrmstop（管理状態への移行）

(1) 形式

```
hrmstop 〔-S システム名〕
```

(2) 機能

Reliable Messaging を実行状態から管理状態に移行します。

実行状態および管理状態については，「4.1.4　Reliable Messaging の状態遷移（永続版リソースアダプタの場合）」を参照してください。

(3) オプション

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) 注意事項

- Reliable Messaging が管理状態のときにこのコマンドを実行するとエラーが発生します。
- アプリケーションの稼働中に管理状態に移行すると，アプリケーションでエラーが発生するおそれがあります。

8.3.27　hrmstopque（キューの抑止）

(1)　形式

永続版リソースアダプタの場合

```
hrmstopque -y {ap_send | ap_receive | trs_send | trs_receive | ap_all | all}
          〔-S　システム名〕　キュー名
```

非永続版リソースアダプタの場合

```
hrmstopque -y {ap_send | ap_receive | ap_all | all}
          〔-S　システム名〕　キュー名
```

(2)　機能

キューのメッセージの送受信を抑止します。

非永続版リソースアダプタの場合，ローカルキューの抑止だけができます。

(3)　オプション

- 永続版リソースアダプタの場合

-y {ap_send ｜ ap_receive ｜ trs_send ｜ trs_receive ｜ ap_all ｜ all}
　抑止形態を指定します。
  - ap_send：メッセージの送信抑止
  - ap_receive：メッセージの受信抑止
  - trs_send：キュー間転送メッセージの送信抑止
  - trs_receive：キュー間転送メッセージの受信抑止
  - ap_all：メッセージの送受信抑止
  - all：指定するキューの種類に応じて，抑止できる形態をすべて抑止

- 非永続版リソースアダプタの場合

-y {ap_send ｜ ap_receive ｜ ap_all ｜ all}
　抑止形態を指定します。
  - ap_send：メッセージの送信抑止
  - ap_receive：メッセージの受信抑止
  - ap_all：メッセージの送受信抑止
  - all：ap_all と同じ意味になります

キューの種類によって，指定できる抑止形態が異なります。キューと抑止形態との対応を次の表に示します。

表 8-4　キューと抑止できる形態

抑止形態	ローカルキュー		転送キュー	送信用共用キュー	受信用共用キュー※1	デッドメッセージキュー
	永続版	非永続版				
ap_send	○	○	○	×	○	×
ap_receive	○	○	×	×	○	○
trs_send	×	×	○	×	×	×
trs_receive	○	×	×	×	×	×
ap_all	○	○	×	×	○	×
all	○	○※2	○	×	○	○

（凡例）

○：指定できます。

×：指定できません。指定した場合，エラーが発生します。

注※1

受信用共用キューへの送信抑止は，すべてのシステムからのメッセージの送信を抑止します。

注※2

非永続版リソースアダプタで all 指定すると，ap_all を指定したときと同じになります。

-S システム名

～＜先頭が英字の 1～3 文字の大文字英字または数字＞

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4)　コマンド引数

キュー名

～＜ 1～20 文字の識別子＞

メッセージを抑止するキューの名前を指定します。

- 永続版リソースアダプタの場合
  指定したキューが存在しない，または送信用共用キューを指定した場合はエラーが発生します。
- 非永続版リソースアダプタの場合
  指定したキューが存在しない場合はエラーが発生します。

(5)　注意事項

- Reliable Messaging の実行状態でこのコマンドを実行した場合，該当のキューに対してのメッセージ送信または受信のトランザクションがすでに存在していたとき，そのトランザクションは有効となります。

- 永続版リソースアダプタの場合
  デッドメッセージキューへのメッセージの移動は抑止されません。
- 非永続版リソースアダプタの場合
  Reliable Messaging を再起動したときは，抑止状態は引き継がれません。再起動後は抑止が解除されます。

8.3.28 hrmstoptrs（送受信抑止）

(1) 形式

```
hrmstoptrs -y {send | receive} 〔-S システム名〕 キュー名
```

(2) 機能

転送メッセージの受信や転送メッセージの送信を抑止します。

(3) オプション

-y {send | receive}

抑止する形式を指定します。

- send：送信抑止
- receive：受信抑止

ローカルキューの場合，receive だけ指定できます。転送キューの場合，send だけ指定できます。

-S システム名

～<先頭が英字の 1～3 文字の大文字英字または数字>

コマンドの操作対象となるシステム名（RMSystemName プロパティ指定値）を指定します。

指定を省略した場合は，HRM_SYSTEM_NAME 環境変数に指定したシステム名が設定されます。HRM_SYSTEM_NAME 環境変数にもシステム名が指定されていないときはエラーが発生します。

(4) コマンド引数

キュー名

～< 1～20 文字の識別子>

送受信を抑止するキューの名前を指定します。

指定したキューが存在しない，またはローカルキュー，転送キュー以外のキューを指定した場合はエラーが発生します。

(5) 注意事項

Reliable Messaging 01-01 以降での，このコマンドの使用は推奨しません。キュー間転送の抑止をする場合は，hrmstopque コマンドの-y オプションに trs_send または trs_receive を指定してください。hrmstopque コマンドについては，「8.3.27　hrmstopque（キューの抑止）」を参照してください。

9 障害対策

Reliable Messaging は障害対策のためにメッセージログとトレースを出力します。

この章では，メッセージログとトレースの種類，設定，および出力内容の詳細について説明します。

9.1　障害時の出力情報概要

Reliable Messaging の運用時に障害が発生した場合は，メッセージログとトレースを参照して原因を調査し，対策してください。

Reliable Messaging のメッセージログとトレースを次の表に示します。

表 9-1　Reliable Messaging のメッセージログとトレース

項番	種類	名称	説明	使用可否	
				永続版リソースアダプタ	非永続版リソースアダプタ
1	メッセージログ	開始停止メッセージログ	Reliable Messaging の開始と停止についての情報とユーザが指定したプロパティの値が出力されます。 詳細については，「9.3.1　開始停止メッセージログ」を参照してください。	○	○
2		Application Server 用メッセージログ	Reliable Messaging の開始と停止についての情報と動作中の情報が出力されます。 詳細については，「9.3.2　Application Server 用メッセージログ」を参照してください。	○	○
3	トレース	メソッドトレース	メソッドの入口情報，出口情報，デバッグ情報およびエラー情報が出力されます。 詳細については，「9.3.3　メソッドトレース」を参照してください。	○	○
4		共用キューイベントトレース	共用キューにメッセージが格納されたことを表すイベントの情報が出力されます。 詳細については，「9.3.4　共用キューイベントトレース」を参照してください。	○	×
5		回線トレース	キュー間転送でやり取りされる電文の情報を出力します。 詳細については，「9.3.5　回線トレース」を参照してください。	○	×
6		PRF トレース	Reliable Messaging の性能問題の特定と早期解決を目的とした性能解析トレース情報が出力されます。 詳細については，「9.3.6　PRF トレース」を参照してください。	○	○

（凡例）

○：使用できます。

×：使用できません。

Reliable Messaging の起動時に各メッセージログおよびトレースを新規作成できない場合または開くことができない場合は，Reliable Messaging の起動を中止します。

永続版リソースアダプタでは，Reliable Messaging の管理情報テーブルを格納する DB で障害が発生した場合は，Reliable Messaging を停止し，DBMS の機能によって原因を調査し対策してください。対象となるテーブルについては，「4.3.6　管理情報テーブルの一覧」を参照してください。DB の回復方法については，HiRDB または Oracle のマニュアルを参照してください。DB の回復後，Reliable Messaging を再度開始します。

9.2　ログとトレースの設定

起動停止メッセージログファイル，Application Server のメッセージログファイル，PRF の設定は Application Server の設定に従います。ここではそれ以外のトレース（メソッドトレース，共用キューイベントトレース，回線トレース）の設定について説明します。トレースは以下の設定が行えます。

- ローテーション方式
- ファイル面数
- ファイルサイズ
- ローテーション時刻

J2EE サーバ単位または，Reliable Messaging のリソースアダプタ単位で設定を行います。各設定をどちらの単位で行うかを次の表に示します。

表 9-2　トレース設定の単位

	ローテーション方式		ファイル面数	ファイルサイズ	ローテーション時刻
	ローテーション形式	ローテーションモード			
J2EE サーバ単位	○	○	－	－	○
Reliable Messaging のリソースアダプタ単位	－	－	○	○	－

（凡例）
○：本単位で設定を行うことを示す
－：本単位で設定を行なわないことを示す

9.2.1　ローテーション方式の設定

ローテーション方式の設定では以下のローテーション形式，およびローテーションモードを設定できます。「ラップアラウンド形式，またはシフト形式」と「ファイルサイズモード，または時刻＋ファイルサイズモード」の組み合わせをサポートします。

ローテーション形式およびローテーションモードの設定方法は Application Server の仕様に従います。詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「3.　トラブルシューティングのための準備」を参照してください。

(1) ローテーション形式

(a) ラップアラウンド形式

ローテーションが発生した場合，出力ファイルを切り替えて，出力を継続します。規定の面数のトレースファイルに出力し終えた場合，ラップアラウンドし，出力を継続します。

(b) シフト形式

ローテーションが発生した場合，出力ファイルをシフト（リネーム）し，バックアップを行った後新しくファイルを作成し，出力を継続します。現在トレースを出力している最新ファイル名は固定となります。バックアップファイルが規定の面数を超えた場合，面番号が最も大きいバックアップファイルを削除します。

(2) ローテーションモード

(a) ファイルサイズモード

トレースファイルが最大サイズに達した場合に，出力ファイルのローテーションが発生します。

(b) 時刻＋ファイルサイズモード

トレースファイルが最大サイズに達した場合，または時刻が毎日規定時刻に達した場合に，出力ファイルのローテーションが発生します。

9.2.2 ファイル面数の設定

ローテーション形式によりファイル面数の意味が異なります。ローテーション形式によるファイル面数の意味を以下に示します。

ラップアラウンド形式を使用する場合

ファイル面数とはトレースファイルが作成される最大数を示します。

シフト形式を使用する場合

ファイル面数とはバックアップファイルの最大数を示します。現在トレースを出力している最新ファイルは面数に含まれません。

どちらの形式を使用する場合でも，ファイル面数は Reliable Messaging のコンフィグレーションプロパティの RMLogTraceFileNum キーで設定可能です。

9.2.3 ファイルサイズの設定

トレースファイル 1 面あたりの最大サイズを設定します。Reliable Messaging のコンフィグレーションプロパティの RMLogTraceFileSize キーで設定可能です。

9.2.4　ローテーション時刻の設定

ローテーションモードで時刻＋ファイルサイズモードを使用する場合に使用します。ローテーションが発生する時刻を設定します。ローテーション時刻の設定方法は Application Server の仕様に従います。詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「3.　トラブルシューティングのための準備」を参照してください。

9.3　障害時の出力情報詳細

障害時の出力情報詳細を説明します。

9.3.1　開始停止メッセージログ

開始停止メッセージログには，Reliable Messaging の開始と停止についての情報，およびユーザが指定したプロパティの値が出力されます。

- 出力先ディレクトリ

  <Application Serverのインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/connectors

- 出力ログファイル名

  ラップアラウンド形式ローテーションの場合

  <Reliable Messagingの表示名><面番号>.log

  シフト形式ローテーションの場合

  - 最新ファイル名

    <Reliable Messagingの表示名>.log

  - バックアップファイル名

    <Reliable Messagingの表示名><面番号>.log

- コントロールファイル名※

  <Reliable Messagingの表示名>.conf

  Reliable Messaging の表示名については，永続版リソースアダプタの場合は「3.4.4(1)　キュー定義ファイルの記述形式」を，非永続版リソースアダプタの場合は「3.5.2(1)　キュー定義ファイルの記述形式」を参照してください。初期値は"Cosminexus_Reliable_Messaging"です。

  面番号およびファイルサイズは，Application Server の設定に従います。詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」を参照してください。

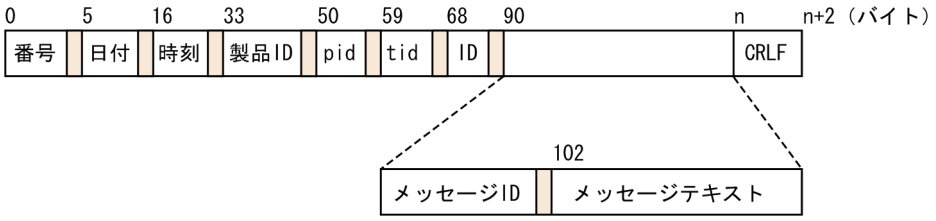
  注※

  ログライブラリが使用するバイナリ形式のファイルを出力先ディレクトリに出力します。

- 出力形式

  開始停止メッセージログの出力形式を次の図に示します。

図 9-1　開始停止メッセージログの出力形式



（凡例）

：1バイト以上の空白を表します。

開始停止メッセージログの出力項目を次の表に示します。

表 9-3　開始停止メッセージログの出力項目

項番	出力項目	長さ (バイト)	説明
1	番号	4	開始停止メッセージログの出力通番です。
2	日付	10	yyyy/mm/dd 形式の出力日付です。
3	時刻	12	hh:mm:ss.sss 形式の出力時刻です。 ローカル時刻でミリ秒単位の時刻です。
4	製品 ID	16	製品を識別するための識別子です。Component Container を表す"HEJB"が出力されます。
5	pid	8	プロセス ID です。
6	tid	8	スレッド ID です。
7	ID	11	空白が出力されます。
8	メッセージ ID	11	KFRMnnnnn-Y 形式のメッセージ ID です。
9	メッセージテキスト	0～512※ (可変長)	メッセージテキストです。
10	CRLF	2	終端記号です。

注※

長さの上限は目安となる値です。上限値を超えて出力されることがあります。ただし，メッセージログ 1 行の長さは 4185 バイトが上限であり，これを超える情報は切り捨てられます。

- 注意事項

開始停止メッセージログを出力するには，リソースアダプタの稼働ログが出力されるように Application Server を設定する必要があります。詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」を参照してください。

9.3.2 Application Server 用メッセージログ

Application Server 用メッセージログには，Reliable Messaging の開始と停止についての情報，および動作中の情報が出力されます。

- 出力先ディレクトリ

  ```
  <Application Serverのインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs
  ```

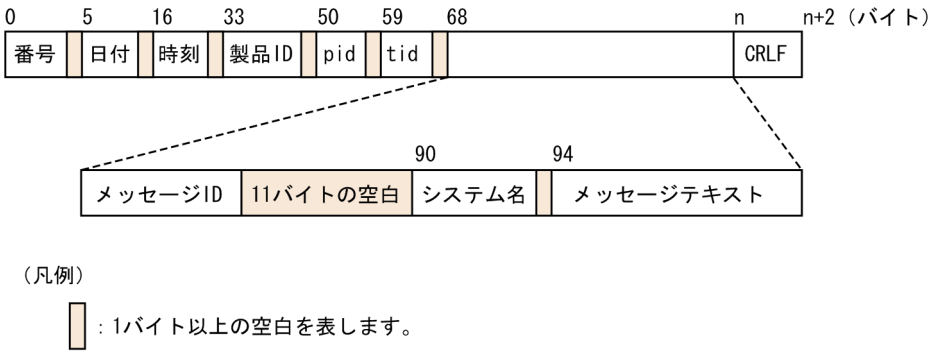
- 出力ファイル名

  Application Server 用メッセージログは，Application Server の稼働ログです。出力ファイル名については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」を参照してください。

- 出力形式

  Application Server 用メッセージログの出力形式を次の図に示します。

図 9–2 Application Server 用メッセージログの出力形式



Application Server 用メッセージログの出力項目を次の表に示します。

表 9–4 Application Server 用メッセージログの出力項目

項番	出力項目	長さ（バイト）	説明
1	番号	4	Application Server 用メッセージログの出力通番です。
2	日付	10	yyyy/mm/dd 形式の出力日付です。
3	時刻	12	hh:mm:ss.sss 形式の出力時刻です。 ローカル時刻でミリ秒単位の時刻です。
4	製品 ID	16	製品を識別するための識別子です。Component Container を表す"HEJB"が出力されます。
5	pid	8	プロセス ID です。
6	tid	8	スレッド ID です。
7	メッセージ ID	11	KFRMnnnnn-Y 形式のメッセージ ID です。
8	システム名	3	Reliable Messaging のシステム名です。
9	メッセージテキスト	0～512※	メッセージテキストです。

項番	出力項目	長さ（バイト）	説明
		（可変長）	
10	CRLF	2	終端記号です。

注※

長さの上限は目安となる値です。上限値を超えて出力されることがあります。ただし，メッセージログ 1 行の長さは 4185 バイトが上限であり，これを超える情報は切り捨てられます。

- 注意事項

出力レベルを指定することで，Application Server のメッセージログファイルに出力するメッセージログの出力量を制御できます。出力レベルは，Application Server のログレベルの設定に従います。Application Server のログレベルの設定については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」を参照してください。

なお，次に示すメッセージは出力レベルに関係なく稼働ログに出力されます。

- KFRM01009-I
- KFRM01807-W
- KFRM13006-W※
- KFRM13007-W
- KFRM13011-W
- KFRM13024-W
- KFRM13025-I
- KFRM13026-I
- KFRM13027-I
- KFRM13028-W
- KFRM13029-I
- KFRM16010-I
- KFRM16010-I
- KFRM17007-W
- KFRM17008-W
- KFRM17009-W
- KFRM17010-W
- KFRM17011-W
- KFRM17012-W
- KFRM17014-W
- KFRM20094-W

- KFRM20121-W
- KFRM20122-W
- KFRM20123-W
- KFRM40004-W※
- KFRM40012-I

注※

デッドメッセージキュー利用時だけ出力レベルに関係なく出力されます。

9.3.3 メソッドトレース

メソッドトレースには，メソッドの入口情報，出口情報，デバッグ情報およびエラー情報が出力されます。

- 出力先ディレクトリ

```
<Application Serverのインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/RM/maintenance
```

- 出力トレースファイル名

```
ラップアラウンド形式ローテーションの場合

mtd_<Reliable Messagingの表示名>_<面番号>.log


シフト形式ローテーションの場合
```

  - 最新ファイル名

```
mtd_<Reliable Messagingの表示名>_.log
```

  - バックアップファイル名

```
mtd_<Reliable Messagingの表示名>_<面番号>.log
```

- コントロールファイル名※

```
mtd_<Reliable Messagingの表示名>_.conf
```

Reliable Messaging の表示名については，永続版リソースアダプタの場合は「3.4.4(1)　キュー定義ファイルの記述形式」を，非永続版リソースアダプタの場合は「3.5.2(1)　キュー定義ファイルの記述形式」を参照してください。

面番号は RMLogTraceFileNum プロパティ指定値が最大数です。また，トレースファイル（拡張子が log のファイル）サイズは RMLogTraceFileSize プロパティ指定値が最大長（単位：バイト）です。

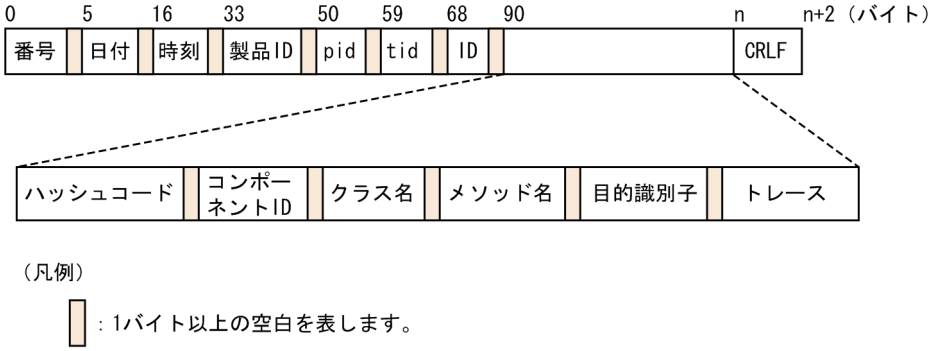
注※

ログライブラリが使用するバイナリ形式のファイルを出力先ディレクトリに出力します。ファイルサイズは固定で 256 バイトです。

- 出力形式

  メソッドトレースの出力形式を次の図に示します。

図 9-3　メソッドトレースの出力形式



メソッドトレースの出力項目を次の表に示します。

表 9-5　メソッドトレースの出力項目

項番	出力項目	長さ (バイト)	説明
1	番号	4	メソッドトレースの出力通番です。
2	日付	10	yyyy/mm/dd 形式の出力日付です。
3	時刻	12	hh:mm:ss.sss 形式の出力時刻です。 ローカル時刻でミリ秒単位の時刻です。
4	製品 ID	16	製品を識別するための識別子です。 Reliable Messaging を表す"RM"が出力されます。
5	pid	8	プロセス ID です。
6	tid	8	スレッド ID です。
7	ID	11	空白が出力されます。
8	ハッシュコード	1～11	トレースを出力するオブジェクトのハッシュコード（16 進数）です。
9	コンポーネント ID	3	トレースを出力するコンポーネントを識別するための識別子です。
10	クラス名	1～25※1 (可変長)	トレースを出力するクラスの名前です。
11	メソッド名	1～25※1 (可変長)	トレースを出力するメソッドの名前です。
12	目的識別子	3	トレースの出力目的を表す識別子※2 です。
13	トレース	0～512※1 (可変長)	次に示す情報が出力されます。 • メソッド入口または出口情報 メソッドの引数または戻り値の値です。 • 他製品呼び出し情報

項番	出力項目	長さ（バイト）	説明
			Application Server または他製品のクラス名，メソッド名および引数です。 • エラー情報 障害要因となったエラー情報です。 なお，他製品から例外が発生した場合は，スタックトレース※3が出力されます。 • デバッグ情報 デバッグ情報です。メソッド内の処理分岐では，分岐の要因となった情報が出力されます。 RMMethodTraceLevel プロパティに指定する出力レベル※4によって出力情報は異なります。
14	CRLF	2	終端記号です。

注※1

長さの上限は目安となる値です。上限値を超えて出力されることがあります。ただし，メソッドトレース 1 行の長さは 4185 バイトが上限であり，これを超える情報は切り捨てられます。

注※2

目的識別子の一覧を次の表に示します。

表 9-6　目的識別子の一覧

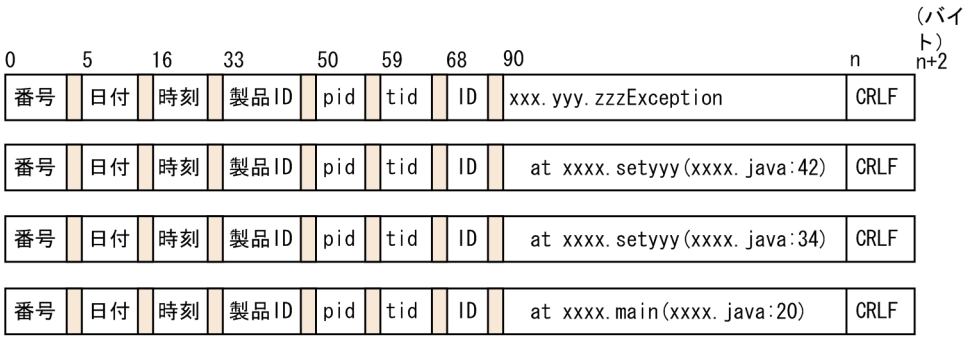
項番	目的識別子	意味
1	STA（start）	メソッドの開始
2	END（end）	メソッドの終了
3	CAL（call）	メソッドの呼び出し
4	RET（return）	メソッドの戻り
5	ERR（error）	メソッドの例外
6	DBG（debug）	メソッドのデバッグ情報（分岐など）
7	INF（information）	メソッドの処理情報（注意，警告など）
8	OPR（operation）	メソッドの処理情報（処理実行情報）

注※3

他製品から例外が発生した場合は，スタックトレースが出力されます。その際，ハッシュコードからトレースまでの出力項目が出力されず，スタックトレースが出力されます。スタックトレースは可変長（0～512）のデータです。

スタックトレースの出力例を次の図に示します。

図 9-4　スタックトレースの出力例



（凡例）

：1バイト以上の空白を表します。

注※4

RMMethodTraceLevel プロパティに指定する出力レベルによって，トレースに出力される情報は異なります。出力情報と出力レベルを次の表に示します。

表 9-7　メソッドトレースの出力情報と出力レベル

項番	出力情報	レベル 1	レベル 2	レベル 3	レベル 4	レベル 5
1	エラー情報 （例外発生原因）	○	○	○	○	○
2	Reliable Messaging 内部スレッドの処理の入口／出口情報 （クラス名，メソッド名，引数，戻り値）	○	○	○	○	○
3	Reliable Messaging のユーザインタフェースのメソッド入口／出口情報 （クラス名，メソッド名，引数，戻り値）	×	○	○	○	○
4	他製品のインタフェースの入口／出口情報 （クラス名，メソッド名，引数，戻り値）	×	×	○	○	○
5	Reliable Messaging 内部インタフェース間の入口／出口情報 （クラス名，メソッド名，引数，戻り値）	×	×	×	○	○
6	Reliable Messaging のデバッグ情報 （処理の分岐などの情報）	×	×	×	×	○
7	Reliable Messaging の処理情報（注意，警告などを示す情報）	×	×	○	○	○
8	Reliable Messaging の処理情報（処理の実行を示す情報）	○	○	○	○	○

（凡例）

○：出力されます。

×：出力されません。

9.3.4　共用キューイベントトレース

共用キューイベントトレースには，イベントの送受信情報が出力されます。

- 出力先ディレクトリ

```
<Application Serverのインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/RM/maintenance
```

- 出力トレースファイル名

```
ラップアラウンド形式ローテーションの場合

shq_<Reliable Messagingの表示名>_<面番号>.log


シフト形式ローテーションの場合
```

  - 最新ファイル名

```
shq_<Reliable Messagingの表示名>.log
```

  - バックアップファイル名

```
shq_<Reliable Messagingの表示名>_<面番号>.log
```

- コントロールファイル名※

```
shq_<Reliable Messagingの表示名>_.conf
```

  Reliable Messaging の表示名については，永続版リソースアダプタの場合は「3.4.4(1)　キュー定義ファイルの記述形式」を，非永続版リソースアダプタの場合は「3.5.2(1)　キュー定義ファイルの記述形式」を参照してください。

  面番号は RMLogTraceFileNum プロパティ指定値が最大数です。また，トレースファイル（拡張子が log のファイル）サイズは RMLogTraceFileSize プロパティ指定値が最大長（単位：バイト）です。

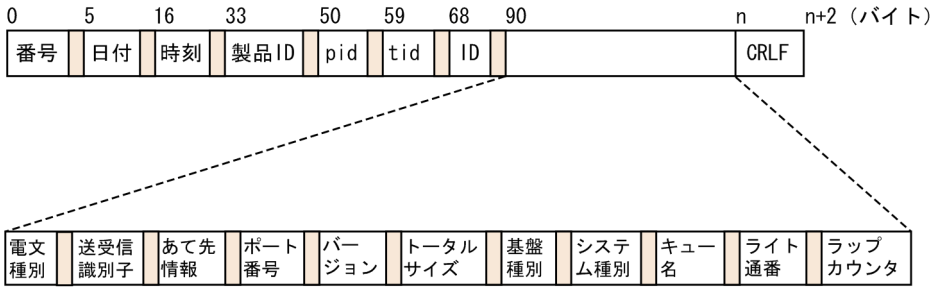
  注※

  ログライブラリが使用するバイナリ形式のファイルを出力先ディレクトリに出力します。ファイルサイズは固定で 256 バイトです。

- 出力形式

  共用キューイベントトレースの出力形式を次の図に示します。

図 9-5　共用キューイベントトレースの出力形式



（凡例）

：1バイト以上の空白を表します。

共用キューイベントトレースの出力項目を次の表に示します。

表 9-8　共用キューイベントトレースの出力項目

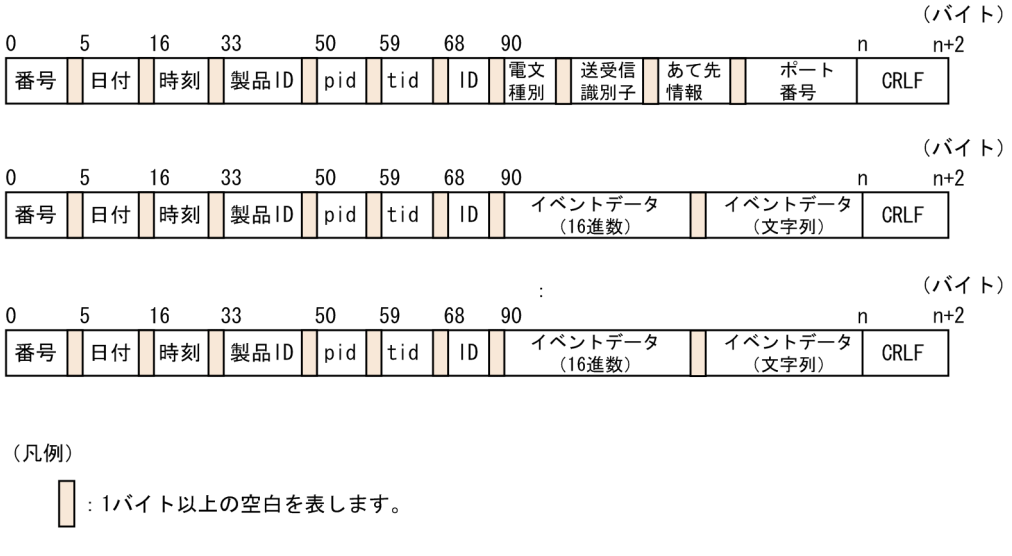
項番	出力項目	長さ (バイト)	説明
1	番号	4	共用キューイベントトレースの出力通番です。
2	日付	10	yyyy/mm/dd 形式の出力日付です。
3	時刻	12	hh:mm:ss.sss 形式の出力時刻です。 ローカル時刻でミリ秒単位の時刻です。
4	製品 ID	16	製品を識別するための識別子です。 Reliable Messaging を表す"RM"が出力されます。
5	pid	8	プロセス ID です。
6	tid	8	スレッド ID です。
7	ID	11	空白が出力されます。
8	電文種別	1～8	受信した電文を特定するための電文種別です。
9	送受信識別子	3	イベントを送信したのか受信したのかを表す識別子です。 送信の場合：SND 受信の場合：RCV
10	あて先情報	1～64 (可変長)	イベントを送信したマシンのホスト名または IP アドレスです。
11	ポート番号	1～11 (可変長)	イベントを送信したポート番号です。
12	バージョン	1～8 (可変長)	イベント転送プロトコルのバージョンです。
13	トータルサイズ	1～11 (可変長)	イベントのヘッダとデータの合計サイズ（10 進数，単位：バイト）です。

項番	出力項目	長さ (バイト)	説明
14	基盤種別	1～8 (可変長)	イベントの送信元を特定するための基盤種別です。Reliable Messaging が送信元の場合は"CosmiRM"が出力されます。
15	システム識別	1～16 (可変長)	イベントの送信元を特定するためのシステム種別です。Reliable Messaging が送信元の場合は RMSystemName プロパティ指定値が出力されます。
16	キュー名	1～32 (可変長)	メッセージが格納されたキューの名前です。
17	ライト通番	1～11 (可変長)	共用キューのメッセージ情報テーブルに書き込んだ際に何番目まで書き込んだかを表す通番です。
18	ラップカウンタ	1～11 (可変長)	共用キューのメッセージ情報テーブルに書き込んだメッセージが書き込み限界数に達したため，1 番目から上書きで書き込みした回数です。
19	CRLF	2	終端記号です。

矛盾不正イベントトレース出力時は，上記の表の項番 12～18 は出力しないで，イベントデータを 16 進数と文字列として最大 128 バイト出力します。

不正イベントトレースの出力形式を次の図に示します。

図 9-6　不正イベントトレース出力時の共用キューイベントトレースの出力形式



9.3.5　回線トレース

キュー間転送でやり取りされる電文の情報を回線トレースファイルに出力します。

- 出力先ディレクトリ

```
<Application Serverのインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/RM/maintenance
```

- 出力トレースファイル名

```
ラップアラウンド形式ローテーションの場合

lin_<Reliable Messagingの表示名>_<面番号>.log


シフト形式ローテーションの場合
```

  - 最新ファイル名

```
lin_<Reliable Messagingの表示名>.log
```

  - バックアップファイル名

```
lin_<Reliable Messagingの表示名>_<面番号>.log
```

- コントロールファイル名※

```
lin_<Reliable Messagingの表示名>_.conf
```

Reliable Messaging の表示名については，永続版リソースアダプタの場合は「3.4.4(1)　キュー定義ファイルの記述形式」を，非永続版リソースアダプタの場合は「3.5.2(1)　キュー定義ファイルの記述形式」を参照してください。

面番号は RMLogTraceFileNum プロパティ指定値が最大数です。また，トレースファイル（拡張子が log のファイル）サイズは RMLogTraceFileSize プロパティ指定値が最大長（単位：バイト）です。

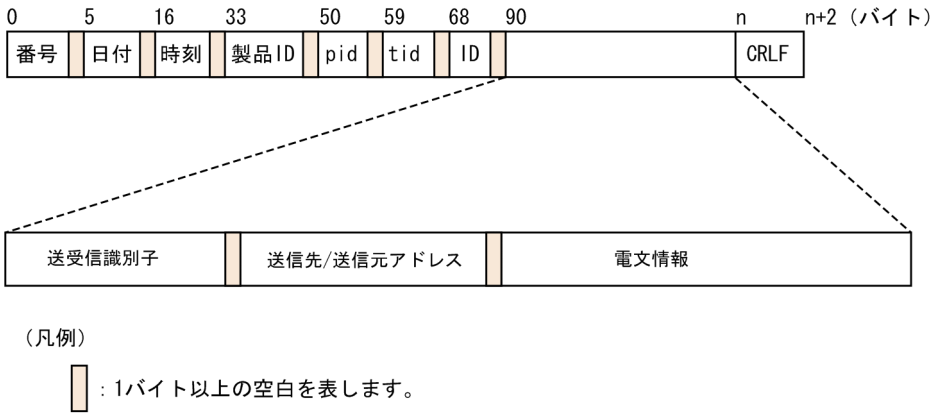
注※

ログライブラリが使用するバイナリ形式のファイルを出力先ディレクトリに出力します。ファイルサイズは固定で 256 バイトです。

- 出力形式

回線トレースの出力形式を次の図に示します。

図 9-7　回線トレースの出力形式



回線トレースの出力項目を次の表に示します。

表 9-9　回線トレースの出力項目

項番	出力項目	長さ（バイト）	説明
1	番号	4	トレースレコードの通番です。
2	日付	10	トレースの取得日付です。 形式は yyyy/mm/dd です。
3	時刻	12	トレースの取得時刻です。 形式は hh:mm:ss.sss です。 ローカル時刻でミリ秒単位の時刻です。
4	製品 ID	16	製品を識別するための識別子です。 Reliable Messaging を表す"RM"が出力されます。
5	pid	8	プロセス ID です。
6	tid	8	スレッド ID です。
7	ID	11	空白が出力されます。
8	送受信識別子	7	出力する電文の種別と送受信タイミングを判別する識別子です。識別子の種類は次のとおりです。 • リクエストメッセージの送信：SND_REQ • リクエストメッセージの受信：RCV_REQ • レスポンスメッセージの送信：SND_RES • レスポンスメッセージの受信：RCV_RES
9	送信先/送信元アドレス	0～533	送信時は出力する電文の送信先アドレス，受信時は，電文を送信した送信元アドレスを示します。
10	電文情報	0～512※1	RMLineTraceLevel プロパティに指定する出力レベル※2 によって出力情報は異なります。 次に示す情報が出力されます。 • プロトコル固有ヘッダ情報 SOAP ヘッダに含まれるプロトコル固有のヘッダ情報です。 • SOAP ヘッダ • SOAP エンベロープ SOAP エンベロープ（SOAP ヘッダ+ SOAP ボディ）情報です。 • SOAP エンベロープ ＋ SOAP アタッチメント情報
11	CRLF	2	終端記号です。

注※1

長さの上限は目安となる値です。上限値を超えて出力されることがあります。SOAP ヘッダを出力する場合は，電文情報を 1000 文字まで出力します。SOAP ヘッダ以外を出力する場合は，トレース 1 行で出力できる範囲まで電文情報を出力します。出力できる範囲を超えた情報は切り捨てます。

注※2

出力情報と出力レベルを次の表に示します。

表 9-10　回線トレースの出力情報と出力レベル

項番	出力情報	レベル 1	レベル 2	レベル 3	レベル 4	レベル 5
1	SOAP ヘッダ	×	×	○	×	×
2	SOAP エンベロープ （SOAP ヘッダ＋ SOAP ボディ）	×	×	×	○	×
3	SOAP エンベロープ＋ SOAP アタッチメント情報	×	×	×	×	○

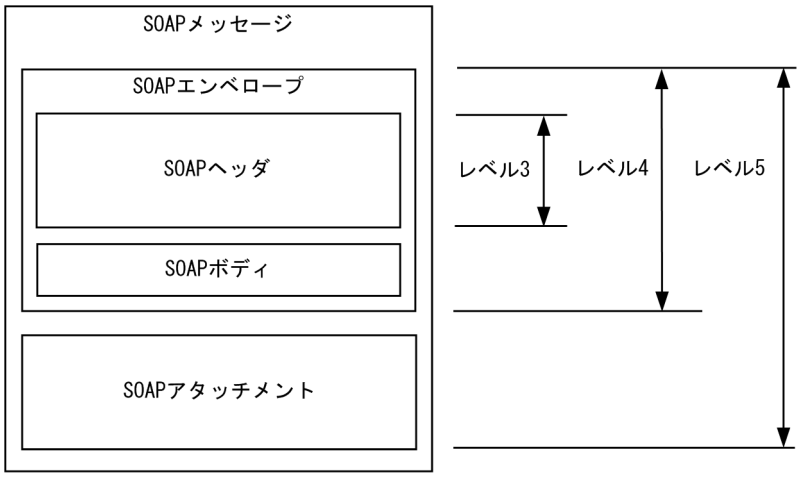
（凡例）

○：出力されます。

×：出力されません。

電文情報の出力範囲と出力レベルを次の図に示します。

図 9-8　電文情報の出力範囲と出力レベル



9.3.6　PRF トレース

PRF トレースとは，クライアントからのリクエストが EIS に至るまで，およびその処理結果がクライアントに返されるまでの，リクエストの一連の処理で出力される性能解析トレース情報です。Reliable Messaging では，アプリケーションからのメッセージ送受信処理や Component Container（TM）からのトランザクション制御などで PRF トレースを取得します。これによって，Reliable Messaging での性能問題を特定でき，早期に問題を解決できます。

PRF トレースの運用方法および機能の詳細については，「4.4　PRF トレースファイルの運用（永続版リソースアダプタの場合）」，「5.3　PRF トレースファイルの運用（非永続版リソースアダプタの場合）」，およびマニュアル「アプリケーションサーバ 機能解説 保守／移行編」を参照してください。

また，出力先ディレクトリおよび出力ファイル名については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の性能解析トレースファイルの出力先と出力情報に関する内容を参照してください。

(1)　トレース取得ポイント

Reliable Messaging では，次に示す処理の入口と出口で PRF トレースを取得します。なお，トレース取得ポイントの詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」を参照してください。

1. アプリケーションからのコネクションの取得と解放処理
2. J2EE サーバ（Component Container）からのコネクションのアソシエート処理
3. アプリケーションからのメッセージ送受信処理
4. J2EE サーバ（Component Container）からのトランザクション制御
5. JMS アプリケーションからの JMS セッションの決着処理
6. J2EE サーバ（Component Container）と連携した Message-driven Bean メッセージ配信処理

各処理での PRF トレースの出力ポイントと位置情報を表すイベント ID，およびトレースの取得レベルについては，「付録 D　PRF トレース取得時のイベント ID」を参照してください。

(2)　PRF トレースの取得情報

Reliable Messaging では PRF トレースから戻り値やキュー名称などが取得できます。PRF トレース情報のヘッダが，Rc，INT，OPR，および OPT の項目が Reliable Messaging で取得できる情報です。また，取得できる情報は PRF トレースの出力ポイントと位置情報を表すイベント ID によって異なります。PRF トレースの出力ポイントと位置情報を表すイベント ID については，「付録 D　PRF トレース取得時のイベント ID」を参照してください。

Reliable Messaging で PRF トレースから取得できる情報について，次に説明します。

(a)　永続版リソースアダプタの場合

永続版リソースアダプタの場合の Reliable Messaging で PRF トレースから取得できる情報を次の表に示します。

Reliable Messaging で取得できる情報以外の PRF トレースの情報については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の性能解析トレースファイルの説明を参照してください。

表 9-11　PRF トレースの取得情報（永続版リソースアダプタの場合）

イベント ID	取得情報			
	Rc	INT	OPR	OPT
0x9360	入口トレースの場合，常に「0」が出力されます。 出口トレースおよび呼び出し後トレースの場合，次のように出力されます。 • 正常終了：0	transacted 引数に指定した値 次のどちらかが出力されます。 • true • false	acknowledgeMode 引数に指定した値 次のどれかが出力されます。※1 • AUTO_ACKNOWLEDGE	−

イベントID	取得情報			
	Rc	INT	OPR	OPT
	• 異常終了：0 以外 トランザクション制御では，XA のエラーコードが出力されます。そのほかのエラー出口では，1 が出力されます。		• DUPS_OK_ACKNOWLEDGE • CLIENT_ACKNOWLEDGE	
0x9361		−	−	−
0x9362		−	−	−
0x9363		−	−	−
0x9364		−	−	−
0x9365		−	−	−
0x9366		−	−	−
0x9367		−	−	−
0x9368		キュー名称	−	−
0x9369		−	−	メッセージの JMS_HITACHI_UnitID プロパティの値，JMSMessageID※5
0x936A		キュー名称	メッセージ送信の優先度	−
0x936B		−	−	メッセージの JMS_HITACHI_UnitID プロパティの値，JMSMessageID※5
0x936C		キュー名称	−	−
0x936D		−	−	メッセージの JMS_HITACHI_UnitID プロパティの値，JMSMessageID※5
0x936E		キュー名称	メッセージ送信の優先度	−
0x936F		−	−	メッセージの JMS_HITACHI_UnitID プロパティの値，JMSMessageID※5

イベントID	取得情報			
	Rc	INT	OPR	OPT
0x9370		キュー名称	メッセージセレクタ※2	－
0x9371		メッセージ取得の可否 次のどちらかが出力されます。 • 取得成功：true • 取得失敗：false	－	メッセージ送信元のルートアプリケーション情報※3
0x9372		キュー名称	タイムアウト値（ミリ秒）	－
0x9373		メッセージ取得の可否 次のどちらかが出力されます。 • 取得成功：true • 取得失敗：false	メッセージセレクタ※2	メッセージ送信元のルートアプリケーション情報※3
0x9374		キュー名称	メッセージセレクタ※2	－
0x9375		メッセージ取得の可否 次のどちらかが出力されます。 • 取得成功：true • 取得失敗：false	－	メッセージ送信元のルートアプリケーション情報※3
0x9376		キュー名称	メッセージセレクタ※2	－
0x9377		－	－	メッセージ送信元のルートアプリケーション情報※3
0x9378		flags 引数の値 次のどれかが出力されます。 • TMNOFLAGS • TMJOIN • TMRESUME	－	－
0x9379		－	－	－
0x937A		flags 引数の値 次のどれかが出力されます。 • TMSUCCESS • TMFAIL • TMSUSPEND	－	－
0x937B		－	－	－
0x937C		－	－	－
0x937D		－	－	－

イベントID	取得情報			
	Rc	INT	OPR	OPT
0x937E		1 フェーズコミットの有無 次のどちらかが出力されます。 • true • false	−	−
0x937F		−	−	−
0x9380		−	−	−
0x9381		−	−	−
0x9382		−	−	−
0x9383		−	−	−
0x9384		flags 引数の値 次のどれかが出力されます。 • TMSTARTRSCAN • TMENDRSCAN • TMNOFLAGS	−	−
0x9385		−	−	−
0x9386		−	−	−
0x9387		−	−	−
0x9388		−	−	−
0x9389		−	−	−
0x938A		−	−	−
0x938B		−	−	−
0x938C		−	−	−
0x938D		−	−	−
0x938E		−	−	−
0x938F		−	−	−
0x9390		−	−	−
0x9391		−	−	−
0x9392		−	−	−
0x9393		−	−	−
0x9394		−	−	−

イベントID	取得情報			
	Rc	INT	OPR	OPT
0x9395		キュー名称	－	メッセージのJMS_HITACHI_UnitID プロパティの値，JMSMessageID※5
0x9396※4		－	－	メッセージ送信元のルートアプリケーション情報※3
0x9397		－	－	－

（凡例）

－：出力されません。

注※1

AUTO_ACKNOWLEDGE，DUPS_OK_ACKNOWLEDGE，または CLIENT_ACKNOWLEDGE 以外を指定した場合は出力されません。

注※2

文字列が 32 文字を超える場合，後半が削除されて，前から数えて 32 文字が出力されます。また，取得できなかった場合は，null が返されます。

注※3

OPT 列に出力されるルートアプリケーション情報（各イベントで一連の処理の先頭になるプロセスで取得した情報）はバイト配列で出力され，ASCII 列（OPT 列の次の列）には OPT 列に出力されるルートアプリケーション情報の ASCII 文字列が出力されます。

ただし，次のメッセージを受信した場合，OPT 列にルートアプリケーション情報は出力されません。

・01-02 以前のバージョンの Reliable Messaging で登録されたメッセージ

・共用キューに登録されたメッセージ

・BytesContainer を利用して転送されたメッセージ

・デッドメッセージキューから再登録されたメッセージ

注※4

正常に処理が終了したときだけ出力されます。エラーが発生したときは出力されません。

注※5

JMS_HITACHI_UnitID プロパティに空文字(“”)，NULL 以外の値を指定した場合だけ出力します。また，JMS_HITACHI_UnitID プロパティと JMSMessageID の間は半角スペース区切りで出力します。「JMS_HITACHI_UnitID プロパティ（半角スペース）JMSMessageID」の文字列が 256 文字を超える場合，後半が削除されて，前から数えて 256 文字が出力されます。

(b) 非永続版リソースアダプタの場合

非永続版リソースアダプタの場合の Reliable Messaging で PRF トレースから取得できる情報を次の表に示します。

Reliable Messaging で取得できる情報以外の PRF トレースの情報については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の性能解析トレースファイルの説明を参照してください。

表 9-12　PRF トレースの取得情報（非永続版リソースアダプタの場合）

イベント ID	取得情報			
	Rc	INT	OPR	OPT
0x9300	入口トレースの場合，常に「0」が出力されます。 出口トレースおよび呼び出し後トレースの場合，次のように出力されます。 • 正常終了：0 • 異常終了：0 以外 トランザクション制御では，XA のエラーコードが出力されます。そのほかのエラー出口では，1 が出力されます。	transacted 引数に指定した値 次のどちらかが出力されます。 • true • false	acknowledgeMode 引数に指定した値 次のどれかが出力されます。※1 • AUTO_ACKNOWLEDGE • DUPS_OK_ACKNOWLEDGE • CLIENT_ACKNOWLEDGE	－
0x9301		－	－	－
0x9302		－	－	－
0x9303		－	－	－
0x9304		－	－	－
0x9305		－	－	－
0x9306		－	－	－
0x9307		－	－	－
0x9308		キュー名称	－	－
0x9309		－	－	メッセージの JMS_HITACHI_UnitID プロパティの値，JMSMessageID※5
0x930A		キュー名称	メッセージ送信の優先度	－
0x930B		－	－	メッセージの JMS_HITACHI_UnitID プロパティの値，JMSMessageID※5
0x930C		キュー名称	－	－
0x930D		－	－	メッセージの JMS_HITACHI_UnitID プロパティの値，

イベントID	取得情報			
	Rc	INT	OPR	OPT
				JMSMessageID※5
0x930E		キュー名称	メッセージ送信の優先度	－
0x930F		－	－	メッセージのJMS_HITACHI_UnitID プロパティの値,JMSMessageID※5
0x9310		キュー名称	メッセージセレクタ※2	－
0x9311		メッセージ取得の可否 次のどちらかが出力されます。 • 取得成功：true • 取得失敗：false	－	メッセージ送信元のルートアプリケーション情報※3
0x9312		キュー名称	タイムアウト値（ミリ秒）	－
0x9313		メッセージ取得の可否 次のどちらかが出力されます。 • 取得成功：true • 取得失敗：false	メッセージセレクタ※2	メッセージ送信元のルートアプリケーション情報※3
0x9314		キュー名称	メッセージセレクタ※2	－
0x9315		メッセージ取得の可否 次のどちらかが出力されます。 • 取得成功：true • 取得失敗：false	－	メッセージ送信元のルートアプリケーション情報※3
0x9316		キュー名称	メッセージセレクタ※2	－
0x9317		－	－	メッセージ送信元のルートアプリケーション情報※3
0x9318		flags 引数の値 次のどれかが出力されます。 • TMNOFLAGS • TMJOIN • TMRESUME	－	－
0x9319		－	－	－

イベントID	取得情報			
	Rc	INT	OPR	OPT
0x931A		flags 引数の値 次のどれかが出力されます。 • TMSUCCESS • TMFAIL • TMSUSPEND	−	−
0x931B		−	−	−
0x931C		−	−	−
0x931D		−	−	−
0x931E		1 フェーズコミットの有無 次のどちらかが出力されます。 • true • false	−	−
0x931F		−	−	−
0x9320		−	−	−
0x9321		−	−	−
0x9322		−	−	−
0x9323		−	−	−
0x9324		flags 引数の値 次のどれかが出力されます。 • TMSTARTRSCAN • TMENDRSCAN • TMNOFLAGS	−	−
0x9325		−	−	−
0x9326		−	−	−
0x9327		−	−	−
0x9328		−	−	−
0x9329		−	−	−
0x932A		−	−	−
0x932B		−	−	−
0x932C		−	−	−
0x932D		−	−	−
0x932E		−	−	−

イベントID	取得情報			
	Rc	INT	OPR	OPT
0x932F		－	－	－
0x9330		－	－	－
0x9331		－	－	－
0x9332		－	－	－
0x9333		－	－	－
0x9335		－	－	－
0x9336		キュー名称	－	メッセージのJMS_HITACHI_UnitID プロパティの値，JMSMessageID※5
0x9337※4		－	－	メッセージ送信元のルートアプリケーション情報※3
0x9338		－	－	－

（凡例）

－：出力されません。

注※1

AUTO_ACKNOWLEDGE，DUPS_OK_ACKNOWLEDGE，または CLIENT_ACKNOWLEDGE 以外を指定した場合は出力されません。

注※2

文字列が 32 文字を超える場合，後半が削除されて，前から数えて 32 文字が出力されます。また，取得できなかった場合は，null が返されます。

注※3

OPT 列に出力されるルートアプリケーション情報（各イベントで一連の処理の先頭になるプロセスで取得した情報）はバイト配列で出力され，ASCII 列（OPT 列の次の列）には OPT 列に出力されるルートアプリケーション情報の ASCII 文字列が出力されます。

注※4

正常に処理が終了したときだけ出力されます。エラーが発生したときは出力されません。

注※5

JMS_HITACHI_UnitID プロパティに空文字(“”)，NULL 以外の値を指定した場合だけ出力します。また，JMS_HITACHI_UnitID プロパティと JMSMessageID の間は半角スペース区切りで出力します。「JMS_HITACHI_UnitID プロパティ（半角スペース）JMSMessageID」の文字列が 256 文字を超える場合，後半が削除されて，前から数えて 256 文字が出力されます。

9.4　キューの障害

Reliable Messaging がキューに関連する障害を検知した場合，該当のキューが閉塞状態になる場合があります。閉塞状態のキューは，hrmdelque コマンドおよび hrmlsque コマンドだけ受け付け，それ以外のコマンドおよびメッセージの送受信は実行できません。Reliable Messaging は，次の契機でキューの障害を検知します。

- Reliable Messaging の復元処理時の管理情報テーブルの不正
- 共用キューのキュー復元時のバージョン不正
- 共用キューが保持するデータの不正および矛盾

キューの障害が発生したときの対処方法について次に説明します。

9.4.1　Reliable Messaging の復元処理時の管理情報テーブルの不正

Reliable Messaging の復元処理時に，管理情報テーブル（キュー情報テーブル，FIFO 情報テーブル，およびメッセージ情報テーブル）のレコードの不正が検知された場合，キューが閉塞します。復元対象と対処方法を次の表に示します。

表 9-13　管理情報テーブルのレコード不正検知時の復元対象とその対処方法

復元対象	対処方法
キュー情報テーブル	hrmdelque コマンドで該当のキューを削除してください。
FIFO 情報テーブル	hrmdelque コマンドで該当のキューを削除してください。※
メッセージ情報テーブル	Reliable Messaging を停止したあと，不正なメッセージ情報テーブルのレコードを削除してください。

注※

該当の FIFO が属するキュー名が不正（存在しない）である場合は，Reliable Messaging がメッセージを出力して，FIFO を削除します。

9.4.2　共用キューのキュー復元時のバージョン不正

共用キューの復元時に，復元した共用キューのバージョンが，使用中の Reliable Messaging のバージョンに対応する共用キューバージョンと互換性がなかった場合，該当の共用キューを閉塞します。受信用共用キューの場合は，共用キュー用の DB テーブルに格納されているバージョンをチェックし，送信用共用キューの場合は，格納先の DB テーブルに格納されているバージョンをチェックします。なお，格納先の DB テーブルが存在しない場合も不正とみなして閉塞します。

Reliable Messaging を停止したあと，受信用共用キューの場合は該当の共用キューを，送信用共用キューの場合は格納先の受信用共用キューをバージョンアップしてください。共用キューのバージョンアップについては，「付録 H.3　共用キューのバージョンアップ」を参照してください。

9.4.3　共用キューが保持するデータの不正および矛盾

Reliable Messaging 起動時，メッセージ送受信時，コマンド実行時など，共用キューの DB テーブルにアクセスする契機で DB の情報に不正や矛盾を検知すると閉塞します。なお，閉塞した共用キューは使用できないため，hrmdelque コマンドで削除してください。

付録

付録 A　JMS 仕様との差異

Reliable Messaging の JMS インタフェースの機能について，Oracle Corporation が提供する JMS Version 1.0.2b との機能差を説明します。

付録 A.1　インタフェースの機能差

Reliable Messaging がサポートするインタフェースを次の表に示します。機能種別ごとにインタフェース名のアルファベット順に説明します。サポートしているインタフェースのメソッドの機能差については，「付録 A.3　メソッドの機能差」を参照してください。

表 A-1　サポートするインタフェース

項番	機能種別	インタフェース名	サポートの有無	注意事項
1	メッセージング共通機能	Connection	○	−
2		ConnectionFactory	○	
3		ConnectionMetaData	○	
4		Destination	○	
5		MessageConsumer	○	
6		MessageProducer	○	
7		Session	○	
8	PTP メッセージング機能	Queue	○	
9		QueueBrowser	○	
10		QueueConnection	○	
11		QueueConnectionFactory	○	
12		QueueReceiver	○	
13		QueueSender	○	
14		QueueSession	○	
15	パブリッシュ／サブスクライブメッセージング機能	TemporaryTopic	×	
16		Topic	×	
17		TopicConnection	×	
18		TopicConnectionFactory	×	
19		TopicPublisher	×	
20		TopicSession	×	

項番	機能種別	インタフェース名	サポートの有無	注意事項
21		TopicSubscriber	×	
22	メッセージ機能	BytesMessage	○	共用キューではヘッダとプロパティの情報を送受信できません。
23		DeliveryMode	○	−
24		MapMessage	×	
25		Message	○	共用キューでは使用できません。共用キューにこのインタフェースを使用してメッセージを送信した場合は例外が発生します。
26		ObjectMessage	○	共用キューでは使用できません。共用キューにこのインタフェースを使用してメッセージを送信した場合は例外が発生します。
27		StreamMessage	×	−
28		TextMessage	○	共用キューでは使用できません。共用キューにこのインタフェースを使用してメッセージを送信した場合は例外が発生します。
29	XA 機能（JMS のオプション機能）	XAConnection	×	−
30		XAConnectionFactory	×	
31		XAQueueConnection	×	
32		XAQueueConnectionFactory	×	
33		XAQueueSession	×	
34		XASession	×	
35		XATopicConnection	×	
36		XATopicConnectionFactory	×	
37		XATopicSession	×	
38	リスナ機能	ExceptionListener	×	
39		MessageListener	○	Message-driven Bean で使用する場合だけサポートしています。
40	その他（JMS のオプション機能）	ConnectionConsumer	×	−
41		ServerSession	×	
42		ServerSessionPool	×	
43		TemporaryQueue	×	

（凡例）

○：サポートしています。

×：未サポートです。

－：特にありません。

付録 A.2　クラスの機能差

Reliable Messaging がサポートするクラスを次の表に示します。機能種別ごとに説明します。

表 A–2　サポートするクラス

項番	機能種別	クラス名	サポートの有無	注意事項
1	PTP メッセージング機能	QueueRequestor	×	－
2	パブリッシュ／サブスクライブメッセージング機能	TopicRequestor	×	

（凡例）

×：未サポートです。

－：特にありません。

付録 A.3　メソッドの機能差

メソッドの機能差について説明します。機能種別ごとの表でインタフェース名およびメソッド名のアルファベット順に説明します。

(1)　メッセージング共通機能のメソッドの機能差

メッセージング共通機能のメソッドの機能差を次の表に示します。

表 A–3　メッセージング共通機能のメソッドの機能差

項番	インタフェース名	メソッド名	機能差
1	Connection	void close()	－
2		java.lang.String getClientID()	このメソッドは未サポートです。このメソッドを発行した場合，必ず null が返されます。
3		ExceptionListener getExceptionListener()	このメソッドは未サポートです。このメソッドを発行した場合，必ず null が返されます。
4		ConnectionMetaData getMetaData()	－

項番	インタフェース名	メソッド名	機能差
5		void setClientID(java.lang.String clientID)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。
6		void setExceptionListener(ExceptionListener listener)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。
7		void start()	−
8		void stop()	−
9	ConnectionFactory	−	−
10	ConnectionMetaData	int getJMSMajorVersion()	1 が返されます。
11		int getJMSMinorVersion()	0 が返されます。
12		java.lang.String getJMSProviderName()	"Cosminexus Reliable Messaging"が返されます。
13		java.lang.String getJMSVersion()	"1.0"が返されます。
14		java.util.Enumeration getJMSXPropertyNames()	"JMSXRcvTimestamp"，"JMSXDeliveryCount"，"JMSXGroupID"および"JMSXGroupSeq"が格納された Enumeration が返されます。
15		int getProviderMajorVersion()	JMS プロバイダのメジャーバージョン番号を返します。この値は，Reliable Messaging のバージョンに対応します。
16		int getProviderMinorVersion()	JMS プロバイダのマイナーバージョン番号を返します。この値は，Reliable Messaging のリビジョンに対応します。
17		java.lang.String getProviderVersion()	JMS プロバイダのバージョンを示す文字列を返します。この文字列は，Reliable Messaging のバージョンおよびリビジョンに対応します。
18	Destination	−	−
19	MessageConsumer	void close()	−
20		MessageListener getMessageListener()	このメソッドは未サポートです。このメソッドを発行した場合，必ず null が返されます。
21		java.lang.String getMessageSelector()	−
22		Message receive()	−
23		Message receive(long timeout)	−
24		Message receiveNoWait()	−

項番	インタフェース名	メソッド名	機能差
25		void setMessageListener(MessageListener listener)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。
26	MessageProducer	void close()	−
27		int getDeliveryMode()	このメソッドは未サポートです。このメソッドを発行した場合，必ず 0 が返されます。
28		boolean getDisableMessageID()	このメソッドは未サポートです。このメソッドを発行した場合，必ず false が返されます。
29		boolean getDisableMessageTimestamp()	このメソッドは未サポートです。このメソッドを発行した場合，必ず false が返されます。
30		int getPriority()	−
31		long getTimeToLive()	このメソッドは未サポートです。このメソッドを発行した場合，必ず 0 が返されます。
32		void setDeliveryMode(int deliveryMode)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。
33		void setDisableMessageID(boolean value)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。
34		void setDisableMessageTimestamp(boolean value)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。
35		void setPriority(int defaultPriority)	−
36		void setTimeToLive(long timeToLive)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。
37	Session	void close()	−
38		void commit()	−
39		BytesMessage createBytesMessage()	−
40		MapMessage createMapMessage()	このメソッドは未サポートです。このメソッドを発行した場合，JMSException が発生します。
41		Message createMessage()	−
42		ObjectMessage createObjectMessage()	−
43		ObjectMessage createObjectMessage(java.io.Serializable object)	−

項番	インタフェース名	メソッド名	機能差
44		StreamMessage createStreamMessage()	このメソッドは未サポートです。このメソッドを発行した場合，JMSException が発生します。
45		TextMessage createTextMessage()	−
46		TextMessage createTextMessage(java.lang.String text)	−
47		MessageListener getMessageListener()	このメソッドは未サポートです。このメソッドを発行した場合，必ず null が返されます。
48		boolean getTransacted()	−
49		void recover()	−
50		void rollback()	−
51		void run()	このメソッドは未サポートです。このメソッドを発行した場合，JMSException が発生します。
52		void setMessageListener(MessageListener listener)	このメソッドは未サポートです。このメソッドで設定した値は実際には使用されません。

（凡例）

−：機能差はありません。

(2) PTP メッセージング機能のメソッドの機能差

PTP メッセージング機能のメソッドの機能差を次の表に示します。

表 A-4 PTP メッセージング機能のメソッドの機能差

項番	インタフェース名	メソッド名	機能差
1	Queue	java.lang.String getQueueName()	−
2		java.lang.String toString()	−
3	QueueBrowser	void close()	−
4		java.util.Enumeration getEnumeration()	現在のキューにあるメッセージの一覧を取得します。一覧を取得したあとにメッセージを参照する際，すでにメッセージは削除または消失していることがあります。
5		java.lang.String getMessageSelector()	−
6		Queue getQueue()	−

項番	インタフェース名	メソッド名	機能差
7	QueueConnection	ConnectionConsumer createConnectionConsumer(Queue queue, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages)	このメソッドは未サポートです。このメソッドを発行した場合, JMSException が発生します。
8		QueueSession createQueueSession(boolean transacted, int acknowledgeMode)	−
9	QueueConnectionFactory	QueueConnection createQueueConnection()	−
10		QueueConnection createQueueConnection(java.lang.String userName, java.lang.String password)	−
11	QueueReceiver	Queue getQueue()	−
12	QueueSender	Queue getQueue()	−
13		void send(Message message)	−
14		void send(Message message, int deliveryMode, int priority, long timeToLive)	このメソッドを発行した場合, deliveryMode 引数と timeToLive 引数を無視して実行されます。
15		void send(Queue queue, Message message)	−
16		void send(Queue queue, Message message, int deliveryMode, int priority, long timeToLive)	このメソッドを発行した場合, deliveryMode 引数と timeToLive 引数を無視して実行されます。
17	QueueSession	QueueBrowser createBrowser(Queue queue)	−
18		QueueBrowser createBrowser(Queue queue, java.lang.String messageSelector)	−
19		Queue createQueue(java.lang.String queueName)	−
20		QueueReceiver createReceiver(Queue queue)	−

項番	インタフェース名	メソッド名	機能差
21		QueueReceiver createReceiver(Queue queue, java.lang.String messageSelector)	−
22		QueueSender createSender(Queue queue)	−
23		TemporaryQueue createTemporaryQueue()	このメソッドは未サポートです。このメソッドを発行した場合, JMSException が発生します。

(凡例)

−：機能差はありません。

(3) メッセージ機能のメソッドの機能差

メッセージ機能のメソッドの機能差を次の表に示します。

表 A-5 メッセージ機能のメソッドの機能差

項番	インタフェース名	メソッド名	機能差
1	BytesMessage	boolean readBoolean()	−
2		byte readByte()	−
3		int readBytes(byte[] value)	−
4		int readBytes(byte[] value, int length)	−
5		char readChar()	−
6		double readDouble()	−
7		float readFloat()	−
8		int readInt()	−
9		long readLong()	−
10		short readShort()	−
11		int readUnsignedByte()	−
12		int readUnsignedShort()	−
13		java.lang.String readUTF()	−
14		void reset()	−
15		void writeBoolean(boolean value)	−
16		void writeByte(byte value)	−
17		void writeBytes(byte[] value)	−

項番	インタフェース名	メソッド名	機能差
18		void writeBytes(byte[] value, int offset, int length)	−
19		void writeChar(char value)	−
20		void writeDouble(double value)	−
21		void writeFloat(float value)	−
22		void writeInt(int value)	−
23		void writeLong(long value)	−
24		void writeObject(java.lang.Object value)	−
25		void writeShort(short value)	−
26		void writeUTF(java.lang.String value)	−
27	DeliveryMode	−	−
28	Message	void acknowledge()	Message-driven Bean で配信されたメッセージには使用できません。 このメソッドを Message-driven Bean で配信されたメッセージに使用した場合は JMSException が発生します。 また，QueueConnection.createQueueSession()メソッドの transacted 引数が true のとき，および transacted 引数が false かつ acknowledgeMode 引数が AUTO_ACKNOWLEDGE または DUPS_OK_ACKNOWLEDGE のときに発行した場合は JMSException が発生します。
29		void clearBody()	−
30		void clearProperties()	−
31		boolean getBooleanProperty(java.lang.String name)	−
32		byte getByteProperty(java.lang.String name)	−
33		double getDoubleProperty(java.lang.String name)	−
34		float getFloatProperty(java.lang.String name)	−
35		int getIntProperty(	−

項番	インタフェース名	メソッド名	機能差
		java.lang.String name)	
36		java.lang.String getJMSCorrelationID()	JMSCorrelationID ヘッダが設定されていないときに発行した場合，null が返されます。
37		byte[] getJMSCorrelationIDAsBytes()	JMSCorrelationID ヘッダが設定されていないときに発行した場合，null が返されます。
38		int getJMSDeliveryMode()	JMSDeliveryMode ヘッダが設定されていないときに発行した場合，0 が返されます。
39		Destination getJMSDestination()	JMSDestination ヘッダが設定されていないときに発行した場合，null が返されます。
40		long getJMSExpiration()	JMSExpiration ヘッダが設定されていないときに発行した場合，0 が返されます。
41		java.lang.String getJMSMessageID()	JMSMessageID ヘッダが設定されていないときに発行した場合，null が返されます。
42		int getJMSPriority()	JMSPriority ヘッダが設定されていないときに発行した場合，4 が返されます。
43		boolean getJMSRedelivered()	JMSRedelivered ヘッダが設定されていないときに発行した場合，false が返されます。
44		Destination getJMSReplyTo()	JMSReplyTo ヘッダが設定されていないときに発行した場合，null が返されます。
45		long getJMSTimestamp()	JMSTimestamp ヘッダが設定されていないときに発行した場合，0 が返されます。
46		java.lang.String getJMSType()	JMSType ヘッダが設定されていないときに発行した場合，null が返されます。
47		long getLongProperty(java.lang.String name)	−
48		java.lang.Object getObjectProperty(java.lang.String name)	−
49		java.util.Enumeration getPropertyNames()	プロパティがない場合は null が返されます。
50		short getShortProperty(java.lang.String name)	−
51		java.lang.String getStringProperty(java.lang.String name)	−
52		boolean propertyExists(java.lang.String name)	−
53		void setBooleanProperty(java.lang.String name,	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。

項番	インタフェース名	メソッド名	機能差
		boolean value)	
54		void setByteProperty(java.lang.String name, byte value)	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。
55		void setDoubleProperty(java.lang.String name, double value)	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。
56		void setFloatProperty(java.lang.String name, float value)	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。
57		void setIntProperty(java.lang.String name, int value)	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。
58		void setJMSCorrelationID(java.lang.String correlationID)	correlationID 引数の文字数が 512 文字を超える場合，JMSException が発生します。
59		void setJMSCorrelationIDAsBytes(byte[] correlationID)	correlationID 引数の文字数が 512 文字を超える場合，JMSException が発生します。
60		void setJMSDeliveryMode(int deliveryMode)	このメソッドで設定した値は実際には使用されません。
61		void setJMSDestination(Destination destination)	このメソッドで設定した値は実際には使用されません。
62		void setJMSExpiration(long expiration)	このメソッドで設定した値は実際には使用されません。
63		void setJMSMessageID(java.lang.String id)	このメソッドで設定した値は実際には使用されません。
64		void setJMSPriority(int priority)	このメソッドで設定した値は実際には使用されません。
65		void setJMSRedelivered(boolean redelivered)	このメソッドで設定した値は実際には使用されません。
66		void setJMSReplyTo(Destination replyTo)	−
67		void setJMSTimestamp(long timestamp)	このメソッドで設定した値は実際には使用されません。
68		void setJMSType(java.lang.String type)	type 引数の文字数が 512 文字を超える場合，JMSException が発生します。
69		void setLongProperty(java.lang.String name, long value)	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。
70		void setObjectProperty(java.lang.String name,	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。

項番	インタフェース名	メソッド名	機能差
		java.lang.Object value)	value 引数の文字数が 512 文字を超える場合，JMSException が発生します。
71		void setShortProperty(java.lang.String name, short value)	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。
72		void setStringProperty(java.lang.String name, java.lang.String value)	name 引数の文字数が 64 文字を超える場合，JMSException が発生します。 value 引数の文字数が 512 文字を超える場合，JMSException が発生します。
73	ObjectMessage	java.io.Serializable getObject()	－
74		void setObject(java.io.Serializable object)	－
75	TextMessage	java.lang.String getText()	－
76		void setText(java.lang.String string)	－

（凡例）

－：機能差はありません。

付録 B　WS-Reliability サポート一覧

サーバ間通信の仕様について，WS-Reliability と Reliable Messaging の違いを中心に説明します。

付録 B.1　通信仕様

Reliable Messaging は，WS-Reliability に規定される HTTP バインディングに従って通信します。Reliable Messaging が Reliable Message を送信する場合，WS-Reliability での Producer は，Reliable Messaging が管理する送信側のキューです。一方，Reliable Messaging が Reliable Message を受信する場合，WS-Reliability での Consumer は，Reliable Messaging が管理する受信側のキューです。

通信に関する制限事項は次のとおりです。

- リプライパターンのうち，Callback および Poll はサポートしません。
- SOAP 1.1 だけをサポートします。SOAP 1.2 はサポートしません。
- トランスポートは HTTP(S)だけをサポートします。
- WSDL インタフェースはサポートしません。

それぞれのリプライパターンでの通信の方法は，WS-Reliability 1.1 のドキュメントを参照してください。

付録 B.2　QoS

QoS（通信品質）について留意しなければならない点は次のとおりです。

OrderedDelivery 時の Acknowledgment の仕様

受信処理性能の向上のため，OrderedDelivery を要求する Reliable Message を受信した場合，その Reliable Message が順序不正（シーケンス番号に抜けのある）のメッセージだった場合も Acknowledgment を返します。このメッセージは Reliable Messaging が保持しておき，順序不正が回復した際に配信されます。

NoDuplicateDelivery を要求しない場合の動作

Reliable Messaging では，NoDuplicateDelivery を要求しない（Request/DuplicateElimination 要素が存在しない）Reliable Message を受信した場合でも，受信した Reliable Message の重複防止を行います。

再送の仕様

Reliable Messaging は，Acknowledgment または「MessageProcessingFailure」以外の RM-Fault の RM-Reply Message が返されるまで Reliable Message を再送します。Reliable Message に設定した有効期限を超過しても再送を継続します。

付録 B.3　リプライパターン

リプライパターンについて留意しなければならない点は次のとおりです。

(1)　Poll リプライパターンに関する動作

Reliable Messaging が Reliable Message の送信側の場合

Reliable Messaging は，Poll リプライパターンを指定した Reliable Message および PollRequest Message は送信しません。

Reliable Messaging が Reliable Message の受信側の場合

Poll リプライパターンを指定された Reliable Message を受信した場合，そのリクエストのレスポンスとして「FeatureNotSupported」の RM-Fault を設定した RM-Reply Message を返信します。その後，PollRequest Message 受信時に，PollRequest Message に指定されたメッセージに対して「FeatureNotSupported」の RM-Fault を設定した RM-Reply Message を返信します。

(2)　Callback リプライパターンに関する動作

Reliable Messaging が Reliable Message の送信側の場合

Reliable Messaging は，Callback リプライパターンを指定した Reliable Message を送信しません。

Reliable Messaging が Reliable Message の受信側の場合

Callback リプライパターンを指定された Reliable Message を受信した場合，そのリクエストのレスポンスとして「FeatureNotSupported」の RM-Fault を設定した RM-Reply Message を返信します。

(3)　有効期限切れのメッセージに対する応答

Reliable Messaging は，受信した Reliable Message のメッセージ有効期限が切れていた場合，その Reliable Message がすでに受信して Acknowledgment を返したメッセージだった場合でも，RM-Fault を返します。

付録 B.4　グループ

グループについて留意しなければならない点は次のとおりです。

(1)　グループの新規作成（切り替え）

Reliable Messaging が Reliable Message の送信側の場合，Reliable Message に設定するグループ ID は，次に示すタイミングで新しいグループに切り替わります。

- InOrder のグループの Reliable Message に「MessageProcessingFailure」および「FeatureNotSupported」以外の RM-Fault が返されたとき
- グループの Reliable Message に「GroupAborted」の RM-Fault が返されたとき

- グループ有効期限が過ぎたとき
- グループ有効期限に Reliable Message の有効期限が追い付いたとき
- シーケンス番号が最大値に到達したとき
- 非永続キューに属するグループを再起動したとき

切り替わり以降に新たに送信されたメッセージは，新しいグループのメッセージとして送信されます。

(2) グループの削除

Reliable Messaging が Reliable Message の受信側の場合，グループの有効期限が超過したときにグループ情報を破棄します。

グループ情報を破棄したあとに，破棄したグループの Reliable Message を受信した場合は，新しいグループの最初のメッセージとして扱います。

(3) グループの閉鎖

Reliable Messaging が Reliable Message の受信側の場合，次に示すタイミングでグループを閉鎖します。

- InOrder のグループの Reliable Message に「MessageProcessingFailure」および「FeatureNotSupported」以外の RM-Fault を返す場合（hrmskipmsg でスキップしたメッセージに「PermanentProcessingFailure」を返す場合を除く）
- InOrder のグループで，滞留しているメッセージが有効期限切れになった場合
- 非永続キューに属するグループについては，Reliable Messaging が再起動した場合

閉鎖されたグループの Reliable Message を受信した場合，「GroupAborted」の RM-Fault を返します。閉鎖されたグループは，グループ削除のタイミングで削除されます。

(4) グループ有効期限の更新

Reliable Messaging が Reliable Message の送信側の場合

同一グループで送信するすべての Reliable Message に，同一のグループ有効期限を設定します。

Reliable Messaging が Reliable Message の受信側の場合

グループの削除を判断するためのグループ有効期限に，グループで受信した最初の Reliable Message のグループ有効期限を使用します。

(5) GroupMaxIdleDuration によるグループの削除

GroupMaxIdleDuration によるグループの削除はサポートしません。GroupMaxIdleDuration が指定された Reliable Message を受信した場合，Reliable Messaging は「FeatureNotSupported」の RM-Fault を返します。

付録 B.5　WS-Reliability のヘッダ要素

Reliable Messaging で送受信するメッセージの SOAP ヘッダ部分には，WS-Reliability プロトコルのヘッダ要素が設定されます。WS-Reliability プロトコルのヘッダ要素は，電文の種別によって設定される要素が異なります。

(1)　Reliable Message の送信時に設定される要素・属性

Reliable Messaging が Reliable Message を送信する時に設定される要素・属性に対する対応を次の表に示します。

表 B-1　Reliable Message 送信時の対応

項番	メッセージヘッダの要素・属性	Reliable Messaging の対応
1	Request	WS-Reliability プロトコルの仕様と同様です。
2	Request/MessageId	WS-Reliability プロトコルの仕様と同様です。
3	Request/MessageId/@groupId※	WS-Reliability プロトコルの仕様と同様です。
4	Request/MessageId/SequenceNum	必ず設定します。
5	Request/MessageId/SequenceNum/@groupExpiryTime	必ず設定します。
6	Request/MessageId/SequenceNum/ @groupMaxIdleDuration	設定しません。
7	Request/MessageId/SequenceNum/@number	WS-Reliability プロトコルの仕様と同様です。
8	Request/MessageId/SequenceNum/@last	設定しません。
9	Request/ExpiryTime	WS-Reliability プロトコルの仕様と同様です。
10	Request/ReplyPattern	WS-Reliability プロトコルの仕様と同様です。
11	Request/ReplyPattern/Value	Response だけ設定します。 Callback および Poll は設定しません。
12	Request/ReplyPattern/ReplyTo	設定しません。
13	Request/ReplyPattern/ReplyTo/@reference-scheme	設定しません。
14	Request/ReplyPattern/ReplyTo/any	設定しません。
15	Request/ReplyPattern/ReplyTo/BareURI	設定しません。
16	Request/AckRequested	必ず設定します。
17	Request/DuplicateElimination	必ず設定します。
18	Request/MessageOrder	WS-Reliability プロトコルの仕様と同様です。

注※

Reliable Messaging で設定する Request/MessageId/@groupId のフォーマットは次のとおりです。

mid://<システム名><グループID作成時刻><グループID通番>@<自マシンのIPアドレスから作成した数値>

グループ ID 作成時刻の出力形式は「YYYYMMDDhhmmssSSS」です。

グループ ID 通番は 4 けた固定です。例えば，グループ通番が 1 の場合は「0001」が設定されます。

(2) Reliable Message の受信時に設定される要素・属性

Reliable Messaging が受信した Reliable Message に設定される要素に対する対応と，指定できる値の範囲を次の表に示します。

表 B–2 Reliable Message 受信時の対応

項番	メッセージヘッダの要素・属性	Reliable Messaging の対応※1	指定できる値の範囲※2
1	Request	WS-Reliability プロトコルの仕様と同様です。	−
2	Request/MessageId	WS-Reliability プロトコルの仕様と同様です。	−
3	Request/MessageId/@groupId	WS-Reliability プロトコルの仕様と同様です。	0～256 文字
4	Request/MessageId/SequenceNum	設定を省略したメッセージには RM-Fault を返します。	−
5	Request/MessageId/SequenceNum/@groupExpiryTime	設定を省略したメッセージには RM-Fault を返します。	1970-01-01T00:00:00Z～9999-12-31T23:59:59Z
6	Request/MessageId/SequenceNum/@groupMaxIdleDuration	設定したメッセージには RM-Fault を返します。	−
7	Request/MessageId/SequenceNum/@number	WS-Reliability プロトコルの仕様と同様です。	−
8	Request/MessageId/SequenceNum/@last	設定したメッセージには RM-Fault を返します。	−
9	Request/ExpiryTime	WS-Reliability プロトコルの仕様と同様です。	1970-01-01T00:00:00Z～9999-12-31T23:59:59Z
10	Request/ReplyPattern	WS-Reliability プロトコルの仕様と同様です。	−
11	Request/ReplyPattern/Value	Response だけ設定します。Callback または Poll を設定したメッセージには RM-Fault を返します。	−
12	Request/ReplyPattern/ReplyTo	設定された値は無視します。	−
13	Request/ReplyPattern/@reference-scheme	設定された値は無視します。	−
14	Request/ReplyPattern/any	設定された値は無視します。	−

項番	メッセージヘッダの要素・属性	Reliable Messaging の対応※1	指定できる値の範囲※2
15	Request/ReplyPattern/**BareURI**	設定された値は無視します。	－
16	Request/**AckRequested**	WS-Reliability プロトコルの仕様と同様です。	－
17	Request/**DuplicateElimination**	WS-Reliability プロトコルの仕様と同様です。	－
18	Request/**MessageOrder**	WS-Reliability プロトコルの仕様と同様です。	－

（凡例）

－：該当しません。

注※1

RM-Fault を返す場合，「FeatureNotSupported」を設定した RM-Reply Message が返信されます。

注※2

指定できる値の範囲を超えている要素または属性を設定している ReliableMessage を受信した場合，次の表に示す RM-Fault が設定された RM-Reply Message を返信します。

表 B-3　設定範囲を超えた場合に返信される RM-Fault

メッセージヘッダの要素・属性	RM-Fault
Request/MessageId/**@groupId**	InvalidMessageId
Request/MessageId/**@groupExpiryTime**	InvalidMessageParameters
Request/**ExpiryTime**	InvalidExpiryTime

(3)　RM-Reply Message の送信時に設定される要素・属性

Reliable Messaging が RM-Reply Message を送信する時に設定される要素に対する対応を次の表に示します。

表 B-4　RM-Reply Message 送信時の対応

項番	メッセージヘッダの要素と属性	Reliable Messaging の対応
1	**Response**	WS-Reliability プロトコルの仕様と同様です。
2	Response/**NonSequenceReply**	設定しません。
3	Response/NonSequenceReply/**@groupId**	設定しません。
4	Response/NonSequenceReply/**@fault**	設定しません。
5	Response/**SequenceReplies**	必ず一つは設定します。
6	Response/SequenceReplies/**@groupId**	WS-Reliability プロトコルの仕様と同様です。

項番	メッセージヘッダの要素と属性	Reliable Messaging の対応
7	Response/SequenceReplies/**ReplyRange**	一つだけ設定します。
8	Response/SequenceReplies/ReplyRange/**@from**	@to と同じ値を設定します。
9	Response/SequenceReplies/ReplyRange/**@to**	@from と同じ値を設定します。
10	Response/SequenceReplies/ReplyRange/**@fault**	WS-Reliability プロトコルの仕様と同様です。

(4) RM-Reply Message の受信時に設定される要素・属性

Reliable Messaging が受信した RM-Reply Message に設定される要素に対する対応と，指定できる値の範囲を次の表に示します。

表 B–5 RM-Reply Message 受信時の対応一覧

項番	メッセージヘッダの要素と属性	Reliable Messaging の対応	指定できる値の範囲※
1	**Response**	WS-Reliability プロトコルの仕様と同様です。	−
2	Response/**NonSequenceReply**	設定されたメッセージは破棄します。	−
3	Response/NonSequenceReply/**@groupId**	設定されたメッセージは破棄します。	−
4	Response/NonSequenceReply/**@fault**	設定されたメッセージは破棄します。	−
5	Response/**SequenceReplies**	一つだけ設定したメッセージには対応します。省略された，または複数設定されたメッセージは破棄します。	−
6	Response/SequenceReplies/**@groupId**	WS-Reliability プロトコルの仕様と同様です。	0～256 文字
7	Response/SequenceReplies/**ReplyRange**	一つだけ設定したメッセージには対応します。省略された，または複数設定されたメッセージは破棄します。	−
8	Response/SequenceReplies/ReplyRange/**@from**	@to と同じ値を設定したメッセージには対応します。@to と異なる値が設定されたメッセージは破棄します。	−
9	Response/SequenceReplies/ReplyRange/**@to**	@from と同じ値を設定したメッセージには対応します。@from と異なる値が設定されたメッセージは破棄します。	−

項番	メッセージヘッダの要素と属性	Reliable Messaging の対応	指定できる値の範囲※
10	Response/SequenceReplies/ReplyRange/@fault	WS-Reliability プロトコルの仕様と同様です。	－

（凡例）

－：該当しません。

注※

指定できる値の範囲を超えている要素または属性を設定している RM-Reply Message を受信した場合，そのメッセージを破棄します。

(5) PollRequest Message の送受信時に設定される要素・属性

Reliable Messaging では，送信，受信ともに PollRequest Message はサポートしません。PollRequest Message を受信した場合は，メッセージ内で指定しているメッセージに対して「FeatureNotSupported」の RM-Fault を設定した RM-Reply Message を返信します。

なお，Reliable Messaging では，PollRequest Message に指定された最初のメッセージに対して「FeatureNotSupported」の RM-Fault を返します。

(6) Reliable Message と RM-Reply Message の同梱

Reliable Messaging は，Reliable Message と RM-Reply Message が同梱されているメッセージに対して，WS-Reliability プロトコルの仕様とは異なる対応をします。

Reliable Message と RM-Reply Message が同梱されているメッセージを受信した場合，Reliable Message に対して「FeatureNotSupported」の RM-Fault を設定した RM-Reply Message の返信を行います。さらに，RM-Reply Message に対してはメッセージを破棄します。

(7) [XML Schema Part 2]dateTime の扱い

Reliable Messaging は，groupExpiryTime や ExpiryTime を表す[XML Schema Part2]dateTime の値に，次の制限を持ちます。

- 送信時は，小数点以下の値は削除してメッセージに設定します。
- 受信時は，小数点以下の値は切り上げて処理します。ただし，小数点以下 3 けたまでの数値がすべて 0 の場合，小数点以下の数値を切り捨てます。

付録 B.6 SOAP アタッチメント

他システムと Reliable Messaging とでメッセージを送受信する場合，メッセージ本体（ペイロード）をプリミティブなバイト配列で SOAP アタッチメントに設定します。MIME パートを含む SOAP メッセージの構成を次の図に示します。

図 B-1　MIME パートを含む SOAP メッセージの構成

メッセージを送信する場合

Reliable Messaging から他システムへメッセージを送信する場合，メッセージ本体の情報を SOAP アタッチメントにプリミティブなバイト配列で設定します。

メッセージを受信する場合

他システムから Reliable Messaging へメッセージを送信する場合，メッセージ本体の情報を SOAP アタッチメントにプリミティブなバイト配列で設定してください。なお，Content-Type が text/plain の場合，Content-Type の charset を UTF-8 に指定してください。SOAP ボディにメッセージ本体の情報を設定すると，Reliable Messaging はメッセージ本体の情報を取得できません。

Reliable Messaging がメッセージを送信するときに，SOAP アタッチメントを設定する MIME パートの MIME ヘッダに設定する内容を次の表に示します。

表 B-6　メッセージ送信時の MIME ヘッダ要素一覧

項番	MIME ヘッダ要素	説明
1	Content-Type	SOAP アタッチメントに設定したメッセージ本体のデータ型を設定します。Content-Type が text/plain の場合，バイト配列を文字コードが UTF-8 のデータとして扱います。文字コードが UTF-8 以外の値を指定してキュー間転送した場合，受信側で同じ値を取得できません。

項番	MIME ヘッダ要素	説明
2	Content-Transfer-Encoding	メッセージ本体を送信するときのエンコーディング方式を設定します。binary を設定します。

付録 B.7　Reliable Messaging が返信するフォルトコード一覧

Reliable Messaging は，メッセージの受信時に障害が発生した場合，WS-Reliability に規定されたフォルトメカニズムに従って，RM-Fault を返します。Response リプライパターンのメッセージへの応答で RM-Fault を返す場合は，SOAP Fault も同時に返します。その際，HTTP ステータスコードに返す値は SOAP 通信基盤の仕様に従います。

Reliable Messaging が返す RM-Fault のフォルトコードを次の表に示します。

表 B–7　Reliable Messaging が返す RM-Fault のフォルトコード

種別	フォルトコード	障害内容
RMFault	InvalidRequest	受信した Reliable Message に次の要素がありません。 • ExpiryTime • ReplyPattern または，次の要素が二つ以上含まれています。 • MessageId • ExpiryTime • ReplyPattern
		受信した Reliable Message の要素が，次に示す順序に従っていません。 • MessageId • ExpiryTime • ReplyPattern
		MessageOrder 要素があるのに，AckRequested 要素と DuplicateElimination 要素がありません。
		SOAP:mustUnderstand 属性がありません。または，SOAP:mustUnderstand 属性の値は「1」以外です。
	InvalidPollRequest	該当しません。
	InvalidMessageId	受信した Reliable Message で，MessageId 要素に含まれる要素や属性が不正です。 • groupId 属性の指定値が 256 バイトを超えています。 • groupId 属性の指定値が URI[RFC2396]と一致しません。 • SequenceNum 要素が 2 個以上指定されています。
	InvalidMessageParameters	受信した Reliable Message の属性で，次のような不正があります。 • groupExpiryTime 属性と groupMaxIdleDuration 属性が両方ともありません。

種別	フォルトコード	障害内容
		• groupExpiryTime 属性と groupMaxIdleDuration 属性が両方とも指定されています。 • groupExpiryTime または groupMaxIdleDuration の指定値が，時刻または時間として読み取れません。 • number 属性の値が 0～18446744073709551615 の範囲外です。
		受信した Reliable Message の groupExpiryTime 属性の時刻が ExpiryTime 要素の時刻と同じか，またはそれ以前の時刻です。
		受信した Reliable Message の groupExpiryTime 属性の時刻が，以前受信した同じグループのメッセージの最大 ExpiryTime の時刻と同じか，またはそれ以前の時刻です。
	InvalidReplyPattern	受信した Reliable Message で，ReplyPattern 要素に含まれる要素が不正です。 ・Value 要素がありません。 ・Value 要素が二つ以上含まれています。 ・Value 要素に Response，Callback，および Poll 以外の文字列が指定されています。
	InvalidExpiryTime	受信した Reliable Message で，ExpiryTime 要素の指定値が時刻として読み取れません。
		受信した Reliable Message で，ExpiryTime 要素から取得した時刻が有効期限切れです。
	FeatureNotSupported	受信した Reliable Message で，SequenceNum 要素が省略されています。
		受信した Reliable Message で，groupMaxIdleDuration 属性が指定されています。
		受信した Reliable Message で，last 属性が指定されています。
		受信した Reliable Message で，ReplyPattern/Value 要素に Callback または Poll が指定されています。
		PollRequest Message を受信しました。
		Reliable Message と RM-Reply Message を同梱した SOAP メッセージを受信しました。
	PermanentProcessingFailure	スキップ済みの受信待ちメッセージを受信しました。
		指定されたキューがないか，またはローカルキューではありません。
		受信処理中に永続的な内部障害が発生しました。
		指定されたキューが閉塞状態です。
	MessageProcessingFailure	配送待ちメッセージがキューのメッセージ最大数に達しているなどの理由で，受信キューにメッセージを登録できません。
		指定されたキューが受信抑止中です。

種別	フォルトコード	障害内容
		受信処理中に一時的な内部障害が発生しました。
	GroupAborted	障害が発生して閉鎖されたグループでメッセージを受信しました。
		指定されたグループは，指定されたキュー以外に属するグループです。
		受信処理中にグループが有効期限切れになり，削除されました。
		非永続キューのグループが再起動によって閉鎖されました。

なお，Reliable Messaging は，SOAP Fault だけを返す場合があります。Reliable Messaging が返す SOAP Fault のフォルトコードを次の表に示します。

表 B-8　Reliable Messaging が返す SOAP Fault のフォルトコード

種別	フォルトコード	障害文字列（faultstring）	障害内容
SOAP Fault	Server	Exception occurred at the server side.	Reliable Messaging が閉塞状態のときにメッセージを受信しました。
			Reliable Messaging が管理状態か，または開始中状態のときメッセージを受信しました。
			受信処理中に内部障害が発生しました。
	Client	Invalid message was transmitted by client side.	受信した SOAP メッセージに WS-Reliability のヘッダが指定されていません（プロトコルが不明の SOAP メッセージです）。
			正常に RM-Fault を返せない不正なメッセージを受信しました。

付録 C　サンプルアプリケーション

Reliable Messaging が提供するサンプルアプリケーションの処理の流れおよび Component Container で実行する手順について説明します。

次に示すサンプルアプリケーションが，Reliable Messaging のインストール先の samples ディレクトリに格納されています。

SessionBean1

ローカルキューを使って，メッセージの送信と受信を実行します。

永続版リソースアダプタの場合と非永続版リソースアダプタの場合で実行する手順が異なります。

SessionBean2

キュー間転送を使って，メッセージの送信と受信を実行します。

永続版リソースアダプタの場合だけ提供されます。

なお，SessionBean1 を動作させるための環境構築のサンプル手順については，「付録 C.1　環境構築のサンプル手順」を参照してください。

付録 C.1　環境構築のサンプル手順

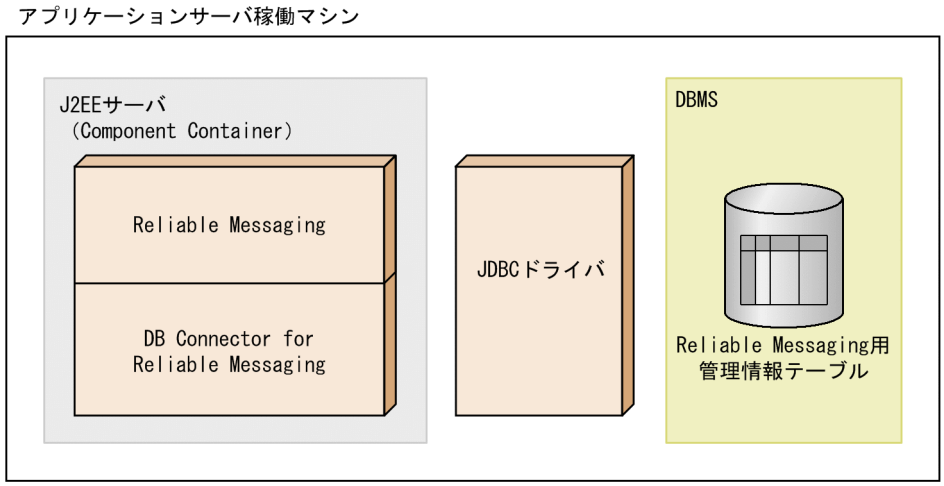
SessionBean1 を動作させるための Reliable Messaging の環境構築のサンプル手順について説明します。

なお，J2EE サーバやリソースアダプタの操作では，Management Server から行う方法とコマンドを使用して行う方法がありますが，ここではコマンドを使用して行う方法について説明します。

(1)　前提となるマシンとソフトウェア構成

ここで説明するサンプル手順は，次に示すマシンとソフトウェア構成を前提として，Reliable Messaging の環境を構築します。

図 C-1　前提となるマシンとソフトウェア構成



各構成要素の項目を次の表に説明します。

表 C-1　構成要素の項目

構成要素名	項目	項目情報
Application Server 稼働マシン	OS	Windows
Application Server（J2EE サーバ）	インストールディレクトリ	C:¥Program Files¥Hitachi¥Cosminexus
	J2EE サーバ名称	MyServer
	CORBA ネーミングサービス（nameserv）のポート番号	900
Reliable Messaging	表示名	Cosminexus_Reliable_Messaging
	リソースアダプタ種別	永続版リソースアダプタ
	Reliable Messaging のシステム名	CRM
	利用トランザクション種別	ローカルトランザクション
	コネクションプール利用数	2（最大/最小）
	作成するキュー	キュー種別：ローカルキュー キューの永続性：永続 キュー名：localq1 表示名：QUEUE1
DB Connector for Reliable Messaging	表示名	DB_Connector_for_HiRDB_Type4_Cosminexus_RM
	種別	HiRDB Type4 JDBC Driver 用（ローカルトランザクション）
JDBC ドライバ	ドライバ種別	HiRDB Type4 JDBC Driver
DBMS	DB 種別	HiRDB Single Server

構成要素名	項目	項目情報
	ポート番号	22200
	Reliable Messaging 用の管理情報テーブルを作成する RD エリア名	RDDATA10

(2)　事前準備

サンプル手順を利用して Reliable Messaging の環境を構築する前に，次の作業を実施しておく必要があります。

(a)　インストール

Reliable Messaging および Reliable Messaging の前提製品をすべてインストールします。詳細については，「3.2　Reliable Messaging のインストール」を参照してください。

(b)　環境変数の設定

Reliable Messaging が動作するマシンで，次に示す環境変数を設定します。詳細については，「3.3.2　環境変数の設定」を参照してください。

表 C-2　設定する環境変数

環境変数	設定値	説明
HRMDIR	C:¥Program Files¥Hitachi¥Cosminexus¥RM	−
HRM_SYSTEM_NAME	CRM	−
HRM_CMD_PORT	900	CORBA ネーミングサービス（nameserv）のポート番号です。
HRM_CMD_HOST	localhost	−
PATH	%PATH%;%HRMDIR%¥bin	コマンドが格納されたディレクトリです。

（凡例）

−：特にありません。

(c)　DBMS（HiRDB）のインストールと初期設定

HiRDB サーバ，HiRDB クライアント，および SQL Executer をインストールして，初期設定します。詳細については，「3.4.1(1)(a)　HiRDB の初期設定」および「3.4.1(2)　DB クライアントの設定」を参照してください。

(d)　DBMS（HiRDB）のユーザ権限の付与とスキーマの定義

HiRDB サーバで，ユーザ権限の付与とスキーマの定義を実施します。詳細については，「3.4.1(1)(b)　HiRDB のユーザ権限の付与」および「3.4.1(1)(c)　HiRDB のスキーマの定義」を参照してください。

(e)　HiRDB Type4 JDBC Driver の設定

HiRDB Type4 JDBC Driver の設定をします。詳細については，「3.4.7(2)　HiRDB Type4 JDBC Driver の設定」を参照してください。

(f)　J2EE サーバ（Application Server）の設定

J2EE サーバ（Application Server）をセットアップします。詳細については，「3.4.3　J2EE サーバ（Application Server）の設定」を参照してください。

(g)　その他の設定

環境構築のための作業ディレクトリとして「D:¥work」を作成します。作成したディレクトリには，読み取り権限および書き込み権限を設定します。

(3)　環境構築手順

Reliable Messaging の環境構築のサンプル手順を次に示します。

(a)　Reliable Messaging の管理情報テーブルの作成

1. 次のコマンドを実行して，テーブル作成用 SQL ファイルを作業ディレクトリにコピーします。

```
copy "C:¥Program Files¥Hitachi¥Cosminexus¥RM¥sql¥createtableshirdb.sql" D:¥work
```

2. コピーした SQL ファイル（D:¥work¥createtableshirdb.sql）をテキストエディタで開き，ファイル中の次の記述を置き換えて保存します。

   <RMSystemName> → CRM

   <RMAREA> → RDDATA10

3. 次のコマンドを実行して，HiRDB SQL Executer を開始します。

```
pdsqlw -u <DBへの接続ユーザ名>/<パスワード> -h localhost -n 22200
```

4. HiRDB SQL Executer の［ファイル］メニューから［ファイルから実行］を選択し，「D:¥work¥createtableshirdb.sql」を指定して実行します。

(b)　J2EE サーバの起動

次のコマンドを実行して，J2EE サーバを起動します。

```
cjstartsv MyServer
```

(c)　Reliable Messaging のプロパティ定義

1. 次のコマンドを実行して，作業ディレクトリに Reliable Messaging の Connector 属性ファイルのテンプレートをコピーします。

```
copy "C:¥Program Files¥Hitachi¥Cosminexus¥RM¥conf¥rm_prop.xml" D:¥work
```

2. コピーした Connector 属性ファイルのテンプレート（D:¥work¥rm_prop.xml）をテキストエディタで開き，次のように設定します。

コンフィグレーションプロパティ（<config-property-name>タグ）

プロパティ名	設定値
RMSystemName	CRM
RMLinkedDBConnectorName	DB_Connector_for_HiRDB_Type4_Cosminexus_RM

実行時プロパティ（<property>タグ）

プロパティ名	設定値
MaxPoolSize	2
MinPoolSize	2
User	<DB ユーザ>
Password	<DB パスワード>

トランザクションサポートレベル（<transaction-support>タグ）

タグ名	設定値
<transaction-support>	LocalTransaction

(d) DB Connector for Reliable Messaging のプロパティ定義

1. 次のコマンドを実行して，作業ディレクトリに DB Connector for Reliable Messaging の Connector 属性ファイルのテンプレートをコピーします。

```
copy "C:¥Program Files¥Hitachi¥Cosminexus¥CC¥admin¥templates¥DBConnector_HiRDB_Type4_CP_Cosminexus_RM_cfg.xml"  D:¥work
```

2. コピーした Connector 属性ファイルのテンプレート（D:¥work¥DBConnector_HiRDB_Type4_CP_Cosminexus_RM_cfg.xml）をテキストエディタで開き，次のように設定します。

コンフィグレーションプロパティ（<config-property-name>タグ）

プロパティ名	設定値
linkedResourceAdapterName	Cosminexus_Reliable_Messaging
description	22200
DBHostName	localhost
environmentVariables	PDSWAITTIME=600;PDCWAITTIME=600;PDSWATCHTIME=0

実行時プロパティ（<property>タグ）

プロパティ名	設定値
MaxPoolSize	2
MinPoolSize	2
User	<DB ユーザ>
Password	<DB パスワード>

トランザクションサポートレベル（<transaction-support>タグ）

タグ名	設定値
<transaction-support>	LocalTransaction

(e) DB Connector for Reliable Messaging のインポート

次のコマンドを実行して，DB Connector for Reliable Messaging をインポートします。

```
cjimportres MyServer -type rar -f ”C:¥Program Files¥Hitachi¥Cosminexus¥CC¥DBConnector¥ReliableMessaging¥DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar”
```

(f) Reliable Messaging のインポート

次のコマンドを実行して，Reliable Messaging をインポートします。

```
cjimportres MyServer -type rar -f ”C:¥Program Files¥Hitachi¥Cosminexus¥RM¥lib¥reliablemessaging.rar”
```

(g) DB Connector for Reliable Messaging のプロパティ設定

次のコマンドを実行して，DB Connector for Reliable Messaging のプロパティを設定します。

```
cjsetresprop MyServer -type rar -resname DB_Connector_for_HiRDB_Type4_Cosminexus_RM -c D:¥work¥DBConnector_HiRDB_Type4_CP_Cosminexus_RM_cfg.xml
```

(h) Reliable Messaging のプロパティ設定

次のコマンドを実行して，Reliable Messaging のプロパティを設定します。

```
cjsetresprop MyServer -type rar -resname Cosminexus_Reliable_Messaging -c D:¥work¥rm_prop.xml
```

(i) DB Connector for Reliable Messaging のデプロイ

次のコマンドを実行して，DB Connector for Reliable Messaging をデプロイします。

```
cjdeployrar MyServer -resname DB_Connector_for_HiRDB_Type4_Cosminexus_RM
```

(j)　Reliable Messaging のデプロイ

次のコマンドを実行して，Reliable Messaging をデプロイします。

```
cjdeployrar MyServer -resname Cosminexus_Reliable_Messaging
```

(k)　DB Connector for Reliable Messaging の開始

次のコマンドを実行して，DB Connector for Reliable Messaging を開始します。

```
cjstartrar MyServer -resname DB_Connector_for_HiRDB_Type4_Cosminexus_RM
```

(l)　Reliable Messaging の接続テスト

次のコマンドを実行して，Reliable Messaging の接続テストを実施します。

```
cjtestres MyServer -type rar -resname Cosminexus_Reliable_Messaging
```

接続テストに失敗した場合は，出力されるメッセージに従って対処し，再度接続テストを実施してください。

(m)　Reliable Messaging の開始

次のコマンドを実行して，Reliable Messaging を開始します。

```
cjstartrar MyServer -resname Cosminexus_Reliable_Messaging
```

(n)　キューの作成

次のコマンドを実行して，Reliable Messaging のキューを作成します。

```
hrmmkque -t local -x QUEUE1 localq1
```

(o)　Reliable Messaging の状態遷移

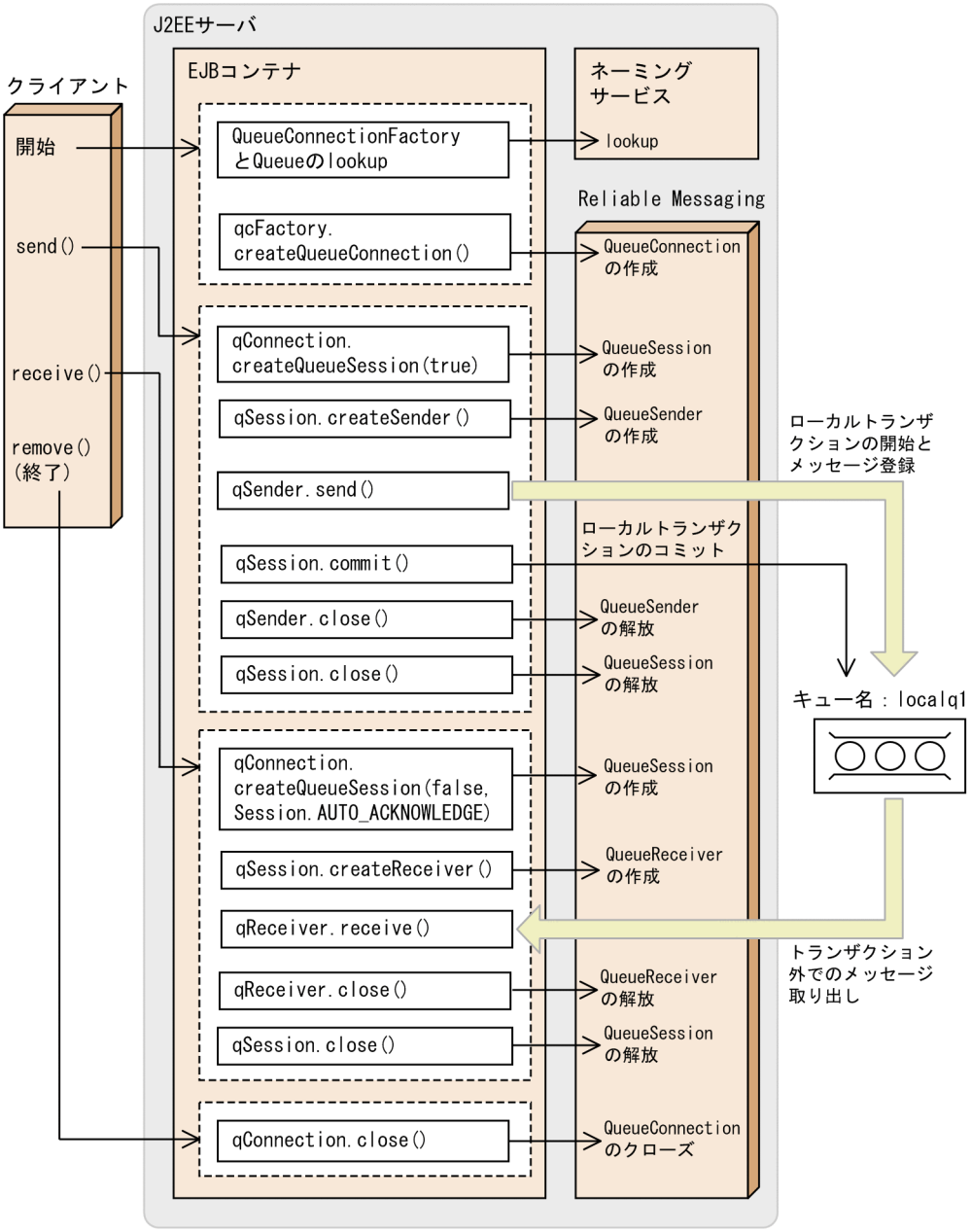
次のコマンドを実行して，Reliable Messaging を実行状態にします。

```
hrmstart
```

付録 C.2　SessionBean1 の処理の流れ

SessionBean1 の処理の流れを次の図に示します。

図 C-2　サンプルアプリケーション（SessionBean1）の処理の流れ



付録 C.3　SessionBean1 の実行手順（永続版リソースアダプタの場合）

永続版リソースアダプタの場合に SessionBean1 を Component Container で実行する手順について説明します。

1. サンプルアプリケーションの格納ディレクトリ（%HRMDIR%¥samples¥SessionBean1）をディレクトリごと任意の別ディレクトリにコピーしてください。以降ではコピーした先の SessionBean1 ディレクトリを作業ディレクトリと呼びます。

2. 作業ディレクトリに移動します。

3. 実行ファイルの編集

   deployApp.bat，testClient.bat および unDeployApp.bat（UNIX の場合，それぞれ deployApp，testClient および unDeployApp）で定義されている環境変数 SERVERNAME を J2EE サーバの名称と同じ値に修正します。

4. コンパイル

   Windows の場合

   compileBean.bat → compileClient.bat の順に実行します。

   UNIX の場合

   compileBean → compileClient の順に実行します。

   作業ディレクトリに jmssample1.ear という J2EE アプリケーション（アプリケーション名 JMSSample1 の EAR ファイル）が作成されます。

5. サーバ管理コマンドで，DB Connector for Reliable Messaging のリソースアダプタをインポートします。

6. サーバ管理コマンドで，Reliable Messaging のリソースアダプタをインポートします。Reliable Messaging の表示名は，デフォルトの"Cosminexus_Reliable_Messaging"から変更しないでください。

7. Reliable Messaging のリソースアダプタのプロパティを設定します。なお，QueueConfigFileName プロパティの設定は不要です。プロパティの詳細については，「6. コンフィグレーションプロパティ」を参照してください。

8. DB Connector for Reliable Messaging のリソースアダプタのプロパティを設定します。プロパティの詳細については，「6. コンフィグレーションプロパティ」を参照してください。

9. DB Connector for Reliable Messaging のリソースアダプタをデプロイし，開始します。

10. Reliable Messaging のリソースアダプタをデプロイし，開始します。

11. ローカルキューを作成するため，hrmmkque コマンドを次のとおり実行します。

```
hrmmkque -t local -x QUEUE1 localq1
```

12. deployApp.bat（UNIX の場合，deployApp）を実行し，J2EE アプリケーション（JMSSample1）の(1)インポート，(2)設定，(3)開始，(4)アプリケーションの状態表示および(5)EJB のスタブおよびインタフェースファイルの取得を実行します。

    (4)では J2EE アプリケーション（JMSSample1）の状態が"running"になっていることを確認できます。なお(1)～(5)はそれぞれ Component Container の(1)cjimportapp，(2)cjsetappprop，(3)cjstartapp，(4)cjlistapp および(5)cjgetstubsjar コマンドをバッチファイル中で実行しています。

13. Reliable Messaging が管理状態である場合は hrmstart コマンドを実行して，実行状態にします。

14. testClient.bat（UNIX の場合，testClient）を実行します。

J2EE サーバを開始しているコンソール画面に"The message is: ******** sample put data JMSSample1 ********"が表示されることで実行を確認できます。

付録 C.4　SessionBean1 の実行手順（非永続版リソースアダプタの場合）

非永続版リソースアダプタの場合に SessionBean1 を Component Container で実行する手順について説明します。

1. サンプルアプリケーションの格納ディレクトリ（%HRMDIR%¥samples¥SessionBean1）をディレクトリごと任意の別ディレクトリにコピーしてください。以降ではコピーした先の SessionBean1 ディレクトリを作業ディレクトリと呼びます。

2. 作業ディレクトリに移動します。

3. 実行ファイルの編集

   deployApp.bat，testClient.bat および unDeployApp.bat（UNIX の場合，それぞれ deployApp，testClient および unDeployApp）で定義されている環境変数 SERVERNAME を J2EE サーバの名称と同じ値に修正します。

4. コンパイル

   Windows の場合

   compileBean.bat → compileClient.bat の順に実行します。

   UNIX の場合

   compileBean → compileClient の順に実行します。

   作業ディレクトリに jmssample1.ear という J2EE アプリケーション（アプリケーション名 JMSSample1 の EAR ファイル）が作成されます。

5. サーバ管理コマンドで，Reliable Messaging のリソースアダプタをインポートします。Reliable Messaging の表示名は，デフォルトの"Cosminexus_Reliable_Messaging"から変更しないでください。

6. リソースアダプタのプロパティを設定します。QueueMakeFileName プロパティに，作業ディレクトリにあるキュー作成ファイル QueueMake.properties を完全パスで指定します。また，そのほかのプロパティも設定します。なお，QueueConfigFileName プロパティの設定は不要です。プロパティの詳細については，「6.　コンフィグレーションプロパティ」を参照してください。

7. リソースアダプタをデプロイし，開始します。

8. deployApp.bat（UNIX の場合，deployApp）を実行し，J2EE アプリケーション（JMSSample1）の(1)インポート，(2)設定，(3)開始，(4)アプリケーションの状態表示および(5)EJB のスタブおよびインタフェースファイルの取得を実行します。

(4)では J2EE アプリケーション（JMSSample1）の状態が"running"になっていることを確認できます。なお(1)～(5)はそれぞれ Component Container の(1)cjimportapp，(2)cjsetappprop，(3)cjstartapp，(4)cjlistapp および(5)cjgetstubsjar コマンドをバッチファイル中で実行しています。

9. testClient.bat（UNIX の場合，testClient）を実行します。

J2EE サーバを開始しているコンソール画面に"The message is: ******** sample put data JMSSample1 ********"が表示されることで実行を確認できます。

付録 C.5　SessionBean2 の処理の流れ（永続版リソースアダプタの場合）

キュー間転送を使うサンプルアプリケーション SessionBean2 は，メッセージ送信用とメッセージ受信用の 2 種類のアプリケーションから構成されるシステムです。SessionBean2 は，永続版リソースアダプタの場合だけ提供されます。

送信用サンプルアプリケーション（SessionBean2Send）

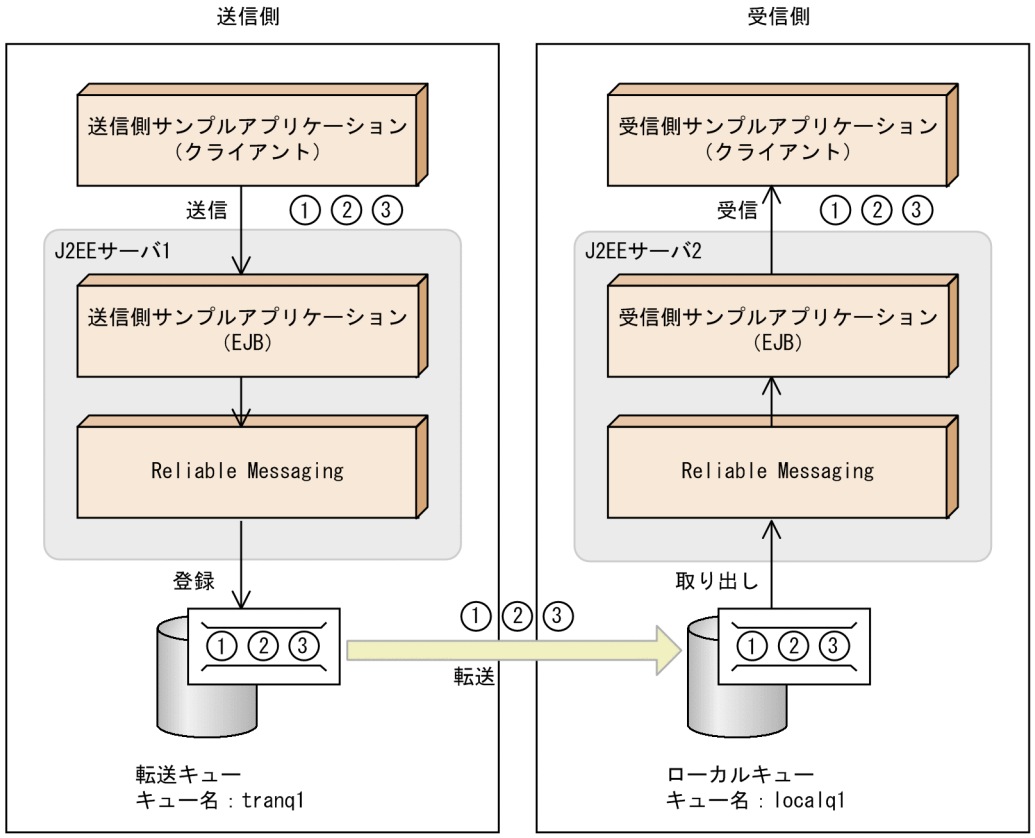
転送キュー（キュー名：tranq1）からローカルキュー（キュー名：localq1）へメッセージを転送します。

受信用サンプルアプリケーション（SessionBean2Receive）

ローカルキュー（キュー名：localq1）からメッセージを受信します。

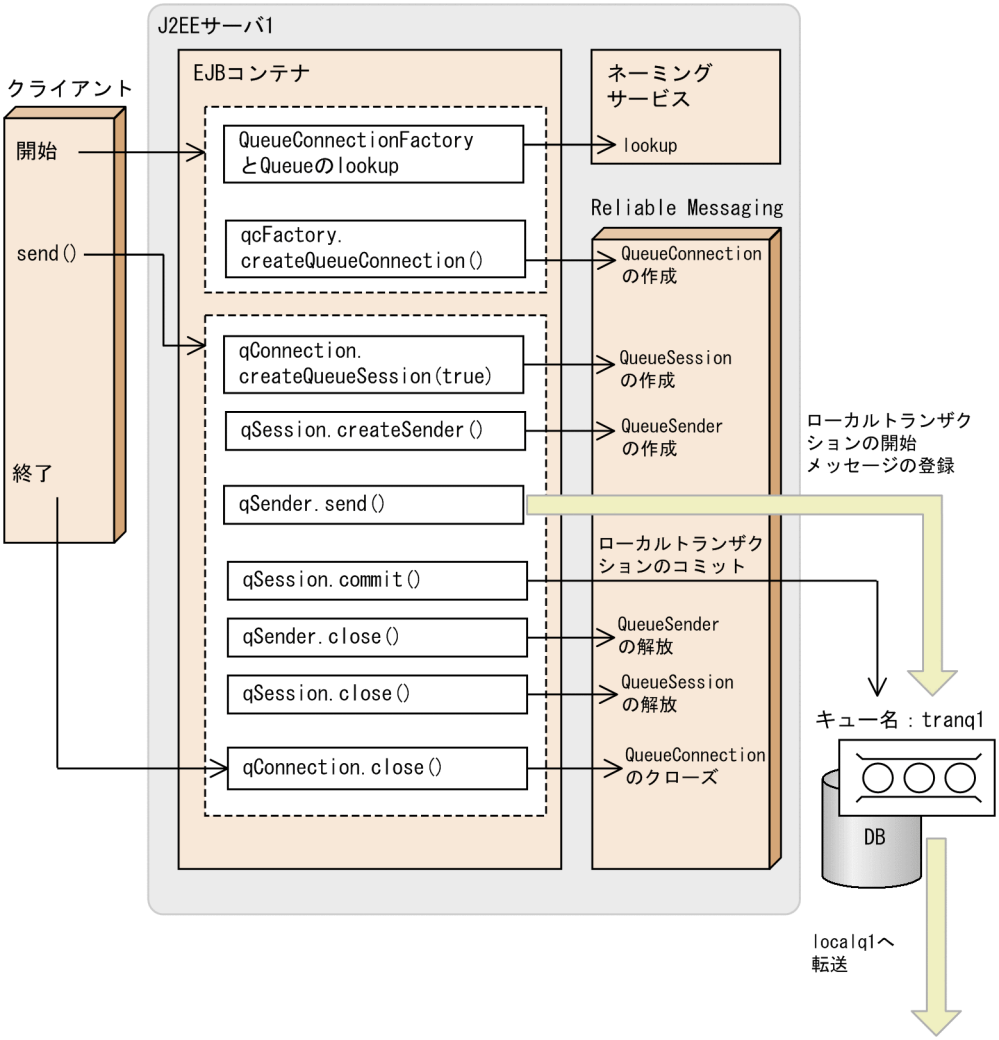
SessionBean2 のシステム構成を次に示します。

図 C-3　サンプルアプリケーション（SessionBean2）のシステム構成



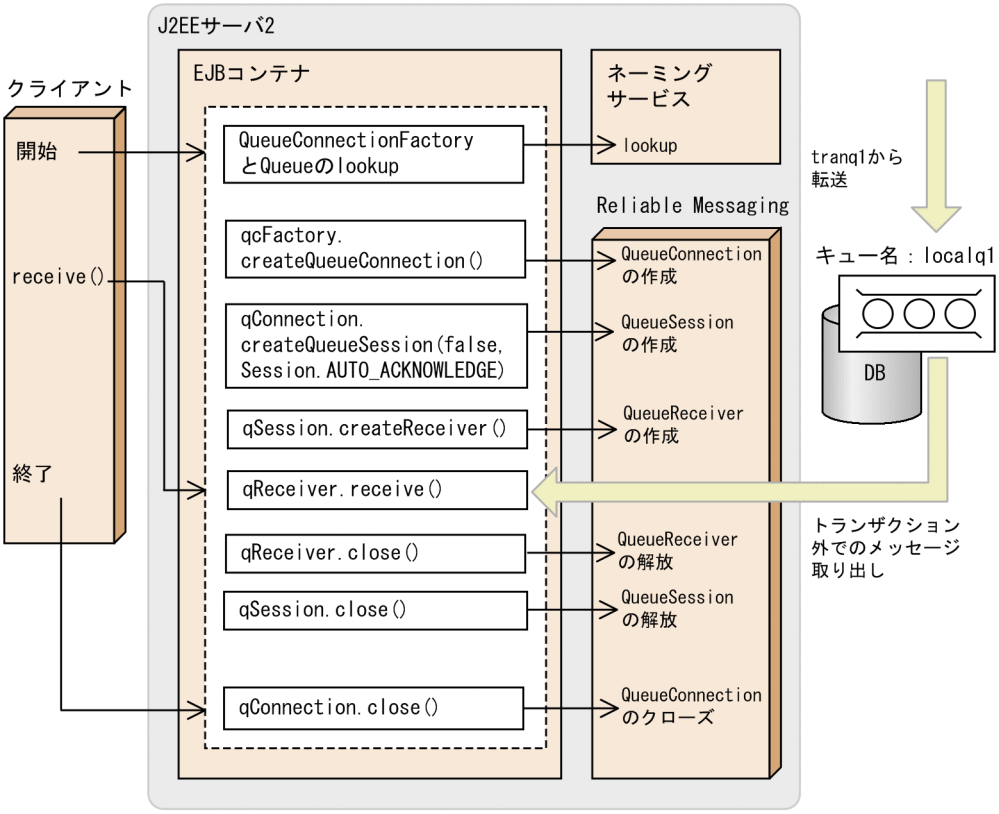
送信用サンプルアプリケーション SessionBean2Send の処理の流れを次の図に示します。

図 C-4　送信用サンプルアプリケーション（SessionBean2Send）の処理の流れ



受信用サンプルアプリケーション SessionBean2Receive の処理の流れを次の図に示します。

図 C-5　受信用サンプルアプリケーション（SessionBean2Receive）の処理の流れ



付録 C.6　SessionBean2 の実行手順（永続版リソースアダプタの場合）

SessionBean2 を実行するために必要な環境設定と実行の手順を説明します。

(1)　環境設定

Component Container の環境設定の手順を説明します。

(a)　送信用サンプルアプリケーション

1. 送信用サンプルアプリケーションの格納ディレクトリ（%HRMDIR%¥samples¥SessionBean2¥Send）を，ディレクトリごと任意の別ディレクトリにコピーしてください。以下ではコピーした先の Send ディレクトリを送信用作業ディレクトリと呼びます。

2. サーバ管理コマンドで，DB Connector for Reliable Messaging のリソースアダプタをインポートします。

3. サーバ管理コマンドで，Reliable Messaging のリソースアダプタをインポートします。Reliable Messaging の表示名は，デフォルトの"Cosminexus_Reliable_Messaging"から変更しないでください。

4. Reliable Messaging のリソースアダプタのプロパティを設定します。なお，QueueConfigFileName プロパティの設定は不要です。プロパティの詳細については，「6.　コンフィグレーションプロパティ」を参照してください。

5. DB Connector for Reliable Messaging のリソースアダプタのプロパティを設定します。プロパティの詳細については，「6.　コンフィグレーションプロパティ」を参照してください。

6. DB Connector for Reliable Messaging のリソースアダプタをデプロイし，開始します。

7. Reliable Messaging のリソースアダプタをデプロイし，開始します。

8. メッセージ送信のあて先を登録するため，hrmmkaddr コマンドを次のとおり実行します。

```
hrmmkaddr -u〈localq1が存在するシステムのあて先アドレス〉 localq1Address
```

9. 転送キューを作成するため，hrmmkque コマンドを次のとおり実行します。

```
hrmmkque -t transmit -a localq1Address -v localq1 -x QUEUE1 tranq1
```

10. Reliable Messaging が管理状態になっている場合，hrmstart コマンドを実行して Reliable Messaging を実行状態にします。

(b)　受信用サンプルアプリケーション

1. 受信用サンプルアプリケーションの格納ディレクトリ（%HRMDIR%¥samples¥SessionBean2¥Receive）を，ディレクトリごと任意の別ディレクトリにコピーしてください。以下ではコピーした先の Receive ディレクトリを受信用作業ディレクトリと呼びます。

2. サーバ管理コマンドで，DB Connector for Reliable Messaging のリソースアダプタをインポートします。

3. サーバ管理コマンドで，Reliable Messaging のリソースアダプタをインポートします。Reliable Messaging の表示名は，デフォルトの"Cosminexus_Reliable_Messaging"から変更しないでください。

4. Reliable Messaging のリソースアダプタのプロパティを設定します。なお，QueueConfigFileName プロパティの設定は不要です。プロパティの詳細については，「6.　コンフィグレーションプロパティ」を参照してください。

5. DB Connector for Reliable Messaging のリソースアダプタのプロパティを設定します。プロパティの詳細については，「6.　コンフィグレーションプロパティ」を参照してください。

6. DB Connector for Reliable Messaging のリソースアダプタをデプロイし，開始します。

7. Reliable Messaging のリソースアダプタをデプロイし，開始します。

8. ローカルキューを作成するため，hrmmkque コマンドを次のとおり実行します。

```
hrmmkque -t local -x QUEUE1 localq1
```

9. Reliable Messaging のキュー間転送用 Web アプリケーションをデプロイし，開始します。

10. Reliable Messaging が管理状態になっている場合，hrmstart コマンドを実行して Reliable Messaging を実行状態にします。

(2) 実行手順

SessionBean2 を Component Container で実行する手順について説明します。

(a) 送信用サンプルアプリケーション

1. 送信用作業ディレクトリに移動します。

2. 実行ファイルの編集

   deployApp.bat，testSendClient.bat および unDeployApp.bat（UNIX の場合，それぞれ deployApp，testSendClient および unDeployApp）で定義されている環境変数 SERVERNAME を J2EE サーバ 1 の名称と同じ値に修正します。

3. コンパイル

   Windows の場合

   compileBean.bat → compileClient.bat の順に実行します。

   UNIX の場合

   compileBean → compileClient の順に実行します。

   送信用作業ディレクトリに jmssample2Send.ear という J2EE アプリケーション（EAR ファイル，アプリケーション名 JMSSample2Send）が作成されます。

4. deployApp.bat（UNIX の場合，deployApp）を実行し，J2EE アプリケーション JMSSample2Send の(1)インポート，(2)設定，(3)開始，(4)アプリケーションの状態表示，(5)EJB のスタブ・インタフェースファイルの取得を行います。(4)で，J2EE アプリケーション JMSSample2Send の状態が"running"になっていることを確認できます。なお(1)～(5)はそれぞれ Component Container の(1)cjimportapp，(2)cjsetappprop，(3)cjstartapp，(4)cjlistapp，(5)cjgetstubsjar の各コマンドをバッチファイル中で実行しています。

5. testSendClient.bat（UNIX の場合，testSendClient）を実行します。

   J2EE サーバを起動しているコンソール画面に以下の文字が表示されることで実行を確認できます。

```
JMSSample2Send sends a message1 to JMSSample2Receive
JMSSample2Send sends a message2 to JMSSample2Receive
JMSSample2Send sends a message3 to JMSSample2Receive
```

(b) 受信用サンプルアプリケーション

1. 受信用作業ディレクトリに移動します。

2. 実行ファイルの編集

   deployApp.bat，testReceiveClient.bat，および unDeployApp.bat（UNIX の場合，それぞれ deployApp，testReceiveClient，および unDeployApp）で定義されている環境変数 SERVERNAME を J2EE サーバ 2 の名称と同じ値に修正します。

3. コンパイル

   Windows の場合

   compileBean.bat → compileClient.bat の順に実行します。

   UNIX の場合

   compileBean → compileClient の順に実行します。

   受信用作業ディレクトリに jmssample2Receive.ear という J2EE アプリケーション（EAR ファイル，アプリケーション名 JMSSample2Receive）が作成されます。

4. deployApp.bat（UNIX の場合，deployApp）を実行し，J2EE アプリケーション JMSSample2Receive の(1)インポート，(2)設定，(3)開始，(4)アプリケーションの状態表示，(5)EJB のスタブ・インタフェースファイルの取得を行います。(4)で，J2EE アプリケーション JMSSample2Receive の状態が"running"になっていることを確認できます。なお(1)～(5)はそれぞれ Component Container の(1)cjimportapp，(2)cjsetappprop，(3)cjstartapp，(4)cjlistapp，(5)cjgetstubsjar の各コマンドをバッチファイル中で実行しています。

5. testReceiveClient.bat(UNIX の場合，testReceiveClient)を実行します。

   J2EE サーバを起動しているコンソール画面に以下の文字が表示されることで実行を確認できます。

```
The message is: ******** JMSSample2Receive receives a message1 from JMSSample2Send ******
**
The message is: ******** JMSSample2Receive receives a message2 from JMSSample2Send ******
**
The message is: ******** JMSSample2Receive receives a message3 from JMSSample2Send ******
**
```

付録 D　PRF トレース取得時のイベント ID

Reliable Messaging では，各処理の入口と出口で PRF トレースを取得します。各処理でのイベント ID，PRF トレースの出力ポイント，およびトレースの取得レベルを説明します。なお，取得レベルは，「A」，「B」で表現されます。「A」レベルは標準，「B」レベルは詳細を意味します。

付録 D.1　永続版リソースアダプタの場合

永続版リソースアダプタの場合の Reliable Messaging では，次の処理の入口と出口で PRF トレースを取得します。

(1)　アプリケーションからのコネクションの取得と解放処理

アプリケーションからのコネクションの取得と解放処理でのイベント ID を次の表に示します。

表 D–1　アプリケーションからのコネクションの取得と解放処理でのイベント ID（永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9360	QueueConnection.createQueueSession の入口	A
0x9361	QueueConnection.createQueueSession の出口	A
0x9362	QueueSession.close の入口	A
0x9363	QueueSession.close の出口	A
0x9364	ManagedConnectionFactory.matchManagedConnection の入口	B
0x9365	ManagedConnectionFactory.matchManagedConnection の出口	B

(2)　J2EE サーバ（Component Container）からのコネクションのアソシエート処理

J2EE サーバ（Component Container）からのコネクションのアソシエート処理でのイベント ID を次の表に示します。

表 D–2　J2EE サーバ（Component Container）からのコネクションのアソシエート処理でのイベント ID（永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9366	ManagedConnection.associateConnection の入口	B
0x9367	ManagedConnection.associateConnection の出口	B

(3) アプリケーションからのメッセージ送受信処理

アプリケーションからのメッセージ送受信処理でのイベント ID を次の表に示します。

表 D-3 アプリケーションからのメッセージ送受信処理でのイベント ID（永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9368	QueueSender.send(msg) の入口	A
0x9369	QueueSender.send(msg) の出口	A
0x936A	QueueSender.send(msg,dlvmd, pri,ttl) の入口	A
0x936B	QueueSender.send(msg,dlvmd, pri,ttl) の出口	A
0x936C	QueueSender.send(que,msg) の入口	A
0x936D	QueueSender.send(que,msg) の出口	A
0x936E	QueueSender.send(que,msg,dlvmd, pri,ttl) の入口	A
0x936F	QueueSender.send(que,msg,dlvmd, pri,ttl) の出口	A
0x9370	QueueReceiver.receive(引数なし) の入口	A
0x9371	QueueReceiver.receive(引数なし) の出口	A
0x9372	QueueReceiver.receive(timeout) の入口	A
0x9373	QueueReceiver.receive(timeout) の出口	A
0x9374	QueueReceiver.receiveNoWait の入口	A
0x9375	QueueReceiver.receiveNoWait の出口	A
0x9376	Enumeration.nextElement の入口	A
0x9377	Enumeration.nextElement の出口	A

(4) J2EE サーバ（Component Container）からのトランザクション制御

J2EE サーバ（Component Container）からのトランザクション制御でのイベント ID を次の表に示します。

表 D-4 J2EE サーバ（Component Container）からのトランザクション制御でのイベント ID（永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9378	XAResource.start の入口	B
0x9379	XAResource.start の出口	B
0x937A	XAResource.end の入口	B

イベント ID	PRF トレース取得ポイント	取得レベル
0x937B	XAResource.end の出口	B
0x937C	XAResource.prepare の入口	B
0x937D	XAResource.prepare の出口	B
0x937E	XAResource.commit の入口	B
0x937F	XAResource.commit の出口	B
0x9380	XAResource.rollback の入口	B
0x9381	XAResource.rollback の出口	B
0x9382	XAResource.forget の入口	B
0x9383	XAResource.forget の出口	B
0x9384	XAResource.recover の入口	B
0x9385	XAResource.recover の出口	B
0x9386	LocalTransaction.begin の入口	B
0x9387	LocalTransaction.begin の出口	B
0x9388	LocalTransaction.commit の入口	B
0x9389	LocalTransaction.commit の出口	B
0x938A	LocalTransaction.rollback の入口	B
0x938B	LocalTransaction.rollback の出口	B

(5) JMS アプリケーションからの JMS セッションの決着処理

JMS アプリケーションからの JMS セッションの決着処理でのイベント ID を次の表に示します。

表 D-5 JMS アプリケーションからの JMS セッションの決着処理でのイベント ID（永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x938C	QueueSession.commit の入口	A
0x938D	QueueSession.commit の出口	A
0x938E	QueueSession.rollback の入口	A
0x938F	QueueSession.rollback の出口	A
0x9390	Message.acknowledge の入口	A
0x9391	Message.acknowledge の出口	A
0x9392	QueueSession.recover の入口	A

イベント ID	PRF トレース取得ポイント	取得レベル
0x9393	QueueSession.recover の出口	A

(6) J2EE サーバ（Component Container）と連携した Message-driven Bean メッセージ配信処理

J2EE サーバ（Component Container）と連携した Message-driven Bean メッセージ配信処理でのイベント ID を次の表に示します。

表 D-6 J2EE サーバ（Component Container）と連携した Message-driven Bean メッセージ配信処理でのイベント ID（永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9394	XAQueueSessionImpl.beforeDelivery の ServerSession.beforeDelivery の呼び出し後	B
0x9395	XAQueueSessionImpl.onMessage の MessageListener.onMessage の呼び出し前	A
0x9396	XAQueueSessionImpl.onMessage の MessageListener.onMessage の呼び出し後	A
0x9397	XAQueueSessionImpl.afterDelivery の ServerSession. afterDelivery の呼び出し前	B

付録 D.2 非永続版リソースアダプタの場合

非永続版リソースアダプタの場合の Reliable Messaging では，次の処理の入口と出口で PRF トレースを取得します。

(1) アプリケーションからのコネクションの取得と解放処理

アプリケーションからのコネクションの取得と解放処理でのイベント ID を次の表に示します。

表 D-7 アプリケーションからのコネクションの取得と解放処理でのイベント ID（非永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9300	QueueConnection.createQueueSession の入口	A
0x9301	QueueConnection.createQueueSession の出口	A
0x9302	QueueSession.close の入口	A
0x9303	QueueSession.close の出口	A
0x9304	ManagedConnection.matchManagedConnection の入口	B
0x9305	ManagedConnection.matchManagedConnection の出口	B

(2) J2EE サーバ（Component Container）からのコネクションのアソシエート処理

J2EE サーバ（Component Container）からのコネクションのアソシエート処理でのイベント ID を次の表に示します。

表 D-8 J2EE サーバ（Component Container）からのコネクションのアソシエート処理でのイベント ID（非永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9306	ManagedConnection.associateConnection の入口	B
0x9307	ManagedConnection.associateConnection の出口	B

(3) アプリケーションからのメッセージ送受信処理

アプリケーションからのメッセージ送受信処理でのイベント ID を次の表に示します。

表 D-9 アプリケーションからのメッセージ送受信処理でのイベント ID（非永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9308	QueueSender.send(msg) の入口	A
0x9309	QueueSender.send(msg) の出口	A
0x930A	QueueSender.send(msg,dlvmd, pri,ttl) の入口	A
0x930B	QueueSender.send(msg,dlvmd, pri,ttl) の出口	A
0x930C	QueueSender.send(que,msg) の入口	A
0x930D	QueueSender.send(que,msg) の出口	A
0x930E	QueueSender.send(que,msg,dlvmd, pri,ttl) の入口	A
0x930F	QueueSender.send(que,msg,dlvmd, pri,ttl) の出口	A
0x9310	QueueReceiver.receive(引数なし) の入口	A
0x9311	QueueReceiver.receive(引数なし) の出口	A
0x9312	QueueReceiver.receive(timeout) の入口	A
0x9313	QueueReceiver.receive(timeout) の出口	A
0x9314	QueueReceiver.receiveNoWait の入口	A
0x9315	QueueReceiver.receiveNoWait の出口	A
0x9316	Enumeration.nextElement の入口	A
0x9317	Enumeration.nextElement の出口	A

(4) J2EE サーバ（Component Container）からのトランザクション制御

J2EE サーバ（Component Container）からのトランザクション制御でのイベント ID を次の表に示します。

表 D-10 J2EE サーバ（Component Container）からのトランザクション制御でのイベント ID（非永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9318	XAResource.start の入口	B
0x9319	XAResource.start の出口	B
0x931A	XAResource.end の入口	B
0x931B	XAResource.end の出口	B
0x931C	XAResource.prepare の入口	B
0x931D	XAResource.prepare の出口	B
0x931E	XAResource.commit の入口	B
0x931F	XAResource.commit の出口	B
0x9320	XAResource.rollback の入口	B
0x9321	XAResource.rollback の出口	B
0x9322	XAResource.forget の入口	B
0x9323	XAResource.forget の出口	B
0x9324	XAResource.recover の入口	B
0x9325	XAResource.recover の出口	B
0x9326	LocalTransaction.begin の入口	B
0x9327	LocalTransaction.begin の出口	B
0x9328	LocalTransaction.commit の入口	B
0x9329	LocalTransaction.commit の出口	B
0x932A	LocalTransaction.rollback の入口	B
0x932B	LocalTransaction.rollback の出口	B

(5) JMS アプリケーションからの JMS セッションの決着処理

JMS アプリケーションからの JMS セッションの決着処理でのイベント ID を次の表に示します。

表 D–11　JMS アプリケーションからの JMS セッションの決着処理でのイベント ID（非永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x932C	QueueSession.commit の入口	A
0x932D	QueueSession.commit の出口	A
0x932E	QueueSession.rollback の入口	A
0x932F	QueueSession.rollback の出口	A
0x9330	Message.acknowledge の入口	A
0x9331	Message.acknowledge の出口	A
0x9332	QueueSession.recover の入口	A
0x9333	QueueSession.recover の出口	A

(6)　J2EE サーバ（Component Container）と連携した Message-driven Bean メッセージ配信処理

J2EE サーバ（Component Container）と連携した Message-driven Bean メッセージ配信処理でのイベント ID を次の表に示します。

表 D–12　J2EE サーバ（Component Container）と連携した Message-driven Bean メッセージ配信処理でのイベント ID（非永続版リソースアダプタの場合）

イベント ID	PRF トレース取得ポイント	取得レベル
0x9335	XAQueueSessionImpl.beforeDelivery の ServerSession.beforeDelivery の呼び出し後	B
0x9336	XAQueueSessionImpl.onMessage の MessageListener.onMessage の呼び出し前	A
0x9337	XAQueueSessionImpl.onMessage の MessageListener.onMessage の呼び出し後	A
0x9338	XAQueueSessionImpl.afterDelivery の ServerSession. afterDelivery の呼び出し前	B

付録 E　Reliable Messaging のメモリ所要量とディスク占有量

Reliable Messaging のメモリ所要量とディスク占有量を示します。

付録 E.1　メモリ所要量

Java ヒープ領域での Reliable Messaging のメモリ所要量を示します。

Application Server のメモリ所要量と，次に示す Reliable Messaging のメモリ所要量の総和の最大値を Application Server の J2EE サーバ用オプション定義ファイル（usrconf.cfg）に指定する必要があります。add.jvm.arg キーの-Xmx オプションに指定してください。

(1)　永続版リソースアダプタの場合

永続版リソースアダプタの場合のメモリ所要量を次に示します。

(a)　メモリ所要量の見積もり式

Reliable Messaging のメモリ所要量の総和を求めるには，次に示す 1.～7.を加算します。

1.および 2.で示すメモリ所要量は，Reliable Messaging が起動している間，常に Java ヒープ領域を消費する値です。

3.～6.で示すメモリ所要量は，該当する処理の間，一時的に使用されるメモリ使用量です。該当する処理が終わると解放されます。

7.で示すメモリ所要量は，JDBC コネクションの生成時に使用されるメモリ所要量です。JDBC コネクションを切断すると解放されます。ただし，Application Server のコネクションプーリング機能が有効な場合で，Application Server で定義したコネクションプーリング数に空きがあるときは，JDBC コネクションは解放されないでプールされるため，メモリは保持され続けます。

1. Reliable Messaging 起動時のメモリ所要量

```
  60KB×キューの数
＋60KB×永続化されたメッセージ数
＋75KB×コンフィグレーションプロパティのMDBのインスタンス数
＋75KB×コンフィグレーションプロパティのRMTRSendThreadNumの値※1
＋20MB
```

2. メッセージをキャッシュした際のメモリ所要量

```
キャッシュしたメッセージのサイズ※2×キャッシュしたメッセージ数
```

3. 送信処理で一時的に使用するメモリ使用量

```
（メッセージのサイズ※2×10＋150KB）×メッセージ同時送信数＋2MB
```

4. 受信処理で一時的に使用するメモリ使用量

（メッセージのサイズ※2×5＋300KB）×メッセージ同時受信数＋2MB

5. MDB による受信処理で一時的に使用するメモリ使用量

（メッセージのサイズ※2×5＋4MB）×メッセージ同時受信数＋2MB

6. キュー間転送で一時的に使用するメモリ使用量

（メッセージのサイズ※2×10＋450KB）×メッセージ同時送信数＋2MB

7. JDBC コネクションでのメモリ所要量

使用する最大のメッセージのサイズ※2×4×コネクション同時使用数

注※1

コンフィグレーションプロパティの RMTRConnectFlag の値を true に設定した場合だけ加算してください。

注※2

メッセージのヘッダ，プロパティ，ペイロードおよびメッセージに付属するオブジェクトのサイズの合計です。

(b)　メモリ所要量の見積もり例

次に示す七つの条件のもとに Reliable Messaging のメモリ所要量を見積もる例を示します。

- 条件 1：デフォルトのオプションで生成されたローカルキューが一つ
- 条件 2：条件 1 のキューに永続化されたメッセージは 10 件
- 条件 3：条件 2 のメッセージはキャッシュされていない状態
- 条件 4：コンフィグレーションプロパティの MDB のインスタンス数の値を 1 に設定
- 条件 5：使用する最大のメッセージのサイズが 1MB
- 条件 6：必須項目以外のコンフィグレーションプロパティはデフォルト値
- 条件 7：メッセージサイズが 1MB のメッセージ

見積もり例 1

条件 1～条件 6 を満たす環境で Reliable Messaging を起動した場合のメモリ所要量は，「付録 E.1(1)(a)　メモリ所要量の見積もり式」で示した「1. Reliable Messaging 起動時のメモリ所要量」と「7. JDBC コネクションでのメモリ所要量」の合計で算出できます。この条件の場合，7.のコネクション同時使用数は 2 となります。

```
（60KB×1＋60KB×10＋75KB×1＋20MB）＋（1MB×4×2）
＝約29MB
```

見積もり例 2

条件 1～条件 5 を満たす環境で，条件 1 で生成したローカルキューに対して条件 7 のメッセージを 1 件送信した際のメモリ所要量は，「付録 E.1(1)(a)　メモリ所要量の見積もり式」で示した「1. Reliable Messaging 起動時のメモリ所要量」，「2. メッセージをキャッシュした際のメモリ所要量」，「3. 送信処理で一時的に使用するメモリ使用量」および「7. JDBC コネクションでのメモリ所要量」の合計で算出できます。この条件の場合，7.のコネクション同時使用数は 2 となります。

```
（60KB×1＋60KB×10＋75KB×1＋20MB）＋1MB＋（（1MB×10＋150KB）×1＋2MB）＋（1MB×4×2）
＝約42MB
```

(c)　メモリ所要量見積もり時の注意事項

次に示す値は，hrmmkque コマンド（-t オプションに shr_receive 指定）の実行時に一時的に使用するメモリ所要量です。

```
3KB×-nオプションの値＋3MB
```

この値は，サーバ側の Java VM でのメモリ使用量です。クライアント側でのメモリ使用量には加算されません。

(2)　非永続版リソースアダプタの場合

非永続版リソースアダプタの場合のメモリ所要量を次に示します。

(a)　メモリ所要量の見積もり式

Reliable Messaging のメモリ所要量の総和を求めるには，次に示す 1.～5.を加算します。

1.および 2.で示すメモリ所要量は，Reliable Messaging が起動している間，常に Java ヒープ領域を消費する値です。

3.～5.で示すメモリ所要量は，該当する処理の間，一時的に使用されるメモリ使用量です。該当する処理が終わると解放されます。

1. Reliable Messaging 起動時のメモリ所要量

```
60KB×キューの数＋20MB
```

2. メッセージをキューに登録した際のメモリ所要量

```
キューに登録したメッセージのサイズ※×メッセージ数
＋60KB×登録されたメッセージ数
```

3. 送信処理で一時的に使用するメモリ使用量

```
（メッセージのサイズ※×10＋150KB）×メッセージ同時送信数＋2MB
```

4. 受信処理で一時的に使用するメモリ使用量

（メッセージのサイズ※×5＋300KB）×メッセージ同時受信数＋2MB

5. Message-driven Bean による受信処理で一時的に使用するメモリ使用量

（メッセージのサイズ※×5＋4MB）×メッセージ同時受信数＋2MB

注※

メッセージのヘッダ，プロパティ，ペイロードおよびメッセージに付属するオブジェクトのサイズの合計です。

(b)　メモリ所要量の見積もり例

次に示す三つの条件のもとに Reliable Messaging のメモリ所要量を見積もる例を示します。

- 条件 1：デフォルトのオプションで生成されたローカルキューが 10 個
- 条件 2：必須項目以外のコンフィグレーションプロパティはデフォルト値
- 条件 3：メッセージサイズが 1MB のメッセージ

見積もり例 1

条件 1～条件 2 を満たす環境で Reliable Messaging を起動した場合のメモリ所要量は，「付録 E.1(2)(a)　メモリ所要量の見積もり式」で示した「1. Reliable Messaging 起動時のメモリ所要量」で算出できます。

```
60KB×10＋20MB
＝約21MB
```

見積もり例 2

条件 1～条件 2 を満たす環境で，ローカルキューに対して条件 3 のメッセージを 1 件送信した際の一時的なメモリ所要量は，「付録 E.1(2)(a)　メモリ所要量の見積もり式」で示した次の値の合計で算出できます。

1. Reliable Messaging 起動時のメモリ所要量
2. メッセージをキューに登録した際のメモリ所要量
3. 送信処理で一時的に使用するメモリ使用量

```
（60KB×10＋20MB）＋（1MB×1＋60KB×1）＋（（1MB×10＋150KB）×1＋2MB）
＝約34MB
```

付録 E.2　ディスク占有量

Reliable Messaging のディスク占有量は，永続版リソースアダプタ，非永続版リソースアダプタ共に 10MB です。

付録 F　HiRDB の見積もり

HiRDB の排他資源と同時アクセス可能実表数の見積もりについて説明します。

付録 F.1　HiRDB の排他資源の見積もり

HiRDB の排他資源数を見積もるには，Reliable Messaging に関する数値が必要となります。その数値とは，一つのトランザクションで発行した複数の SQL 実行での排他資源の総和です。Reliable Messaging で排他資源数が最大となる場合の排他資源数を算出して pd_lck_pool_size オペランドに指定します。

排他資源数を見積もる計算式については，マニュアル「HiRDB システム定義」を参照してください。

Reliable Messaging の排他資源数は処理ごとのテーブルへのアクセス回数および DB テーブルの BINARY 型列数を使用して算出します。それぞれを次の表に示します。

表 F-1　Reliable Messaging が使用する SQL

項番	処理	キュー種別	使用 SQL※1	使用テーブル	アクセス回数
1	Reliable Messaging 起動	永続キュー属性のローカルキューまたは転送キュー	SELECT	キュー情報	テーブルごとに 1 回
				FIFO 情報	
				メッセージ情報	
		非永続キュー属性のローカルキューまたは転送キュー	SELECT	キュー情報	
				FIFO 情報	
		受信用共用キュー	SELECT	キュー情報	
				FIFO 情報	
				共用キュー受信用ライト管理	
				共用キュー受信用リード管理	
		送信用共用キュー	SELECT	キュー情報	
				FIFO 情報	
		キューの種別にかかわらない場合	SELECT	あて先情報	1 回
2	メッセージ送信	永続キュー属性のローカルキューまたは転送キュー	INSERT	メッセージ情報	テーブルごとに 1 回
		送信用共用キュー	UPDATE	メッセージ情報	

項番	処理	キュー種別	使用 SQL※1	使用テーブル	アクセス回数
				共用キュー受信用ライト管理	
3	メッセージ受信	永続キュー属性のローカルキュー	SELECT	メッセージ情報	テーブルごとに 1 回
			UPDATE※2	メッセージ情報	
		受信用共用キュー	SELECT	メッセージ情報	
			UPDATE	共用キュー受信用リード管理	
4	メッセージ閲覧	永続キュー属性のローカルキュー	SELECT	メッセージ情報	テーブルごとに 1 回
		受信用共用キュー	SELECT	メッセージ情報	
5	メッセージ削除	永続キュー属性のローカルキュー	DELETE	メッセージ情報	削除対象のメッセージの数
				FIFO 情報	削除対象のグループの数
			UPDATE	FIFO 情報	2 回
			SELECT※3	メッセージ情報	削除対象のメッセージの数
			INSERT※3	メッセージ情報	削除対象のメッセージの数
		非永続キュー属性のローカルキュー	DELETE	FIFO 情報	削除対象のグループの数
		永続キュー属性の転送キュー	DELETE	メッセージ情報	削除対象のメッセージの数
				FIFO 情報	削除対象のグループの数
			UPDATE	FIFO 情報	2 回
		非永続キュー属性の転送キュー	DELETE	FIFO 情報	削除対象のグループの数

注※1

SQL の発行形態は次のとおりです。

・SELECT：LOCK TABLE なし，かつ WITHOUT LOCK 指定なし

・INSERT：VALUES 句指定，LOCK TABLE なし，かつ WITHOUT LOCK 指定なし

・UPDATE：LOCK TABLE なし

・DELETE：LOCK TABLE なし

注※2

メッセージの即時削除機能を利用する場合，UPDATE の代わりにメッセージ削除の DELETE がメッセージ情報テーブルに発行されます。

注※3

デッドメッセージキューを指定している場合だけ使用します。

表 F-2　DB テーブルの BINARY 型列数

項番	DB テーブル種別	BINARY 型列数	備考
1	システム管理情報	0	－
2	キュー情報	0	－
3	FIFO 情報	2	Reliable Messaging の内部情報として利用します。データ長はキュー間転送受信で順序保証を行ったときのスキップしたメッセージ数や未受信メッセージ数に依存します。1 メッセージにつき，最大 20 けたの java.math.BigInteger 型と java.lang.Long 型を組み合わせたデータが格納されます。
4	メッセージ情報	1	永続キュー属性のローカルキューまたは転送キューの最大メッセージ長。詳細については，「2.5.4　メッセージサイズを見積もる方法」を参照してください。
5	共用キュー受信用メッセージ情報	2	1 列は，受信用共用キューの最大メッセージ長。詳細については，「2.5.4　メッセージサイズを見積もる方法」を参照してください。なお，もう 1 列は使用しません。
6	共用キュー受信用ライト管理	1	使用しません。
7	共用キュー受信用リード管理	2	1 列は，Reliable Messaging の内部情報として 256 バイト固定で使用します。なお，もう 1 列は使用しません。
8	あて先情報	0	－

（凡例）

－：該当しません。

注意事項

- 各テーブルのテーブル数，行数，列数およびインデックス数については，「4.3.6　管理情報テーブルの一覧」を参照してください。
- 各テーブルを格納する RD エリアを HiRDB で決定させる場合は，テーブルごとに RD エリアを割り当てた数値としてください。
- 次の機能は使用していません。

  表分割

  インデックス分割

  ビュー表

LOB 列定義
ユーザ定義
LOB 属性
上位型
関数
プラグイン
ルーチン
ページ排他

付録 F.2　同時アクセス可能実表数の見積もり

Reliable Messaging の構成によっては，HiRDB の同時アクセス可能実表数の値を超える場合があります。Reliable Messaging で同時アクセス可能実表数が最大となるのは，Reliable Messaging の起動時です。

Reliable Messaging 利用時の同時アクセス可能実表数は，「付録 F.1　HiRDB の排他資源の見積もり」を参照して見積もってください。算出した同時アクセス可能実表数は，pd_max_access_tables オペランドに指定します。

同時アクセス可能実表数については，マニュアル「HiRDB システム定義」を参照してください。

付録 G　コネクションプールの見積もり

コネクションプールに確保する最大コネクション数を見積もるための計算式は次のとおりです。

```
コネクションプールに確保する最大コネクション数
＝アプリケーションが消費するコネクション数
＋Reliable Messagingが消費するコネクション数
```

アプリケーションが消費するコネクション数，および Reliable Messaging が消費するコネクション数の算出方法について説明します。

付録 G.1　アプリケーションが消費するコネクション数

Reliable Messaging を利用するアプリケーションごとに次の計算をし，個々のアプリケーションで計算した値の総和が，アプリケーションが消費するコネクション数となります。

```
アプリケーションの最大同時実行数（多重度）×アプリケーションでの消費コネクション数
```

アプリケーションでの消費コネクション数は，アプリケーションの種類が MDB の場合と MDB 以外（SessionBean やサーブレットなど）の場合とで算出方法が異なります。それぞれの場合の算出方法について説明します。

(1)　MDB の場合

MDB の場合，メッセージを一つ配信する契機でコネクションを一つ消費します。この消費コネクションを「配信用コネクション」と定義します。配信用コネクションは，MDB アプリケーション内で生成した javax.jms.QueueSession および java.sql.Connection とコネクションシェアリングできます。これを「配信用コネクションとのシェアリング」と定義します。

アプリケーションでの消費コネクション数は，配信用コネクションとのシェアリングが適用される場合と適用されない場合によって次のように異なります。

配信用コネクションとのシェアリングが適用される場合

```
アプリケーションでの消費コネクション数＝1
```

配信用コネクションとのシェアリングが適用されない場合

```
アプリケーションでの消費コネクション数
＝1（配信用コネクション）
＋MDBアプリケーション内でコネクションシェアリングされたグループの数※1
＋MDBアプリケーション内でコネクションシェアリングされないjavax.jms.QueueSessionの生成数
＋MDBアプリケーション内でコネクションシェアリングされないjava.sql.Connection※2の生成数
```

注※1

配信用コネクションとはシェアリングされないで，MDB アプリケーション内で生成した複数の javax.jms.QueueSession および java.sql.Connection がコネクションシェアリングされることを示します。

注※2

DB Connector for Reliable Messaging の javax.sql.DataSource から生成した java.sql.Connection が対象です。また，Reliable Messaging のコンフィグレーションプロパティ RMAssociateJDBCFlag の値が true の場合だけ生成できます。

配信用コネクションとのシェアリングが適用されるか適用されないかは，次の条件によって決まります。

配信用コネクションとのシェアリングが適用される場合

次の条件をすべて満たす場合，配信用コネクションとのシェアリングが適用されます。

- リソース参照に関する設定の，コネクションシェアリングの有無で「Shareable」を設定
- トランザクション管理方法を「Container」に設定
- トランザクション属性を「Required」に設定

配信用コネクションとのシェアリングが適用されない場合

次のどれかの条件を満たす場合，配信用コネクションとのシェアリングは適用されません。

- リソース参照に関する設定の，コネクションシェアリングの有無で「Unshareable」を設定
- トランザクション管理方法を「Bean」に設定
- トランザクション属性を「Not Supported」に設定

(2) MDB 以外（SessionBean やサーブレットなど）の場合

アプリケーションで生成する javax.jms.QueueSession および java.sql.Connection の合計数が消費コネクション数となります。ただし，コネクションシェアリングが適用された場合，コネクションシェアリングされた一つのグループにつき，消費コネクションは一つとなります。よって，アプリケーションでの消費コネクション数は次のようになります。

```
アプリケーションでの消費コネクション数
＝コネクションシェアリングされたグループの数
＋コネクションシェアリングされないjavax.jms.QueueSessionの生成数
＋コネクションシェアリングされないjava.sql.Connection※の生成数
```

注※

DB Connector for Reliable Messaging の javax.sql.DataSource から生成した java.sql.Connection が対象です。また，Reliable Messaging のコンフィグレーションプロパティ RMAssociateJDBCFlag の値が true の場合だけ生成できます。

コネクションシェアリングが適用される条件については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」を参照してください。

付録 G.2　Reliable Messaging が消費するコネクション数

Reliable Messaging が消費するコネクション数は，キュー間転送を利用している場合だけ算出する必要があります。Reliable Messaging が消費するコネクション数は次のようになります。

Reliable Messagingが消費するコネクション数＝キュー間転送受信時の同時リクエスト実行数

付録 H.1　uCosminexus Reliable Messaging 01-00 からの移行

uCosminexus Reliable Messaging 01-00 から Reliable Messaging 09-00 へ移行するときの手順について，次に示します。サーバ管理コマンドについては，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」を参照してください。

1. uCosminexus Reliable Messaging の停止

   uCosminexus Reliable Messaging を停止します。詳細については，「4.1.3　Reliable Messaging の停止（永続版リソースアダプタの場合）」を参照してください。

2. J2EE サーバの停止

   サーバ管理コマンドを使用して，J2EE サーバを停止します。

3. リソースアダプタのバージョンアップ

   サーバ管理コマンドを使用して，Reliable Messaging および DB Connector をバージョンアップしてください。

4. 環境設定の変更

   %HRMDIR%を Reliable Messaging 09-00 がインストールされたディレクトリに合わせて修正してください。PATH 環境変数などに指定している場合は同様に修正してください。

5. 共用キューのバージョンアップ

   共用キューを使用している場合は，共用キューをバージョンアップしてください。詳細については，「付録 H.3　共用キューのバージョンアップ」を参照してください。

6. usrconf.cfg の修正

   キュー間転送を使用している場合，usrconf.cfg ファイルをテキストエディタで開き，次の行を削除してください。

   ```
   add.class.path=<uCosminexus Reliable Messaging 01-00のインストールディレクトリ>¥conf
   ```

7. キュー間転送用 Web アプリケーションの入れ替え

   キュー間転送を使用している場合，uCosminexus Reliable Messaging 01-00 のキュー間転送用 Web アプリケーションがデプロイされています。これを Reliable Messaging 09-00 のキュー間転送用 Web アプリケーションに入れ替えてください。

次に示す手順は，必須ではありませんが，実施することをお勧めします。

1. uCosminexus Reliable Messaging 01-00 のアンインストール

   uCosminexus Reliable Messaging 01-00 のコマンドやライブラリを誤って使用しないように，アンインストールしてください。

2. サポートされないプロパティの削除

Reliable Messaging 09-00 ではサポートされないプロパティを削除してください。Reliable Messaging 09-00 ではサポートされないプロパティを次に示します。

- RMReceiveThreadNum
- RMLogTraceFileDir
- RMTRTransferControl

3. 送信用共用キューの再作成

Reliable Messaging 09-00 に移行したあと，uCosminexus Reliable Messaging 01-00 で作成した送信用共用キューを削除し，新たに送信用共用キューを作成してください。

注意事項

- uCosminexus Reliable Messaging 01-00 で作成した転送キューを Reliable Messaging 09-00 に移行した場合，転送モードが互換モード（compatible）の転送キューとして扱われます。
- uCosminexus Reliable Messaging 01-00 から本バージョンの Reliable Messaging にシステム変更した場合は，コネクションプールの見積もりを見直してください。

付録 H.2　Reliable Messaging 01-01～08-00 からの移行

Reliable Messaging 01-01～08-00 から Reliable Messaging 09-00 へ移行するときの手順は，uCosminexus Reliable Messaging 01-00 から Reliable Messaging 09-00 へ移行するときの手順 1.～手順 3.と同様です（手順 1.の uCosminexus Reliable Messaging は Reliable Messaging 01-01 に読み替えてください）。手順の詳細は，「付録 H.1　uCosminexus Reliable Messaging 01-00 からの移行」を参照してください。

注意事項

Reliable Messaging 01-01～01-02 で作成した転送キューを Reliable Messaging 09-00 に移行した場合，転送モードが互換モード（compatible）の転送キューとして扱われます。

付録 H.3　共用キューのバージョンアップ

共用キューのバージョンを 0100 または 0200 から 0300 に移行する手順について，次に示します。

1. Reliable Messaging の停止

Reliable Messaging を停止します。詳細については，「4.1.3　Reliable Messaging の停止（永続版リソースアダプタの場合）」を参照してください。

2. DB のバックアップ

バージョンアップによって，次に示す三つのテーブルが変更されます。必要に応じて DB のバックアップをとってください。

- 共用キュー受信用メッセージ情報テーブル
- 共用キュー受信用ライト管理テーブル
- 共用キュー受信用リード管理テーブル

3. スクリプトファイルの編集

共用キューをバージョンアップするためのスクリプトファイル（%HRMDIR%¥sql¥shqupdate_V1toV2.sql）を任意の場所にコピーしてから，テキストエディタで開き，ファイル中の"<RMSystemName>"の部分を RMSystemName の指定値に置き換えてください。また，ファイル中の"<QueueName>"の部分をバージョンアップする受信用共用キューのキュー名に置き換えてください。

4. SQL ファイルの実行

SQL ファイルを実行して，受信用共用キューのテーブルを拡張します。

Windows の場合

次に示すコマンドで HiRDB SQL Executer を開始したあと，HiRDB SQL Executer の［ファイル］メニューから［ファイルから実行］を選択し，手順 3.で編集した SQL ファイルを指定して実行します。

```
pdsqlw -u <接続ユーザ名>/<パスワード>※
       -h <HiRDBサーバのホスト名またはIPアドレス>
       -n <HiRDBサーバのポート番号>
```

UNIX の場合

環境変数 PDUSER※，PDHOST，PDNAMEPORT を設定したあと，次に示すコマンドで HiRDB SQL Executer から SQL ファイルを実行します。

```
pdsql  <  <手順3.で編集したSQLファイルのパス>
```

注※

接続ユーザ名には，権限を付与した接続ユーザ名を指定します。詳細については，「3.4.1(1)(b) HiRDB のユーザ権限の付与」を参照してください。

5. バージョンアップしたい受信用共用キューに応じて手順 2.から手順 4.までを繰り返してください。

付録 H.4　移行時の注意事項

移行時の注意事項を次に示します。

- Reliable Messaging 01-02 以前のバージョンで作成した転送キューを Reliable Messaging 01-03 以降に移行した場合，転送モードが互換モード（compatible）の転送キューとして扱われます。
- 01-02 以前の Reliable Messaging 同士をキュー間転送機能を利用して連携している場合，片側のシステムだけを 01-03 以降にバージョンアップしたときは，それ以降，新規に転送キューを追加する際は転送キューの転送モード属性を互換モードで作成してください。
- Reliable Messaging のバージョンアップや修正パッチを適用した場合，コマンドのバッチファイル（Windows の場合）またはシェルスクリプト（UNIX の場合）に記載されている，コマンドのリクエストタイムアウト値「ejbserver.rmi.request.timeout」の値は，0（タイムアウト無し）で上書きされます。0 以外を指定していた場合は，移行後に値を設定し直してください。
- 01-03 より hrmmkaddr コマンドで指定する宛先名に入力可能な文字が文字列から識別子に変更となりました。識別子とは，先頭が英字で始まる英数字列，__（アンダーバー）を示します。01-02 以前のバージョンで作成済みの宛先名はそのまま利用することができますが，新規に識別子の該当しない文字列で宛先名を作成するとエラーとなります。
- Reliable Messaging がシステム閉塞状態の時に Message-driven Bean を開始した場合，01-02 以前のバージョンではエラーとなり，開始に失敗していましたが，01-03 以降では Message-driven Bean は開始します。ただし，このとき Message-driven Bean にメッセージは配送されません。

付録 I　用語解説

マニュアルで使用する用語について

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 用語解説」を参照してください。

索引

A

B

C

D

F

G

H

I

J

L

M

O

P

Q

R

S

T

V

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

は

ひ

ふ

へ

ほ

め

ゆ

よ

り

れ

ろ