HITACHI
Inspire the Next

Cosminexus アプリケーションサーバ V8

機能解説 基本・開発編（EJB コンテナ）

解説書

3020-3-U06-60

■対象製品

●**適用** OS：Windows Server 2003，Windows Server 2003 R2，Windows Server 2003（x64）※1，Windows Server 2003 R2（x64）※1，Windows Server 2008 x86，Windows Server 2008 x64 ※1，Windows Server 2008 R2 ※1

P-2443-7B84 uCosminexus Application Server Standard-R 08-70

P-2443-7D84 uCosminexus Application Server Standard 08-70

P-2443-7K84 uCosminexus Application Server Enterprise 08-70

P-2443-7M84 uCosminexus Web Redirector 08-70

P-2443-7S84 uCosminexus Service Platform 08-70 ※2

●**適用** OS：Windows Server 2003，Windows Server 2003 R2，Windows Vista，Windows XP，Windows 7（32bit），Windows 7（x64）※1

P-2443-7E84 uCosminexus Developer Standard 08-70

P-2443-7F84 uCosminexus Developer Professional 08-70

P-2443-7T84 uCosminexus Service Architect 08-70 ※2

●**適用** OS：Windows Server 2003，Windows Server 2003 R2，Windows Server 2003（x64）※1，Windows Server 2003 R2（x64）※1，Windows Server 2008 x86，Windows Server 2008 x64 ※1，Windows Server 2008 R2 ※1，Windows Vista，Windows XP，Windows 7（32bit），Windows 7（x64）※1

P-2443-7H84 uCosminexus Client 08-70

●**適用** OS：Windows Server 2003（x64），Windows Server 2003 R2（x64），Windows Server 2008 x64，Windows Server 2008 R2

P-2943-7B84 uCosminexus Application Server Standard-R 08-70

P-2943-7D84 uCosminexus Application Server Standard 08-70

P-2943-7K84 uCosminexus Application Server Enterprise 08-70

P-2943-7S84 uCosminexus Service Platform 08-70 ※2

●**適用** OS：AIX 5L V5.3，AIX V6.1，AIX V7.1

P-1M43-7D81 uCosminexus Application Server Standard 08-70 ※2

P-1M43-7K81 uCosminexus Application Server Enterprise 08-70 ※2

P-1M43-7S81 uCosminexus Service Platform 08-70 ※2

●**適用** OS：HP-UX 11i V2（IPF），HP-UX 11i V3（IPF）

P-1J43-7D81 uCosminexus Application Server Standard 08-70

P-1J43-7K81 uCosminexus Application Server Enterprise 08-70

P-1J43-7S81 uCosminexus Service Platform 08-70 ※2

●**適用** OS：Red Hat Enterprise Linux AS 4（x86），Red Hat Enterprise Linux ES 4（x86），Red Hat Enterprise Linux AS 4（AMD64 & Intel EM64T），Red Hat Enterprise Linux ES 4（AMD64 & Intel EM64T），Red Hat Enterprise Linux 5 Advanced Platform（x86），Red Hat Enterprise Linux 5（x86），Red Hat Enterprise Linux 5 Advanced Platform（AMD/Intel 64），Red Hat Enterprise Linux 5（AMD/Intel 64），Red Hat Enterprise Linux Server 6（32-bit x86），Red Hat Enterprise Linux Server 6（64-bit x86_64）

P-9S43-7B81 uCosminexus Application Server Standard-R 08-70 ※2

P-9S43-7D81 uCosminexus Application Server Standard 08-70 ※2

P-9S43-7K81 uCosminexus Application Server Enterprise 08-70 ※2

P-9S43-7M81 uCosminexus Web Redirector 08-70 ※2

P-9S43-7S81 uCosminexus Service Platform 08-70 ※2

注※１　WOW64（Windows On Windows 64）環境だけで使用できます。
注※２　この製品については，サポート時期をご確認ください。
これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。
本製品では日立トレース共通ライブラリをインストールします。

■輸出時の注意

本製品を輸出される場合には，外国為替および外国貿易法ならびに米国の輸出管理関連法規などの規制をご確認の上，必要な手続きをお取りください。
なお，ご不明な場合は，弊社担当営業にお問い合わせください。

■商標類

Active Directory は，米国 Microsoft Corporation の，米国およびその他の国における登録商標または商標です。
AIX は，米国およびその他の国における International Business Machines Corporation の商標です。
AIX 5L は，米国およびその他の国における International Business Machines Corporation の商標です。
AMD は，Advanced Micro Devices, Inc. の商標です。
AX2000 は，A10 Networks, Inc. の商品名称です。
Borland のブランド名および製品名はすべて，米国 Borland Software Corporation の米国およびその他の国における商標または登録商標です。
CORBA は，Object Management Group が提唱する分散処理環境アーキテクチャの名称です。
HP-UX は，Hewlett-Packard Company のオペレーティングシステムの名称です。
IIOP は，OMG 仕様による ORB(Object Request Broker) 間通信のネットワークプロトコルの名称です。
Internet Explorer は，米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
Itanium は，アメリカ合衆国およびその他の国における Intel Corporation の商標です。
J2EE は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
Java は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
JDBC は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
JDK は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
JSP は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
Linux は，Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。
Microsoft は，米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
Microsoft Internet Information Services は，米国 Microsoft Corporation の商品名称です。
Microsoft Office Excel は，米国 Microsoft Corporation の商品名称です。
MyEclipse は，米国 Genuitec 社の商品名称です。
OMG，CORBA，IIOP，UML，Unified Modeling Language，MDA，Model　Driven　Architecture は，Object Management Group, Inc. の米国及びその他の国における登録商標または商標です。

■マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

製品名	表記	
Microsoft(R) Active Directory(R)	Active Directory	
Microsoft(R) Office Excel	Excel	
Microsoft(R) Internet Information Services 6.0	Microsoft IIS 6.0	Microsoft IIS
Microsoft(R) Internet Information Services 7.0	Microsoft IIS 7.0	
Microsoft(R) Internet Information Services 7.5	Microsoft IIS 7.5	
Microsoft(R) SQL Server 2000	SQL Server 2000	SQL Server
Microsoft(R) SQL Server 2005	SQL Server 2005	
Microsoft(R) SQL Server 2008	SQL Server 2008	

製品名	表記		
Microsoft(R) SQL Server 2000 Driver for JDBC	SQL Server 2000 Driver for JDBC	SQL Server の JDBC ドライバ	
Microsoft(R) SQL Server 2005 JDBC Driver	SQL Server 2005 JDBC Driver		
Microsoft(R) SQL Server JDBC Driver 2.0	SQL Server JDBC Driver		
Microsoft(R) SQL Server JDBC Driver 3.0			
Microsoft(R) Windows(R) 7 Enterprise（32bit）	Windows 7（32bit）	Windows 7	Windows
Microsoft(R) Windows(R) 7 Professional（32bit）			
Microsoft(R) Windows(R) 7 Ultimate（32bit）			
Microsoft(R) Windows(R) 7 Enterprise（x64）	Windows 7（x64）		
Microsoft(R) Windows(R) 7 Professional（x64）			
Microsoft(R) Windows(R) 7 Ultimate（x64）			
Microsoft(R) Windows Server(R) 2003, Enterprise Edition 日本語版	Windows Server 2003 Enterprise Edition	Windows Server 2003	
Microsoft(R) Windows Server(R) 2003, Standard Edition 日本語版	Windows Server 2003 Standard Edition		
Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition 日本語版	Windows Server 2003 R2 Enterprise Edition	Windows Server 2003 R2	
Microsoft(R) Windows Server(R) 2003 R2, Standard Edition 日本語版	Windows Server 2003 R2 Standard Edition		
Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition 日本語版	Windows Server 2003 Enterprise x64 Edition	Windows Server 2003（x64）	
Microsoft(R) Windows Server(R) 2003, Standard x64 Edition 日本語版	Windows Server 2003 Standard x64 Edition		
Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition 日本語版	Windows Server 2003 R2 Enterprise x64 Edition	Windows Server 2003 R2（x64）	
Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition 日本語版	Windows Server 2003 R2 Standard x64 Edition		
Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit 日本語版	Windows Server 2008 x86	Windows Server 2008	
Microsoft(R) Windows Server(R) 2008 Standard 32-bit 日本語版			
Microsoft(R) Windows Server(R) 2008 Enterprise 日本語版	Windows Server 2008 x64		

製品名	表記		
Microsoft(R) Windows Server(R) 2008 Standard 日本語版			
Microsoft(R) Windows Server(R) 2008 R2 Enterprise 日本語版	Windows Server 2008 R2		
Microsoft(R) Windows Server(R) 2008 R2 Standard 日本語版			
Microsoft(R) Windows Vista(R) Business	Windows Vista Business	Windows Vista	
Microsoft(R) Windows Vista(R) Enterprise	Windows Vista Enterprise		
Microsoft(R) Windows Vista(R) Ultimate	Windows Vista Ultimate		
Microsoft(R) Windows(R) XP Professional Operating System	Windows XP		
Windows(R) Internet Explorer(R)	Internet Explorer		

■発行

2011 年 7 月 3020-3-U06-60

■著作権

変更内容

変更内容（3020-3-U06-60）uCosminexus Application Server Enterprise 08-70，uCosminexus Application Server Standard 08-70，uCosminexus Application Server Standard-R 08-70，uCosminexus Client 08-70，uCosminexus Developer Professional 08-70，uCosminexus Developer Standard 08-70，uCosminexus Service Architect 08-70，uCosminexus Service Platform 08-70，uCosminexus Web Redirector 08-70

追加・変更内容	変更個所
アプリケーションサーバ 08-70 での主な機能変更についての説明を追加した。また，アプリケーションサーバ 08-53 での主な機能変更についての説明を付録 B.1 に移動した。	1.4，付録 B.1
Connector 1.5 仕様に準拠したリソースアダプタで Transacted Delivery を使用できるようにした。これに伴い，TP1 インバウンドアダプタ使用時以外も CMT に Required を指定できるようにした。	2.7.3，4.3.5
インターセプタに関する定義，およびアプリケーション例外に関する定義を DD で設定できるようにした。	2.15，4.2.16
EJB QL の finder メソッドまたは select メソッドに関する注意を追加した。	4.3.4
次の製品の適用 OS に AIX，HP-UX（IPF）を追加した。 • uCosminexus Application Server Enterprise • uCosminexus Application Server Standard • uCosminexus Service Platform	－
次の製品の適用 OS に Red Hat Enterprise Linux Server 6（32-bit x86）および Red Hat Enterprise Linux Server 6（64-bit x86_64）を追加した。 • uCosminexus Application Server Enterprise • uCosminexus Application Server Standard • uCosminexus Application Server Standard-R • uCosminexus Service Platform • uCosminexus Web Redirector	－

単なる誤字・脱字などはお断りなく訂正しました。

変更内容（3020-3-U06-40）uCosminexus Application Server Enterprise 08-53，uCosminexus Application Server Standard 08-53，uCosminexus Application Server Standard-R 08-53，uCosminexus Client 08-53，uCosminexus Developer Professional 08-53，uCosminexus Developer Standard 08-53，uCosminexus Service Architect 08-53，uCosminexus Service Platform 08-53，uCosminexus Web Redirector 08-53

追加・変更内容
機能の分類とマニュアルの対応の説明に，次の機能を追加した。 • JavaMail の利用
アプリケーションサーバ 08-53 での主な機能変更についての説明を追加した。また，アプリケーションサーバ 08-50 での主な機能変更についての説明を付録 B.1 に移動した。

追加・変更内容
Connector 1.5 トランザクションに参加するメッセージ配信をサポートし，トランザクション属性に Required を指定できるようにした。
PRFSPOOL 環境変数の設定に関する注意事項を追加した。
remove メソッドによるリファレンスの解放に関する説明を追加した。
Message-driven Bean のトランザクション設定時の注意（Connector 1.5 に準拠したリソースアダプタを使用する場合）に，TP1 インバウンド連携機能を使用する場合の説明を追加した。
Microsoft IIS 7.0 および Microsoft IIS 7.5 に対応した。
SQL Server 2008 に対応した。これに伴い，使用できる JDBC ドライバに SQL Server JDBC Driver 2.0，および SQL Server JDBC Driver 3.0 を追加した。
対象製品として uCosminexus Application Server Standard-R を追加した。
次の製品の適用 OS から AIX，HP-UX，Linux（IPF）を削除した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise • uCosminexus Service Platform

変更内容（3020-3-U06-20）uCosminexus Application Server Enterprise 08-50，uCosminexus Application Server Standard 08-50，uCosminexus Client 08-50，uCosminexus Developer Professional 08-50，uCosminexus Developer Standard 08-50，uCosminexus Service Architect 08-50，uCosminexus Service Platform 08-50，uCosminexus Web Redirector 08-50

追加・変更内容
機能の分類とマニュアルの対応の説明に，次の機能を追加した。 • OpenTP1 からのアプリケーションサーバの呼び出し（TP1 インバウンド連携機能） • Cosminexus JMS プロバイダ • スレッドの非同期並行処理 • ホスト単位管理モデルを対象にした系切り替えシステム（クラスタソフトウェアとの連携）
アプリケーションサーバ独自の機能として，CMT でトランザクション管理をする場合に，Stateful Session Bean（SessionSynchronization）にトランザクション属性として Supports，NotSupported および Never を指定できるようにした。
EJB でジェネリクスを使用できるようにした。
アプリケーションサーバ 08-50 での主な機能変更についての説明を追加した。また，アプリケーションサーバ 08-00 での主な機能変更についての説明を付録 B.1 に移動した。
ビジネスメソッドとして使用できないメソッドについて，説明を追加した。
次の製品の適用 OS に Windows Server 2008 R2 を追加した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise • uCosminexus Web Redirector • uCosminexus Service Platform • uCosminexus Client
次の製品の適用 OS から Solaris を削除した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise

追加・変更内容
次の製品の適用 OS に Windows 7 を追加した。 • uCosminexus Developer Standard • uCosminexus Developer Professional • uCosminexus Service Architect • uCosminexus Client

はじめに

このマニュアルは，Cosminexus（コズミネクサス）のアプリケーションサーバの機能について説明したものです。このマニュアルでは，Enterprise Bean の実行基盤である，EJB コンテナの機能，および EJB コンテナのクライアントである EJB クライアントアプリケーションの機能について説明します。
アプリケーションサーバでは，次に示すプログラムプロダクトを使用してシステムを構築，運用します。

- uCosminexus Application Server Enterprise
- uCosminexus Application Server Standard
- uCosminexus Application Server Standard-R
- uCosminexus Client
- uCosminexus Developer Professional
- uCosminexus Developer Standard
- uCosminexus Service Architect
- uCosminexus Service Platform
- uCosminexus Web Redirector

このマニュアルでは，これらのプログラムプロダクトの構成ソフトウェアのうち，次に示す構成ソフトウェアについて説明しています。

- Cosminexus Component Container
- Cosminexus Component Container - Client
- Cosminexus Component Container - Redirector
- Cosminexus Component Transaction Monitor
- Cosminexus Developer's Kit for Java
- Cosminexus Performance Tracer
- Cosminexus TPBroker

なお，オペレーティングシステム（OS）の種類によって，機能が異なる場合があります。

■対象読者

このマニュアルは，アプリケーションサーバのシステムを設計，構築または運用する方，およびアプリケーションサーバで動作するアプリケーションを開発する方を対象としています。
次の内容を理解されていることを前提としています。

システムを設計，構築または運用する方

- Windows またはご使用の UNIX（AIX，HP-UX，Linux(R)）のシステムの設計，構築および運用に関する知識
- Java EE に関する知識
- SQL およびリレーショナルデータベースに関する基本的な知識
- CORBA に関する基本的な知識

JP1 連携機能を使用する場合は，次の内容も理解されていることを前提とします。

- JP1 の統合管理，ジョブ管理，ネットワーク管理およびアベイラビリティ管理に関する基本的な知識

アプリケーションを開発する方

- Windows の基本操作に関する知識
- Java によるプログラム開発に関する基本的な知識
- 使用する IDE（Eclipse）に関する基本的な知識

■ご利用の製品ごとの用語の読み替えについて

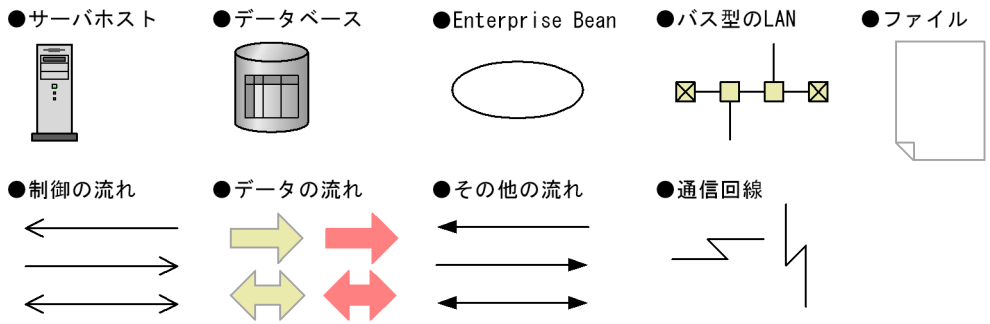
ご利用の製品によっては，マニュアルで使用している用語を，ご利用の製品名に読み替える必要があります。
次の表に従って，マニュアルで使用している用語をご利用の製品名に読み替えてください。

ご利用の製品名	マニュアルで使用している用語
uCosminexus Application Server Standard-R	Application Server
uCosminexus Developer Professional ※1	Application Server および Application Server Enterprise
uCosminexus Developer Standard ※1，※2	Application Server
uCosminexus Service Architect ※1	Application Server および Application Server Enterprise
uCosminexus Service Platform	

注※1　テスト環境で使用している場合にだけ読み替えが必要です。
注※2　uCosminexus Developer Standard と Application Server には一部機能差があります。機能差については，マニュアル「Cosminexus アプリケーションサーバ アプリケーション開発ガイド」の「付録 D　Developer Standard 使用時の注意事項」を参照してください。

■図中で使用している記号

このマニュアルの図中で使用している記号について次に示します。



目次

1 アプリケーションサーバの機能 1

1.1 機能の分類 2

1.1.1 アプリケーションの実行基盤としての機能 3

1.1.2 アプリケーションの実行基盤を運用・保守するための機能 4

1.1.3 機能とマニュアルの対応 5

1.2 システムの目的と機能の対応 8

1.2.1 EJB コンテナの機能 8

1.2.2 EJB クライアントの機能 9

1.3 このマニュアルに記載している機能の説明 11

1.3.1 分類の意味 11

1.3.2 分類を示す表の例 11

1.4 アプリケーションサーバ 08-70 での主な機能変更 13

2 EJB コンテナ 17

2.1 この章の構成 19

2.2 Enterprise Bean の実行 21

2.2.1 Enterprise Bean の種類 21

2.2.2 Enterprise Bean のインタフェース 24

2.2.3 Enterprise Bean のライフサイクル 28

2.3 EJB 仕様準拠のチェック 34

2.4 CMP フィールドとデータ型のマッピング 35

2.4.1 CMP でサポートする Java データ型の範囲 35

2.4.2 CMP フィールドとデータベースのマッピング 36

2.4.3 CMP を使用する場合の注意事項 40

2.5 EJB コンテナの JNDI 名前空間へのリファレンス登録 41

2.5.1 java:comp/env 名前空間へのリファレンスの登録 41

2.5.2 cosminexus.xml での定義 42

2.5.3 実行環境での設定 42

2.6 外部リソースとの接続 44

2.7 Enterprise Bean でのトランザクション管理 45

2.7.1 Enterprise Bean でのトランザクション管理方法の種類 45

2.7.2 BMT 46

2.7.3 CMT 47

2.7.4 cosminexus.xml での定義 56
2.7.5 実行環境での設定 57
2.8 Entity Bean のキャッシュモデル 58
2.8.1 Entity Bean のキャッシュモデルの種類 58
2.8.2 cosminexus.xml での定義 59
2.8.3 実行環境での設定 59
2.9 Enterprise Bean のプールの管理 60
2.9.1 Stateless Session Bean のプーリング 60
2.9.2 Entity Bean のプーリング 60
2.9.3 Message-driven Bean のプーリング 61
2.9.4 cosminexus.xml での定義 61
2.9.5 実行環境での設定 62
2.10 Enterprise Bean へのアクセス制御 63
2.10.1 Enterprise Bean へのアクセス制御の抑止 63
2.10.2 実行環境での設定 64
2.11 EJB コンテナでのタイムアウトの設定 65
2.11.1 タイムアウトの種類 65
2.11.2 Stateful Session Bean のタイムアウト 66
2.11.3 Entity Bean の EJB オブジェクトのタイムアウト 66
2.11.4 インスタンス取得待ちのタイムアウト 67
2.11.5 RMI-IIOP 通信のタイムアウト 67
2.11.6 cosminexus.xml での定義 69
2.11.7 RMI-IIOP 通信のタイムアウトの実装 70
2.11.8 実行環境での設定 70
2.11.9 通信のタイムアウト設定時の注意事項 72
2.12 Timer Service の機能 74
2.12.1 Timer Service の概要 74
2.12.2 EJB タイマの生成とコールバック実行時の動作 78
2.12.3 J2EE サーバ起動時の EJB タイマの自動生成（サーブレット方式） 81
2.12.4 J2EE サーバ起動時の EJB タイマの自動生成（Management イベント方式） 83
2.12.5 EJB タイマの削除 84
2.12.6 Timer Service の運用機能 85
2.12.7 EJB タイマとコールバックの動作 87
2.12.8 Timer Service を使用するアプリケーションの実装 92
2.12.9 Timer Service 実装時の注意事項 94
2.12.10 実行環境での設定 97
2.12.11 Timer Service を利用する場合の注意事項 97

2.13 EJB のリモートインタフェースの呼び出し 99
2.13.1 EJB のリモートインタフェースでのローカル呼び出しの最適化 99
2.13.2 EJB のリモートインタフェースの値の参照渡し 100
2.13.3 EJB のリモートインタフェースの通信障害発生時の動作 101
2.13.4 cosminexus.xml での定義 103
2.13.5 実行環境での設定 103
2.13.6 EJB のリモートインタフェースの呼び出しに関する注意事項 105
2.14 EJB コンテナの通信ポートと IP アドレスの固定（TPBroker のオプション） 108
2.14.1 通信ポートの固定 108
2.14.2 IP アドレスの固定 108
2.14.3 実行環境での設定 109
2.15 インターセプタの使用 110
2.15.1 インターセプタの使用の概要 110
2.15.2 アノテーションまたは DD での定義 110
2.15.3 上位レベルインターセプタの呼び出し抑止 114
2.15.4 インターセプタの実行順序 114
2.15.5 実行環境での設定 120
2.15.6 インターセプタに関する注意事項 120

3 EJB クライアント 121
3.1 この章の構成 122
3.2 EJB クライアントで使用できる機能 123
3.3 EJB クライアントアプリケーションの開始 125
3.3.1 EJB クライアントアプリケーション開始に使用するコマンド 125
3.3.2 cjclstartap コマンドの場合 126
3.3.3 vbj コマンドの場合 128
3.3.4 EJB クライアントアプリケーションの実行に必要な環境変数の設定 129
3.3.5 EJB クライアントアプリケーションのプロパティの設定 131
3.4 Enterprise Bean の呼び出し 132
3.4.1 EJB クライアントアプリケーションからの Enterprise Bean 呼び出しの流れ 132
3.4.2 Enterprise Bean を呼び出すための実装 133
3.5 EJB クライアントアプリケーションでのトランザクションの実装 136
3.5.1 EJB クライアントでトランザクションを使用する手順 136
3.5.2 ルックアップを使用した UserTransaction の取得方法 138
3.5.3 EJB クライアントアプリケーションでのトランザクション実装時の注意事項 139
3.6 EJB クライアントアプリケーションでのセキュリティの実装 141

3.6.1 セキュリティを実装する場合の前提条件 141
3.6.2 セキュリティを実装した場合のサンプルプログラム 142
3.7 RMI-IIOP スタブ，インタフェースの取得 144
3.7.1 RMI-IIOP スタブ，インタフェースの取得の概要 144
3.7.2 サーバ管理コマンドによる手動ダウンロード 145
3.7.3 ダイナミッククラスローディング 145
3.7.4 EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定 146
3.7.5 uCosminexus Client 使用時の注意 149
3.8 EJB クライアントアプリケーションのシステムログ出力 150
3.8.1 EJB クライアントアプリケーションのシステムログの概要 150
3.8.2 システムログの出力先のサブディレクトリ 150
3.8.3 システムログの出力先や出力レベルの変更 152
3.8.4 複数プロセスでのログ出力先のサブディレクトリの共有 155
3.8.5 ログ出力先のサブディレクトリ数の管理 156
3.8.6 ログ出力先ディレクトリのアクセス権の設定 156

4 Enterprise Bean 実装時の注意事項 159
4.1 この章の構成 160
4.2 Enterprise Bean 共通の注意事項 162
4.2.1 Enterprise Bean および関連するクラスの命名規則 162
4.2.2 リソースのコネクションの取得と解放 163
4.2.3 ローカルインタフェースとリモートインタフェースの使い分け 163
4.2.4 ローカル呼び出し最適化機能の利用について 164
4.2.5 ほかの J2EE アプリケーション内にある Enterprise Bean をコンポーネントインタフェースによって呼び出す方法 164
4.2.6 ほかの J2EE アプリケーション内にある Enterprise Bean をビジネスインタフェースによって呼び出す方法 166
4.2.7 クラスローダの取得に関する注意 167
4.2.8 URLConnection クラス使用時の注意 168
4.2.9 ネイティブライブラリのロードに関する注意 168
4.2.10 Entity Bean（CMP，BMP 共通）のアクセス排他のタイムアウトについて 168
4.2.11 Entity Bean（CMP，BMP 共通）使用時のデッドロックの発生について 168
4.2.12 javax.ejb.EJBContext インタフェースメソッドについての注意事項 169
4.2.13 Entity Bean（CMP，BMP 共通）属性ファイルの <prim-key-class> タグについて 170
4.2.14 EJB の仕様に関する注意 170
4.2.15 Unicode の補助文字の送受信に関する注意 170

4.2.16 EJB 3.0 に関する注意 171
4.2.17 ジェネリクスの使用に関する注意 171
4.3 Enterprise Bean の種類ごとの注意事項 175
4.3.1 Stateless Session Bean 実装時の注意事項 175
4.3.2 Stateful Session Bean 実装時の注意事項 176
4.3.3 Entity Bean（BMP）実装時の注意事項 177
4.3.4 Entity Bean（CMP）実装時の注意事項 178
4.3.5 Message-driven Bean 実装時の注意事項 179

付録 181
付録 A uCosminexus Client 182
付録 A.1 uCosminexus Client の機能 182
付録 A.2 インストール手順 182
付録 A.3 uCosminexus Client のディレクトリ構成 184
付録 B 各バージョンでの主な機能変更 185
付録 B.1 08-53 での主な機能変更 185
付録 B.2 08-50 での主な機能変更 187
付録 B.3 08-00 での主な機能変更 190
付録 C このマニュアルの参考情報 194
付録 C.1 関連マニュアル 194
付録 C.2 マニュアル体系の変更について 197
付録 C.3 このマニュアルでの表記 197
付録 C.4 英略語 201
付録 C.5 KB（キロバイト）などの単位表記について 202
付録 D 用語解説 203

索引 213

1 アプリケーションサーバの機能

この章では，アプリケーションサーバの機能の分類と目的，および機能とマニュアルの対応について説明します。また，このバージョンで変更した機能についても説明しています。

1.1 **機能の分類**

1.2 **システムの目的と機能の対応**

1.3 **このマニュアルに記載している機能の説明**

1.4 **アプリケーションサーバ** 08-70 **での主な機能変更**

1.1 機能の分類

アプリケーションサーバは，Java EE 5 に準拠した J2EE サーバを中心としたアプリケーションの実行環境を構築したり，実行環境上で動作するアプリケーションを開発したりするための製品です。Java EE の標準仕様に準拠した機能や，アプリケーションサーバで独自に拡張された機能など，多様な機能を使用できます。目的や用途に応じた機能を選択して使用することで，信頼性の高いシステムや，処理性能に優れたシステムを構築・運用できます。

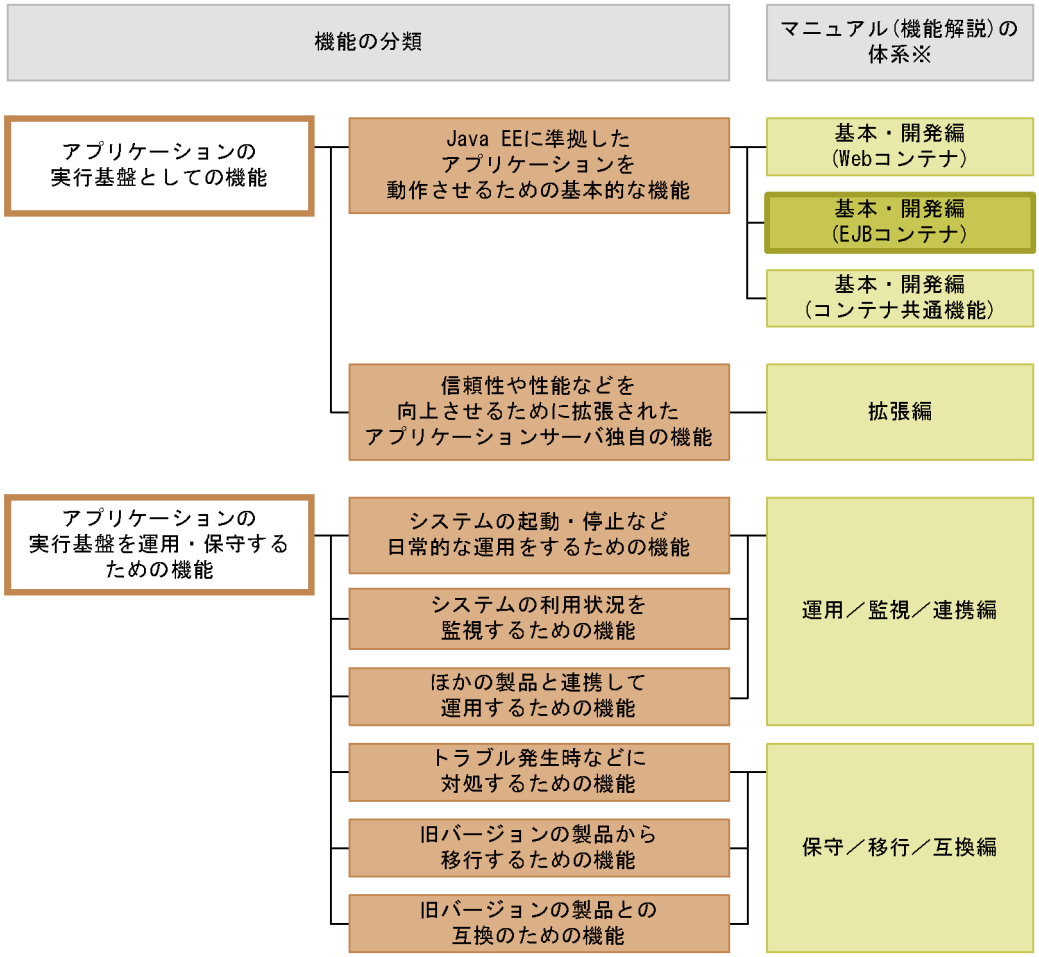
アプリケーションサーバの機能は，大きく分けて，次の二つに分類できます。

- アプリケーションの実行基盤としての機能
- アプリケーションの実行基盤を運用・保守するための機能

二つの分類は，機能の位置づけや用途によって，さらに詳細に分類できます。アプリケーションサーバのマニュアルは，機能の分類に合わせて提供しています。

アプリケーションサーバの機能の分類と対応するマニュアル（機能解説）の体系について，次の図に示します。

図 1-1　アプリケーションサーバの機能の分類と対応するマニュアル（機能解説）の体系



注※　マニュアル名称の「Cosminexus アプリケーションサーバ 機能解説」を省略しています。

ここでは，機能の分類について，マニュアルとの対応と合わせて説明します。

1.1.1　アプリケーションの実行基盤としての機能

アプリケーションとして実装されたオンライン業務やバッチ業務を実行する基盤となる機能です。システムの用途や求められる要件に応じて，使用する機能を選択します。

アプリケーションの実行基盤としての機能を使用するかどうかは，システム構築やアプリケーション開発よりも前に検討する必要があります。

アプリケーションの実行基盤としての機能について，分類ごとに説明します。

(1) Java EE に準拠したアプリケーションを動作させるための基本的な機能（基本・開発機能）

Java EE に準拠したアプリケーション（J2EE アプリケーション）を動作させるための基本的な機能が該当します。主に J2EE サーバの機能が該当します。

アプリケーションサーバでは，Java EE 5 に準拠した J2EE サーバを提供しています。J2EE サーバでは，標準仕様に準拠した機能のほか，アプリケーションサーバ独自の機能も提供しています。

基本・開発機能は，機能を使用する J2EE アプリケーションの形態に応じて，さらに三つに分類できます。アプリケーションサーバの機能解説のマニュアルは，この分類に応じて分冊されています。

それぞれの分類の概要を説明します。

- **Web アプリケーションを実行するための機能（Web コンテナ）**
  Web アプリケーションの実行基盤である Web コンテナの機能，および Web コンテナと Web サーバが連携して実現する機能が該当します。
- **Enterprise Bean を実行するための機能（EJB コンテナ）**
  Enterprise Bean の実行基盤である EJB コンテナの機能が該当します。また，Enterprise Bean を呼び出す EJB クライアントの機能も該当します。
- **Web アプリケーションと Enterprise Bean の両方で使用する機能（コンテナ共通機能）**
  Web コンテナ上で動作する Web アプリケーションおよび EJB コンテナ上で動作する Enterprise Bean の両方で使用できる機能が該当します。

(2) 信頼性や性能などを向上させるために拡張されたアプリケーションサーバ独自の機能（拡張機能）

アプリケーションサーバで独自に拡張された機能が該当します。バッチサーバ，CTM，データベースなど，J2EE サーバ以外のプロセスを使用して実現する機能も含まれます。

アプリケーションサーバでは，システムの信頼性を高め，安定稼働を実現するための多様な機能が拡張されています。また，J2EE アプリケーション以外のアプリケーション（バッチアプリケーション）を Java の環境で動作させる機能も拡張しています。

1.1.2 アプリケーションの実行基盤を運用・保守するための機能

アプリケーションの実行基盤を効率良く運用したり，保守したりするための機能です。システムの運用開始後に，必要に応じて使用します。ただし，機能によっては，あらかじめ設定やアプリケーションの実装が必要なものがあります。

アプリケーションの実行基盤を運用・保守するための機能について，分類ごとに説明し

ます。

(1) システムの起動・停止など日常的な運用をするための機能（運用機能）

システムの起動や停止，アプリケーションの開始や停止，入れ替えなどの，日常運用で使用する機能が該当します。

(2) システムの利用状況を監視するための機能（監視機能）

システムの稼働状態や，リソースの枯渇状態などを監視するための機能が該当します。また，システムの操作履歴など，監査で使用する情報を出力する機能も該当します。

(3) ほかの製品と連携して運用するための機能（連携機能）

JP1 やクラスタソフトウェアなど，ほかの製品と連携して実現する機能が該当します。

(4) トラブル発生時などに対処するための機能（保守機能）

トラブルシューティングのための機能が該当します。トラブルシューティング時に参照する情報を出力するための機能も含みます。

(5) 旧バージョンの製品から移行するための機能（移行機能）

旧バージョンのアプリケーションサーバから新しいバージョンのアプリケーションサーバに移行するための機能が該当します。

(6) 旧バージョンの製品との互換のための機能（互換機能）

旧バージョンのアプリケーションサーバとの互換用の機能が該当します。なお，互換機能については，「1.1.1　アプリケーションの実行基盤としての機能」に該当する機能などの推奨機能に移行することをお勧めします。

1.1.3 機能とマニュアルの対応

アプリケーションサーバの機能解説のマニュアルは，機能の分類に合わせて分冊されています。

機能の分類と，それぞれの機能について説明しているマニュアルとの対応を次の表に示します。

表 1-1　機能の分類とマニュアルの対応

分類	機能	マニュアル※1
基本・開発機能	Web コンテナ	基本・開発編（Web コンテナ）
	Web サーバ連携	
	インプロセス HTTP サーバ	
	サーブレットおよび JSP の実装	

分類	機能	マニュアル※1
	EJB コンテナ	基本・開発編 (EJB コンテナ) ※2
	EJB クライアント	
	Enterprise Bean 実装時の注意事項	
	ネーミング管理	基本・開発編 (コンテナ共通機能)
	リソース接続とトランザクション管理	
	OpenTP1 からのアプリケーションサーバの呼び出し（TP1 インバウンド連携機能）	
	アプリケーションサーバでの JPA の利用	
	Cosminexus JPA プロバイダ	
	Cosminexus JMS プロバイダ	
	JavaMail の利用	
	セキュリティ管理	
	アプリケーションの属性管理	
	アノテーションの使用	
	J2EE アプリケーションの形式とデプロイ	
	コンテナ拡張ライブラリ	
拡張機能	バッチサーバによるアプリケーションの実行	拡張編
	CTM によるリクエストのスケジューリングと負荷分散	
	バッチアプリケーションのスケジューリング	
	J2EE サーバ間のセッション情報の引き継ぎ（セッションフェイルオーバ機能）	
	明示管理ヒープ機能を使用したフルガーベージコレクションの抑止	
	クライアント性能の測定と分析	
	統合ユーザ管理	
	複数の構築済み実行環境の切り替え	
	アプリケーションのユーザログ出力	
	スレッドの非同期並行処理	
運用機能	システムの起動と停止	運用／監視／連携編
	J2EE アプリケーションの運用	
監視機能	稼働情報の監視（稼働情報収集機能）	
	リソースの枯渇監視	
	監査ログ出力機能	
	データベース監査証跡連携機能	

分類	機能	マニュアル※1
	運用管理コマンドによる稼働情報の出力	
	Management イベントの通知と Management アクションによる処理の自動実行	
	CTM の稼働統計情報の収集	
	コンソールログの出力	
連携機能	JP1 と連携したシステムの運用	
	システムの構成定義および管理（JP1/IM-CM との連携）	
	システムの集中監視（JP1/IM との連携）	
	ジョブによるシステムの自動運転（JP1/AJS との連携）	
	シナリオによるシステムの自動運転（JP1/AJS2 - SO との連携）	
	監査ログの収集および一元管理（JP1/NETM/Audit との連携）	
	クラスタソフトウェアとの連携	
	1:1 系切り替えシステム（クラスタソフトウェアとの連携）	
	相互系切り替えシステム（クラスタソフトウェアとの連携）	
	N:1 リカバリシステム（クラスタソフトウェアとの連携）	
	ホスト単位管理モデルを対象にした系切り替えシステム（クラスタソフトウェアとの連携）	
保守機能	トラブルシューティング関連機能	保守／移行／互換編
	性能解析トレースを使用したシステムの性能解析	
	日立固有の JavaVM の機能	
移行機能	旧バージョンのアプリケーションサーバからの移行	
	推奨機能への移行	
互換機能	旧バージョンとの互換用機能（ベーシックモード）	
	旧バージョンとの互換用機能（サーブレットエンジンモード）	
	Cosminexus DABroker Library を使用したデータベース接続	

注※1　マニュアル名称の「Cosminexus アプリケーションサーバ 機能解説」を省略しています。
注※2　このマニュアルです。

1.2 システムの目的と機能の対応

アプリケーションサーバでは，構築・運用するシステムの目的に合わせて，適用する機能を選択する必要があります。

この節では，Enterprise Bean を実行するためのそれぞれの機能をどのようなシステムの場合に使用するとよいかを示します。機能ごとに，次の項目への対応を示しています。

● 信頼性
高い信頼が求められるシステムの場合に使用するとよい機能です。
アベイラビリティ（安定稼働性）およびフォールトトレランス（耐障害性）を高める機能や，ユーザ認証などのセキュリティを高めるための機能が該当します。

● 性能
性能を重視したシステムの場合に使用するとよい機能です。
システムのパフォーマンスチューニングで使用する機能などが該当します。

● 運用・保守
効率の良い運用・保守をしたい場合に使用するとよい機能です。

● 拡張性
システム規模の拡大・縮小および構成の変更への柔軟な対応が必要な場合に使用するとよい機能です。

● そのほか
そのほかの個別の目的に対応するための機能です。

また，Enterprise Bean を実行するための機能には，Java EE 標準機能とアプリケーションサーバが独自に拡張した機能があります。機能を選択するときには，必要に応じて，Java EE 標準への準拠についても確認してください。

1.2.1 EJB コンテナの機能

EJB コンテナの機能を次の表に示します。システムの目的に合った機能を選択してください。機能の詳細については，参照先を確認してください。

表 1-2 EJB コンテナの機能とシステムの目的の対応

機能	システムの目的					Java EE 標準への準拠		参照先
	信頼性	性能	運用・保守	拡張性	そのほか	標準	拡張	
Enterprise Bean の実行	—	—	—	—	—	○	○	2.2
EJB 仕様準拠のチェック	—	—	—	—	—	○	○	2.3
CMP フィールドとデータ型のマッピング	—	—	—	—	—	○	○	2.4

機能	システムの目的					Java EE 標準への準拠		参照先
	信頼性	性能	運用・保守	拡張性	そのほか	標準	拡張	
EJB コンテナの JNDI 名前空間へのリファレンス登録※	―	―	―	○	―	○	○	2.5
外部リソースとの接続	―	―	―	○	―	○	―	2.6
Enterprise Bean でのトランザクション管理	―	―	―	―	―	○	○	2.7
Entity Bean のキャッシュモデル（コミットオプション指定）	―	○	―	―	―	○	―	2.8
Stateless Session Bean, Entity Bean のプールの管理	―	○	―	―	―	○	○	2.9
Enterprise Bean へのアクセス制御	○	―	―	―	―	○	―	2.10
EJB コンテナでのタイムアウトの設定	―	○	―	―	―	○	○	2.11
Timer Service の機能	―	―	―	―	―	○	○	2.12
EJB のリモートインタフェースの呼び出し	―	○	―	―	―	○	○	2.13
EJB コンテナの通信ポートと IP アドレスの固定（TPBroker のオプション）	○	―	―	―	―	―	○	2.14
デフォルトインターセプタの使用	―	―	―	―	―	○	○	2.15

（凡例） ○：対応する。　　―：対応しない。

注

「Java EE 標準への準拠」の「標準」と「拡張」の両方に○が付いている機能は，Java EE 標準の機能にアプリケーションサーバ独自の機能が拡張されていることを示します。「拡張」だけに○が付いている機能はアプリケーションサーバ独自の機能であることを示します。

注※

ネーミング管理を利用して実現します。

1.2.2 EJB クライアントの機能

EJB クライアントの機能を次の表に示します。システムの目的に合った機能を選択してください。機能の詳細については，参照先を確認してください。

表 1-3 EJB クライアントの機能とシステムの目的の対応

機能	システムの目的					Java EE 標準への準拠		参照先
	信頼性	性能	運用・保守	拡張性	そのほか	標準	拡張	
EJB クライアントアプリケーションの開始	―	―	―	○	―	○	○	3.3
Enterprise Bean の呼び出し	―	―	―	○	―	○	○	3.4
EJB クライアントアプリケーションでのトランザクションの実装	―	―	―	○	―	○	○	3.5
EJB クライアントアプリケーションでのセキュリティの実装	―	―	―	○	―	○	○	3.6
RMI-IIOP スタブ，インタフェースの取得	―	―	―	○	―	○	○	3.7
EJB クライアントアプリケーションのシステムログ出力	―	―	―	○	―	○	○	3.8

（凡例） ○：対応する。　　―：対応しない。

注

「Java EE 標準への準拠」の「標準」と「拡張」の両方に○が付いている機能は，Java EE 標準の機能にアプリケーションサーバ独自の機能が拡張されていることを示します。「拡張」だけに○が付いている機能はアプリケーションサーバ独自の機能であることを示します。

1.3 このマニュアルに記載している機能の説明

ここでは，このマニュアルで機能を説明するときの分類の意味と，分類を示す表の例について説明します。

1.3.1 分類の意味

このマニュアルでは，各機能について，次の五つに分類して説明しています。マニュアルを参照する目的によって，必要な個所を選択して読むことができます。

- 解説
  機能の解説です。機能の目的，特長，仕組みなどについて説明しています。機能の概要について知りたい場合にお読みください。
- 実装
  コーディングの方法や DD の記載方法などについて説明しています。アプリケーションを開発する場合にお読みください。
- 設定
  システム構築時に必要となるプロパティなどの設定方法について説明しています。システムを構築する場合にお読みください。
- 運用
  運用方法の説明です。運用時の手順や使用するコマンドの実行例などについて説明しています。システムを運用する場合にお読みください。
- 注意事項
  機能を使用するときの全般的な注意事項について説明しています。注意事項の説明は必ずお読みください。

1.3.2 分類を示す表の例

機能説明の分類については，表で説明しています。表のタイトルは，「この章の構成」または「この節の構成」となっています。

次に，機能説明の分類を示す表の例を示します。

機能説明の分類を示す表の例

表 X-1 この章の構成（○○機能）

分類	タイトル	参照先
解説	○○機能とは	X.1
実装	アプリケーションの実装	X.2
	DD および cosminexus.xml ※での定義	X.3
設定	実行環境での設定	X.4
運用	○○機能を使用した運用	X.5

分類	タイトル	参照先
注意事項	○○機能使用時の注意事項	X.6

注※

cosminexus.xml については，マニュアル「Cosminexus 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「10. アプリケーションの属性管理」を参照してください。

ポイント

cosminexus.xml を含まないアプリケーションのプロパティ設定

cosminexus.xml を含まないアプリケーションでは，実行環境へのインポート後にプロパティを設定，または変更します。設定済みのプロパティも実行環境で変更できます。実行環境でのアプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。サーバ管理コマンドおよび属性ファイルでのアプリケーションの設定については，マニュアル「Cosminexus 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「3.3.2 J2EE アプリケーションのプロパティの設定手順」を参照してください。

属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。DD または cosminexus.xml と属性ファイルのタグの対応については，マニュアル「Cosminexus 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「3. J2EE アプリケーションの設定で使用する属性ファイル」を参照してください。

なお，各属性ファイルで設定するプロパティは，アプリケーション統合属性ファイルでも設定できます。

1.4 アプリケーションサーバ 08-70 での主な機能変更

この節では，アプリケーションサーバ 08-70 での主な機能の変更について，変更目的ごとに説明します。

説明内容は次のとおりです。

- アプリケーションサーバ 08-70 で変更になった主な機能と，その概要を説明しています。機能の詳細については参照先の記述を確認してください。「参照先マニュアル」および「参照個所」には，その機能についての主な記載個所を記載しています。
- 「参照先マニュアル」に示したマニュアル名の「Cosminexus アプリケーションサーバ V8」は省略しています。

(1) 導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 1-4 導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
運用管理ポータル改善	運用管理ポータルの画面で，リソースアダプタの属性を定義するプロパティ（Connector 属性ファイルの設定内容）の設定，および接続テストができるようになりました。また，運用管理ポータルの画面で，J2EE アプリケーション（ear ファイルおよび zip ファイル）を Management Server にアップロードできるようになりました。	ファーストステップガイド	3.4.1
		運用管理ポータル操作ガイド	―
page/tag ディレクティブの import 属性暗黙インポート機能の追加	page/tag ディレクティブの import 属性暗黙インポート機能を使用できるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.3.7
仮想化環境での JP1 製品に対する環境設定の自動化対応	仮想サーバへのアプリケーションサーバ構築時に，仮想サーバに対する JP1 製品の環境設定を，フックスクリプトで自動的に設定できるようになりました。	仮想化システム構築・運用ガイド	7.7.2

項目	変更の概要	参照先マニュアル	参照個所
統合ユーザ管理機能の改善	ユーザ情報リポジトリでデータベースを使用する場合に，データベース製品の JDBC ドライバを使用して，データベースに接続できるようになりました。Cosminexus DABroker Library の JDBC ドライバによるデータベース接続はサポート外になりました。 簡易構築定義ファイルおよび運用管理ポータルの画面で，統合ユーザ管理機能に関する設定ができるようになりました。 また，Active Directory の場合，DN で日本語などの 2 バイト文字に対応しました。	機能解説 拡張編	10 章
		運用管理ポータル操作ガイド	3.5, 10.9.1
		リファレンス 定義編 (サーバ定義)	14.3
Hitachi Web Server 設定項目の拡充	簡易構築定義ファイルおよび運用管理ポータルの画面で，Hitachi Web Server の動作環境を定義するディレクティブ（httpsd.conf（Hitachi Web Server 定義ファイル）の設定内容）を直接設定できるようになりました。	システム構築・運用ガイド	8.4.3
		運用管理ポータル操作ガイド	10.10.1
		リファレンス 定義編 (サーバ定義)	4.13

（凡例）－：マニュアル全体を参照する

(2) 標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 1-5 標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
ejb-jar.xml の指定項目の追加	ejb-jar.xml に，クラスレベルインターセプタおよびメソッドレベルインターセプタを指定できるようになりました。	このマニュアル	2.15
パラレルコピーガーベージコレクションへの対応	パラレルコピーガーベージコレクションを選択できるようになりました。	リファレンス 定義編 (サーバ定義)	19.5
Connector 1.5 仕様に準拠した Inbound リソースアダプタのグローバルトランザクションへの対応	Connector 1.5 仕様に準拠したリソースアダプタで Transacted Delivery を使用できるようにしました。これによって，Message-driven Bean を呼び出す EIS がグローバルトランザクションに参加できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	3.16.3
TP1 インバウンドアダプタの MHP への対応	TP1 インバウンドアダプタを使用してアプリケーションサーバを呼び出す OpenTP1 のクライアントとして，MHP を使用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	4 章

項目	変更の概要	参照先マニュアル	参照個所
cjrarupdate コマンドの FTP インバウンドアダプタへの対応	cjrarupdate コマンドでバージョンアップできるリソースアダプタに FTP インバウンドアダプタを追加しました。	リファレンス コマンド編	2.2

(3) 信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 1-6 信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
データベースセッションフェイルオーバ機能の改善	性能を重視するシステムで，グローバルセッション情報を格納したデータベースのロックを取得しないモードを選択できるようになりました。また，データベースを更新しない，参照専用のリクエストを定義できるようになりました。	機能解説 拡張編	6 章
OutOfMemory ハンドリング機能の対象となる処理の拡大	OutOfMemory ハンドリング機能の対象となる処理を追加しました。	機能解説 保守／移行／互換編	2.5.6
		リファレンス 定義編 (サーバ定義)	19.2
HTTP セッションで利用する Explicit ヒープの省メモリ化機能の追加	HTTP セッションで利用する Explicit ヒープのメモリ使用量を抑止する機能を追加しました。	機能解説 拡張編	8.9

(4) 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 1-7 運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
仮想化環境での JP1 製品を使用したユーザ認証への対応（クラウド運用対応）	JP1 連携時に，JP1 製品の認証サーバを利用して，仮想サーバマネージャを使用するユーザを管理・認証できるようになりました。	仮想化システム構築・運用ガイド	1.2.2, 3 章, 4 章, 5 章, 6 章, 7.8

(5) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 1-8 そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
負荷分散機への API（REST アーキテクチャ）を使用した直接接続の対応	負荷分散機への接続方法として，API（REST アーキテクチャ）を使用した直接接続に対応しました。 また，使用できる負荷分散機の種類に ACOS（AX2500）が追加になりました。	システム構築・運用ガイド	4.3.1
		仮想化システム構築・運用ガイド	2.1
		リファレンス 定義編 (サーバ定義)	4.5
snapshot ログ収集時のタイムアウトへの対応と収集対象の改善	snapshot ログの収集が指定した時間で終了（タイムアウト）できるようになりました。 一次送付資料として収集される内容が変更になりました。	機能解説 保守／移行／互換編	付録 A

2 EJB コンテナ

この章では，Enterprise Bean の実行基盤である，EJB コンテナで利用できる機能について説明します。EJB コンテナの機能は，Enterprise Bean を使用した J2EE アプリケーションを実行する場合に使用します。
なお，EJB コンテナの機能のうち，JNDI 名前空間機能，Enterprise Bean でのトランザクション設定やアクセス制御では，J2EE サービスの機能も利用します。マニュアル「Cosminexus 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「3. リソース接続とトランザクション管理」もあわせて参照してください。

2.1 この章の構成

2.2 Enterprise Bean の実行

2.3 EJB 仕様準拠のチェック

2.4 CMP フィールドとデータ型のマッピング

2.5 EJB コンテナの JNDI 名前空間へのリファレンス登録

2.6 外部リソースとの接続

2.7 Enterprise Bean でのトランザクション管理

2.8 Entity Bean のキャッシュモデル

2.9 Enterprise Bean のプールの管理

2.10 Enterprise Bean へのアクセス制御

2.11 EJB コンテナでのタイムアウトの設定

2.12 Timer Service の機能

2.13 EJB のリモートインタフェースの呼び出し

2.14 EJB コンテナの通信ポートと IP アドレスの固定（TPBroker のオプション）

2.15 インターセプタの使用

2.1 この章の構成

EJB コンテナは，Enterprise Bean の実行を制御し，Enterprise Bean に各種のサービスを提供する実行環境です。

EJB コンテナの機能と参照先を次の表に示します。

表 2-1 EJB コンテナの機能と参照先

機能	参照先
Enterprise Bean の実行	2.2
EJB 仕様準拠のチェック	2.3
CMP フィールドとデータ型のマッピング	2.4
EJB コンテナの JNDI 名前空間へのリファレンス登録※ 1	2.5
外部リソースとの接続	2.6
Enterprise Bean でのトランザクション管理※ 2	2.7
Entity Bean のキャッシュモデル	2.8
Enterprise Bean のプールの管理	2.9
Enterprise Bean へのアクセス制御	2.10
EJB コンテナでのタイムアウトの設定	2.11
Timer Service の機能	2.12
EJB のリモートインタフェースの呼び出し	2.13
EJB コンテナの通信ポートと IP アドレスの固定（TPBroker のオプション）	2.14
インターセプタの使用	2.15

注※ 1

J2EE サービスのネーミング管理機能の利用によって実現します。管理機能の概要については，マニュアル「Cosminexus 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「2. ネーミング管理」を参照してください。

注※ 2

J2EE サービスのトランザクション管理機能の利用によって実現します。J2EE サービスのトランザクション管理の概要については，マニュアル「Cosminexus 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「3. リソース接続とトランザクション管理」を参照してください。

なお，アプリケーションサーバで提供する EJB コンテナの機能には，JavaEE で規定された機能にアプリケーションサーバ独自の機能を拡張したものと，アプリケーションサーバ独自の機能として提供しているものがあります。アプリケーションサーバ独自の機能かどうかについては，「1. アプリケーションサーバの機能」を参照してください。

また，EJB クライアントで使用できる機能については，「3. EJB クライアント」を参照してください。

2.2 Enterprise Bean の実行

この節では，Enterprise Bean を実行する機能について説明します。

Enterprise Bean は，ビジネスロジックを EJB アーキテクチャに従って作成したプログラムです。業務処理プログラムに該当します。Enterprise Bean は，EJB コンテナ上で動作します。

この節の構成を次の表に示します。

表 2-2 この節の構成（Enterprise Bean の実行）

分類	タイトル	参照先
解説	Enterprise Bean の種類	2.2.1
	Enterprise Bean のインタフェース	2.2.2
	Enterprise Bean のライフサイクル	2.2.3

注 「実装」,「設定」,「運用」および「注意事項」について，この機能固有の説明はありません。

2.2.1 Enterprise Bean の種類

アプリケーションサーバで提供している EJB コンテナでは，次の表に示す Enterprise Bean を実行できます。

表 2-3 EJB コンテナで実行できる Enterprise Bean の分類

大分類	小分類
Session Bean	Stateless Session Bean
	Stateful Session Bean
Entity Bean	BMP（Bean Managed Persistence）
	CMP（Container Managed Persistence）
Message-driven Bean	なし

ここでは，それぞれの Enterprise Bean の特徴について説明します。

(1) Session Bean

Session Bean は，クライアントからのセッション単位に生成され，クライアントが終了すると消滅する Enterprise Bean です。Session Bean のライフサイクルが，ユーザがシステムの利用を開始してから終了するまでの範囲内を超えることはありません。Session Bean は状態管理をするかどうかによって，Stateless Session Bean と Stateful Session Bean に分類されます。

(a) Stateless Session Bean

セッションの状態を管理しないモデルです。クライアントからの 1 セッションは Bean のビジネスメソッドの 1 回の呼び出しで完結させる必要があります。

(b) Stateful Session Bean

セッションの状態を管理するモデルです。EJB コンテナが状態管理をします。クライアントからの 1 セッションが EJB の複数のビジネスメソッドを呼び出す場合でも，ビジネスメソッドの呼び出し間で状態が保存されます。

(2) Entity Bean

Entity Bean は，エンティティを表現し，データベースに保存（永続化）されることを前提としています。そのため，クライアントが終了しても Entity Bean の状態はデータベースに存在し続けます。Session Bean と比較してライフサイクルが長い Enterprise Bean です。EJB 仕様では，次の二つの管理モデルを規定しています。

(a) BMP（Bean Managed Persistence）

Enterprise Bean のビジネスメソッド内でデータの永続化を管理するモデルです。データベースへの接続，SQL の組み立てや実行などの処理は，Enterprise Bean の開発者が実装しなければなりません。

(b) CMP（Container Managed Persistence）

EJB コンテナがデータの永続化を管理するモデルです。データベースへの接続とデータの保存は EJB コンテナによって実行されるので，Enterprise Bean のビジネスメソッドで実行する必要はありません。Enterprise Bean 内のデータと格納先データベースのテーブルおよびカラムの対応を EJB コンテナが提供する方法で定義します。一方，接続先データベースのホスト名やポート番号などの接続情報をリソースアダプタまたはデータソースに定義します。EJB コンテナは，これらの定義情報を参照して SQL 文を組み立て，接続先データベースのテーブルにデータの参照および格納をします。

なお，EJB 2.0 で追加された CMP 2.0 では，EJB QL によって，使用するデータベースに依存することなく，データベースの検索処理を SQL のような構文で DD に記述できます。また，Entity Bean 同士に関連を持たせる CMR（Container-managed relationship）によって，Entity Bean 間の関連を DD で設定し，EJB コンテナで管理できます。

アプリケーションサーバで提供する EJB コンテナの CMP 機能での Entity Bean の Java データ型とデータベースの SQL データ型の対応については，「2.4.2　CMP フィールドとデータベースのマッピング」を参照してください。

(3) Message-driven Bean

Message-driven Bean は，JMS と連携するメッセージ駆動タイプの Bean です。EJB コ

ンテナは JMS の Destination からの JMS メッセージ受信を契機に Bean を起動します。Session Bean または Entity Bean と異なり，ホームインタフェース，コンポーネントインタフェースを持たないため，クライアントから直接呼び出されません。

Message-driven Bean は，EJB 2.0 の場合と EJB 2.1 以降の場合で実装するインタフェースが異なります。

- EJB 2.0 の場合は，次のインタフェースを実装します。
  - javax.ejb.MessageDrivenBean インタフェース
  - javax.jms.MessageListener インタフェース
- EJB 2.1 以降の場合は，次のインタフェースを実装します。
  - javax.ejb.MessageDrivenBean インタフェース
  - EIS が提供する任意のメッセージリスナのインタフェース

EJB 2.0 と EJB 2.1 以降では，それぞれ対応する Connector のバージョンが異なります。EJB と Connector のバージョンの対応を次の表に示します。

表 2-4　EJB と Connector のバージョンの対応

EJB のバージョン	Connector のバージョン	
	Connector 1.0	Connector 1.5
EJB 2.0	○	×
EJB 2.1 以降	○	◎

（凡例）

○：Cosminexus Reliable Messaging または TP1/Message Queue - Access から送信されたメッセージを受信できる。

◎：任意の形式のリスナインタフェースを使用して送信されたメッセージを受信できる。

×：該当するリソースアダプタから送信されたメッセージを受信できない。

また，EJB 2.0 と EJB 2.1 以降では，次の表に示す機能差があります。

表 2-5　EJB 2.0 と EJB 2.1 以降の機能差

機能	EJB 2.0	EJB 2.1 以降	
		Connector 1.0 の機能を使用する場合	Connector 1.5 の機能を使用する場合
接続できるリソースアダプタ	Connector1.0 仕様に準拠したリソースアダプタ。	Connector1.0 仕様に準拠したリソースアダプタ。	Connector 1.5 仕様に準拠し，Inbound が定義されているリソースアダプタ。

機能	EJB 2.0	EJB 2.1 以降	
		Connector 1.0 の機能を使用する場合	Connector 1.5 の機能を使用する場合
使用できる EIS	• Cosminexus Reliable Messaging • TP1/Message Queue - Access	• Cosminexus Reliable Messaging • TP1/Message Queue - Access	Connector 1.5 をサポートする任意の EIS（JMS を含みます）。※
キューの定義方法	キュー定義ファイルに定義します。	キュー定義ファイルに定義します。	リソースアダプタの DD（ra.xml）内の管理対象オブジェクトに定義します。
JMS のバージョン	JMS1.0.2b	JMS1.0.2b	JMS1.1
メッセージリスナでのコネクションの管理方法	アプリケーションの属性（pooled-instance）に指定します。 Message-driven Bean の method-ready プールと同じになります。	アプリケーションの属性（pooled-instance）に指定します。 Message-driven Bean の method-ready プールと同じになります。	使用するリソースアダプタによって異なります。

注※

Cosminexus Reliable Messaging または TP1/Message Queue - Access は Connector 1.5 仕様に対応していないため，Connector 1.5 の機能は使用できません。

2.2.2 Enterprise Bean のインタフェース

Enterprise Bean の実装に必要なインタフェースについて説明します。利用できるインタフェースの一覧を次の表に示します。

表 2-6 利用できるインタフェースの一覧

インタフェース名	説明
リモートホームインタフェース※1	EJB 仕様 1.1 以降で規定されている，javax.ejb.EJBHome を継承した，リモートクライアント用のインタフェースです。主に，Enterprise Bean インスタンスの取得のために使用します。
リモートコンポーネントインタフェース※2	EJB 仕様 1.1 以降で規定されている，javax.ejb.EJBObject を継承した，リモートクライアント用のインタフェースです。主に，ビジネスメソッドを定義します。
リモートビジネスインタフェース	リモートクライアントから Enterprise Bean を呼び出すためのビジネスメソッドを定義するインタフェースです。規定されたインタフェースを継承する必要はありません。
ローカルホームインタフェース	EJB 仕様 2.0 以降で規定されている，javax.ejb.EJBLocalHome を継承した，ローカルクライアント用のインタフェースです。主に，Enterprise Bean インスタンスの取得のために使用します。
ローカルコンポーネントインタフェース	EJB 仕様 2.0 以降で規定されている，javax.ejb.EJBLocalObject を継承した，ローカルクライアント用のインタフェースです。主に，ビジネスメソッドを定義します。

インタフェース名	説明
ローカルビジネスインタフェース	ローカルクライアントから Enterprise Bean を呼び出すためのビジネスメソッドを定義するインタフェースです。規定されたインタフェースを継承する必要はありません。

注※ 1

EJB 仕様 1.1 ではホームインタフェースと呼ばれるインタフェースです。

注※ 2

EJB 仕様 1.1 ではリモートインタフェースと呼ばれるインタフェースです。

なお，このマニュアルの説明では，複数のインタフェースをまとめた総称を使用することがあります。説明で使用するインタフェースの総称を次に示します。

表 2-7　インタフェースの総称

インタフェースの総称	説明
ホームインタフェース	次のインタフェースの総称です。 • リモートホームインタフェース • ローカルホームインタフェース
コンポーネントインタフェース	次のインタフェースの総称です。 • リモートコンポーネントインタフェース • ローカルコンポーネントインタフェース
ビジネスインタフェース	次のインタフェースの総称です。 • リモートビジネスインタフェース • ローカルビジネスインタフェース
リモートインタフェース	次のインタフェースの総称です。 • リモートホームインタフェース • リモートコンポーネントインタフェース • リモートビジネスインタフェース ただし，リモートコンポーネントインタフェース，リモートビジネスインタフェースだけを指すこともあります。
ローカルインタフェース	次のインタフェースの総称です。 • ローカルホームインタフェース • ローカルコンポーネントインタフェース • ローカルビジネスインタフェース ただし，ローカルコンポーネントインタフェース，ローカルビジネスインタフェースだけを指すこともあります。

Session Bean または Entity Bean の場合，少なくとも一つのインタフェースを持ちます。ただし，Message-driven Bean の場合はこれらのインタフェースを持ちません。

(1) リモートインタフェース

リモートインタフェースでは Java RMI のインタフェースの規定に従い，RMI-IIOP 通信によって Enterprise Bean の呼び出しをします。クライアントが異なる JavaVM に存在する Enterprise Bean を呼び出すことができますが，実行時に通信上のオーバーヘッドが発生します。メソッド実行時の引数および戻り値は値渡し（pass by value）になり

ます。

(2) ローカルインタフェース

ローカルインタフェースの場合，Enterprise Bean の呼び出しは Java のメソッド呼び出しで実行され，通信が発生しません。クライアントは同一 J2EE アプリケーションに存在する場合にだけ，このインタフェースを利用できます。また，ローカルインタフェースを利用する場合には，リモートインタフェースと異なり，メソッド実行時の引数および戻り値は参照渡し（pass by reference）になります。

(3) ビジネスインタフェースに対応する機能

ビジネスインタフェースを使用した場合でも，次の機能を使用できます。

● ローカル呼び出し最適化機能
機能の詳細については，「2.13.1　EJB のリモートインタフェースでのローカル呼び出しの最適化」を参照してください。

● リモートインタフェースの値の参照渡し機能
機能の詳細については，「2.13.2　EJB のリモートインタフェースの値の参照渡し」を参照してください。

● RMI-IIOP 通信でのタイムアウト
機能の詳細については，「2.11.5　RMI-IIOP 通信のタイムアウト」を参照してください。
通信タイムアウトの設定は，定義ファイルまたは API で設定します。ビジネスインタフェースを使用した場合のタイムアウトの設定についての注意事項を次に示します。
- 定義ファイルで設定する場合
  DI 機能を使用するときに，EJB コンテナ内で JNDI を使用するため，JNDI の通信タイムアウトのプロパティも有効になります。
- API で設定する場合
  スレッドには設定できますが，オブジェクトには設定できません。

また，タイムアウトの発生時には，ビジネスインタフェースが java.rmi.Remote を継承している場合は java.rmi.RemoteException(org.omg.CORBA.TIMEOUT)，java.rmi.Remote を継承していない場合は javax.ejb.EJBException(RemoteException(org.omg.CORBA.TIMEOUT)) が送出されます。

● EJB のチェック機能
機能の詳細については，「2.3　EJB 仕様準拠のチェック」を参照してください。

● J2EE アプリケーションの実行時間監視のメソッドタイムアウト機能
機能の詳細については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 運用／監視／連携編」の「5.3.2　J2EE アプリケーション実行時間の監視とは」を参照してください。

アノテーションを指定した EJB のメソッド呼び出し処理では，次の表に示すメソッド単位にタイムアウト値を適用できます。

表 2-8 タイムアウトの適用範囲

インタフェース	メソッド	StatelessSession Bean	StatefulSessionBean	MessageDrivenBean	PersistenceAPI
ビジネスインタフェース	ビジネスメソッド※1	○	○	－	－
ホームまたはコンポーネントインタフェース※2	create	×	○	－	－
	ビジネスメソッド	○	○	－	－
	remove	×	○	－	－
javax.ejb.TimedObject	ejbTimeout	○	－	－	－

（凡例） ○：適用される。 ×：適用されない。 －：該当なし。

注※ 1

@Timeout アノテーション，@Remove アノテーションが付いたメソッドを含みます。

注※ 2

@RemoteHome または @LocalHome を使用したインタフェースを指します。

(4) ビジネスインタフェースの Enterprise Bean 呼び出し

ここでは，同一 J2EE サーバ内のほかの J2EE アプリケーションで動作するビジネスインタフェースを呼び出す場合と，ほかの J2EE サーバで動作するビジネスインタフェースを呼び出す場合について説明します。

どちらの場合も，呼び出し元の EJB-JAR ファイルまたは WAR ファイルに，呼び出し先の Enterprise Bean のビジネスインタフェース，およびインタフェースで使用するユーザ作成クラスを含めます。また，JNDI 名前空間に自動的にバインドされる名称またはユーザ指定名前空間機能で設定した別名を使用してルックアップします。ルックアップで使用する名称については，マニュアル「Cosminexus 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「2.3.2 EJB のリファレンスが自動的にバインドされる名称」を参照してください。

● 同一 J2EE サーバ内のほかの J2EE アプリケーションで動作するビジネスインタフェースの Enterprise Bean を呼び出す場合

呼び出し元の EJB-JAR ファイルまたは WAR ファイルに，呼び出し先の Enterprise Bean のビジネスインタフェースおよびインタフェースで使用するユーザ作成クラスを含めてください。

また，usrconf.properties の ejbserver.rmi.localinvocation.scope キーに「all」または「none」を指定してください。ただし，「none」を指定する場合は，呼び出し先の Enterprise Bean に対して cjgetstubsjar コマンドを使用してスタブを取得し，呼び出

し元の EAR ファイルに含めてください。※

● **ほかの** J2EE **サーバで動作するビジネスインタフェースの** Enterprise Bean **を呼び出す場合**

呼び出し先の Enterprise Bean に対して cjgetstubsjar コマンドを使用してスタブを取得し，呼び出し元の EAR ファイルに含めてください。※

注※

アプリケーション間のビジネスインタフェース呼び出しで，ダイナミッククラスローディング機能を使用する場合は，スタブを含める必要はありません。ただし，ダイナミッククラスローディング機能は，性能上，推奨しません。

2.2.3 Enterprise Bean のライフサイクル

Enterprise Bean のライフサイクルについて，Enterprise Bean の種類ごとに説明します。

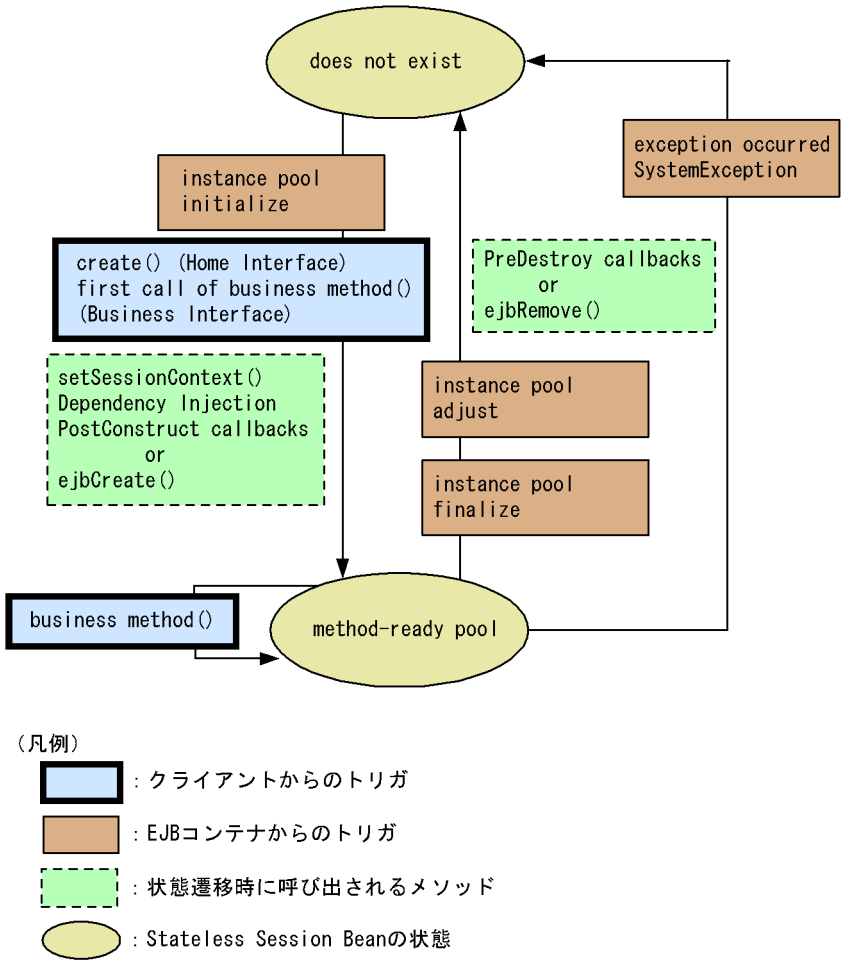
（1） Session Bean のライフサイクル

Session Bean のライフサイクルは，Stateless Session Bean の場合と Stateful Session Bean の場合とで異なります。

（a） Stateless Session Bean の場合

Stateless Session Bean のライフサイクルを次の図に示します。

図 2-1 Stateless Session Bean のライフサイクル



does not exist：

Stateless Session Bean が存在しない状態

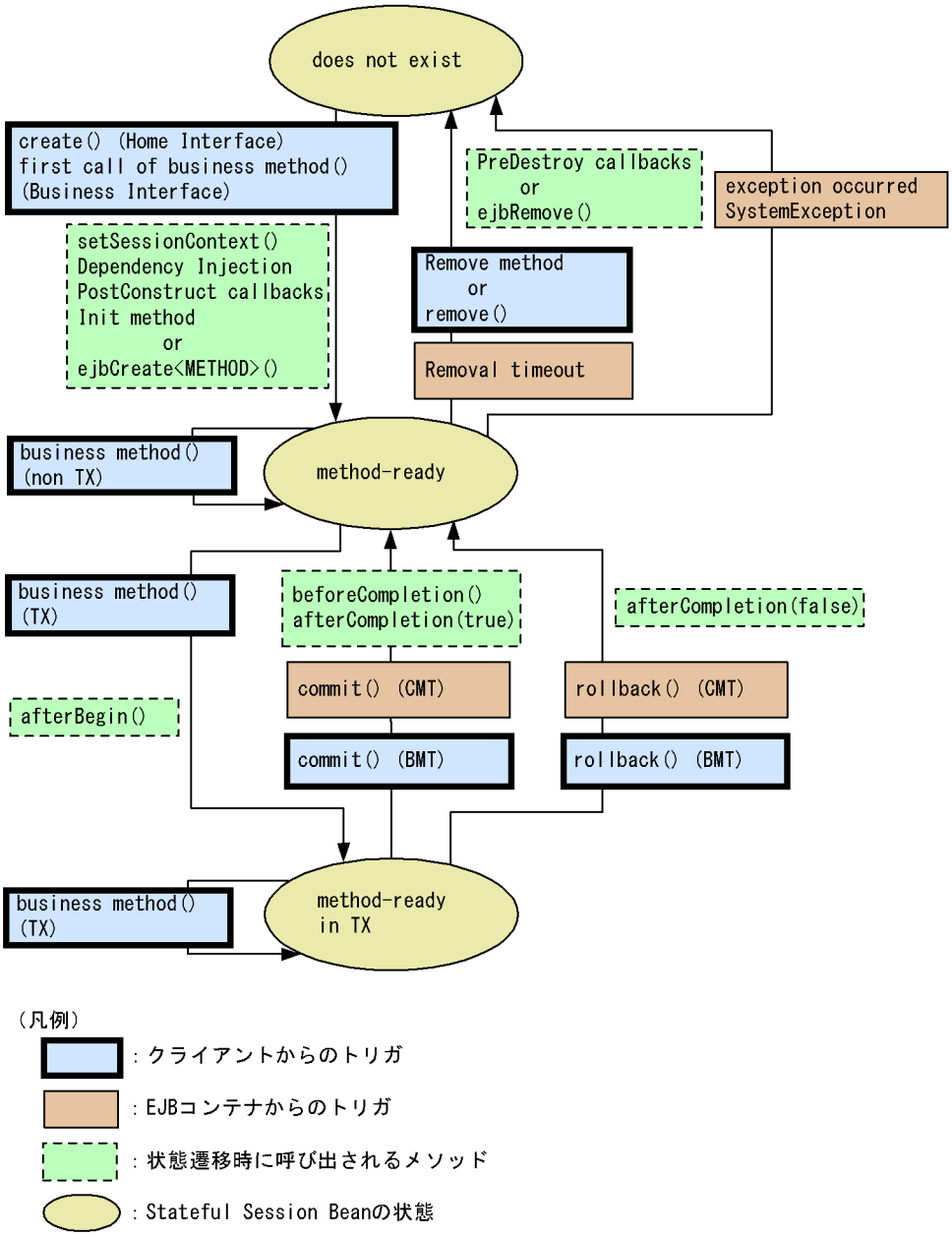
method-ready pool：

Stateless Session Bean が実行可能状態として method-ready プール内に存在する状態

(b) Stateful Session Bean の場合

Stateful Session Bean のライフサイクルを次の図に示します。

図 2-2 Stateful Session Bean のライフサイクル



does not exist：

Stateful Session Bean が存在しない状態

method-ready：

Stateful Session Bean がアクティベイトされ，実行可能状態として method-ready プール内に存在する状態（トランザクションなし）

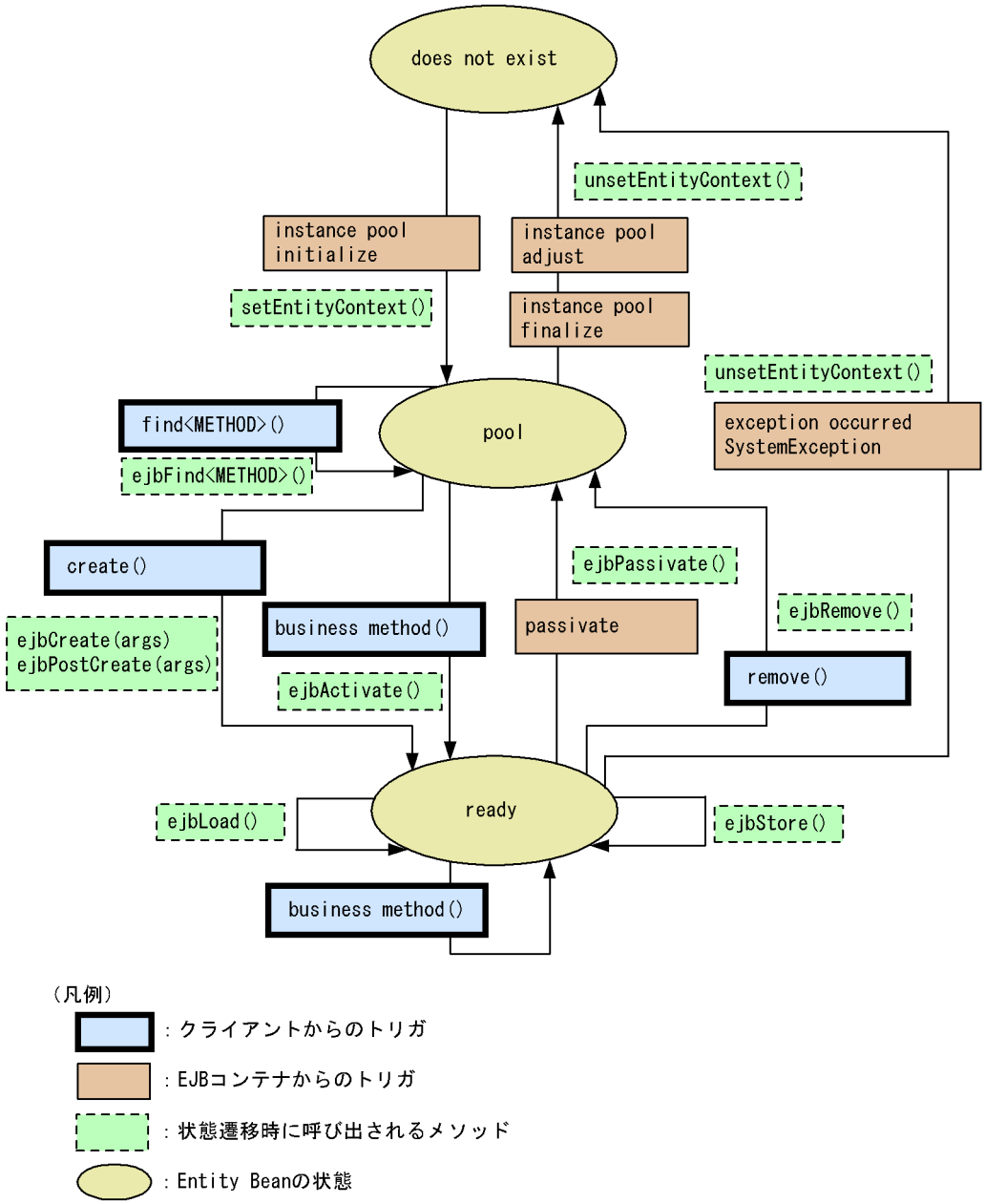
method-ready in TX：

Stateful Session Bean がアクティベイトされ，実行可能状態として method-ready プール内に存在する状態（トランザクションあり）

（2） Entity Bean のライフサイクル

Entity Bean のライフサイクルを次の図に示します。

図 2-3 Entity Bean のライフサイクル



does not exist：

Entity Bean が存在しない状態

pool：

Entity Bean がパッシベイトされ，passive プール内に存在する状態

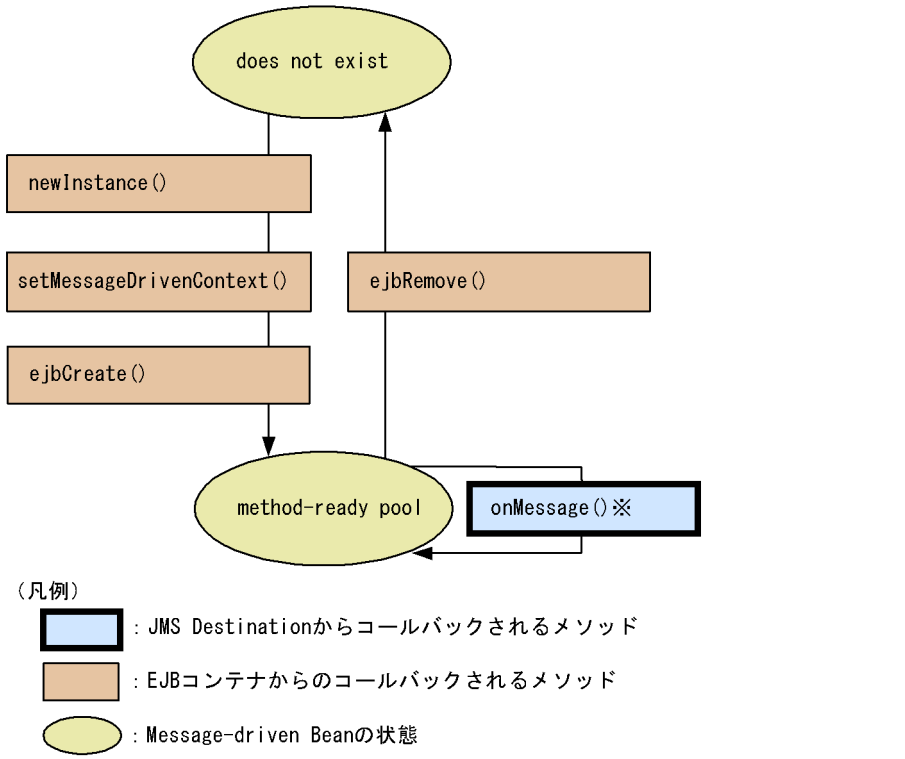
ready：

Entity Bean がアクティベイトされ，ready プール内に存在する状態

(3) Message-driven Bean のライフサイクル

Message-driven Bean のライフサイクルを次の図に示します。

図 2-4 Message-driven Bean のライフサイクル



注※ javax.jms.MessageListenerインタフェースを使用していない Message-driven Beanの場合は，任意のMessageListenerインタフェースのメソッドがコールバックされます。

does not exist：

Message-driven Bean が存在しない状態

method-ready pool：

Message-driven Bean が method-ready 状態で，ServerSession プール内に存在する状態

2.3 EJB 仕様準拠のチェック

EJB コンテナでは，J2EE アプリケーションの開始時に，各 Enterprise Bean に対して，EJB の仕様に準拠しているかどうかをチェックします。このチェックの結果，仕様に準拠しない Enterprise Bean があった場合は，Enterprise Bean を含む J2EE アプリケーションの開始に失敗します。このとき，エラーメッセージが出力されます。

EJB のチェック処理は，次のタイミングで動作します。

- J2EE アプリケーションを J2EE サーバに最初にインポートして開始する時。
- J2EE アプリケーションの構成を変更して（リデプロイ機能による変更を含む）開始する時。
- J2EE サーバをベーシックモードから 1.4 モードに変更したあと，最初に J2EE サーバを起動する時。
- アップグレードインストールをしたあと，最初に J2EE サーバを起動する時。

これらのタイミングで開始に成功すると，以降の J2EE アプリケーションの開始と停止では，EJB チェック処理は動作しません。

2.4 CMP フィールドとデータ型のマッピング

この節では，CMP フィールドとデータ型のマッピングについて説明します。

CMP フィールドでは，Java データ型ごとに，指定できる範囲とプライマリキーへの指定可否が決まっています。また，Java データ型とデータベースの SQL データ型の対応は，データベースの種類ごとに異なります。

この節の構成を次の表に示します。

表 2-9 この節の構成（CMP フィールドとデータ型のマッピング）

分類	タイトル	参照先
解説	CMP でサポートする Java データ型の範囲	2.4.1
	CMP フィールドとデータベースのマッピング	2.4.2
注意事項	CMP を使用する場合の注意事項	2.4.3

注 「実装」,「設定」および「運用」について，この機能固有の説明はありません。

2.4.1 CMP でサポートする Java データ型の範囲

EJB コンテナの Entity Bean の CMP でサポートする Java データ型の範囲と，プライマリキーへの指定の可否を次の表に示します。

表 2-10 CMP でサポートする Java データ型の範囲

Java データ型	値の範囲	プライマリキーへの指定
boolean	true，false	×
java.lang.Boolean		○
byte	-128 ～ 127	×
java.lang.Byte		○
char	'¥u0000' ～ '¥uffff'(0 ～ 65535)	×
java.lang.Character		○
short	-32768 ～ 32767	×
java.lang.Short		○
int	-2147483648 ～ 2147483647	×
java.lang.Integer		○
long	-9223372036854775808 ～ 9223372036854775807	×
java.lang.Long		○

Java データ型	値の範囲	プライマリキーへの指定
float ※	± 1.40239846e-45 ～± 3.40282347e+38	×
java.lang.Float ※		○
double ※	± 4.94065645841246544e-324 ～± 1.79769313486231570e+308	×
java.lang.Double ※		○
byte[]	1Byte ～ 2147483647Byte	×
java.lang.String	－	○
java.math.BigDecimal	－	×
java.sql.Date	－	×
java.sql.Time	00:00:00 ～ 23:59:59	×
java.sql.TimeStamp	－	×
Serializable な型	－	×

（凡例）
○：プライマリキーに指定できる。
×：プライマリキーに指定できない。
－：該当しない。

注※
浮動小数点を扱う場合，丸めが発生する可能性があります。

2.4.2 CMP フィールドとデータベースのマッピング

ここでは，CMP フィールドとデータベースのマッピングについて説明します。マッピングは，データベースの種類ごとに異なります。

（1） HiRDB の場合のマッピング

HiRDB の場合の CMP フィールドとデータベースのマッピングについて，次の表に示します。

なお，表中の「Java データ型」は CMP がサポートする Java のデータ型，「JDBC データ型」は Java のデータ型に対応する JDBC の java.sql.Types. のデータ型，「SQL データ型」は Java データ型とのマッピングで推奨する DB カラムの型となります。

表 2-11 CMP でのフィールドとデータベースのマッピング（HiRDB 使用時）

Java データ型	JDBC データ型	SQL データ型
boolean	SMALLINT	SMALLINT
java.lang.Boolean		

Java データ型	JDBC データ型	SQL データ型
byte	SMALLINT	SMALLINT
java.lang.Byte		
char ※1	CHAR	CHAR(4)
java.lang.Character ※1		
short	SMALLINT	SMALLINT
java.lang.Short		
int	INTEGER	INTEGER
java.lang.Integer		
long	DECIMAL	DECIMAL(22)
java.lang.Long		
float	REAL	REAL，SMALLFLT
java.lang.Float		
double	FLOAT	DOUBLE PRECISION
java.lang.Double		
byte[] ※2	LONGVARBINARY	BLOB
java.lang.String ※1	VARCHAR	VARCHAR(m) CHAR(n) MVARCHAR(m) MCHAR(n) NVARCHAR(x) NCHAR(y) ※3
java.math.BigDecimal	DECIMAL	DECIMAL(m,n) ※4
java.sql.Date	DATE	DATE ※5
java.sql.Time	TIME	TIME
java.sql.TimeStamp ※6	CHAR	CHAR(29)
Serializable な型※2	LONGVARBINARY	BLOB

注※1

固定長文字列の SQL 型を使用する場合の注意については，「2.4.3　CMP を使用する場合の注意事項」を参照してください。

注※2

[HiRDB BLOB 最大値] 2147483647 バイト

ただし，扱えるデータサイズの上限は JDBC ドライバの上限に依存します。

HiRDB Type4 JDBC Driver を使用する場合は，JDBC ドライバによる制限はありません。

注※3

m，n，x，y の範囲はそれぞれ次のとおりです。
m：1 ～ 32000，n：1 ～ 30000，x：1 ～ 16000，y：1 ～ 15000

注※ 4
m，n の範囲はそれぞれ次のとおりです。
m：1 ～ 29，n：1 ～ 29

注※ 5
DATE の範囲は，0001/01/01 ～ 9999/12/31 です。

注※ 6
「yyyy-mm-dd hh:mm:ss.ffffffff」（JDBC 日付エスケープ）形式の文字列として格納します。

（2） Oracle の場合のマッピング一覧

Oracle の場合の CMP フィールドとデータベースのマッピングについて，次の表に示します。

なお，表中の「Java データ型」は CMP がサポートする Java のデータ型，「JDBC データ型」は Java のデータ型に対応する JDBC の java.sql.Types. のデータ型，「SQL データ型」は Java データ型とのマッピングで推奨する DB カラムの型となります。

表 2-12　CMP でのフィールドとデータベースのマッピング（Oracle 使用時）

Java データ型	JDBC データ型	SQL データ型
boolean	NUMERIC	NUMBER(38)
java.lang.Boolean		
byte	NUMERIC	NUMBER(38)
java.lang.Byte		
char ※ 1	CHAR	CHAR(4)
java.lang.Character ※ 1		
short	NUMERIC	NUMBER(38)
java.lang.Short		
int	NUMERIC	NUMBER(38)
java.lang.Integer		
long	NUMERIC	NUMBER(22)
java.lang.Long		
float	NUMERIC	NUMBER
java.lang.Float		
double ※ 2	FLOAT	FLOAT(126)
java.lang.Double ※ 2		
byte[] ※ 3	LONGVARBINARY	LONG RAW

Java データ型	JDBC データ型	SQL データ型
java.lang.String ※1	VARCHAR	VARCHAR(m) CHAR(n) LONG ※4
java.math.BigDecimal	NUMERIC	NUMBER(m,n) ※5
java.sql.Date	DATE	DATE ※6, ※7
java.sql.Time	CHAR	CHAR(8) ※8
java.sql.TimeStamp	TimeStamp	DATE ※7, ※9
Serializable な型※3	LONGVARBINARY	LONG RAW

注　java.sql.Types.BLOB にマッピングされる BLOB，および java.spl.Types.CLOB にマッピングされる CLOB は扱えません。

注※ 1

固定長文字列の SQL 型を使用する場合の注意については，「2.4.3　CMP を使用する場合の注意事項」を参照してください。

注※ 2

[Oracle FLOAT(126) の範囲] 1E-125 ～ 9.9・・・9E125

ただし，丸めが発生するおそれがあります。

注※ 3

[Oracle LONG RAW 最大値] 2 ギガバイト。

ただし，扱えるデータサイズの上限は JDBC ドライバの上限に依存します。

注※ 4

m，n の範囲はそれぞれ次のとおりです。

m=1 ～ 4000，n=1 ～ 2000

また，""（長さゼロの空文字）を Oracle に格納すると NULL に変換されます。

注※ 5

m，n の範囲はそれぞれ次のとおりです。

m=1 ～ 38，n=-84 ～ 127

注※ 6

DATE の範囲は，-4712/01/01 ～ 9999/12/31 です。

注※ 7

紀元前データは Java-JDBC ドライバ間でマイナス値が正しく扱えないため，値は保証されません。

注※ 8

「hh:mm:ss」（JDBC 日付エスケープ）形式の文字列として格納します。

注※ 9

DATE の範囲は，-4712/01/01 00:00:00 ～ 9999/12/31 23:59:59 です。

2.4.3 CMP を使用する場合の注意事項

アプリケーションサーバの EJB コンテナで CMP を使用する場合の注意事項について説明します。

● 固定長文字列の SQL 型を CMP フィールドに用いる場合

固定長文字列の SQL 型（Oracle，HiRDB の CHAR 型）を用いる場合，データベース格納時にけた数に満たない文字数を空白で埋めるため，作成時のデータの後ろに空白が挿入される可能性があります。したがって，使用するときは十分に注意してください。

● 固定長文字列の SQL 型をプライマリキーに用いる場合

固定長文字列の SQL 型（Oracle，HiRDB の CHAR 型）をキーに用いる場合，データベース格納時にけた数に満たない文字数を空白で埋めるため，作成時のデータと値が異なり，対象 Entity Bean を取得できない可能性があります。したがって，使用するときは十分に注意してください。

● remove メソッドで例外が発生した場合の対応

remove メソッド実行中に例外が発生すると，データベース上にデータが残ります。手動で削除してください。

2.5 EJB コンテナの JNDI 名前空間へのリファレンス登録

この節では，EJB コンテナの JNDI 名前空間へのリファレンス登録について説明します。

EJB コンテナでは，java:comp/env 名前空間へのリファレンスの登録をサポートしています。

この節の構成を次の表に示します。

表 2-13 この節の構成（EJB コンテナの JNDI 名前空間へのリファレンス登録）

分類	タイトル	参照先
解説	java:comp/env 名前空間へのリファレンスの登録	2.5.1
実装	cosminexus.xml での定義	2.5.2
設定	実行環境での設定	2.5.3

注 「運用」および「注意事項」について，この機能固有の説明はありません。

なお，アプリケーションサーバで使用できるネーミング管理機能の詳細については，マニュアル「Cosminexus 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「2. ネーミング管理」を参照してください。

2.5.1 java:comp/env 名前空間へのリファレンスの登録

EJB コンテナでは，java:comp/env 名前空間へのリファレンスの登録をサポートしています。なお，該当するリファレンスは，() 内に示す名前空間に登録することを推奨しています。

- 環境エントリ（java:comp/env）
- EJB ホームオブジェクトリファレンス（java:comp/env/ejb）
- ビジネスインタフェースのリファレンス（java:comp/env/ejb）
- JMS（java:comp/env/jms）
- JDBC データソース（java:comp/env/jdbc）
- JavaMail セッション（java:comp/env/mail）
- uCosminexus TP1 Connector（java:comp/env/eis）
- JavaBeans リソース（java:comp/env/bean）

これによって，java:comp/env の JNDI 名前空間を使用して，間接的なルックアップができます。

2.5.2 cosminexus.xml での定義

リファレンスマッピングの定義は，cosminexus.xml の <ejb-jar> タグ内に指定します。

java:comp/env を使用する場合，参照元のアプリケーションでリファレンスマッピングを定義する必要があります。設定するタグは，設定対象になる Enterprise Bean の種類ごとに異なります。

cosminexus.xml でのリファレンスマッピングの定義について次の表に示します。

表 2-14 cosminexus.xml でのリファレンスマッピングの定義

項目	指定するタグ	設定内容
リソースへのリファレンスの解決	Session Bean の場合 　<session>-<resource-ref> タグ 　<session>-<resource-env-ref> タグ Entity Bean の場合 　<entity>-<resource-ref> タグ 　<entity>-<resource-env-ref> タグ Message-driven Bean の場合 　<message-driven>-<resource-ref> タグ 　<message-driven>-<resource-env-ref> タグ	• <resource-ref> タグに，リソースの情報を設定します。 • <resource-env-ref> タグに，リソース環境の情報を設定します。

指定するタグの詳細は，マニュアル「Cosminexus 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「2.2.2 EJB-JAR 属性の詳細」を参照してください。

2.5.3 実行環境での設定

リファレンスマッピングは，実行環境で設定することもできます。J2EE サーバにインポートした J2EE アプリケーションに設定します。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ実行してください。

実行環境での J2EE アプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。リファレンスマッピングの定義には，次の属性ファイルを使用します。

表 2-15 リファレンスマッピングの定義に使用する属性ファイル

設定対象	属性ファイル
Session Bean	Session Bean 属性ファイル
Entity Bean	Entity Bean 属性ファイル
Message-driven Bean	Message-driven Bean 属性ファイル

属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。なお，Enterprise Bean へのリファレンスを解決するか，リソースへのリファレンスを解決する

かによって，指定するタグは異なります。

cosminexus.xml での定義については，「2.5.2　cosminexus.xml での定義」を参照してください。

2.6 外部リソースとの接続

EJB コンテナで使用できるリソース，および J2EE リソースの詳細については，マニュアル「Cosminexus 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「3. リソース接続とトランザクション管理」を参照してください。

2.7 Enterprise Bean でのトランザクション管理

この節では，Enterprise Bean でのトランザクション管理について説明します。

Enterprise Bean のトランザクションには，Enterprise Bean で管理する方法と EJB コンテナで管理する方法があります。

この節の構成を次の表に示します。

表 2-16 この節の構成（Enterprise Bean でのトランザクション管理）

分類	タイトル	参照先
解説	Enterprise Bean でのトランザクション管理方法の種類	2.7.1
	BMT	2.7.2
	CMT	2.7.3
実装	cosminexus.xml での定義	2.7.4
設定	実行環境での設定	2.7.5

注 「運用」および「注意事項」について，この機能固有の説明はありません。

なお，アプリケーションサーバで使用できるトランザクション管理機能の詳細については，マニュアル「Cosminexus 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「3.4 トランザクション管理」を参照してください。

2.7.1 Enterprise Bean でのトランザクション管理方法の種類

EJB コンテナがサポートしている Enterprise Bean のトランザクション管理方法には，次の 2 種類があります。

- BMT（Bean-Managed Transaction）
- CMT（Container-Managed Transaction）

これらの機能は，J2EE サービスのトランザクション管理機能によって実現されます。

なお，トランザクション管理方法は，J2EE アプリケーションに含まれる Session Bean または Message-driven Bean の属性（プロパティ）として設定します。J2EE アプリケーションの設定については，「2.7.4 cosminexus.xml での定義」を参照してください。

トランザクション管理については，マニュアル「Cosminexus 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「3. リソース接続とトランザクション管理」を参照してください。

2.7.2 BMT

Enterprise Bean でトランザクション管理をするモデルです。BMT は，Session Bean，および Message-driven Bean が対象です（Entity Bean は対象外で，常に CMT になります)。

BMT では，Enterprise Bean のビジネスメソッドで javax.transaction.UserTransaction を利用し，次の操作をします。

1. トランザクションを開始
2. リソースマネジャを更新
3. トランザクションをコミットまたはロールバック

Stateless Session Bean では，1 ビジネスメソッド内で 1 トランザクションを決着（コミットまたはロールバック）させる必要があります。また，Message-driven Bean では，次のメソッド内で，トランザクションを決着（コミットまたはロールバック）させる必要があります。メソッドは EJB のバージョンごとに異なります。

- onMessage メソッド（EJB 2.0 の場合）
- 任意のメッセージリスナのメソッド（EJB 2.1 以降の場合）

一方，Stateful Session Bean では，複数のビジネスメソッドを一つのトランザクションスコープに含めることができます。このとき，EJB コンテナは，Bean インスタンスとトランザクションの関係を保持します。BMT での Stateful Session Bean を次の図に示します。

図 2-5　BMT での Stateful Session Bean

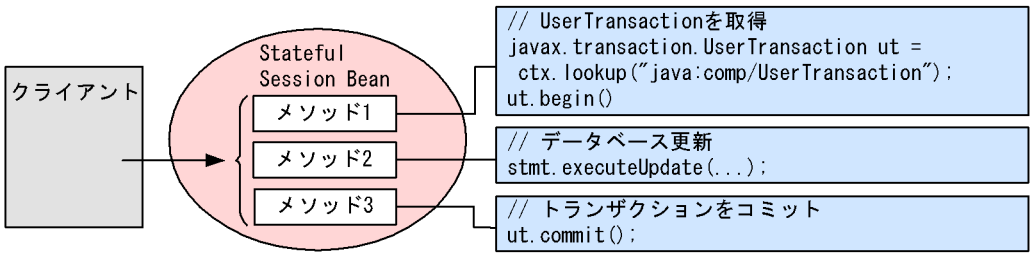


表 2-17　BMT のトランザクション制御

クライアント側トランザクション	Bean インスタンスのトランザクション	メソッドに結び付くトランザクション
なし	なし	なし
T1	なし	なし
なし	T2	T2
T1	T2	T2

（凡例）

なし：トランザクションを開始しない，またはトランザクションが関連づかない。
T1：クライアント側で開始するトランザクション。
T2：Bean 側で開始するトランザクション。

注意事項

トランザクションが決着していないときに，UserTransaction.begin メソッドを使って新しいトランザクションを開始した場合，EJB コンテナは javax.transaction.NotSupportedException を送出します。

2.7.3 CMT

EJB コンテナでトランザクション管理をするモデルです。CMT は，Session Bean，Entity Bean，Message-driven Bean が対象です。CMT では，Bean のメソッド単位でトランザクションの属性を指定します。なお，トランザクション属性は，J2EE アプリケーションに含まれる Session Bean，Entity Bean，または Message-driven Bean の属性（プロパティ）として設定します。J2EE アプリケーションの設定については，「2.7.4 cosminexus.xml での定義」を参照してください。

トランザクションがコミットできない場合，EJB コンテナは次の処理をします。

1. アプリケーションエラーのロギング
2. トランザクションのロールバック
3. Bean インスタンスの破棄
4. リモートコンポーネントインタフェースで呼び出すクライアントには java.rmi.RemoteException，ローカルコンポーネントインタフェースまたはビジネスインタフェースで呼び出すクライアントには javax.ejb.EJBException を送出。ただし，リモートビジネスインタフェースが java.rmi.Remote を継承している場合には，java.rmi.RemoteException を送出

（1） トランザクション属性の種類と振る舞い

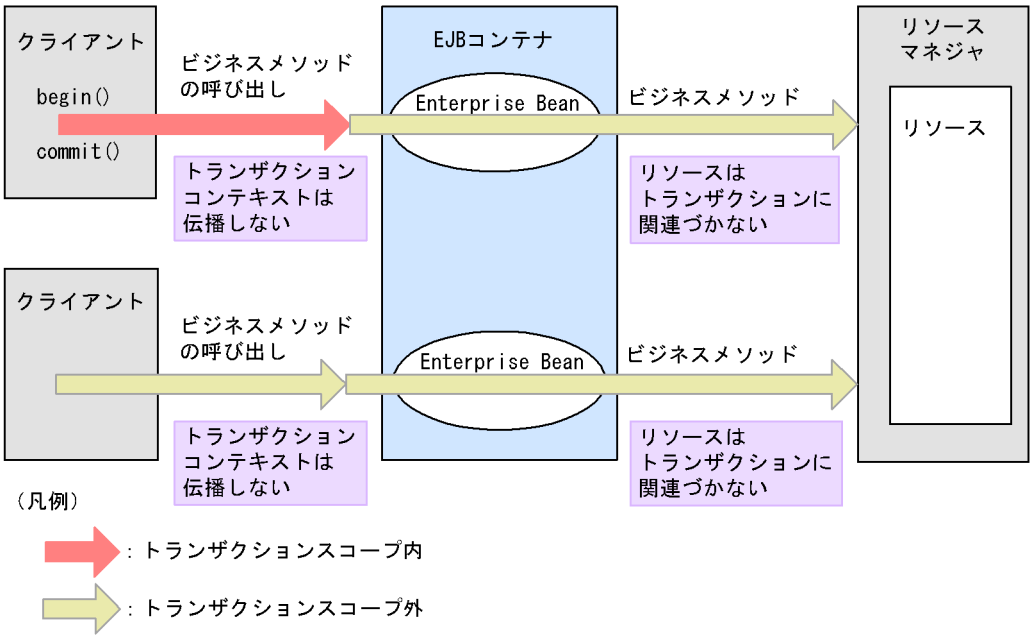
トランザクション属性の振る舞いについて，トランザクション属性の種類ごとに図に示して説明します。

● NotSupported 属性

クライアントがトランザクションスコープ内で Enterprise Bean のビジネスメソッドを呼び出したとき，トランザクションコンテキストは Enterprise Bean 側には伝播しません。また，クライアントがトランザクションスコープ外で Enterprise Bean のビジネスメソッドを呼び出したときも，トランザクションコンテキストは Enterprise Bean には伝播しません。

NotSupported 属性の振る舞いを次の図に示します。

図 2-6　NotSupported 属性

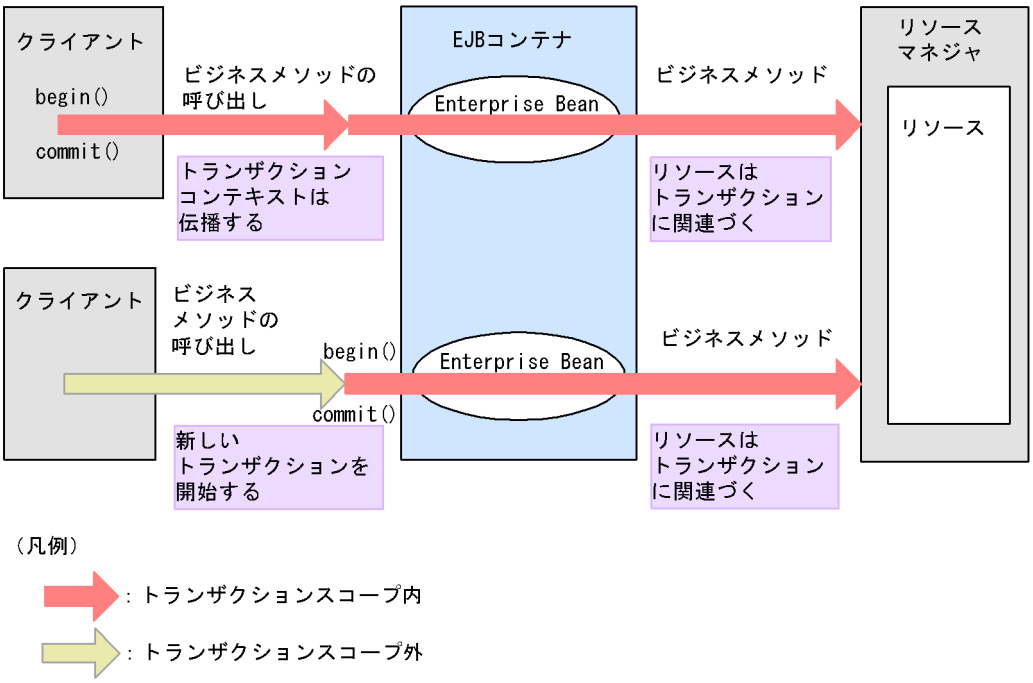


● Required 属性

クライアントがトランザクションスコープ内で Enterprise Bean のビジネスメソッドを呼び出したとき，トランザクションコンテキストが Enterprise Bean 側に伝播し，Enterprise Bean のビジネスメソッドは呼び出し側のトランザクションスコープ内に入ります。また，クライアントがトランザクションスコープ外で Enterprise Bean のビジネスメソッドを呼び出したとき，Enterprise Bean 側で新しいトランザクションが開始されます。

Required 属性の振る舞いを次の図に示します。

図 2-7 Required 属性

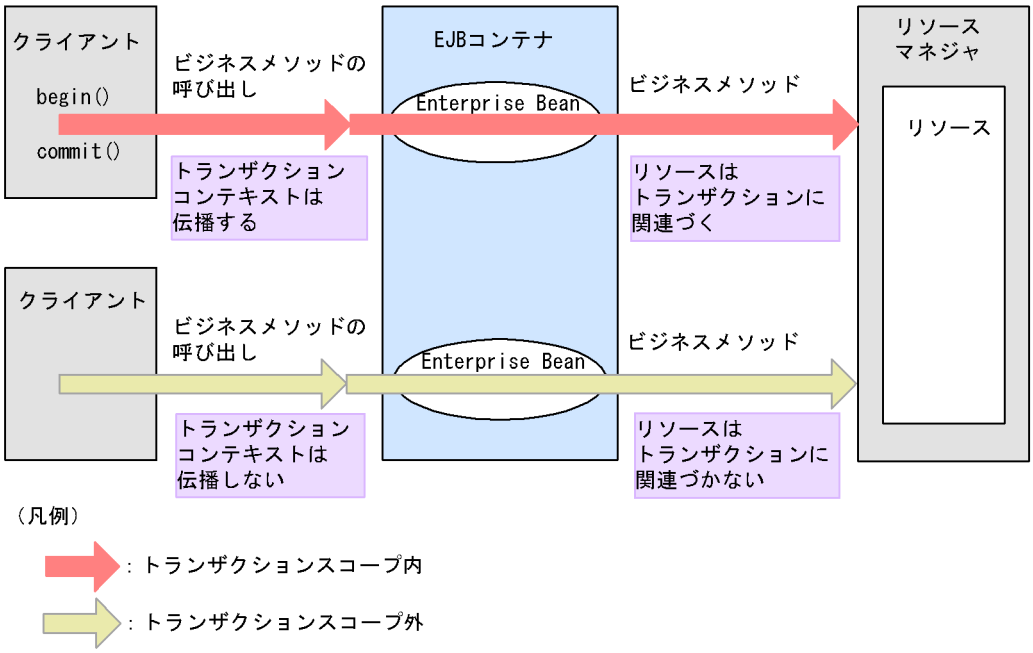


● Supports 属性

クライアントがトランザクションスコープ内で Enterprise Bean のビジネスメソッドを呼び出したとき，トランザクションコンテキストが Enterprise Bean 側に伝播し，Enterprise Bean のビジネスメソッドは呼び出し側のトランザクションスコープ内に入ります。また，クライアントがトランザクションスコープ外で Enterprise Bean のビジネスメソッドを呼び出したとき，トランザクションコンテキストは Enterprise Bean に伝播しません。

Supports 属性の振る舞いを次の図に示します。

図 2-8　Supports 属性

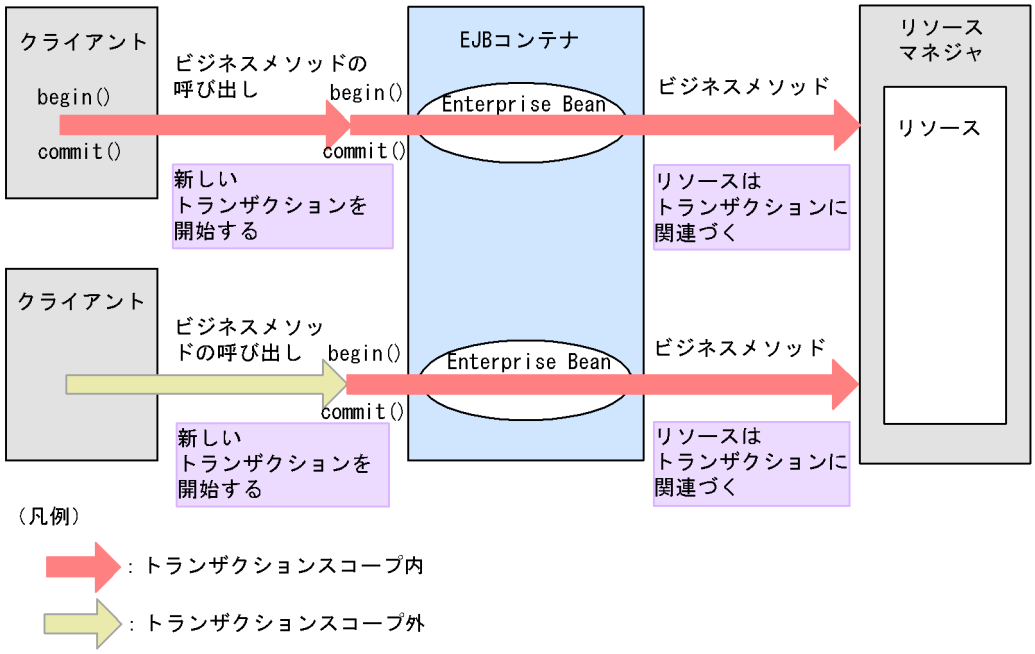


● RequiresNew 属性

クライアントがトランザクションスコープ内で Enterprise Bean のビジネスメソッドを呼び出したとき，EJB コンテナは新しいトランザクションを開始します。また，クライアントがトランザクションスコープ外で Enterprise Bean のビジネスメソッドを呼び出したときも，同様に新しいトランザクションが EJB コンテナによって開始されます。

RequiresNew 属性の振る舞いを次の図に示します。

図 2-9 RequiresNew 属性



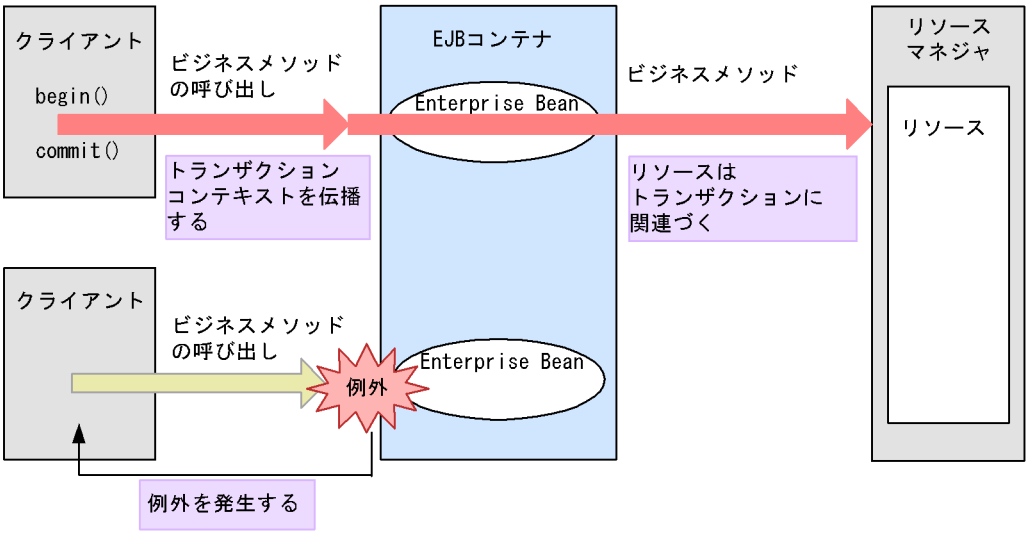
● Mandatory 属性

クライアントがトランザクションスコープ内で Enterprise Bean のビジネスメソッドを呼び出したとき，トランザクションコンテキストが Enterprise Bean 側に伝播し，Enterprise Bean のビジネスメソッドは呼び出し側のトランザクションスコープ内に入ります。また，クライアントがトランザクションスコープ外で Enterprise Bean のビジネスメソッドを呼び出したとき，EJB コンテナはクライアントに次の例外を送出します。

- ビジネスインタフェースを使用している場合，
  javax.ejb.EJBTransactionRequiredException を送出します。ただし，
  java.rmi.Remote を継承しているリモートビジネスインタフェースの場合は，
  javax.transaction.TransactionRequiredException を送出します。
- リモートコンポーネントインタフェースを使用している場合，
  javax.transaction.TransactionRequiredException を送出します。
- ローカルコンポーネントインタフェースを使用している場合，
  javax.ejb.TransactionRequiredLocalException を送出します。

Mandatory 属性の振る舞いを次の図に示します。

図 2-10 Mandatory 属性



（凡例）

：トランザクションスコープ内

：トランザクションスコープ外

● Never 属性

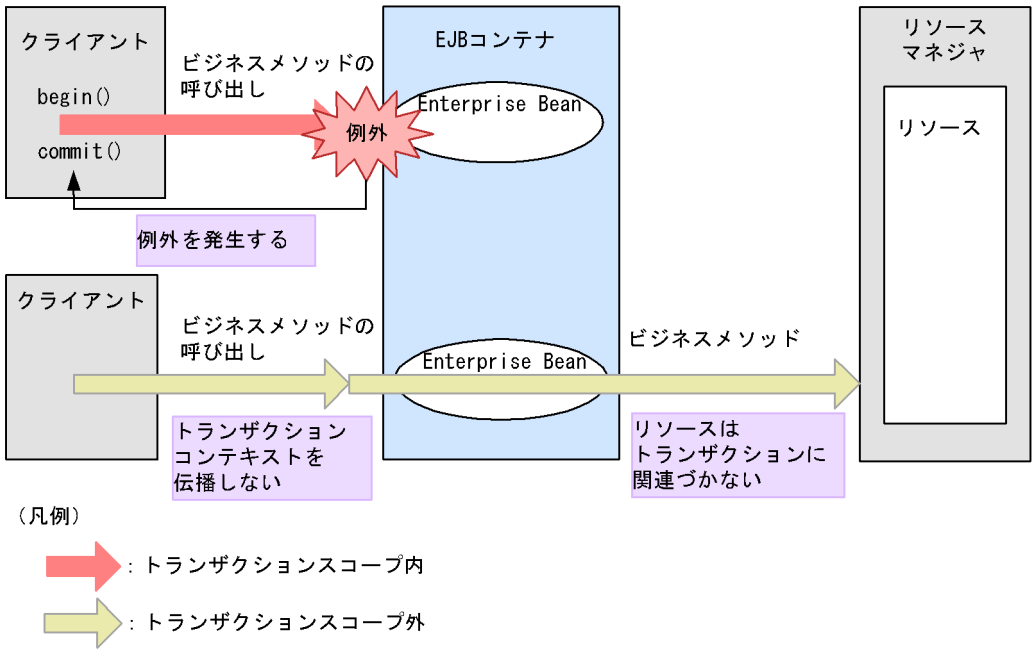
クライアントがトランザクションスコープ内で Enterprise Bean のビジネスメソッドを呼び出したとき，EJB コンテナはクライアントに次の例外を送出します。

- ビジネスインタフェースを使用している場合，javax.ejb.EJBException を送出します。ただし，java.rmi.Remote を継承しているリモートビジネスインタフェースの場合は，java.rmi.RemoteException を送出します。
- リモートコンポーネントインタフェースを使用している場合，java.rmi.RemoteException を送出します。
- ローカルコンポーネントインタフェースを使用している場合，javax.ejb.EJBException を送出します。

また，クライアントがトランザクションスコープ外で Enterprise Bean のビジネスメソッドを呼び出したとき，トランザクションコンテキストは Enterprise Bean に伝播しません。

Never 属性の振る舞いを次の図に示します。

図 2-11　Never 属性



（2）Enterprise Bean の種別ごとに指定できるトランザクション属性

Enterprise Bean の種別ごとに指定できるトランザクション属性およびデフォルト値を次の表に示します。EJB 仕様では Bean 種別ごとに指定できるトランザクション属性が規定されています。CMP 2.0 については，EJB 仕様でオプションとされている，ほかのトランザクション属性は指定できません。デフォルト値については EJB 仕様では規定されていません。アプリケーションサーバの場合，EJB の DD で CMT の指定があり，トランザクション属性の指定がない場合のデフォルト設定は，次の表のようになります。

表 2-18　Enterprise Bean の種別ごとに指定できるトランザクション属性およびデフォルト値

種別	指定できるトランザクション属性	デフォルト値
Stateless Session Bean Stateful Session Bean （SessionSynchronization 以外） Entity Bean（BMP，CMP 1.1）	• Supports • NotSupported • Required • RequiresNew • Mandatory • Never	Supports

種別	指定できるトランザクション属性	デフォルト値
Stateful Session Bean（SessionSynchronization）	• Supports • NotSupported • Required • RequiresNew • Mandatory • Never	Required
Entity Bean（CMP 2.0）	• Required • RequiresNew • Mandatory	Required
Message-driven Bean（Connector 1.0 に準拠したリソースアダプタを使用する場合）	• Required • NotSupported	Required
Message-driven Bean（Connector 1.5 に準拠したリソースアダプタを使用する場合）	• Required ※ • NotSupported	Required ※
アノテーションを使用した，DD を持たない Session Bean	• Supports • NotSupported • Required • RequiresNew • Mandatory • Never	Required

注

DD を持たない Enterprise Bean の場合，デフォルト値（Required）が規定されています。

注※

CJMSP リソースアダプタまたは FTP インバウンドアダプタの場合は指定できません。

（3） Stateful Session Bean（SessionSynchronization）のトランザクション属性

SessionSynchronization は，トランザクションの開始および停止に同期してメッセージを通知するためのインタフェースです。EJB 仕様では，SessionSynchronization を実装した Stateful Session Bean のトランザクション属性に指定できるのは，Required，RequiresNew または Mandatory と規定されています。これに対して，アプリケーションサーバでは，Supports，NotSupported および Never も指定できます。

ここでは，指定したトランザクション属性とメソッド呼び出しの有無の対応，およびコールバックメソッドを呼び出すタイミングについて説明します。

（a） トランザクション属性とメソッド呼び出しの有無

SessionSynchronization を実装した Stateful Session Bean のトランザクション属性と，ビジネスメソッドまたはコールバックメソッドの呼び出し有無の対応について，次の表に示します。なお，コールバックメソッドとは，afterBegin メソッド，beforeCompletion メソッドおよび afterCompletion メソッドのことです。

表 2-19 ビジネスメソッドまたはコールバックメソッドでのメソッド呼び出し有無

トランザクション属性	クライアントのトランザクション有無	ビジネスメソッドの呼び出し	SessionSynchronization のコールバックメソッド呼び出し
Supports	あり	○※1	○※1
	なし	○※1	×
NotSupported	あり	○※1	×
	なし	○※1	×
Required	あり	○	○
	なし	○	○
RequiresNew	あり	○	○
	なし	○	○
Mandatory	あり	○	○
	なし	×※2	×
Never	あり	×※2	×
	なし	○※1	×

（凡例）

○：メソッドを呼び出す。

×：メソッドを呼び出さない。

注※ 1

アプリケーションサーバ独自の動作です。

注※ 2

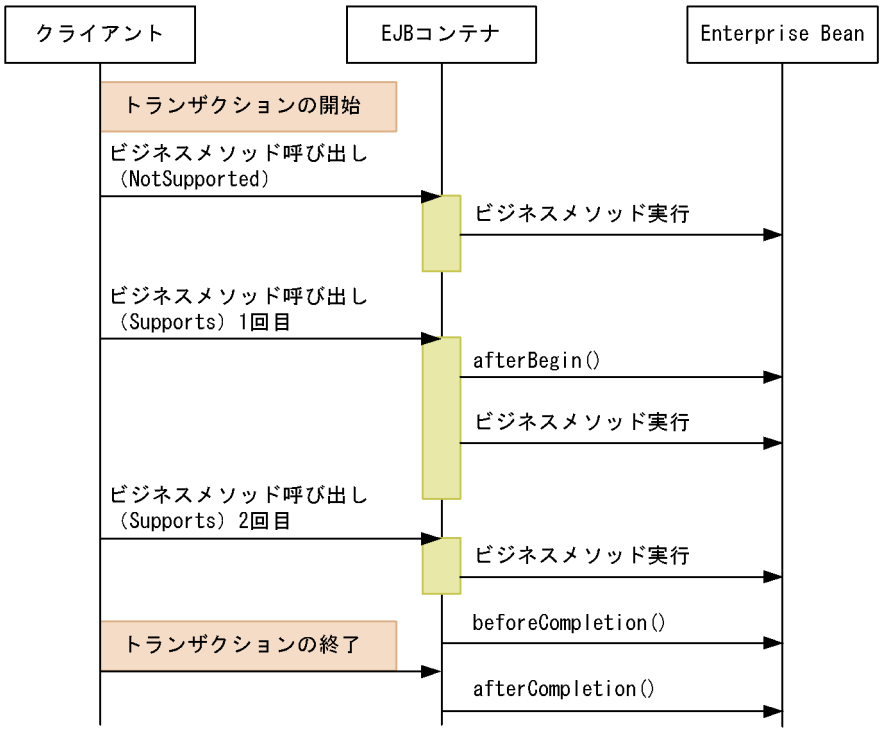
EJB 仕様で規定された例外がスローされます。

(b) コールバックメソッドを呼び出すタイミング

同一の Enterprise Bean のビジネスメソッドを複数回呼び出す場合，トランザクションに最初に Entprise Bean が参加するタイミングで，コールバックメソッドである afterBegin メソッドが呼び出されます。

afterBegin メソッドを呼び出すタイミングの例として，トランザクションに参加しないビジネスメソッド実行後にトランザクションに参加するビジネスメソッドを実行した場合の動作を次の図に示します。

図 2-12 コールバックメソッドを呼び出すタイミングの例



・最初のNotSupportedでのビジネスメソッド呼び出しでは，Enterprise Beanは
トランザクションに参加していないため，afterBeginメソッドは呼び出されません。

・1回目のSupportsでのビジネスメソッド呼び出しで，初めてEnterprise Beanが
トランザクションに参加します。このため，このタイミングでafterBeginメソッドが
呼び出されます。

・2回目のSupportsでのビジネスメソッド呼び出しでは，すでにafterBeginメソッドは
実行されているため，afterBeginメソッドは呼び出されません。

2.7.4 cosminexus.xml での定義

Enterprise Bean のトランザクション管理方法の定義は，cosminexus.xml の <ejb-jar> タグ内に指定します。設定するタグは，設定対象になる Enterprise Bean の種類ごとに異なります。

cosminexus.xml での Enterprise Bean のトランザクション管理方法の定義について次の表に示します。

表 2-20 cosminexus.xml での Enterprise Bean のトランザクション管理方法の定義

項目	指定するタグ	設定内容
Enterprise Bean でのトランザクション管理方法（BMT または CMT）の選択※	Session Bean の場合 <session>-<transaction-type> タグ Message-driven Bean の場合 <message-driven>-<transaction-type> タグ	Bean（BMT）または Container（CMT）のどちらを選択するかを指定します。
メソッドに割り当てるトランザクション属性（CMT の場合）	Session Bean の場合または Entity Bean の場合 <assembly-descriptor>-<container-transaction>-<trans-attribute> Message-driven Bean の場合 <message-driven>-<container-transaction>-<trans-attribute>	メソッドに割り当てるトランザクション属性を指定します。

注※

BMT を選択した場合，トランザクションの制御は，API（javax.transaction.UserTransaction クラスのメソッドなど）で実行する必要があります。

2.7.5 実行環境での設定

Enterprise Bean のトランザクション管理方法は，実行環境で設定することもできます。J2EE サーバにインポートした J2EE アプリケーションに設定します。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ実行してください。

実行環境での J2EE アプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。リファレンスマッピングの定義には，次の属性ファイルを使用します。

表 2-21 Enterprise Bean のトランザクションの管理方法の定義に使用する属性ファイル

設定対象	属性ファイル
Session Bean	Session Bean 属性ファイル
Entity Bean	Entity Bean 属性ファイル
Message-driven Bean	Message-driven Bean 属性ファイル

属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。cosminexus.xml での定義については，「2.7.4 cosminexus.xml での定義」を参照してください。

2.8 Entity Bean のキャッシュモデル

この節では，Entity Bean のキャッシュモデルについて説明します。

Entity Bean では，3 種類のキャッシュモデルをサポートしています。

この節の構成を次の表に示します。

表 2-22 この節の構成（Entity Bean のキャッシュモデル）

分類	タイトル	参照先
解説	Entity Bean のキャッシュモデルの種類	2.8.1
実装	cosminexus.xml での定義	2.8.2
設定	実行環境での設定	2.8.3

注 「運用」および「注意事項」について，この機能固有の説明はありません。

2.8.1 Entity Bean のキャッシュモデルの種類

Entity Bean では，次に示す 3 種類の CMP フィールドのキャッシュ方法，および Entity Bean の状態遷移をサポートしています。

- Full caching（commit option A）
- Caching（commit option B）
- No caching（commit option C）

なお，コミットオプションは，J2EE アプリケーションに含まれる Entity Bean の属性（プロパティ）として設定します。J2EE アプリケーションの設定については，「2.8.2 cosminexus.xml での定義」を参照してください。

（1） Full caching（commit option A）

Full caching は，参照系の Entity Bean 用のキャッシュモデルです。トランザクション開始時にデータベースから Entity Bean インスタンスにデータが読み込まれません。このため，Entity Bean が前回のトランザクションコミット時と同じ状態のままでトランザクションが開始されます。

例えば，前回のトランザクションコミット時からトランザクション開始時の間にほかの J2EE サーバが Entity Bean を更新した場合，Entity Bean の状態の一貫性が保たれません。

（2） Caching（commit option B）

Caching は，更新系の Entity Bean のキャッシュモデルです。トランザクション開始時

にデータベースから Entity Bean インスタンスにデータが読み込まれます。このため，Entity Bean がデータベースの最新状態と同じ状態でトランザクションが開始されます。

（3） No caching（commit option C）

No caching は，更新系の Entity Bean のキャッシュモデルです。トランザクションコミット時に Entity Bean が非活性化されます。トランザクション開始時には，一度活性化され，データベースから Entity Bean インスタンスにデータが読み込まれます。このため，Entity Bean がデータベースの最新状態と同じ状態でトランザクションが開始されます。このため，多数の Entity Bean を利用する場合に適用するキャッシュモデルです。

2.8.2 cosminexus.xml での定義

Entity Bean のキャッシュモデルのコミットオプションの定義は，cosminexus.xml の <ejb-jar> タグ内に指定します。

指定するタグ

<entity>-<caching-model> タグ

設定内容

CMP フィールドのキャッシュ方法を指定します。

2.8.3 実行環境での設定

Entity Bean のキャッシュモデルのコミットオプションの定義は，実行環境で設定することもできます。J2EE サーバにインポートした J2EE アプリケーションに設定します。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ実行してください。

実行環境での J2EE アプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。Entity Bean のキャッシュモデルのコミットオプションの定義には，Entity Bean 属性ファイルを使用します。

Entity Bean 属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。cosminexus.xml での定義については，「2.8.2　cosminexus.xml での定義」を参照してください。

2.9 Enterprise Bean のプールの管理

この節では，Enterprise Bean のプールの管理について説明します。

EJB コンテナでは，Enterprise Bean の種類ごとに，プールを作成して管理します。

この節の構成を次の表に示します。

表 2-23 この節の構成（Enterprise Bean のプールの管理）

分類	タイトル	参照先
解説	Stateless Session Bean のプーリング	2.9.1
	Entity Bean のプーリング	2.9.2
	Message-driven Bean のプーリング	2.9.3
実装	cosminexus.xml での定義	2.9.4
設定	実行環境での設定	2.9.5

注 「運用」および「注意事項」について，この機能固有の説明はありません。

2.9.1 Stateless Session Bean のプーリング

Stateless Session Bean のプーリングは，クライアント側からのアクセス量に応じて Stateless Session Bean をプーリングする機能です。EJB コンテナでは Stateless Session Bean ごとにプールを作成して管理します。最大値※，最小値を指定することで，プーリングの動作をカスタマイズできます。

J2EE アプリケーション開始時には，最小値分の Stateless Session Bean が生成され，プーリングされます。プーリングされた Stateless Session Bean は method-ready 状態で，クライアントからアクセスされるとすぐに実行されます。プーリングされる Stateless Session Bean の数は，クライアントからのアクセス量に応じて，最大値と最小値の間になります。

この Stateless Session Bean に対するクライアント要求数が最大数を超えると，インスタンスが使用できるようになるまで実行が待たされます。

注※

Stateless Session Bean でのプーリングの最大値は，クライアントが同時に確立できる最大セッション数となります。

2.9.2 Entity Bean のプーリング

Entity Bean のプーリングは，クライアント側からのアクセス量に応じて Entity Bean をプーリングする機能です。EJB コンテナでは，Entity Bean ごとにプールを作成して

管理します。最大値，最小値を指定することで，プーリングの動作をカスタマイズできます。

J2EE アプリケーション開始時には，最小値分の Entity Bean が生成され，プーリングされます。プーリングされる Entity Bean の数は，クライアントからのアクセス量に応じて，最大値と最小値の間になります。

さらに，プーリングされた Entity Bean の状態には，ready 状態と pool 状態の二つがあります。

ready 状態の Entity Bean
: データがデータベース上からインスタンス中に読み込まれた状態のもので，Entity Bean としてのアイデンティティを持っています。ready 状態のものは，クライアントからアクセスされた時点ですでに実行可能状態になっています。

pool 状態の Entity Bean
: データがデータベース上からインスタンス中に読み込まれていない状態のもので，Entity Bean としてのアイデンティティを持っていません。pool 状態のものは，一度活性化され，ready 状態になってから実行可能状態になります。

ready 状態の Entity Bean が多くなると，幾つかが非活性化され，pool 状態になります。ただし，このとき，ready 状態の Entity Bean の中でトランザクション中のものについては，非活性化の対象になりません。

2.9.3 Message-driven Bean のプーリング

Message-driven Bean のプーリングは，メッセージの数に応じて Message-driven Bean をプーリングする機能です。最大値を指定することで，プーリングの動作をカスタマイズできます。

JMS からの要求に従って，プール内のインスタンスの最大数に指定した値分のインスタンスが，デプロイ時に生成されます。

インスタンスは，JMS からの要求に従って破棄されます。

2.9.4 cosminexus.xml での定義

Enterprise Bean のプール管理の定義は，cosminexus.xml の <ejb-jar> タグ内に指定します。設定するタグは，設定対象になる Enterprise Bean の種類ごとに異なります。

cosminexus.xml での Enterprise Bean のプール管理の定義について次の表に示します。

表 2-24 cosminexus.xml での Enterprise Bean のプール管理の定義

項目	指定するタグ	設定内容
Stateless Session Bean のプーリング	<session>-<stateless>-<pooled-instance> タグ	プールするインスタンス数の最大値と最小値を指定します。
Entity Bean のプーリング	<entity>-<pooled-instance> タグ	
Message-driven Bean のプーリング	<message>-<pooled-instance> タグ	

2.9.5 実行環境での設定

Enterprise Bean のプール管理の定義は，実行環境で設定することもできます。J2EE サーバにインポートした J2EE アプリケーションに設定します。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ実行してください。

実行環境での J2EE アプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。リファレンスマッピングの定義には，次の属性ファイルを使用します。

表 2-25 Enterprise Bean のトランザクションの管理方法の定義に使用する属性ファイル

設定対象	属性ファイル
Session Bean	Session Bean 属性ファイル
Entity Bean	Entity Bean 属性ファイル
Message-driven Bean	Message-driven Bean 属性ファイル

属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。cosminexus.xml での定義については，「2.9.4 cosminexus.xml での定義」を参照してください。

2.10 Enterprise Bean へのアクセス制御

この節では，Enterprise Bean へのアクセス制御について説明します。

EJB コンテナでは，J2EE サービスのセキュリティ管理機能によって，クライアントから Enterprise Bean へのアクセスを制御できます。

この節の構成を次の表に示します。

表 2-26 この節の構成（Enterprise Bean へのアクセス制御）

分類	タイトル	参照先
解説	Enterprise Bean へのアクセス制御の抑止	2.10.1
設定	実行環境での設定	2.10.2

注 「実装」,「運用」および「注意事項」について，この機能固有の説明はありません。

なお，Enterprise Bean へのアクセス制御以外のセキュリティ管理の機能については，マニュアル「Cosminexus 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「9. セキュリティ管理」を参照してください。

2.10.1 Enterprise Bean へのアクセス制御の抑止

EJB コンテナでは，J2EE サービスのセキュリティ管理機能によって，Enterprise Bean へのアクセスを制御できます。また，J2EE サーバのデフォルトの動作で，アプリケーションがアクセス制御の機能を利用していない場合でも，アクセス制御のための基本処理が動作します。

これに対して，Enterprise Bean へのアクセス制御を抑止するオプション（Enterprise Bean へのアクセス制御の抑止オプション）を使用すると，ビジネスメソッド呼び出し時に呼び出し元の実行権限のチェックを抑止できます。チェックを抑止すると，EJB コンテナではアクセス制御のための処理がまったく実施されないため，Enterprise Bean のビジネスメソッドの呼び出し処理が軽くなります。このため，アクセス制御の機能を利用しない場合は，抑止オプションを使用することをお勧めします。

ただし，アクセス制御の抑止オプションを利用した J2EE サーバから，異なる J2EE サーバ上のアクセス制御機能を利用した Enterprise Bean を呼び出す場合は，エラーが発生するので注意してください。

Enterprise Bean へのアクセス制御の抑止の設定は，J2EE サーバのプロパティをカスタマイズして設定します。

2.10.2 実行環境での設定

Enterprise Bean へのアクセス制御を使用する場合，J2EE サーバの設定が必要です。

簡易構築定義ファイルでの Enterprise Bean へのアクセス制御の抑止の定義は，論理 J2EE サーバ（j2ee-server）の <configuration> タグ内に指定します。簡易構築定義ファイルの J2EE サーバの JavaVM のシステムプロパティに設定します。

パラメタ名と設定内容は次のとおりです。

指定するパラメタ名
　ejbserver.container.security.disabled

設定内容
　Enterprise Bean へのアクセス制御機能を抑止するかどうかを指定します。

2.11 EJB コンテナでのタイムアウトの設定

この節では，EJB コンテナでのタイムアウトの設定について説明します。

EJB コンテナでは，次に示すタイムアウトを設定できます。

- Stateful Session Bean のタイムアウト
- Entity Bean の EJB オブジェクトのタイムアウト
- RMI-IIOP 通信のタイムアウト
- インスタンス取得待ちのタイムアウト

この節の構成を次の表に示します。

表 2-27 この節の構成（EJB コンテナでのタイムアウトの設定）

分類	タイトル	参照先
解説	タイムアウトの種類	2.11.1
	Stateful Session Bean のタイムアウト	2.11.2
	Entity Bean の EJB オブジェクトのタイムアウト	2.11.3
	インスタンス取得待ちのタイムアウト	2.11.4
	RMI-IIOP 通信のタイムアウト	2.11.5
実装	cosminexus.xml での定義	2.11.6
	RMI-IIOP 通信のタイムアウトの実装	2.11.7
設定	実行環境での設定	2.11.8
注意事項	通信のタイムアウト設定時の注意事項	2.11.9

注 「運用」について，この機能固有の説明はありません。

2.11.1 タイムアウトの種類

EJB コンテナで設定できるタイムアウトの種類，タイムアウト機能の概要，および参照先を次の表に示します。

表 2-28 タイムアウトの種類

タイムアウトの種類	タイムアウトの概要
Stateful Session Bean のタイムアウト	Stateful Session Bean が最後にアクセスされた時点からのタイムアウト時間を設定します。指定された時間が経過したものを削除します。
Entity Bean の EJB オブジェクトのタイムアウト	非活性化された Entity Bean に結び付く EJB オブジェクトにタイムアウト時間を設定します。指定された時間が経過したものを削除します。

タイムアウトの種類	タイムアウトの概要
RMI-IIOP 通信のタイムアウト	EJB クライアント－ CORBA ネーミングサービス間，および EJB クライアント－ Enterprise Bean 間の通信にタイムアウト時間を設定します。
インスタンス取得待ちのタイムアウト	Stateless Session Bean，Entity Bean に，リクエスト受信時のインスタンス取得の待ち時間にタイムアウト時間を設定します。

2.11.2 Stateful Session Bean のタイムアウト

Stateful Session Bean が最後にアクセスされた時点からの経過時間を監視し，指定された時間が経過してもクライアントからアクセスされない場合，タイマによって Stateful Session Bean を削除する機能です。EJB コンテナでは，タイムアウト時間を指定できます。ただし，トランザクション中の Stateful Session Bean については，削除の対象になりません。

なお，タイムアウトによって削除された Stateful Session Bean を呼び出すと，インタフェースの種類によって，次の例外が送出されます。

- リモートコンポーネントインタフェースの場合
  java.rmi.NoSuchObjectException が送出されます。
- ローカルコンポーネントインタフェースの場合
  java.ejb.NoSuchObjectLocalException が送出されます。
- ビジネスインタフェースの場合
  javax.ejb.NoSuchEJBException が送出されます。ただし，ビジネスインタフェースが java.rmi.Remote を継承している場合は，java.rmi.NoSuchObjectException が送出されます。

Stateful Session Bean のタイムアウトの設定は，J2EE アプリケーションに含まれる Session Bean の属性（プロパティ）として設定します。設定方法については，「2.11.6 cosminexus.xml での定義」を参照してください。

2.11.3 Entity Bean の EJB オブジェクトのタイムアウト

非活性化された Entity Bean に結び付く EJB オブジェクトのうち，指定された時間が経過したものを削除する機能です。EJB コンテナでは，タイムアウト時間を指定できます。なお，タイムアウトによって EJB オブジェクトを削除された Entity Bean を呼び出した場合，例外（java.rmi.NoSuchObjectException）が発生します。

ローカルインタフェースの EJB ローカルオブジェクトも同様にタイムアウトによる削除対象になります。Entity Bean が EJB オブジェクト，EJB ローカルオブジェクトの両方を持つ場合，非活性化された Entity Bean に対するアクセスが，指定された時間内にどちらのインタフェースからもないときは，EJB オブジェクト，EJB ローカルオブジェク

トを削除します。削除された EJB ローカルオブジェクトを呼び出した場合，例外（javax.ejb.NoSuchObjectLocalException）が発生します。

Entity Bean の EJB オブジェクトのタイムアウトの設定は，J2EE アプリケーションに含まれる Entity Bean の属性（プロパティ）として設定します。設定方法については，「2.11.6　cosminexus.xml での定義」を参照してください。

2.11.4　インスタンス取得待ちのタイムアウト

EJB コンテナは，リクエストを受け付けるとインスタンスの割り当てをします。割り当て時に，インスタンスプール（Stateless Session Bean の method-ready プールおよび Entity Bean の pool プール）に最大値が指定されている場合で，すべてのインスタンスでほかのリクエストを処理しているときには，インスタンスの取得待ちが発生します。この待ち時間に，タイムアウトを設定できます。

タイムアウトを設定していると，設定した時間内にインスタンスが取得できない場合は，クライアントに例外が返ります。

インスタンス取得待ちのタイムアウトの設定は，J2EE アプリケーションに含まれる Session Bean または Entity Bean の属性（プロパティ）として設定します。設定方法については，「2.11.6　cosminexus.xml での定義」を参照してください。

2.11.5　RMI-IIOP 通信のタイムアウト

EJB クライアント－ CORBA ネーミングサービス間，EJB クライアント－ Enterprise Bean 間の通信に対して，タイムアウトを設定できます。また，EJB クライアントと J2EE サーバの間に CTM を配置している場合，通信タイムアウトは，EJB クライアント－ CTM 間，および CTM － J2EE サーバ間に設定できます。

RMI-IIOP の通信のタイムアウトには，EJB コンテナが RMI-IIOP の通信基盤として利用する Cosminexus TPBroker のリクエストタイムアウト機能を利用します。

タイムアウトの設定は，設定する範囲によって，定義ファイルのプロパティ，またはアプリケーションサーバが提供する API のどちらかで行います。

(1) タイムアウトを設定できる RMI-IIOP 通信

タイムアウトを設定できる RMI-IIOP 通信を，CTM 連携ありの場合となしの場合について図に示し，タイムアウトの設定範囲について説明します。

図 2-13 タイムアウトを設定できる通信（CTM 連携ありの場合）

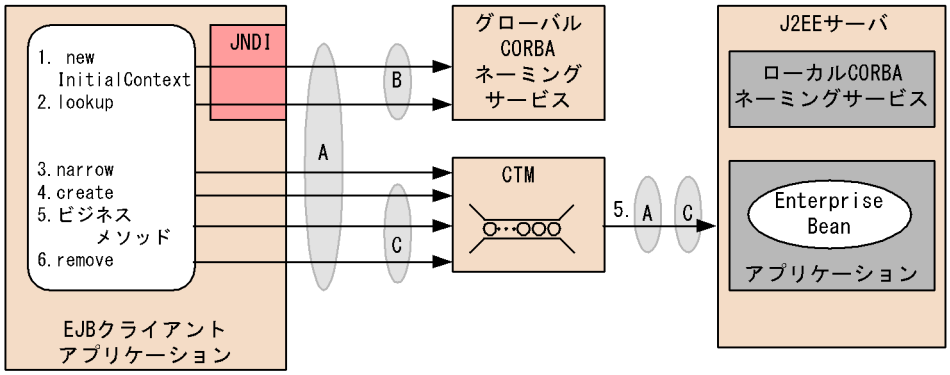
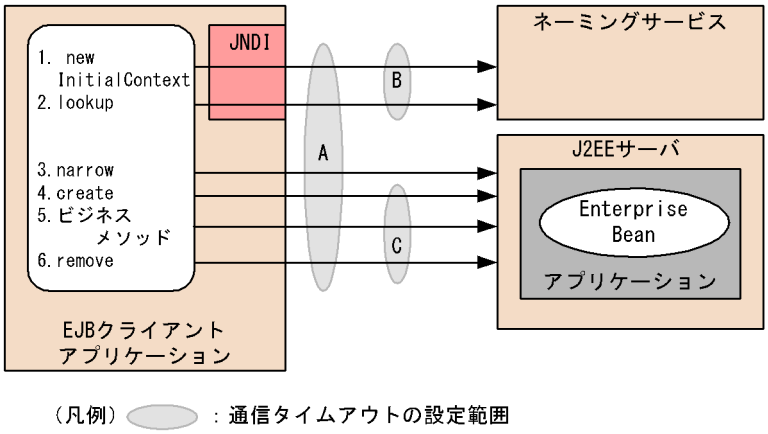


図 2-14 タイムアウトを設定できる通信（CTM 連携なしの場合）



RMI-IIOP 通信のタイムアウトは，図の A，B，C の 3 か所に設定できます。

- A の場合
  図 2-13 または図 2-14 の，項番 1. ～ 6. の RMI-IIOP 通信すべてにタイムアウトが有効になります。タイムアウトの設定は，定義ファイルで行います。定義ファイルでの設定方法については，「2.11.8 (1) RMI-IIOP 通信のタイムアウトの設定（J2EE サーバおよび EJB クライアントアプリケーションの設定）」を参照してください。
- B の場合
  図 2-13 または図 2-14 の，項番 1.，2. の CORBA ネーミングサービスとの通信にタイムアウトが有効になります。タイムアウトの設定は，定義ファイルで行います。定義ファイルでの設定方法については，「2.11.8(1) RMI-IIOP 通信のタイムアウトの設定（J2EE サーバおよび EJB クライアントアプリケーションの設定）」を参照してください。
- C の場合

図 2-13 または図 2-14 の，項番 4. ～ 6. の API の通信にタイムアウトが有効になります。
EJB クライアントアプリケーションでの create ～ビジネスメソッド～ remove までの API の通信のタイムアウト時間の設定は，アプリケーション開発時に API（RequestTimeoutConfigFactory クラスおよび RequestTimeoutConfig クラスのメソッド）によって設定します。API での設定方法については，「2.11.7 RMI-IIOP 通信のタイムアウトの実装」を参照してください。

(2) 通信タイムアウトの設定範囲とタイミング

この機能による通信タイムアウトは，ORB に対して設定します。つまり，ORB 配下のすべての RMI-IIOP 通信に対して設定されます。

設定のタイミングは，クライアント起動後の最初の new javax.naming.InitialContext() 実行時です。

CORBA ネーミングサービスを利用しない場合でも，この機能を利用するときは，new javax.naming.InitialContext() をクライアントの処理の最初で実行してください。

(3) 通信タイムアウト発生時のクライアントの処理

クライアントからのリクエストに対してプロパティ指定値以内にレスポンスが返らない場合，該当リクエストはタイムアウトとしてキャンセルされます。この機能によって通信タイムアウトが発生した場合，java.rmi.RemoteException（org.omg.CORBA.TIMEOUT）の例外が発生します。この機能を利用するクライアントでは，Enterprise Bean のビジネスメソッド呼び出しなどで，この例外が発生することを考慮する必要があります。

2.11.6 cosminexus.xml での定義

EJB コンテナでのタイムアウトの設定のうち，Stateful Session Bean，Entity Bean の EJB オブジェクトまたはインスタンス取得待ちのタイムアウトの定義は，cosminexus.xml の <ejb-jar> タグ内に指定します。設定するタグは，設定対象になる Enterprise Bean の種類ごとに異なります。

cosminexus.xml での EJB コンテナでのタイムアウトの定義について次の表に示します。

表 2-29 cosminexus.xml での EJB コンテナでのタイムアウトの定義

項目	指定するタグ	設定内容
Stateful Session Bean のタイムアウト	<session>-<stateful>-<removal-timeout> タグ	セッションが削除されるまでに非アクティブ状態に保持しておく時間を指定します。
Entity Bean の EJB オブジェクトのタイムアウト	<entity>-<entity-timeout> タグ	EJB オブジェクトの存在時間を指定します。

項目	指定するタグ	設定内容
インスタンス取得待ちのタイムアウト	Session Bean の場合 <session>-<stateless>-<instance-timeout> タグ Entity Bean の場合 <entity>-<instance-timeout> タグ	インスタンス取得タイムアウト時間を指定します。

2.11.7 RMI-IIOP 通信のタイムアウトの実装

RMI-IIOP 通信のタイムアウトは，API で設定できます。

API で設定できるのは，図 2-13，図 2-14 の C です。API で設定する場合，com.hitachi.software.ejb.ejbclient パッケージの API を利用します。com.hitachi.software.ejb.ejbclient パッケージの API の機能と文法については，マニュアル「Cosminexus アプリケーションサーバ リファレンス API 編」の「4. EJB クライアントアプリケーションで使用する API」を参照してください。

RMI-IIOP の通信タイムアウトは，クライアントのプロセス起動後の，最初の InitialContext 生成時にプロパティに指定した値が設定されます。ネーミングサービスを利用しない場合でも，RMI-IIOP の通信タイムアウトを設定する場合は，InitialContext を生成する必要があります。

ネーミングサービスの通信タイムアウトを設定する場合は，InitialContext 生成や lookup などの，JNDI の API 呼び出し時にプロパティに指定した値が設定されます。

2.11.8 実行環境での設定

EJB コンテナでのタイムアウトのうち，RMI-IIOP 通信のタイムアウトは，クライアントプロセスである J2EE サーバまたは EJB クライアントアプリケーションで設定できます。

また，Stateful Session Bean，Entity Bean の EJB オブジェクトまたはインスタンス取得待ちのタイムアウトは，J2EE アプリケーションで設定できます。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合に参照してください。

(1) RMI-IIOP 通信のタイムアウトの設定（J2EE サーバおよび EJB クライアントアプリケーションの設定）

設定できるのは，図 2-13，図 2-14 の A または B です。

プロパティで設定する場合，タイムアウトの設定方法は Enterprise Bean がどこから呼び出されるか（EJB クライアントの形態）によって異なります。設定は，呼び出し元（EJB クライアント側）の J2EE サーバまたは EJB クライアントアプリケーションのプ

ロパティとして設定します。

(a) EJB クライアントの形態が Enterprise Bean，JSP またはサーブレットの場合

設定は，クライアント側の Enterprise Bean，JSP またはサーブレットが動作する J2EE サーバに設定します。J2EE サーバの設定は，簡易構築定義ファイルで実施します。

簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の <configuration> タグ内に次のパラメタを指定してください。

指定するパラメタ
ejbserver.rmi.request.timeout

設定内容
RMI-IIOP 通信のクライアントとサーバ間の通信タイムアウト時間を指定します。

(b) EJB クライアントの形態が EJB クライアントアプリケーションの場合

EJB クライアントアプリケーションに，EJB クライアントアプリケーションの実行時に有効なプロパティとして設定します。

usrconf.properties（Java アプリケーション用ユーザプロパティファイル）に，次のキーを指定してください。

指定するキー
ejbserver.rmi.request.timeout キー

設定内容
RMI-IIOP 通信のクライアントとサーバ間の通信タイムアウト時間を指定します。

(2) Stateful Session Bean，Entity Bean の EJB オブジェクトまたはインスタンス取得待ちのタイムアウトの定義

Stateful Session Bean，Entity Bean の EJB オブジェクトまたはインスタンス取得待ちのタイムアウトの定義は，実行環境で設定することもできます。J2EE サーバにインポートした J2EE アプリケーションに設定します。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ実行してください。

実行環境での J2EE アプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。リファレンスマッピングの定義には，次の属性ファイルを使用します。

表 2-30 Stateful Session Bean，Entity Bean の EJB オブジェクトまたはインスタンス取得待ちのタイムアウトの定義に使用する属性ファイル

設定対象	属性ファイル
Session Bean	Session Bean 属性ファイル
Entity Bean	Entity Bean 属性ファイル

属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。cosminexus.xml での定義については，「2.11.6　cosminexus.xml での定義」を参照してください。

2.11.9　通信のタイムアウト設定時の注意事項

通信タイムアウト設定時の注意事項について説明します。

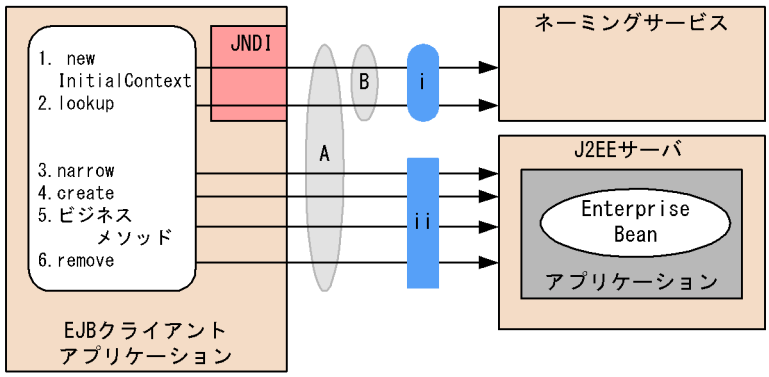
● 通信タイムアウトの定義が重複している場合の注意事項

図 2-13 または図 2-14 の A の範囲と B の範囲の両方に通信タイムアウトを設定している場合，および A の範囲と C の範囲に通信タイムアウトを設定している場合は次のように動作します。

- **A の範囲と B の範囲の両方に通信タイムアウトを設定している場合の動作**

  A の範囲と B の範囲の両方に通信タイムアウトを設定している場合，通信タイムアウトは次の図のようになります。

図 2-15　A の範囲と B の範囲の両方に通信タイムアウトを設定している場合の動作



（凡例）　：通信タイムアウトの設定範囲

i の部分の通信タイムアウト
Aの通信タイムアウト設定あり，Bの通信タイムアウト設定省略の場合
　Aの通信タイムアウトが適用されます。

Aの通信タイムアウト設定省略，Bの通信タイムアウト設定ありの場合
　Bの通信タイムアウトが適用されます。

Aの通信タイムアウト設定あり，Bの通信タイムアウト設定ありの場合
　Bの通信タイムアウトが適用されます。
　なお，Bの通信タイムアウト設定に0秒が設定されている場合は，Aの通信タイムアウトの設定値にかかわらず，通信タイムアウトは行いません。

なお，ii の部分の通信タイムアウトには，Aの通信タイムアウトが適用されます。

- **A の範囲と C の範囲の両方に通信タイムアウトを設定している場合の動作**

  A の範囲と C の範囲の両方に通信タイムアウトを設定している場合，C の範囲の通信タイムアウト設定が有効となります。

● クライアント実装時の注意事項

RMI-IIOP の通信タイムアウト，ネーミングサービスの通信タイムアウトと同時に，クライアントからのリクエストに対して指定値以内にレスポンスが返らない場合，該当するリクエストはタイムアウトとしてキャンセルされます。このとき，例外 java.rmi.RemoteException（org.omg.CORBA.TIMEOUT など）や javax.naming.NamingException が送出されます。通信タイムアウトを利用するクライアントを実装する場合，Enterprise Bean のメソッド呼び出しや JNDI の API などでこれらの例外が発生することを考慮してください。

● タイムアウト発生後のサーバ側の動作についての注意事項

クライアントからのリクエストがサーバ（ネーミングサービスや Enterprise Bean）に到着したあとに，サーバ側の処理中にタイムアウトが発生すると，クライアントに例外が返されます。ただし，タイムアウト発生後もサーバ側では正常に処理が継続されるため，Enterprise Bean のインスタンスの破棄や，リソースコネクションなどの資源の解放はされません。

2.12 Timer Service の機能

この節では，Timer Service の機能について説明します。

Timer Service は，指定した時刻，経過時間，または間隔で EJB コンテナが Enterprise Bean を呼び出す機能です。

この節の構成を次の表に示します。

表 2-31 この節の構成（Timer Service の機能）

分類	タイトル	参照先
解説	Timer Service の概要	2.12.1
	EJB タイマの生成とコールバック実行時の動作	2.12.2
	J2EE サーバ起動時の EJB タイマの自動生成（サーブレット方式）	2.12.3
	J2EE サーバ起動時の EJB タイマの自動生成（Management イベント方式）	2.12.4
	EJB タイマの削除	2.12.5
	Timer Service の運用機能	2.12.6
	EJB タイマとコールバックの動作	2.12.7
実装	Timer Service を使用するアプリケーションの実装	2.12.8
	Timer Service 実装時の注意事項	2.12.9
設定	実行環境での設定	2.12.10
注意事項	Timer Service を利用する場合の注意事項	2.12.11

注 「運用」について，この機能固有の説明はありません。

2.12.1 Timer Service の概要

Timer Service とは，指定した時刻，経過時間，または間隔で Enterprise Bean を呼び出す機能です。この機能は EJB コンテナが提供します。Timer Service を使用すると，マシン負荷が低い時間を指定したバッチ処理や，一定間隔での日次処理など，時刻を指定した処理を容易に実行できます。

ここでは，Timer Service で設定できるタイムアウトの内容や，タイムアウトを設定する EJB タイマの動作，Timer Service の運用などについて説明します。

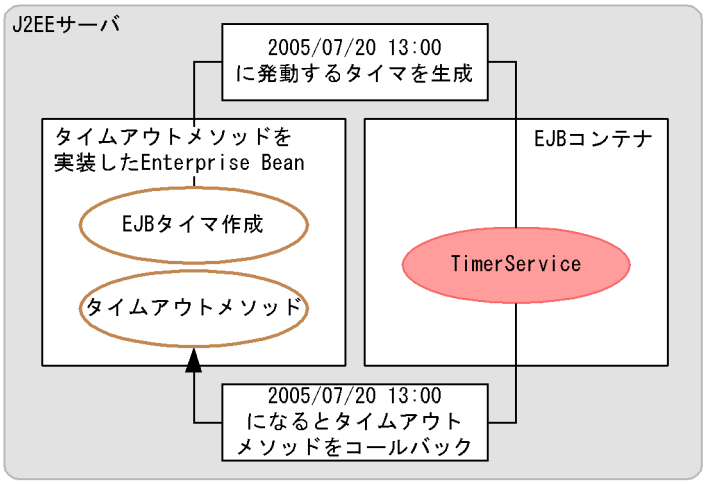
Timer Service で設定できるタイムアウトや Timer Service のサポート範囲，Timer Service でのトランザクション管理について説明します。

(1) Timer Service と EJB タイマ

Timer Service は，Java EE に規定された API を使用して，Enterprise Bean から操作します。時刻を指定した処理を行うには，EJB **タイマ**を生成します。EJB タイマにはタイムアウト時刻を指定します。生成された EJB タイマは，EJB コンテナに管理され，タイムアウト時刻になると，EJB コンテナによって Enterprise Bean のメソッドがコールバックされます。この際にコールバックされるメソッドを，**タイムアウトメソッド**と呼びます。

Timer Service の処理の概要を次の図に示します。

図 2-16　Timer Service の処理の概要



(2) EJB タイマの種別

EJB タイマに設定できるタイマ種別には次の 2 種類があります。

● single-event タイマ

タイムアウトメソッドを 1 回だけ実行するための EJB タイマです。
タイムアウトの設定には，タイムアウトメソッドを実行する時刻を指定する方法と，EJB タイマ生成メソッドを呼び出してからタイムアウトメソッドを実行するまでの時間を指定する方法があります。

● interval タイマ

一定間隔で，繰り返しタイムアウトメソッドを実行するための EJB タイマです。
タイムアウトの設定には，1 回目のタイムアウトメソッドを実行する時刻を指定する方法と，EJB タイマ生成メソッドを呼び出してから 1 回目のタイムアウトメソッドを実行するまでの時間を指定する方法があります。また，この設定のほかに，2 回目以降のタイムアウトメソッドを実行するためのタイムアウトの間隔を指定します。この間隔は，タイムアウトから次のタイムアウトまでの間隔です。

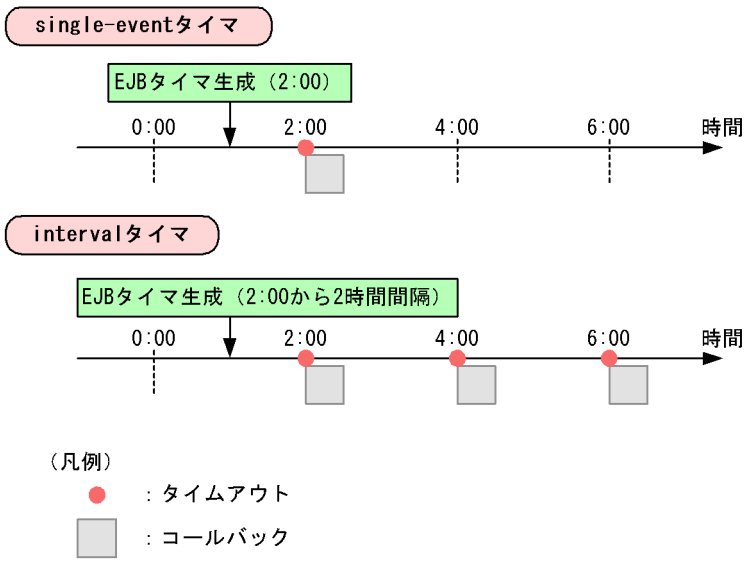
EJB タイマに設定できるタイマ種別と設定例を次の表に示します。

表 2-32 EJB タイマに設定できるタイマ種別と設定例

タイマ種別	設定例	説明
single-event	2006/4/15 12：00	2006/4/15 12：00 にタイムアウトメソッドを 1 回だけ実行します。
	24 時間後	EJB タイマが生成されてから 24 時間後に，タイムアウトメソッドを 1 回だけ実行します。
interval	2006/4/1 12：00 から 24 時間間隔	2006/4/1 12：00 に 1 回目のタイムアウトメソッドを実行します。そのあと，24 時間間隔でタイムアウトメソッドを繰り返し実行します。
	24 時間後から 10 時間間隔	EJB タイマが生成されてから 24 時間後に，1 回目のタイムアウトメソッドを実行します。そのあと，10 時間間隔でタイムアウトメソッドを繰り返し実行します。

single-event タイマと interval タイマの動作を次の図に示します。この図では，2：00 にタイムアウトメソッドを 1 回だけコールバックする single-event タイマと，2：00 から 2 時間間隔でタイムアウトメソッドをコールバックする interval タイマの動作を示しています。

図 2-17 single-event タイマと interval タイマの動作



(3) Timer Service のサポート範囲

Java EE の仕様で規定されている，Timer Service の機能のサポート状況を次の表に示します。

表 2-33 Java EE の仕様で規定されている Timer Service の機能のサポート状況

Java EE の仕様で規定されている Timer Service の機能	サポート状況
トランザクション	○
EJB タイマ永続性	×※
TimerService オブジェクトの取得（DI，JNDI ルックアップ，EJBContext）	○
タイムアウトメソッドの指定（アノテーション，TimedObject 実装）	○
タイムアウトメソッドの指定（DD による指定）	×

（凡例） ○：利用できる。 ×：利用できない。

注※
障害などで J2EE サーバを再起動したときは，再起動前に使用していた EJB タイマは引き継がれません。J2EE サーバ起動時に EJB タイマを自動で生成する場合は，「2.12.2(2) J2EE サーバ起動時の EJB タイマの自動生成」を参照してください。

Enterprise Bean の種別ごとに，Timer Service の機能のサポート状況を次の表に示します。

表 2-34 Enterprise Bean の種別ごとの，Timer Service の機能のサポート状況

Timer Service の機能	Message-driven Bean	Session Bean		Entity Bean
		Stateful Session Bean	Stateless Session Bean	
TimerService オブジェクトの取得	×	－	○	×
Timer Service に関するオブジェクト（TimerService，Timer，TimerHandle）の操作	×	○	○	×

（凡例） ○：利用できる。 ×：利用できない。 －：利用できない（Java EE 仕様）。

(4) Timer Service でのトランザクション管理

Timer Service は，トランザクションをサポートしています。具体的には，EJB タイマの生成，EJB タイマの削除，およびタイムアウトメソッドがトランザクションに対応しています。ここでは，タイムアウトメソッドのトランザクション管理について説明します。

EJB タイマの生成については，「2.12.2 EJB タイマの生成とコールバック実行時の動作」を，EJB タイマの削除については，「2.12.5 EJB タイマの削除」を参照してください。

(a) タイムアウトメソッドに設定できるトランザクション属性

タイムアウトメソッドには，トランザクション管理に BMT または CMT を選択できます。

CMT の場合に，タイムアウトメソッドに指定できるトランザクション属性を次に示します。

- Required 属性
- RequiresNew 属性
- NotSupported 属性

これ以外の属性が指定された場合，J2EE アプリケーションの開始に失敗します。

(b) タイムアウトメソッドのコールバックに対するトランザクション管理

タイムアウトメソッドに，CMT で Required 属性または RequiresNew 属性を指定した場合，タイムアウトメソッドのコールバック中にトランザクションがロールバックすると，コールバックをリトライします。コールバックのリトライについては「2.12.6(2) タイムアウトメソッドのコールバックリトライ」を参照してください。

2.12.2 EJB タイマの生成とコールバック実行時の動作

Timer Service によって指定した時刻に処理を実行するには，EJB タイマを生成します。また，EJB タイマによる処理の実行を停止するには，EJB タイマを削除します。EJB タイマの生成，削除のタイミングは，EJB タイマの種別や，トランザクションの管理下で処理が実行されるかどうかによって異なります。ここでは，EJB タイマの生成と削除のタイミング，および EJB タイマの生成時と削除時のタイムアウトメソッドのコールバックの動作について説明します。

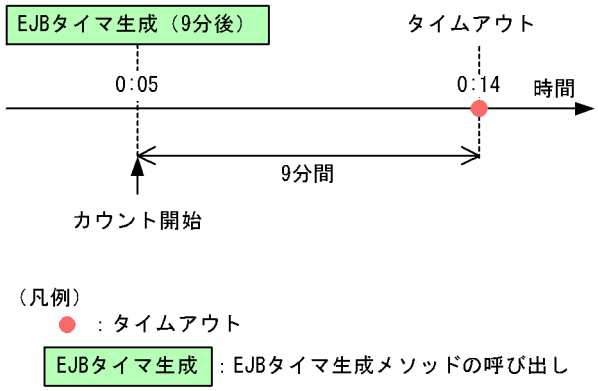
(1) EJB タイマの生成

EJB タイマ生成では，EJB タイマ生成メソッド（javax.ejb.TimerService オブジェクトの createTimer メソッド）によって，EJB タイマが一つ生成されます。EJB タイマが生成されると，指定した時刻にタイムアウトが発生してタイムアウトメソッドがコールバックされます。

(a) タイムアウト時刻のカウント

EJB タイマ生成時に，EJB タイマを呼び出してから 1 回目のタイムアウトメソッドを実行するまでの時間を指定している EJB タイマの場合，時間のカウントは，EJB タイマ生成メソッドが呼ばれた時刻を基点に開始されます。EJB タイマの生成とカウントの開始について次の図に示します。

図 2-18 EJB タイマの生成とカウントの開始



(b) EJB タイマ生成とタイムアウトメソッドのコールバック実行のタイミング

EJB タイマが生成されるタイミングおよびタイムアウトメソッドがコールバックされるタイミングは，EJB タイマ生成がトランザクションの管理下で行われるかどうかによって異なります。

EJB タイマ生成がトランザクションの管理下で行われる場合

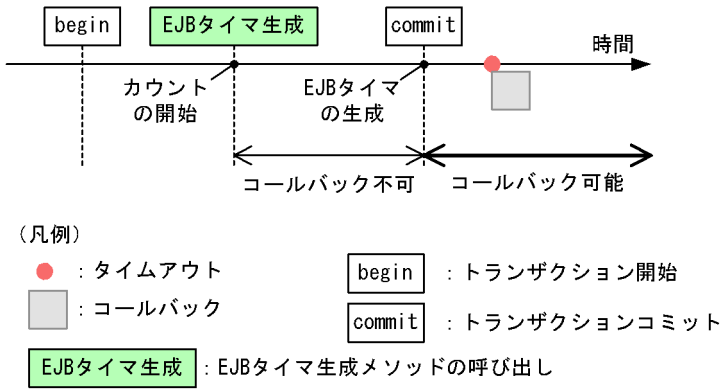
EJB タイマは，トランザクションがコミットした時に生成されます。

タイムアウトメソッドのコールバックが実行されるタイミングは，トランザクションがコミットしたあとにタイムアウト時刻となる EJB タイマの場合と，トランザクションがコミットする前にタイムアウト時刻となる EJB タイマの場合とで異なります。なお，トランザクションがロールバックされた場合は，EJB タイマの生成は取り消されます。

- **トランザクションがコミットしたあとにタイムアウト時刻となる EJB タイマの場合**

  指定した時刻どおりにタイムアウトメソッドがコールバックされます。EJB タイマ生成とコールバックの実行について次の図に示します。

図 2-19 EJB タイマ生成とコールバックの実行（トランザクションがコミットしたあとにタイムアウト時刻となる EJB タイマの場合）

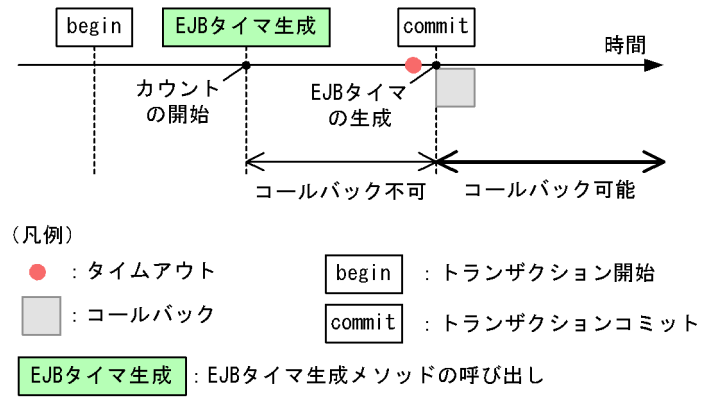


この図では，EJB タイマ生成メソッドが呼ばれた時点から，タイムアウト時刻までのカウントが開始され，指定した時刻どおりにタイムアウトメソッドがコールバックされます。EJB タイマが生成されるのは，トランザクションコミット時です。

- **トランザクションがコミットする前にタイムアウト時刻となる EJB タイマの場合**

トランザクションがコミットした直後にタイムアウトメソッドがコールバックされます。EJB タイマ生成とコールバックの実行について次の図に示します。

図 2-20 EJB タイマ生成とコールバックの実行（トランザクションがコミットする前にタイムアウト時刻となる EJB タイマの場合）

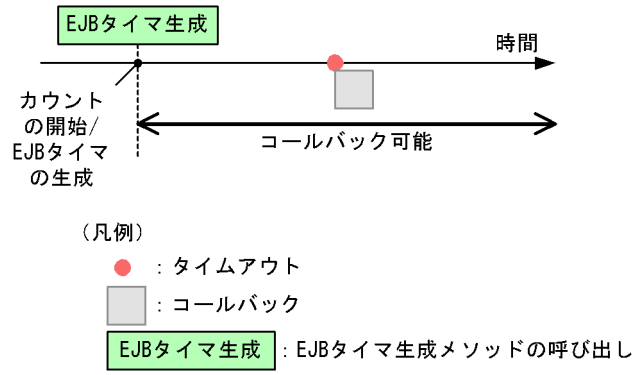


この図では，EJB タイマ生成メソッドが呼ばれた時点から，タイムアウト時刻までのカウントが開始され，トランザクションコミットの前にタイムアウト時刻になります。トランザクションコミット時まで EJB タイマは生成されないため，タイムアウト時刻になってもタイムアウトメソッドはコールバックされません。トランザクションがコミットした直後にコールバックされます。

EJB タイマ生成がトランザクションの管理下で行われない場合

EJB タイマは EJB タイマ生成メソッドが呼ばれた直後に生成されます。
トランザクションの管理下でない場合の EJB タイマ生成について次の図に示します。

図 2-21　EJB タイマ生成とコールバックの実行（トランザクションの管理下でない場合）



この図では，EJB タイマ生成メソッドが呼ばれた直後に EJB タイマが生成されます。そのあと，指定した時刻どおりにタイムアウトメソッドがコールバックされます。

(2) J2EE サーバ起動時の EJB タイマの自動生成

Timer Service を使用している場合に，J2EE サーバ起動時に EJB タイマを自動で生成する方法について説明します。EJB タイマの自動生成には次の方法があります。

- サーブレット方式
  サーブレット内で，EJB タイマの自動生成処理を実装する方法です。
- Management イベント方式
  Management イベントを使用して EJB タイマを自動生成する方法です。J2EE サーバの起動時に出力されるメッセージを契機に発生する Management イベントとして，EJB タイマの自動生成を設定します。
  EJB タイマを生成する Enterprise Bean を呼び出す EJB クライアントプログラムを作成している場合は，J2EE アプリケーションの修正は不要です。

Timer Service の機能については，「2.12.1　Timer Service の概要」を参照してください。

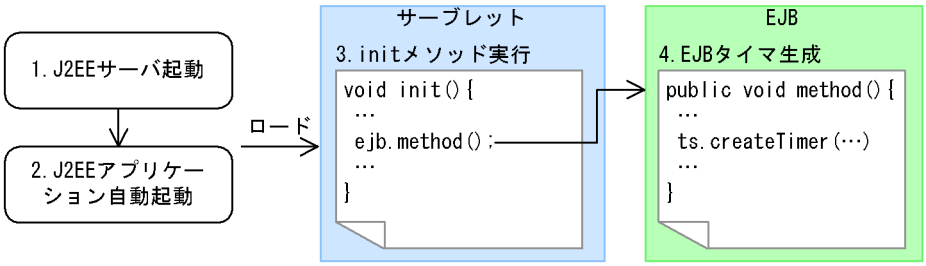
2.12.3　J2EE サーバ起動時の EJB タイマの自動生成（サーブレット方式）

EJB タイマの自動生成処理をサーブレットで実装する場合の，EJB タイマ自動生成の流れと設定について説明します。

(1) EJB タイマの自動生成の流れ

サーブレット方式では，次のような流れで EJB タイマが自動生成されます。

図 2-22 EJB タイマの自動生成の流れ（サーブレット方式）



1. J2EE **サーバを再起動する。**
2. J2EE **アプリケーションが自動で開始する。**
   J2EE サーバ停止時に開始状態だった J2EE アプリケーションが自動で開始されます（アプリケーションの自動起動を有効にしておく必要があります）。
3. init **メッドが実行される。**
   J2EE アプリケーションが開始するとサーブレットがロードされ，init メッドが実行されます。init メソッドでは，Enterprise Bean を生成してビジネスメソッドを呼び出します。
4. EJB **タイマが生成される。**
   Enterprise Bean のビジネスメソッドで，Timer Service オブジェクトから EJB タイマが生成されます。

(2) EJB タイマを自動生成するための設定

サーブレットで EJB タイマを自動生成するには，次の設定をします。

1. Enterprise Bean **に** EJB **タイマを生成するビジネスメソッドを作成する。**
2. **サーブレットを作成する。**
   init メソッドで，手順 1. で作成したビジネスメソッドを呼び出すようにします。
3. J2EE **アプリケーションの開始時に，手順** 2. **で作成したサーブレットをロードさせる。**
   DD（web.xml）で，手順 2. で用意したサーブレットの <load-on-startup> タグの値を 0 以上に設定します。
4. J2EE **サーバ起動時に，アプリケーションの自動起動を有効にする。**

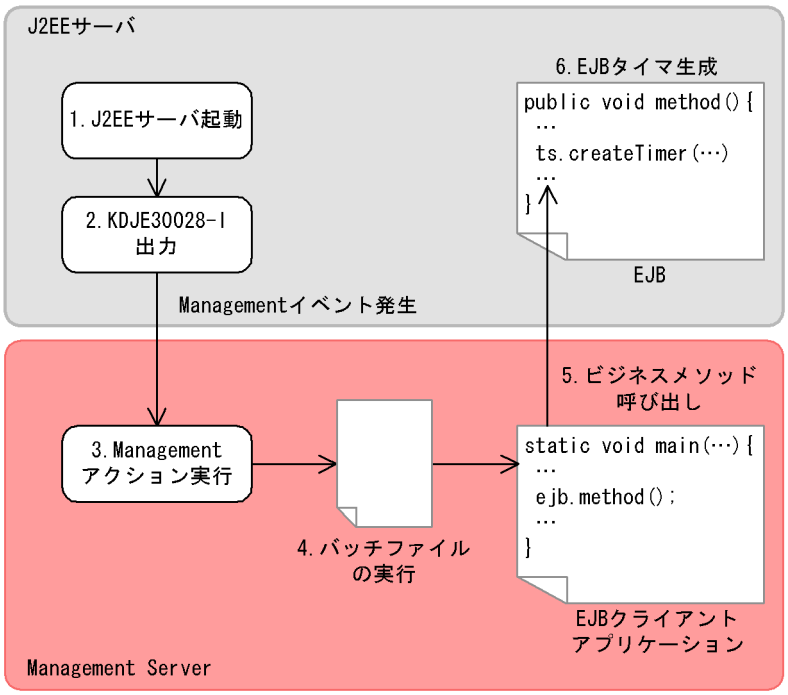
2.12.4 J2EE サーバ起動時の EJB タイマの自動生成（Management イベント方式）

EJB タイマの自動生成処理を Management イベントとして設定する場合の，EJB タイマ自動生成の流れと設定について説明します。

（1） EJB タイマの自動生成の流れ

Management イベント方式では，次のような流れで EJB タイマが自動生成されます。

図 2-23 EJB タイマの自動生成の流れ（Management イベント方式）



1. J2EE **サーバを再起動する。**
2. **メッセージが出力される。**
   J2EE サーバの起動が完了するとメッセージ KDJE30028-I が出力されます。このメッセージが出力されることで，Management イベントが発生します。
3. Management **アクションが実行される。**
   Management イベントを受けて，Management アクションが実行されます。
   Management アクションでは，バッチファイルが実行されます。
4. **バッチファイルが実行される。**
   バッチファイルで，EJB クライアントアプリケーションが実行されます。
5. EJB **クライアントアプリケーションでビジネスメソッドが呼ばれる。**

EJB クライアントアプリケーションで，Enterprise Bean を生成してビジネスメソッドを呼び出します。

6. **EJB タイマが生成される。**
   Enterprise Bean のビジネスメソッドで，Timer Service オブジェクトから EJB タイマが生成されます。

(2) EJB タイマを自動生成するための設定

Management イベントを使用して EJB タイマを自動生成するには，次の設定をします。

1. **Enterprise Bean に EJB タイマを生成するビジネスメソッドを作成する。**
2. **EJB クライアントアプリケーションを作成する。**
   手順 1. で作成したビジネスメソッドを呼び出すようにします。
3. **手順 2. で作成した EJB クライアントアプリケーションを実行する環境，およびバッチファイルを作成します。**
4. **Management イベントの自動実行を設定する。**
   Management イベントを有効にします。J2EE サーバ開始完了時に出力される KDJE30028-I を Management イベントに設定し，この Management イベントに対する Management アクションとして，手順 3. で準備したバッチファイルを設定します。
   Management イベントの詳細については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 運用／監視／連携編」の「9. Management イベントの通知と Management アクションによる処理の自動実行」を参照してください。

2.12.5 EJB タイマの削除

EJB タイマを削除するには，EJB タイマキャンセルを行います。

なお，single-event タイマの場合は，タイムアウトメソッドのコールバック完了時に EJB タイマが削除されます。interval タイマの場合，EJB タイマキャンセルが行われるまで EJB タイマは削除されません。

EJB タイマキャンセルでは，EJB タイマキャンセルメソッド（javax.ejb.Timer オブジェクトの cancel メソッド）によって，EJB タイマが一つ削除されます。EJB タイマが削除されると，それ以降のコールバックは実行されません。EJB タイマキャンセルによって EJB タイマが削除されるタイミングは，EJB タイマキャンセルがトランザクションの管理下で行われるかどうかによって異なります。

EJB タイマキャンセルがトランザクションの管理下で行われる場合

EJB タイマは，トランザクションのコミット時に削除されます。そのため，EJB タイマキャンセルメソッドが呼ばれてから，トランザクションがコミットするまでの間にタイムアウト時刻になることがあります。

トランザクションがロールバックされた場合は，EJB タイマキャンセルは取り消されます。

EJB タイマキャンセルがトランザクションの管理下で行われない場合

EJB タイマは，EJB タイマキャンセルメソッドが呼ばれた直後に削除されます。

参考

J2EE アプリケーションの停止時には，停止対象の J2EE アプリケーションの EJB タイマはすべて削除されます。J2EE サーバの停止または異常終了時には，J2EE サーバ上の EJB タイマはすべて削除されます。

2.12.6 Timer Service の運用機能

Timer Service の運用で使用する機能について説明します。運用機能には次の二つがあります。

- タイムアウトメソッドのコールバックスレッド数制御機能
- タイムアウトメソッドのコールバックリトライ機能

これらの機能は，J2EE サーバのプロパティをカスタマイズして設定します。設定するプロパティについては，「2.12.10　実行環境での設定」を参照してください。

(1) タイムアウトメソッドのコールバックスレッド数制御機能

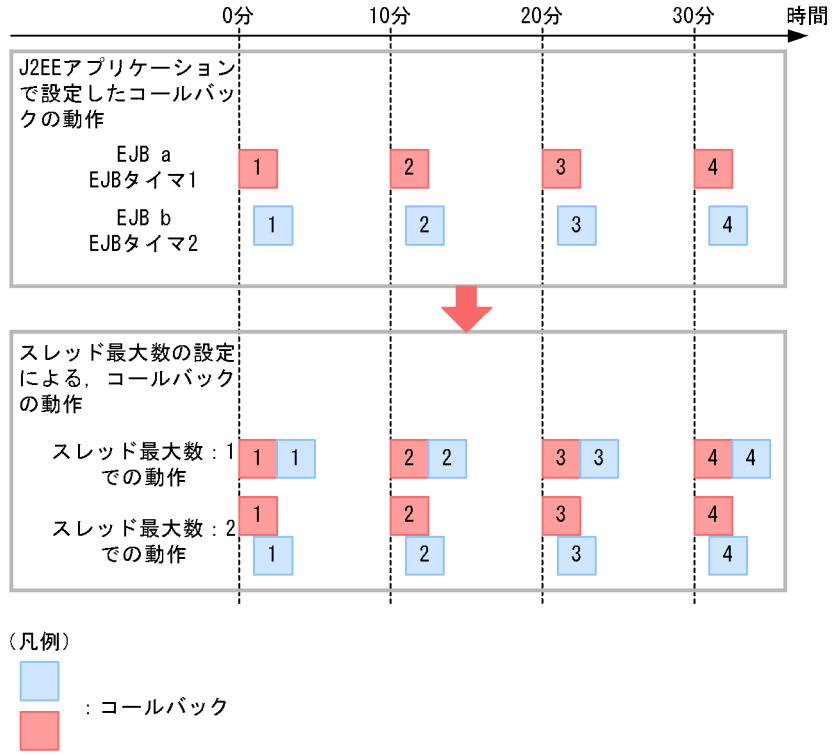
タイムアウトメソッドのコールバックを処理するスレッドを，J2EE サーバ全体で同時に幾つ実行するかを設定します。これをコールバック最大スレッド数といいます。

複数の EJB タイマが同時にタイムアウトした場合，タイムアウトメソッドのコールバックの最大スレッド数の設定によって，次のような動作になります。

- コールバック最大スレッド数：1（デフォルト）の場合
  コールバック処理は順次行われます。そのため，EJB タイマに設定した時刻よりも遅れてコールバック処理が開始されることがあります。
- コールバック最大スレッド数：2 以上の場合
  設定した数だけ並行してコールバック処理が実行されます。最大スレッド数を増加させた場合，その分だけリソースを消費するため，並行に実行させたいスレッド数を適切に設定してください。

コールバック最大スレッド数の設定とコールバック処理の関係について次の図に示します。

図 2-24　コールバック最大スレッド数の設定とコールバック処理の関係



この図の場合，EJB タイマ 1 のタイムアウトの 1 分後に，EJB タイマ 2 のタイムアウトになる設定にしています。コールバック最大スレッド数を 1 に設定した場合は，EJB タイマ 1 のコールバック処理が終了したあと，EJB タイマ 2 のコールバック処理が開始されます。コールバック最大スレッド数を 2 に設定した場合は，二つのコールバックを並行して処理できるため，設定どおりに EJB タイマ 1 の 1 分後に EJB タイマ 2 のコールバック処理が開始されます。

注意事項

コールバックのスレッド数に余裕がある場合でも Enterprise Bean のインスタンスが不足していると，インスタンスが解放されるのを待ってコールバック処理が行われます。そのため，コールバックするインスタンス数を考慮して Enterprise Bean インスタンスプールを設定してください。

(2) タイムアウトメソッドのコールバックリトライ

タイムアウトメソッドのコールバックに失敗した場合は，コールバックをリトライします。

コールバックが失敗する要因として次のケースが考えられます。

- タイムアウトによるコールバック時に，非検査例外（java.lang.RuntimeException, java.lang.Error およびそれらのサブクラス）がスローされた場合
- タイムアウトメソッドが CMT の Required 属性または RequiresNew 属性で，そのトランザクションがロールバックした場合

コールバックのリトライを行うには，次の二つを設定します。

リトライ回数
リトライする回数を設定します。0 を設定した場合は，リトライしません。また，リトライ実行回数が設定したリトライ回数に達すると，そのタイムアウトでのリトライは行いません。

リトライの実行間隔
コールバックが失敗してから，リトライのコールバックを行うまでの時間を設定します。

2.12.7 EJB タイマとコールバックの動作

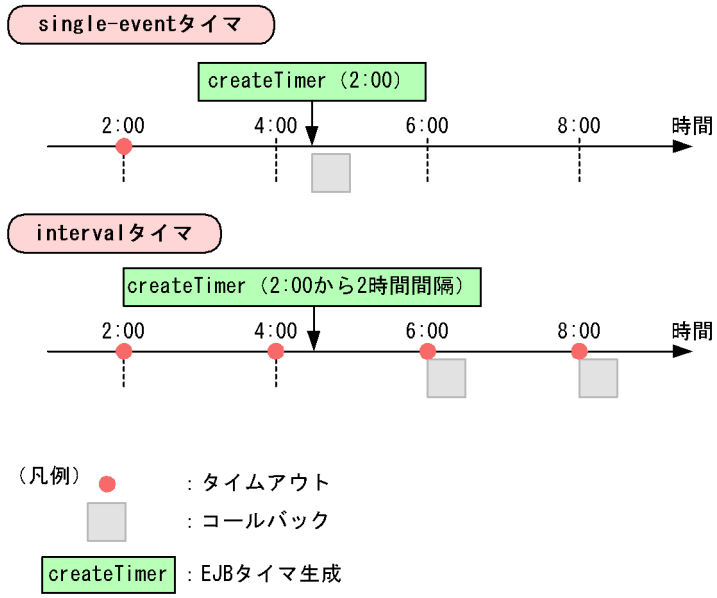
次の場合の EJB タイマの動作およびコールバックの動作について説明します。

- EJB タイマに過去の時刻が指定された場合
- 一つの Enterprise Bean のクラスで複数の EJB タイマを設定している場合
- タイムアウト時に，前回のコールバックが終了していない場合
- 複数スレッドから EJB タイマキャンセルが呼ばれた場合
- コールバック実行中に EJB タイマキャンセルが呼ばれた場合
- コールバック時に，EJB タイマキャンセルを行った未決着トランザクションがある場合

（1） EJB タイマに過去の時刻が指定された場合

EJB タイマに過去の時刻が指定された場合，single-event タイマのときは EJB タイマの生成直後に 1 回タイムアウトメソッドをコールバックします。interval タイマの場合は，EJB タイマ生成後の時刻に発生するタイムアウト時にタイムアウトメソッドをコールバックします。EJB タイマに過去の時刻が指定された場合の動作を次の図に示します。

図 2-25 EJB タイマに過去の時刻が指定された場合の動作



single-event タイマの場合

2：00 にタイムアウトが発生する single-event タイマが，すでにタイムアウトの時刻を過ぎた 4：15 に生成された場合，生成直後に 1 回タイムアウトメソッドをコールバックします。

interval タイマの場合

2：00 から 2 時間間隔でタイムアウトが発生する interval タイマが，すでにタイムアウトの時刻を過ぎた 4：15 に生成された場合，それ以降の時刻に発生するタイムアウト時（1 回目は 6：00）にタイムアウトメソッドをコールバックします。

(2) 一つの Enterprise Bean のクラスで複数の EJB タイマを設定している場合

一つの Enterprise Bean クラスに対して，複数の EJB タイマがある場合に，それらの複数の EJB タイマのコールバックが重なったときは，コールバック処理が並列に行われます。ただし，並列に処理できるコールバックスレッドや Enterprise Bean インスタンスがある場合に限ります。このようなコールバックスレッドやインスタンスがない場合は，スレッドやインスタンスが解放されるまで待ちます。

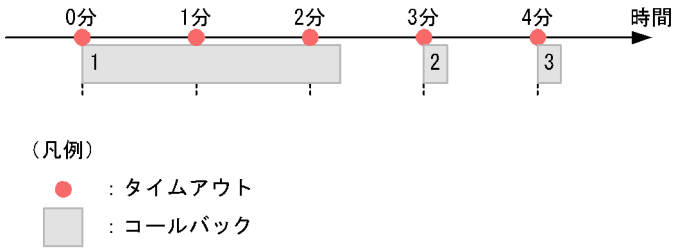
(3) タイムアウト時に，前回のコールバックが終了していない場合

一つの EJB タイマで同時に実行できるコールバックは一つだけです。タイムアウトメソッドの処理は，処理内容によっては時間が長く掛かることがあります。そのため，interval タイマの場合，コールバック処理の実行中に次のタイムアウトの時刻を過ぎてしまい，一つのコールバックが完了するまでに複数のタイムアウト時刻を経過することがあります。この場合，タイムアウトの時刻どおりに実行されなかったコールバック処

理は行わないで，前回のコールバックが終了した時刻よりあとに発生するタイムアウトのコールバック処理を行います。

コールバック時に，前回のコールバックが終了していない場合の動作を次の図に示します。

図 2-26　コールバック時に前回のコールバックが終了していない場合の動作



この図の場合，1 分間隔でタイムアウトが発生してコールバック処理を実行する設定にしています。1 回目のコールバックの処理中に，2 回目と 3 回目のタイムアウトの時刻を過ぎてしまった場合，1 回目のコールバック処理が終了したあとの，次のタイムアウト時刻（予定では 4 回目のタイムアウトの時刻）に 2 回目のコールバック処理を行います。タイムアウト時刻を過ぎてしまった 2 回分のコールバック処理は行われません。

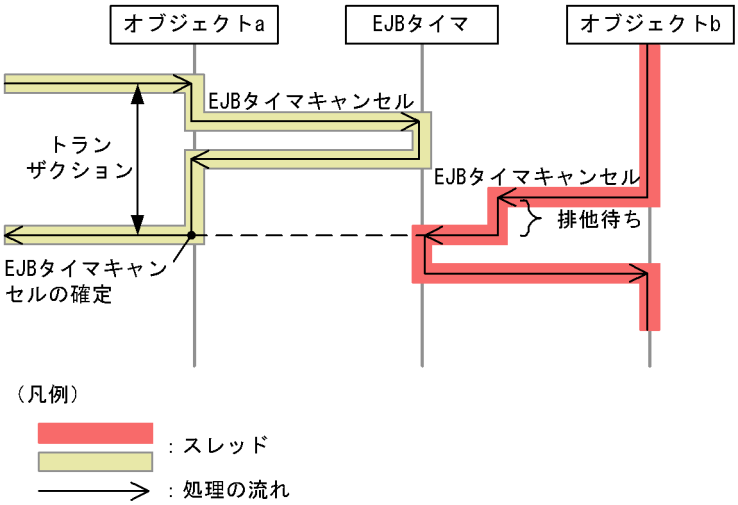
（4） 複数スレッドから EJB タイマキャンセルが呼ばれた場合

同時に複数のスレッドから，同一 EJB タイマの EJB タイマキャンセルメソッドが呼ばれた場合，キャンセルの処理は順次行われます。

ただし，トランザクション管理下で EJB タイマキャンセルが行われた場合，トランザクションが決着するまでは，EJB タイマが削除されるかどうかを確定できません。そのため，そのほかのスレッドが同じ EJB タイマのキャンセルを行うと，排他待ちとなります。

同時に複数のスレッドから，同一の EJB タイマのキャンセルが呼ばれた場合の動作を次の図に示します。

図 2-27 複数スレッドからの EJB タイマのキャンセル



(5) コールバック実行中に EJB タイマキャンセルが呼ばれた場合

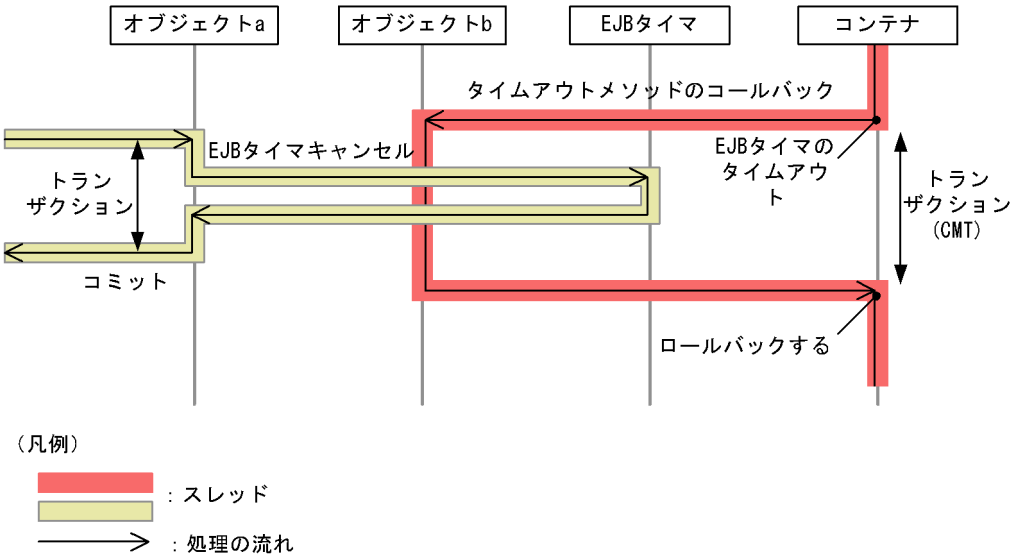
EJB タイマのキャンセル時に，タイムアウトメソッドのコールバックが実行中の場合，次のような動作になります。

- EJB タイマキャンセルメソッドの呼び出しは正常終了します。
- タイムアウトメソッドのコールバック処理は継続して実行されますが，コールバック完了後に次の動作をします。
  - タイムアウトメソッドが CMT のトランザクションを使用している場合，そのトランザクションはロールバックされます。
  - タイムアウトメソッドが CMT のトランザクションを使用していない場合，コールバック中に行われたトランザクションの決着には関与しません。
- 通常ではコールバックをリトライする次のケースで終了した場合でも，リトライしません。
  - CMT のトランザクションがロールバックされた場合
  - 非検査例外がスローされた場合
- コールバック完了時に，メッセージログにメッセージが出力されます。
  - タイムアウトメソッドが CMT のトランザクションを使用している場合，KDJE43161-W が出力されます。
  - タイムアウトメソッドが CMT のトランザクションを使用していない場合は，KDJE43160-W が出力されます。
- CMT のトランザクションを使用するタイムアウトメソッドのコールバック実行中に，別のトランザクションから EJB タイマキャンセルが行われた場合と，CMT のトランザクションを使用するタイムアウトメソッドの中で EJB タイマキャンセルが行われた

場合では動作が異なります。

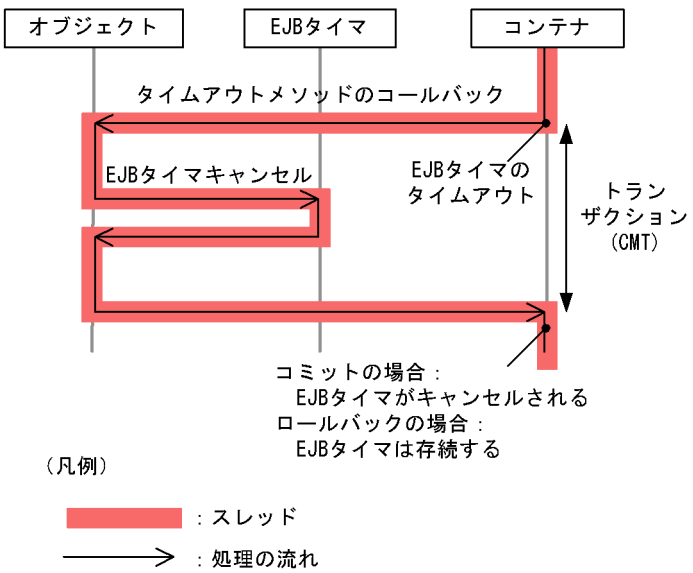
- CMT のトランザクションを使用するタイムアウトメソッドのコールバック実行中に，別のトランザクションから EJB タイマキャンセルが行われた場合の動作を次の図に示します。

図 2-28 コールバック実行中のキャンセル



- CMT トランザクションを使用するタイムアウトメソッドの中で EJB タイマキャンセルが行われた場合の動作を次の図に示します。

図 2-29 タイムアウトメソッドでの EJB タイマキャンセル

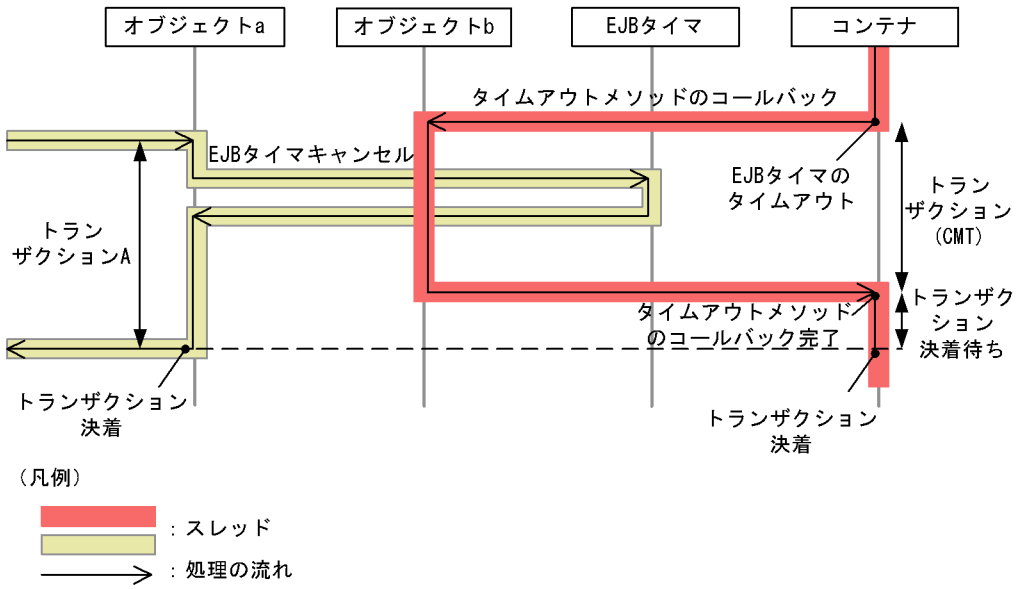


この図の場合，実際に EJB タイマがキャンセルされるのは，タイムアウトメソッド完了後です。このため，トランザクションがコミットした場合，EJB タイマはキャンセルされます。トランザクションがロールバックした場合，EJB タイマは存続します。

(6) コールバック時に，EJB タイマキャンセルを行った未決着トランザクションがある場合

EJB タイマキャンセルを行った未決着のトランザクション A が存在する場合，タイムアウトメソッドのコールバックの処理は，コールバック完了時からトランザクション A が決着するまで排他待ちとなります。したがって，タイムアウトメソッドが CMT のトランザクションの管理下にある場合，そのトランザクションの決着は，トランザクション A が決着するまで待ちます。この場合の流れを次の図に示します。

図 2-30　キャンセルを行った未決着トランザクションがある場合のコールバックの動作



2.12.8　Timer Service を使用するアプリケーションの実装

Timer Service を使用する場合のアプリケーションの実装内容を示します。

● タイムアウトメソッドの指定

次に示すどちらかの方法で，コールバックするタイムアウトメソッドを指定します。

- タイムアウトメソッドとして利用するメソッドに，Timeout アノテーションを指定します
- Enterprise Bean に TimedObject インタフェースを実装します。この場合，TimedObject インタフェースで定義されている ejbTimeout メソッドがタイムアウ

トメソッドとなります。

● javax.ejb.TimerService オブジェクトの取得

DI や EJBContext インタフェースの getTimerService メソッドや，JNDI の lookup メソッドなどを使用して，TimerService オブジェクトを取得します。

● EJB タイマ（Timer）の生成

TimerService オブジェクトの createTimer メソッドを呼び出し，Timer を生成するコードを実装します。

● EJB タイマ（Timer）のキャンセル

キャンセルする処理が必要な場合は，Timer をキャンセルするコードを実装します。Timer は，javax.ejb.TimerService や javax.ejb.TimerHandle から取得します。

これらの処理を実装した例と，実装時の注意事項を示します。

（1） DI を使用した場合の実装例（タイムアウトメソッドを Timeout アノテーションで指定）

アノテーションを使用した場合の実装例を示します。この例では，Timeout アノテーション（@Timeout）でタイムアウトメソッド（myTimeout）を指定しています。

```
@Stateless public class TimerSessionBean{
  @Resource TimerService timerService;

  public void createMyTimer(long intervalDuration){
    Timer timer = timerService.createTimer
                          (intervalDuration, "MyTimer");
  }

  @Timeout public void myTimeout(Timer timer) {
    System.out.println("TimerSessionBean: myTimeout ");
  }

  public void cancelTimers(){
    Collection<Timer> timers = timerService.getTimers();
    for(Timer timer: timers) {
      timer.cancel();
    }
  }
}
```

（2） EJBContext を利用した場合の実装例（TimedObject インタフェースを実装）

EJBContext のサブクラスである SessionContext を利用して，TimerService オブジェクトを取得する例を示します。この例では，TimedObject インタフェースをインプリメントして実装しています。

```
public class TimerSessionBean implements SessionBean, TimedObject{
  private SessionContext context;

  public void createMyTimer(long intervalDuration) {
    System.out.println("TimerSessionBean: start createTimer ");
    TimerService timerService = context.getTimerService();
    Timer timer = timerService.createTimer
                              (intervalDuration, "MyTimer");
  }

  public void ejbTimeout(Timer timer) {
    System.out.println("TimerSessionBean: ejbTimeout ");
  }

  public void setSessionContext(SessionContext sc) {
    context = sc;
  }
}
```

（3） lookup を利用した場合の実装例（TimedObject インタフェースを実装）

JNDI を使用して，TimerService オブジェクトを取得する例を示します。この例では，TimedObject インタフェースをインプリメントして実装しています。

```
public class TimerSessionBean implements SessionBean, TimedObject{
  private SessionContext context;

  public void createMyTimer(long intervalDuration) {
    System.out.println("TimerSessionBean: start createTimer ");
    InitialContext context = new InitialContext();
    TimerService timerService =
      (TimerService)context.lookup("java:comp/TimerService");
    Timer timer = timerService.createTimer
                              (intervalDuration, "MyTimer");
  }

  public void ejbTimeout(Timer timer){
    System.out.println("TimerSessionBean: ejbTimeout ");
  }
}
```

2.12.9 Timer Service 実装時の注意事項

Timer Service を実装するときの注意事項を示します。

（1） createTimer メソッドの引数 info の指定

Timer オブジェクトの getInfo メソッドは，TimerService オブジェクトの createTimer メソッドで引数 info に指定されたオブジェクト自体を戻り値とします。そのため，getInfo メソッドの戻り値が createTimer メソッド実行時のオブジェクトと異なる状態である場合があり，トラブルが生じやすくなっています。

これを防ぐために，createTimer メソッドで引数 info に指定するオブジェクトは String や Integer などの不変オブジェクトにする，または引数 info に指定したオブジェクトの

状態を変更しないことを推奨します。変更した場合には，getInfo メソッドの戻り値は変更後のオブジェクトになります。

(2) DD および属性ファイルでのタイムアウトメソッド指定

DD や属性ファイルで <method> タグにタイムアウトメソッドを指定する場合は，次のどちらかとしてください。

- <method> タグ直下に <method-intf> タグの定義を追加しない。
- <method-intf> タグの要素を空にする。

(3) Timer Service オブジェクトをサポートしていない種別の Bean から Timer Service オブジェクトを取得しようとした場合の動作

Timer Service をサポートしない取得手段を次の表に示します。Timer Service をサポートしない種別の Bean やサーブレットから TimerService オブジェクトを取得しようとした場合，手段によって次のような動作をします。

表 2-35 Timer Service オブジェクトをサポートしていない種別の Bean から Timer Service オブジェクトを取得しようとした場合の動作

TimerService オブジェクトの取得手段	動作
EJBContext#getTimerService	IllegalStateException 例外をスローします。
JNDI のルックアップ	NamingException 例外をスローします。
DI	デプロイに失敗します。

Timer Service をサポートしている種別の Bean は，タイムアウトメソッドを実装しているかどうかに関係なく，TimerService オブジェクトを取得できます。

(4) TimerService が提供する API の動作仕様

TimerService が提供する API を呼び出したときの動作のうち，EJB 仕様書で明確に示されていない動作仕様があります。ここでは，javax.ejb.TimerService および javax.ejb.Timer のアプリケーションサーバでの動作仕様を示します。

● javax.ejb.TimerService

javax.ejb.TimerService のメソッドを呼び出した Bean がタイムアウトメソッドを実装している場合，およびタイムアウトメソッドを実装していないときの動作を，次の表に示します。

表 2-36 Bean から javax.ejb.TimerService の API を利用したときの動作

タイムアウトメソッドの実装	javax.ejb.TimerService のメソッドの種類	
	createTimer	getTimers
実装している場合※	EJB タイマ生成処理を実行します。	EJB タイマのコレクションを返します。
実装していない場合	IllegalStateException 例外をスローします。	空のコレクションを返します。

注※
タイムアウトメソッドを実装している場合は，EJB の仕様どおりに動作します。

● javax.ejb.Timer の API

タイムアウトメソッドが実行している場合，および実行していない場合に，javax.ejb.Timer に対してメソッドを呼び出したときの動作を，次の表に示します。

表 2-37 javax.ejb.Timer の API を利用したときの動作

タイムアウトメソッドの実行	javax.ejb.Timer のメソッドの種類			
	cancel	getHandle, getInfo	getNextTimeout	getTimeRemaining
実行していない場合※	「2.12.7 EJB タイマとコールバックの動作」を参照してください。	仕様どおりに動作します。	次のタイムアウトが起こる時刻を返します。	次のタイムアウトまでの時間を返します。
実行している場合			実行中のタイムアウトの開始予定時刻として登録されていた時刻を返します。	0 を返します。

注※
タイムアウトメソッドを実行していない場合は，EJB の仕様どおりに動作します。

参考

アプリケーションサーバでは，Timer Service のサンプルプログラムを提供しています。サンプルプログラムの概要および実行方法については，マニュアル「Cosminexus アプリケーションサーバ システム構築・運用ガイド」の「付録 F アプリケーションサーバが提供するサンプルプログラム」を参照してください。

(5) Timer Service を使用できる EJB のバージョン

Timer Service は，EJB 2.1 以降で使用できます。

2.12.10 実行環境での設定

TimerService を使用する場合，J2EE サーバの設定が必要です。

J2EE サーバの設定は，簡易構築定義ファイルで実施します。簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の <configuration> タグ内に指定します。

簡易構築定義ファイルでの TimerService の定義について次の表に示します。

表 2-38 簡易構築定義ファイルでの TimerService の定義

項目	指定するパラメタ	設定内容
リトライ最大回数	ejbserver.ejb.timerservice.retryCount	Timer Service のタイムアウトメソッドのコールバックをリトライする最大回数を指定します。
リトライ間隔	ejbserver.ejb.timerservice.retryInterval	Timer Service のタイムアウトメソッドのコールバックをリトライする間隔を秒単位で指定します。
タイムアウトメソッドをコールバックする最大スレッド数	ejbserver.ejb.timerservice.maxCallbackThreads	J2EE サーバ全体での Timer Service のタイムアウトメソッドをコールバックする最大スレッド数を指定します。

2.12.11 Timer Service を利用する場合の注意事項

Timer Service を利用する場合の注意事項について説明します。

- Timer Service を利用した場合，EJB タイマに指定した時刻と，実際にタイムアウトメソッドがコールバックされる時刻に差が生じることがあります。この場合，次の要因が考えられます。
  - ガーベージコレクションの実行
    EJB タイマで指定した時刻に，JavaVM でガーベージコレクションが実行された場合，ガーベージコレクションの処理が優先されます。ガーベージコレクションが終了してからタイムアウトメソッドがコールバックされるため，指定した時刻との差異が生じることがあります。
  - プラットフォームやハードウェア環境
    Timer Service では，JavaVM の時間を使用します。JavaVM の時間は，プラットフォームやハードウェア環境に依存します。コールバックも JavaVM の時間に従って実行されるため，指定した時刻との差異が生じることがあります。
- J2EE サーバが稼働するマシンで，NTP クライアントソフトウェアなどによってシステム時刻が修正された場合，登録済みの EJB タイマのタイムアウト時刻は次のような動作になります。
  - single-event タイマの場合
    修正前のシステム時刻が継続されているものとして，タイムアウトが発生します。
  - interval タイマの場合

修正後，修正前のシステム時刻が継続されているものとして 1 回目のタイムアウトが発生します。2 回目以降は，修正後のシステム時刻に従ってタイムアウトが発生します。

2.13 EJB のリモートインタフェースの呼び出し

この節では，EJB のリモートインタフェースの呼び出しについて説明します。

この節の構成を次の表に示します。

表 2-39 この節の構成（EJB のリモートインタフェースの呼び出し）

分類	タイトル	参照先
解説	EJB のリモートインタフェースでのローカル呼び出しの最適化	2.13.1
	EJB のリモートインタフェースの値の参照渡し	2.13.2
	EJB のリモートインタフェースの通信障害発生時の動作	2.13.3
設定	cosminexus.xml での定義	2.13.4
	実行環境での設定	2.13.5
注意事項	EJB のリモートインタフェースの呼び出しに関する注意事項	2.13.6

注 「実装」および「運用」について，この機能固有の説明はありません。

2.13.1 EJB のリモートインタフェースでのローカル呼び出しの最適化

ここでは，EJB のリモートインタフェースでのローカル呼び出しの最適化について説明します。

EJB のリモートインタフェースに定義されるメソッドの呼び出しは，RMI-IIOP で行われますが，この呼び出しについてローカル呼び出し最適化を適用できます。

なお，EJB のローカルインタフェースに定義されるメソッドの呼び出しについては，RMI-IIOP を利用しない，通常の Java のメソッド呼び出しとなるため，この機能は適用外となります。

リモートインタフェースでのローカル呼び出しの最適化では，ローカル呼び出しを最適化する範囲を選択できます。最適化する範囲は，J2EE サーバのプロパティをカスタマイズして設定します。J2EE サーバの動作設定のカスタマイズについては，「2.13.5　実行環境での設定」を参照してください。

次に，ローカル呼び出し最適化機能の範囲および動作と，J2EE サーバのプロパティ（usrconf.properties）のキーの指定値の対応を次の表に示します。

表 2-40 ローカル呼び出し最適化機能の範囲と動作

項目	ejbserver.rmi.localinvocation.scope キー※の値		
	all	app	none
ローカル呼び出し最適化の範囲	同一 J2EE サーバ内となります。	同一アプリケーション内となります。	範囲はありません。
スレッド構成	Caller と Callee は常に同一スレッドとなります。	同一アプリケーション内でだけ Caller と Callee は同一スレッドとなります。	Caller と Callee は常に別スレッドとなります。
クラスローダ構成	EJB はコンテナクラスローダ（J2EE サーバ単位）でロードされます。	EJB はアプリケーションクラスローダ（アプリケーション単位）でロードされます。	
ローカルトランザクション	J2EE サーバ内で利用できます。	同一アプリケーション内で利用できます。	同一 J2EE コンポーネント内で利用できます。

注※
usrconf.properties に指定するキーです。

2.13.2 EJB のリモートインタフェースの値の参照渡し

通常，リモートインタフェースを持つ EJB メソッドを呼び出すとき，引数や戻り値をコピーして値を渡しますが（pass by value），引数や戻り値を参照で返すこともできます（pass by reference）。値を参照で渡す場合は，値をコピーして渡す場合より，負荷の軽減が図れます。

ただし，値を参照で渡す場合は，引数や戻り値を直接参照するため，引数および戻り値の変更や，参照渡しをするクライアントとアプリケーションの配置には注意が必要です。

java.io.Serializable インタフェースを実装したオブジェクトを，メソッドの引数や戻り値に定義している場合，EJB のリモートインタフェースの値の参照渡しを適用することで，負荷の軽減を期待できます。オブジェクトの数やサイズが大きければより効果を期待できます。

設定方法には次の二つがあります。どちらか一つの方法で設定していれば，値の参照渡しが有効になります。

● **EJB 単位で設定する方法**

EJB 単位で機能の有効 / 無効を，Session Bean または Entity Bean の属性として定義します。

● **J2EE サーバ単位で設定する方法**

J2EE サーバ単位で機能の有効 / 無効を，J2EE サーバのプロパティとして，一括して定義します。

2.13.3 EJB のリモートインタフェースの通信障害発生時の動作

EJB クライアントからリモートインタフェースとして定義された EJB メソッドが呼び出しを実施している際に通信障害が発生したときのクライアント側の動作を次のどちらかから選択できます。

- コネクションを再接続して，リクエストを再送信する
- コネクションの再接続も，リクエストの再送信もしない

この機能を使用するための設定は，J2EE サーバまたは EJB クライアントアプリケーションのプロパティとして設定します。

なお，EJB クライアントアプリケーションの場合は，API（java.lang.System クラスの setProperty メソッドなど）で設定することもできます。java.lang.System.setProperty メソッドで定義する場合は，EJB クライアントアプリケーションのプロセス起動後，最初に Enterprise Bean のメソッドを呼び出す前に定義してください。

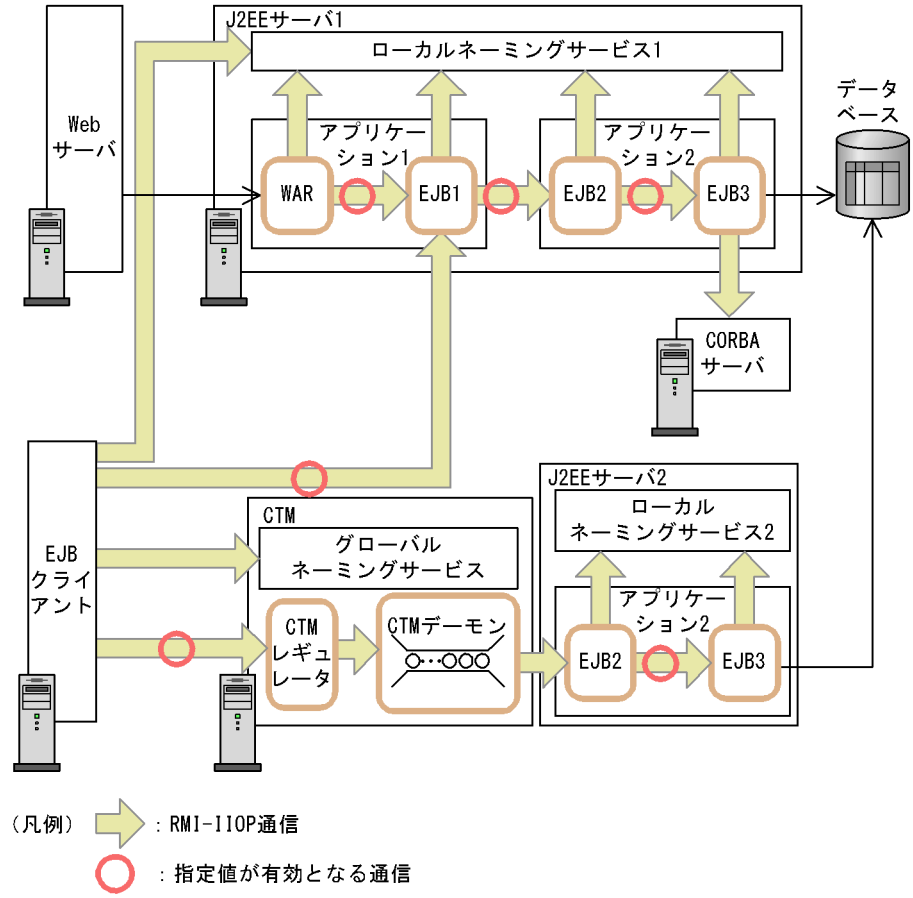
（1） 設定が有効になる通信

通信障害発生時の動作は，リモートインタフェースとして定義された EJB メソッドの呼び出しで通信障害が起きた場合に有効になります。EJB メソッドの呼び出しとは，次の呼び出しを指します。

- Web アプリケーションからの EJB の呼び出し
- EJB クライアントからの EJB の呼び出し
- EJB からの EJB の呼び出し

設定が有効となる通信を次の図に示します。

図 2-31　設定が有効になる通信



なお，次の場合は無効になるので注意してください。

- ローカルインタフェースとして定義された EJB メソッドの呼び出し
- ローカル呼び出し最適化が有効となる範囲での，リモートインタフェースとして定義された EJB メソッドの呼び出し
- ネーミングサービスの呼び出し

(2) 推奨する設定

システムの形態によって，次のように設定することをお勧めします。

検索および参照系のシステムの場合

コネクションを再接続して，リクエストを再送信する設定にすることをお勧めします。これによって，リクエストを失敗させることなく，結果を取得できるようになります。

更新系のシステムの場合

コネクションの再接続も，リクエストの再送信もしない設定にすることをお勧めし

ます。更新系のシステムの場合，コネクションの再接続およびリクエストの再送信ありの設定をすると，リクエストの二重送信をするおそれがあります。

2.13.4 cosminexus.xml での定義

EJB のリモートインタフェース呼び出し機能のうち，どの Enterprise Bean で EJB のリモートインタフェースの参照渡し機能を有効にするかの設定は，cosminexus.xml で定義します。

定義は，cosminexus.xml の <ejb-jar> タグ内に指定します。設定するタグは，設定対象になる Enterprise Bean の種類ごとに異なります。

cosminexus.xml での EJB コンテナでのタイムアウトの定義を次に示します。

指定するタグ

Session Bean の場合

<session>-<pass-by-reference> タグ

Entity Bean の場合

<entity>-<pass-by-reference> タグ

指定内容

Enterprise Bean 単位で EJB のリモートインタフェースでの値の参照渡しを有効にするかどうかを指定します。

2.13.5 実行環境での設定

EJB のリモートインタフェース呼び出し機能のうち，次の機能の設定は，J2EE サーバで設定する必要があります。

- EJB のリモートインタフェースのローカル呼び出し最適化機能の範囲
- EJB のリモートインタフェースの参照渡し機能を有効にするかどうかの設定
- EJB のリモートインタフェースでの通信障害発生時の EJB クライアントの動作※

注※

EJB クライアントの形態が EJB クライアントアプリケーションの場合は，EJB クライアントアプリケーションのプロパティで設定します。

また，どの Enterprise Bean で EJB のリモートインタフェースの参照渡し機能を有効にするかの設定は，J2EE アプリケーションで設定できます。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合に参照してください。

(1) J2EE サーバの設定

J2EE サーバの設定は，簡易構築定義ファイルで実施します。簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の <configuration> タグ内に指定します。

簡易構築定義ファイルでの EJB のリモートインタフェース呼び出し機能の定義について次の表に示します。

表 2-41 簡易構築定義ファイルでの EJB のリモートインタフェース呼び出し機能の定義

項目	指定するパラメタ	設定内容
ローカル呼び出し最適化機能の範囲	ejbserver.rmi.localinvocation.scope	EJB のリモートインタフェースでのローカル呼び出しの最適化の範囲を指定します。
リモートインタフェースの参照渡し機能	ejbserver.rmi.passbyreference ※	リモートインタフェースの参照渡し機能を有効にするかどうかを指定します。
リモートインタフェースでの通信障害発生時の EJB クライアントの動作	ejbserver.container.rebindpolicy	指定先の J2EE サーバがほかの J2EE サーバのクライアントである場合に，EJB クライアント側でのコネクションの再接続動作とリクエストの再送動作を指定します。

注※

J2EE アプリケーションでは，Enterprise Bean ごとに参照渡し機能を有効にするかどうかを設定できます。J2EE サーバまたは Enterprise Bean のどちらかで有効を指定していれば，参照渡し機能は有効になります。

(2) EJB クライアントアプリケーションの設定

EJB クライアントの形態が EJB クライアントアプリケーションの場合は，EJB のリモートインタフェースでの通信障害発生時の EJB クライアントの動作を EJB クライアントアプリケーションのプロパティで設定します。

指定するキー

ejbserver.container.rebindpolicy キー

設定内容

EJB クライアント側でのコネクションの再接続動作とリクエストの再送動作を指定します。

(3) J2EE アプリケーションの設定

Enterprise Bean ごとの EJB のリモートインタフェースの参照渡し機能を有効にするかどうかの設定は，実行環境で設定することもできます。J2EE サーバにインポートした J2EE アプリケーションに設定します。cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ実行してください。

実行環境での J2EE アプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。リファレンスマッピングの定義には，次の属性ファイルを使用します。

表 2-42 Enterprise Bean のトランザクションの管理方法の定義に使用する属性ファイル

設定対象	属性ファイル
Session Bean	Session Bean 属性ファイル
Entity Bean	Entity Bean 属性ファイル

属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。cosminexus.xml での定義については，「2.13.4 cosminexus.xml での定義」を参照してください。

2.13.6 EJB のリモートインタフェースの呼び出しに関する注意事項

EJB のリモートインタフェースの呼び出しに関する注意事項について説明します。

(1) ローカル呼び出しの最適化適用時の注意事項

● 同一アプリケーション内での処理で，EJB のリモートインタフェースのローカル呼び出し最適化を適用する場合（usrconf.properties に「ejbserver.rmi.localinvocation.scope=app」を指定する場合），次のプロバイダ URL には同じホストを指定してください。

- J2EE サーバで使用するプロバイダ URL のホスト
- J2EE アプリケーションから使用するプロバイダ URL ホスト

ホストの指定が異なると，ローカル呼び出しの最適化が行われません。
また，同じホストを指定している場合でも，ホスト名に指定した文字列が異なると，ローカル呼び出しの最適化は行われません。例えば，大文字と小文字の違いや，IP アドレス指定とホスト名指定の違いなどがあると，ローカル呼び出しの最適化は行われません。

● J2EE サーバが使用するプロバイダ URL は次の優先順位で決定されます。

ejbserver.naming.startupMode=inprocess の場合

1. vbroker.se.iiop_tp.host プロパティの値
2. InetAddress.getLocalHost().getHostName() の値

ejbserver.naming.startupMode=automatic の場合

1. InetAddress.getLocalHost().getHostName() の値

ejbserver.naming.startupMode=manual の場合

1. ejbserver.naming.host プロパティの値
2. InetAddress.getLocalHost().getHostName() の値

● J2EE アプリケーションから使用するプロバイダ URL は次の優先順位で決定されます。

1. ネーミング切り替え機能使用時に lookup に渡す引数部分
2. InitialContext 生成時に指定する java.naming.provider.url プロパティ
3. J2EE サーバが使用するプロバイダ URL

(2) EJB のリモートインタフェースの通信障害発生時の動作設定時の注意事項

システムプロパティの ejbserver.container.rebindpolicy キーで "NO_RECONNECT"（再接続なし／再送なし）を選択した場合，通信障害によってコネクションが切断されると，再接続が抑止されているため該当するオブジェクトリファレンスは再利用できなくなります。このため，EJB クライアントでは，次回 Enterprise Bean のメソッド呼び出しをする場合には，EJB ホームオブジェクトのときは lookup メソッド，EJB オブジェクトのときは create メソッドを再実行したあと，メソッド呼び出しを実行してください。なお，Enterprise Bean のメソッド呼び出し中にコネクションが切断された場合，メソッドは次のどちらかの例外を送出します。

java.rmi.RemoteException
: detail フィールドが org.omg.CORBA.REBIND のインスタンス

java.rmi.MarshalException
: detail フィールドが org.omg.CORBA.COMM_FAILURE のインスタンス

これらの例外をキャッチするクライアントのコーディング例を次に示します。

```
try {
     //JNDI.lookup()
     //EJBHome.create()
     //EJBObject.invoke()
    } catch (java.rmi.MarshalException e) {
      if (e.detail instanceof org.omg.CORBA.COMM_FAILURE) {
        //通信障害に対応する処理
      }
    } catch (java.rmi.RemoteException e) {
      if (e.detail instanceof org.omg.CORBA.REBIND) {
        //通信障害に対応する処理
      }
    }
```

(3) ネーミング管理機能でのキャッシング機能を有効にしている場合の注意事項

ネーミング管理機能でのキャッシング機能を有効にしている場合，コネクション切断後に lookup メソッドを実行すると，キャッシュ上の無効なオブジェクトリファレンスが取得されます。この場合，取得したオブジェクトリファレンスを使用して javax.rmi.PortableRemoteObject.narrow メソッドや create メソッドを実行したときに，CORBA 例外（org.omg.CORBA.OBJECT_NOT_EXIST など）が発生することがあります。

コネクション切断後の無効なキャッシュはクリアしてください。手順については，マ

ニュアル「Cosminexus 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「2.6.2　ネーミングで使用したキャッシュのクリア」を参照してください。

2.14 EJB コンテナの通信ポートと IP アドレスの固定（TPBroker のオプション）

この節では，EJB コンテナの通信ポートと IP アドレスの固定（TPBroker のオプション）について説明します。

J2EE サーバ経由で Cosminexus TPBroker のオプションを定義することで，EJB コンテナの通信ポート，および IP アドレスを固定して運用できます。使用するポートを最小限にしてシステムのセキュリティを高める場合は，ポート固定の設定を強く推奨します。Cosminexus TPBroker のオプションの詳細については，マニュアル「TPBroker ユーザーズガイド」を参照してください。

この節の構成を次の表に示します。

表 2-43 この節の構成（EJB コンテナの通信ポートと IP アドレスの固定（TPBroker のオプション））

分類	タイトル	参照先
解説	通信ポートの固定	2.14.1
	IP アドレスの固定	2.14.2
設定	実行環境での設定	2.14.3

注 「実装」,「運用」および「注意事項」について，この機能固有の説明はありません。

2.14.1 通信ポートの固定

EJB コンテナの通信ポートは，デフォルトでは Cosminexus TPBroker によってランダムな値が割り当てられています。

これに対して，Cosminexus TPBroker のオプションとして任意の値を指定して，J2EE サーバ単位で通信ポートを固定できます。ポート番号については，ほかのプログラムなどと重複しないようにしてください。

2.14.2 IP アドレスの固定

EJB コンテナの IP アドレスは，デフォルトでは Cosminexus TPBroker によって実行環境のマシンのシステムから取得して割り当てられています。

これに対して，Cosminexus TPBroker のオプションとして任意の値を指定して，J2EE サーバ単位で IP アドレスを固定できます。

2.14.3 実行環境での設定

EJB コンテナの通信ポートと IP アドレスを固定する場合，J2EE サーバの設定が必要です。

J2EE サーバの設定は，簡易構築定義ファイルで実施します。定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の <configuration> タグ内に指定します。

EJB コンテナの通信ポートと IP アドレスを固定するための簡易構築定義ファイルでの定義について次の表に示します。

表 2-44 簡易構築定義ファイルでの EJB コンテナの通信ポートと IP アドレスを固定するための定義

項目	指定するパラメタ	設定内容
EJB コンテナの通信ポート	vbroker.se.iiop_tp.scm.iiop_tp.listener.port	EJB コンテナの通信ポートを指定します。
IP アドレスまたはホスト名を固定するかどうか	vbroker.se.iiop_tp.host	EJB コンテナの使用する IP アドレスまたはホスト名を固定するかどうかを指定します。

2.15 インターセプタの使用

この節では，インターセプタの使用について説明します。

アプリケーションサーバでは，DD，属性ファイル，またはアノテーション（デフォルトインターセプタを除く）で設定できます。

この節の構成を次の表に示します。

表 2-45 この節の構成（インターセプタの使用）

分類	タイトル	参照先
解説	インターセプタの使用の概要	2.15.1
	上位レベルインターセプタの呼び出し抑止	2.15.3
	インターセプタの実行順序	2.15.4
実装	アノテーションまたは DD での定義	2.15.2
設定	実行環境での設定	2.15.5
注意事項	インターセプタに関する注意事項	2.15.6

注 「運用」について，この機能固有の説明はありません。

2.15.1 インターセプタの使用の概要

アプリケーションサーバでは，次のインターセプタを使用できます。

● デフォルトインターセプタ
EJB-JAR に含まれるすべてのコンポーネントに適用されるインターセプタです。クラスレベルインターセプタ，およびメソッドレベルインターセプタの上位レベルのインターセプタとなります。DD または属性ファイルを使用して設定します。

● クラスレベルインターセプタ
指定したクラスに適用されるインターセプタです。メソッドレベルインターセプタの上位レベルのインターセプタとなります。アノテーション，DD，または属性ファイルを使用して設定します。

● メソッドレベルインターセプタ
指定したビジネスメソッドに適用されるインターセプタです。アノテーション，DD，または属性ファイルを使用して設定します。

2.15.2 アノテーションまたは DD での定義

インターセプタは，アノテーションで指定するか，DD または属性ファイルを使用して EJB-JAR の属性として設定します。

(1) デフォルトインターセプタの定義

デフォルトインターセプタは，EJB-JAR に含まれるすべてのコンポーネントに適用されるインターセプタです。ここでは，DD を使用したデフォルトインターセプタを設定する方法，および実行環境で EJB-JAR 属性ファイルを使用してデフォルトインターセプタを設定する方法について説明します。

DD で，<ejb-jar> タグ下の <interceptor-binding> タグを記述することで，デフォルトインターセプタの情報を指定できます。

<interceptor-binding> タグ下に指定する要素を次の表に示します。

表 2-46 DD を使用する場合に <interceptor-binding> タグ下に指定する要素（デフォルトインターセプタ）

タグ名	必須／任意	指定内容
<description>	任意	任意の情報を指定します。
<ejb-name>	必須	"*"（ワイルドカード）を指定します。
<interceptor-class>	必須	要素にインターセプタクラスのクラス名を指定します。

これ以外のタグで指定した要素の値は，反映されません。

デフォルトインターセプタを使用する場合の DD の記述例を次に示します。

```
<ejb-jar>
  …(略)…
  <assembly-descriptor>
   <interceptor-binding>
     <description xml:lang="en">Default Interceptor</description>
     <ejb-name>*</ejb-name>
     <interceptor-class>test.ejb30.MyDefaultIC</interceptor-class>
     <interceptor-class>test.ejb30.MyDefaultIC2</interceptor-class>
   </interceptor-binding>
  </assembly-descriptor>
  …(略)…
</ejb-jar>
```

この記述例では，デフォルトインターセプタクラスとして，「test.ejb30.MyDefaultIC」と「test.ejb30.MyDefaultIC2」の二つのクラスを指定しています。

● インターセプタクラスの指定規則

<interceptor-class> タグでのクラス名の指定方法は，EJB 3.0 仕様に準じます。次の規則に従って指定してください。

- 一つの <interceptor-class> タグに指定できるインターセプタクラスのクラス名は一つです。
- <interceptor-class> タグは複数記述できます。<interceptor-class> タグを複数記述し

た場合は，記述した順序でインターセプタが呼び出されます。

- <interceptor-class> タグで指定したインターセプタクラスには，次に示すアノテーションを使用してインターセプタメソッドを指定できます。
  - @AroundInvoke
  - @PostConstruct
  - @PreDestroy

  アノテーションについての詳細は，マニュアル「Cosminexus アプリケーションサーバ リファレンス API 編」の「2. アプリケーションサーバが対応しているアノテーションおよび Dependency Injection」を参照してください。

(2) クラスレベルインターセプタの定義

クラスレベルインターセプタは，指定したクラスに適用されるインターセプタです。ここでは，DD を使用したクラスレベルインターセプタを設定する方法，および実行環境で EJB-JAR 属性ファイルを使用してクラスレベルインターセプタを設定する方法について説明します。

DD で，<ejb-jar> タグ下の <interceptor-binding> タグを記述することで，クラスレベルインターセプタの情報を指定できます。

<interceptor-binding> タグ下に指定する要素を次の表に示します。

表 2-47 DD を使用する場合に <interceptor-binding> タグ下に指定する要素（クラスレベルインターセプタ）

タグ名	必須／任意	指定内容
<ejb-name>	必須	EJB 名を指定します。
<interceptor-class>	必須	要素にインターセプタクラスのクラス名を指定します。

なお，<method> タグには要素を指定しません。

(3) メソッドレベルインターセプタの定義

メソッドレベルインターセプタは，指定したビジネスメソッドに適用されるインターセプタです。ここでは，DD を使用したメソッドレベルインターセプタを設定する方法，および実行環境で EJB-JAR 属性ファイルを使用してメソッドレベルインターセプタを設定する方法について説明します。

DD で，<ejb-jar> タグ下の <interceptor-binding> タグを記述することで，メソッドレベルインターセプタの情報を指定できます。

<interceptor-binding> タグ下に指定する要素を次の表に示します。

表 2-48 DD を使用する場合に <interceptor-binding> タグ下に指定する要素（メソッドレベルインターセプタ）

タグ名	必須／任意	指定内容
<ejb-name>	必須	EJB 名を指定します。
<interceptor-class>	必須	要素にインターセプタクラスのクラス名を指定します。
<method-name>	必須	要素にメソッド名を指定します。
<method-params>	任意	要素にメソッドの引数リストを指定します。

<method-name> タグにビジネスメソッドを指定し，<method-params> タグの指定値を省略した場合，または <method-params> タグに引数リストを指定した場合の，メソッドレベルインターセプタが定義されるビジネスメソッドの範囲を次の表に示します。なお，<method-name> タグに "*"（ワイルドカード）を指定した場合，メソッドレベルインターセプタの定義は使用されません。

表 2-49 メソッドレベルインターセプタが定義されるビジネスメソッドの範囲

<method-params> タグの指定値	メソッドレベルインターセプタが定義されるビジネスメソッドの範囲
省略する	メソッド名が完全一致したすべてのビジネスメソッドに対してインターセプタが定義されます。
引数リストを指定する	メソッド名と引数リストが完全一致したビジネスメソッドに対してインターセプタが定義されます。 引数リストを指定した場合，アノテーションで指定した情報を DD で上書きできます。

● メソッドレベルインターセプタの適用規則

ビジネスメソッドが実行される際に使用するメソッドレベルインターセプタの定義は，次の順序で決定されます。

1. 実行するビジネスメソッドと，<method-name> タグに指定したメソッド名と <method-params> タグに指定した引数リストが完全一致する定義があれば，そのメソッドレベルインターセプタの定義が使用されます。
2. 実行するビジネスメソッドと <method-name> タグに指定したメソッド名が一致し，<method-params> タグに引数リストを指定しない定義があれば，そのメソッドレベルインターセプタの定義が使用されます。
3. 実行するビジネスメソッドと <method-name> タグに指定したメソッド名が一致するメソッドレベルインターセプタの定義がない場合，メソッドレベルインターセプタの定義は使用されません。

なお，実行するビジネスメソッドが，<method-name> タグに指定したメソッド名と <method-params> タグに指定した引数リストが完全一致する定義，および <method-name> タグに指定したメソッド名が一致し，<method-params> タグに引数リ

ストを指定しない定義の両方に該当する場合は，メソッド名と引数リストが完全一致する定義が使用されます。

また，アノテーションで指定されたメソッドレベルインターセプタは，メソッド名と引数リストが完全一致する定義として扱われます。アノテーションで指定されたメソッドレベルインターセプタを DD で上書きする場合は，メソッド名と引数リストが完全一致する定義で DD に記載する必要があります。

2.15.3 上位レベルインターセプタの呼び出し抑止

クラスレベルインターセプタ，メソッドレベルインターセプタでは，アノテーション，または DD で上位レベルのインターセプタの実行を抑止できます。

インターセプタの実行を抑止できる範囲を次の表に示します。

表 2-50 上位レベルインターセプタの実行を抑止できる範囲

インターセプタの種別	抑止する対象		
	デフォルトインターセプタ	クラスレベルインターセプタ	メソッドレベルインターセプタ
デフォルトインターセプタ	×	×	－
クラスレベルインターセプタ	○※	×	－
メソッドレベルインターセプタ	○	○	－

（凡例）
○：上位レベルのインターセプタの呼び出しを抑止できる。
×：上位レベルに該当しないため，呼び出し抑止の定義をしても無視される。
－：上位レベルのインターセプタの呼び出しの定義はできない。

注※
クラスレベルインターセプタでデフォルトインターセプタの呼び出し抑止の定義が指定された場合，メソッドレベルインターセプタでの定義に関係なく，デフォルトインターセプタの呼び出しを抑止します。

2.15.4 インターセプタの実行順序

インターセプタの実行順序は，デフォルトでは次の規則に従って決定されます。

● EJB 3.0 仕様の実行順序

EJB 3.0 仕様で次の実行順序が規定されています。

1. デフォルトインターセプタ
2. クラスレベルインターセプタ
3. メソッドレベルインターセプタ
4. Bean クラスに指定されたインターセプタメソッド

● 同一レベルのインターセプタが複数ある場合の規則
同一レベルのインターセプタが複数ある場合，アノテーション，または DD で記述された順番に実行されます。

● インターセプタクラスに親クラスがあり，親クラスにインターセプタメソッドが定義されている場合
親クラスから実行されます。

● 上位レベルインターセプタの呼び出し抑止の定義による規則
上位レベルのインターセプタの呼び出しを抑止する定義を指定した場合，指定されたレベルのインターセプタは呼び出されません。

● <interceptor-order> タグでの順序制御
次のビジネスメソッドを実行する場合，DD または属性ファイルの <interceptor-order> タグを定義したインターセプタよりも上位のインターセプタは呼び出されません。
- <interceptor-order> タグで実行順序を定義したクラスレベルインターセプタが適用されている，Enterprise Bean クラスのビジネスメソッド
- <interceptor-order> タグで実行順序を定義したメソッドレベルインターセプタが適用されているビジネスメソッド

<interceptor-order> タグを使用しているインターセプタの中で，いちばん低いレベルのインターセプタの定義とそれより下位のレベルのインターセプタが実行されます。

なお，ここで示したデフォルトのインターセプタの実行順序は，次の方法で変更できます。

- @ExcludeDefaultInterceptors アノテーションまたは DD の <exclude-default-interceptors> タグでデフォルトインターセプタクラスの実行を対象外にできます。
- @ExcludeClassInterceptors アノテーションまたは DD の <exclude-class-interceptors> タグでクラスレベルインターセプタクラスの実行を対象外にできます。
- すべてのレベルのインターセプタクラスについて，DD の <interceptor-order> タグを記述することで順番を入れ替えることができます。

これらの規則と，DD によるアノテーションの上書きの規則を組み合わせた実行順序を (1) ～ (4) で説明します。DD によるアノテーションの上書きについては，マニュアル「Cosminexus 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「11.6.2　DD によるアノテーションの上書き」を参照してください。

(1) デフォルトインターセプタ，およびクラスレベルインターセプタの呼び出し抑止の定義が指定されている場合

デフォルトインターセプタ，およびクラスレベルインターセプタの呼び出し抑止の定義が指定されている場合の，インターセプタの実行順序を次の表に示します。この表では，

各項番の「<interceptor-order> の使用」に○×で示した組み合わせの場合の実行順序を，「インターセプタの実行順序」の DD またはアノテーションの表記の前にある数字で示しています。

表 2-51　インターセプタの実行順序（デフォルトインターセプタ，およびクラスレベルインターセプタの呼び出し抑止の定義が指定されている場合）

項番	<interceptor-order> の使用			インターセプタの実行順序			
	デフォルト	クラスレベル	メソッドレベル	デフォルト	クラスレベル	メソッドレベル	インターセプタメソッド
1	○	○	○	－	－	1. DD	2. アノテーション
2	○	○	×	－	－	1. アノテーション 2. DD	3. アノテーション
3	○	×	○	－	－	1. DD	2. アノテーション
4	○	×	×	－	－	1. アノテーション 2. DD	3. アノテーション
5	×	○	○	－	－	1. DD	2. アノテーション
6	×	○	×	－	－	1. アノテーション 2. DD	3. アノテーション
7	×	×	○	－	－	1. DD	2. アノテーション
8	×	×	×	－	－	1. アノテーション 2. DD	3. アノテーション

（凡例）

デフォルト：デフォルトインターセプタ
クラスレベル：クラスレベルインターセプタ
メソッドレベル：メソッドレベルインターセプタ
インターセプタメソッド：Bean クラスに指定されたインターセプタメソッド
○：<interceptor-order> タグを使用して実行順序を指定している。
×：<interceptor-order> タグを使用して実行順序を指定していない。
－：実行されない。
DD：DD で指定されたインターセプタが実行される。
アノテーション：アノテーションで指定されたインターセプタが実行される。

(2) デフォルトインターセプタの呼び出し抑止の定義が指定されている場合

デフォルトインターセプタの呼び出し抑止の定義が指定されている場合の，インターセ

プタの実行順序を次の表に示します。この表では，各項番の「<interceptor-order> の使用」に○×で示した組み合わせの場合の実行順序を，「インターセプタの実行順序」の DD またはアノテーションの表記の前にある数字で示しています。

表 2-52　インターセプタの実行順序（デフォルトインターセプタの呼び出し抑止の定義が指定されている場合）

項番	<interceptor-order> の使用			インターセプタの実行順序			
	デフォルト	クラスレベル	メソッドレベル	デフォルト	クラスレベル	メソッドレベル	インターセプタメソッド
1	○	○	○	－	－	1. DD	2. アノテーション
2	○	○	×	－	1. DD	2. アノテーション 3. DD	4. アノテーション
3	○	×	○	－	－	1. DD	2. アノテーション
4	○	×	×	－	1. アノテーション 2. DD	3. アノテーション 4. DD	5. アノテーション
5	×	○	○	－	－	1. DD	2. アノテーション
6	×	○	×	－	1. DD	2. アノテーション 3. DD	4. アノテーション
7	×	×	○	－	－	1. DD	2. アノテーション
8	×	×	×	－	1. アノテーション 2. DD	3. アノテーション 4. DD	5. アノテーション

（凡例）

デフォルト：デフォルトインターセプタ
クラスレベル：クラスレベルインターセプタ
メソッドレベル：メソッドレベルインターセプタ
インターセプタメソッド：Bean クラスに指定されたインターセプタメソッド
○：<interceptor-order> タグを使用して実行順序を指定している。
×：<interceptor-order> タグを使用して実行順序を指定していない。
－：実行されない。
DD：DD で指定されたインターセプタが実行される。
アノテーション：アノテーションで指定されたインターセプタが実行される。

(3) クラスレベルインターセプタの呼び出し抑止の定義が指定されている場合

クラスレベルインターセプタの呼び出し抑止の定義が指定されている場合の，インターセプタの実行順序を次の表に示します。この表では，各項番の「<interceptor-order> の使用」に○×で示した組み合わせの場合の実行順序を，「インターセプタの実行順序」の DD またはアノテーションの表記の前にある数字で示しています。

表 2-53 インターセプタの実行順序（クラスレベルインターセプタの呼び出し抑止の定義が指定されている場合）

項番	<interceptor-order> の使用			インターセプタの実行順序			
	デフォルト	クラスレベル	メソッドレベル	デフォルト	クラスレベル	メソッドレベル	インターセプタメソッド
1	○	○	○	－	－	1. DD	2. アノテーション
2	○	○	×	1. DD	－	2. アノテーション 3. DD	4. アノテーション
3	○	×	○	－	－	1. DD	2. アノテーション
4	○	×	×	1. DD	－	2. アノテーション 3. DD	4. アノテーション
5	×	○	○	－	－	1. DD	2. アノテーション
6	×	○	×	1. DD	－	2. アノテーション 3. DD	4. アノテーション
7	×	×	○	－	－	1. DD	2. アノテーション
8	×	×	×	1. DD	－	2. アノテーション 3. DD	4. アノテーション

（凡例）

デフォルト：デフォルトインターセプタ
クラスレベル：クラスレベルインターセプタ
メソッドレベル：メソッドレベルインターセプタ
インターセプタメソッド：Bean クラスに指定されたインターセプタメソッド
○：<interceptor-order> タグを使用して実行順序を指定している。
×：<interceptor-order> タグを使用して実行順序を指定していない。
－：実行されない。
DD：DD で指定されたインターセプタが実行される。

アノテーション：アノテーションで指定されたインターセプタが実行される。

(4) 上位レベルインターセプタの呼び出し抑止の定義が指定されていない場合

上位レベルインターセプタの呼び出し抑止の定義が指定されていない場合の，インターセプタの実行順序を次の表に示します。この表では，各項番の「<interceptor-order> の使用」に○×で示した組み合わせの場合の実行順序を，「インターセプタの実行順序」の DD またはアノテーションの表記の前にある数字で示しています。

表 2-54 インターセプタの実行順序（上位レベルインターセプタの呼び出し抑止の定義が指定されていない場合）

項番	<interceptor-order> の使用			インターセプタの実行順序			
	デフォルト	クラスレベル	メソッドレベル	デフォルト	クラスレベル	メソッドレベル	インターセプタメソッド
1	○	○	○	－	－	1. DD	2. アノテーション
2	○	○	×	－	1. DD	2. アノテーション 3. DD	4. アノテーション
3	○	×	○	－	－	1. DD	2. アノテーション
4	○	×	×	1. DD	2. アノテーション 3. DD	4. アノテーション 5. DD	6. アノテーション
5	×	○	○	－	－	1. DD	2. アノテーション
6	×	○	×	－	1. DD	2. アノテーション 3. DD	4. アノテーション
7	×	×	○	－	－	1. DD	2. アノテーション
8	×	×	×	1. DD	2. アノテーション 3. DD	4. アノテーション 5. DD	6. アノテーション

（凡例）

デフォルト：デフォルトインターセプタ
クラスレベル：クラスレベルインターセプタ
メソッドレベル：メソッドレベルインターセプタ
インターセプタメソッド：Bean クラスに指定されたインターセプタメソッド
○：<interceptor-order> タグを使用して実行順序を指定している。
×：<interceptor-order> タグを使用して実行順序を指定していない。
－：実行されない。

DD：DD で指定されたインターセプタが実行される。
アノテーション：アノテーションで指定されたインターセプタが実行される。

2.15.5 実行環境での設定

インターセプタは，実行環境で設定することもできます。実行環境で設定する場合，J2EE サーバにインポートした J2EE アプリケーションに設定します。

実行環境での J2EE アプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。インターセプタの定義には，EJB-JAR 属性ファイルを使用します。なお，J2EE アプリケーションに含まれない EJB-JAR ファイルに対して，サーバ管理コマンド（cjsetresprop -type ejb コマンド）を使用して設定することはできません。

EJB-JAR 属性ファイルで指定するタグは，DD と対応しています。DD（ejb-jar.xml）での定義については，「2.15.2 アノテーションまたは DD での定義」を参照してください。

2.15.6 インターセプタに関する注意事項

- <ejb-name> タグに "*"（ワイルドカード）を指定した <interceptor-binding> タグが複数ある場合は，いちばん上に記述された <interceptor-binding> タグの内容が有効になります。2 番目以降の内容は設定されません。
- 属性ファイルの場合，<ejb-name> タグの値と，<named-method> タグおよびその配下の要素すべてが一致する <interceptor-binding> タグが複数存在する場合，いちばん上に記述された <interceptor-binding> タグの内容が有効になります。2 番目以降の内容は設定されません。
- DD の場合，<ejb-name> タグの値と，<method> タグおよびその配下の要素すべてが一致する <interceptor-binding> タグが複数存在する場合，いちばん上に記述された <interceptor-binding> タグの内容が有効になります。2 番目以降の内容は設定されません。

3 EJB クライアント

この章では，EJB クライアントで使用できる機能について説明します。EJB クライアントは，Enterprise Bean を呼び出すクライアントプログラムです。

3.1 この章の構成

3.2 EJB クライアントで使用できる機能

3.3 EJB クライアントアプリケーションの開始

3.4 Enterprise Bean の呼び出し

3.5 EJB クライアントアプリケーションでのトランザクションの実装

3.6 EJB クライアントアプリケーションでのセキュリティの実装

3.7 RMI-IIOP スタブ，インタフェースの取得

3.8 EJB クライアントアプリケーションのシステムログ出力

3.1 この章の構成

EJB クライアントとは，J2EE サーバ上の EJB コンテナで実行されている Enterprise Bean を呼び出すクライアントプログラムです。

EJB クライアントには，次の種類があります。

- EJB クライアントアプリケーション
- サーブレットまたは JSP などの Web アプリケーション
- ほかの Enterprise Bean

EJB クライアントアプリケーションとは，J2EE サーバ上で実行される Enterprise Bean を呼び出すクライアントアプリケーションのことです。

EJB クライアントの機能と参照先を次の表に示します。

表 3-1 EJB クライアントの機能と参照先

機能	参照先
EJB クライアントで使用できる機能	3.2
EJB クライアントアプリケーションの開始	3.3
Enterprise Bean の呼び出し	3.4
EJB クライアントアプリケーションでのトランザクションの実装	3.5
EJB クライアントアプリケーションでのセキュリティの実装	3.6
RMI-IIOP スタブ，インタフェースの取得	3.7
EJB クライアントアプリケーションのシステムログ出力	3.8

なお，この章では，EJB クライアントアプリケーションの機能を中心に説明します。

3.2　EJB クライアントで使用できる機能

EJB クライアントで使用できる機能について，次の表に示します。それぞれの機能の詳細については，参照先の説明を参照してください。なお，「参照先マニュアル」に示したマニュアル名の「Cosminexus アプリケーションサーバ V8」は省略しています。

表 3-2　EJB クライアントで使用できる機能

分類		機能概要	参照先マニュアル	参照個所
JNDI	基本機能	EJB ホームオブジェクトリファレンスおよびビジネスインタフェースのリファレンスの検索，取得ができます。	機能解説 基本・開発編（コンテナ共通機能）	2 章
	拡張機能	複数のネーミングサービスと J2EE サーバで構成されるシステムで，EJB クライアントからのルックアップをラウンドロビンで実行できます。これによって，負荷分散を実現できます。		
		ネーミングサービスからルックアップしたオブジェクトをメモリ上に保持（キャッシュ）できます。キャッシュの利用によって，ネーミングサービスへのアクセスの性能上のコストを削減できます。		
EJB		EJB コンテナで実行されている Enterprise Bean を呼び出せます。	このマニュアル	2.2，3.4※，3.7※
		EJB 呼び出し実行時に通信障害が発生した場合に，送信動作を選択できます。	このマニュアル	2.13.3
トランザクション		EJB クライアントでトランザクションを開始・決着できます。	機能解説 基本・開発編（コンテナ共通機能）	3 章
Security		J2EE サーバで定義されたユーザとパスワードを使用してユーザ認証ができます。EJB クライアントからログインして，セキュリティロールが設定された EJB のビジネスメソッドを呼び出せます。	このマニュアル	3.6※
			機能解説 基本・開発編（コンテナ共通機能）	9 章
そのほか		EJB クライアントとネーミングサービス間，および EJB クライアントと J2EE サーバ間の通信で，通信タイムアウトを設定できます。	このマニュアル	2.11
		EJB クライアントの性能解析トレースを出力できます。	機能解説 保守／移行／互換編	4.6

注※　アプリケーションの実装方法について説明しています。

EJB クライアントアプリケーションで使用できる拡張機能について，次の表に示します。なお，それぞれの機能の詳細については，参照先の説明を参照してください。なお，EJB クライアントアプリケーションでは，データソース（JDBC）およびコネクタ（Connector）は使用できません。

表 3-3　EJB クライアントアプリケーションで使用できる拡張機能

分類	機能概要	機能についての参照先
トランザクション	EJB クライアントアプリケーションで UserTransaction を取得し，トランザクションを開始・決着できます。 UserTransaction の取得方法には次の 2 種類の方法があります。 1. UserTransactionFactory クラスを使用する方法 2. ルックアップを使用する方法 なお，Cosminexus では，1. の方法を推奨しています。	3.5
そのほか	EJB クライアントのログを出力できます。	3.8 ※
	EJB クライアントアプリケーションをコマンド（cjclstartap コマンド）で開始できます。	3.3.1

注※　このマニュアルでは，EJB クライアントアプリケーションのシステムログについて説明しています。EJB クライアントアプリケーションが出力するユーザログについては，マニュアル「Cosminexus アプリケーションサーバ 機能解説 拡張編」の「12.　アプリケーションのユーザログ出力」を参照してください。

次の節以降では，EJB クライアントアプリケーションの機能について説明しています。

3.3 EJB クライアントアプリケーションの開始

この節では，EJB クライアントアプリケーション開始について説明します。

EJB クライアントアプリケーションは，コマンドを使用して開始します。

この節の構成を次の表に示します。

表 3-4 この節の構成（EJB クライアントアプリケーションの開始）

分類	タイトル	参照先
解説	EJB クライアントアプリケーション開始に使用するコマンド	3.3.1
	cjclstartap コマンドの場合	3.3.2
	vbj コマンドの場合	3.3.3
設定	EJB クライアントアプリケーションの実行に必要な環境変数の設定	3.3.4
	EJB クライアントアプリケーションのプロパティの設定	3.3.5

注 「実装」,「運用」および「注意事項」について，この機能固有の説明はありません。

この節では，EJB クライアントアプリケーションの開始に使用するコマンド，および使用するコマンドごとの EJB クライアントアプリケーションの開始の流れについて説明します。

3.3.1 EJB クライアントアプリケーション開始に使用するコマンド

EJB クライアントアプリケーションの開始に使用するコマンドを次の表に示します。

表 3-5 EJB クライアントアプリケーションの開始に使用するコマンド

コマンド名	説明
cjclstartap	cjclstartap コマンドは，EJB クライアントアプリケーションの実行に必要なオプションやプロパティを，あらかじめオプション定義ファイルやプロパティファイルに記述しておくことができます。なお，各ファイルでオプションやプロパティの指定を省略した場合には，デフォルト値が適用されてコマンドが実行されます。 Cosminexus では，トラブルシュート機能を強化した cjclstartap コマンドでの EJB クライアントアプリケーションの運用をお勧めします。 また，cjclstartap コマンドを使用すると，Java アプリケーションも開始できます。 コマンドの格納場所を次に示します。 • Windows の場合 <Cosminexus のインストールディレクトリ >¥CC¥client¥bin • UNIX の場合 /opt/Cosminexus/CC/client/bin

コマンド名	説明
vbj	vbj コマンドは，EJB クライアントアプリケーションの実行に必要なオプションやプロパティをコマンドの引数として指定する必要があります。オプションやプロパティの省略はできません。 vbj コマンドは，旧バージョン互換用のコマンドです。従来の方法で vbj コマンドを使用して EJB クライアントアプリケーションを実行したい場合に使用してください。 コマンドの格納場所を次に示します。 • Windows の場合 <Cosminexus のインストールディレクトリ >¥TPB¥bin¥vbj • UNIX の場合 /opt/Cosminexus/TPB/bin/vbj

注意事項

uCosminexus Client を使用して EJB クライアント環境を構築する場合は，格納ディレクトリの「<Cosminexus のインストールディレクトリ >¥CC」を，「<Cosminexus のインストールディレクトリ >¥CCL」と読み替えてください。

それぞれのコマンドによる EJB クライアントアプリケーションの開始の流れについて以降の項で説明します。

3.3.2 cjclstartap コマンドの場合

cjclstartap コマンドによる EJB クライアントアプリケーションの開始の流れを次に示します。

1. **EJB クライアントアプリケーションの実行に必要な環境変数を設定します。**
   EJB クライアントアプリケーションの実行に必要な環境変数のうち，cjclstartap コマンドでの実行に必要な環境変数を設定してください。必要な環境変数については，「3.3.4 EJB クライアントアプリケーションの実行に必要な環境変数の設定」を参照してください。
2. **EJB クライアントアプリケーションのオプション定義ファイル（usrconf.cfg）に，Java のオプションや JAR ファイルのクラスパスを指定します。**

   usrconf.cfg の格納場所

   usrconf.cfg のひな型が次の場所に格納されていますので，このひな型ファイルを任意の場所にコピーして，使用してください。
   Windows の場合
   <Cosminexus のインストールディレクトリ
   >¥CC¥client¥templates¥usrconf.cfg
   UNIX の場合
   /opt/Cosminexus/CC/client/templates/usrconf.cfg

   JavaVM 起動オプションの指定

   usrconf.cfg の add.jvm.arg キーに指定します。指定できるオプションの詳細につ

いては，マニュアル「Cosminexus 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)」の「19. JavaVM 起動オプション」を参照してください。

JAR ファイルのクラスパスの指定

usrconf.cfg の add.class.path キーに指定します。クラスパスへの設定が必要な JAR ファイルについては，「3.7.4 EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定」を参照してください。

3. **EJB クライアントアプリケーションのプロパティファイル（usrconf.properties）に，プロパティを指定します。**

usrconf.properties の格納場所

usrconf.properties のひな型が次の場所に格納されていますので，このひな型ファイルを任意の場所にコピーして，使用してください。

Windows の場合

<Cosminexus のインストールディレクトリ>¥CC¥client¥templates¥usrconf.properties

UNIX の場合

/opt/Cosminexus/CC/client/templates/usrconf.properties

プロパティの指定

プロパティで指定できる内容については，「3.3.5 EJB クライアントアプリケーションのプロパティの設定」を参照してください。また，必要に応じて，「3.5.3 EJB クライアントアプリケーションでのトランザクション実装時の注意事項」，「3.8 EJB クライアントアプリケーションのシステムログ出力」，およびマニュアル「Cosminexus アプリケーションサーバ 機能解説 拡張編」の「12.10 EJB クライアントアプリケーションのユーザログ出力の設定（cjclstartap コマンドを使用する場合)」を参照してください。

指定できるプロパティの詳細については，マニュアル「Cosminexus 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)」の「15.3 usrconf.properties (Java アプリケーション用ユーザプロパティファイル)」を参照してください。

4. **usrconf.cfg と usrconf.properties を cjclstartap コマンドを実行するカレントディレクトリ以外に格納した場合，環境変数「CJCLUSRCONFDIR」で，usrconf.cfg と usrconf.properties の格納場所の絶対パスを指定します。**

作成した usrconf.cfg と usrconf.properties を同じディレクトリに格納して，そのディレクトリの絶対パスを環境変数「CJCLUSRCONFDIR」で指定してください。なお，usrconf.cfg と usrconf.properties を cjclstartap コマンドを実行するカレントディレクトリに格納した場合はこの作業は必要ありません。手順 5. に進んでください。

5. **cjclstartap コマンドを使用して EJB クライアントアプリケーションを開始します。**

cjclstartap コマンドについては，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjclstartap（Java アプリケーションの開始)」を参照

してください。

注意事項

uCosminexus Client を使用して EJB クライアント環境を構築する場合は，格納ディレクトリの「<Cosminexus のインストールディレクトリ >¥CC」を，「<Cosminexus のインストールディレクトリ >¥CCL」と読み替えてください。

なお，ejb.client.directory.shareable に false が設定されている場合，cjclstartap コマンドを実行すると，コマンドが使用するワークファイルが作成されます。ワークファイルが破損した場合，cjclstartap コマンド，または cjcldumpap コマンドの動作は保証されません。ワークファイルの出力先，およびファイル名を次に示します。

Windows の場合

< カレントディレクトリ >¥.cjclstartap.lock
<ejb.client.log.directory>¥.ejbclientlog.lock
< カレントディレクトリ >¥cjclstartap.pid

UNIX の場合

< カレントディレクトリ >/.cjclstartap.lock
<ejb.client.log.directory>/.ejbclientlog.lock
< カレントディレクトリ >/cjclstartap.pid
< カレントディレクトリ >/.COSMINEXUS_CC_EJBCLIENT_ プロセス ID

参考

cjclstartap コマンドを使用すると，Java アプリケーションも開始できます。cjclstartap コマンドを使用した Java アプリケーションの開始方法については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjclstartap（Java アプリケーションの開始）」を参照してください。

3.3.3 vbj コマンドの場合

vbj コマンドによる EJB クライアントアプリケーションの開始の流れを次に示します。なお，バッチファイルまたはシェルスクリプトファイルを使用して EJB クライアントアプリケーションを実行する場合は，バッチファイルまたはシェルスクリプトファイルにこれらの内容を記述できます。

1. **EJB クライアントアプリケーションの実行に必要な環境変数を設定します。**
   EJB クライアントアプリケーションの実行に必要な環境変数のうち，vbj コマンドでの実行に必要な環境変数を設定してください。必要な環境変数については，「3.3.4 EJB クライアントアプリケーションの実行に必要な環境変数の設定」を参照してください。

2. JavaVM 起動オプションを指定します。
   vbj コマンドに適切な JavaVM 起動オプションを指定します。指定できるオプションの詳細については，マニュアル「Cosminexus 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)」の「19. JavaVM 起動オプション」を参照してください。
3. JAR ファイルのクラスパスを指定します。
   vbj コマンドに適切なクラスパスを指定します。必要なクラスパスについては，「3.7.4 EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定」を参照してください。
4. プロパティを指定します。
   プロパティで指定できる内容については，「3.3.5 EJB クライアントアプリケーションのプロパティの設定」を参照してください。また，必要に応じて，「3.8 EJB クライアントアプリケーションのシステムログ出力」，「3.5.3 EJB クライアントアプリケーションでのトランザクション実装時の注意事項」，およびマニュアル「Cosminexus アプリケーションサーバ 機能解説 拡張編」の「12.11 EJB クライアントアプリケーションのユーザログ出力の実装と設定（vbj コマンドを使用する場合)」を参照してください。
   指定できるプロパティの詳細については，マニュアル「Cosminexus 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)」の「15.3 usrconf.properties（Java アプリケーション用ユーザプロパティファイル)」を参照してください。
5. vbj コマンドを使用して EJB クライアントアプリケーションを開始します。

注意事項

uCosminexus Client を使用して EJB クライアント環境を構築する場合は，格納ディレクトリの「<Cosminexus のインストールディレクトリ >¥CC」を，「<Cosminexus のインストールディレクトリ >¥CCL」と読み替えてください。

3.3.4 EJB クライアントアプリケーションの実行に必要な環境変数の設定

この節では，EJB クライアントアプリケーションの環境変数の設定について説明します。EJB クライアントアプリケーションの実行に必要な環境変数を次に示します。

表 3-6 EJB クライアントアプリケーションの実行に必要な環境変数（Windows の場合）

環境変数	値	コマンド	
		cjclstartap	vbj
PATH ※1	<Cosminexus のインストールディレクトリ >¥jdk¥bin	―	○

環境変数	値	コマンド	
		cjclstartap	vbj
	<Cosminexus のインストールディレクトリ >¥TPB¥bin	○	－
	<Cosminexus のインストールディレクトリ >¥PRF¥bin	－	○
VBROKER_ADM	<Cosminexus のインストールディレクトリ >¥TPB¥adm	○	○
PRFSPOOL※2	<Cosminexus のインストールディレクトリ >¥PRF¥spool	▲	▲
TZ	JST-9　など	○	○

（凡例）

○：コマンドに環境変数を指定する必要がある。設定は必須である。

▲：インストーラによって設定される。設定は任意である。

－：設定する必要がない。

注※ 1

環境変数「PATH」の先頭に「<Cosminexus のインストールディレクトリ >¥jdk¥bin」を指定してください。

注※ 2

インストーラによって設定される PRFSPOOL 環境変数配下には，Cosminexus Performance Tracer のログが出力されますが，EJB クライアントアプリケーションを実行するマシンに PRF デーモンを配置しない場合，モジュールトレースが単調増加してしまいます。

PRF デーモンを配置しない場合は，PRFSPOOL 環境変数を設定しないでください。具体的には次のどちらかの方法を実行してください。

・システム環境変数から PRFSPOOL 環境変数を削除する。

・EJB クライアント実行時に，PRFSPOOL 環境変数を無効にする。

表 3-7　EJB クライアントアプリケーションの実行に必要な環境変数（UNIX の場合）

環境変数	値	コマンド	
		cjclstartap	vbj
LIBPATH，または LD_LIBRARY_PATH※1	/opt/Cosminexus/TPB/lib /opt/Cosminexus/PRF/lib	○	○
PATH※2	/opt/Cosminexus/jdk/bin	－	○
	/opt/Cosminexus/TPB/bin	○	－
	/bin /usr/bin	－	○
VBROKER_ADM	/opt/Cosminexus/TPB/adm	○	○
PRFSPOOL※3	/opt/Cosminexus/PRF/spool	△	△

環境変数	値	コマンド	
		cjclstartap	vbj
TZ	JST-9　など	○	○

（凡例）
○：コマンドに環境変数を指定する必要がある。設定は必須である。
△：設定は任意である。
－：設定する必要がない。

注※ 1
OS によって，使用する環境変数名が異なります。
LIBPATH：AIX の場合
LD_LIBRARY_PATH：HP-UX（IPF），Linux，または Solaris の場合

注※ 2
環境変数「PATH」の先頭に「/opt/Cosminexus/jdk/bin」を指定してください。

注※ 3
PRFSPOOL 環境変数配下には Cosminexus Performance Tracer のログが出力されますが，EJB クライアントアプリケーションを実行するマシンに PRF デーモンを配置しない場合，モジュールトレースが単調増加してしまいます。
PRF デーモンを配置しない場合は PRFSPOOL 環境変数を設定しないでください。具体的には，EJB クライアント実行時に，PRFSPOOL 環境変数を無効にしてください。

実行環境が AIX の場合は，上記の表に示す環境変数とは別に，AIX 固有の環境変数を設定する必要があります。詳細については，マニュアル「Cosminexus アプリケーションサーバ システム構築・運用ガイド」の「7.3　環境変数の設定」を参照してください。

3.3.5　EJB クライアントアプリケーションのプロパティの設定

EJB クライアントアプリケーションでは，使用する機能に応じたプロパティを設定できます。EJB クライアントアプリケーションで設定できるプロパティについては，マニュアル「Cosminexus 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）」の「15.3　usrconf.properties（Java アプリケーション用ユーザプロパティファイル）」を参照してください。

3.4 Enterprise Bean の呼び出し

この節では，EJB クライアントアプリケーションから Enterprise Bean を呼び出す方法，およびメソッドの呼び出しで通信障害が発生したときの，クライアント側の動作設定について説明します。

この節の構成を次の表に示します。

表 3-8 この節の構成（Enterprise Bean の呼び出し）

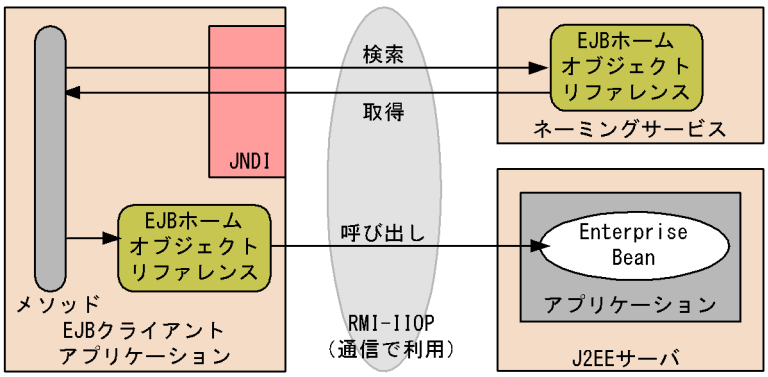
分類	タイトル	参照先
解説	EJB クライアントアプリケーションからの Enterprise Bean 呼び出しの流れ	3.4.1
実装	Enterprise Bean を呼び出すための実装	3.4.2

注 「設定」,「運用」および「注意事項」について，この機能固有の説明はありません。

3.4.1 EJB クライアントアプリケーションからの Enterprise Bean 呼び出しの流れ

EJB ホームオブジェクトのリファレンスを検索，取得する場合を例に，Enterprise Bean を呼び出す流れを示します。

図 3-1 EJB クライアントアプリケーションからホームインタフェースを利用して Enterprise Bean を呼び出す流れ



EJB クライアントアプリケーションからホームインタフェースを利用して Enterprise Bean を呼び出すには，JNDI を利用して EJB ホームオブジェクトのリファレンスを取得することで実現します。したがって，EJB クライアントアプリケーションで JNDI ネーミングコンテキストを生成し，EJB ホームオブジェクトのリファレンスを検索できるように実装する必要があります。Enterprise Bean の呼び出しの詳細については，

「3.4.2 Enterprise Bean を呼び出すための実装」を参照してください。また，EJB クライアントアプリケーションは通信手段に RMI-IIOP を利用するため，RMI-IIOP のスタブやインタフェースを参照できるようにしておく必要があります。RMI-IIOP のスタブやインタフェースの取得については，「3.7 RMI-IIOP スタブ，インタフェースの取得」を参照してください。

3.4.2 Enterprise Bean を呼び出すための実装

EJB クライアントアプリケーションから Enterprise Bean を呼び出すには，JNDI を利用します。ここでは，EJB ホームオブジェクトのリファレンスをルックアップする場合，およびビジネスインタフェースのリファレンスをルックアップする場合に分けて，Enterprise Bean を呼び出す方法を説明します。

(1) EJB ホームオブジェクトのリファレンスを検索して Enterprise Bean を呼び出す方法

EJB ホームオブジェクトのリファレンスをルックアップして Enterprise Bean を呼び出す方法を，実装例を基に説明します。

(a) JNDI ネーミングコンテキストの生成

EJB ホームオブジェクトのリファレンスのルックアップに利用する JNDI ネーミングコンテキストを生成します。

```
javax.naming.Context ctx = new javax.naming.InitialContext();
```

(b) EJB ホームオブジェクトのリファレンスの検索と取得

生成した JNDI ネーミングコンテキストを利用して，EJB ホームオブジェクトのリファレンスを取得します。EJB ホームオブジェクトのリファレンスを取得するには，自動的にバインドされる名称またはユーザ指定名前空間機能を利用して付与した名称でルックアップします。次の例では，ユーザ指定名前空間を利用してルックアップして，取得しています。ルックアップの方法については，マニュアル「Cosminexus 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「2.3 JNDI 名前空間へのオブジェクトのバインドとルックアップ」を参照してください。

```
String ejbName = "MySample";
java.lang.Object obj = ctx.lookup(ejbName);
SampleHome sampleHome =
  (SampleHome)javax.rmi.PortableRemoteObject.narrow(obj,
SampleHome.class);
```

(c) Enterprise Bean の生成とメソッドの呼び出し

EJB ホームオブジェクトの create メソッドによって，Enterprise Bean のインスタンス

を生成します。これによって，アプリケーションに必要となる Enterprise Bean のメソッドを呼び出せるようになります。

```
Sample remoteSample = sampleHome.create(); //Enterprise Bean インスタンスの生成
String result = remoteSample.getData("data"); // ビジネスメソッドの呼び出し
```

なお，Entity Bean で Collection 型が返される find メソッドを利用する場合は，コレクションから取り出したオブジェクトに対して，Enterprise Bean のクラスでナロウする必要があります。

```
Collection c = home.findByXXX(keyValue);
Iterator i=c.iterator();
while (i.hasNext()) {
  Sample
remoteSample=(Sample)javax.rmi.PortableRemoteObject.narrow(i.next(),
Sample.class);
                //RemoteSampleに対してビジネスメソッドを呼び出します。
}
```

(2) ビジネスインタフェースのリファレンスを検索して Enterprise Bean を呼び出す方法

ビジネスインタフェースのリファレンスをルックアップして Enterprise Bean を呼び出す方法を，実装例を基に説明します。

(a) InitialContext の生成

ビジネスインタフェースを使用して Enterprise Bean を呼び出すには，最初に InitialContext を生成します。

```
// InitialContextを生成
InitialContext ctx = new InitialContext();
```

(b) ビジネスインタフェースのリファレンスの検索と取得

生成した InitialContext を利用して，ビジネスインタフェースのリファレンスを取得します。ビジネスインタフェースのリファレンスを取得するには，自動的にバインドされる名称またはユーザ指定名前空間機能を利用して付与した名称でルックアップします。自動的にバインドされる名称を使用したビジネスインタフェースのルックアップの方法については，マニュアル「Cosminexus 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「2.3.2　EJB のリファレンスが自動的にバインドされる名称」を参照してください。

```
// ビジネスインタフェースのリファレンスを取得
Sample sample = (Sample)ctx.lookup("HITACHI_EJB/SERVERS/MyServer/EJBBI/
SampleApp/Sample");
```

(c) メソッドの呼び出し

ビジネスインタフェースのリファレンスを取得したことで，ビジネスメソッドを呼び出せるようになります。

```
// ビジネスメソッドの呼び出し
String result = sample.getData("data");
```

3.5 EJB クライアントアプリケーションでのトランザクションの実装

この節では，EJB クライアントアプリケーションでのトランザクションの実装について説明します。

この節の構成を次の表に示します。

表 3-9 この節の構成（EJB クライアントアプリケーションでのトランザクションの実装）

分類	タイトル	参照先
実装	EJB クライアントでトランザクションを使用する手順	3.5.1
	ルックアップを使用した UserTransaction の取得方法	3.5.2
注意事項	EJB クライアントアプリケーションでのトランザクション実装時の注意事項	3.5.3

注 「運用」について，この機能固有の説明はありません。

ポイント

「解説」と「設定」については，それぞれ次の説明を参照してください。

解説：

マニュアル「Cosminexus 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「3. リソース接続とトランザクション管理」

設定：

マニュアル「Cosminexus 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「3.20 EJB クライアントアプリケーションでトランザクションを開始する場合の注意事項」

注意事項

uCosminexus Client を使用して EJB クライアント環境を構築する場合は，EJB クライアントアプリケーションのトランザクションは使用できません。

3.5.1 EJB クライアントでトランザクションを使用する手順

ここでは，EJB クライアントでトランザクションを使用する場合の手順を説明します。

1. EJB クライアントアプリケーション起動時のクラスパスに，次の JAR ファイルを追

加します。

- Windows の場合
  <Cosminexus のインストールディレクトリ >¥TPB¥lib¥tpotsinproc.jar
  <Cosminexus のインストールディレクトリ >¥CC¥lib¥ejbserver.jar ※
- UNIX の場合
  /opt/Cosminexus/TPB/lib/tpotsinproc.jar
  /opt/Cosminexus/CC/lib/ejbserver.jar ※

注※

ejbserver.jar は HiEJBClientStatic.jar よりも後ろに追加してください。

2. EJB **クライアントアプリケーション起動時に必要なシステムプロパティを追加します。**
   システムプロパティの追加方法については，マニュアル「Cosminexus 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「3.20　EJB クライアントアプリケーションでトランザクションを開始する場合の注意事項」を参照してください。
3. EJB **クライアントアプリケーションのプロセスの起動直後に，**EJB **クライアントアプリケーションに実装したユーザコードから，サービスの初期化処理を実行します。**
   サービスの初期化処理を実行するには，EJBClientInitializer クラス（com.hitachi.software.ejb.ejbclient.EJBClientInitializer）を呼び出します。
   なお，EJBClientInitializer クラスの initialize メソッドを呼び出す前に，javax.naming.InitialContext を生成した場合や，UserTransactionFactory クラスの getUserTransaction メソッドを呼び出した場合は，その時点で初期化処理が行われます。このため，EJBClientInitializer クラスによる初期化処理は不要です。
   EJBClientInitializer クラスの文法や機能については，マニュアル「Cosminexus アプリケーションサーバ リファレンス API 編」の「4.2　EJBClientInitializer クラス」を参照してください。
4. UserTransaction **オブジェクトを取得します。**
   UserTransaction の取得方法には，次の 2 種類があります。
   - UserTransactionFactory クラスを使用する
     com.hitachi.software.ejb.ejbclient.UserTransactionFactory クラスの getUserTransaction メソッドを使用して取得します。UserTransactionFactory クラスの文法や機能については，マニュアル「Cosminexus アプリケーションサーバ リファレンス API 編」の「4.6　UserTransactionFactory クラス」を参照してください。
   - ルックアップを使用する
     ネーミングサービスからルックアップして取得します。ルックアップを使用した UserTransaction の取得方法については，マニュアル「Cosminexus 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「3.4.8 UserTransaction インタフェースを使用する場合の処理概要と留意点」を参照して

ください。

EJB クライアントでは，UserTransactionFactory クラスを使用する方法を推奨しています。ただし，他社のアプリケーションサーバからの移行などによって，EJB クライアントアプリケーションのソースコードを変更できない場合は，ルックアップを使用します。

5. Enterprise Bean を呼び出すスレッドから，UserTransaction インタフェースの begin メソッドを呼び出し，トランザクションを開始します。
6. サーバ上の Enterprise Bean を呼び出します。
7. Enterprise Bean を呼び出したスレッドから，UserTransaction インタフェースの commit メソッドまたは rollback メソッドを呼び出して，トランザクションを決着します。

3.5.2 ルックアップを使用した UserTransaction の取得方法

EJB クライアントアプリケーションから UserTransaction をルックアップする場合に指定する検索文字列を示します。

- 通常モードのアプリケーションを呼び出す場合
  HITACHI_EJB/SERVERS/< サーバ名称 >/SERVICES/UserTransaction
- テストモードのアプリケーションを呼び出す場合
  $HITACHI_TEST/HITACHI_EJB/SERVERS/< サーバ名称 >/SERVICES/UserTransaction

ルックアップの結果によって得られるオブジェクトは，java.lang.Object 型であるため，javax.transaction.UserTransaction 型にキャストして使用します。

また，ルックアップに失敗した場合，javax.naming.NamingException 例外が発生します。

UserTransaction 型へのキャストや例外については，J2EE サーバ上の Web アプリケーションや Enterprise Bean から UserTransaction をルックアップする場合と同様の仕様となります。

注意事項

次に示す環境での UserTransaction のルックアップはサポートされません。

- CTM 連携時に使用するグローバルネーミングサービスに対する UserTransaction のルックアップ
- ラウンドロビン検索機能などで使用されるユーザ指定名前空間管理機能を使用した UserTransaction のルックアップ

この環境で UserTransaction を使用する場合は，UserTransactionFactory クラスを使用して UserTransaction を取得してください。

3.5.3 EJB クライアントアプリケーションでのトランザクション実装時の注意事項

EJB クライアントでトランザクションを実装する場合の注意事項を示します。

- サービスの初期化処理で例外が発生した場合は，システムプロパティが正しく設定されていない可能性があります。例外のメッセージに従って対処してください。
- 呼び出される Enterprise Bean を，コンテナ管理のトランザクション（CMT）で mandatory，required，supports などの属性にすれば，EJB クライアントアプリケーションで開始したトランザクションの範囲内で実行されます。
- EJB クライアントアプリケーションが，トランザクション処理中に障害などの理由でダウンした場合は，EJB クライアントアプリケーションを再起動して，グローバルトランザクションのリカバリ処理を行う必要があります。EJB クライアントアプリケーションの再起動後に，EJBClientInitializer クラスの initialize メソッドを呼び出し，グローバルトランザクションのリカバリ処理が開始するように設計してください。なお，リカバリ処理はバックグラウンドで実行されるため，initialize メソッドはリカバリ処理の完了を待たないでリターンします。
- EJB クライアントアプリケーションでトランザクションを開始している場合は，必ずすべてのトランザクションを決着してから，EJB クライアントアプリケーションのプロセスが停止するように設計してください。トランザクションの決着処理を待たないで EJB クライアントアプリケーションのプロセスを停止すると，プリペア状態のトランザクションが決着されないまま残るおそれがあります。この状態になると，アプリケーションサーバの正常停止やリソースアダプタの停止ができなくなります。また，リソースのロックが解放されない場合があります。
  万一，プリペア状態のトランザクションが残ってしまった場合は，EJB クライアントアプリケーションを再起動して，グローバルトランザクションのリカバリ処理を行う必要があります。
- EJB クライアントアプリケーションでは，JTA や OTS が出力する性能解析トレースに，ルートアプリケーション情報やクライアントアプリケーション情報が含まれていません。リクエストをトレースする場合は，スレッドのハッシュコードや XID の情報

を使用してください。なお，トランザクションタイムアウトが発生した時に出力されるメッセージには，ルートアプリケーション情報の代わりに，トランザクションを開始したスレッドのハッシュコードが含まれています。

3.6 EJB クライアントアプリケーションでのセキュリティの実装

この節では，EJB クライアントアプリケーションでのセキュリティの実装について説明します。

EJB クライアントアプリケーションでは，J2EE サーバで定義されたユーザとパスワードを使用してユーザを認証できます。EJB クライアントアプリケーションからユーザを認証してログインすると，セキュリティロールが設定された Enterprise Bean のメソッドを呼び出せます。

この節の構成を次の表に示します。

表 3-10 この節の構成（EJB クライアントアプリケーションでのセキュリティの実装）

分類	タイトル	参照先
実装	セキュリティを実装する場合の前提条件	3.6.1
	セキュリティを実装した場合のサンプルプログラム	3.6.2

注 「設定」,「運用」および「注意事項」について，この機能固有の説明はありません。

ポイント

「解説」については，次の説明を参照してください。

解説：

マニュアル「Cosminexus 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「9. セキュリティ管理」

3.6.1 セキュリティを実装する場合の前提条件

EJB クライアントアプリケーションでのセキュリティは，Cosminexus が提供する API を使用して実装します。ここでは，セキュリティを実装する場合の前提条件および実装方法を示します。API の機能と文法については，マニュアル「Cosminexus アプリケーションサーバ リファレンス API 編」の「4. EJB クライアントアプリケーションで使用する API」を参照してください。

セキュリティを実装する前に，次の前提条件を満たしているか確認してください。

- J2EE サーバ側にユーザが登録されている必要があります。
- 登録されているユーザにセキュリティロールが設定されている必要があります。

(1) セキュリティの実装方法

EJB クライアントアプリケーションでセキュリティを実装する場合，次の手順で処理を実装します。

1. **セキュリティ API のパッケージをインポートします。**
   セキュリティ API を利用するために，次に示すパッケージをインポートします。

```
import com.hitachi.software.ejb.security.base.authentication.*
```

2. **LoginInfoManager のオブジェクトを取得します。**
   Enterprise Bean のメソッドを呼び出すプログラム上で LoginInfoManager オブジェクトを取得します。取得には LoginInfoManager オブジェクトに用意されているスタティックメソッドの getLoginInfoManager メソッドを使用します。

```
LoginInfoManager  lm = LoginInfoManager.getLoginInfoManager();
```

3. **ユーザ名とパスワードでログインします。**
   LoginInfoManager オブジェクト取得後，login メソッドを呼び出します。

```
lm.login(username, password);
```

4. **Enterprise Bean のメソッドを呼び出します。**
   login メソッド成功後，Enterprise Bean のメソッドを呼び出します。

5. **ログアウトします。**
   Enterprise Bean のメソッド呼び出しが終了したあと，logout メソッドで J2EE サーバからログアウトします。

```
lm.logout();
```

注意事項

EJB クライアントアプリケーションでセキュリティを実装する場合，
HiEJBClientStatic.jar をクラスパスに追加してコンパイルする必要があります。

3.6.2 セキュリティを実装した場合のサンプルプログラム

Enterprise Bean 名が account の場合に，getAccountID メソッドを呼び出すサンプルプログラムを次に示します。

```
import com.hitachi.software.ejb.security.base.authentication.*;
    :
  try {
    LoginInfoManager lm = LoginInfoManager.getLoginInfoManager();
    String userName = System.getProperty("username");
    String password = System.getProperty("password");
    if(lm.login(userName , password)) {
      try {
        System.out.println("user:" + userName + "login success");
        Context ctx = new InitialContext();
        java.lang.Object obj = ctx.lookup(appUnitPath + "Account");
        AccountHome aHome =

(AccountHome)PortableRemoteObject.narrow(obj,AccountHome.class);
        Account account  = aHome.create();
        account.getAccountID();
      } finally {
        lm.logout();
      }
    }
  } catch(NotFoundServerException e) {
    System.out.println("not found server");
  } catch(InvalidUserNameException e) {
    System.out.println("invalid user name");
  } catch(InvalidPasswordException e) {
    System.out.println("invalid password");
  } catch(Exception e) {
    e.printStackTrace();
  }
```

3.7 RMI-IIOP スタブ，インタフェースの取得

この節では，RMI-IIOP スタブ，およびインタフェースの取得について説明します。

この節の構成を次の表に示します。

表 3-11 この節の構成（RMI-IIOP スタブ，インタフェースの取得）

分類	タイトル	参照先
解説	RMI-IIOP スタブ，インタフェースの取得の概要	3.7.1
	サーバ管理コマンドによる手動ダウンロード	3.7.2
	ダイナミッククラスローディング	3.7.3
設定	EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定	3.7.4
注意事項	uCosminexus Client 使用時の注意	3.7.5

注 「実装」および「運用」について，この機能固有の説明はありません。

3.7.1 RMI-IIOP スタブ，インタフェースの取得の概要

EJB クライアントアプリケーションは，Cosminexus TPBroker の RMI-IIOP の機能を利用してアプリケーションを呼び出します。

EJB ホームオブジェクトのリファレンスを検索，取得する場合，EJB クライアントアプリケーションは次に示すスタブおよびクラスを参照できるようにする必要があります。

- Enterprise Bean の EJB オブジェクトのスタブ
- Enterprise Bean の EJB ホームオブジェクトのスタブ
- スタブが利用する各種クラス

また，EJB クライアントアプリケーションからは，次に示すインタフェース，および各種クラスを参照できるようにする必要があります。

- リモートインタフェース
- ホームインタフェース
- インタフェースが参照する各種クラス

ビジネスインタフェースを使用して Enterprise Bean を呼び出す場合は，ビジネスインタフェース，およびビジネスインタフェースを呼び出すためのクラスを参照できるようにする必要があります。

これらのクラス（RMI-IIOP スタブ，RMI-IIOP インタフェース）は，サーバ管理コマンドまたはダイナミッククラスローディングを使用してダウンロードします。

注意事項

サーバ管理コマンドとダイナミッククラスローディングの使い分け

ダイナミッククラスローディングを使用するよりも，サーバ管理コマンドを使用して呼び出しに必要なクラスをダウンロードした方が性能上優れています。したがって，実運用ではサーバ管理コマンドの使用を推奨します。一方，開発，テスト時にはスタブの取得と更新に手間の掛からない，ダイナミッククラスローディングの使用を推奨します。

3.7.2 サーバ管理コマンドによる手動ダウンロード

サーバ管理コマンドの cjgetstubsjar コマンドで RMI-IIOP スタブおよび RMI-IIOP インタフェースをダウンロードできます。cjgetstubsjar コマンドの使用方法については，マニュアル「Cosminexus 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「10.8　RMI-IIOP スタブとインタフェースの取得」を参照してください。

3.7.3 ダイナミッククラスローディング

RMI-IIOP スタブをクラスパスに指定しないで，EJB クライアントアプリケーションを開始します。EJB クライアントアプリケーションが Enterprise Bean を呼び出したときに RMI-IIOP スタブおよび RMI-IIOP インタフェースが自動的に読み込まれます。

RMI-IIOP スタブのダイナミッククラスローディングを利用する場合の手順を示します。

1. **簡易構築定義ファイルを編集します。**
   簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の <configuration> タグ内に，ejbserver.DynamicStubLoading.Enabled パラメタで「true」を指定します。
2. **該当する J2EE サーバを再開始します。**
   J2EE サーバを開始します。J2EE アプリケーションが開始済みの場合は，一度停止してから再開始してください。

注意事項

ダイナミッククラスローディング使用上の注意事項を次に示します。

- ダイナミッククラスローディングを使用する場合，同じ J2EE サーバ内で，同じパッケージ名称，インタフェース名称の Enterprise Bean を持つ J2EE アプリケーションを複数開始できません。なお，インタフェースの継承元となる親クラス名称が同じ場合も J2EE アプリケーションを複数開始できません。ここでの継承元となる親クラスとは，ユーザが作成するクラスであり，J2SE や J2EE が提供するクラスではありません。
- クライアントプログラムのクラスパスにスタブを指定していない場合，シリアライズされた Enterprise Bean オブジェクトの Handle（javax.ejb.Handle）を復元できません。
- CTM のグローバル CORBA ネーミングサービスをルックアップする場合には，ダイナミッククラスローディング機能は使用できません。

3.7.4 EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定

ここでは，EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定方法について説明します。EJB クライアントアプリケーションの開始に使用するコマンドによって，クラスパスへの JAR ファイルの設定方法が異なります。

● cjclstartap コマンドの場合

cjclstartap コマンドを使用する場合は，EJB クライアントアプリケーションのオプション定義ファイル（usrconf.cfg）で，クラスパスに JAR ファイルを設定します。

● vbj コマンドの場合

vbj コマンドを使用する場合は，バッチファイル / シェルスクリプトファイル，またはコマンドの引数で設定します。

EJB クライアントアプリケーションの実行に必要な JAR ファイルを次の表に示します。

表 3-12 EJB クライアントアプリケーションの実行に必要な JAR ファイル

JAR ファイル名	[種別] ※1 JAR ファイルの配置場所	含まれる内容	コマンド	
			cjclstartap	vbj
hitj2ee.jar	[固定] • Windows の場合 <Cosminexus のインストールディレクトリ >¥CC¥lib • UNIX の場合 /opt/Cosminexus/CC/lib	製品提供クラス	−	○
HiEJBClientStatic.jar	[固定] • Windows の場合 <Cosminexus のインストールディレクトリ >¥CC¥client¥lib • UNIX の場合 /opt/Cosminexus/CC/client/lib	製品提供クラス	−	○
vbjorb.jar vbsec.jar	[固定] • Windows の場合 <Cosminexus のインストールディレクトリ >¥TPB¥lib • UNIX の場合 /opt/Cosminexus/TPB/lib	製品提供クラス	−	○
cprf.jar	[固定] • Windows の場合 <Cosminexus のインストールディレクトリ >¥PRF¥lib • UNIX の場合 /opt/Cosminexus/PRF/lib	製品提供クラス	−	○

JAR ファイル名	[種別] ※1JAR ファイルの配置場所	含まれる内容	コマンド	
			cjclstartap	vbj
hntrlib2j.jar または hntrlib2j64.jar（サブディレクトリ専有モードの場合）※2	［固定］ • Windows の場合 <Program Files>¥Hitachi¥HNTRLib2¥classes • UNIX の場合 /opt/hitachi/HNTRLib2/classes	製品提供クラス	－	○
hntrlibMj.jar または hntrlibMj64.jar（サブディレクトリ共有モードの場合）※2 ※3	［固定］ • Windows の場合 <Program Files>¥Hitachi¥HNTRLib2¥classes • UNIX の場合 /opt/hitachi/HNTRLib2/classes	製品提供クラス	－	○
tpotsinproc.jar	［トランザクション使用］ • Windows の場合 <Cosminexus のインストールディレクトリ >¥TPB¥lib • UNIX の場合 /opt/Cosminexus/TPB/lib	製品提供クラス	－	△
ejbserver.jar	［トランザクション使用］ • Windows の場合 <Cosminexus のインストールディレクトリ >¥CC¥lib • UNIX の場合 /opt/Cosminexus/CC/lib		△	△
stubs.jar	［RMI-IIOP のスタブ］ J2EE サーバからダウンロード，またはダイナミッククラスローディング	RMI-IIOP のスタブ • EJB オブジェクトのスタブ • EJB ホームオブジェクトのスタブ • スタブが参照するクラス	△	△
< 数字 >.jar	［RMI-IIOP のインタフェース］ J2EE サーバからダウンロード	RMI-IIOP のインタフェース • リモートインタフェース • ホームインタフェース • インタフェースが参照するクラス	○	○

JAR ファイル名	[種別] ※1JAR ファイルの配置場所	含まれる内容	コマンド	
			cjclstartap	vbj
ユーザ作成の JAR ファイル	ユーザ作成のクラス	EJB クライアントアプリケーションで利用するユーザ作成のクラスです。 ユーザが作成した独自の Filter クラス，Formatter クラス，または Handler クラスを EJB クライアントアプリケーションのユーザログ機能で使用する場合には，そのクラスもクラスパスに指定してください。※3	○	○

（凡例）

○：クラスパスに指定する必要がある。

△：必要に応じてクラスパスに指定する。

－：クラスパスに指定する必要がない。

注※ 1

種別には，次に示すものがあります。

- ［固定］
  EJB クライアントアプリケーションから呼び出すアプリケーションの数，内容に関係なく，指定する JAR ファイルのファイル名と配置場所は固定です。
- ［トランザクション使用］
  EJB クライアントアプリケーションでトランザクションを使用する場合に，該当する JAR ファイルを指定します。詳細については，マニュアル「Cosminexus 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「3.20　EJB クライアントアプリケーションでトランザクションを開始する場合の注意事項」を参照してください。
- ［RMI-IIOP のスタブ］
  EJB クライアントアプリケーションから呼び出すアプリケーション単位に該当する JAR ファイルを指定します。ダイナミッククラスローディングを利用する場合は，指定する必要はありません。
- ［RMI-IIOP のインタフェース］
  RMI-IIOP のインタフェースを EJB クライアントアプリケーション側に取得していない場合，アプリケーション単位に該当する JAR ファイルをダウンロードして指定します。すでに RMI-IIOP のインタフェースを EJB クライアントアプリケーション側に取得している場合は，取得済みのクラスまたは JAR ファイルを指定します。

注※ 2

EJB クライアントアプリケーションの動作モードに合わせて，JAR ファイルを使用してください。サブディレクトリ専有モードは 06-50 よりも前のバージョンとの互換用に使用するモードであるため，EJB クライアントアプリケーションを新規作成する場合は，サブディレクトリ共有モードを使用することをお勧めします。EJB クライアントアプリケーションのユーザログ機能を使用する場合は，サブディレクトリ共有モードを使用してください。サブディレクトリ専

有モード，およびサブディレクトリ共有モードについては，「3.8.2　システムログの出力先のサブディレクトリ」を参照してください。
また，使用している OS に合わせて，JAR ファイルを使用してください。HP-UX（IPF）の場合および Linux（IPF）の場合は，hntrlib2j64.jar または hntrlibMj64.jar を指定してください。それ以外の場合は，hntrlib2j.jar または hntrlibMj.jar を指定してください。

注※ 3

EJB クライアントアプリケーションのユーザログ出力機能を使用する場合に指定してください。EJB クライアントアプリケーションのユーザログ出力の設定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 拡張編」の「12.8　J2EE アプリケーションのユーザログ出力の設定」を参照してください。

注意事項

- uCosminexus Client を使用して EJB クライアント環境を構築する場合は，格納ディレクトリの「<Cosminexus のインストールディレクトリ >¥CC」を，「<Cosminexus のインストールディレクトリ >¥CCL」と読み替えてください。
- uCosminexus Client を使用して EJB クライアント環境を構築する場合は，EJB クライアントアプリケーションのトランザクションは使用できません。
- クラスパスで JAR ファイルを設定する場合，JAR ファイルの設定順序に注意してください。
  トランザクションを使用する場合，クラスパスには，tpotsinproc.jar と ejbserver.jar を設定します。このとき，ejbserver.jar よりも先に，HiEJBClientStatic.jar を設定してください。
  性能解析トレース機能を利用する場合，クラスパスには，cprf.jar を設定します。このとき，HiEJBClientStatic.jar よりも先に cprf.jar を設定してください。
  逆の順序で設定した場合は，性能解析トレースの初期化に失敗します。また，逆の順序で設定した場合に，EJB クライアントアプリケーションのログレベルを Warning 以上に設定しているときは，KDJE51008-W のメッセージが理由コード「-4」で出力されます。なお，性能解析トレースの初期化に失敗した場合は，性能解析トレースは出力されませんが，EJB クライアントアプリケーションの処理は継続できます。

3.7.5　uCosminexus Client 使用時の注意

uCosminexus Client には cjgetstubsjar コマンドは含まれていないため，サーバ管理コマンドで取得できません。uCosminexus Client で RMI-IIOP のスタブを手動で取得する場合の手順を次に示します。

1. **Application Server が稼働するマシンで cjgetstubsjar コマンドを実行し，RMI-IIOP のスタブとインタフェースのファイルを任意のディレクトリに置きます。**
2. **uCosminexus Client が稼働するマシンから Application Server が稼働するマシンにアクセスし，ファイル転送などの方法で RMI-IIOP のスタブとインタフェースのファイルをダウンロードします。**

3.8 EJB クライアントアプリケーションのシステムログ出力

この節では，EJB クライアントアプリケーションのシステムログ出力について説明します。

この節の構成を次の表に示します。

表 3-13 この節の構成（EJB クライアントアプリケーションのシステムログ出力）

分類	タイトル	参照先
解説	EJB クライアントアプリケーションのシステムログの概要	3.8.1
	システムログの出力先のサブディレクトリ	3.8.2
設定	システムログの出力先や出力レベルの変更	3.8.3
	複数プロセスでのログ出力先のサブディレクトリの共有	3.8.4
	ログ出力先のサブディレクトリ数の管理	3.8.5
	ログ出力先ディレクトリのアクセス権の設定	3.8.6

注 「実装」,「運用」および「注意事項」について，この機能固有の説明はありません。

3.8.1 EJB クライアントアプリケーションのシステムログの概要

EJB クライアントアプリケーションのシステムログでは，メッセージログ，例外ログ，保守用ログの 3 種類のログが出力されます。EJB クライアントアプリケーションでは，必要に応じて，システムログの出力先や出力レベル，出力先のサブディレクトリの設定を変更できます。

3.8.2 システムログの出力先のサブディレクトリ

EJB クライアントアプリケーションのシステムログは，EJB クライアントアプリケーションのプロセス単位で出力されます。システムログでは，プロセスごとにログ出力先のサブディレクトリを作成するか，複数のプロセスでログ出力先のサブディレクトリを共有することができます。プロセスごとにログ出力先のサブディレクトリを作成する動作モードのことを**サブディレクトリ専有モード**，複数のプロセスでログ出力先のサブディレクトリを共有する動作モードのことを**サブディレクトリ共有モード**といいます。なお，既存の EJB クライアントアプリケーションをそのまま使用する場合は，サブディレクトリ専有モードで動作します。

サブディレクトリ共有モードとサブディレクトリ専有モードの違いを次の表に示します。

表 3-14 サブディレクトリ共有モードとサブディレクトリ専有モードの違い

項目	サブディレクトリ共有モード	サブディレクトリ専有モード
複数のプロセスでのサブディレクトリ共有の可否	共有できます。	共有できません。
ログ管理ファイル作成	作成されます。	作成されません。
ejbserver.client.ejb.log キー※のデフォルト値	system	ejb
ejbserver.client.log.appid キー※のデフォルト値	ejbcl	EJB クライアントアプリケーション識別 ID
ejbserver.client.log.directorynum キー※の指定	常に無効です。	ejbserver.client.log.appid キーを指定した場合は，無効です。
ejbserver.logger.channels.define.< チャネル名 >.filenum キー※に指定できる面数	1 ～ 64	1 ～ 16
ejbserver.logger.channels.define.< チャネル名 >.filesize キー※に指定できる容量（バイト）	4,096 ～ 16,777,216	4,096 ～ 2,147,483,647

注※ EJB クライアントアプリケーションの開始時に指定するシステムプロパティです。

ポイント

サブディレクトリ専有モードは 06-50 よりも前のバージョンとの互換用に使用するモードであるため，EJB クライアントアプリケーションを新規作成する場合は，サブディレクトリ共有モードを使用することをお勧めします。

注意事項

- cjclstartap コマンドで EJB クライアントアプリケーションを実行する場合は，サブディレクトリ共有モードを使用してください。
- EJB クライアントアプリケーションのユーザログ機能を使用する場合は，サブディレクトリ共有モードを使用してください。

● 動作モードの指定

サブディレクトリ共有モードとサブディレクトリ専有モードのどちらを使用するかは，クラスパスで指定できます。使用する動作モードによって，クラスパスに指定する JAR ファイルが異なります。クラスパスの指定については，「3.7.4 EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定」を参照してください。

● サブディレクトリの共有

サブディレクトリ共有モードを使用している場合は，ログ出力先のサブディレクトリを共有できます。サブディレクトリの共有については，「3.8.4 複数プロセスでのログ出力先のサブディレクトリの共有」を参照してください。

● サブディレクトリ数の管理

サブディレクトリ専有モードを使用している場合は，ログ出力先のサブディレクトリ数を管理できます。サブディレクトリ数の管理については，「3.8.5　ログ出力先のサブディレクトリ数の管理」を参照してください。

3.8.3　システムログの出力先や出力レベルの変更

(1) システムログの設定方法

EJB クライアントアプリケーションの開始に使用するコマンドによって，EJB クライアントアプリケーションのシステムログの設定方法が異なります。

● cjclstartap コマンドの場合

cjclstartap コマンドを使用する場合は，EJB クライアントアプリケーションのプロパティファイル（usrconf.properties）で，システムログのプロパティを設定します。

● vbj コマンドの場合

vbj コマンドを使用する場合は，バッチファイル / シェルスクリプトファイル，またはコマンドの引数で，システムログのプロパティを設定します。

(2) システムログの出力先や出力レベルの設定

Java アプリケーション用オプション定義ファイル（usrconf.cfg），または Java アプリケーション用ユーザプロパティファイル（usrconf.properties）をカスタマイズすることで，EJB クライアントアプリケーションのシステムログの属性を変更できます。変更できる項目と，変更を設定するプロパティのキーを次の表に示します。また，この表では，EJB クライアントアプリケーションの実行コマンドに指定するプロパティの指定の要否についても説明します。

表 3-15　EJB クライアントアプリケーションのシステムログの出力先や出力レベルを変更するキー

変更できる項目	Java アプリケーション用オプション定義ファイルのキー	Java アプリケーション用ユーザプロパティファイルのキー	種別
ログの出力先※1	ejb.client.log.directory	ejbserver.client.log.directory	可変
EJB クライアントアプリケーション単位で作成されるログ出力先ディレクトリ名※1	ejb.client.ejb.log	ejbserver.client.ejb.log	選択可変
EJB クライアントアプリケーションのプロセス単位で作成されるログ出力先サブディレクトリ名※1	ejb.client.log.appid	ejbserver.client.log.appid	選択可変

変更できる項目	Java アプリケーション用オプション定義ファイルのキー	Java アプリケーション用ユーザプロパティファイルのキー	種別
標準出力へのメッセージ出力停止※2	ejb.client.log.stdout.enabled	―	選択可変
EJB クライアントアプリケーションのプロセス単位で作成されるログ出力先サブディレクトリ数	―	ejbserver.client.log.directorynum	可変
ログファイル面数	―	ejbserver.logger.channels.define.<チャネル名※3>.filenum	選択可変
ログファイルサイズ	―	ejbserver.logger.channels.define.<チャネル名※3>.filesize	選択可変
ログの出力レベル※4	―	ejbserver.logger.enabled.*	選択可変
Cosminexus TPBroker のトレースファイルの出力先	―	vbroker.orb.htc.tracePath	選択可変
Cosminexus TPBroker のトレースファイルの個数	―	vbroker.orb.htc.comt.fileCount	選択可変
Cosminexus TPBroker のトレースファイル 1 個当たりのエントリ数	―	vbroker.orb.htc.comt.entryCount	選択可変

（凡例）

可変：システムの実行環境に従って値を指定する必要がある。
選択可変：システムの実行環境に従って値を指定するか，または指定を省略する。
―：設定できない。

注※ 1

usrconf.cfg では，稼働ログ，ログ稼働ログ，障害発生時の例外情報，および保守情報の設定を変更できます。
また，usrconf.cfg と usrconf.properties で同じ項目が設定された場合，usrconf.properties で設定した内容が優先されます。

注※ 2

稼働ログ，cjclstartap コマンドログ，および起動プロセス標準出力情報のメッセージを標準出力に出力しないように設定できます。

注※ 3

チャネル名として，ログの種類を示す次の名称が設定されます。
ClientMessageLogFile（稼働ログ）（ファイル名：cjclmessage[n].log）
ClientExceptionLogFile（障害発生時の例外情報）（ファイル名：cjclexception[n].log）
ClientMaintenanceLogFile（保守情報）（ファイル名：cjclmaintenance[n].log）

注※ 4

シェルスクリプトを使用してシステムプロパティを設定する場合，ログの出力レベル（ejbserver.logger.enabled.* キー）は指定できません。

システムプロパティを指定できるかどうかは，EJB クライアントアプリケーションの使用形態によって異なる場合があります。EJB クライアントアプリケーションの使用形態とシステムプロパティの関係を次の表に示します。なお，表 3-16 と表 3-17 の番号は対応しています。

表 3-16　EJB クライアントアプリケーションの使用形態

EJB クライアントアプリケーションの種類	EJB クライアントアプリケーションの同時起動の多重度		
	1 多重	2 ～ 8 多重	9 ～ 16 多重
1 種類	1.	2.	3.
2 種類以上	4.	5.	6.

表 3-17　EJB クライアントアプリケーションの使用形態とシステムプロパティ

システムプロパティの指定	EJB クライアントアプリケーションの使用形態					
	1.	2.	3.	4.	5.	6.
ejbserver.client.ejb.log	可能	可能	必須	必須	必須	必須
ejbserver.client.log.appid	可能	不可	不可	可能	不可	不可
ejbserver.client.log.directorynum	可能	可能	必須	可能	可能	必須

上記表の番号について説明します。

1. ejbserver.client.ejb.log キーは，デフォルトディレクトリでもかまいません。
   ejbserver.client.log.appid キーを指定した場合は，ejbserver.client.log.directorynum キーの指定は無効となります。
2. ejbserver.client.ejb.log キーは，デフォルトディレクトリでもかまいません。
   複数同時に起動するため，ejbserver.client.log.appid キーは指定しないでください。
   ejbserver.client.log.directorynum キーを指定する場合は，必ず ejbserver.client.ejb.log キーを指定してください。
3. ejbserver.client.ejb.log キーを EJB クライアントアプリケーションごとに必ず指定してください。複数同時に起動するため，ejbserver.client.log.appid キーを指定しないでください。ejbserver.client.log.directorynum キーの値を多重度に合わせて指定してください。
4. ejbserver.client.ejb.log キーを EJB クライアントアプリケーションごとに必ず指定してください。
   ejbserver.client.log.appid キーを指定した場合は，ejbserver.client.log.directorynum キーの指定は無効となります。
5. ejbserver.client.ejb.log キーを EJB クライアントアプリケーションごとに必ず指定し

てください。複数同時に起動するため，ejbserver.client.log.appid キーを指定しないでください。
ejbserver.client.log.directorynum キーを指定する場合は，必ず
ejbserver.client.ejb.log キーを指定してください。

6. ejbserver.client.ejb.log キーを EJB クライアントアプリケーションごとに必ず指定してください。複数同時に起動するため，ejbserver.client.log.appid キーを指定しないでください。ejbserver.client.log.directorynum キーの値を多重度に合わせて指定してください。

注意事項

- EJB クライアントアプリケーションの同時起動の多重度が 17 多重以上の場合は，ejbserver.client.log.directorynum キーに 0 を指定してください。cjcldellog コマンドを使用してサブディレクトリ数を管理してください。
- EJB クライアントアプリケーションを複数同時に起動すると，KDJE51005-W のメッセージが出力されることがあります。EJB クライアントアプリケーションの動作には問題ありませんが，サブディレクトリ数が
ejbserver.client.log.directorynum キーに指定した値を超えている場合があります。このメッセージが頻繁に出力される場合は，
ejbserver.client.log.directorynum キーの値を 0 にし，cjcldellog コマンドを使用してサブディレクトリ数を管理してください。ログ出力先のサブディレクトリ数を管理できるのは，サブディレクトリ専有モードを使用している場合だけです。サブディレクトリ共有モードを使用している場合は，サブディレクトリ数の管理はできません。

3.8.4 複数プロセスでのログ出力先のサブディレクトリの共有

サブディレクトリ共有モードを使用している場合は，ログ出力先のサブディレクトリを共有できます。

EJB クライアントアプリケーションのシステムログは，EJB クライアントアプリケーション単位に作成されるログ出力先ディレクトリ（ejbserver.client.log.directory キー，および ejbserver.client.ejb.log キーに指定したディレクトリ）下のサブディレクトリに格納されます。なお，サブディレクトリ名は，ejbserver.client.log.appid キーで指定できます。

サブディレクトリ共有モードを使用する場合，ログ出力先ディレクトリを指定する ejbserver.client.ejb.log キーには，サブディレクトリ専有モードを使用する場合とは異なる値を指定してください。サブディレクトリ専有モードと同じ値を指定した場合には，サブディレクトリ専有モードでのログ出力先のサブディレクトリ数が管理できなくなります。

サブディレクトリ共有モードで動作する EJB クライアントがログの稼働情報

（cjlogger.log ファイル）に KDJE90002-E メッセージを出力して終了する場合，または KDJE90003-E メッセージを出力する場合は，ログファイルの排他に失敗しているおそれがあります。ejbserver.client.log.lockRetryCount，および ejbserver.client.log.lockInterval キーを使用してリトライ回数とリトライ間隔を大きくすることで，ログファイルの排他の失敗を回避できます。

EJB クライアントアプリケーションのシステムログの取得については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編」の「4.5　EJB クライアントアプリケーションのシステムログ」を参照してください。

注意事項

EJB クライアントアプリケーションを初回起動時に複数同時に起動すると，同時刻に起動したほかの EJB クライアントアプリケーションとログディレクトリ生成処理が衝突し，KDJE51003-E メッセージを出力して異常終了する場合があります。EJB クライアントアプリケーションの初回起動時には複数同時に起動しないようにするか，あらかじめログ出力先ディレクトリを作成してください。

3.8.5　ログ出力先のサブディレクトリ数の管理

サブディレクトリ専有モードを使用している場合は，ログ出力先のサブディレクトリ数を管理できます。

EJB クライアントアプリケーションのシステムログは，EJB クライアントアプリケーション単位に作成されるログ出力先ディレクトリ（ejbserver.client.log.directory キー，および ejbserver.client.ejb.log キーに指定したディレクトリ）下の，サブディレクトリに格納されます。EJB クライアントアプリケーションを複数起動すると，起動したプロセス分のサブディレクトリが生成されます。このサブディレクトリが単調増加しないように管理できます。

サブディレクトリ数は，EJB クライアントアプリケーションのプロセス単位に作成されるサブディレクトリの数となります。ただし，EJB クライアントアプリケーション単位に作成されるログ出力先ディレクトリ下にユーザ作成のサブディレクトリがある場合，ユーザ作成のサブディレクトリの数もサブディレクトリとして数えられます。

デフォルトでは，ログ出力先のサブディレクトリ数を 8 個までとし，古いサブディレクトリから削除されるようになっています。サブディレクトリ数を変更する場合には，ejbserver.client.log.directorynum キーで変更してください。EJB クライアントアプリケーションのシステムログの取得については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編」の「4.5　EJB クライアントアプリケーションのシステムログ」を参照してください。

3.8.6　ログ出力先ディレクトリのアクセス権の設定

UNIX の場合，複数ユーザアカウントで同じログ出力先ディレクトリを使用して EJB ク

ライアントアプリケーションを実行するときは，ログ出力先ディレクトリのアクセス権を設定する必要があります。

ログ出力先ディレクトリのモードを，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し，umask を 0 に設定してから EJB クライアントアプリケーションを実行してください。

4 Enterprise Bean 実装時の注意事項

この章では，Enterprise Bean を実装するときの注意事項について説明します。

4.1 この章の構成

4.2 Enterprise Bean 共通の注意事項

4.3 Enterprise Bean の種類ごとの注意事項

4.1 この章の構成

この章では，アプリケーションサーバ上で動作するアプリケーションのプログラムとして，Enterprise Bean を実装するときの注意事項について説明します。

この章の構成を次の表に示します。

表 4-1 この章の構成（Enterprise Bean 実装時の注意事項）

分類	タイトル	参照先
Enterprise Bean 共通の注意事項	Enterprise Bean および関連するクラスの命名規則	4.2.1
	リソースのコネクションの取得と解放	4.2.2
	ローカルインタフェースとリモートインタフェースの使い分け	4.2.3
	ローカル呼び出し最適化機能の利用について	4.2.4
	ほかの J2EE アプリケーション内にある Enterprise Bean をコンポーネントインタフェースによって呼び出す方法	4.2.5
	ほかの J2EE アプリケーション内にある Enterprise Bean をビジネスインタフェースによって呼び出す方法	4.2.6
	クラスローダの取得に関する注意	4.2.7
	URLConnection クラス使用時の注意	4.2.8
	ネイティブライブラリのロードに関する注意	4.2.9
	Entity Bean（CMP，BMP 共通）のアクセス排他のタイムアウトについて	4.2.10
	Entity Bean（CMP，BMP 共通）使用時のデッドロックの発生について	4.2.11
	javax.ejb.EJBContext インタフェースメソッドについての注意事項	4.2.12
	Entity Bean（CMP，BMP 共通）属性ファイルの <prim-key-class> タグについて	4.2.13
	EJB の仕様に関する注意	4.2.14
	Unicode の補助文字の送受信に関する注意	4.2.15
	EJB 3.0 に関する注意	4.2.16
	ジェネリクスの使用に関する注意	4.2.17
Enterprise Bean の種類ごとの注意事項	Stateless Session Bean 実装時の注意事項	4.3.1
	Stateful Session Bean 実装時の注意事項	4.3.2
	Entity Bean（BMP）実装時の注意事項	4.3.3
	Entity Bean（CMP）実装時の注意事項	4.3.4

分類	タイトル	参照先
	Message-driven Bean 実装時の注意事項	4.3.5

注　この章には，注意事項以外の説明はありません。

4.2　Enterprise Bean 共通の注意事項

Enterprise Bean を実装するときの共通の注意事項を示します。

4.2.1　Enterprise Bean および関連するクラスの命名規則

Enterprise Bean クラス，ホームインタフェース，コンポーネントインタフェース，ビジネスインタフェース，インターセプタクラス，およびそれらで使用するクラスを実装する場合は，次に示す命名規則に従ってください。

- Enterprise Bean および関連するクラスのクラス名と同じ名前で始まるパッケージに，それらのクラスを配置しないでください。
- 「Wrappers.」で始まるパッケージ名は使用できません。
- Enterprise Bean クラス，ホームインタフェース，コンポーネントインタフェース，およびビジネスインタフェースの名称では，英数字および記号を使用します。アンダースコア（_）で始まるメソッド名，およびメンバ変数名は使用できません。
- 次に示すメソッドをビジネスメソッドとして使用しないでください。使用した場合，アプリケーションの開始時にコンパイルエラーとなったり，EJB が不正に動作することがあります。
  - java.lang.Object で定義された次のメソッド
    equals(Object)，hashCode()，toString()，clone()，finalize()
  - javax.ejb.EJBObject，javax.ejb.EJBLocalObject で定義された次のメソッド
    getEJBHome()，getEJBLocalHome()，getHandle()，getPrimaryKey()，isIdentical(EJBLocalObject)，isIdentical(EJBObject)，remove()
- Entity Bean の CMP ではアンダースコア（_）で始まる CMP フィールド名，および CMR フィールド名は指定できません。
- 大文字，小文字の違いだけのパッケージ名称，およびクラス名称は利用できません。RMI-IIOP では大文字，小文字が区別されないため，Enterprise Bean に正常にアクセスできなくなるおそれがあります。
- JAR ファイル中にあるホームインタフェースなど，java.rmi.Remote を実装したクラスがあると，スタブは作業ディレクトリ内に生成されます。作業ディレクトリのパス長がプラットフォームの上限に達しないように java.rmi.Remote を実装したクラスのパッケージ名やクラス名を指定してください。作業ディレクトリのパス長の見積もりについては，次に示すマニュアルの個所を参照してください。
  - J2EE サーバの場合
    マニュアル「Cosminexus アプリケーションサーバ システム構築・運用ガイド」の「7.10　J2EE サーバの作業ディレクトリ」
  - バッチサーバの作業ディレクトリ
    マニュアル「Cosminexus アプリケーションサーバ システム構築・運用ガイド」の「7.11　バッチサーバの作業ディレクトリ」

- EJB-JAR ファイルが別の場合でも，同じアプリケーション内では，クラスが重複しないようにしてください。
- Enterprise Bean のリモートインタフェース，リモートコンポーネントインタフェースのクラス名は，VisiBroker の予約名を使用しないでください。なお，VisiBroker の予約名には，「Helper」,「Holder」,「Package」,「Operations」,「POA」, および「POATie」があります。VisiBroker の予約名については，マニュアル「Borland(R) Enterprise Server VisiBroker(R) プログラマーズリファレンス」を参照してください。
- デフォルトパッケージに Enterprise Bean クラスを作成する場合，次に示す文字列で終わるクラスをデフォルトパッケージに作成しないでください。
  - _LocalHomeImpl
  - _LocalComponentImpl
  - _RemoteHomeImpl
  - _RemoteComponentImpl
  - _LocalBIProxyImpl
  - _LocalBIClientSideProxyImpl
  - _RemoteBIProxyInterface
  - _RemoteBIProxyImpl
  - _RemoteBIClientSideProxyImpl
  - _CallbackWrapperImpl
  - _InvocationContextImpl

4.2.2 リソースのコネクションの取得と解放

Enterprise Bean で JDBC，JMS などの J2EE リソースのコネクションを使用する場合，Connection クラスの close メソッドでコネクションを解放してください。コネクションを解放しないと，予期しないうちにリソースを消費し尽くしてしまいます。このような状況を避けるため，次のような点を考慮して実装してください。

● コネクションを Enterprise Bean クラスのメンバ変数に保持しないようにします。

コネクションをメンバ変数に保持していると，Bean の実行に関係なくコネクションが使用状態のままになり，パフォーマンスが低下します。そのため，SQL 実行時にコネクションの取得と解放をするようにしてください。

● コネクションプールを設定します。

コネクションプールを設定すると，物理的コネクションを再利用するためコネクション取得のオーバーヘッドを削減できます。また，JTA のトランザクション配下で実行する場合はコネクションプールの設定をする必要があります。

4.2.3 ローカルインタフェースとリモートインタフェースの使い分け

EJB 2.0 仕様で追加されたローカルインタフェースを利用すると，RMI-IIOP 通信処理の

オーバーヘッドを削減できますが，ローカルインタフェースだけを持つ Enterprise Bean はリモート呼び出しができません。また，ローカルインタフェースを使用する場合，メソッドの引数，戻り値が pass by reference で動作できる必要があります。次に示すローカルインタフェースの使用条件を参考に，ローカルインタフェースとリモートインタフェースを使い分けてください。

ローカルインタフェースの使用が推奨される場合

Enterprise Bean とクライアント（Enterprise Bean または JSP/ サーブレット）が同じ J2EE アプリケーションに含まれている場合，ローカルインタフェースの使用が推奨されます。

ローカルインタフェースの使用が必須の場合

CMP2.x の CMR は，ローカルインタフェースの使用が前提のため，ローカルインタフェースを使用してください。

ローカルインタフェースが使用できない場合

次の場合は，ローカルインタフェースが使用できません。

- Enterprise Bean とクライアント（Enterprise Bean または JSP/ サーブレット）が別の J2EE アプリケーションに含まれている場合。
- Enterprise Bean とクライアント（Enterprise Bean または JSP/ サーブレット）がほかの J2EE サーバ（別の JavaVM）で実行される場合。
- EJB 1.1 ですでに実装されている場合。

4.2.4 ローカル呼び出し最適化機能の利用について

リモートインタフェース使用時にローカル呼び出し最適化機能を利用すると，RMI-IIOP 通信処理のオーバーヘッドを削減できます。ローカル呼び出しは，メソッド呼び出しとほぼ同等の呼び出し形式になりますが，メソッドの引数，戻り値は pass by value で処理されます。

4.2.5 ほかの J2EE アプリケーション内にある Enterprise Bean をコンポーネントインタフェースによって呼び出す方法

同一 J2EE サーバ内にある場合と，ほかの J2EE サーバ内にある場合に分けて，ほかの J2EE アプリケーション内にある Enterprise Bean をコンポーネントインタフェースで呼び出す方法を示します。

（1） 同一 J2EE サーバ内のほかのアプリケーションで動作する Enterprise Bean の場合

次の手順で呼び出します。

1. **呼び出し元の** EJB-JAR **ファイルまたは** WAR **ファイルに，呼び出し先** Enterprise Bean **のリモートホームインタフェース，リモートインタフェースおよび各インタフェースで使用するユーザ作成クラスを含めます。**
2. J2EE **サーバ用ユーザ定義ファイルに次に示すプロパティを指定します。**
   usrconf.properties の ejbserver.deploy.stub.generation.scope キーに「app」を指定します。
   usrconf.properties については，マニュアル「Cosminexus 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）」の「15.3　usrconf.properties（Java アプリケーション用ユーザプロパティファイル）」を参照してください。
3. J2EE **サーバ用ユーザ定義ファイルに次に示すプロパティを指定します。**
   usrconf.properties の ejbserver.rmi.localinvocation.scope キーに「all」または「none」を指定します。
   usrconf.properties については，マニュアル「Cosminexus 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）」の「15.3　usrconf.properties（Java アプリケーション用ユーザプロパティファイル）」を参照してください。
4. **ネーミングの切り替え機能を使用して** corbaname **で始まる，ほかの** J2EE **アプリケーション内の** Enterprise Bean **のルックアップ名を指定します。**
   指定方法については，次に示すマニュアルの個所を参照してください。
   - サーバ管理コマンドを使って操作する場合
     マニュアル「Cosminexus 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「9.3.1　ほかの Enterprise Bean のリファレンス定義」
   - Server Plug-in を使って操作する場合
     マニュアル「Cosminexus 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「18.2.1　ほかの Enterprise Bean のリファレンス定義」

   指定例を次に示します。

```
corbaname::NamingHost:900#HITACHI_EJB/SERVERS/MyServer/EJB/
MyApplication/MyBean
```

（2） ほかの J2EE サーバ上で動作する Enterprise Bean の場合

次の手順で呼び出します。

1. **呼び出し元の** EJB-JAR **ファイルまたは** WAR **ファイルに，呼び出し先** Enterprise Bean **のリモートホームインタフェース，リモートインタフェースおよび各インタフェースで使用するユーザ作成クラスを含めます。**
2. J2EE **サーバ用ユーザ定義ファイルに次に示すプロパティを指定します。**
   usrconf.properties の ejbserver.deploy.stub.generation.scope キーに「app」を指定します。
3. **ネーミングの切り替え機能を使用して** corbaname **で始まるほかの** J2EE **アプリケーション内の** Enterprise Bean **のルックアップ名を指定します。**

指定方法については，次に示すマニュアルの個所を参照してください。

- サーバ管理コマンドを使って操作する場合
  マニュアル「Cosminexus 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「9.3.1　ほかの Enterprise Bean のリファレンス定義」を参照してください。
- Server Plug-in を使って操作する場合
  マニュアル「Cosminexus 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「18.2.1　ほかの Enterprise Bean のリファレンス定義」

指定例を次に示します。

```
corbaname::NamingHost:900#HITACHI_EJB/SERVERS/MyServer/EJB/
MyApplication/MyBean
```

4.2.6 ほかの J2EE アプリケーション内にある Enterprise Bean をビジネスインタフェースによって呼び出す方法

同一 J2EE サーバ内にある場合と，ほかの J2EE サーバ内にある場合に分けて，ほかの J2EE アプリケーション内にある Enterprise Bean をビジネスインタフェースで呼び出す方法を示します。

（1） 同一 J2EE サーバ内のほかのアプリケーションで動作する Enterprise Bean の場合

次の手順で呼び出します。

1. **呼び出し元の** EJB-JAR **ファイルまたは** WAR **ファイルに，呼び出し先** Enterprise Bean **のビジネスインタフェースおよびインタフェースで使用するユーザ作成クラスを含めます。**
2. J2EE **サーバ用ユーザ定義ファイルの** ejbserver.rmi.localinvocation.scope **キーに「**all**」または，「**none**」を指定します。**
3. **「**none**」の場合は，呼び出し先の** Enterprise Bean **に対して，**cjgetstubsjar **コマンドでスタブを取得し，呼び出し元の** EAR **ファイルに含めます。**※
   Enterprise Bean は EJB 参照を使用しないでルックアップします。ルックアップ名を指定する形式を示します。
   「HITACHI_EJB/SERVERS/< サーバ名 > /EJBBI/<J2EE APP 名 >/<Enterprise Bean 名 >」
   < サーバ名 >：J2EE サーバのサーバ名称
   <J2EE APP 名 >：J2EE アプリケーションのルックアップ名称
   <Enterprise Bean 名 >：Enterprise Bean のルックアップ名称

   注※
   　アプリケーション間のビジネスインタフェース呼び出しで，ダイナミッククラス

ローディング機能を使用すればスタブを含める必要はありませんが，ダイナミッククラスローディング機能は性能上推奨しません。

（2） ほかの J2EE サーバ上で動作する Enterprise Bean の場合

次の手順で呼び出します。

1. **呼び出し元の** EJB-JAR **ファイルまたは** WAR **ファイルに，呼び出し先** Enterprise Bean **のビジネスインタフェースおよびインタフェースで使用するユーザ作成クラスを含めます。**
2. **呼び出し先の** Enterprise Bean **に対して** cjgetstubsjar **コマンドでスタブを取得し，呼び出し元の** EAR **ファイルに含めます。**※
   Enterprise Bean は EJB 参照を使用しないでルックアップします。ルックアップ名を指定する形式を示します。
   「HITACHI_EJB/SERVERS/< サーバ名 > /EJBBI/<J2EE APP 名 >/<Enterprise Bean 名 >」
   < サーバ名 >：J2EE サーバのサーバ名称
   <J2EE APP 名 >：J2EE アプリケーションのルックアップ名称
   <Enterprise Bean 名 >：Enterprise Bean のルックアップ名称

   注※
   アプリケーション間のビジネスインタフェース呼び出しで，ダイナミッククラスローディング機能を使用すればスタブを含める必要はありませんが，ダイナミッククラスローディング機能は性能上推奨しません。

4.2.7 クラスローダの取得に関する注意

J2EE アプリケーション内のコードから Cosminexus Component Container のクラスローダを取得して，次に示す API を使用する場合に，java.net.JarURLConnection クラスが使用されます。

- getResource(String).openConnection().getInputStream();
- getResource(String).openStream()

上記メソッドの延長で java.net.JarURLConnection クラスの openConnection メソッドが呼び出され，該当する URL に指定された JAR ファイルがオープンされます。JAR ファイルに対する操作が必要で java.net.JarURLConnection クラスの openConnection メソッドを使用する場合には，該当する JAR ファイルに対して close メソッドを必ず呼び出すようにしてください。明示的に close メソッドを呼ばないかぎり，オープンされたままになり削除できません。また，上記メソッドは J2EE アプリケーション内で使用しないでください。

4.2.8 URLConnection クラス使用時の注意

java.net.URLConnection クラスは setUseCaches(boolean) メソッドを使用して，指定された URL に対してコネクションを取得するときにキャッシュの情報を利用するかどうかを指定できます。URLConnection クラスに対して setUseCaches(false) メソッドを指定した場合に，コネクションごとに対象のオブジェクトが生成されます。J2EE アプリケーション内のコードから使用する場合には，J2EE サーバの JavaVM がメモリリークするおそれがあります。

4.2.9 ネイティブライブラリのロードに関する注意

System.loadLibrary メソッドを使用して，Enterprise Bean からネイティブライブラリをロードしないでください。Enterprise Bean でネイティブライブラリをロードすると，JNI 仕様の制約によって，java.lang.UnsatisfiedLinkError が発生することがあります。ネイティブライブラリのロードが必要な場合は，System.loadLibrary メソッドを呼び出すコンテナ拡張ライブラリを作成し，Enterprise Bean からコンテナ拡張ライブラリを参照するように実装してください。コンテナ拡張ライブラリの作成については，マニュアル「Cosminexus 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「13. コンテナ拡張ライブラリ」を参照してください。

4.2.10 Entity Bean（CMP，BMP 共通）のアクセス排他のタイムアウトについて

Entity Bean（CMP，BMP）を使用する場合，同一プライマリキーの Entity Bean に対するアクセスには排他が掛かります。同時に同一のプライマリキーの Entity Bean に対してアクセスし，処理時間が掛かる場合は，排他待ちが行われます。排他待ちの結果，排他を取得できなかった場合は，タイムアウト（デフォルト 45 秒）で例外 IllegalStateException が発生します。この場合，J2EE サーバ用ユーザ定義ファイル（/opt/Cosminexus/CC/server/usrconf/ejb/< サーバ名称 >/usrconf.properties）の ejbserver.server.mutex.invocation.timeout キーにタイムアウト時間を指定してタイムアウトを発生させないようにすることができます。
ejbserver.server.mutex.invocation.timeout キーについては，マニュアル「Cosminexus 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)」の「2.4 usrconf.properties（J2EE サーバ用ユーザプロパティファイル）」を参照してください。

4.2.11 Entity Bean（CMP，BMP 共通）使用時のデッドロックの発生について

アプリケーションサーバでは，Entity Bean のトランザクション処理にデータベースのトランザクション機能を使用しています。データベースへのアクセスの順番，データの排他にかかわるテーブルの設定，データベースのシステムの設定，および呼び出す SQL

文によっては，デッドロックが発生する場合があります。詳細は使用しているデータベースのマニュアルを参照してください。

4.2.12 javax.ejb.EJBContext インタフェースメソッドについての注意事項

javax.ejb.EJBContext インタフェースの getUserTransaction メソッド，getRollbackOnly メソッド，setRollbackOnly メソッドについては，Enterprise Bean のトランザクション管理モデルによっては発行できません。また，Enterprise Bean のメソッドのうち，EJB 仕様で "Unspecified transaction" で動作するとされるものでは，発行できません。発行できない場合，EJB コンテナは java.lang.IllegalStateException を送出します。各メソッドの発行可否についてそれぞれの表に示します。

表 4-2 トランザクション管理モデル別の発行可否

javax.ejb.EJBContext メソッド	発行可否	
	BMT	CMT
getUserTransaction	○	×
getRollbackOnly	×	○
setRollbackOnly	×	○

（凡例）○：発行できる。　×：発行できない。

表 4-3 EJB のメソッド別の発行可否

Bean 種別	メソッド	発行可否
SessionBean	コンストラクタ	×
	setSessionContext	×
	ejbCreate	×
	ejbRemove	×
	ejbPassivate	×
	ejbActivate	×
	ビジネスメソッド	○
	afterBegin	○
	beforeCompletion	○
	afterCompletion	×
EntityBean	コンストラクタ	○
	setEntityContext	×
	unsetEntityContext	×
	ejbCreate	○

Bean 種別	メソッド	発行可否
	ejbPostCreate	○
	ejbRemove	○
	ejbHome	○
	ejbPassivate	×
	ejbActivate	×
	ejbLoad	○
	ejbStore	○
	ビジネスメソッド	○
Message-driven Bean	コンストラクタ	×
	ejbCreate	×
	onMessage	○
	メッセージリスナのメソッド	○
	ejbRemove	×

（凡例）○：発行できる。　×：発行できない。

4.2.13 Entity Bean（CMP，BMP 共通）属性ファイルの <prim-key-class> タグについて

Entity Bean 属性ファイルの <prim-key-class> タグにインタフェースおよび抽象クラスを指定した場合，次のように動作します。

- Entity Bean（CMP）の場合
  デプロイ時にエラーメッセージ KDJE42039-E が出力され，デプロイ処理はエラーで終了します。
- Entity Bean（BMP）の場合
  クラスを指定したときと同様，デプロイおよび実行できます。

4.2.14 EJB の仕様に関する注意

EJB 仕様に従っていない記述をした場合，エラー（例外）が発生することがあります。以前のバージョンでエラーが発生しなくても，バージョンアップをしたときにエラーになることがあります。エラーが発生した場合は，エラーの内容を確認し，EJB 仕様に従って変更してください。

4.2.15 Unicode の補助文字の送受信に関する注意

構成ソフトウェアに Cosminexus TPBroker を含むアプリケーションサーバでは，RMI-IIOP 通信による Enterprise Bean のメソッド呼び出しで，Unicode の補助文字を

送受信できます。この場合，送信側と受信側のアプリケーションサーバのバージョンによって，Unicode の補助文字の送受信時の動作が異なります。アプリケーションサーバのバージョンの組み合わせと，Unicode の補助文字送受信時の動作を次の表に示します。

表 4-4　アプリケーションサーバのバージョンの組み合わせと，Unicode の補助文字送受信時の動作

送信側のアプリケーションサーバのバージョン	受信側のアプリケーションサーバのバージョン	
	07-50 以降	07-50 より前
07-50 以降	○	×
07-50 より前	×	×

（凡例）
○：Unicode の補助文字を送受信できる。
×：Unicode の補助文字を送受信できない。

Unicode の補助文字送受信ができないバージョンの場合，RMI-IIOP 通信による Unicode の補助文字送受信時に，java.rmi.RemoteException の例外または java.rmi.MarshalException の例外がスローされます。

なお，Unicode の補助文字を送受信した場合の動作を，07-50 より前と同じ動作に設定できます。07-50 より前と同じ動作を設定したい場合は，usrconf.properties（J2EE サーバ用ユーザプロパティファイル，および Java アプリケーション用ユーザプロパティファイル）で，vbroker.orb.htc.surrogateCheckOff キーに「false」を設定してください。キーの詳細については，マニュアル「TPBroker ユーザーズガイド」を参照してください。

4.2.16　EJB 3.0 に関する注意

08-70 以降のアプリケーションサーバでは，次の要素を EJB3.0 の ejb-jar.xml に記述できます。

- <display-name> 要素
- <interceptor-binding> 要素，およびその配下の要素（インターセプタに関する定義）
- <application-exception> 要素，およびその配下の要素（アプリケーション例外に関する定義）

4.2.17　ジェネリクスの使用に関する注意

ジェネリクスは，J2SE 5.0 以降で使用できる機能です。ジェネリクスを使用すると，さまざまな型のオブジェクトを型やメソッドで使用する場合に，コンパイル時に型の安全性を保証できます。また，ジェネリクスを使用することで，データの型に依存しない，型そのものをパラメタとして扱うプログラミングを実現できます。

型を変数化することを**型変数**といいます。型変数は，ジェネリクスを使用したクラス，インタフェース，メソッドまたはコンストラクタの宣言時に使用します。「List<E> extends Collection<E>」という定義の場合，<E> の部分が型変数に該当します。

また，ジェネリクスを使用したクラス，インタフェース，メソッド，またはコンストラクタに対して，具体的なパラメタの型を指定することを**パラメタ化**といいます。例えば，「List<String>」，「Collection<Integer>」などが，パラメタ化されたクラスです。

(1) 型変数を使用できるインタフェース

インタフェースでは，メソッドの引数，戻り値，および例外で型変数を使用できます。使用する型変数は，インタフェース定義で宣言します。

(a) 型変数を使用できるインタフェースの種類

型変数を使用できるインタフェースの種類を Enterprise Bean の種別ごとに次の表に示します。

表 4-5 型変数を使用できるインタフェースの種類

インタフェースの種類	Enterprise Bean の種別		
	Session Bean	Entity Bean	Message-driven Bean
ビジネスインタフェース	△	－	－
Home インタフェース	×	×	－
Component インタフェース	×	×	－
メッセージリスナインタフェース	－	－	○
その他（任意のインタフェース）	○	○	○

（凡例）
○：使用できる。
△：使用できる。ただし，宣言した型変数はビジネスインタフェースでパラメタ化する必要がある。パラメタ化しなかった場合，エラーが発生するか，保証されない動作になる。
×：使用できない。
－：使用できない（Java EE 仕様）。

(b) ビジネスインタフェースで型変数を使用した場合の注意事項

インタフェースで定義した型変数をビジネスインタフェースでパラメタ化した際に，パラメタ化したビジネスインタフェースのメソッドを再定義した場合，型変数を使用した個所とインタフェース種別の組み合わせによって，エラーになることがあります。エラーが発生した場合は，再定義したメソッドを削除して対処してください。

ビジネスインタフェースでメソッドを再定義した場合にエラーとなる組み合わせを次の表に示します。

表 4-6 ビジネスインタフェースでメソッドを再定義した場合にエラーとなる組み合わせ

型変数を使用した個所	ローカルインタフェース	リモートインタフェース
戻り値	○	○
引数	×	×
例外	○	○

（凡例）
○：アプリケーションの開始に成功する。
×：アプリケーションの開始時にエラーが発生する。

エラーが発生するのは，次の項目の両方に該当する場合です。

- インタフェースで定義した型変数をパラメタ化している。
- 型変数をパラメタ化したインタフェースを継承したインタフェース内で，メソッドを再定義している。

エラーが発生するコーディングの例を次に示します。この例の場合，エラーを回避するためには，MyInterface の定義からメソッド「String get(Float args)」の定義を削除する必要があります。

```
public interface SuperInterface<T> {
String get(T args);
}

// SuperInterfaceをパラメタ化して継承したメソッドを再定義している。
public interface MyInterface
extends SuperInterface<Float> {
// エラー回避のためには，次の再定義したメソッド定義を削除する必要がある。
    String get(Float args);
}
```

(2) 型変数を使用できるクラス

クラスでは，メソッドのシグネチャおよびメソッド内で型変数を使用できます。

型変数を使用できるクラスの種類を Enterprise Bean の種別ごとに次の表に示します。

表 4-7 型変数を使用できるクラスの種類

クラスの種類	Enterprise Bean の種別		
	Session Bean	Entity Bean	Message-driven Bean
Enterprise Bean クラス	○	○	○
インターセプタクラス	○	－	－
その他（任意のクラス）	○	○	○

（凡例）

○：使用できる。
－：使用できない（アプリケーションサーバでのサポート外の組み合わせである）。

4.3　Enterprise Bean の種類ごとの注意事項

Enterprise Bean の種類ごとの注意事項を示します。

4.3.1　Stateless Session Bean 実装時の注意事項

Stateless Session Bean を実装するときの注意事項を示します。

（1） remove メソッドによるリファレンスの解放

ホームインタフェースを使用して Stateless Session Bean を呼び出す場合，ホームインタフェースの create メソッドを呼び出してリファレンスを取得しますが，Session Bean の呼び出しが完了したあとに，必ず remove メソッドを呼び出してリファレンスを解放してください。リファレンスを解放しない場合，J2EE サーバ上のメモリを消費したままの状態になります。

また，remove メソッド呼び出しを不要にするオプションを指定することで，Stateless Session Bean の EJB オブジェクトに対する，remove メソッドの呼び出しが不要になります。このオプションを有効にすると，remove メソッド呼び出しのあとに，ビジネスメソッドの呼び出しができます。

このオプションを無効にしている場合は，remove メソッドを呼び出す必要があります。また，remove メソッド呼び出しのあとにビジネスメソッドを呼び出した場合，java.rmi.NoSuchObjectException 例外が呼び出し元に返ります。

remove メソッド呼び出しを不要にするオプションは，usrconf.properties の ejbserver.rmi.stateless.unique_id.enabled キーに指定し，J2EE サーバ単位に定義します。キーの詳細については，マニュアル「Cosminexus 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)」の「2.4　usrconf.properties（J2EE サーバ用ユーザプロパティファイル)」を参照してください。また，構成ソフトウェアに Cosminexus TPBroker を含まない uCosminexus Developer Standard では，プロパティの設定に関係なく常にこのオプションは有効になります。

（2） ejbCreate メソッドおよび @PostConstruct アノテーションを指定したメソッドでのリソースマネジャへのアクセスについて

EJB 仕様では，ejbCreate メソッドまたは @PostConstruct アノテーションを指定したメソッドで，リソースマネジャへアクセスすることは許されていません。

（3） Bean クラスの共有についての注意

同じ Session Bean を同じ J2EE アプリケーション内で Stateful Session Bean，および Stateless Session Bean として同時に利用しないでください。

(4) ejbRemove メソッドまたは @PreDestroy アノテーションを指定したメソッドでの javax.transaction.UserTransaction の begin メソッドの呼び出しについて

Stateless Session Bean の ejbRemove メソッドまたは @PreDestroy アノテーションを指定したメソッドでは，javax.transaction.UserTransaction の begin メソッドを呼び出せる場合がありますが，EJB の仕様上，呼び出して使用することはできません。呼び出さないようにしてください。

4.3.2 Stateful Session Bean 実装時の注意事項

Stateful Session Bean を実装するときの注意事項を示します。

(1) remove メソッドまたは @Remove アノテーションを指定したメソッドによる EJB インスタンスの削除とリファレンスの解放

- ホームインタフェースを使用して Stateful Session Bean を呼び出す場合，ホームインタフェースの create メソッドを呼び出してリファレンスを取得しますが，Session Bean の呼び出しが完了したあとに，必ず remove メソッドを呼び出して，EJB インスタンスの削除とリファレンスを解放してください。
- ビジネスインタフェースを使用して Stateful Session Bean を呼び出す場合，ビジネスメソッドの呼び出しが完了したあとに，必ず @Remove アノテーションを指定したメソッドを呼び出して，EJB インスタンスを削除し，リファレンスを解放してください。
- EJB インスタンスの削除およびリファレンスの解放をしない場合，J2EE サーバ上のメモリを消費したままの状態になります。

(2) Bean クラスの共有についての注意

同じ Session Bean を同じ J2EE アプリケーション内で Stateful Session Bean，および Stateless Session Bean として同時に利用しないでください。

(3) SessionSynchronization のインスタンスの破棄についての注意

SessionSynchronization の beforeCompletion メソッドおよび afterCompletion メソッドでシステム例外が発生した場合，EJB コンテナでは該当する Session Bean のインスタンスを破棄しません。

(4) ベーシックモードでの SessionSynchronization のメソッド呼び出しについての注意

ベーシックモードで，クライアントでトランザクション開始後に SessionSynchronization の Stateful Session Bean（トランザクション属性は Required または Mandatory）を呼び出して，afterBegin メソッドまたは setRollbackOnly メソッ

ドを呼び出すと，クライアントでのコミット時に beforeCompletion メソッドが呼び出されません。1.4 モードのときは，beforeCompletion メソッドが呼び出されます。

(5) setSessionContext メソッドでの javax.transaction.UserTransaction の begin メソッドの呼び出しについて

Stateful Session Bean の setSessionContext メソッドでは，javax.transaction.UserTransaction の begin メソッドを呼び出せる場合がありますが，EJB の仕様上，呼び出して使用することはできません。呼び出さないようにしてください。

(6) beforeCompletion メソッドからの Enterprise Bean の呼び出しについて

J2EE サーバモードの動作モードとしてベーシックモードを使用している場合，Stateful Session Bean の beforeCompletion メソッドからほかの Enterprise Bean は呼び出せません。ほかの Enterprise Bean を呼び出した場合，java.rmi.RemoteException 例外が発生します。

(7) afterCompletion メソッドからの Enterprise Bean の呼び出しについて

Stateful Session Bean の afterCompletion メソッドからほかの Enterprise Bean を呼び出した場合，J2EE サーバモードの動作モードによって次の動作をします。

- ベーシックモード：java.rmi.RemoteException 例外が発生します。
- 1.4 モード：Enterprise Bean を呼び出せます。

EJB の仕様上，Stateful Session Bean の afterCompletion メソッドからほかの Enterprise Bean を呼び出せないので，呼び出さないようにしてください。

4.3.3 Entity Bean（BMP）実装時の注意事項

Entity Bean（BMP）を実装するときの注意事項を示します。

(1) setEntityContext メソッドでのリソースマネジャへのアクセスについて

EJB 1.1 仕様および EJB 2.0 仕様では，setEntityContext メソッドでのリソースマネジャへのアクセスは許されていません。

(2) プライマリキークラスへのインタフェース指定について

BMP Entity Bean の DD の <prim-key-class> タグにインタフェースおよび抽象クラスを指定した場合，クラスを指定したときと同様に，デプロイおよび実行ができます。

(3) remove メソッドによるリファレンスの解放

ホームインタフェースを使用して Entity Bean を呼び出す場合，ホームインタフェース

の create メソッドを呼び出してリファレンスを取得しますが，Entity Bean の呼び出しが完了したあとに，必ず remove メソッドを呼び出してリファレンスを解放してください。

リファレンスを解放しない場合，J2EE サーバ上のメモリを消費したままの状態になります。

4.3.4 Entity Bean（CMP）実装時の注意事項

Entity Bean（CMP）を実装するときの注意事項を示します。

（1） setEntityContext メソッドでのリソースマネジャへのアクセスについて

EJB 1.1 仕様および EJB 2.0 仕様では，setEntityContext メソッドでのリソースマネジャへのアクセスは許されていません。

（2） ユーザ定義型の CMP フィールドの使用についての注意

プライマリキーとして複合プライマリキーを使用する以外で，ユーザ定義型の CMP フィールドを使用できません。

（3） CMR フィールド使用時のトランザクションに関する注意

Collection 型の CMR フィールドや Collection 型の CMR フィールドの Iterator を使用する場合，CMR フィールドを取得したときのトランザクションの範囲内で，CMR フィールドや Iterator へアクセスできます。次のコーディング例の callTeam メソッドでは，CMR フィールドの getter メソッドである getPlayers メソッドと，それに続く Iterator を使用した操作をすべて同一トランザクション内で実行する必要があります（［a］から［b］の間）。

```
public void callTeam() {
          :
  //[a]
  Collection playersInTeam = team.getPlayers();
  Iterator i = playersInTeam.iterator();
  while (i.hasNext()) {
    LocalPlayer p = (LocalPlayer) i.next();
          :
  }
  //[b]
}
```

このコーディング例をトランザクション外で実行した場合は，IllegalStateException 例外が発生します。これを避けるために，トランザクション内で実行されるように CMT などで設定してください。

(4) CMR の cascade-delete の使用についての注意

CMR の cascade-delete を使用する場合，次の制限があります。

- cascade-delete で Entity Bean を remove する場合，呼び出し元となるクライアントプログラムは，最初に remove される Bean の remove メソッドのメソッドパーミッションだけでなく，cascade-delete の対象となるすべての Bean での，コンポーネントインタフェースの remove メソッドのメソッドパーミッションを持つように設定されている必要があります。
- cascade-delete の対象となるすべての Bean での，コンポーネントインタフェースの remove メソッドのトランザクション属性は，Required に設定してください。
- cascade-delete を指定している関係が複数の Bean を循環するように設定されている場合，循環した関係にあるすべての Bean で remove メソッドの実行は保証されません。

(5) プライマリキークラスへのインタフェース指定について

CMP Entity Bean の DD の <prim-key-class> タグにインタフェースおよび抽象クラスを指定した場合，デプロイ時にエラーメッセージ KDJE42039-E が出力され，デプロイ処理はエラー終了されます。

(6) remove メソッドによるリファレンスの解放

ホームインタフェースを使用して Entity Bean を呼び出す場合，ホームインタフェースの create メソッドを呼び出してリファレンスを取得しますが，Entity Bean の呼び出しが完了したあとに，必ず remove メソッドを呼び出してリファレンスを解放してください。

リファレンスを解放しない場合，J2EE サーバ上のメモリを消費したままの状態になります。

(7) EJB QL の finder メソッドまたは select メソッドに関する注意

EJB QL の finder メソッドまたは select メソッドの引数の型として，配列を指定できません。

4.3.5 Message-driven Bean 実装時の注意事項

Message-driven Bean を実装するときの注意事項を示します。

(1) Message-driven Bean のトランザクション設定時の注意（Connector 1.0 に準拠したリソースアダプタを使用する場合）

Message-driven Bean へのメッセージ配信と Message-driven Bean 内のデータベースアクセス処理をトランザクションで同期を取る場合，Message-driven Bean のトランザク

ション設定は CMT で Required に設定します。これによって，トランザクションがロールバックした場合に Message-driven Bean へメッセージが再配信されます。しかし，トランザクションのロールバックが繰り返されると再配信も繰り返されるため，Message-driven Bean 側で javax.jms.Message クラスの getJMSRedelivered メソッドを使用して再配信を確認するなどの対処が必要になります。CMT の NotSupported 設定，または BMT ではいったん Message-driven Bean でメッセージを受信すると，トランザクションがロールバックしても再配信されなくなります。

（2）Message-driven Bean のトランザクション設定時の注意（Connector 1.5 に準拠したリソースアダプタを使用する場合）

TP1 インバウンド連携機能を使用した OpenTP1 と Message-driven Bean 内のリソースアクセス処理で，グローバルトランザクションを使用する場合，Message-driven Bean のトランザクション管理方法は CMT を選択してください。また，トランザクション属性は Required を設定してください。

そのほかの Message-driven Bean にメッセージを配信する EIS と Message-driven Bean 内のリソースアクセス処理で，グローバルトランザクションを使用する場合も，Message-driven Bean のトランザクション管理方法は，CMT を選択してください。また，トランザクション属性は Required に設定してください。

ただし，CJMSP リソースアダプタまたは FTP インバウンドアダプタを使用する場合は，Message-driven Bean へのメッセージ配信と Message-driven Bean 内のリソースアクセス処理で，グローバルトランザクションを使用して同期を取ることはできません。このため，CJMSP リソースアダプタまたは FTP インバウンドアダプタからのメッセージを受信する場合，Message-driven Bean のトランザクション管理方法を BMT にするか，または CMT にしてトランザクション属性を NotSupported に設定してください。

付録

付録 A　uCosminexus Client

付録 B　各バージョンでの主な機能変更

付録 C　このマニュアルの参考情報

付録 D　用語解説

付録 A　uCosminexus Client

uCosminexus Client は，EJB クライアントアプリケーションの実行環境（EJB クライアント環境）を構築するための製品です。アプリケーションサーバ上で動作している J2EE アプリケーション内の Enterprise Bean を，Web サーバ経由ではなく，クライアントマシン上のプログラムから直接呼び出すシステムを構築したい場合に，クライアントとして使用できます。

なお，uCosminexus Client の対象 OS は Windows だけです。

uCosminexus Client は，Application Server に含まれるコンポーネントのうち，クライアント環境に必要なコンポーネント，またはそのサブセットに当たるコンポーネントによって構成されています。

ここでは，uCosminexus Client の機能とインストール方法について説明します。

付録 A.1　uCosminexus Client の機能

uCosminexus Client には，次の機能があります。

- **EJB クライアントアプリケーションの実行**
  Session Bean などを RMI-IIOP 通信で呼び出す Java アプリケーションである，EJB クライアントアプリケーションを実行できます。また，Java アプリケーションを開始できます。
- **性能解析トレース出力**
  EJB クライアントアプリケーションからリクエストを送信するときの性能解析情報を出力できます。出力した性能解析情報は，CSV 形式などに変換して，ほかの J2EE サーバの各機能が出力する性能解析情報とあわせて分析できます。

付録 A.2　インストール手順

ここでは，uCosminexus Client のインストール手順について説明します。

製品のインストールには，HITACHI 総合インストーラを使用します。

uCosminexus Client のインストール手順を次に示します。

1. **uCosminexus Client をインストールします。**
   インストール後の uCosminexus Client のディレクトリ構成については，「付録 A.3 uCosminexus Client のディレクトリ構成」を参照してください。
2. **環境変数を設定します。**
   uCosminexus Client で設定が必要な環境変数については，「3.3.4　EJB クライアントアプリケーションの実行に必要な環境変数の設定」を参照してください。

3. **リソースの見積もりと設定をします。**
   uCosminexus Client をインストールしたマシンで使用するリソースの見積もりと設定をします。使用するリソースの見積もりについては，次に示すマニュアルの個所を参照してください。
   - J2EE アプリケーション実行基盤の場合
     マニュアル「Cosminexus アプリケーションサーバ システム設計ガイド」の「5. 使用するリソースの見積もり（J2EE アプリケーション実行基盤）」
   - バッチアプリケーション実行基盤の場合
     マニュアル「Cosminexus アプリケーションサーバ システム設計ガイド」の「6. 使用するリソースの見積もり（バッチアプリケーション実行基盤）」

インストールが完了したら，EJB クライアントアプリケーションを開始します。EJB クライアントアプリケーションの動作は，直接設定ファイルを編集したり，環境変数を設定したりして設定します。

EJB クライアントアプリケーションは，cjclstartap コマンドなどを使用して開始します。手順については，「3.3　EJB クライアントアプリケーションの開始」を参照してください。

ポイント

uCosminexus Client を使用して EJB クライアント環境を構築する場合は，各ユーザ定義ファイルの格納ディレクトリは，「<Cosminexus のインストールディレクトリ >¥CCL」になります。

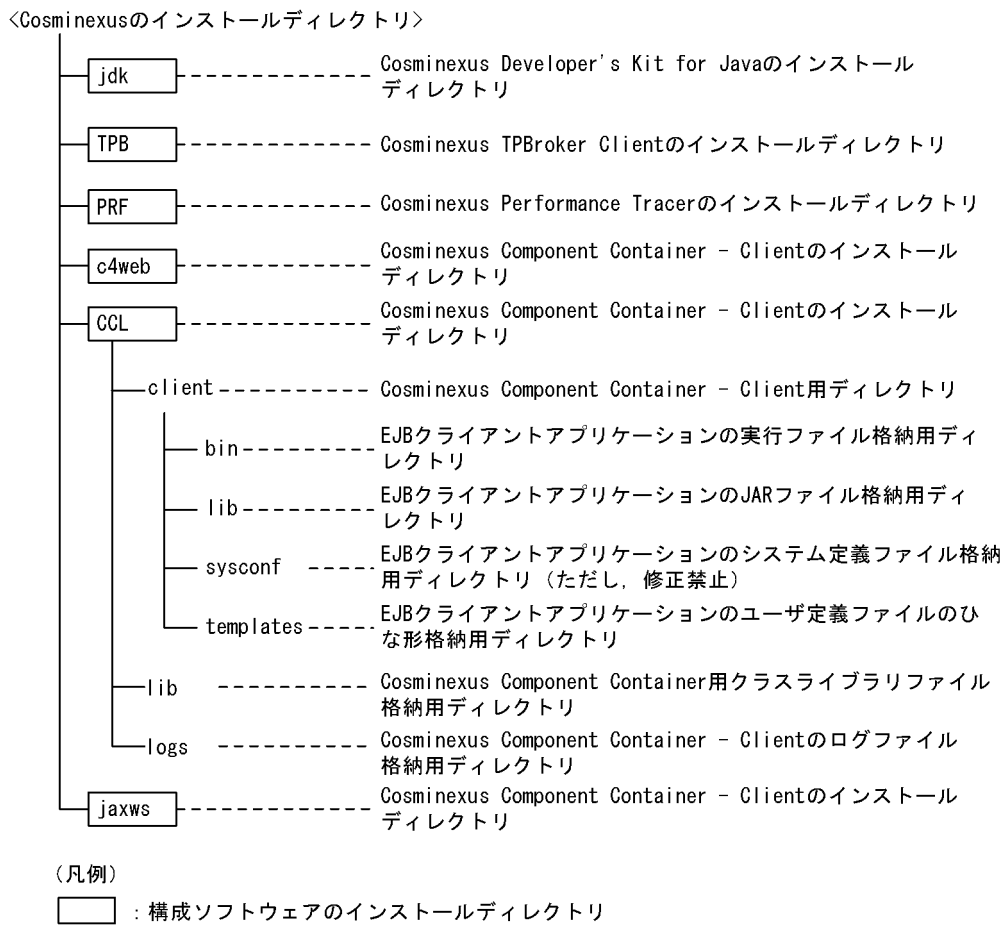
注意事項

Windows Server 2008，Window 7，または Windows Vista で，ログの出力先として C:¥Program Files 以下のディレクトリを指定している場合，EJB クライアントアプリケーションは，管理者特権で実行する必要があります。管理者特権のないユーザが，ログの出力先として C:¥Program Files 以下のディレクトリを指定して，EJB クライアントアプリケーションを実行した場合には，ログは保存されません。この場合，ログは次に示すディレクトリに保存されます。
C:¥Users¥< ユーザ名 >¥AppData¥Local¥VirtualStore
07-50 より前のバージョンで使用していた EJB クライアントアプリケーションを，Windows Server 2008，Window 7，または Windows Vista で使用する場合は，ログの出力先を見直す必要があります。
Windows Server 2008，Window 7，または Windows Vista の場合に管理者特権で実行する方法については，マニュアル「Cosminexus アプリケーションサーバ システム構築・運用ガイド」の「1.9.1　管理者特権で実行する必要がある操作」を参照してください。

付録 A.3　uCosminexus Client のディレクトリ構成

uCosminexus Client と，その構成ソフトウェアである Cosminexus Component Container - Client のディレクトリ構成を次の図に示します。

図 A-1　uCosminexus Client のディレクトリ構成



付録 B　各バージョンでの主な機能変更

ここでは，08-70 よりも前のアプリケーションサーバの各バージョンでの主な機能の変更について，変更目的ごとに説明します。08-70 での主な機能変更については，「1.4　アプリケーションサーバ 08-70 での主な機能変更」を参照してください。

説明内容は次のとおりです。

- アプリケーションサーバの各バージョンで変更になった主な機能と，その概要を説明しています。機能の詳細については，「参照先マニュアル」の「参照個所」の記述を確認してください。「参照先マニュアル」および「参照個所」には，その機能についての主な記載個所を記載しています。
- 「参照先マニュアル」に示したマニュアル名の「Cosminexus アプリケーションサーバ V8」は省略しています。

付録 B.1　08-53 での主な機能変更

(1) 導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 B-1　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
さまざまなハイパーバイザに対応した仮想化環境の構築	さまざまなハイパーバイザを使用して実現する仮想サーバ上に，アプリケーションサーバを構築できるようになりました。 また，複数のハイパーバイザが混在する環境にも対応しました。	仮想化システム構築・運用ガイド	2 章， 3 章， 5 章

(2) 標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 B-2　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
トランザクション連携に対応した OpenTP1 からの呼び出し	OpenTP1 からアプリケーションサーバ上で動作する Message-driven Bean を呼び出すときに，トランザクション連携ができるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	4 章
JavaMail	POP3 に準拠したメールサーバと連携して，JavaMail 1.3 に準拠した API を使用したメール受信機能を利用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	8 章

(3) 信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 B-3　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JavaVM のトラブルシュート機能強化	JavaVM のトラブルシュート機能として，次の機能が使用できるようになりました。 • OutOfMemoryError 発生時の動作を変更できるようになりました。 • JIT コンパイル時に，C ヒープ確保量の上限値を設定できるようになりました。 • スレッド数の上限値を設定できるようになりました。 • 拡張 verbosegc 情報の出力項目を拡張しました。	機能解説 保守／移行／互換編	4 章，5 章，8 章

(4) 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 B-4　運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JP1/ITRM への対応	IT リソースを一元管理する製品である JP1/ITRM に対応しました。	仮想化システム構築・運用ガイド	1.3，2.1

(5) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 B-5　そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
Microsoft IIS 7.0 および Microsoft IIS 7.5 への対応	Web サーバとして Microsoft IIS 7.0 および Microsoft IIS 7.5 に対応しました。	－	－
HiRDB Version 9 および SQL Server 2008 への対応	データベースとして次の製品に対応しました。 • HiRDB Server Version 9 • HiRDB/Developer's Kit Version 9 • HiRDB/Run Time Version 9 • SQL Server 2008 また，SQL Server 2008 に対応する JDBC ドライバとして，SQL Server JDBC Driver に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	3 章

（凡例）－：該当なし。

付録 B.2　08-50 での主な機能変更

(1) 導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 B-6　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Web サービスプロバイダ側での web.xml の指定必須タグの変更	Web サービスプロバイダ側での web.xml で，listener タグ，servlet タグおよび servlet-mapping タグの指定を必須から任意に変更しました。	リファレンス 定義編 (サーバ定義)	2.4
論理サーバのネットワークリソース使用	J2EE アプリケーションからほかのホスト上にあるネットワークリソースやネットワークドライブにアクセスするための機能を追加しました。	機能解説 運用／監視／連携編	1.2.3, 5.2, 5.8
サンプルプログラムの実行手順の簡略化	一部のサンプルプログラムを EAR 形式で提供することによって，サンプルプログラムの実行手順を簡略化しました。	ファーストステップガイド	3.4.1
		システム構築・運用ガイド	付録 F
運用管理ポータルの画面の動作の改善	画面の更新間隔のデフォルトを「更新しない」から「3 秒」に変更しました。	運用管理ポータル操作ガイド	7.4.1
セットアップウィザードの完了画面の改善	セットアップウィザード完了時の画面に，セットアップで使用した簡易構築定義ファイルおよび Connector 属性ファイルが表示されるようになりました。	システム構築・運用ガイド	2.2.12, 3.3
仮想化環境の構築	ハイパーバイザを使用して実現する仮想サーバ上に，アプリケーションサーバを構築する手順を追加しました。※	仮想化システム構築・運用ガイド	3 章, 5 章

注※

08-50 モードで構築する場合は，マニュアル「Cosminexus アプリケーションサーバ 仮想化システム構築・運用ガイド」の「付録 D　08-50 モードの仮想サーバマネージャを利用する場合の設定」を参照してください。

(2) 標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 B-7　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
OpenTP1 からの呼び出しへの対応	OpenTP1 からアプリケーションサーバ上で動作する Message-driven Bean を呼び出せるようになりました。	機能解説 基本・開発編（コンテナ共通機能）	4 章
JMS への対応	JMS1.1 仕様に対応した Cosminexus JMS プロバイダ機能を使用できるようになりました。	機能解説 基本・開発編（コンテナ共通機能）	7 章
Java SE 6 への対応	Java SE 6 の機能が使用できるようになりました。	機能解説 保守／移行／互換編	5.5, 5.8.1
ジェネリクスの使用への対応	EJB でジェネリクスを使用できるようになりました。	このマニュアル	4.2.17

(3) 信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 B-8　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
明示管理ヒープ機能の使用性向上	自動配置設定ファイルを使用して，明示管理ヒープ機能を容易に使用できるようになりました。	システム設計ガイド	7.1.1, 7.6.3, 7.10.5, 7.11.1
		機能解説 拡張編	8 章
データベースセッションフェイルオーバ機能の URI 単位での抑止	データベースセッションフェイルオーバ機能を使用する場合に，機能の対象外にするリクエストを URI 単位で指定できるようになりました。	機能解説 拡張編	6.6
仮想化環境での障害監視	仮想化システムで，仮想サーバの障害を監視して，障害が発生した仮想サーバを停止するための設定を追加しました。	仮想化システム構築・運用ガイド	付録 D

(4) 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 B-9　運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
管理ユーザアカウントの省略	運用管理ポータル，Management Server のコマンド，Smart Composer 機能のコマンドで，ユーザのログイン ID およびパスワードの入力を省略できるようになりました。	システム構築・運用ガイド	7.6.1，7.6.2，7.6.5
		運用管理ポータル操作ガイド	2.2，7.1.1，7.1.2，7.1.3，8.1，8.2.1，付録 G.2
		リファレンス コマンド編	1.4，mngsvrctl（Management Server の起動／停止／セットアップ），mngsvrutil（Management Server の運用管理コマンド），8.3，cmx_admin_passwd（Management Server の管理ユーザアカウントの設定）
仮想化環境の運用	仮想化システムで，複数の仮想サーバを対象にした一括起動・一括停止，スケールイン・スケールアウトなどの運用手順を追加しました。※	仮想化システム構築・運用ガイド	4 章，6 章

注※

08-50 モードで運用する場合は，マニュアル「Cosminexus アプリケーションサーバ 仮想化システム構築・運用ガイド」の「付録 D　08-50 モードの仮想サーバマネージャを利用する場合の設定」を参照してください。

(5) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 B-10　そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
Tenured 領域内不要オブジェクト統計機能	Tenured 領域内で不要となったオブジェクトだけを特定できるようになりました。	機能解説 保守／移行／互換編	8.8
Tenured 増加要因の基点オブジェクトリスト出力機能	Tenured 領域内不要オブジェクト統計機能を使って特定した，不要オブジェクトの基点となるオブジェクトの情報を出力できるようになりました。		8.9
クラス別統計情報解析機能	クラス別統計情報を CSV 形式で出力できるようになりました。		8.10

項目	変更の概要	参照先マニュアル	参照個所
論理サーバの自動再起動回復オーバー数検知によるクラスタ系切り替え	Management Server を系切り替えの監視対象としているクラスタ構成の場合，論理サーバが異常停止状態 (自動再起動回数をオーバーした状態または自動再起動回数の設定が 0 なら障害を検知した状態) になったタイミングでの系切り替えができるようになりました。	機能解説 運用／監視／連携編	20.4.3, 20.5.3, 22.2.2, 22.3.3, 22.3.4
ホスト単位管理モデルを対象とした系切り替えシステム	クラスタソフトウェアと連携したシステム運用で，ホスト単位管理モデルを対象にした系切り替えができるようになりました。		22 章
ACOS（AX2000, BS320）のサポート	使用できる負荷分散機の種類に ACOS（AX2000，BS320）が追加になりました。	システム構築・運用ガイド	4.3.1, 8.2, 8.3, 8.11.6, 付録 A，付録 A.2
		リファレンス 定義編 (サーバ定義)	4.5, 4.6.2, 4.6.4, 4.6.5, 4.6.6, 4.10.1
CMT でトランザクション管理をする場合に Stateful Session Bean（SessionSynchronization）に指定できるトランザクション属性の追加	CMT でトランザクション管理をする場合に，Stateful Session Bean（SessionSynchronization）にトランザクション属性として Supports，NotSupported および Never を指定できるようになりました。	このマニュアル	2.7.3
OutOfMemory 発生時の運用管理エージェントの強制終了	Java VM で OutOfMemory が発生したときに，運用管理エージェントが強制終了するようになりました。	機能解説 保守／移行／互換編	2.5.7
スレッドの非同期並行処理	TimerManager および WorkManager を使用して，非同期タイマ処理および非同期スレッド処理を実現できるようになりました。	機能解説 拡張編	13 章

付録 B.3　08-00 での主な機能変更

(1) 開発生産性の向上

開発生産性の向上を目的として変更した項目を次の表に示します。

表 B-11　開発生産性の向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
ほかのアプリケーションサーバ製品からの移行容易化	ほかのアプリケーションサーバ製品からの移行を円滑に実施するため，次の機能を使用できるようになりました。 • HTTP セッションの上限が例外で判定できるようになりました。 • JavaBeans の ID が重複している場合や，カスタムタグの属性名と TLD の定義で大文字・小文字が異なる場合に，トランスレーションエラーが発生することを抑止できるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.3, 2.7.5
cosminexus.xml の提供	Cosminexus アプリケーションサーバ独自の属性を cosminexus.xml に記載することによって，J2EE アプリケーションを J2EE サーバにインポート後，プロパティの設定をしないで開始できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	10.3

(2) 標準機能への対応

標準機能への対応を目的として変更した項目を次の表に示します。

表 B-12　標準機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Servlet 2.5 への対応	Servlet 2.5 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	2.2, 2.5.4, 2.6，5 章
JSP 2.1 への対応	JSP 2.1 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	2.3.1, 2.3.3, 2.5, 2.6，5 章
JSP デバッグ	MyEclipse を使用した開発環境で JSP デバッグができるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.4
タグライブラリのライブラリ JAR への格納と TLD のマッピング	タグライブラリをライブラリ JAR に格納した場合に，Web アプリケーション開始時に Web コンテナによってライブラリ JAR 内の TLD ファイルを検索し，自動的にマッピングできるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.3.4
application.xml の省略	J2EE アプリケーションで application.xml が省略できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	10.4

項目	変更の概要	参照先マニュアル	参照個所
アノテーションと DD の併用	アノテーションと DD を併用できるようになり，アノテーションで指定した内容を DD で更新できるようになりました。	機能解説 基本・開発編（コンテナ共通機能）	11.5
アノテーションの Java EE 5 標準準拠（デフォルトインターセプタ）	デフォルトインターセプタをライブラリ JAR に格納できるようになりました。また，デフォルトインターセプタから DI できるようになりました。	機能解説 基本・開発編（コンテナ共通機能）	10.4
@Resource の参照解決	@Resource でリソースの参照解決ができるようになりました。	機能解説 基本・開発編（コンテナ共通機能）	11.4
JPA への対応	JPA 仕様に対応しました。	機能解説 基本・開発編（コンテナ共通機能）	5 章，6 章

（3） 信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 B-13　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
セッション情報の永続化	HTTP セッションのセッション情報をデータベースに保存して引き継げるようになりました。	機能解説 拡張編	5 章，6 章
フルガーベージコレクションの抑止	フルガーベージコレクションの要因となるオブジェクトを Java ヒープ外に配置することで，フルガーベージコレクション発生を抑止できるようになりました。	機能解説 拡張編	8 章
クライアント性能モニタ	クライアント処理に掛かった時間を調査・分析できるようになりました。	機能解説 拡張編	9 章

（4） 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 B-14　運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
運用管理ポータルでのアプリケーション操作性向上	アプリケーションおよびリソースの操作について，サーバ管理コマンドと運用管理ポータルの相互運用ができるようになりました。	運用管理ポータル操作ガイド	1.1.4

(5) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 B-15　そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
無効な HTTP Cookie の削除	無効な HTTP Cookie を削除できるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.7.4
ネーミングサービスの障害検知	ネーミングサービスの障害が発生した場合に，EJB クライアントが，より早くエラーを検知できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	2.7
コネクション障害検知タイムアウト	コネクション障害検知タイムアウトのタイムアウト時間を指定できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	3.15.1
Oracle11g への対応	データベースとして Oracle11g が使用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	3 章
バッチ処理のスケジューリング	バッチアプリケーションの実行を CTM によってスケジューリングできるようになりました。	機能解説 拡張編	4 章
バッチ処理のログ	バッチ実行コマンドのログファイルのサイズ，面数，ログの排他処理失敗時におけるリトライ回数とリトライ間隔を指定できるようになりました。	リファレンス 定義編 (サーバ定義)	3.6
snapshot ログ	snapshot ログの収集内容が変更されました。	機能解説 保守／移行／互換編	付録 A.1, 付録 A.2
メソッドキャンセルの保護区公開	メソッドキャンセルの対象外となる保護区リストの内容を公開しました。	機能解説 運用／監視／連携編	付録 C
統計前のガーベージコレクション選択機能	クラス別統計情報を出力する前に，ガーベージコレクションを実行するかどうかを選択できるようになりました。	機能解説 保守／移行／互換編	8.7
Survivor 領域の年齢分布情報出力機能	Survivor 領域の Java オブジェクトの年齢分布情報を日立 JavaVM ログファイルに出力できるようになりました。	機能解説 保守／移行／互換編	8.11
ファイナライズ滞留解消機能	JavaVM のファイナライズ処理の状態を監視して，処理の滞留を解消できるようになりました。	機能解説 保守／移行／互換編	8.12
サーバ管理コマンドの最大ヒープサイズの変更	サーバ管理コマンドが使用する最大ヒープサイズが変更されました。	リファレンス 定義編 (サーバ定義)	5.2, 5.3
推奨しない表示名を指定された場合の対応	J2EE アプリケーションで推奨しない表示名を指定された場合にメッセージが出力されるようになりました。	メッセージ 2	KDJE42374-W

付録 C　このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

付録 C.1　関連マニュアル

アプリケーションサーバのマニュアルについて次に示します。

- Cosminexus **アプリケーションサーバ** V8 **概説**（3020-3-U01）
  アプリケーションサーバの概要について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **ファーストステップガイド**（3020-3-U02）
  Application Server または Developer を使用して，サンプルプログラムを動かすためのシステムを構築する手順について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **システム設計ガイド**（3020-3-U03）
  システム設計時に，システムの目的に応じたシステム構成や運用方法を検討するための指針について説明しています。また，チューニングの方法についても説明しています。
- Cosminexus **アプリケーションサーバ** V8 **システム構築・運用ガイド**（3020-3-U04）
  セットアップウィザードおよび Smart Composer 機能を使用したシステムの構築・運用の手順について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **機能解説 基本・開発編** (Web **コンテナ**)（3020-3-U05）
  アプリケーションサーバで提供する Web コンテナの機能，および Web コンテナに関連する機能（Web サーバ，サーブレット／ JSP など）について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **機能解説 基本・開発編** (**コンテナ共通機能**)（3020-3-U07）
  Web コンテナおよび EJB コンテナで共通して利用する機能について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **機能解説 拡張編**（3020-3-U08）
  アプリケーションサーバで提供する拡張機能（セッションフェイルオーバ機能，バッチサーバ，CTM など）について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **機能解説 運用／監視／連携編**（3020-3-U09）
  アプリケーションサーバで提供する運用・監視機能，およびほかのプログラムとの連携について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **機能解説 保守／移行／互換編**（3020-3-U10）
  アプリケーションサーバで構築したシステムの保守に関する機能，移行情報，および互換用機能について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **アプリケーション設定操作ガイド**（3020-3-U12）
  アプリケーションサーバで動作するアプリケーションの操作方法について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **運用管理ポータル操作ガイド**（3020-3-U13）

運用管理ポータルの使用方法について説明しています。

- Cosminexus **アプリケーションサーバ** V8 **リファレンス コマンド編**（3020-3-U14）
  アプリケーションサーバを構築・運用するときに使用するコマンドについて説明しています。
- Cosminexus **アプリケーションサーバ** V8 **リファレンス 定義編 (サーバ定義)**（3020-3-U15）
  アプリケーションサーバを構築・運用するとき，またはアプリケーションを開発するときに，使用するファイルのうち，J2EE サーバや Management Server などのサーバの定義に使用するファイルの形式について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **リファレンス 定義編 (アプリケーション／リソース定義)**（3020-3-U16）
  アプリケーションサーバを構築・運用するとき，またはアプリケーションを開発するときに使用するファイルのうち，アプリケーションやリソースの属性設定に使用するファイルの形式について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **仮想化システム構築・運用ガイド**（3020-3-U18）
  アプリケーションサーバを仮想化したサーバ上に構築する場合の設計，構築，運用の手順について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **アプリケーション開発ガイド**（3020-3-U25）
  アプリケーションサーバで動作させるアプリケーションの開発方法について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **リファレンス** API **編**（3020-3-U26）
  アプリケーションを開発するときに使用する API の形式について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **メッセージ** 1 KDAL-KDCG **および** Hitachi Web Server **編**（3020-3-U41）
  アプリケーションサーバで出力される KDAL から KDCG までのメッセージ，および Hitachi Web Server のメッセージについて説明しています。
- Cosminexus **アプリケーションサーバ** V8 **メッセージ** 2 KDJE-KDJW **編**（3020-3-U42）
  アプリケーションサーバで出力される KDJE から KDJW までのメッセージについて説明しています。
- Cosminexus **アプリケーションサーバ** V8 **メッセージ** 3 KECX-KEDT ／ KEOS02000-29999 ／ KEUC-KFRM **編**（3020-3-U43）
  アプリケーションサーバで出力される KECX から KEDT までのメッセージ，KEOS02000 から KEOS29999 までのメッセージ，および KEUC から KFRM までのメッセージについて説明しています。
- Cosminexus **アプリケーションサーバ** V8 **メッセージ** 4 **監査ログ編**（3020-3-U44）
  アプリケーションサーバで出力される監査ログメッセージについて説明しています。

また，このマニュアルと関連するこのほかのマニュアルを次に示します。必要に応じてお読みください。

- TPBroker Version 5 トランザクショナル分散オブジェクト基盤 TPBroker ユーザーズガイド（3020-3-U19）
- VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R) プログラマーズリファレンス（3020-3-U29）

なお，このマニュアルでは，次のマニュアルについて，バージョン番号を省略して表記しています。マニュアルの正式名称とこのマニュアルでの表記を次の表に示します。

正式名称	このマニュアルでの表記
Cosminexus アプリケーションサーバ V8 ファーストステップガイド	Cosminexus アプリケーションサーバ ファーストステップガイド
Cosminexus アプリケーションサーバ V8 システム設計ガイド	Cosminexus アプリケーションサーバ システム設計ガイド
Cosminexus アプリケーションサーバ V8 システム構築・運用ガイド	Cosminexus アプリケーションサーバ システム構築・運用ガイド
Cosminexus アプリケーションサーバ V8 機能解説 基本・開発編(Web コンテナ)	Cosminexus アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)
Cosminexus アプリケーションサーバ V8 機能解説 基本・開発編(コンテナ共通機能)	Cosminexus 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)
Cosminexus アプリケーションサーバ V8 機能解説 拡張編	Cosminexus アプリケーションサーバ 機能解説 拡張編
Cosminexus アプリケーションサーバ V8 機能解説 運用／監視／連携編	Cosminexus アプリケーションサーバ 機能解説 運用／監視／連携編
Cosminexus アプリケーションサーバ V8 機能解説 保守／移行／互換編	Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編
Cosminexus アプリケーションサーバ V8 アプリケーション設定操作ガイド	Cosminexus 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
Cosminexus アプリケーションサーバ V8 運用管理ポータル操作ガイド	Cosminexus アプリケーションサーバ 運用管理ポータル操作ガイド
Cosminexus アプリケーションサーバ V8 リファレンス コマンド編	Cosminexus アプリケーションサーバ リファレンス コマンド編
Cosminexus アプリケーションサーバ V8 リファレンス 定義編(サーバ定義)	Cosminexus 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)

正式名称	このマニュアルでの表記
Cosminexus アプリケーションサーバ V8 リファレンス 定義編 (アプリケーション／リソース定義)	Cosminexus 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)
Cosminexus アプリケーションサーバ V8 仮想化システム構築・運用ガイド	Cosminexus アプリケーションサーバ 仮想化システム構築・運用ガイド
Cosminexus アプリケーションサーバ V8 リファレンス API 編	Cosminexus アプリケーションサーバ リファレンス API 編
Cosminexus アプリケーションサーバ V8 メッセージ 1 KDAL-KDCG および Hitachi Web Server 編	Cosminexus アプリケーションサーバ メッセージ 1
Cosminexus アプリケーションサーバ V8 メッセージ 2 KDJE-KDJW 編	Cosminexus アプリケーションサーバ メッセージ 2
TPBroker Version 5 トランザクショナル分散オブジェクト基盤 TPBroker ユーザーズガイド	TPBroker ユーザーズガイド
VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R) プログラマーズリファレンス	Borland(R) Enterprise Server VisiBroker(R) プログラマーズリファレンス

付録 C.2　マニュアル体系の変更について

08-00 では，07-60 のマニュアル「Cosminexus　機能解説」，マニュアル「Cosminexus　システム構築ガイド」およびマニュアル「Cosminexus　システム運用ガイド」の内容を 6 分冊して，マニュアル体系を変更しました。変更後のマニュアル体系については，「1.1　機能の分類」を参照してください。

付録 C.3　このマニュアルでの表記

このマニュアルで使用している表記と，対応する製品名を次に示します。

表記		製品名
ACOS	AX2000	AX2000
	AX2500	AX2500
	BS320	BS320 ロードバランサブレード
Application Server	Application Server Enterprise	uCosminexus Application Server Enterprise
	Application Server Standard	uCosminexus Application Server Standard
Developer	Developer Professional	uCosminexus Developer Professional
	Developer Standard	uCosminexus Developer Standard
Eclipse		Eclipse 3.6.1

表記		製品名
HiRDB		HiRDB Server Version 9
		HiRDB/Parallel Server Version 8
		HiRDB/Single Server Version 8
IPF		Itanium(R) Processor Family
JP1/AJS	JP1/AJS - Agent	JP1/Automatic Job Management System 2 - Agent
		JP1/Automatic Job Management System 3 - Agent
	JP1/AJS - Manager	JP1/Automatic Job Management System 2 - Manager
		JP1/Automatic Job Management System 3 - Manager
	JP1/AJS - View	JP1/Automatic Job Management System 2 - View
		JP1/Automatic Job Management System 3 - View
Oracle	Oracle10g	Oracle 10*g*
		Oracle 10*g* R2
		Oracle Database 10*g*
	Oracle11g	Oracle Database 11*g*
		Oracle Database 11*g* R2
	Oracle9i	Oracle9*i*
		Oracle9*i* R2
UNIX	AIX	AIX 5L V5.3
		AIX V6.1
		AIX V7.1
	HP-UX または HP-UX（IPF）	HP-UX 11i V2（IPF）
		HP-UX 11i V3（IPF）

表記			製品名
	Linux	Linux（IPF）	Red Hat Enterprise Linux(R) AS 4（IPF）
			Red Hat Enterprise Linux(R) 5 Advanced Platform（Intel Itanium）
			Red Hat Enterprise Linux(R) 5（Intel Itanium）
		Linux（x86/AMD64 & Intel EM64T）	Red Hat Enterprise Linux(R) AS 4（x86）
			Red Hat Enterprise Linux(R) ES 4（x86）
			Red Hat Enterprise Linux(R) AS 4(AMD64 & Intel EM64T）
			Red Hat Enterprise Linux(R) ES 4(AMD64 & Intel EM64T）
			Red Hat Enterprise Linux(R) 5 Advanced Platform（x86）
			Red Hat Enterprise Linux(R) 5（x86）
			Red Hat Enterprise Linux(R) 5 Advanced Platform（AMD/Intel 64）
			Red Hat Enterprise Linux(R) 5（AMD/Intel 64）
			Red Hat Enterprise Linux(R) Server 6（32-bit x86）
			Red Hat Enterprise Linux(R) Server 6（64-bit x86_64）
	Solaris		Solaris 10（SPARC）
			Solaris 10（x64）
			Solaris 9（SPARC）
Web Redirector			uCosminexus Web Redirector

なお，Application Server および Developer を総称して，アプリケーションサーバと表記します。

また，Linux に関しては，バージョンごとに次のように表記することがあります。

表記	OS 名
Red Hat Enterprise Linux 4	Red Hat Enterprise Linux(R) AS 4（IPF）
	Red Hat Enterprise Linux(R) AS 4（x86）
	Red Hat Enterprise Linux(R) ES 4（x86）
	Red Hat Enterprise Linux(R) AS 4（AMD64 & Intel EM64T）
	Red Hat Enterprise Linux(R) ES 4（AMD64 & Intel EM64T）
Red Hat Enterprise Linux 5	Red Hat Enterprise Linux(R) 5 Advanced Platform（Intel Itanium）

表記	OS 名
	Red Hat Enterprise Linux(R) 5（Intel Itanium）
	Red Hat Enterprise Linux(R) 5 Advanced Platform（x86）
	Red Hat Enterprise Linux(R) 5（x86）
	Red Hat Enterprise Linux(R) 5 Advanced Platform（AMD/Intel 64）
	Red Hat Enterprise Linux(R) 5（AMD/Intel 64）
Red Hat Enterprise Linux Server 6	Red Hat Enterprise Linux(R) Server 6（32-bit x86）
	Red Hat Enterprise Linux(R) Server 6（64-bit x86_64）

このマニュアルで使用している表記と，対応するアプリケーションサーバの機能名を次に示します。

表記	アプリケーションサーバの機能名
Cosminexus Developer's Kit for Java	Cosminexus Developer's Kit for Java™
Cosminexus RM	Cosminexus Reliable Messaging
CTM	Cosminexus Component Transaction Monitor
Management Server	Cosminexus Management Server
MyEclipse	MyEclipse for Cosminexus
PRF	Cosminexus Performance Tracer
Smart Composer	Cosminexus Smart Composer

このマニュアルで使用している表記と，対応する Java 関連用語を次に示します。

表記	Java 関連用語
Connector 1.0	J2EE™ Connector Architecture 1.0
Connector 1.5	J2EE™ Connector Architecture 1.5
DI	Dependency Injection
EAR	Enterprise ARchive
EJB または Enterprise JavaBeans	Enterprise JavaBeans™
EJB QL	EJB™ Query Language
J2EE または Java 2 Platform, Enterprise Edition	J2EE™
	Java™ 2 Platform, Enterprise Edition
J2SE	Java™ 2 Platform, Standard Edition

表記	Java 関連用語
JAAS	Java™ Authentication and Authorization Service
JAR	Java™ Archive
Java	Java™
Java 2 Runtime Environment, Standard Edition	Java™ 2 Runtime Environment, Standard Edition
Java 2 SDK, Standard Edition	Java™ 2 Software Development Kit, Standard Edition
JavaBeans	JavaBeans™
Java EE または Java Platform, Enterprise Edition	Java™ Platform, Enterprise Edition
Java SE	Java™ Platform, Standard Edition
JavaMail	JavaMail™
Java SE	Java™ Platform, Standard Edition
JavaVM	Java™ Virtual Machine
JDBC	JDBC™
	Java™ Database Connectivity
JDK	JDK™
	Java™ Development Kit
JMS	Java™ Message Service
JNDI	Java Naming and Directory Interface™
JNI	Java™ Native Interface
JSP	JSP™
	JavaServer Pages™
JTA	Java™ Transaction API
Servlet またはサーブレット	Java™ Servlet
WAR	Web ARchive

付録 C.4　英略語

このマニュアルで使用している英略語を次に示します。

英略語	英字での表記
API	Application Programming Interface
BMP	Bean-Managed Persistence
BMT	Bean-Managed Transaction

英略語	英字での表記
CMP	Container-Managed Persistence
CMT	Container-Managed Transaction
CORBA	Common Object Request Broker Architecture
CSV	Comma Separated Value
CUI	Character User Interface
DB	Database
DBMS	Database Management System
DD	Deployment Descriptor
EIS	Enterprise Information System
EL	Expression Language
HA	High Availability
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IIOP	Internet Inter-Orb Protocol
OS	Operating System
OTS	Object Transaction Service
POP3	Post Office Protocol - Version 3
RAC	Real Application Clusters
RDB	Relational Database
RMI	Remote Method Invocation
SPP	Service Providing Program
URL	Uniform Resource Locator
VM	Virtual Machine
XML	Extensible Markup Language

付録 C.5　KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

付録 D　用語解説

（数字）

1.4 モード

サーバの動作モードです。J2EE 1.4 以降の機能を使用できます。データベースを含む複数のリソースのトランザクション管理ができます。

（英字）

Application Server

アプリケーションサーバの実行環境を構築する基盤製品です。Application Server Standard と，Application Server Enterprise の総称です。

BMP（Bean-Managed Persistence）

Enterprise Bean のビジネスメソッド内でデータの永続化を管理するモデルです。

BMT（Bean-Managed Transaction）

Enterprise Bean でトランザクションを管理するモデルです。Session Bean および Message-driven Bean が対象になります。Entity Bean は対象外です。

CMP（Container-Managed Persistence）

EJB コンテナがデータの永続化を管理するモデルです。

CMT（Container-Managed Transaction）

EJB コンテナがトランザクションを管理するモデルです。Session Bean，Entity Bean および Message-driven Bean が対象になります。

CORBA ネーミングサービス

CORBA の仕様に準拠した，リモートオブジェクトの格納場所を管理するためのネーミングサービスです。アプリケーションサーバの構成ソフトウェアである Cosminexus TPBroker によって提供される機能です。

cosminexus.xml

Cosminexus アプリケーションサーバ独自の情報を定義するための属性ファイルです。

Cosminexus Component Container

サーバ・サイドの業務処理プログラム（ビジネスロジック）をコンポーネントとして実行するための構成ソフトウェアです。また，アプリケーションサーバの運用管理をするための Management Server，統合ユーザ管理，snapshot ログ収集などの機能も提供しています。

Cosminexus Component Transaction Monitor

クライアントからのリクエストのスケジューリングを実現するための構成ソフトウェアです。

Cosminexus Performance Tracer

リクエストが処理されるときに，決められたポイントごとに各機能が出力する性能解析情報をファイルに出力するための構成ソフトウェアです。

Cosminexus Reliable Messaging

アプリケーションサーバで構築したシステム上の J2EE 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構成ソフトウェアです。JMS インタフェースでのメッセージ通信機能を J2EE アプリケーションに提供します。

Cosminexus アプリケーションサーバ

アプリケーションサーバを中核とした，性能および信頼性の高いアプリケーションを実行および開発するためのシステム構築基盤製品です。

DD

アプリケーションを運用環境に配置するときの定義情報を記述したものです。

EAR ファイル

J2EE アプリケーションを構成する複数の EJB-JAR ファイル，WAR ファイル，および DD を EAR ファイル形式でパッケージ化したものです。

EIS

データベースやトランザクションサーバなど，企業内に構築されているバックエンドシステムです。

EJB（Enterprise JavaBeans）

業務ロジックをプログラムとして記述したビジネスロジック，および永続的データを格納するオブジェクトであるエンティティを Java コンポーネント化したものです。

EJB クライアント

J2EE サーバ上で開始されている Enterprise Bean を呼び出すクライアントプログラムです。次の 3 種類があります。

- EJB クライアントアプリケーション
- サーブレットまたは JSP などの Web アプリケーション
- ほかの Enterprise Bean

EJB クライアントアプリケーション

Enterprise Bean を呼び出す Java アプリケーションです。

EJB コンテナ

Enterprise Bean を制御する実行環境です。また，通信，トランザクション管理などのシステムレベルのサービスも提供します。Enterprise Bean の実体は，EJB コンテナの中で実行されます。

EJB タイマ

EJB コンテナの Timer Service の機能を利用して，タイムアウト時間を管理するタイマです。指定された時刻または間隔で，タイムアウトメソッドをコールバックし，イベントを実行します。

Inbound

Connector 1.5 仕様で定められた J2EE サーバとリソースアダプタ間の通信モデルの一つです。リ

ソースアダプタから J2EE サーバにアクセスする通信モデルです。

J2EE アプリケーション

JSP，サーブレット，EnterpriseBean などで構成されるアプリケーションです。アプリケーションサーバで扱う J2EE アプリケーションの形式には，EAR ファイル形式でパッケージ化されたアーカイブ形式のアプリケーションと，アーカイブ化しない展開ディレクトリ形式のアプリケーションがあります。EAR ファイル形式でパッケージ化されたアプリケーションの場合，複数の EJB-JAR ファイル，複数の WAR ファイル，および一つの DD から構成されます。

J2EE コンポーネント

サーブレット，JSP，Enterprise Bean などのユーザアプリケーションプログラムのことです。

J2EE サーバ

J2EE コンテナを生成，実行する環境です。

J2EE サーバモード

J2EE サーバの動作モードの一つです。このモードの場合，J2EE コンテナ上で動作するアプリケーションから，J2EE 関連の API を利用できます。なお，J2EE サーバモードには，1.4 モードと，互換用の動作モードであるベーシックモードがあります。

J2EE サービス

J2EE コンテナの部品機能として利用されます。J2EE コンポーネントに JNDI，JDBC，JTA，RMI-IIOP および JavaMail の API を提供します。

J2EE リソース

サーバから利用できるリソースです。J2EE サーバの場合，データベース，OpenTP1，SMTP サーバなどを利用できます。また，バッチサーバの場合，データベースを利用できます。

JNDI（Java Naming and Directory Interface）

Java プラットフォーム用の標準拡張機能で，Java テクノロジに対応したアプリケーションに，企業内の複数のネーミングおよびディレクトリサービスへの統一したインタフェースを提供します。アプリケーションサーバでは，Cosminexus TPBroker の CORBA ネーミングサービスを利用して，JNDI の機能を実現しています。

JNDI 名前空間

JNDI によってアクセスする名前情報が管理されている名前空間です。Application Server では，J2EE アプリケーションをデプロイした時に，EJB オブジェクトリファレンスが JNDI 名前空間の名前にバインドされます。
アプリケーションサーバで使用する JNDI 名前空間の名前には，別名を付けることができます。これによって，Enterprise Bean や J2EE リソースを別名でルックアップできるようになります。

Management アクション

Management イベント発生時に自動的に実行される処理です。Management アクションとしてどのような処理をするかは，Management Server 側であらかじめ定義しておきます。

Management イベント

Management Server を利用して運用している場合に，運用管理ドメイン内の論理サーバで発生した事象に応じて Management Server で発行するイベントです。論理サーバでメッセージが出力されたタイミングで発行できます。例えば，リソース監視中に設定したしきい値を超えた場合などに発行できます。
Management イベントに対して Management アクションを定義しておくと，発生した事象に応じたアクションを自動的に実行できます。

OTS（Object Transaction Service）

1.4 モードで，TPBroker OTS による分散トランザクションを使用するために必要なサービスです。

Timer Service

EJB コンテナで提供される機能です。指定した時刻，経過時間，または間隔で，EJB コンテナがタイムアウトメソッドをコールバックします。

uCosminexus Batch Job Execution Server

オープン環境で基幹系のバッチ業務を実行・運用するためのバッチジョブ実行基盤を提供する製品です。JP1/AJS2 とあわせて使用します。

uCosminexus Client

EJB クライアント環境を構築するための製品です。

Web アプリケーション

Web ブラウザを備えたクライアントを対象に作成されたアプリケーションです。具体的には，サーブレットプログラム，JSP ファイル，HTML/XML ドキュメントなどの集合体です。

Web コンテナ

Web アプリケーションを制御する実行環境です。また，セキュリティ，トランザクションなどの各種サービスも提供します。Web アプリケーションは，Web コンテナ上で動作します。
Java Servlet2.5 仕様，および JavaServer Pages Specification v2.1 仕様に準拠した Web アプリケーションを実行できます。

Web サーバ

Web ブラウザからのリクエスト受信および Web ブラウザへのデータ送信に関連する処理を実行するプログラムです。アプリケーションサーバでは，Hitachi Web Server，Microsoft IIS，またはインプロセス HTTP サーバを使用できます。インプロセス HTTP サーバは，J2EE サーバプロセス内で動作する Web サーバです。
なお，Management Server を利用する場合，Hitachi Web Server は論理サーバとして扱えます。

Web サーバ連携

アプリケーションサーバで使用する Web サーバとして，Hitachi Web Server または Microsoft IIS を使用する方法です。
Hitachi Web Server または Microsoft IIS に Cosminexus Component Container が提供するリダイレクタモジュールを組み込んで使用します。

（ア行）

アノテーション
ソースコードにクラスやメソッドの付加情報などを埋め込むための記述方式です。

アプリケーションサーバ
情報システムの中間に位置し，ユーザの要求（プレゼンテーション層）とデータベースなどの業務システム（データ層）の処理を橋渡しするためのアプリケーション層を構築するためのミドルウェアです。

運用管理コマンド（mngsvrutil）
Management Server を操作するための CUI です。

（カ行）

稼働情報収集機能
J2EE サーバやバッチサーバ内の各機能の稼働情報を定期的に収集し，ファイルに出力する機能です。出力されたファイルを稼働情報ファイルといいます。

監査証跡
データベースの機能で出力する，データベースに対する操作記録のことです。データベースに対する権限の運用が適切に行われているかどうかをチェックするために使用します。アプリケーションサーバでは，監査証跡にアプリケーションサーバで構築したシステムの情報を出力するためのデータベース監査証跡連携機能を提供しています。

監査ログ
システム構築者やシステム運用者がアプリケーションサーバのプログラムに対して実行した操作，およびその操作に伴うプログラムの動作の履歴が出力されるファイルです。監査者が監査ログを調査することで，「いつ」「だれが」「何をしたか」を知ることができて，システムの運用が法規制，セキュリティ評価基準，または業界ごとの各種の基準に準拠していることを証明できます。

管理対象オブジェクト（AdminObject）
Connector 1.5 仕様で定義された，キューやトピックなどを使用したメッセージの送受信で使用するオブジェクトです。メッセージ送信や，同期または非同期でのメッセージ受信でのメッセージ変換で使用します。
管理対象オブジェクトは，メッセージプロバイダごとに特有のオブジェクトです。リソースアダプタが使用する管理対象オブジェクトは，リソースアダプタの DD（ra.xml）の <adminobject> タグで指定します。

クライアントアプリケーション情報
性能解析トレースに出力される情報です。次に示す Enterprise Bean を呼び出す単位で設定される情報です。
- Web コンテナから EJB コンテナの呼び出し
- EJB クライアントから EJB コンテナの呼び出し
- EJB コンテナから EJB コンテナの呼び出し

クラスタ

ある共通の機能を提供するサーバの集合です。
Management Server では，J2EE サーバクラスタまたは Web サーバクラスタを論理サーバとして扱えます。

グローバル CORBA ネーミングサービス

CTM によってリクエストをスケジューリングする場合に，CTM ドメイン内に含まれる複数の J2EE サーバに登録されている業務処理プログラム（Stateless Session Bean）の情報を共有管理するネーミングサービスです。

グローバルトランザクション

J2EE サーバが提供するトランザクションマネジャによって管理されるトランザクションです。2 フェーズコミットプロトコルを使用できるので，トランザクションに複数のリソースを参加させることができます。

コミットオプション

CMP フィールドのキャッシュ方法と Entity Bean の状態遷移のキャッシュモデルを指定するオプションです。
Full caching の場合，Entity Bean は前回のトランザクションコミット時と同じ状態で開始されます。
Caching の場合，Entity Bean はデータベースの最新状態と同じ状態で開始されます。
No caching の場合，Entity Bean はデータベースの最新状態と同じ状態で開始され，かつ，トランザクションコミット時に Entity Bean は非活性になります。

コンテナ拡張ライブラリ

ユーザ作成ライブラリのインタフェースおよび機能を，J2EE コンテナ，Web コンテナ，またはバッチサーバから利用できるように拡張したライブラリです。Enterprise Bean，サーブレットおよび JSP から共通して利用できます。

（サ行）

サーバ管理コマンド

サーバで管理しているアプリケーションおよびリソースの設定をするためのコマンド群です。

性能解析トレース

アプリケーションサーバで構築したシステムの性能を解析するためのトレース情報です。

セッション

Web アプリケーションに対する一連の作業を示す単位です。セッションは通常，Web クライアントから Web サーバへの複数のリクエストの集合から構成されます。

（タ行）

データソース

JDBC を使用してデータベースに接続する機能です。

デプロイ

J2EE アプリケーションの場合，J2EE サーバ内にインポートした J2EE アプリケーションを，クライアントから実行可能な状態にすることです。
J2EE リソースアダプタの場合，J2EE サーバ内にインポートした J2EE リソースアダプタを，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可能な状態にすることです。

統合ユーザ管理

アプリケーションサーバで構築したシステムにログインするユーザを統合管理するための仕組みです。ユーザ認証を実現する統合ユーザ管理フレームワークが使用できます。また，運用管理ポータルを使用して，リポジトリ管理およびリソース監視を実現できます。

動作モード

サーバの動作モードです。J2EE サーバモードとサーブレットエンジンモードがあります。J2EE サーバモードには，さらに，1.4 モード，ベーシックモードがあります。
ただし，ベーシックモードおよびサーブレットエンジンモードは旧バージョンとの互換用の動作モードとなります。

（ナ行）

ネーミング切り替え

JNDI を介して EJB ホームオブジェクトの登録，削除，検索をする時に，JNDI が接続する CORBA ネーミングサービスを切り替えることです。

ネーミングサービス

オブジェクトに名前を付けて格納場所を管理しておくことで，格納先を知らなくても名前からそのオブジェクトを利用できるようにするサービスです。アプリケーションサーバでは CORBA ネーミングサービスを利用します。
Management Server では，ネーミングサービスを論理サーバとして扱えます。

（ハ行）

ビジネスインタフェース

Enterprise Bean を呼び出すためのビジネスメソッドを定義するインタフェースです。

日立トレース共通ライブラリ

日立が提供するトレース採取用のライブラリです。複数の製品で共通のライブラリを利用するので，各製品のトレース形式が統一できます。

ベーシックモード

旧バージョンとの互換性を確保するためのサーバの動作モードの一つです。
単一リソースとのトランザクション管理ができます。

別名

JNDI 名前空間に登録する EJB ホームオブジェクトリファレンスや，J2EE リソースに付けられる任意の名前です。ユーザ指定名前空間機能を使用している場合，別名を付けられます。Optional

Name ともいいます。

（マ行）

未決着トランザクション

決着していない，仕掛かり中のトランザクションです。

メソッドタイムアウト機能

J2EE アプリケーション実行時間の監視機能の機能の一つです。監視基盤にあるリクエストのうち，一定時間内に終了しなかったメソッド処理を，タイムアウトとしてユーザに通知します。

面数

ログファイルを保持する数です。ログファイルは面数分作成され，循環して使用されます。

（ヤ行）

ユーザ指定名前空間機能

JNDI 名前空間に登録する EJB ホームオブジェクトリファレンスや J2EE リソースの名前に，ユーザが任意の別名を付けられる機能です。別名を付与することによって，CTM によってリクエストを振り分けたり，負荷分散機やロードバランサによって負荷分散したりする場合に，サーバ名やアプリケーション名に依存しない名前でルックアップできるようになります。

ユーザログ

J2EE アプリケーション，バッチアプリケーション，または EJB クライアントアプリケーションが出力するログのことです。

（ラ行）

ラウンドロビン検索

複数の CORBA ネーミングサービス上にある同一名称の EJB ホームオブジェクトを，ラウンドロビンポリシーに従ってルックアップする検索機能のことです。

リクエストのスケジューリング

クライアントからのリクエストを実行するサーバを負荷に応じて的確に振り分けたり，サーバに送信するリクエストの数を制限したりする機能です。

リソースアダプタ

J2EE Connector Architecture によって，J2EE サーバまたはバッチサーバと，EIS を接続するための接続機能です。
アプリケーションサーバで構築したシステムでは，データベースに接続するためのリソースアダプタである DB Connector および DB Connector for Cosminexus RM を提供しています。また，OpenTP1 の SPP と接続するためのリソースアダプタである uCosminexus TP1 Connector，TP1/Message Queue と接続するためのリソースアダプタである TP1/Message Queue - Access，データベース上に実現したキューに接続するためのリソースアダプタである Cosminexus RM も使用でき

ます。

リソースマネジャ

リソースを管理する機能です。DBMS などが該当します。

リデプロイ機能

J2EE アプリケーションを入れ替えるときに使用する機能です。リデプロイ機能による入れ替えは，J2EE アプリケーションのテスト時などに，修正した J2EE アプリケーションと動作中の J2EE アプリケーションを入れ替えたいときに使用します。

リモートインタフェース

Java RMI のインタフェース規定に従い，RMI-IIOP 通信によって Enterprise Bean の呼び出しをするインタフェースです。
リモートホームインタフェースとリモートビジネスインタフェースが含まれます。

ローカルインタフェース

Enterprise Bean の呼び出しを Java のメソッド呼び出しで実行するインタフェースです。通信は発生しません。
ローカルホームインタフェースとローカルビジネスインタフェースが含まれます。

ローカルトランザクション

接続先のリソースマネジャによって管理されるトランザクションです。単一のリソースだけがトランザクションに参加できます。

ロール

コンテキストに対し，アクセス制御するときに使用される単位です。ロールはグループごとに定義されます。また，アクセス制御するコンテキストについては，そのコンテキストにアクセスするのに必要なロールが定義されます。アクセスしたユーザの持つロールがコンテキストに定義されたロールと一致した場合，そのコンテキストへのアクセスは成功します。

索引

数字

1.4 モード 203

A

afterCompletion メソッドからの Enterprise Bean の呼び出しについて 177
Application Server 203

B

Bean クラスの共有についての注意 175,176
beforeCompletion メソッドからの Enterprise Bean の呼び出しについて 177
BMP 22
BMP（Bean-Managed Persistence） 203
BMT 46
BMT（Bean-Managed Transaction） 203

C

Caching（commit option B） 58
cjclstartap 125
CMP 22
CMP（Container-Managed Persistence） 203
CMR の cascade-delete の使用についての注意 179
CMR フィールド使用時のトランザクションに関する注意 178
CMT 47
CMT（Container-Managed Transaction） 203
CORBA ネーミングサービス 203
cosminexus.xml 203
Cosminexus Component Container 203
Cosminexus Component Transaction Monitor 203
Cosminexus Performance Tracer 204
Cosminexus Reliable Messaging 204
Cosminexus アプリケーションサーバ 204

D

DD 204

E

EAR ファイル 204
EIS 204
EJB（Enterprise JavaBeans） 204
EJB QL の finder メソッドまたは select メソッドに関する注意 179
ejbserver.container.rebindpolicy 104
ejbserver.container.security.disabled 64
ejbserver.ejb.timerservice.maxCallbackThreads 97
ejbserver.ejb.timerservice.retryCount 97
ejbserver.ejb.timerservice.retryInterval 97
ejbserver.rmi.localinvocation.scope 104
ejbserver.rmi.passbyreference 104
ejbserver.rmi.request.timeout 71
EJB クライアント 122,204
EJB クライアントアプリケーション 122,204
EJB クライアントアプリケーションでのセキュリティの実装 141
EJB クライアントアプリケーションでのトランザクションの実装 136
EJB クライアントアプリケーションのクラスパスへの JAR ファイルの設定 146
EJB クライアントアプリケーションの実行に必要な環境変数の設定 129
EJB クライアントアプリケーションの使用形態 154
EJB クライアントアプリケーションの使用形態とシステムプロパティ 154
EJB コンテナ 204
EJB コンテナでのタイムアウトの設定 65
EJB 仕様準拠のチェック 34
EJB タイマ 204
EJB タイマキャンセル 84
EJB タイマ生成 78

EJB の仕様に関する注意 170
EJB のリモートインタフェースでのローカル呼び出しの最適化 99
EJB のリモートインタフェースの値の参照渡し 100
EJB のリモートインタフェースの通信障害発生時の動作 101
EJB のリモートインタフェースの呼び出し 99
EJB ホームオブジェクトのリファレンスの検索と取得 133
EJB ホームオブジェクトのリファレンスを検索して Enterprise Bean を呼び出す方法 133
Enterprise Bean 21
Enterprise Bean および関連するクラスの命名規則 162
Enterprise Bean 共通の注意事項 162
Enterprise Bean でのトランザクション管理 45
Enterprise Bean のインタフェース 24
Enterprise Bean の実行 21
Enterprise Bean の種別ごとに指定できるトランザクション属性 53
Enterprise Bean の種類 21
Enterprise Bean の生成とメソッドの呼び出し 133
Enterprise Bean のプールの管理 60
Enterprise Bean のライフサイクル 28
Enterprise Bean へのアクセス制御 63
Enterprise Bean へのアクセス制御の抑止 63
Enterprise Bean へのアクセス制御の抑止オプション 63
Entity Bean 22
Entity Bean（BMP）実装時の注意事項 177
Entity Bean（CMP）実装時の注意事項 178
Entity Bean（CMP，BMP 共通）使用時のデッドロックの発生について 168
Entity Bean（CMP，BMP 共通）属性ファイルの <prim-key-class> タグについて 170
Entity Bean（CMP，BMP 共通）のアクセス排他のタイムアウトについて 168
Entity Bean の EJB オブジェクトのタイムアウト 66
Entity Bean のキャッシュモデル 58
Entity Bean のプーリング 60
Entity Bean のライフサイクル 31

F

Full caching（commit option A） 58

I

Inbound 204
IP アドレスの固定 108

J

J2EE アプリケーション 205
J2EE コンポーネント 205
J2EE サーバ 205
J2EE サーバモード 205
J2EE サービス 205
J2EE リソース 205
javax.ejb.EJBContext インタフェースメソッドについての注意事項 169
JNDI（Java Naming and Directory Interface） 205
JNDI 名前空間 205
JNDI ネーミングコンテキストの生成 133

M

Management アクション 205
Management イベント 206
Mandatory 属性 51
Message-driven Bean 22
Message-driven Bean 実装時の注意事項 179
Message-driven Bean のトランザクション設定時の注意（Connector 1.0 に準拠したリソースアダプタを使用する場合） 179

Message-driven Bean のトランザクション設定時の注意（Connector 1.5 に準拠したリソースアダプタを使用する場合） 180
Message-driven Bean のプーリング 61
Message-driven Bean のライフサイクル 33

N

Never 属性 52
No caching（commit option C） 59
NotSupported 属性 47

O

OTS（Object Transaction Service） 206

P

pool 状態の Entity Bean 61

R

ready 状態の Entity Bean 61
remove メソッドによるリファレンスの解放 175,177,179
Required 属性 48
RequiresNew 属性 50
RMI-IIOP 通信のタイムアウト 67

S

Session Bean 21
Session Bean のライフサイクル 28
SessionSynchronization のインスタンスの破棄についての注意 176
setEntityContext メソッドでのリソースマネジャへのアクセスについて 177,178
setSessionContext メソッドでの javax.transaction.UserTransaction の begin メソッドの呼び出しについて 177
Stateful Session Bean 22
Stateful Session Bean 実装時の注意事項 176
Stateful Session Bean のタイムアウト 66
Stateful Session Bean のライフサイクル 29
Stateless Session Bean 22
Stateless Session Bean 実装時の注意事項 175
Stateless Session Bean のプーリング 60
Stateless Session Bean のライフサイクル 28
Supports 属性 49

T

Timer Service 74,206
Timer Service を使用するアプリケーションの実装 92

U

uCosminexus Batch Job Execution Server 206
uCosminexus Client 206
Unicode の補助文字の送受信に関する注意 170
URLConnection クラス使用時の注意 168

V

vbj 126
vbroker.se.iiop_tp.host 109
vbroker.se.iiop_tp.scm.iiop_tp.listener.port 109

W

Web アプリケーション 206
Web コンテナ 206
Web サーバ 206
Web サーバ連携 206

あ

アノテーション 207
アプリケーションサーバ 207
アプリケーションサーバ 08-70 での主な機能変更 13

い

インスタンス取得待ちのタイムアウト 67

インターセプタ 110
インターセプタの実行順序 114

う

運用管理コマンド（mngsvrutil） 207

か

外部リソースとの接続 44
型変数 172
稼働情報収集機能 207
監査証跡 207
監査ログ 207
管理対象オブジェクト（AdminObject） 207

く

クライアントアプリケーション情報 207
クラスタ 208
クラスレベルインターセプタ 110
クラスローダの取得に関する注意 167
グローバル CORBA ネーミングサービス 208
グローバルトランザクション 208

こ

コミットオプション 208
コンテナ拡張ライブラリ 208

さ

サーバ管理コマンド 208
サブディレクトリ共有モード 150
サブディレクトリ専有モード 150

し

システムログの出力先や出力レベルの変更 152

せ

性能解析トレース 208
セッション 208

た

タイムアウトメソッド 75
タイムアウトを設定できる RMI-IIOP 通信 67

つ

通信ポートの固定 108

て

データソース 208
デフォルトインターセプタ 110
デプロイ 209

と

統合ユーザ管理 209
動作モード 209
トランザクション属性の種類 47

ね

ネイティブライブラリのロードに関する注意 168
ネーミング切り替え 209
ネーミングサービス 209

は

パラメタ化 172

ひ

ビジネスインタフェース 209
ビジネスインタフェースに対応する機能 26
ビジネスインタフェースの Enterprise Bean 呼び出し 27
ビジネスインタフェースのリファレンスを検索して Enterprise Bean を呼び出す方法 134
日立トレース共通ライブラリ 209

ふ

プライマリキークラスへのインタフェース指定について 177,179

へ

ベーシックモード 209
ベーシックモードでの SessionSynchronization のメソッド呼び出しについての注意 176
別名 209

ほ

ほかの J2EE アプリケーション内にある Enterprise Bean をコンポーネントインタフェースによって呼び出す方法 164
ほかの J2EE アプリケーション内にある Enterprise Bean をビジネスインタフェースによって呼び出す方法 166

み

未決着トランザクション 210

め

メソッドタイムアウト機能 210
メソッドレベルインターセプタ 110
面数 210

ゆ

ユーザ指定名前空間機能 210
ユーザ定義型の CMP フィールドの使用についての注意 178
ユーザログ 210

ら

ラウンドロビン検索 210

り

リクエストのスケジューリング 210
リソースアダプタ 210
リソースのコネクションの取得と解放 163
リソースマネジャ 211
リデプロイ機能 211
リモートインタフェース 25,211
リモートインタフェースでの通信障害発生時の EJB クライアントの動作 104
リモートインタフェースの参照渡し機能 104

る

ルックアップを使用した UserTransaction の取得方法 138

ろ

ローカルインタフェース 26,211
ローカルインタフェースとリモートインタフェースの使い分け 163
ローカルトランザクション 211
ローカル呼び出し最適化機能の範囲 104
ローカル呼び出し最適化機能の利用について 164
ロール 211
ログ出力先のサブディレクトリ数の管理 156