HITACHI
Inspire the Next

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

操作書

3020-3-U12-40

■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

※印の製品については，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

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。
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。
GIF は，米国 CompuServe Inc. が開発したフォーマットの名称です。
GIF は，米国 CompuServe Inc. のサービス名称です。
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。
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。
JSTL は，The Jakarta Project のタグライブラリ名称です。
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。
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。
ORACLE 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。
Oracle 及び Oracle 10g 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。
Oracle 及び Oracle8i 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。

■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) SQL Server 2000	SQL Server 2000	SQL Server	
Microsoft(R) SQL Server 2005	SQL Server 2005		
Microsoft(R) SQL Server 2008	SQL Server 2008		
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）			

製品名	表記	
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	
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	
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	

製品名	表記	
Microsoft(R) Windows(R) XP Professional Operating System	Windows XP	

■発行

2011 年 7 月 3020-3-U12-40

■著作権

変更内容

変更内容（3020-3-U12-40）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
インターセプタの設定に関する説明を変更・追加した。	9.1(2)，9.21
SPI の注意事項を追加した。	11.4
次の属性設定ページの別名に指定できる文字にハイフン（-）を追加した。 • Connector 属性設定 • Session Bean 属性設定 • Entity Bean 属性設定	21.2.2(12)，21.6.2(21)，21.7.2(2)
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），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	－

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	変更個所
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。	4.2，4.2.1，4.2.2(4)
HiRDB Version 9 に対応した。	－
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	－

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-U12-20）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
Connector 1.5 対応のリソースアダプタを提供したため，アプリケーションサーバで提供しているリソースアダプタの仕様についての説明を変更した。
Management Server リモート管理機能へのログインで，管理ユーザアカウントの入力を省略できるように変更した。
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
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

はじめに

このマニュアルはサーバ管理コマンドまたは Server Plug-in を使用して Cosminexus（コズミネクサス）のアプリケーションサーバを構築する際に必要な，サーバ管理コマンドの操作方法および Server Plug-in の画面の詳細について説明したものです。
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 Server Plug-in
- 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構築する方を対象にしています。
次の内容を理解されていることを前提としています。

- Windows またはご使用の UNIX のシステム構築に関する知識
- 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
- Eclipse に関する基本的な知識

■ご利用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

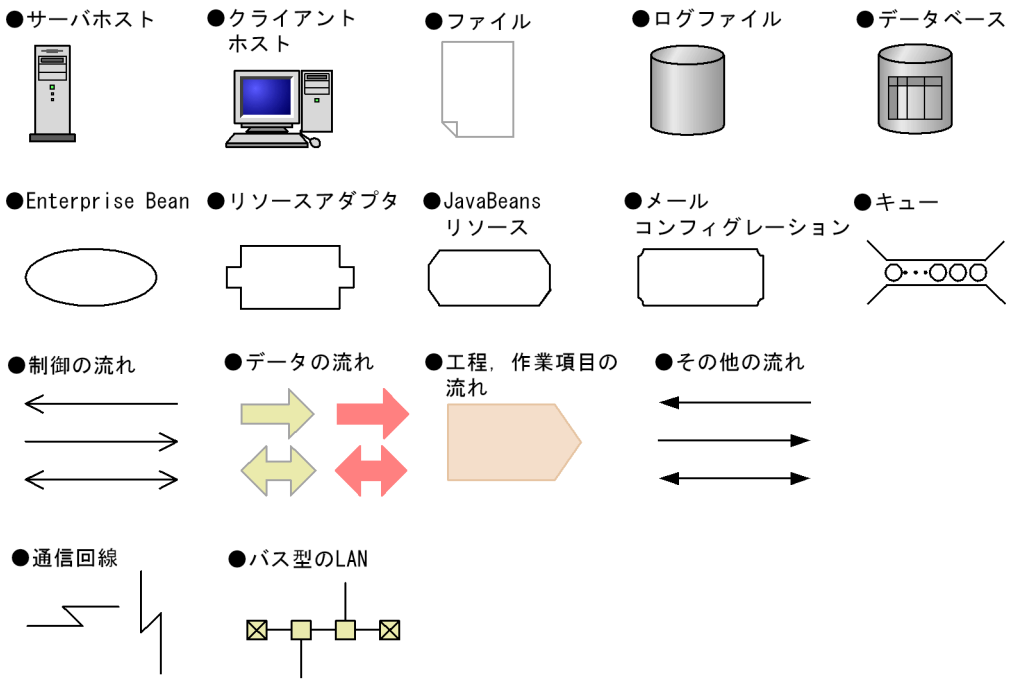
要があります。
次の表に従って，マニュアルで使用している用語をご利用の製品名に読み替えてください。

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。



■文法で使用している記号

このマニュアルの文法で使用している記号について次に示します。

記号	意　味
｜	横に並べられた複数の項目に対する項目間の区切りを示し，「または」を意味します。 （例） A ｜ B A または B を指定することを示します。
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを示します。項目が横に並べられ，記号｜で区切られている場合は，そのうちの一つを選択します。 （例） {A ｜ B ｜ C} A，B または C のどれかを指定することを示します。
[ ]	この記号で囲まれている項目は省略してもよいことを示します。複数の項目が横に並べて記述されている場合には，すべてを省略するか，記号 { } と同じくどれか一つを選択します。 （例１） [A] 「何も指定しない」か「A を指定する」ことを示します。 （例２） [B ｜ C] 「何も指定しない」か「B または C を指定する」ことを示します。
< >	この記号で囲まれている項目は，属性ファイルの該当する要素（タグ）を示します。 （例） <abc-xyz> abc-xyz タグを示します。
< >-< >	タグの階層を示します。- の前に上位のタグを示します。
＜　＞	この記号で囲まれている項目は，ユーザの環境によって異なることを示します。 （例） ＜プロパティ名＞ プロパティを示す名称を示します。

■ CUI および GUI の説明で使用している記号

このマニュアルの CUI および GUI の説明で使用している記号について次に示します。

記号	意味
[　]	画面の名称および画面に表示されている項目を示します。
[　]-[　]	- の前に示したメニューから，- の後ろのメニューを選択することを示します。
＜　＞	＜＞内の名称がユーザの環境によって異なることを示します。
「　」	入力値，可変値，またはメッセージなどを示します。

目次

第 1 編　概要

第 2 編　サーバ管理コマンドによる操作

1 アプリケーション設定操作の概要

この章では，アプリケーション設定操作の目的と位置づけについて説明します。また，アプリケーション設定操作でできることについても説明します。

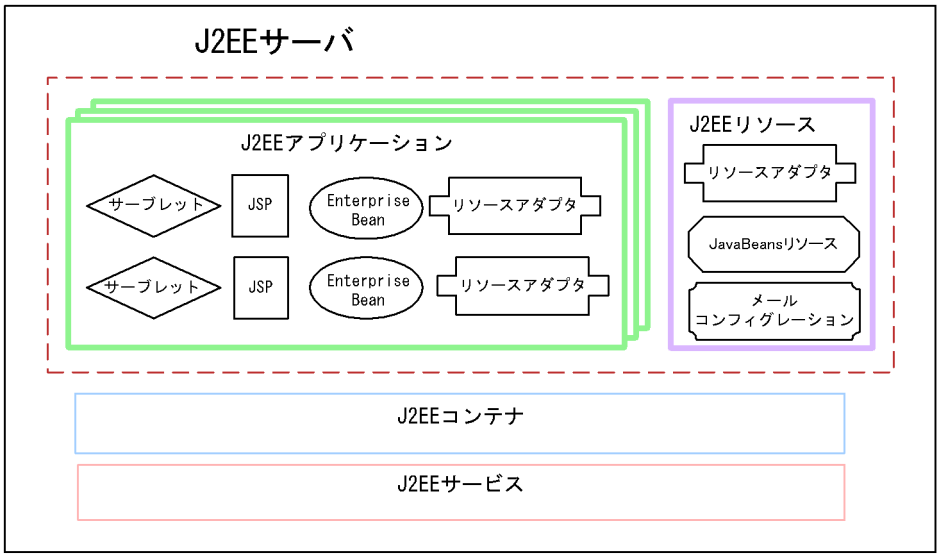
1.1　アプリケーション設定操作の目的

アプリケーション設定操作とは，アプリケーションやリソースを J2EE サーバ上で動作させるために必要な設定およびカスタマイズ操作のことです。次のような場合に実行します。

- Cosminexus システムの動作環境の構築，運用で，リソースや J2EE アプリケーションを設定する場合
- J2EE アプリケーションの開発で，リソースや J2EE アプリケーションを設定する場合

J2EE サーバの構成およびアプリケーション設定操作で管理するアーキテクチャの範囲を次の図に示します。

図 1-1　アプリケーション設定操作で管理するアーキテクチャの範囲



（凡例）

・・・アプリケーション設定操作で管理できる範囲

次のインタフェースを使用して，アプリケーション設定操作を実行できます。

- サーバ管理コマンド（CUI）
- Server Plug-in（GUI）

サーバ管理コマンドについては，「2.2　サーバ管理コマンドの機能」および「第 2 編　サーバ管理コマンドによる操作」を参照してください。Server Plug-in については，「2.3　Server Plug-in の機能」および「第 3 編　Server Plug-in による操作」を参照してください。

なお，インタフェースによって提供している機能に差異があります。機能の差異については，「13.1　J2EE リソース設定機能」および「16.1　J2EE アプリケーション設定の機能一覧」を参照してください。

1.2 J2EE リソースの管理

この節では，アプリケーション設定操作での J2EE リソースの管理の概要について説明します。

1.2.1 管理する J2EE リソースの種類

アプリケーション設定操作で管理する J2EE リソースを次に示します。

(1) リソースアダプタ

リソースアダプタは，次のどちらかの方法で管理します。

- **J2EE リソースアダプタとしてデプロイして使用する。**
  J2EE サーバにインポートしたリソースアダプタを，共有スタンドアロンモジュールとしてデプロイします。その J2EE サーバ上で動作するすべての J2EE アプリケーションで使用できるようになります。J2EE サーバ上にデプロイされたリソースアダプタを J2EE リソースアダプタといいます。
  J2EE リソースアダプタのアプリケーション設定操作については，「1.2.2 リソースアダプタのアプリケーション設定操作」を参照してください。
- **J2EE アプリケーションに含めて使用する。**
  J2EE アプリケーションに含まれるリソースアダプタを，その J2EE アプリケーションで使用できるようにします。
  J2EE アプリケーションに含まれるリソースアダプタのアプリケーション設定操作については，「1.2.3 J2EE アプリケーションに含まれるリソースアダプタのアプリケーション設定操作」を参照してください。

! 注意事項

J2EE アプリケーションは，これらのどちらのリソースアダプタも使用できます。ただし，同じ表示名のリソースアダプタを同時に使用することはできません。同じ表示名に設定するとエラーとなります。

J2EE リソースアダプタとして使用するリソースアダプタと J2EE アプリケーションに含めて使用するリソースアダプタの比較を次に示します。

表 1-1 管理するリソースアダプタの種類

リソースアダプタ			J2EE リソースアダプタとして使用	J2EE アプリケーションに含めて使用
DB Connector	トランザクションサポートレベル	NoTransaction または LocalTransaction	○	○
		XATransaction	○	×
	クラスタ構成		○	×
uCosminexus TP1 Connector	トランザクションサポートレベル	NoTransaction または LocalTransaction	○	○
		XATransaction	○	×
TP1/Message Queue - Access			○	×
DB Connector for Cosminexus RM および Cosminexus RM			○	×
そのほかのリソースアダプタ			○	○※

（凡例） ○：使用できる ×：使用できない

注※ 次のリソースアダプタは，J2EE アプリケーションに含めて管理することはできません。

- トランザクションサポートレベルにグローバルトランザクション管理（XATransaction）が設定されたリソースアダプタ
- ネイティブライブラリを含むリソースアダプタ
- 起動順序の制御が必要なリソースアダプタ

(2) リソースアダプタ以外の J2EE リソース

リソースアダプタ以外に，次の J2EE リソースを管理します。

- JavaBeans リソース
- メールコンフィグレーション

リソースアダプタ以外の J2EE リソースのアプリケーション設定操作については，「1.2.4 リソースアダプタ以外の J2EE リソースのアプリケーション設定操作」を参照してください。

1.2.2 リソースアダプタのアプリケーション設定操作

J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの，アプリケーション設定操作について説明します。

(1) リソースアダプタの種類

J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの種類は次のとおりです。

- DB Connector
  JDBC インタフェースを使用してデータベースに接続する場合に使用します。
- DB Connector for Cosminexus RM および Cosminexus RM
  JMS インタフェースを使用してデータベースに接続する場合に使用します。
  DB Connector for Cosminexus RM については，マニュアル「Cosminexus Reliable Messaging」の「2.7　DB Connector for Cosminexus RM の機能」を参照してください。Cosminexus RM の管理については，マニュアル「Cosminexus Reliable Messaging」の「2.3　メッセージの管理」を参照してください。
- uCosminexus TP1 Connector
  OpenTP1 の SPP に接続する場合に使用します。
- TP1/Message Queue - Access
  TP1/Message Queue に接続する場合に使用します。
- そのほかのリソースアダプタ
  任意のリソースに接続するためのリソースアダプタとして，Connector 1.5 に準拠したリソースアダプタを使用できます。

(2) リソースアダプタの管理の流れ

リソースアダプタを管理する基本的な流れを次に示します。

1. **リソースアダプタのインポート**
2. **リソースアダプタのプロパティ定義**
3. **リソースアダプタのデプロイ**
4. J2EE **リソースアダプタの接続テスト**
5. J2EE **リソースアダプタの開始・停止**
6. J2EE **リソースアダプタのエクスポート**

各手順について説明します。

● リソースアダプタのインポート

リソースアダプタをインポートします。

● リソースアダプタのプロパティ定義

データベースと接続するための情報を定義します。

● リソースアダプタのデプロイ

インポートしたリソースアダプタを J2EE リソースアダプタとしてデプロイします。
J2EE リソースアダプタとしてデプロイされたリソースアダプタは，J2EE サーバ上で動作するすべての J2EE アプリケーションから使用できます。

● J2EE リソースアダプタの接続テスト

接続テストを実施して，J2EE リソースアダプタが正しく動くことを確認します。

● J2EE リソースアダプタの開始・停止

設定が完了した J2EE リソースアダプタを開始します。また，運用に応じて J2EE リソースアダプタを停止します。

● J2EE リソースアダプタのエクスポート

J2EE リソースアダプタを，運用に応じてエクスポートします。

データベースと接続する場合，データベースの設定が必要になります。

また，リソースアダプタを使用する場合，J2EE アプリケーションでリソースアダプタのリファレンスを解決する必要があります。リソースアダプタのリファレンス解決は，J2EE アプリケーションのカスタマイズで実行します。J2EE アプリケーションのカスタマイズの概要については，「1.3　J2EE アプリケーションの管理」を参照してください。

(3) リソースアダプタのアプリケーション設定操作

リソースアダプタの管理で使用する，アプリケーション設定操作の機能を次に示します。

表 1-2　リソースアダプタの管理に使用するアプリケーション設定操作

J2EE リソース管理の操作	機能	コマンド
リソースアダプタのインポート	リソースアダプタを J2EE サーバにインポートします。	cjimportres（-type rar 指定）
リソースアダプタのプロパティ定義	デプロイ前のリソースアダプタの属性を取得して，属性ファイルを生成します。	cjgetresprop（-type rar 指定）
	デプロイ前のリソースアダプタの属性を，属性ファイルに指定された値に変更します。	cjsetresprop（-type rar 指定）
	デプロイされている J2EE リソースアダプタの属性を取得して，属性ファイルを生成します。	cjgetrarprop
	デプロイされている J2EE リソースアダプタの属性を，属性ファイルに指定された値に変更します。	cjsetrarprop
リソースアダプタのデプロイ	リソースアダプタをデプロイします。	cjdeployrar
J2EE リソースアダプタの接続テスト	J2EE リソースアダプタの接続テストをします。	cjtestres（-type rar 指定）
J2EE リソースアダプタの開始と停止	J2EE リソースアダプタを開始します。	cjstartrar

J2EE リソース管理の操作	機能	コマンド
	開始されている J2EE リソースアダプタを停止します。	cjstoprar
J2EE リソースアダプタのエクスポート	J2EE サーバ上のリソースアダプタをエクスポートします。	cjexportrar
リソースの一覧表示	インポート済みのリソースアダプタの一覧を表示します。	cjlistres（-type rar 指定）
	J2EE リソースアダプタの一覧を表示します。	cjlistrar
リソースの削除	リソースアダプタを削除します。	cjdeleteres（-type rar 指定）
	J2EE リソースアダプタをアンデプロイします。	cjundeployrar
そのほかの操作	リソースアダプタのコネクションプールの状態を表示します。	cjlistpool ※1
	リソースアダプタのコネクションプールのコネクションを削除します。	cjclearpool ※1
	クラスタコネクションプールのメンバコネクションプールを一時停止します。	cjsuspendpool ※2
	一時停止したクラスタコネクションプールのメンバコネクションプールを再開します。	cjresumepool ※2
	リソースアダプタの属性をコピーします。	cjcopyres（-type rar 指定）

注※1　DB Connector を使用してコネクションプールをクラスタ化した場合，ルートリソースアダプタには使用できません。メンバリソースアダプタに使用できます。
注※2　DB Connector を使用してコネクションプールをクラスタ化した場合のメンバリソースアダプタだけに使用できます。

GUI インタフェースの Server Plug-in は，サーバ管理コマンドに対応しています。サーバ管理コマンドとの対応については，「第 3 編　Server Plug-in による操作」の各章の「対応するサーバ管理コマンド」を参照してください。

1.2.3　J2EE アプリケーションに含まれるリソースアダプタのアプリケーション設定操作

J2EE アプリケーションに含まれるリソースアダプタのアプリケーション設定操作について説明します。

(1) リソースアダプタの種類

アプリケーション設定操作で管理する，J2EE アプリケーションに含めて使用するリソースアダプタの種類は次のとおりです。

- DB Connector
  JDBC インタフェースを使用してデータベースに接続する場合に使用します。
  グローバルトランザクションおよびクラスタ構成の場合は利用できません。
- uCosminexus TP1 Connector
  OpenTP1 の SPP に接続する場合に使用します。
  グローバルトランザクションの場合は利用できません。
- そのほかのリソースアダプタ（Connector 1.5 に準拠したリソースアダプタ）
  任意のリソースに接続するためのリソースアダプタとして，Connector 1.5 に準拠したリソースアダプタを使用できます。
  J2EE アプリケーションに含めて利用できるリソースアダプタの種類については，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「3.3.2　リソースアダプタの種類」を参照してください。

(2) リソースアダプタの管理の流れ

J2EE アプリケーションに含まれるリソースアダプタを管理する基本的な流れを次に示します。

1. J2EE **アプリケーションへのリソースアダプタの追加**
2. J2EE **アプリケーションに含まれるリソースアダプタのプロパティ定義**
3. J2EE **アプリケーションに含まれるリソースアダプタの接続テスト**
4. J2EE **アプリケーションの開始・停止**
5. J2EE **アプリケーションのエクスポート**

各手順について説明します。

● J2EE アプリケーションへのリソースアダプタの追加

次のどちらかの方法で，J2EE アプリケーションにリソースアダプタを追加します。

- アプリケーション開発環境で，リソースアダプタを追加した J2EE アプリケーションを作成します。その J2EE アプリケーションを J2EE サーバにインポートします。
- リソースアダプタを J2EE サーバにインポートして，J2EE アプリケーションに追加します。

● J2EE アプリケーションに含まれるリソースアダプタのプロパティ定義

データベースと接続するための情報を定義します。

● J2EE アプリケーションに含まれるリソースアダプタの接続テスト

接続テストを実施して，J2EE アプリケーションに含まれるリソースアダプタが正しく動くことを確認します。

● J2EE アプリケーションの開始・停止

設定が完了したリソースアダプタを含む J2EE アプリケーションを開始します。J2EE アプリケーションの開始で，J2EE アプリケーションに含まれるリソースアダプタが開始されます。

また，運用に応じて J2EE アプリケーションを停止します。J2EE アプリケーションの停止で，J2EE アプリケーションに含まれるリソースアダプタが停止されます。

● J2EE アプリケーションのエクスポート

運用に応じて，リソースアダプタを J2EE アプリケーションに含めてエクスポートします。

データベースと接続する場合，データベースの設定が必要になります。

また，リソースアダプタを使用する場合，J2EE アプリケーションでリソースアダプタのリファレンスを解決する必要があります。リソースアダプタのリファレンス解決は，J2EE アプリケーションのカスタマイズで実行します。J2EE アプリケーションのカスタマイズの概要については，「1.3　J2EE アプリケーションの管理」を参照してください。

(3) リソースアダプタのアプリケーション設定操作

J2EE アプリケーションに含まれるリソースアダプタの管理で使用する，アプリケーション設定操作の機能を次に示します。

表 1-3　J2EE アプリケーションに含まれるリソースアダプタの管理に使用するアプリケーション設定操作

J2EE リソース管理の操作	機能	コマンド
リソースアダプタのインポート	リソースアダプタを J2EE サーバにインポートします。	cjimportres（-type rar 指定）
J2EE アプリケーションへのリソースアダプタの追加	J2EE アプリケーションに，リソースアダプタを追加します。	cjaddapp（-type rar 指定）
J2EE アプリケーションのインポート	リソースアダプタを含む J2EE アプリケーションをインポートします。	cjimportapp
リソースアダプタのプロパティ定義	J2EE アプリケーションに含まれるリソースアダプタの属性を取得して，属性ファイルを生成します。	cjgetappprop（-type rar 指定）
	J2EE アプリケーションに含まれるリソースアダプタの属性を，属性ファイルに指定された値に変更します。	cjsetappprop（-type rar 指定）

J2EE リソース管理の操作	機能	コマンド
リソースアダプタの接続テスト	J2EE アプリケーションに含まれるリソースアダプタの接続テストをします。	cjtestres（-name ［アプリケーション名］-type rar 指定）
J2EE アプリケーションの開始と停止	J2EE アプリケーションを開始します。J2EE アプリケーションの開始で，J2EE アプリケーションに含まれるすべてのリソースアダプタが開始されます。リソースアダプタ単位の開始はできません。	cjstartapp
	開始されている J2EE アプリケーションを停止します。J2EE アプリケーションの停止で，J2EE アプリケーションに含まれるすべてのリソースアダプタが停止されます。リソースアダプタ単位の停止はできません。	cjstopapp
J2EE アプリケーションのエクスポート	J2EE アプリケーションに含めて，リソースアダプタをエクスポートします。	cjexportapp
J2EE アプリケーションからリソースアダプタの削除	J2EE アプリケーションから，J2EE リソースのリソースアダプタを削除します。	cjdeleteapp（-type rar 指定）
リソースの一覧表示	J2EE アプリケーションに含まれるリソースアダプタの一覧を表示します。	cjlistapp（-type rar 指定）
そのほかの操作	J2EE アプリケーションに含まれるリソースアダプタのコネクションプールの状態を表示します。	cjlistpool（-name ［アプリケーション名］指定）
	J2EE アプリケーションに含まれるリソースアダプタのコネクションプールのコネクションを削除します。	cjclearpool（-name ［アプリケーション名］指定）

GUI インタフェースの Server Plug-in は，サーバ管理コマンドに対応しています。サーバ管理コマンドとの対応については，「第 3 編　Server Plug-in による操作」の各章の「対応するサーバ管理コマンド」を参照してください。

1.2.4　リソースアダプタ以外の J2EE リソースのアプリケーション設定操作

リソースアダプタ以外の J2EE リソースの，アプリケーション設定操作について説明します。

(1) リソースアダプタ以外の J2EE リソースの種類

アプリケーション設定操作で管理する，リソースアダプタ以外の J2EE リソースは次のとおりです。

- JavaBeans リソース
- メールコンフィグレーション

(2) J2EE リソース管理の流れ

J2EE リソースの管理の基本的な流れを次に示します。

1. J2EE リソースのインポート
2. J2EE リソースのプロパティ定義
3. J2EE リソースの接続テスト
4. J2EE リソースの開始・停止

サーバ管理コマンドを使用した J2EE リソースの管理では，J2EE リソースの種別によって，必要な設定および操作方法が異なります。

J2EE リソースの種別ごとに，それぞれの手順で必要な作業を次の表に示します。手順の列に記載されている番号に従った順序で作業を実施してください。

表 1-4　J2EE リソースの管理で行う必要な作業

手順		JavaBeans リソース	メールコンフィグレーション
1.	J2EE リソースのインポート	○	―
2.	J2EE リソースのプロパティ定義	○	○※
3.	J2EE リソースの接続テスト	―	○
4.	J2EE リソースの開始・停止	○	―

（凡例）　○：必要　―：該当項目なし
注※　メールコンフィグレーションのプロパティ定義には，メールコンフィグレーションの新規作成が含まれます。

各手順について説明します。

● J2EE リソースのインポート

JavaBeans リソースをインポートします。

● J2EE リソースのプロパティ定義

JavaBeans リソースやメールコンフィグレーションを管理する情報を定義します。

● J2EE リソースの接続テスト

メールコンフィグレーションが正しく動くことを確認します。

● J2EE リソースの開始・停止

設定が完了した JavaBeans リソースを開始します。また，運用に応じて JavaBeans リソースを停止します。

(3) J2EE リソースのアプリケーション設定操作

J2EE リソース管理で使用する，アプリケーション設定操作の機能を次に示します。

表 1-5 J2EE リソースの管理に使用するアプリケーション設定操作

J2EE リソース管理の操作	機能	コマンド
J2EE リソースのインポート	JavaBeans リソースを J2EE サーバにインポートします。	cjimportjb
J2EE リソースのプロパティ定義	JavaBeans リソースの属性を取得して，属性ファイルを生成します。	cjgetjbprop
	JavaBeans リソースの属性を，属性ファイルに指定された値に変更します。	cjsetjbprop
	メールに含まれるリソースの属性を取得して，属性ファイルを生成します。	cjgetresprop（-type mail 指定）
	メールに含まれるリソースの属性を，属性ファイルに指定された値に変更します。	cjsetresprop（-type mail 指定）
J2EE リソースの接続テスト	メールの接続テストをします。	cjtestres（-type mail 指定）
J2EE リソースの開始と停止	JavaBeans リソースを開始します。	cjstartjb
	JavaBeans リソースを停止します。	cjstopjb
リソースの一覧表示	インポート済みの JavaBeans リソースの一覧を表示します。	cjlistjb
	インポート済みのメール一覧を表示します。	cjlistres（-type mail 指定）
リソースの削除	JavaBeans リソースを削除します。	cjdeletejb
	メールを削除します。	cjdeleteres（-type mail 指定）
そのほかの操作	メールの属性をコピーします。	cjcopyres（-type mail 指定）

1.3 J2EE アプリケーションの管理

この節では，アプリケーション設定操作での J2EE アプリケーションの管理の概要について説明します。

1.3.1 管理する J2EE アプリケーション

管理する J2EE アプリケーションの形式と，J2EE アプリケーションを構成するコンポーネントについて説明します。

(1) J2EE アプリケーションの形式

アプリケーション設定操作で管理するのは，次のどちらかの形式のアプリケーションです。

- **アーカイブ形式の J2EE アプリケーション**
  J2EE サーバの作業ディレクトリ下に，J2EE アプリケーションのコンポーネントの構成ファイルを持つ形式です。
- **展開ディレクトリ形式の J2EE アプリケーション**
  J2EE サーバの作業ディレクトリ下に，J2EE アプリケーションのコンポーネントの構成ファイルを持たないで，J2EE サーバの外部にある構成ファイルを論理的な J2EE アプリケーションとして使用する形式です。

J2EE アプリケーションの形式の詳細については，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「12. J2EE アプリケーションの形式とデプロイ」を参照してください。

(2) J2EE アプリケーションを構成するコンポーネント

アプリケーション設定操作では，次のコンポーネントで構成される J2EE アプリケーションを管理します。

- **サーブレット・JSP で構成された J2EE アプリケーション**
- **Enterprise Bean で構成された J2EE アプリケーション**
- **サーブレット・JSP と Enterprise Bean で構成された J2EE アプリケーション**

また，J2EE アプリケーションのコンポーネントにリソースアダプタを含めて，管理することもできます。

アーカイブ形式の J2EE アプリケーションの場合は，コンポーネントは，J2EE アプリケーションへの追加や新規 J2EE アプリケーションの作成で使うことができます。J2EE アプリケーションの作成については，「7. J2EE アプリケーションの作成」を参照してください。

1.3.2　J2EE アプリケーション管理の流れ

J2EE アプリケーションの管理の流れを次に示します。

1. **ファイルのインポート**
2. J2EE **アプリケーションの作成**※
3. J2EE **アプリケーションのプロパティ定義**
4. J2EE **アプリケーションの開始・停止**
5. J2EE **アプリケーションのエクスポート**

注※　すでにアプリケーション開発ツールなどを使用して作成，設定されている J2EE アプリケーションの場合は実施しません。

J2EE アプリケーションの管理で必要な作業を次の表に示します。手順の列に記載されている番号に従った順序で作業を実施してください。

表 1-6　J2EE **アプリケーションの管理で行う必要な作業**

手順		新規に作成する J2EE アプリケーション	作成済みの J2EE アプリケーション ※1	
			アーカイブ形式	展開ディレクトリ形式
1.	ファイルのインポート	○※2	○	○
2.	J2EE アプリケーションの作成	○	＿※3	―
3.	J2EE アプリケーションのプロパティ定義	○	○	○※4
4.	J2EE アプリケーションの開始・停止	○	○	○
5.	J2EE アプリケーションのエクスポート	○	○	○

（凡例）　○：必要　―：該当項目なし

注※ 1　アプリケーション開発ツールなどを使用して作成した J2EE アプリケーションです。J2EE アプリケーションの形式によって，インポート方法が異なります。

- アーカイブ形式の J2EE アプリケーションを設定する場合，アーカイブファイルのパスを指定して J2EE アプリケーションをインポートしてください。
- 展開ディレクトリ形式の J2EE アプリケーションを設定する場合，アプリケーションディレクトリパスまたは展開ディレクトリパスを指定して，J2EE アプリケーションをインポートしてください。

注※ 2　コンポーネントの構成ごとに，次のファイルをインポートします。

- サーブレット・JSP が含まれる J2EE アプリケーションを作成する場合，サーブレット・JSP をインポートしてください。

- Enterprise Bean が含まれる J2EE アプリケーションを作成する場合，Enterprise Bean をインポートしてください。
- リソースアダプタが含まれる J2EE アプリケーションを作成する場合，リソースアダプタをインポートしてください。

注※ 3　作成済みのアーカイブ形式の J2EE アプリケーションに，コンポーネントやライブラリ JAR を追加・削除できます。
注※ 4　サーバ管理コマンドを実行するホストと異なるホストで稼働している J2EE サーバの展開ディレクトリ形式の J2EE アプリケーションについては，J2EE アプリケーションの設定およびカスタマイズはできません。

各手順について説明します。

● ファイルのインポート

J2EE アプリケーションを構成するコンポーネント（サーブレット・JSP，Enterprise Bean，リソースアダプタおよびライブラリ JAR）や，作成済みの J2EE アプリケーションをインポートします。

● J2EE アプリケーションの作成

インポートしたファイルで，J2EE アプリケーションを作成します。

● J2EE アプリケーションのプロパティ定義

作成した J2EE アプリケーションまたはインポートした作成済みの J2EE アプリケーションに対して，J2EE アプリケーションを構成するコンポーネントに応じた属性を設定します。

- DD 文に対応する属性の定義
- 実行時プロパティや動作の設定
- J2EE リソースのリファレンス解決

● J2EE アプリケーションの開始・停止

設定やカスタマイズの完了した J2EE アプリケーションを開始します。また，運用に応じて停止します。

● J2EE アプリケーションのエクスポート

運用に応じて，J2EE アプリケーションをエクスポートします。

1.3.3 J2EE アプリケーション管理に使用するアプリケーション設定操作

J2EE アプリケーション管理で使用する，アプリケーション設定操作の機能を次に示します。

表 1-7 J2EE アプリケーションの管理に使用するサーバ管理コマンド

J2EE アプリケーション管理の操作	機能	コマンド
ファイルのインポート	J2EE アプリケーションを構成するコンポーネント（WAR ファイル，EJB-JAR ファイル，RAR ファイル）を，J2EE サーバにインポートします。	cjimportres
	ライブラリ JAR ファイルを，J2EE サーバにインポートします。	cjimportlibjar
	J2EE アプリケーションを，J2EE サーバにインポートします。	cjimportapp
J2EE アプリケーションの作成	インポート済みの EJB-JAR ファイル，WAR ファイル，RAR ファイルを J2EE アプリケーションに追加します。	cjaddapp
J2EE アプリケーションのプロパティ定義	J2EE アプリケーションまたは J2EE アプリケーションに含まれるコンポーネントの属性を取得して，属性ファイルを生成します。	cjgetappprop
	J2EE アプリケーションまたは J2EE アプリケーションに含まれるコンポーネントの属性を，指定されたアプリケーション属性ファイルの値に変更します。	cjsetappprop
J2EE アプリケーションの開始と停止	J2EE アプリケーションを開始して，クライアントからのリクエストを受け取ることができるようにします。	cjstartapp
	J2EE アプリケーションを停止して，クライアントからのリクエストを受け取らないようにします。	cjstopapp
J2EE アプリケーションのエクスポート	J2EE サーバ上の J2EE アプリケーションをエクスポートします。	cjexportapp
J2EE アプリケーションの一覧表示	すべての J2EE アプリケーションについての名称と状態，J2EE アプリケーションに含まれる EJB-JAR ファイル，WAR ファイル，または RAR ファイルの一覧を表示します。	cjlistapp
	J2EE アプリケーションに含まれるライブラリ JAR の一覧を表示します。	cjlistlibjar
	J2EE サーバで稼働しているトランザクション情報の一覧を表示します。	cjlisttrn
	J2EE サーバで停止中のトランザクション情報の一覧を表示します。	cjlisttrnfile
J2EE アプリケーションの削除	J2EE アプリケーションまたは J2EE アプリケーションに含まれる EJB-JAR ファイル，WAR ファイル，RAR ファイルを，指定された J2EE サーバから削除します。	cjdeleteapp

J2EE アプリケーション管理の操作	機能	コマンド
	ライブラリ JAR を J2EE アプリケーションから削除します。	cjdeletelibjar
そのほかの操作	J2EE アプリケーションの動作モードを変更します。	cjchmodapp
	展開ディレクトリ形式の J2EE アプリケーションを入れ替えます。※	cjreloadapp
	アーカイブ形式の J2EE アプリケーションを入れ替えます。※	cjreplaceapp
	J2EE アプリケーションの名称を変更します。	cjrenameapp
	起動されているリソースアダプタのコネクションプールの状態を表示します。	cjlistpool
	CMP2.x Entity Bean 用の SQL 文を生成します。	cjgencmpsql
	ロールにユーザを追加します。	cjmapsec
	ユーザまたはロールを追加します。	cjaddsec
	ユーザまたはロールを削除します。	cjdeletesec
	ユーザまたはロールの一覧を表示します。	cjlistsec
	J2EE アプリケーションについて，RMI-IIOP スタブおよびインタフェースを取得します。	cjgetstubsjar

注※　J2EE アプリケーションの形式については，「1.3.1　管理する J2EE アプリケーション」を参照してください。

GUI インタフェースの Server Plug-in は，サーバ管理コマンドに対応しています。サーバ管理コマンドとの対応については，「第 3 編　Server Plug-in による操作」の各章の「対応するサーバ管理コマンド」を参照してください。

1.4　アプリケーション設定操作の制約

この節では，J2EE アプリケーション設定操作を実行する場合の，実行時ホストおよび J2EE アプリケーションの状態による制約について説明します。

1.4.1　アプリケーション設定操作の実行ホストについての制約

アプリケーション設定操作は，J2EE サーバおよび CORBA ネーミングサービスとは別のホストから実行できます。

ただし，展開ディレクトリ形式の J2EE アプリケーションの場合，J2EE アプリケーションの設定とカスタマイズについては J2EE サーバと同じホストだけで実行できます。

別のホストの J2EE サーバを操作する場合，あらかじめ次の設定が必要です。

- **J2EE サーバへのアクセス権の設定**
  操作対象の J2EE サーバへのアクセス権限が必要です。J2EE サーバへのアクセス権を設定するには，J2EE サーバ用の usrconf.properties ファイルで，webserver.connector.http.permitted.hosts キーを定義します。J2EE サーバ用の usrconf.properties ファイルの詳細については，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。
  J2EE サーバに対するアクセス権限を持っていないときには，エラーメッセージが出力され，アプリケーション設定操作は実行できません。
- **ネットワークの設定**
  サーバ管理コマンド実行ホストから J2EE サーバ実行ホストのホスト名称を解決できるよう，あらかじめネットワークを設定しておく必要があります。

アプリケーション設定操作，J2EE サーバおよび CORBA ネーミングサービスを別のホストで実行する場合，それぞれにインストールされている Cosminexus のバージョンが異なると正常に実行できないときがあります。それぞれに同じバージョンの Cosminexus がインストールされている環境で使用してください。

1.4.2　J2EE アプリケーションの状態による操作の制約

J2EE アプリケーションの操作には，J2EE アプリケーションの状態によって実行できるものと実行できないものがあります。

J2EE アプリケーションの状態を次の表に示します。

表 1-8 J2EE アプリケーションの状態

項番	状態※	説明
1	開始	J2EE アプリケーションが開始されています。
2	閉塞中	閉塞処理が実施されています。
3	閉塞	閉塞処理が正常終了しました。
4	通常停止中	通常停止処理が実施されています。
5	強制停止中	強制停止操作で強制停止処理が実施されています。
6	停止	J2EE アプリケーションが停止されています。
7	閉塞失敗	閉塞処理が異常終了しました。
8	通常停止失敗	通常停止処理が異常終了しました。
9	強制停止失敗	強制停止処理が異常終了しました。

注※ J2EE アプリケーションの状態を参照する方法については「10.3 J2EE アプリケーションの一覧の参照」を参照してください。

J2EE アプリケーションがそれぞれの状態の場合に実行できる操作，および実行できない操作を次の表に示します。

表 1-9 J2EE アプリケーションの状態と実行できる操作

操作（コマンド）		J2EE アプリケーションの状態								
		1	2	3	4	5	6	7	8	9
ファイルのインポート	WAR ファイル，EJB-JAR ファイルおよび RAR ファイルのインポート（cjimportres）	○	○	○	○	○	○	○	○	○
	ライブラリ JAR ファイルのインポート（cjimportlibjar）	－	－	－	－	－	○	－	－	－
J2EE アプリケーションの作成（cjaddapp）		－	－	－	－	－	○	－	－	－
J2EE アプリケーションの設定とカスタマイズ	属性ファイルの取得（cjgetappprop）※1	○	－	－	－	－	○	－	－	－
	属性の設定（cjsetappprop）※1	－	－	－	－	－	○	－	－	－

操作（コマンド）		J2EE アプリケーションの状態								
		1	2	3	4	5	6	7	8	9
J2EE アプリケーションの開始と停止	J2EE アプリケーションの開始（cjstartapp）	−	−	−	−	−	○	−	−	−
	J2EE アプリケーションの通常停止（cjstopapp）	○	−	−	−	−	−	−	−	−
	J2EE アプリケーションの手動強制停止（cjstopapp）	−	−	○	○	−	−	−	−	−
	J2EE アプリケーションの自動強制停止（cjstopapp）	○	−	−	−	−	−	−	−	−
J2EE アプリケーションのエクスポート（cjexportapp）		○	−	−	−	−	○	−	−	−
J2EE アプリケーションの一覧表示	J2EE アプリケーションの状態表示（cjlistapp）	○	○	○	○	○	○	○	○	○
	ライブラリ JAR の一覧表示（cjlistlibjar）	○	○	○	○	○	○	○	○	○
	稼働中のトランザクション情報一覧表示（cjlisttrn）	○	○	○	○	○	○	○	○	○
	停止中のトランザクション情報一覧表示（cjlisttrnfile）	○	○	○	○	○	○	○	○	○
J2EE アプリケーションの削除	J2EE アプリケーションおよび構成コンポーネントの削除（cjdeleteapp）	−	−	−	−	−	○	−	−	−
	ライブラリ JAR の削除（cjdeletelibjar）	−	−	−	−	−	○	−	−	−
そのほかの操作	J2EE アプリケーションの動作モード変更（cjchmodapp）	−	−	−	−	−	○	−	−	−
	展開ディレクトリ形式の J2EE アプリケーションの入れ替え（cjreloadapp）	○	−	−	−	−	○	−	−	−

操作（コマンド）		J2EE アプリケーションの状態								
		1	2	3	4	5	6	7	8	9
	アーカイブ形式の J2EE アプリケーションの入れ替え（cjreplaceapp）	○	－	－	－	－	○	－	－	－
	J2EE アプリケーション名称の変更（cjrenameapp）	－	－	－	－	－	○	－	－	－
	リソースアダプタのコネクションプールの状態表示（cjlistpool）	○	－	－	－	－	－	－	－	－
	SQL 文の生成（cjgencmpsql）	－	－	－	－	－	○	－	－	－
	ロールにユーザを登録（cjmapsec）	○	○	○	○	○	○	○	○	○
	ユーザまたはロールの追加（cjaddsec）	○	○	○	○	○	○	○	○	○
	ユーザまたはロールの削除（cjdeletesec）	○	○	○	○	○	○	○	○	○
	ユーザまたはロールの一覧表示（cjlistsec）	○	○	○	○	○	○	○	○	○
	RMI-IIOP スタブおよびインタフェースの取得（cjgetstubsjar）	○	○	○	○	○	○※2	○	○	○

（凡例）

数字：表 1-8 の項番に対応した，J2EE アプリケーションの状態

○：実行できる

－：実行できない

注※1　J2EE アプリケーションのカスタマイズおよび各コンポーネントのプロパティを変更する場合，J2EE アプリケーションが開始されていても属性ファイルは取得できます。変更内容に編集した属性ファイルの値を反映するためには，J2EE アプリケーションを停止させてください。

注※2　J2EE アプリケーションが一度も開始されていない場合は，実行できません。

1.4.3　そのほかの制約

サーバ管理コマンドによる処理が実行されているときに別のサーバ管理コマンドを実行した場合，処理の実行はサーバ単位および処理を実行しているサーバ管理コマンドの系統ごとに制御されます。サーバ管理コマンドを同時に実行した場合の制御については，「3.2　サーバ管理コマンドの排他制御」を参照してください。

2 アプリケーション設定操作で使用するインタフェース

この章では，アプリケーション設定操作のインタフェースである，サーバ管理コマンドと Server Plug-in の概要について説明します。

2.1　インタフェースの種類

アプリケーション設定操作で使用するインタフェースの種類および各インタフェースの使い方と機能の相違について説明します。

アプリケーション設定操作では，インタフェースとしてサーバ管理コマンド（CUI）とServer Plug-in（GUI）を提供しています。

2.1.1　インタフェースの特長

それぞれのインタフェースの特長について説明します。

● サーバ管理コマンドを利用する場合

サーバ管理コマンドは，J2EE サーバで動作するリソースや J2EE アプリケーションを設定，および管理するためのコマンド群です。GUI での対話処理が不要な運用環境の操作などに適しています。

サーバ管理コマンドの機能については，「2.2　サーバ管理コマンドの機能」を参照してください。

● Server Plug-in を利用する場合

Server Plug-in は論理サーバ，J2EE リソース，および J2EE アプリケーションの操作をGUI で対話的に実行します。J2EE アプリケーションの開発を支援する MyEclipse と連携して，J2EE アプリケーションの開発を円滑に行えます。

アプリケーション設定操作の機能のうち，提供していないものもあります。Server Plug-in の機能については，「2.3　Server Plug-in の機能」を参照してください。

MyEclipse 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」を参照してください。

2.1.2　機能の比較

サーバ管理コマンドと Server Plug-in で異なる機能を次に示します。

（1）リソースの設定

Server Plug-in では，次の機能は提供していません。

- JavaBeans リソースの設定
- メールコンフィグレーションの設定
- J2EE リソースのコピー

リソース設定についてのサーバ管理コマンドと Server Plug-in の比較詳細は，「13.1 J2EE リソース設定機能」を参照してください。

(2) J2EE アプリケーションの設定

サーバ管理コマンドに比べて，Server Plug-in では，次の機能に差異があります。

- **J2EE アプリケーション（EAR）の作成**
  Server Plug-in では，作成済みの J2EE アプリケーションだけがインポートできます。J2EE アプリケーションの構成コンポーネント（EJB-JAR，WAR，RAR）をインポートして，新規に J2EE アプリケーションを作成することはできません。
- **展開ディレクトリ形式の J2EE アプリケーションのインポート**
  Server Plug-in では，アプリケーションディレクトリパスを指定して，展開ディレクトリ形式のアプリケーションをインポートできます。
  展開ディレクトリパスを指定して，アーカイブ形式の J2EE アプリケーションを展開ディレクトリに変換してインポートすることはできません。
  展開ディレクトリ形式の J2EE アプリケーションについては，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「12.4　展開ディレクトリ形式の J2EE アプリケーション」を参照してください。
- **プロパティの定義**
  Server Plug-in では，次のプロパティは定義できません。
  - セキュリティアイデンティティの設定
  - Entity Bean の CMP 定義
- **J2EE アプリケーションの操作**
  Server Plug-in では，J2EE アプリケーションに次の操作はできません。
  - JSP をコンパイルして J2EE アプリケーションを開始
  - J2EE アプリケーションの自動強制停止
  - トランザクション一覧の参照

J2EE アプリケーションの設定についての，サーバ管理コマンドと Server Plug-in の比較詳細は，「16.1　J2EE アプリケーション設定の機能一覧」を参照してください。

(3) ほかの機能

Server Plug-in では，Management Server リモート管理機能と接続しているホストについて，次の単位で起動，停止ができます。

- 運用管理ドメイン
- 運用管理ドメイン内のホスト
- 各論理サーバ

また，各論理サーバの稼働状態も監視できます。

Server Plug-in を使用しない場合のこれらの起動，停止については，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「7.7.2 Management Server の起動／停止と設定」，およびマニュアル「Cosminexus アプリケーションサーバ 機能解説 運用／監視／連携編」の「2.2　日常運用での起動と停止」

を参照してください。Server Plug-in を使用しない場合のこれらの稼働状態の監視の方法については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 運用／監視／連携編」の「3. 稼働情報の監視（稼働情報収集機能）」を参照してください。

2.2　サーバ管理コマンドの機能

サーバ管理コマンドの機能を次の表に示します。サーバ管理コマンドの操作については，「第 2 編　サーバ管理コマンドによる操作」を参照してください。

表 2-1　サーバ管理コマンドの機能一覧

機能		参照先
リソースアダプタの設定	データベースと接続するための設定	4.2
	データベースと接続するための設定（コネクションプールのクラスタ化の場合）	4.3
	そのほかのリソースアダプタと接続するための設定	4.4
J2EE アプリケーションに含まれるリソースアダプタの設定		5 章
リソースアダプタ以外の J2EE リソースの設定	JavaBeans リソースの設定	6.2
	メールコンフィグレーションの設定	6.3
J2EE アプリケーション（EAR）の作成		7 章
J2EE アプリケーションのインポート※ 1 とエクスポート		8 章
J2EE アプリケーションのプロパティの設定		9 章
J2EE アプリケーションの実行	J2EE アプリケーションの開始と停止	10.2
	J2EE アプリケーションの一覧の参照	10.3
	J2EE アプリケーションの削除	10.4
	テストモードによる J2EE アプリケーションの実行	10.5
	J2EE アプリケーションの入れ替え※ 2	10.6
	J2EE アプリケーション名の変更	10.7
	RMI-IIOP スタブとインタフェースの取得	10.8
	トランザクション一覧の参照	10.9

注※ 1

次の形式の J2EE アプリケーションをインポートできます。

- アーカイブ形式の J2EE アプリケーション（EAR ファイル）
- 展開ディレクトリ形式の J2EE アプリケーション（アプリケーションディレクトリ）

注※ 2

インポート時の J2EE アプリケーションの形式によって，デプロイした J2EE アプリケーションを入れ替える場合，次のように操作が異なります。

- デプロイしたアーカイブ形式の J2EE アプリケーションを入れ替える場合，一度

J2EE アプリケーションを再起動（停止，入れ替え，開始）する必要があります。

- 展開ディレクトリ形式の J2EE アプリケーションを入れ替える場合，開始中の J2EE アプリケーションを再起動（停止，入れ替え，開始）しないで，サーバ管理コマンドを実行して，アプリケーションを入れ替えることができます（リロード機能）。

アプリケーションの入れ替えについては，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「5.6　J2EE アプリケーションの入れ替え」を参照してください。

2.3　Server Plug-in の機能

Server Plug-in の機能一覧を次に示します。Server Plug-in の操作については，「第 3 編　Server Plug-in による操作」を参照してください。

表 2-2　Server Plug-in の機能一覧

機能		参照先
論理サーバ※1 の運用管理	運用管理ドメイン単位の論理サーバの起動	12.2
	運用管理ドメイン単位の論理サーバの停止	12.3
	ホスト単位の論理サーバの起動	12.2
	ホスト単位の論理サーバの停止	12.3
リソースアダプタの設定※2		14 章
J2EE アプリケーションに含まれるリソースアダプタの設定		15 章
J2EE アプリケーションのインポート※3 とエクスポート		17 章
J2EE アプリケーションのプロパティ設定※4		18 章
J2EE アプリケーションの実行	J2EE アプリケーションの開始	19.2
	J2EE アプリケーションの停止	19.3
	J2EE アプリケーションの入れ替え※3	19.5
	J2EE アプリケーション名の変更	19.6
	RMI-IIOP スタブとインタフェースの取得	19.7
	J2EE アプリケーションの削除	19.4
	アプリケーション統合属性のインポートとエクスポート	20 章

注※ 1

運用管理（起動・停止）を実行する運用管理ドメインの種類については，「12.1　論理サーバの運用管理の概要」を参照してください。

注※ 2

Server Plug-in で設定できる J2EE リソースは，リソースアダプタだけです。

注※ 3

Server Plug-in では，MyEclipse での J2EE アプリケーションの作成と連携して，運用環境で，次の形式の J2EE アプリケーションをインポートできます。

- アーカイブ形式の J2EE アプリケーション（EAR ファイル）
- 展開ディレクトリ形式の J2EE アプリケーション（アプリケーションディレクトリ）

注※ 4

J2EE アプリケーションのプロパティの設定項目には，Server Plug-in で設定できない項目もあります。設定できない項目についてはサーバ管理コマンドで設定してください。サーバ管理コマンドとのプロパティ設定項目の相違については，「18.1 J2EE アプリケーションのプロパティ設定の概要」を参照してください。

3 サーバ管理コマンドの基本操作

この章では，サーバ管理コマンドでアプリケーションを設定する場合の基本的な操作について説明します。

3.1　サーバ管理コマンドの実行の前提条件

サーバ管理コマンドを使用する場合，次のことを確認してください。

- サーバ管理コマンドは，J2EE サーバが起動している状態で実行できるコマンドです。使用する場合は，J2EE サーバおよびその前提プロセスを起動してから実行してください。J2EE サーバおよびその前提プロセスの起動については，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「2.3　論理サーバの起動・停止の仕組み」を参照してください。
- サーバ管理コマンドの実行には，root 権限，または Component Container 管理者の権限が必要です。実行に必要な権限を設定してください。
- Windows 7 または Windows Vista で Cosminexus のコマンドを使用する場合 , 管理者特権で実行する必要があります。Windows 7 または Windows Vista で Cosminexus のコマンドを使用する場合の注意事項については，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「1.4　コマンド使用時の注意事項」を参照してください。

サーバ管理コマンドを使用する場合の注意事項については，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「1.4　コマンド使用時の注意事項」を参照してください。

3.2　サーバ管理コマンドの排他制御

サーバ管理コマンドの系統とサーバ管理コマンドの排他制御について説明します。

3.2.1　サーバ管理コマンドの系統

サーバ管理コマンドによる処理が実行されている場合に別のサーバ管理コマンドを実行したとき，処理の実行はサーバ単位および処理を実行しているサーバ管理コマンドの系統ごとに制御されます。

サーバ管理コマンドには次の三つの系統があります。

- **参照系コマンド**
  インポートしたJ2EEアプリケーションの数や状態，含まれるリソース名など，J2EEサーバの構成状態を表示するサーバ管理コマンドです。J2EEサーバの内容は更新しません。
- **更新系コマンド**
  J2EEサーバの内容を更新したり，構成情報を取得したりするサーバ管理コマンドです。
- **特権系コマンド**
  J2EEサーバの内容を更新するコマンドで，常にほかのコマンドよりも優先して処理が実施されるサーバ管理コマンドです。

なお，それぞれのサーバ管理コマンドの系統については，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」を参照してください。

3.2.2　サーバ管理コマンドの排他制御

コマンド実行の排他制御の条件を，サーバ管理コマンドの系統ごとに次の表に示します。

表 3-1　コマンド実行の排他制御の条件

実行中のコマンド	あとから実行するコマンド		
	更新系コマンド	参照系コマンド	特権系コマンド
更新系コマンド	○	◎	◎
参照系コマンド	◎	○	◎
特権系コマンド	◎	◎	×

（凡例）
◎：コマンドの実行が許可される。
○：あとから実行するコマンドは実行中のコマンドが終了するまで待つ。
×：コマンドの実行を中止する。排他エラーが返る。

ほかのコマンドの実行中に別のサーバ管理コマンドを実行しようとした場合，実行する

コマンドの種類によって，異なる形で制御されます。

(1) サーバ管理コマンドが処理をしている J2EE サーバに対して異なるサーバ管理コマンドを実行した場合の制御

サーバ管理コマンドが処理をしている J2EE サーバに対して異なる系統のサーバ管理コマンドを実行した場合，それぞれのサーバ管理コマンドの処理は並行して実行されます。また，サーバ管理コマンドが処理をしている J2EE サーバに対して，同じ系統のサーバ管理コマンドを同時に実行した場合，あとに実行したサーバ管理コマンドの処理要求は，実行された順番で待ち状態になり，先に実行したサーバ管理コマンドの処理が終了してから開始されます。したがって，同時に処理を実行できるサーバ管理コマンドの数は，それぞれの系統のサーバ管理コマンドで一つずつです。また，待ち状態にあるサーバ管理コマンドには最大数の制限はありません。処理の実行が待ち状態になると，J2EE サーバ側にメッセージ KDJE42211-I が表示されます。

(2) Server Plug-in が処理をしている J2EE サーバに対してサーバ管理コマンドを実行した場合の制御

Server Plug-in が処理をしている J2EE サーバに対しては，参照系と更新系に分類されるサーバ管理コマンドを実行することはできません。あとに実行したサーバ管理コマンドは排他エラーとなります。Server Plug-in の処理が終了してから次のサーバ管理コマンドを実行してください。なお，特権系に分類されるコマンドは実行できます。

(3) シャットダウンコマンドの処理中にサーバ管理コマンドを使用する場合の制御

シャットダウンコマンドが処理中，または実行待ち状態にある J2EE サーバに対しては，更新系と参照系に分類されるサーバ管理コマンドは実行できません。シャットダウン処理時に，参照系と更新系に分類されるサーバ管理コマンドを実行した場合は排他エラーとなり，メッセージ KDJE42212-E が表示されます。なお，特権系に分類されるサーバ管理コマンドは実行できます。

(4) 特権系コマンドを使用する場合の制御

特権系コマンドは，ほかのすべてのコマンドより優先して実行できます。また，特権系コマンドの実行中であっても，ほかのコマンドの実行に対して影響することはありません。ただし，一つの J2EE サーバに対して，同時に実行できる特権系コマンドは一つです。二つ以上の特権系コマンドを一つの J2EE サーバに実行した場合はエラーとなり，メッセージ KDJE42230-E が表示されます。

3.2.3 サーバ管理コマンドの排他制御の強制解除

しばらく時間を置いて，何度かサーバ管理コマンドを実行しても同じ排他エラーメッセージ（実行中コマンドの情報も同じ）が表示される場合，サーバ管理コマンドの異常終了などによって，サーバ管理コマンドの排他情報に矛盾が生じているおそれがありま

す。この場合，メッセージ中に示されている実行中コマンドが，実際には実行されていないことをよく確認した上で，サーバ管理コマンドの排他制御を強制解除し，排他情報をリセットしてください。

排他制御の強制解除を実施するには，強制解除の対象にする J2EE サーバを指定する必要があります。

注意

サーバ管理コマンド実行中に排他制御を解除すると，複数のサーバ管理コマンドが同時に実行され，J2EE サーバに矛盾が生じることがあります。このため，これらのサーバ管理コマンドが正常に実行されているときに誤って排他情報を解除しないように注意してください。

サーバ管理コマンドの排他制御を強制解除する手順を次に示します。

1. **排他制御を強制解除する前に，サーバ管理コマンドが実際には実行中でないことを確認します。**
   サーバ管理コマンドを実行しようとすると，メッセージ KDJE37057-E または KDJE37301-E が表示されます。これらのメッセージに示されるサーバ管理コマンドが実際には実行されていないことをよく確認してください。
2. cjresetsv **コマンドを実行して，排他制御を強制解除します。**
   実行形式

```
cjresetsv ［<J2EEサーバ名>］ ［-nameserver <プロバイダURL>］
```

実行例
cjresetsv MyServer

3.3　属性ファイルによるプロパティの設定

サーバ管理コマンドでは，属性ファイルを使って，J2EE リソースおよび J2EE アプリケーションのプロパティを設定します。

なお，cosminexus.xml を使用した属性の設定方法については，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「10.2　属性の管理」を参照してください。

3.3.1　J2EE リソースのプロパティの設定手順

リソースアダプタとリソースアダプタ以外の J2EE リソースのプロパティ設定手順について説明します。

(1) リソースアダプタのプロパティの設定手順

J2EE リソースアダプタとしてデプロイして使用するリソースアダプタのプロパティの設定手順を次に示します。

なお，J2EE アプリケーションに含まれるリソースアダプタのプロパティの設定手順については，「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。

1. **属性ファイルを取得します。**
   デプロイする前のリソースアダプタの属性ファイルを取得する場合は，cjgetresprop コマンドを実行します。リソースアダプタをデプロイしたあとで，リソースアダプタの属性ファイルを取得する場合は，cjgetrarprop コマンドを実行します。
   取得する属性ファイルは，Connector 属性ファイルです。
   cjgetresprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetresprop（リソースの属性の取得)」を参照してください。cjgetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetrarprop（RAR ファイルの属性の取得)」を参照してください。

2. **プロパティ設定項目を編集します。**
   取得した属性ファイルをテキストエディタで編集します。属性ファイルの編集項目については，各プロパティの設定の説明を参照してください。
   属性ファイルの詳細については，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 (アプリケーション／リソース定義)」の「4.1　Connector 属性ファイル」を参照してください。

3. **属性を設定します。**
   デプロイする前のリソースアダプタのプロパティを設定する場合は，cjsetresprop コマンドを実行します。デプロイしているリソースアダプタのプロパティを設定する場

合は，cjsetrarprop コマンドを実行します。
cjsetresprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetresprop（リソースの属性設定）」を参照してください。cjsetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetrarprop（RAR 属性設定）」を参照してください。

(2) リソースアダプタ以外の J2EE リソースのプロパティ設定手順

JavaBeans リソースおよびメールコンフィグレーションのプロパティの設定手順を次に示します。

1. **属性ファイルを取得します。**
   JavaBeans リソースの属性ファイルを取得する場合は，cjgetjbprop コマンドを実行します。メールコンフィグレーションの属性ファイルを取得する場合は，cjgetresprop コマンドを実行します。
   取得する属性ファイルは，次のとおりです。
   - JavaBeans リソースの場合
     JavaBeans リソース属性ファイル
   - メールコンフィグレーションの場合
     メール属性ファイル

   cjgetjbprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetjbprop（JavaBeans リソースの属性の取得）」を参照してください。cjgetresprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetresprop（リソースの属性の取得）」を参照してください。

2. **プロパティ設定項目を編集します。**
   取得した属性ファイルをテキストエディタで編集します。属性ファイルの編集項目については，各プロパティの設定の説明を参照してください。
   属性ファイルの詳細については，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（アプリケーション／リソース定義）」の「2.2.9　Connector 属性の詳細」を参照してください。

3. **属性を設定します。**
   JavaBeans リソースのプロパティを設定する場合は，cjsetjbprop コマンドを実行します。メールコンフィグレーションのプロパティを設定する場合は，cjsetresprop コマンドを実行します。
   cjsetjbprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetjbprop（JavaBeans リソースの属性設定）」を参照してください。cjsetresprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetresprop（リソースの属性設定）」を参照してください。

3.3.2 J2EE アプリケーションのプロパティの設定手順

J2EE アプリケーションのプロパティの設定手順を次に示します。

1. **属性ファイルを取得します。**
   アプリケーション属性，および J2EE アプリケーションを構成するコンポーネント（サーブレット・JSP，Enterprise Bean，リソースアダプタ）属性の属性ファイルを取得するため，cjgetappprop コマンドを実行します。
   取得する属性ファイルは，プロパティを設定する対象となるコンポーネントに応じて異なります。また，そのアプリケーションのコンポーネントの属性情報を一括して編集する場合は，アプリケーション統合属性ファイルを取得します。
   cjgetappprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetappprop（アプリケーションの属性の取得）」を参照してください。

2. **プロパティ設定項目を編集します。**
   取得した属性ファイルをテキストエディタで編集します。属性ファイルの編集項目については，各プロパティの設定の説明を参照してください。
   属性ファイルの詳細については，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（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。

3. **属性を設定します。**
   アプリケーション属性，および J2EE アプリケーションを構成するコンポーネント（サーブレット・JSP，Enterprise Bean，リソースアダプタ）属性の属性ファイルを設定するため，cjsetappprop コマンドを実行します。
   cjsetappprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetappprop（アプリケーションの属性設定）」を参照してください。

参考

J2EE アプリケーションに J2EE アプリケーションを構成するコンポーネント（サーブレット・JSP，Enterprise Bean，リソースアダプタ）を追加して，新たに J2EE アプリケーションを作成する場合，追加する前のコンポーネントのプロパティ定義は，cjgetresprop コマンドと cjsetresprop コマンドを使用してください。cjgetresprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetresprop（リソースの属性の取得）」を参照してください。cjsetresprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetresprop（リソースの属性設定）」を参照してください。

3.4 サーバ管理コマンドで指定するプロバイダ URL

サーバ管理コマンドでは，オプションに＜プロバイダ URL ＞という値を指定できます。

＜プロバイダ URL ＞を省略して，サーバ管理コマンドを実行した場合，
usrconf.properties ファイルに設定した値がデフォルトで指定されます。

usrconf.properties ファイルは次の場所に格納されています。

Windows の場合

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

UNIX の場合

```
/opt/Cosminexus/CC/admin/usrconf/usrconf.properties
```

usrconf.properties ファイルに設定した値で実行する場合，および cjlistapp コマンドで，プロバイダ URL を指定して実行する場合の例を次に示します。

usrconf.properties ファイルに設定した値で実行する場合の例

Windows の場合

```
＜Cosminexusのインストールディレクトリ＞¥CC¥admin¥bin¥cjlistapp
［＜サーバ名称＞］
```

UNIX の場合

```
/opt/Cosminexus/CC/admin/bin/cjlistapp ［＜サーバ名称＞］
```

プロバイダ URL を指定して実行する場合の例

Windows の場合

```
＜Cosminexusのインストールディレクトリ＞¥CC¥admin¥bin¥cjlistapp
［＜サーバ名称＞］ -nameserver ＜プロバイダURL＞
```

UNIX の場合

```
/opt/Cosminexus/CC/admin/bin/cjlistapp ［＜サーバ名称＞］
-nameserver ＜プロバイダURL＞
```

プロバイダ URL は，次の形式で指定します。

実行形式

```
＜プロトコル名＞::＜ホスト名＞:＜ポート番号＞
```

＜プロトコル名＞

CORBA ネーミングサービスのプロトコル名。corbaname 固定です。iioploc ま

たは iiopname を指定した場合，corbaname に読み替えられます。

＜ホスト名＞

CORBA ネーミングサービスの稼働しているホスト名。

＜ポート番号＞

CORBA ネーミングサービスの稼働しているポート番号。

プロバイダ URL で指定した CORBA ネーミングサービス上のサーバ管理コマンド排他情報が解除されます。

3.5　このマニュアルでのサーバ管理コマンドの記載方法

このマニュアルでは，サーバ管理コマンドを使用する場合，次のように記載しています。

- 実行形式および実行例では，その操作に主に関連する引数だけを示して説明しています。指定できる引数の詳細など，コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照してください。
- 実行するコマンドについては，コマンド名だけを表示しています。パスを指定してコマンドを実行する場合，コマンドの格納先を含めたパスを指定してください。
  コマンドの格納先は次のとおりです。

  Windows の場合

  ＜ Cosminexus のインストールディレクトリ＞ ¥CC¥admin¥bin¥

  UNIX の場合

  /opt/Cosminexus/CC/admin/bin/

4 リソースアダプタの設定

この章では，J2EE リソースアダプタとしてデプロイして使用するリソースアダプタのアプリケーション設定操作について説明します。J2EE リソースアダプタは，J2EE サーバに共有スタンドアロンモジュールとしてデプロイしたリソースアダプタです。
J2EE アプリケーションに含まれるリソースアダプタのアプリケーション設定操作については，「5. J2EE アプリケーションに含まれるリソースアダプタの設定」を参照してください。

4.1 リソースアダプタの設定の概要

リソースアダプタの設定とは，リソースアダプタを，J2EE アプリケーションから利用できる状態にするための操作です。

J2EE リソースアダプタとしてデプロイして使用できるリソースアダプタの種類とアプリケーション設定操作の概要について説明します。

4.1.1 利用できるリソースアダプタ

J2EE リソースアダプタとして，デプロイして利用できるリソースアダプタを次に示します。

- DB Connector
- DB Connector for Cosminexus RM および Cosminexus RM
- uCosminexus TP1 Connector
- TP1/Message Queue - Access
- そのほかの Connector 1.0 に準拠したリソースアダプタ，または Connector 1.5 に準拠したリソースアダプタ※

注※　そのほかの Connector 1.0 に準拠したリソースアダプタ，または Connector 1.5 に準拠したリソースアダプタについては，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「3.3.2　リソースアダプタの種類」を参照してください。

注意事項

J2EE リソースアダプタと J2EE アプリケーションに含まれるリソースアダプタは，J2EE アプリケーションから同時に使用できます。ただし，J2EE リソースアダプタと J2EE アプリケーションに含まれるリソースアダプタを，同じ表示名で同時に使用することはできません。同じ表示名に設定するとエラーとなります。

4.1.2 設定する項目と操作の概要

次のリソースアダプタの設定でのアプリケーション設定操作の概要を示します。

- データベースと接続するための設定
- データベースと接続するための設定（コネクションプールのクラスタ化の場合）
- そのほかのリソースと接続するための設定（リソースアダプタを使用する場合）
- リソースアダプタ共通の設定

データベース上のキューと接続するための設定については，マニュアル「Cosminexus Reliable Messaging」を参照してください。

(1) データベースと接続するための設定

DB Connector を使用してデータベースに接続する場合に必要な作業です。

表 4-1 データベースと接続するための設定の概要

設定項目	内容	参照先
DB Connector のインポート	DB Connector の RAR ファイルをインポートして，J2EE サーバから使用できるリソースに追加します。	4.2.1
DB Connector のプロパティ定義	データベースとの接続についての次の情報を設定します。 • DB Connector の一般情報 • コンフィグレーションプロパティ • 実行時プロパティ • 別名情報	4.2.2
DB Connector のデプロイ	インポートした RAR ファイルを基に DB Connector をデプロイします。DB Connector は，デプロイすると J2EE リソースアダプタとして利用できます。	4.2.3
DB Connector の接続テスト	DB Connector に設定した内容が正しいことを検証します。	4.2.4
DB Connector の開始	DB Connector を開始します。	4.2.5
DB Connector の停止	DB Connector を停止します。	4.2.6
DB Connector のアンデプロイ	デプロイされている DB Connector を削除します。リソースアダプタを入れ替えする場合に，必要に応じて実行してください。	4.2.7
DB Connector のエクスポート	J2EE リソースアダプタを RAR ファイルとしてエクスポートします。 必要に応じて実行してください。	4.2.8

(2) データベースと接続するための設定（コネクションプールのクラスタ化の場合）

DB Connector のコネクションプールをクラスタ化で使用する場合に必要な作業です。

表 4-2 データベースと接続するための設定（コネクションプールのクラスタ化の場合）の概要

設定項目	内容	参照先
メンバリソースアダプタ用 DB Connector の設定	メンバリソースアダプタ用 DB Connector をインポート，設定，接続テストまで実行します。	4.3.2
ルートリソースアダプタ用 DB Connector の設定	ルートリソースアダプタ用 DB Connector をインポート，設定，接続テストまで実行します。	4.3.3
メンバリソースアダプタ用 DB Connector の開始と停止	メンバリソースアダプタ用 DB Connector を開始します。また，開始状態にあるメンバリソースアダプタ用 DB Connector を停止します。	4.3.4

設定項目	内容	参照先
ルートリソースアダプタ用 DB Connector の開始と停止	ルートリソースアダプタ用 DB Connector を開始します。また，開始状態にあるルートリソースアダプタ用 DB Connector を停止します。	4.3.5
コネクションプールの状態の確認	メンバコネクションプールの状態を参照します。	4.3.6
コネクションプールの一時停止	メンバコネクションプールを一時停止の状態にします。	4.3.7
コネクションプールの再開	一時停止の状態のメンバコネクションプールを再開します。	4.3.8

(3) そのほかのリソースと接続するための設定（リソースアダプタを使用する場合）

リソースアダプタを使用して OpenTP1 などの各種リソースに接続する場合に必要な作業です。

表 4-3 そのほかのリソースと接続するための設定（リソースアダプタを使用する場合）の概要

設定項目	内容	参照先
リソースアダプタのインポート	リソースアダプタ（RAR ファイル）をインポートして，J2EE サーバから使用できるリソースに追加します。	4.4.1
リソースアダプタのプロパティ定義	リソースとの接続についての次の情報を設定します。 • リソースアダプタの一般情報 • コンフィグレーションプロパティ • 実行時プロパティ • 別名情報	4.4.2 または 4.4.3
リソースアダプタのデプロイ	インポートしたリソースアダプタを基にリソースアダプタをデプロイします。 リソースアダプタは，デプロイすると J2EE リソースアダプタとして利用できます。	4.4.4
J2EE リソースアダプタの接続テスト	リソースアダプタに設定した内容が正しいかどうかを検証します。	4.4.5
J2EE リソースアダプタの開始	J2EE リソースアダプタを開始します。	4.4.6
J2EE リソースアダプタの停止	J2EE リソースアダプタを停止します。	4.4.7
J2EE リソースアダプタのアンデプロイ	J2EE リソースアダプタを削除します。 リソースアダプタを入れ替えする場合に，必要に応じて実行してください。	4.4.8
J2EE リソースアダプタのエクスポート	J2EE リソースアダプタを RAR ファイルとしてエクスポートします。 必要に応じて実行してください。	4.4.9

(4) リソースアダプタ共通の設定

リソースアダプタに共通の設定です。

表 4-4 リソースアダプタに共通な設定の概要

設定項目	内容	参照先
J2EE リソースアダプタの状態と一覧の表示	デプロイされている J2EE リソースアダプタの状態，コネクション定義識別子の一覧などを参照します。	4.5
リソースアダプタの一覧の参照	インポートされているリソースアダプタの一覧，コネクション定義識別子の一覧などを参照します。	4.6
コネクションプールの状態の確認	リソースアダプタのコネクションプールの状態を参照します。また，必要に応じてコネクションプール内のコネクションを削除します。	4.7
JNDI 名前空間に登録されるリソースアダプタ名の参照と変更	リソースアダプタの JNDI 名前空間への登録名を参照して，必要に応じて別名を付与します。	4.8
リソースアダプタの削除	リソースアダプタを削除します。	4.9
リソースアダプタのコピー	リソースアダプタをほかのフォルダなどにコピーできます。	4.10

4.2 データベースと接続するための設定

データベースと接続するために，DB Connector の設定をします。DB Connector はデータベースと接続するためのリソースアダプタです。

ここで説明する DB Connector を使用したデータベースへの接続とは，JDBC インタフェースを使用してデータベースのテーブルだけにアクセスすることです。

JMS インタフェースを使用してキューにアクセスする場合は，DB Connector for Cosminexus RM および Cosminexus RM を使用してデータベースに接続します。詳細は，マニュアル「Cosminexus Reliable Messaging」を参照してください。

データベースをコネクションプールのクラスタ化で使用する場合は，「4.3　データベースと接続するための設定（コネクションプールのクラスタ化の場合）」を参照してください。

DB Connector の設定をするには，あらかじめ使用するデータベースの環境を設定しておいてください。また，使用する DB Connector の種類に応じて，HiRDB Type4 JDBC Driver，Oracle JDBC Thin Driver，SQL Server 2000 Driver for JDBC，SQL Server 2005 JDBC Driver，または SQL Server JDBC Driver の設定も必要です。DB Connector の設定の詳細については，マニュアル「Cosminexus アプリケーションサーバ システム構築・運用ガイド」の「7.5　データベースを使用するための設定」を参照してください。

DB Connector の設定は次の手順で実施します。

1. **DB Connector をインポートします。**
2. **プロパティを定義します。**
3. **DB Connector をデプロイします。**
   デプロイとは，DB Connector を J2EE サーバに共有スタンドアロンモジュール（J2EE リソースアダプタ）として配備することです。
4. **接続を確認します。**
   正しく接続できるかどうかは，接続テストによって確認できます。

デプロイした DB Connector は，J2EE アプリケーションのプロパティ設定で，リソースアダプタのリファレンスを解決する必要があります。詳細については，「9.3.3　リソースアダプタのリファレンス定義」を参照してください。

参考

DB Connector のプロパティを新規に設定する場合，Cosminexus Component Container が提供しているテンプレートファイルが利用できます。
Connector 属性ファイルのテンプレートファイルは，次に示すディレクトリに格納されています。

- Windows の場合
  <Cosminexus インストールディレクトリ >¥CC¥admin¥templates¥
- UNIX の場合
  /opt/Cosminexus/CC/admin/templates/

このテンプレートファイルを使用すると，DB Connector をインポートする前に，Connector 属性ファイルを編集しておくことができます。テンプレートファイルはコピーして使用してください。
Connector 属性ファイルのテンプレートファイル名については，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「4.1.14　Connector 属性ファイルのテンプレートファイル」を参照してください。
なお，すでにプロパティが設定されている DB Connector のプロパティを変更する場合は，テンプレートファイルは使用しないでください。インポートした DB Connector の Connector 属性を取得して，Connector 属性ファイルを編集してください。

4.2.1　DB Connector のインポート

次に示すコマンドを実行して DB Connector をインポートします。

実行形式

```
cjimportres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -f <ファイルパス>
```

<ファイルパス>には，RAR ファイルを指定してください。

RAR ファイルは，次のディレクトリに格納されています。

- Windows の場合
  < Cosminexus のインストールディレクトリ> ¥CC¥DBConnector¥
- UNIX の場合
  /opt/Cosminexus/CC/DBConnector/

DB Connector をリソースアダプタとしてインポートする場合，トランザクションの管理方法，および使用する JDBC ドライバの種類によって，次のどれかの RAR ファイルを指定します。なお，XDM/RD E2，SQL Server に接続する場合，グローバルトランザクションは使用できません。

● DBConnector_HiRDB_Type4_CP.rar

HiRDB Type4 JDBC Driver 用の DB Connector です。ローカルトランザクションを使用する場合，またはトランザクション管理なしで使用する場合（トランザクションのサポートレベルに LocalTransaction または NoTransaction を指定する場合）に選択します。
HiRDB Type4 JDBC Driver の ConnectionPoolDataSource を使用して HiRDB および XDM/RD E2 に接続します。

● DBConnector_HiRDB_Type4_XA.rar
HiRDB Type4 JDBC Driver 用の DB Connector です。グローバルトランザクションを使用する場合（トランザクションサポートレベルに XATransaction を指定する場合）に選択します。
HiRDB Type4 JDBC Driver の XADataSource を使用して HiRDB に接続します。

● DBConnector_Oracle_CP.rar
Oracle JDBC Thin Driver 用の DB Connector です。ローカルトランザクションを使用する場合，またはトランザクション管理なしで使用する場合（トランザクションのサポートレベルに LocalTransaction または NoTransaction を指定する場合）に選択します。
Oracle JDBC Thin Driver の ConnectionPoolDataSource を使用して Oracle に接続します。

● DBConnector_Oracle_XA.rar
Oracle JDBC Thin Driver 用の DB Connector です。グローバルトランザクションを使用する場合（トランザクションのサポートレベルに XATransaction を指定する場合）に選択します。
Oracle JDBC Thin Driver の XADataSource を使用して Oracle に接続します。

● DBConnector_SQLServer_CP.rar
SQL Server 2000 Driver for JDBC 用の DB Connector です。ローカルトランザクションを使用する場合，またはトランザクション管理なしで使用する場合（トランザクションサポートレベルに LocalTransaction または NoTransaction を指定する場合）に選択します。
SQL Server 2000 Driver for JDBC の ConnectionPoolDataSource を使用して SQL Server 2000 に接続します。

● DBConnector_SQLServer2005_CP.rar
SQL Server 2005 JDBC Driver 用の DB Connector です。ローカルトランザクションを使用する場合，またはトランザクション管理なしで使用する場合（トランザクションサポートレベルに NoTransaction または LocalTransaction を指定する場合）に選択します。
SQL Server 2005 JDBC Driver の ConnectionPoolDataSource を使用して，SQL Server 2005 に接続します。または，SQL Server JDBC Driver の ConnectionPoolDataSource を使用して，SQL Server 2008 に接続します。

実行例

```
cjimportres MyServer -type rar -f DBConnector_DABJ_CP.rar
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

4.2.2 DB Connector のプロパティ定義

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

プロパティの設定手順については，「3.3　属性ファイルによるプロパティの設定」を参照してください。次に DB Connector のプロパティ定義について説明します。

(1) 編集する属性ファイル

Connector 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して，DB Connector の Connector 属性ファイルを取得します。

実行形式

```
cjgetresprop ＜サーバ名称＞ [-nameserver ＜プロバイダURL＞] -type rar -resname ＜
DB Connectorの表示名＞ -c ＜Connector属性ファイルパス＞
```

実行例

```
cjgetresprop MyServer -type rar -resname account-rar -c
AccountProp.xml
```

● 属性の設定

次に示すコマンドを実行して，Connector 属性ファイルの値を反映します。

実行形式

```
cjsetresprop ＜サーバ名称＞ [-nameserver ＜プロバイダURL＞] -type rar -resname ＜
DB Connectorの表示名＞ -c ＜Connector属性ファイルパス＞
```

実行例

```
cjsetresprop MyServer -type rar -resname account-rar -c
```

```
AccountProp.xml
```

注意事項

リソースアダプタをデプロイしたあとでプロパティを定義する場合は，cjgetrarprop コマンドと cjsetrarprop コマンドを使用してください。cjgetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetrarprop（RAR ファイルの属性の取得)」を参照してください。cjsetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetrarprop（RAR 属性設定)」を参照してください。

(3) 編集する属性設定項目

DB Connector のプロパティ設定項目を次に示します。

- DB Connector の一般情報
- コンフィグレーションプロパティ
- 実行時プロパティ
- 別名情報

(a) DB Connector の一般情報

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

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

（凡例） ○：必須

プロパティの設定項目については，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「4.1.1　Connector 属性ファイルの指定内容」を参照してください。

(b) コンフィグレーションプロパティ

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

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

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

注※

コンフィグレーションプロパティの値をクリアする場合，空タグ（<config-property-value></config-property-value>）を指定します。<config-property-value> タグ自体がない場合は，コンフィグレーションプロパティの値は変更されません。

定義するコンフィグレーションプロパティの数だけ，<config-property> タグ下の設定を繰り返してください。

設定する必要がある項目は，インポートした DB Connector の種類によって一部異なります。<config-property> タグに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（アプリケーション／リソース定義）」の「4.1.10　DB Connector に設定する <config-property> タグに指定できるプロパティ」を参照してください。

なお，<config-property-name> タグに「XAOpenString」が設定されている場合，cjgetresprop コマンドを実行すると，コンフィグレーションプロパティの値（<config-property-value>）は次のように表示されます。

- プロパティの値が設定されている場合
  コンフィグレーションプロパティの値（<config-property-value>）は取得されないで，「<!-- The config-property-value has already been set. -->」と表示されます。「XAOpenString」の値を変更する場合には，<config-property-value> タグを追加して，変更後の値を設定してください。
- プロパティの値が設定されていない場合
  空タグ（<config-property-value></config-property-value>）が表示されます。「XAOpenString」の値を設定する場合には，<config-property-value> タグに値を追加してください。

ステートメントプーリング機能を使用する場合の，ステートメントプーリングの動作，および注意事項については，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「3.14.4　ステートメントプーリング」を参照してください。

使用する DB Connector のコンフィグレーションプロパティの設定例については，「(4) コンフィグレーションプロパティの設定例」を参照してください。

(c) 実行時プロパティ

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

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

注※

プロパティの値をクリアする場合，空タグ（<property-value></property-value>）を指定します。

<property-value> タグ自体がない場合は，プロパティの値は変更されません。

定義するプロパティの数だけ，上記の設定を繰り返してください。

プロパティの値（<property-value>）を設定した場合，プロパティ値のデフォルト（<property-default-value>）で設定されている値は無効となります。

プロパティ名（<property-name>）には，次の項目を設定します。

プロパティ項目	プロパティ名（<property-name>）の項目名
ユーザ名※	User
パスワード※	Password
コネクションプールにプールするコネクションの最小値	MinPoolSize
コネクションプールにプールするコネクションの最大値	MaxPoolSize
プール内のコネクションに障害が発生しているかどうかをチェックする方法の選択	ValidationType
プール内のコネクションに障害が発生しているかどうかのチェックを定期的に実行する場合の間隔	ValidationInterval
コネクションの取得に失敗した場合の，再取得する回数	RetryCount
コネクションの取得に失敗した場合の，再取得する間隔	RetryInterval
ログを出力するかどうかの選択	LogEnabled
コネクションの最終利用時刻から，コネクションを自動破棄（コネクションスイーパ）するかを判定するまでの時間	ConnectionTimeout
コネクションの自動破棄（コネクションスイーパ）が動作する間隔	SweeperInterval
コネクション枯渇時にコネクション取得要求をキューで管理するかどうかの選択	RequestQueueEnable
コネクション枯渇時のコネクション取得要求をキューで管理する場合の待ち時間の最大値	RequestQueueTimeout

プロパティ項目	プロパティ名（<property-name>）の項目名
コネクションプールを監視するかどうかの選択	WatchEnabled
コネクションプールを監視する間隔	WatchInterval
コネクションプール使用状態を監視するしきい値	WatchThreshold
コネクションプール監視結果のファイルを出力するかどうかの選択	WatchWriteFileEnabled
コネクション数調節機能が動作する間隔	ConnectionPoolAdjustmentInterval
コネクションプールのウォーミングアップ機能を有効にするかどうかの選択	Warmup
ネットワーク障害検知機能のタイムアウトを有効にするかどうかの選択	NetworkFailureTimeout

注※

<property-name> タグに「User」,「Password」が設定されている場合，cjgetresprop コマンドを実行すると，プロパティの値（<property-value>）は次のように表示されます。

- プロパティの値が設定されている場合
  プロパティの値（<property-value>）は取得されないで，「<!-- The property-value has already been set. -->」と表示されます。
  「User」および「Password」の値を変更する場合には，<property-value> タグを追加して，変更後の値を設定してください。
- プロパティの値が設定されていない場合
  空タグ（<property-value></property-value>）が表示されます。
  「User」および「Password」の値を設定する場合には，<property-value> タグに値を追加してください。

プロパティの設定項目については，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「4.1.1　Connector 属性ファイルの指定内容」を参照してください。

コネクションプーリング機能を使用する場合のコネクションプールの動作，および注意事項については，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「3.14.1　コネクションプーリング」を参照してください。

(d) 別名情報

DB Connector の別名情報（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <resource-external-property> タグ）の設定項目を次に示します。

項目	必須	対応するタグ
リソースの別名	○	<optional-name>

項目	必須	対応するタグ
リソース認証方式	△	<res-auth>
リソース共有の有無	△	<res-sharing-scope>

（凡例） ○：必須 △：任意

プロパティの設定項目については，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「4.1.1 Connector 属性ファイルの指定内容」を参照してください。

DB Connector の別名の使用法については，「4.8 JNDI 名前空間に登録されるリソースアダプタ名の参照と変更」を参照してください。

(4) コンフィグレーションプロパティの設定例

次の DB Connector について，コンフィグレーションプロパティの設定例を示します。

- DBConnector_HiRDB_Type4_CP.rar で，HiRDB，または XDM/RD E2 を使用する場合
- DBConnector_HiRDB_Type4_XA.rar で，HiRDB を使用する場合
- DBConnector_Oracle_CP.rar で，Oracle を使用する場合
- DBConnector_Oracle_XA.rar で，Oracle を使用する場合
- DBConnector_SQLServer_CP.rar で，SQL Server 2000 を使用する場合
- DBConnector_SQLServer2005_CP.rar で，SQL Server 2005，または SQL Server 2008 を使用する場合

(a) DBConnector_HiRDB_Type4_CP.rar で，HiRDB または XDM/RD E2 を使用する場合

DBConnector_HiRDB_Type4_CP.rar で，HiRDB または XDM/RD E2 を使用する場合の，コンフィグレーションプロパティの設定例を次の表に示します。

表 4-5 データベースとして HiRDB または XDM/RD E2 を使用する場合のコンフィグレーションプロパティの設定例（DBConnector_HiRDB_Type4_CP.rar の場合）

項目名	HiRDB の場合の設定例	XDM/RD E2 の場合の設定例
description	＜HiRDB ポート番号＞※	＜データベースコネクションサーバのサーバスケジュール番号＞※
DBHostName	＜HiRDB ホスト名＞	＜XDM/RD E2 ホスト名＞
environmentVariables	＜HiRDB クライアント環境変数名＞	＜HiRDB クライアント環境変数名＞
loginTimeout	8	8
encodeLang	—	—

項目名	HiRDB の場合の設定例	XDM/RD E2 の場合の設定例
JDBC_IF_TRC	false	false
TRC_NO	500	500
uapName	－	－
LONGVARBINARY_Access	REAL	REAL
SQLInNum	300	300
SQLOutNum	300	300
SQLWarningLevel	SQLWARN	SQLWARN
SQLWarningIgnore	false	false
HiRDBCursorMode	false	false
maxBinarySize	0	0
LONGVARBINARY_AccessSize	0	0
LONGVARBINARY_TruncError	true	true
PreparedStatementPoolSize	10	10
CallableStatementPoolSize	10	10
CancelStatement	true	true
logLevel	ERROR	ERROR

（凡例） －：設定は不要
注※ HiRDB クライアントの環境変数グループ名（Windows の場合）または環境変数グループの設定ファイルのパス（UNIX の場合）も指定できます。

(b) DBConnector_HiRDB_Type4_XA.rar で，HiRDB を使用する場合

DBConnector_HiRDB_Type4_XA.rar で，HiRDB を使用する場合の，コンフィグレーションプロパティの設定例を次の表に示します。

表 4-6 データベースとして HiRDB を使用する場合のコンフィグレーションプロパティの設定例（DBConnector_HiRDB_Type4_XA.rar の場合）

項目名	HiRDB の場合の設定例
description	＜環境変数グループ識別子＞
DBHostName	＜ HiRDB ホスト名＞
environmentVariables	＜ HiRDB クライアント環境変数名＞
XAOpenString	＜環境変数グループ識別子※1＞＋＜環境変数グループの設定ファイルのパス※2＞
loginTimeout	8
encodeLang	－
JDBC_IF_TRC	false

項目名	HiRDB の場合の設定例
TRC_NO	500
uapName	―
LONGVARBINARY_Access	REAL
SQLInNum	300
SQLOutNum	300
SQLWarningLevel	SQLWARN
SQLWarningIgnore	false
HiRDBCursorMode	false
maxBinarySize	0
LONGVARBINARY_AccessSize	0
LONGVARBINARY_TruncError	true
XACloseString	―
XALocalCommitMode	true
PreparedStatementPoolSize	10
CallableStatementPoolSize	10
CancelStatement	true
logLevel	ERROR

（凡例） ―：設定は不要
注※1 ［Description］フィールドに入力した値を指定します。
注※2 HiRDB の環境変数グループの設定ファイルのパスを指定します。詳細については，マニュアル「Cosminexus アプリケーションサーバ システム構築・運用ガイド」の「7.5.1 HiRDB の設定（HiRDB Type4 JDBC Driver の場合）」を参照してください。

(c) DBConnector_Oracle_CP.rar で，Oracle を使用する場合

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

表 4-7 データベースとして Oracle を使用する場合のコンフィグレーションプロパティの設定例（DBConnector_Oracle_CP.rar の場合）

項目名	Oracle の場合の設定例
databaseName	＜ Oracle SID ＞
serverName	＜ Oracle のホスト名称，または IP アドレス＞
portNumber	1521
url	―
loginTimeout	8000
PreparedStatementPoolSize	10

項目名	Oracle の場合の設定例
CallableStatementPoolSize	10
CancelStatement	true
ConnectionIDUpdate	false
logLevel	ERROR

（凡例） －：設定は不要

(d) DBConnector_Oracle_XA.rar で，Oracle を使用する場合

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

表 4-8 データベースとして Oracle を使用する場合のコンフィグレーションプロパティの設定例（DBConnector_Oracle_XA.rar の場合）

項目名	Oracle の場合の設定例
databaseName	＜ Oracle SID ＞
serverName	＜ Oracle のホスト名称，または IP アドレス＞
portNumber	1521
url	－
loginTimeout	8000
sessionTimeout	300
PreparedStatementPoolSize	10
CallableStatementPoolSize	10
CancelStatement	true
ConnectionIDUpdate	false
logLevel	ERROR

（凡例） －：設定は不要

(e) DBConnector_SQLServer_CP.rar で，SQL Server 2000 を使用する場合

DBConnector_SQLServer_CP.rar で，SQL Server 2000 を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 4-9 データベースとして SQL Server 2000 を使用する場合

項目名	SQL Server 2000 の場合の設定例
databaseName	＜ SQL Server 2000 のデータベース名＞
serverName	＜ SQL Server 2000 のホスト名または IP アドレス＞
hostProcess	0

項目名	SQL Server 2000 の場合の設定例
netAddress	―
loginTimeout	8
portNumber	＜SQL Server 2000 のポート番号＞
programName	―
selectMethod	cursor
sendStringParametersAsUnicode	true
WSID	―
PreparedStatementPoolSize	10
CallableStatementPoolSize	10
CancelStatement	true
logLevel	ERROR

（凡例） ―：設定は不要

(f) DBConnector_SQLServer2005_CP.rar で，SQL Server 2005，または SQL Server 2008 を使用する場合

DBConnector_SQLServer2005_CP.rar で，SQL Server 2005，または SQL Server 2008 を使用する場合のコンフィグレーションプロパティの設定例を次の表に示します。

表 4-10 データベースとして SQL Server 2005，または SQL Server 2008 を使用する場合

項目名	SQL Server 2005，または SQL Server 2008 の場合の設定例
databaseName	＜SQL Server 2005，または SQL Server 2008 のデータベース名＞
serverName	＜SQL Server 2005，または SQL Server 2008 のホスト名または IP アドレス＞
applicationName	＜SQL Server 2005，または SQL Server 2008 に接続するアプリケーション名＞
instanceName	＜接続する SQL Server 2005，または SQL Server 2008 のインスタンス名＞
lastUpdateCount	true
lockTimeout	-1
loginTimeout	8
portNumber	1433
selectMethod	cursor
sendStringParametersAsUnicode	true

項目名	SQL Server 2005，または SQL Server 2008 の場合の設定例
workstationID	＜アプリケーションサーバのホスト名＞
xopenStates	false
failoverPartner	＜データベースミラーリング構成で使用されるフェイルオーバサーバ名＞
integratedSecurity	false
packetSize	4096
PreparedStatementPoolSize	10
CallableStatementPoolSize	10
CancelStatement	true
logLevel	ERROR

4.2.3 DB Connector のデプロイ

DB Connector は，デプロイすると J2EE リソースアダプタとして使用できます。**J2EE リソースアダプタ**とは，J2EE サーバに共有スタンドアロンモジュールとして配備したリソースアダプタのことです。サーバ管理コマンドでインポートした，リソースアダプタをデプロイすると，その J2EE サーバ上で動作するすべての J2EE アプリケーションから使用できるようになります。なお，デプロイしたあとで，プロパティを定義することもできます。デプロイ後に定義する場合は，該当する DB Connector を停止した状態で実行してください。プロパティを定義する方法については，「4.2.2　DB Connector のプロパティ定義」を参照してください。

次に示すコマンドを実行して DB Connector をデプロイします。

実行形式

```
cjdeployrar ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -resname ＜DB Connectorの表示名＞
```

実行例

```
cjdeployrar MyServer -resname account-rar
```

cjdeployrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeployrar（リソースアダプタのデプロイ）」を参照してください。

4.2.4 DB Connector の接続テスト

DB Connector に設定した情報が正しいかどうか，接続テストによって検証します。

次に示すコマンドを実行して DB Connector の接続テストを実施します。

実行形式

```
cjtestres ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -type rar -resname ＜
DB Connectorの表示名>
```

実行例

```
cjtestres MyServer -type rar -resname account-rar
```

cjtestres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjtestres（リソースの接続テスト）」を参照してください。

注意事項

一度接続テストをした DB Connector は，J2EE サーバを再起動するまで削除できません。DB Connector を削除する場合は，DB Connector を停止してから，J2EE サーバを再起動してください。

4.2.5 DB Connector の開始

次に示すコマンドを実行して DB Connector を開始します。

実行形式

```
cjstartrar ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -resname ＜DB
Connectorの表示名>
```

実行例

```
cjstartrar MyServer -resname account-rar
```

cjstartrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstartrar（リソースアダプタの開始）」を参照してください。

注意事項

- J2EE アプリケーション中の J2EE リソースが DB Connector を参照している場合は，DB Connector を開始してから，J2EE アプリケーションを開始してください。
- 一度開始した DB Connector は，J2EE サーバを再起動するまで削除できません。DB Connector を削除する場合は，DB Connector を停止してから，J2EE サーバを再起動してください。

4.2.6 DB Connector の停止

次に示すコマンドを実行して DB Connector を停止します。

実行形式

```
cjstoprar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <DB
Connectorの表示名>
```

実行例

```
cjstoprar MyServer -resname account-rar
```

cjstoprar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstoprar（リソースアダプタの停止）」を参照してください。

注意事項

J2EE アプリケーション中の J2EE リソースが DB Connector を参照している場合は，J2EE アプリケーションを停止してから，DB Connector を停止してください。

4.2.7 DB Connector のアンデプロイ

準備

デプロイされている DB Connector を削除する前に，DB Connector を停止して，J2EE サーバを再起動してください。また，DB Connector に対し，一度でも開始または接続テストを試みた場合も同様に J2EE サーバを再起動してください。

次に示すコマンドを実行して DB Connector をアンデプロイします。

実行形式

```
cjundeployrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <DB
Connectorの表示名>
```

実行例

```
cjundeployrar MyServer -resname account-rar
```

cjundeployrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjundeployrar（リソースアダプタのアンデプロイ）」を参照してください。

4.2.8 DB Connector のエクスポート

DB Connector の内容を RAR ファイルとして出力（エクスポート）します。

次に示すコマンドを実行して DB Connector をエクスポートします。

実行形式

```
cjexportrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -f <ファイルパス>
-resname <DB Connectorの表示名>
```

実行例

```
cjexportrar MyServer -f res1.rar -resname account-rar
```

cjexportrar コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjexportrar（リソースアダプタのエクスポート）」を参照してください。

注意事項

Management Server を利用して運用している場合，サーバ管理コマンドと Management Server の実行ホストが異なるときは，DB Connector をエクスポートして，Management Server の実行ホストに格納する必要があります。

4.3 データベースと接続するための設定（コネクションプールのクラスタ化の場合）

DB Connector で，Oracle10g RAC のクラスタ化されたデータベースに接続する場合，コネクションプールをクラスタ化して使うことができます。コネクションプールのクラスタ化の詳細については，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「3.17　コネクションプールのクラスタ化機能」を参照してください。

この節では，DB Connector のコネクションプールをクラスタ化するための設定について説明します。

4.3.1 コネクションプールのクラスタ化の概要

クラスタ化されたコネクションプールをクラスタコネクションプールといいます。ルートリソースアダプタとメンバリソースアダプタで構成されます。メンバリソースアダプタのコネクションプールをメンバコネクションプールといいます。

DB Connector のクラスタコネクションプールを使用するための作業と状態の制御方法を次に示します。

(1) クラスタコネクションプールの設定

クラスタコネクションプールを使用するためには，クラスタコネクションプールを使用できるリソースアダプタの設定が必要です。

クラスタコネクションプール用の DB Connector の設定は，次の手順で実施します。

なお，手順 1，2 は，必要なメンバリソースアダプタの数だけ繰り返してください。

1. **メンバリソースアダプタ用 DB Connector を設定します。**
   メンバリソースアダプタ用 DB Connector を，次の手順で設定します。
   - メンバリソースアダプタ用の DB Connector の RAR ファイルをインポートします。
   - プロパティを定義します。
   - DB Connector をデプロイします。
   - 接続を確認します。
     正しく接続できるかどうかは，接続テストで確認できます。
2. **メンバリソースアダプタ用 DB Connector を開始します。**
3. **ルートリソースアダプタ用 DB Connector を設定します。**
   ルートリソースアダプタ用 DB Connector を，次の手順で設定します。
   - ルートリソースアダプタ用の DB Connector の RAR ファイルをインポートします。
   - プロパティを定義します。
   - DB Connector をデプロイします。

- 接続を確認します。
  正しく接続できるかどうかは，接続テストで確認できます。

4. **ルートリソースアダプタ用** DB Connector **を開始します。**

ルートリソースアダプタは，J2EE アプリケーションから直接アクセスされるので，デプロイしたルートリソースアダプタは，J2EE アプリケーションのプロパティ設定でリファレンスを解決する必要があります。詳細については，「9.3.3　リソースアダプタのリファレンス定義」を参照してください。

参考

DB Connector のプロパティを新規に設定する場合，Cosminexus Component Container が提供しているテンプレートファイルが利用できます。
Connector 属性ファイルのテンプレートファイルは，次に示すディレクトリに格納されています。

- Windows の場合
  <Cosminexus インストールディレクトリ >¥CC¥admin¥templates¥
- UNIX の場合
  /opt/Cosminexus/CC/admin/templates/

このテンプレートファイルを使用すると，DB Connector をインポートする前に，Connector 属性ファイルを編集しておくことができます。なお，テンプレートファイルはコピーして使用してください。
Connector 属性ファイルのテンプレートファイル名については，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「4.1.14　Connector 属性ファイルのテンプレートファイル」を参照してください。
なお，すでにプロパティが設定されている DB Connector のプロパティを変更する場合は，テンプレートファイルは使用しないでください。インポートした DB Connector の Connector 属性を取得して，Connector 属性ファイルを編集してください。

(2) クラスタコネクションプールの状態と実行できる操作

メンバコネクションプールは，データベースの障害や保守などの場合，手動で一時停止できます。一時停止状態になると，ルートリソースアダプタへのコネクション取得要求時に処理が実行されません。

また，一時停止したメンバコネクションプールは，手動で再開できます。ルートリソースアダプタへのコネクション取得要求時には，開始状態のコネクションプールだけ処理が実行されます。

コネクションプールの状態制御については，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「3.17.3　クラスタコネクションプールの動作」を参照してください。

(3) リソースアダプタの種類による機能差異

リソースアダプタの種類で，使用できる機能を次に示します。

表 4-11 リソースアダプタの種類による機能一覧

機能	リソースアダプタの種類	
	ルートリソースアダプタ	メンバリソースアダプタ
コネクションプーリング	×	◎
コネクションプールのウォーミングアップ	×	○
コネクションシェアリング・アソシエーション	×	○
ステートメントプーリング	×	○
DataSource オブジェクトのキャッシング	○	×
DB Connector のコンテナ管理でのサインオンの最適化	×	○
コネクション障害検知	×	◎
コネクション障害検知のタイムアウト	×	◎
コネクション枯渇時のコネクション取得待ち	×	◎
コネクション取得リトライ	×	×
コネクションプール情報表示	×	○
コネクションプールクリア	×	○
コネクション自動クローズ	×	○
コネクションスイーパ	×	○
接続テスト	○	○
コネクションプール数調節機能	×	○
コネクションプール情報の表示	×	○
コネクションプールの一時停止※	×	○
コネクションプールの再開※	×	○
J2EE リソースのユーザ指定名前空間機能	○	×

（凡例） ◎：必ず有効になる ○：使用できる ×：使用できない
注※ コネクションプールをクラスタで使用しない場合，コネクションプールの一時停止および再開は実行できません。

DB Connector の属性を設定するリソースアダプタの種類を次に示します。

表 4-12 リソースアダプタの種類と属性設定

設定項目	リソースアダプタの種類	
	ルートリソースアダプタ	メンバリソースアダプタ
トランザクションサポートレベル	×	○
ログ取得可否	○	○
データベース接続情報	－	○
DB Connector 固有の設定（ステートメントプールなど）	－	○
セキュリティ情報（ユーザ名，パスワード）	×	○
コネクションプールサイズ	×	○
クラスタコネクションプール固有の設定	○	－

（凡例） ○：設定要 ×：設定不要 －：設定項目なし

4.3.2 メンバリソースアダプタ用 DB Connector の設定

メンバリソースアダプタ用 DB Connector を，次の手順で設定します。

1. **メンバリソースアダプタ用の** DB Connector **をインポートします。**
2. **プロパティを定義します。**
3. **メンバリソースアダプタ用の** DB Connector **をデプロイします。**
4. **接続を確認します。**

（1） メンバリソースアダプタ用の DB Connector のインポート

次に示すコマンドを実行してメンバリソースアダプタ用の DB Connector をインポートします。

実行形式

```
cjimportres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -f <ファイルパス>
```

<ファイルパス>には，RAR ファイルを指定してください。

RAR ファイルは，次のディレクトリに格納されています。

- Windows の場合
  < Cosminexus のインストールディレクトリ> ¥CC¥DBConnector¥ClusterPool¥
- UNIX の場合

```
/opt/Cosminexus/CC/DBConnector/ClusterPool/
```

メンバリソースアダプタとしてインポートする RAR ファイルについて説明します。

DBConnector_Oracle_CP_ClusterPool_Member.rar

クラスタコネクションプールのメンバリソースアダプタです。ローカルトランザクションまたはトランザクションなし（トランザクションサポートレベルに LocalTransaction または NoTransaction を指定する）で使用します。Oracle JDBC Thin Driver の ConnectionPoolDataSource を使用して，Oracle に接続します。J2EE アプリケーションのリソースリファレンスに設定して使用することはできません。

実行例

```
cjimportres MyServer -type rar -f "c:¥Program
Files¥Hitachi¥Cosminexus¥CC¥DBConnector¥ClusterPool¥DBConnector
_Oracle_CP_ClusterPool_Member.rar"
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

(2) メンバリソースアダプタ用の DB Connector のプロパティ定義

メンバリソースアダプタ用の DB Connector のプロパティを定義します。プロパティを定義する手順については，「4.2.2　DB Connector のプロパティ定義」を参照してください。ここでは，メンバリソースアダプタ用の DB Connector のプロパティの設定項目を説明します。

(a) メンバリソースアダプタ用 DB Connector の一般情報

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

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

（凡例）　○：必須
注※　一つのクラスタコネクションプールを構成するメンバリソースアダプタのトランザクションサポートレベルは，すべて同じにしてください。

プロパティの設定項目については，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「4.1.1　Connector 属性ファイルの指定内容」を参照してください。

(b) メンバリソースアダプタ用コンフィグレーションプロパティ

メンバリソースアダプタ用 DB Connector のコンフィグレーションプロパティ（<config-property> タグ）と設定内容は，対応するリソースアダプタ（DBConnector_Oracle_CP.rar）と同じです。対応するリソースアダプタのコンフィグレーションプロパティについては，「4.2.2　DB Connector のプロパティ定義」を参照してください。

(c) 実行時プロパティ

メンバリソースアダプタ用 DB Connector の実行時プロパティ（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> タグ）の設定項目を次に示します。

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

定義するプロパティの数だけ，上記の設定を繰り返してください。

プロパティ名（<property-name>）には，次の項目を設定します。

プロパティ項目	プロパティ名（<property-name>）の設定項目
ユーザ名※	User
パスワード※	Password
コネクションプールにプールするコネクションの最小値	MinPoolSize
コネクションプールにプールするコネクションの最大値	MaxPoolSize
ログを出力するかどうかの選択	LogEnabled
コネクションの最終利用時刻から，コネクションを自動破棄（コネクションスイーパ）するかを判定するまでの時間	ConnectionTimeout
コネクションの自動破棄（コネクションスイーパ）が動作する間隔	SweeperInterval
コネクション枯渇時のコネクション取得要求をキューで管理する場合の待ち時間の最大値	RequestQueueTimeout
コネクションプールを監視するかどうかの選択	WatchEnabled
コネクションプールを監視する間隔	WatchInterval
コネクションプール使用状態を監視するしきい値	WatchThreshold

プロパティ項目	プロパティ名（<property-name>）の設定項目
コネクションプール監視結果のファイルを出力するかどうかの選択	WatchWriteFileEnabled
コネクション数調節機能が動作する間隔	ConnectionPoolAdjustmentInterval
コネクションプールのウォーミングアップ機能を有効にするかどうかの選択	Warmup

注※　一つのクラスタコネクションプールを構成するメンバリソースアダプタのユーザ名は，すべて同じにしてください。

注意事項

メンバリソースアダプタでは，次の項目は設定の有無に関係なく，常に「有効」になります。

プロパティ項目	有効値	プロパティ名（<property-name>）の設定項目
コネクションプールにプールするコネクションの最小値	コネクションプーリング機能は常に有効「MinPoolSize」，「MaxPoolSize」に「0」を設定しても，デフォルト値の「10」が仮定されます。	MinPoolSize
コネクションプールにプールするコネクションの最大値		MaxPoolSize
プール内のコネクションに障害が発生しているかどうかをチェックする方法の選択	常に「1」（コネクション取得時の障害検知）	ValidationType
コネクション枯渇時にコネクション取得要求をキューで管理するかどうかの選択	常に「true」	RequestQueueEnable
ネットワーク障害検知機能のタイムアウトを有効にするかどうかの選択	常に「true」	NetworkFailureTimeout

また，実行時プロパティのコネクションリトライ回数（「RetryCount」）とコネクションリトライ待ち時間（「RetryInterval」）の設定に関係なく，コネクション取得リトライ機能は，常に無効となります。

(3) メンバリソースアダプタ用の DB Connector のデプロイ

メンバリソースアダプタ用の DB Connector は，デプロイすると J2EE リソースアダプタとして使用できます。なお，デプロイしたあとで，プロパティを定義することもできます。デプロイ後に定義する場合は，該当するメンバリソースアダプタ用の DB Connector が所属するルートリソースアダプタと，メンバリソースアダプタ用の DB Connector を停止した状態で実行してください。プロパティを定義する方法については，「(2)　メンバリソースアダプタ用の DB Connector のプロパティ定義」を参照してくださ

い。

次に示すコマンドを実行してメンバリソースアダプタ用の DB Connector をデプロイします。

実行形式

```
cjdeployrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <メンバリソースアダプタ用のDB Connector表示名>
```

実行例

```
cjdeployrar MyServer -resname
DB_Connector_for_Oracle_ClusterPool_Member
```

cjdeployrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeployrar（リソースアダプタのデプロイ）」を参照してください。

(4) メンバリソースアダプタ用の DB Connector の接続テスト

メンバリソースアダプタ用の DB Connector に設定した情報が正しいかどうか，接続テストで検証します。

次に示すコマンドを実行して，メンバリソースアダプタ用の DB Connector の接続テストを実施します。

実行形式

```
cjtestres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname <メンバリソースアダプタ用のDB Connectorの表示名>
```

実行例

```
cjtestres -type rar -resname
DB_Connector_for_Oracle_ClusterPool_Member
```

cjtestres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjtestres（リソースの接続テスト）」を参照してください。

注意事項

一度接続テストをしたメンバリソースアダプタ用の DB Connector は，J2EE サーバを再起動するまで削除できません。メンバリソースアダプタ用の DB Connector を削除する場合は，そのメンバリソースアダプタ用の DB Connector が所属するルートリソースアダプタとメンバリソースアダプタ用の DB Connector を停止してから，

J2EE サーバを再起動してください。

4.3.3 ルートリソースアダプタ用 DB Connector の設定

ルートリソースアダプタ用 DB Connector を，次の手順で設定します。

1. **ルートリソースアダプタ用の** DB Connector **をインポートします。**
2. **プロパティを定義します。**
3. **ルートリソースアダプタ用の** DB Connector **をデプロイします。**
4. **接続を確認します。**

（1）ルートリソースアダプタ用の DB Connector のインポート

次に示すコマンドを実行してルートリソースアダプタ用の DB Connector をインポートします。

実行形式

```
cjimportres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -f <ファイルパス>
```

<ファイルパス>には，RAR ファイルを指定してください。

RAR ファイルは，次のディレクトリに格納されています。

- Windows の場合
  < Cosminexus のインストールディレクトリ> ¥CC¥DBConnector¥ClusterPool¥
- UNIX の場合
  /opt/Cosminexus/CC/DBConnector/ClusterPool/

ルートリソースアダプタとしてインポートする RAR ファイルについて説明します。

DBConnector_CP_ClusterPool_Root.rar

クラスタコネクションプールのルートリソースアダプタです。属するメンバリソースアダプタがローカルトランザクションまたはトランザクションなし（トランザクションサポートレベルに LocalTransaction または NoTransaction を指定する）で，データベースに接続する場合に使用します。
J2EE アプリケーションのリソースリファレンスに設定して使用します

実行例

```
cjimportres MyServer -type rar -f "c:¥Program Files¥Hitachi¥Cosminexus¥CC¥DBConnector¥ClusterPool¥DBConnector_CP_ClusterPool_Root.rar"
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーション

サーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

(2) ルートリソースアダプタ用の DB Connector のプロパティ定義

ルートリソースアダプタ用の DB Connector のプロパティを定義します。プロパティを定義する手順については，「4.2.2　DB Connector のプロパティ定義」を参照してください。ここでは，ルートリソースアダプタ用の DB Connector のプロパティの設定項目を説明します。

ルートリソースアダプタ用の DB Connector で，有効なプロパティを次に示します。下記以外の項目は，設定しても無視されます。

- ルートリソースアダプタ用の DB Connector の説明（<description>）
- ルートリソースアダプタ用の DB Connector 名称（<display-name>）
- コンフィグレーションプロパティ（<outbound-resourceadapter> - <connection-definition> - <config-property>）
- 実行時プロパティ（<outbound-resourceadapter> - <connection-definition> - <connector-runtime>）のログを出力するかどうかの選択（<LogEnabled>）
- 別名情報（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <resource-external-property>）

コンフィグレーションプロパティの設定項目を次に示します。

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

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

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

表 4-13　DBConnector_CP_ClusterPool_Root.rar を使用する場合のコンフィグレーションプロパティの設定例

項目名	設定例
algorithm	RoundRobin
enableAutoPoolSuspend	true
enableAutoPoolResume	true
dbCheckInterval	30
memberResourceAdapterName1	DB_Connector_for_Oracle_ClusterPool_Member1

項目名	設定例
memberResourceAdapterName2	DB_Connector_for_Oracle_ClusterPool_Member2
logLevel	ERROR

コンフィグレーションプロパティ名（<config-property-name>）の「memberResourceAdapterName[n]」プロパティは，メンバリソースアダプタの表示名を設定します。memberResourceAdapterName[n] は，デフォルトでは優先度 2 まで定義してあります。さらに，メンバリソースアダプタを指定する場合には，プロパティを追加してください。優先度 n は，1 ～ 100 の範囲で指定してください。n は，連続している必要はありません。

(3) ルートリソースアダプタ用の DB Connector のデプロイ

ルートリソースアダプタ用の DB Connector は，デプロイすると J2EE リソースアダプタとして使用できます。なお，デプロイしたあとで，プロパティを定義することもできます。デプロイ後に定義する場合は，該当するルートリソースアダプタ用の DB Connector を停止した状態で実行してください。プロパティを定義する方法については，「(2) ルートリソースアダプタ用の DB Connector のプロパティ定義」を参照してください。

次に示すコマンドを実行してルートリソースアダプタ用の DB Connector をデプロイします。

実行形式

```
cjdeployrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <ルートリソースアダプタ用のDB Connector表示名>
```

実行例

```
cjdeployrar MyServer -resname DB_Connector_for_ClusterPool_Root
```

cjdeployrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeployrar（リソースアダプタのデプロイ）」を参照してください。

(4) ルートリソースアダプタ用の DB Connector の接続テスト

ルートリソースアダプタ用の DB Connector に設定した情報が正しいかどうか，接続テストで検証します。

次に示すコマンドを実行して，ルートリソースアダプタ用の DB Connector の接続テストを実施します。

実行形式

```
cjtestres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname <
ルートリソースアダプタ用のDB Connectorの表示名>
```

実行例

```
cjtestres -type rar -resname DB_Connector_for_ClusterPool_Root
```

cjtestres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjtestres（リソースの接続テスト）」を参照してください。

注意事項

一度接続テストをしたルートリソースアダプタ用の DB Connector は，J2EE サーバを再起動するまで削除できません。ルートリソースアダプタ用の DB Connector を削除する場合は，ルートリソースアダプタ用の DB Connector を停止してから，J2EE サーバを再起動してください。

4.3.4 メンバリソースアダプタ用 DB Connector の開始と停止

(1) メンバリソースアダプタ用の DB Connector の開始

次に示すコマンドを実行してメンバリソースアダプタ用の DB Connector を開始します。

実行形式

```
cjstartrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <メンバリソー
スアダプタ用のDB Connectorの表示名>
```

実行例

```
cjstartrar MyServer -resname
DB_Connector_for_Oracle_ClusterPool_Member
```

cjstartrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstartrar（リソースアダプタの開始）」を参照してください。

(2) メンバリソースアダプタ用の DB Connector の停止

次に示すコマンドを実行して，メンバリソースアダプタ用の DB Connector を停止します。

実行形式

```
cjstoprar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <メンバリソースアダプタ用のDB Connectorの表示名>
```

実行例

```
cjstoprar MyServer -resname
DB_Connector_for_Oracle_ClusterPool_Member
```

cjstoprar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstoprar（リソースアダプタの停止）」を参照してください。

注意事項

メンバリソースアダプタが所属するルートリソースアダプタを停止してから，メンバリソースアダプタを停止してください。

4.3.5 ルートリソースアダプタ用 DB Connector の開始と停止

(1) ルートリソースアダプタ用の DB Connector の開始

次に示すコマンドを実行してルートリソースアダプタ用の DB Connector を開始します。

実行形式

```
cjstartrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <ルートリソースアダプタ用のDB Connectorの表示名>
```

実行例

```
cjstartrar MyServer -resname DB_Connector_for_ClusterPool_Root
```

cjstartrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstartrar（リソースアダプタの開始）」を参照してください。

注意事項

ルートリソースアダプタに所属するメンバリソースアダプタを開始してから，ルートリソースアダプタを開始してください。

(2) ルートリソースアダプタ用の DB Connector の停止

次に示すコマンドを実行して，ルートリソースアダプタ用の DB Connector を停止します。

実行形式

```
cjstoprar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <ルートリソースアダプタ用のDB Connectorの表示名>
```

実行例

```
cjstoprar MyServer -resname DB_Connector_for_ClusterPool_Root
```

cjstoprar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstoprar（リソースアダプタの停止）」を参照してください。

注意事項

J2EE アプリケーション中の J2EE リソースがルートリソースアダプタ用の DB Connector を参照している場合は，J2EE アプリケーションを停止してから，ルートリソースアダプタ用の DB Connector を停止してください。

4.3.6 コネクションプールの状態の確認

コネクションプールのクラスタ化で使用しているコネクションプールの状態は，次の二つの方法で参照できます。

- cjlistrar コマンドで，デプロイされているすべてのリソースアダプタの，リソースアダプタ名とメンバコネクションプールの状態を参照します。
- cjlistpool コマンドで，メンバコネクションプールの情報を参照します。
  メンバコネクションプールの情報を参照して，必要に応じてメンバコネクションプールを閉塞 / 一時停止 / 再開します。コネクションプールの状態定義については，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「3.17.3　クラスタコネクションプールの動作」を参照してください。

(1) コネクションプールの状態の参照

デプロイされているすべてのリソースアダプタについて，リソースアダプタ名とリソースアダプタの状態が表示されます。リソースアダプタがクラスタコネクションプールのメンバリソースアダプタの場合には，コネクションプールの状態も表示されます。

次に示すコマンドを実行して，コネクションプールの状態を参照します。

実行形式

```
cjlistrar ［<サーバ名>］ ［-nameserver <プロバイダURL>］ -clusterpool
```

実行例

```
cjlistrar MyServer -clusterpool
```

cjlistrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistrar（リソースアダプタの一覧表示）」を参照してください。

(2) メンバコネクションプールの情報の参照

コネクションプールの情報が表示されます。リソースアダプタがクラスタコネクションプールのメンバリソースアダプタの場合には，メンバコネクションプールの情報も表示されます。

次に示すコマンドを実行して，すべてのリソースアダプタのコネクションプール情報を参照します。

実行形式

```
cjlistpool ［<サーバ名>］ ［-nameserver <プロバイダURL>］ -resall
```

実行例

```
cjlistpool MyServer -resall
```

特定のリソースアダプタについて，コネクションプールの情報を表示する場合は次のコマンドを実行します。

実行形式

```
cjlistpool ［<サーバ名>］ -resname <リソースアダプタの表示名>
```

実行例

```
cjlistpool MyServer -resname
DB_Connector_for_Oracle_ClusterPool_Member
```

リソースアダプタのコネクションプール情報が表示されます。

cjlistpool コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistpool（コネクションプールの一覧表示）」を参照してください。

注意事項

cjlistpool コマンドは，ルートリソースアダプタに対しては実行できません。

4.3.7 コネクションプールの一時停止

データベースの障害や保守などの場合，メンバコネクションプールを手動で一時停止で

きます。一時停止したコネクションプールは，コネクション取得要求を受け付けません。

次のコマンドを実行して，クラスタコネクションプールのメンバコネクションプールを一時停止します。

実行形式

```
cjsuspendpool ［<サーバ名>］ ［-nameserver <プロバイダURL>］ -resname <一時停止対象となるメンバリソースアダプタの表示名>
```

実行例

```
cjsuspendpool MyServer -resname
DB_Connector_for_Oracle_ClusterPool_Member
```

cjsuspendpool コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsuspendpool（メンバコネクションプールの一時停止）」を参照してください。

注意事項

cjsuspendpool コマンドを使用して，手動で一時停止したコネクションプールは，自動再開はできません。cjresumepool コマンドで手動再開してください。

4.3.8 コネクションプールの再開

一時停止したメンバコネクションプールは手動で再開できます。J2EE アプリケーションがルートリソースアダプタにコネクション取得要求をしたときに，再開したコネクションプールは，再びコネクション取得要求を受け付けることができるようになります。

次のコマンドを実行して，クラスタコネクションプールのメンバコネクションプールを再開します。

実行形式

```
cjresumepool ［<サーバ名>］ ［-nameserver <プロバイダURL>］ -resname <再開対象となるメンバリソースアダプタの表示名>
```

実行例

```
cjresumepool MyServer -resname
DB_Connector_for_Oracle_ClusterPool_Member
```

cjresumepool コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjresumepool（メンバコネクションプールの再開始）」を参照してください。

4.4　そのほかのリソースと接続するための設定

OpenTP1 などのリソースとの接続には，リソースアダプタを使用します。ここでは，DB Connector，DB Connector for Cosminexus RM，および Cosminexus RM 以外のリソースアダプタと接続するための基本的な設定について説明します。

リソースアダプタは，次の手順で設定します。

1. **リソースアダプタをインポートします。**
2. **プロパティを定義します。**
3. **リソースアダプタをデプロイします。**
   リソースアダプタのデプロイとは，リソースアダプタを J2EE サーバに共有スタンドアロンモジュール（J2EE リソースアダプタ）として配備することです。
4. **接続を確認します。**
   正しく接続できるかどうかは，接続テストによって確認できます。

デプロイをした J2EE リソースアダプタは，J2EE アプリケーションのプロパティ設定で，リソースアダプタのリファレンスを解決する必要があります。詳細については，「9.3.3　リソースアダプタのリファレンス定義」を参照してください。

ポイント

そのほかのリソースと接続するリソースアダプタとして，次のリソースアダプタを使用できます。

- Connector 1.0 に準拠するリソースアダプタ
- Connector 1.5 に準拠するリソースアダプタ

Connector 1.0 の仕様に準拠するリソースアダプタのプロパティの定義については，「4.4.2　リソースアダプタのプロパティ定義（Connector 1.0 の場合）」を，Connector 1.5 の仕様に準拠するリソースアダプタのプロパティの定義については，「4.4.3　リソースアダプタのプロパティ定義（Connector 1.5 の場合）」を参照してください。

4.4.1　リソースアダプタのインポート

リソースアダプタをインポートします。

なお，アプリケーションサーバで提供されていない独自のリソースアダプタをインポートする場合は，注意事項に記載されている内容に従ってインポートしてください。

次に示すコマンドを実行してリソースアダプタをインポートします。

実行形式

```
cjimportres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -f <ファイルパス>
```

実行例

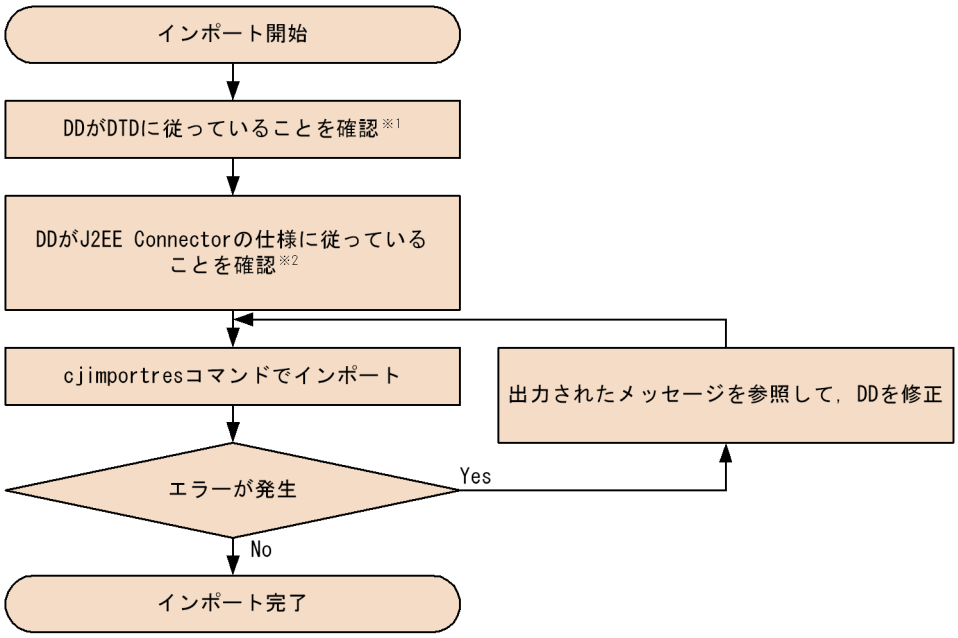
```
cjimportres MyServer -type rar -f mqcadpt.rar
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

注意事項

- Connector 1.0 以降の仕様に準拠しない DD を持つ RAR はインポートできません。インポートできるのは，Connector 1.0 以降の仕様に準拠した DD を持つ RAR だけです。DD が DTD に従っていることを確認してから，インポートを実行してください。
  インポート手順を次に示します。

図 4-1 リソースアダプタ（RAR）のインポート手順



注※ 1

DD は検証済み XML 文書として扱われます。このため，DD 中の DOCTYPE 宣言が誤っていた場合，または DD が DTD の仕様に従っていない場合はエラーになります。

注※ 2

J2EE Connector の仕様に従った値が DD に指定されていない場合はエラーとなります。詳細は，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「8.8.3　リソースアダプタのインポートと開始」を参照してください。

また，メッセージが出力された場合は，マニュアル「Cosminexus アプリケーションサーバ メッセージ 2」を参照して対処ください。

- インポート時に指定した RAR ファイル名は作業ディレクトリ中のディレクトリ名として使用されます。また，RAR ファイル内の JAR ファイルやネイティブライブラリは作業ディレクトリ内に展開されます。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 RAR ファイル名および RAR ファイル中のファイル名を指定してください。J2EE アプリケーションを実行するシステムの場合の作業ディレクトリのパス長の見積もりについては，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「7.10.2 作業ディレクトリのパス長の見積もり式」を参照してください。バッチアプリケーションを実行するシステムの場合の作業ディレクトリのパス長の見積もりについては，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「7.11.2　作業ディレクトリのパス長の見積もり式」を参照してください。
- ra.xml の display-name タグが設定されていない場合，RAR ファイル名を基に Display name が付与されます。
- RAR ファイルには，次の名称のファイルを含めないでください。
  rar.properties
  fileinfo.properties
  META-INF/hitachi-ra.xml

4.4.2　リソースアダプタのプロパティ定義（Connector 1.0 の場合）

Connector 1.0 の仕様に準拠するリソースアダプタのプロパティを定義します。リソースアダプタのプロパティ定義は，J2EE リソースアダプタをデプロイしたあとでも実行できます。なお，設定済みのリソースアダプタのプロパティを変更する場合は，該当するリソースアダプタを停止した状態で実行してください。

プロパティの設定手順の概要については，「3.3　属性ファイルによるプロパティの設定」を参照してください。次にリソースアダプタのプロパティ定義について説明します。

（1）編集する属性ファイル

Connector 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して，リソースアダプタの Connector 属性ファイルを取得します。

実行形式

```
cjgetresprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname
<リソースアダプタの表示名> -c <Connector属性ファイルパス>
```

実行例

```
cjgetresprop MyServer -type rar -resname Rar1 -c AccountProp.xml
```

● 属性の設定

次に示すコマンドを実行して，Connector 属性ファイルの値を反映します。

実行形式

```
cjsetresprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname
<リソースアダプタ表示名> -c <Connector属性ファイルパス>
```

実行例

```
cjsetresprop MyServer -type rar -resname Rar1 -c AccountProp.xml
```

! 注意事項

リソースアダプタをデプロイしたあとでプロパティを定義する場合は，cjgetrarprop コマンドと cjsetrarprop コマンドを使用してください。cjgetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetrarprop（RAR ファイルの属性の取得)」を参照してください。cjsetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetrarprop（RAR 属性設定)」を参照してください。

(3) 編集する属性設定項目

リソースアダプタのプロパティ設定項目を次に示します。

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

(a) リソースアダプタの一般情報

リソースアダプタの一般情報（<outbound-resourceadapter> タグ）の設定項目を次に示します。

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

（凡例） ○：必須

プロパティの設定項目については，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「4.1.1　Connector 属性ファイルの指定内容」を参照してください。

(b) コンフィグレーションプロパティ

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

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

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

定義するコンフィグレーションプロパティ名（<config-property-name>）については，「4.2.2　DB Connector のプロパティ定義」を参照してください。

(c) 実行時プロパティ

リソースアダプタの実行時プロパティ（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> タグ）の設定項目を次に示します。

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

定義するプロパティの数だけ，上記の設定を繰り返してください。

定義する実行時プロパティ名（<property-name>）およびコネクションプールの動作と注意事項については，「4.2.2　DB Connector のプロパティ定義」を参照してください。

(d) 別名情報

リソースアダプタの別名情報（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <resource-external-property> タグ）の設定項目を次に示します。

項目	必須	対応するタグ
リソースの別名	○	<optional-name>
リソース認証方式	△	<res-auth>
リソース共有の有無	△	<res-sharing-scope>

（凡例） ○：必須　△：任意

プロパティの設定項目については，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「4.1.1　Connector 属性ファイルの指定内容」を参照してください。

リソースアダプタの別名の使用法については，「4.8　JNDI 名前空間に登録されるリソースアダプタ名の参照と変更」を参照してください。

4.4.3 リソースアダプタのプロパティ定義（Connector 1.5 の場合）

Connector 1.5 の仕様に準拠するリソースアダプタに接続する場合のプロパティを定義します。

リソースアダプタのプロパティ定義は，J2EE リソースアダプタをデプロイしたあとでも実行できます。なお，リソースアダプタの設定済みのプロパティを変更する場合は，該当するリソースアダプタを停止した状態で実行してください。

プロパティの設定手順の概要については，「3.3　属性ファイルによるプロパティの設定」を参照してください。次にリソースアダプタのプロパティ定義について説明します。

(1) 編集する属性ファイル

Connector 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して，リソースアダプタの Connector 属性ファイルを取得します。

実行形式

```
cjgetresprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname
<リソースアダプタの表示名> -c <Connector属性ファイルパス>
```

実行例

```
cjgetresprop MyServer -type rar -resname Rar1 -c AccountProp.xml
```

● 属性の設定

次に示すコマンドを実行して，Connector 属性ファイルの値を反映します。

実行形式

```
cjsetresprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname
<リソースアダプタ表示名> -c <Connector属性ファイルパス>
```

実行例

```
cjsetresprop MyServer -type rar -resname Rar1 -c AccountProp.xml
```

! 注意事項

リソースアダプタをデプロイしたあとでプロパティを定義する場合は，cjgetrarprop コマンドと cjsetrarprop コマンドを使用してください。cjgetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetrarprop（RAR ファイルの属性の取得）」を参照してください。cjsetrarprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetrarprop（RAR 属性設定）」を参照してください。

(3) 編集する属性設定項目

Connector 1.5 の仕様に準拠するリソースアダプタは，コネクション定義を複数持つ場合があります。

プロパティ項目は，次のように分けて設定します。

● リソースアダプタのプロパティ設定

次のプロパティを設定します。

- コンフィグレーションプロパティ
- Outbound リソースアダプタのプロパティ
- Inbound リソースアダプタのプロパティ
- 管理対象オブジェクトのプロパティ
- 実行時プロパティ

プロパティ設定項目については，「(4) リソースアダプタのプロパティ設定」を参照してください。

● Outbound リソースアダプタのコネクション単位のプロパティ設定

Outbound リソースアダプタを使用する場合は，次のプロパティを設定します。

- コンフィグレーションプロパティ
- 実行時プロパティ
- 別名情報

Outbound リソースアダプタのコネクション単位のプロパティ設定項目については，「(5) Outbound リソースアダプタのコネクション単位のプロパティ設定」を参照してください。

(4) リソースアダプタのプロパティ設定

リソースアダプタのプロパティ設定項目を次に示します。

(a) コンフィグレーションプロパティ

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

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

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

定義するコンフィグレーションプロパティ名（<config-property-name>）については，「4.2.2 DB Connector のプロパティ定義」を参照してください。

(b) Outbound リソースアダプタのプロパティ

Outbound リソースアダプタのプロパティ（<outbound-resourceadapter> タグ）の設定項目を次に示します。

項目	必須	対応するタグ
コネクション単位でのプロパティ設定※1	△	<connection-definition>
トランザクションサポートのレベル※2	○	<transaction-support>
再認証のサポート有無	○	<reauthentication-support>

（凡例） ○：必須 △：任意

注※1 Outbound リソースアダプタのコネクション単位のプロパティ設定については，「(5) Outbound リソースアダプタのコネクション単位のプロパティ設定」を参照してください。

注※2 Connector 1.5 の仕様に準拠するリソースアダプタの場合，グローバルトランザクション（XATransaction）は指定できません。

プロパティの設定項目については，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「4.1.1 Connector 属性ファイルの指定内容」を参照してください。

(c) Inbound リソースアダプタのプロパティ

Inbound リソースアダプタのプロパティ（<inbound-resourceadapter> - <messageadapter> - <messagelistener> タグ）の設定項目を次に示します。

項目	対応するタグ
メッセージリスナのタイプ	<messagelistener-type>
ActivationSpec の情報	<activationspec>

(d) 管理対象オブジェクトのプロパティ

リソースアダプタの管理対象オブジェクトのプロパティ（<adminobject> - <config-property> タグ）の設定項目を次に示します。

項目	対応するタグ
管理対象オブジェクトのプロパティ名	<config-property-name>
管理対象オブジェクトのプロパティのデータ型	<config-property-type>
管理対象オブジェクトのプロパティの値	<config-property-value>

(e) 実行時プロパティ

リソースアダプタの実行時プロパティ（<resourceadapter-runtime> - <property> タグ）の設定項目を次に示します。

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

注※ プロパティの値の設定方法については，「4.2.2 DB Connector のプロパティ定義」を参照してください。

定義するプロパティの数だけ，上記の設定を繰り返してください。

プロパティ名（<property-name>）には，次の項目を設定します。

プロパティ項目	プロパティ名（<property-name>）の項目名
スレッドプールで同時に実行される最大スレッド数	MaxTPoolSize
スレッドプールに存在する最小スレッド数	MinTPoolSize
スレッドプールのスレッド解放までのタイムアウト値（秒）	TPoolKeepalive

注　ライフサイクル管理機能が有効な場合に設定します。ライフサイクル管理機能を使用していない（<resourceadapter-class> を指定していない）場合，プロパティ値（<property-value>）は無視されます。ライフサイクル管理機能については，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「3.16.1　リソースアダプタのライフサイクル管理」を参照してください。

プロパティの設定項目については，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「4.1.1　Connector 属性ファイルの指定内容」を参照してください。

（5） Outbound リソースアダプタのコネクション単位のプロパティ設定

Outbound リソースアダプタのコネクション単位のプロパティ設定項目を次に示します。

（a） コンフィグレーションプロパティ

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

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

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

定義するコンフィグレーションプロパティ名（<config-property-name>）については，「4.2.2　DB Connector のプロパティ定義」を参照してください。

なお，コネクション単位のコンフィグレーションプロパティには，コネクション単位に設定されたコンフィグレーションプロパティ（<outbound-resourceadapter> - <connection-definition> - <config-property> タグ）とリソースアダプタに設定されたコンフィグレーションプロパティ（<resourceadapter> - <config-property> タグ）を合わせたものが設定されます。同じコンフィグレーションプロパティ名が設定されている場合は，コネクション単位に設定されたコンフィグレーションプロパティが優先されます。

(b) 実行時プロパティ

Outbound リソースアダプタの実行時プロパティ（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> タグ）の設定項目を次に示します。

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

定義するプロパティの数だけ，上記の設定を繰り返してください。

定義する実行時プロパティ名（<property-name>）およびコネクションプールの動作と注意事項については，「4.2.2　DB Connector のプロパティ定義」を参照してください。

(c) 別名情報

Outbound リソースアダプタの別名情報（<outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <resource-external-property> タグ）の設定項目を次に示します。

項目	必須	対応するタグ
リソースの別名	○	<optional-name>
リソース認証方式	△	<res-auth>
リソース共有の有無	△	<res-sharing-scope>

（凡例）　○：必須　△：任意

プロパティの設定項目については，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「4.1.1　Connector 属性ファイルの指定内容」を参照してください。

リソースアダプタの別名の使用方法については，「4.8　JNDI 名前空間に登録されるリソースアダプタ名の参照と変更」を参照してください。

4.4.4　リソースアダプタのデプロイ

リソースアダプタは，デプロイすると J2EE リソースアダプタとして使用できます。

サーバ管理コマンドでインポートしたリソースアダプタをデプロイすると，その J2EE サーバ上で動作するすべての J2EE アプリケーションから使用できるようになります。なお，リソースアダプタをデプロイしたあとで，リソースアダプタのプロパティを定義することもできます。プロパティを定義する方法については，リソースアダプタが対応しているコネクタアーキテクチャの仕様に応じて，「4.4.2　リソースアダプタのプロパ

ティ定義（Connector 1.0 の場合）」または「4.4.3　リソースアダプタのプロパティ定義（Connector 1.5 の場合）」を参照してください。

次に示すコマンドを実行してリソースアダプタをデプロイします。

実行形式

```
cjdeployrar ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -resname ＜リソースアダプタの表示名＞ ［-resname ＜リソースアダプタの表示名＞］
```

実行例

```
cjdeployrar MyServer -resname Rar1
```

cjdeployrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeployrar（リソースアダプタのデプロイ）」を参照してください。

注意事項

開始状態の実行時情報を含んだリソースアダプタを指定すると，デプロイ完了後に自動開始されます。自動開始処理に失敗した場合でも，この J2EE リソースアダプタは削除されません。

4.4.5　J2EE リソースアダプタの接続テスト

リソースアダプタに設定した情報が正しいかどうか，接続テストによって検証します。

次に示すコマンドを実行してリソースアダプタの接続テストを実施します。

実行形式

```
cjtestres ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -type rar -resname ＜リソースアダプタの表示名＞ ［-resname ＜リソースアダプタの表示名＞］
```

実行例

```
cjtestres MyServer -type rar -resname Rar1
```

cjtestres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjtestres（リソースの接続テスト）」を参照してください。

注意事項

- 一度接続テストをしたリソースアダプタは，J2EE サーバを再起動するまで削除できません。リソースアダプタを削除する場合は，リソースアダプタを停止してから，J2EE サーバを再起動してください。

- Connector 1.5 の仕様に準拠するリソースアダプタにコネクション定義が複数ある場合，すべてのコネクション定義に対して接続テストが実行されます。コネクション定義のどれかでエラーが発生した場合でも，すべてのコネクション定義に対して接続テストが実行されます。ただし，その場合，接続テスト結果は「失敗」となります。
- Connector 1.5 の仕様に準拠するリソースアダプタの場合は，Outbound リソースアダプタを含むときだけ，接続テストを実施できます。

4.4.6 J2EE リソースアダプタの開始

J2EE リソースアダプタを開始します。

次に示すコマンドを実行して J2EE リソースアダプタを開始します。

実行形式

```
cjstartrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <リソースアダプタの表示名>
```

実行例

```
cjstartrar MyServer -resname Rar1
```

cjstartrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstartrar（リソースアダプタの開始)」を参照してください。

注意事項

- J2EE アプリケーション中の J2EE リソースが J2EE リソースアダプタを参照している場合は，J2EE リソースアダプタを開始してから，J2EE アプリケーションを開始してください。
- 一度開始した J2EE リソースアダプタは，J2EE サーバを再起動するまで削除できません。J2EE リソースアダプタを削除するには，J2EE リソースアダプタを停止してから，J2EE サーバを再起動し，削除しようとする J2EE リソースアダプタが J2EE アプリケーションの参照解決の対象外であることを確認してください。

4.4.7 J2EE リソースアダプタの停止

J2EE リソースアダプタを停止します。

次に示すコマンドを実行して J2EE リソースアダプタを停止します。

実行形式

```
cjstoprar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <リソースアダプタの表示名>
```

実行例

```
cjstoprar MyServer -resname Rar1
```

cjstoprar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstoprar（リソースアダプタの停止）」を参照してください。

注意事項

J2EE アプリケーション中の J2EE リソースが J2EE リソースアダプタを参照している場合は，J2EE アプリケーションを停止してから，J2EE リソースアダプタを停止してください。

4.4.8 J2EE リソースアダプタのアンデプロイ

デプロイされている J2EE リソースアダプタを削除します。

準備

J2EE リソースアダプタを削除する前に，J2EE リソースアダプタを停止して J2EE サーバを再起動してください。また，J2EE リソースアダプタに対し，一度でも開始または接続テストを試みた場合も同様に J2EE サーバを再起動してください。

次に示すコマンドを実行して J2EE リソースアダプタをアンデプロイします。

実行形式

```
cjundeployrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <リソースアダプタの表示名>
```

実行例

```
cjundeployrar MyServer -resname Rar1
```

cjundeployrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjundeployrar（リソースアダプタのアンデプロイ）」を参照してください。

4.4.9 J2EE リソースアダプタのエクスポート

J2EE リソースアダプタの内容を RAR ファイルとして出力（エクスポート）します。

次に示すコマンドを実行して J2EE リソースアダプタをエクスポートします。

実行形式

```
cjexportrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -f <ファイルパス>
-resname <J2EEリソースアダプタの表示名>
```

実行例

```
cjexportrar MyServer -f res1.rar -resname Rar1
```

cjexportrar コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjexportrar（リソースアダプタのエクスポート）」を参照してください。

注意事項

Management Server を利用して運用している場合，サーバ管理コマンドと Management Server の実行ホストが異なる場合，J2EE リソースアダプタをエクスポートして，Management Server の実行ホストに格納する必要があります。

4.5 J2EE リソースアダプタの状態と一覧の参照

デプロイされているリソースアダプタのアダプタ名と状態を参照できます。また，Connector 1.5 の仕様に準拠するリソースアダプタの場合，コネクション定義識別子の一覧や，メッセージリスナのタイプの一覧などの情報も参照できます。

4.5.1 J2EE リソースアダプタの状態の参照

次に示すコマンドを実行して，デプロイされているすべてのリソースアダプタの名称と状態（開始状態または停止状態）を参照します。

実行形式

```
cjlistrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-spec］
```

実行例

```
cjlistrar MyServer
```

-spec オプションを指定すると，リソースアダプタの状態表示一覧にコネクタアーキテクチャの仕様バージョンが表示されます。

cjlistrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistrar（リソースアダプタの一覧表示）」を参照してください。

4.5.2 コネクション定義識別子の一覧の参照（Outbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，デプロイされている Outbound リソースアダプタのコネクション定義識別子の一覧を参照できます。

実行形式

```
cjlistrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <RARの表示名
> -outbound
```

実行例

```
cjlistrar MyServer -resname account-rar -outbound
```

cjlistrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistrar（リソースアダプタの一覧表示）」を参照してください。

4.5.3 メッセージリスナのタイプの一覧の参照（Inbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，デプロイされている Inbound リソースアダプタのメッセージリスナのタイプの一覧を参照できます。

実行形式

```
cjlistrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <RARの表示名
> -inbound
```

実行例

```
cjlistrar MyServer -resname account-rar -inbound
```

cjlistrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistrar（リソースアダプタの一覧表示）」を参照してください。

4.5.4 メッセージリスナのアクティブ化に必要なプロパティ名の一覧の参照（Inbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，デプロイされている Inbound リソースアダプタのメッセージリスナのアクティブ化に必要なプロパティ名の一覧を参照できます。

実行形式

```
cjlistrar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <RARの表示名
> -listenertype <メッセージリスナのタイプ名>
```

実行例

```
cjlistrar MyServer -resname account-rar -listenertype
javax.jms.MessageListener
```

cjlistrar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistrar（リソースアダプタの一覧表示）」を参照してください。

4.6 リソースアダプタの一覧の参照

インポートされているリソースアダプタの一覧を参照できます。また，Connector 1.5 の仕様に準拠するリソースアダプタの場合，コネクション定義識別子の一覧や，メッセージリスナのタイプの一覧などの情報も参照できます。

4.6.1 リソースアダプタの一覧の参照

次に示すコマンドを実行して，インポートされているリソースアダプタの一覧を参照します。

実行形式

```
cjlistres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar ［-spec］
```

実行例

```
cjlistres MyServer -type rar
```

-spec オプションを指定すると，一覧にコネクタアーキテクチャの仕様バージョンが表示されます。

cjlistres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistres（リソースの一覧表示）」を参照してください。

4.6.2 コネクション定義識別子の一覧の参照（Outbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，J2EE リソースアダプタとしてデプロイする Outbound リソースアダプタのコネクション定義識別子の一覧を参照します。

実行形式

```
cjlistres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname <
RARの表示名> -outbound
```

実行例

```
cjlistres MyServer -type rar -resname account-rar -outbound
```

cjlistres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistres（リソースの一覧表示）」を参照してください。

4.6.3 メッセージリスナのタイプの一覧の参照（Inbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，J2EE リソースアダプタとしてデプロイする Inbound リソースアダプタのメッセージリスナのタイプの一覧を参照できます。

実行形式

```
cjlistres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname <
RARの表示名> -inbound
```

実行例

```
cjlistres MyServer -type rar -resname account-rar -inbound
```

cjlistres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistres（リソースの一覧表示）」を参照してください。

4.6.4 メッセージリスナのアクティブ化に必要なプロパティ名の一覧の参照（Inbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，J2EE リソースアダプタとしてデプロイする Inbound リソースアダプタのメッセージリスナのアクティブ化に必要なプロパティ名の一覧を参照できます。

実行形式

```
cjlistres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname <
RARの表示名> -listenertype <メッセージリスナのタイプ名>
```

実行例

```
cjlistres MyServer -type rar -resname account-rar -listenertype
javax.jms.MessageListener
```

cjlistres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistres（リソースの一覧表示）」を参照してください。

4.7 コネクションプールの状態の確認

リソースアダプタのコネクションプールの情報およびコネクションの状態（使用中，未使用）を参照します。また，必要に応じて，リソースアダプタのコネクションを削除します。

4.7.1 コネクションプールの状態表示

次に示すコマンドを使用して，リソースアダプタのコネクションプールの情報および状態を表示します。

実行形式

```
cjlistpool ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <リソースアダプタの表示名>
```

実行例

```
cjlistpool MyServer -resname Rar1
```

cjlistpool コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistpool（コネクションプールの一覧表示）」を参照してください。

参考

Connector 1.5 の仕様に準拠するリソースアダプタの場合，-resname オプションの＜リソースアダプタの表示名＞は次の形式で指定します。
< リソースアダプタの表示名 >!< コネクション定義識別子 >

4.7.2 コネクションプールの削除

次に示すコマンドを使用して，リソースアダプタのコネクションプールを削除します。未使用のコネクションプールはすべて削除されます。

実行形式

```
cjclearpool ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type connector -resname <リソースアダプタの表示名>
```

実行例

```
cjclearpool MyServer -type connector -resname Rar1
```

cjclearpool コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjclearpool（コネクションプール内のコネクション削除）」を参照してください。

参考

Connector 1.5 の仕様に準拠するリソースアダプタの場合，-resname オプションの＜リソースアダプタの表示名＞は次の形式で指定します。
< リソースアダプタの表示名 >!< コネクション定義識別子 >

4.8 JNDI 名前空間に登録されるリソースアダプタ名の参照と変更

JNDI 名前空間に登録されるリソースアダプタ名を変更します。

リソースアダプタ名には別名も付けられます。別名を付けることで，JNDI 名前空間から任意の名前でリソースアダプタを参照できるようになります。なお，この機能を**ユーザ指定名前空間機能**といいます。

リソースアダプタの JNDI 名前空間については，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。ユーザ指定名前空間機能については，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「2.4　Enterprise Bean または J2EE リソースへの別名付与（ユーザ指定名前空間機能）」を参照してください。

別名は，Connector 属性ファイルの <resource-external-property> タグで追加できます。ただし，JMS インタフェースを使用するリソースアダプタには，別名は付与できません。

Connector 属性ファイルの <resource-external-property> タグの編集については，使用するリソースアダプタの種類に応じて，「4.2.2　DB Connector のプロパティ定義」，「4.4.2　リソースアダプタのプロパティ定義（Connector 1.0 の場合）」または「4.4.3　リソースアダプタのプロパティ定義（Connector 1.5 の場合）」の別名情報の設定を参照してください。

注意事項

開始状態の J2EE アプリケーションがある場合，別名を設定しているリソースアダプタの停止および削除はできません。J2EE サーバで開始されているすべての J2EE アプリケーションを停止させてください。

4.9 リソースアダプタの削除

次に示すコマンドを実行してリソースアダプタを削除します。

実行形式

```
cjdeleteres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -resname
<リソースアダプタの表示名>
```

実行例

```
cjdeleteres MyServer -type rar -resname account-rar
```

cjdeleteres コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjdeleteres（リソースの削除)」を参照してください。

注意事項

cosminexus.xml を含むアプリケーションから J2EE リソースを削除した場合，cosminexus.xml に含まれている削除対象の J2EE リソースに関する Cosminexus 独自の定義情報は削除されません。cosminexus.xml を含むアプリケーションから J2EE リソースを削除する場合の詳細は，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「10.3.6 cosminexus.xml を含むアプリケーションの運用」を参照してください。

4.10 リソースアダプタのコピー

次に示すコマンドを実行してリソースアダプタのプロパティをコピーします。

実行形式

```
cjcopyres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -src <コピー元のリソースアダプタの表示名> -dst <コピー先のリソースアダプタの表示名>
```

実行例

```
cjcopyres MyServer -type rar -src account-rar -dst account-rar2
```

cjcopyres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjcopyres（リソースのコピー）」を参照してください。

5 J2EE アプリケーションに含まれるリソースアダプタの設定

この章では，J2EE アプリケーションに含まれるリソースアダプタのアプリケーション設定操作について説明します。

5.1　J2EE アプリケーションに含まれるリソースアダプタの設定の概要

J2EE アプリケーションに含まれるリソースアダプタの設定とは，J2EE アプリケーションに含まれるリソースアダプタを，その J2EE アプリケーションで利用できる状態にするための操作です。

J2EE アプリケーションに含めて使用できるリソースアダプタの種類と，アプリケーション設定操作の概要について説明します。

5.1.1　利用できるリソースアダプタ

J2EE アプリケーションに含めて利用できるリソースアダプタを次に示します。

- DB Connector
- uCosminexus TP1 Connector
- そのほかの Connector 1.0 に準拠したリソースアダプタ，または Connector 1.5 に準拠したリソースアダプタ※

注※　そのほかの Connector 1.0 に準拠したリソースアダプタ，または Connector 1.5 に準拠したリソースアダプタについては，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「3.3.2　リソースアダプタの種類」を参照してください。

注意事項

J2EE リソースアダプタと J2EE アプリケーションに含まれるリソースアダプタは，J2EE アプリケーションから同時に使用できます。ただし，J2EE リソースアダプタと J2EE アプリケーションに含まれるリソースアダプタを，同じ表示名で同時に使用することはできません。同じ表示名に設定するとエラーとなります。

5.1.2　設定する項目と操作の概要

J2EE アプリケーションに含まれるリソースアダプタの設定の概要について，次の表に示します。

表 5-1 J2EE アプリケーションに含まれるリソースアダプタの設定の概要

設定項目		内容	参照先
リソースアダプタを含んだ J2EE アプリケーションの作成※	J2EE アプリケーションへのリソースアダプタの追加	RAR ファイルを J2EE サーバにインポートして，J2EE アプリケーションに追加します。	5.2
	リソースアダプタを含む J2EE アプリケーションのインポート	アプリケーション開発環境で作成した，RAR ファイルを含んだ J2EE アプリケーションを J2EE サーバにインポートします。	5.3
リソースアダプタのプロパティ定義		データベースとの接続についての情報を設定します。 • リソースアダプタの一般情報 • コンフィグレーションプロパティ • 実行時プロパティ • 別名情報	5.4
リソースアダプタの接続テスト		リソースアダプタに設定した内容が正しいかどうかを検証します。	5.5
リソースアダプタを含む J2EE アプリケーションの開始と停止		リソースアダプタを含む J2EE アプリケーションを開始または停止します。	5.6
J2EE アプリケーションに含まれるリソースアダプタの一覧の参照		J2EE アプリケーションに含まれるリソースアダプタの一覧，コネクション定義識別子の一覧などを参照します。	5.7
コネクションプールの一覧表示と削除		J2EE アプリケーションに含まれるリソースアダプタのコネクションプールの状態を参照します。また，必要に応じてコネクションを削除します。	5.8

注※ リソースアダプタを含んだ J2EE アプリケーションは次のどちらかの方法で作成します。

- J2EE アプリケーションにリソースアダプタを追加します。
  EJB-JAR または WAR を J2EE アプリケーションに追加する場合と同じ方法で，RAR を J2EE アプリケーションに追加します。
- リソースアダプタを含む J2EE アプリケーションをインポートします。
  アプリケーション開発環境でリソースアダプタを含む J2EE アプリケーションを成して，その J2EE アプリケーションを J2EE サーバにインポートします。

5.2 J2EE アプリケーションへのリソースアダプタの追加

J2EE アプリケーションにリソースアダプタを追加します。

追加を実行する前に，追加するリソースアダプタと同じ表示名，または同じ別名のリソースアダプタが，デプロイされていないことを確認してください。同じ表示名，または同じ別名のリソースアダプタがデプロイされている場合，そのリソースアダプタは J2EE アプリケーションに追加できません。なお，別の J2EE アプリケーションに含まれるリソースアダプタと同じ表示名，別名は使用できます。

手順を次に示します。

1. **リソースアダプタの** RAR **ファイルを** J2EE **サーバにインポートします。**
   J2EE アプリケーションに追加する RAR ファイルのインポートについては，「7.4　リソースアダプタ（RAR）のインポート」を参照してください。
2. J2EE **アプリケーションにインポートした** RAR **を追加します。**
   J2EE アプリケーションへのリソースアダプタ（RAR）の追加については，「7.5 J2EE アプリケーションの新規作成」の「(3)　RAR ファイルの追加」を参照してください。

5.3　リソースアダプタを含む J2EE アプリケーションのインポート

リソースアダプタを含む J2EE アプリケーションをインポートします。

インポートを実行する前に，J2EE アプリケーションに含まれるリソースアダプタと同じ表示名，または同じ別名のリソースアダプタが，デプロイされていないことを確認してください。同じ表示名，または同じ別名のリソースアダプタがデプロイされている場合，そのリソースアダプタを含む J2EE アプリケーションはインポートできません。なお，別の J2EE アプリケーションに含まれるリソースアダプタと同じ表示名，別名は使用できます。

次の J2EE アプリケーションをインポートします。

- アーカイブ形式の J2EE アプリケーション
  J2EE サーバの作業ディレクトリ以下に，EJB-JAR/WAR/RAR ファイルのコンポーネントを持つ J2EE アプリケーションです。アーカイブ形式の J2EE アプリケーションのインポートについては，「8.1.1　アーカイブ形式の J2EE アプリケーションのインポート」を参照してください。
- 展開ディレクトリ形式の J2EE アプリケーション
  EJB-JAR や WAR などのアプリケーションの実体を，J2EE サーバの外部にある一定のルールに従ったファイル／ディレクトリに持つ J2EE アプリケーションです。ただし，展開ディレクトリ形式の J2EE アプリケーションに含まれる RAR ファイルは展開ファイル形式ではなく，アーカイブ形式で格納されます。展開ディレクトリ形式の J2EE アプリケーションのインポートについては，「8.1.2　展開ディレクトリ形式の J2EE アプリケーションのインポート」を参照してください。

5.4 リソースアダプタのプロパティ定義

J2EE アプリケーションに含まれるリソースアダプタのプロパティを設定します。

リソースアダプタのプロパティの設定は，リソースアダプタを含む J2EE アプリケーションを停止した状態で実行してください。

プロパティの設定手順については，「3.3 属性ファイルによるプロパティの設定」を参照してください。

(1) 編集する属性ファイル

次のどちらかの属性ファイルを利用して，J2EE アプリケーションに含まれるリソースアダプタのプロパティを設定できます。

- Connector 属性ファイルを利用します。
- アプリケーション統合属性ファイルを利用します。
  J2EE アプリケーション統合属性ファイルを構成するコンポーネントの属性ファイルのうち，Connector 属性の要素でプロパティを設定します。

J2EE アプリケーションを構成するコンポーネントのプロパティの設定では，Connector 属性ファイルとアプリケーション統合属性ファイルを個々に利用することも，あわせて利用することもできます。

この章では，Connector 属性ファイルを使用してのプロパティ設定について説明しています。アプリケーション統合属性ファイルを使用してのプロパティ設定の手順については，「9.2 アプリケーション統合属性ファイルによるプロパティ設定」を参照してください。

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してリソースアダプタの属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type rar -resname <リソースアダプタ表示名> -c <Connctor属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name App1 -type rar -resname account-rar
-c AccountProp.xml
```

● **属性の設定**

次に示すコマンドを実行して，リソースアダプタの属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type rar -resname <リソースアダプタ表示名> -c <Connector属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name App1 -type rar -resname account-rar
-c AccountProp.xml
```

(3) 編集する属性設定項目

リソースアダプタの種類に応じて，プロパティを定義します。

- Connector 1.0 の仕様に準拠するリソースアダプタの場合
  Connector 1.0 の仕様に準拠するリソースアダプタの属性設定項目については，「4.4.2 リソースアダプタのプロパティ定義（Connector 1.0 の場合）」の「(3) 編集する属性設定項目」を参照してください。
- Connector 1.5 の仕様に準拠するリソースアダプタの場合
  Connector 1.5 の仕様に準拠するリソースアダプタの属性設定項目については，「4.4.3 リソースアダプタのプロパティ定義（Connector 1.5 の場合）」の「(3) 編集する属性設定項目」を参照してください。

5.5 リソースアダプタの接続テスト

J2EE アプリケーションに含まれるリソースアダプタに設定した情報が正しいかどうか，接続テストによって検証します。

次に示すコマンドを実行して J2EE アプリケーションに含まれるリソースアダプタの接続テストを実行します。

実行形式

```
cjtestres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-test］ -name <J2EE
アプリケーション名> -type rar -resname <リソースアダプタの表示名>
```

実行例

```
cjtestres -name App1 -type rar -resname
DB_Connector_for_Cosminexus_Driver
```

J2EE アプリケーションをテストモードにして，リソースアダプタの接続テストを実行する場合は，-test オプションを指定します。

J2EE アプリケーションに含まれるリソースアダプタを接続テストする場合の注意事項については，次の説明を参照してください。

- データベースと接続するための設定（DB Connector を使用する場合）
  「4.2.4　DB Connector の接続テスト」
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）
  「4.4.5　J2EE リソースアダプタの接続テスト」

cjtestres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjtestres（リソースの接続テスト）」を参照してください。

5.6　リソースアダプタを含む J2EE アプリケーションの開始と停止

J2EE アプリケーションを開始すると，J2EE アプリケーションに含まれるすべてのリソースアダプタは開始されます。また，J2EE アプリケーションを停止すると，J2EE アプリケーションに含まれるすべてのリソースアダプタは停止されます。リソースアダプタの開始順序および停止順序の制御はできません。

J2EE アプリケーションの開始については，「10.2.1　J2EE アプリケーションの開始」を参照してください。

J2EE アプリケーションの停止については，「10.2.2　J2EE アプリケーションの停止」を参照してください。

注意事項

J2EE アプリケーションがリソースアダプタを含み，application.xml の <module> タグ以下で RAR を記述している場合，EJB-JAR や WAR が，リソースアダプタを参照していなくても，J2EE アプリケーション開始時にリソースアダプタは開始されます。

5.7 J2EE アプリケーションに含まれるリソースアダプタの一覧の参照

J2EE アプリケーションに含まれるリソースアダプタの一覧を参照できます。また，Connector 1.5 の仕様に準拠するリソースアダプタの場合，コネクション定義識別子の一覧や，メッセージリスナのタイプの一覧などの情報も参照できます。

5.7.1 リソースアダプタの一覧の参照

次に示すコマンドを実行して，J2EE アプリケーションに含まれるリソースアダプタの一覧を参照します。

実行形式

```
cjlistapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-test］ -name
<J2EEアプリケーション名> -type rar ［-spec］
```

実行例

```
cjlistapp MyServer -name App1 -type rar
```

テストモードの J2EE アプリケーションに対してコマンドを実行する場合，-test オプションを指定します。

-spec オプションを指定すると，一覧にコネクタアーキテクチャの仕様バージョンが表示されます。

cjlistapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistapp（アプリケーションの一覧表示）」を参照してください。

5.7.2 コネクション定義識別子の一覧の参照（Outbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，J2EE アプリケーションに含まれる Outbound リソースアダプタのコネクション定義識別子の一覧を参照します。

実行形式

```
cjlistapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-test］ -name <J2EE
アプリケーション名> -type rar -resname <リソースアダプタの表示名> -outbound
```

実行例

```
cjlistapp MyServer -name App1 -type rar -resname account-rar
-outbound
```

テストモードの J2EE アプリケーションに対してコマンドを実行する場合，-test オプションを指定します。

cjlistapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistapp（アプリケーションの一覧表示）」を参照してください。

5.7.3 メッセージリスナのタイプの一覧の参照（Inbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，J2EE アプリケーションに含まれる Inbound リソースアダプタのメッセージリスナのタイプの一覧を参照します。

実行形式

```
cjlistapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-test］ -name <J2EE
アプリケーション名> -type rar -resname <リソースアダプタの表示名> -inbound
```

実行例

```
cjlistapp MyServer -name App1 -type rar -resname account-rar
-inbound
```

テストモードの J2EE アプリケーションに対してコマンドを実行する場合，-test オプションを指定します。

cjlistapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistapp（アプリケーションの一覧表示）」を参照してください。

5.7.4 メッセージリスナのアクティブ化に必要なプロパティ名の一覧の参照（Inbound リソースアダプタ）

Connector 1.5 の仕様に準拠するリソースアダプタの場合，次に示すコマンドを実行して，J2EE アプリケーションに含まれる Inbound リソースアダプタのメッセージリスナのアクティブ化に必要なプロパティ名の一覧を参照します。

実行形式

```
cjlistapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-test］ -name <J2EE
アプリケーション名> -type rar -resname <リソースアダプタの表示名> -listenertype <
メッセージリスナのタイプ名>
```

実行例

```
cjlistapp MyServer -name App1 -type rar -resname account-rar
-listenertype javax.jms.MessageListener
```

テストモードの J2EE アプリケーションに対してコマンドを実行する場合，-test オプションを指定します。

cjlistapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistapp（アプリケーションの一覧表示）」を参照してください。

5.8 コネクションプールの一覧表示と削除

J2EE アプリケーションに含まれるリソースアダプタの，コネクションプールの情報およびコネクションの状態（使用中，未使用）を参照します。また，必要に応じて，リソースアダプタのコネクションを削除します。

5.8.1 コネクションプールの状態表示

次に示すコマンドを使用して，J2EE アプリケーションに含まれるリソースアダプタの，コネクションプールの情報および状態を表示します。

実行形式

```
cjlistpool ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ ［-test］
-name ＜J2EEアプリケーション名> -resname ＜リソースアダプタの表示名>
```

実行例

```
cjlistpool MyServer -name App1 -resname Rar1
```

テストモードの J2EE アプリケーションに対してコマンドを実行する場合，-test オプションを指定します。

cjlistpool コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistpool（コネクションプールの一覧表示)」を参照してください。

参考

Connector 1.5 の仕様に準拠するリソースアダプタの場合，-resname オプションの＜リソースアダプタの表示名＞は次の形式で指定します。
< リソースアダプタの表示名 >!< コネクション定義識別子 >

5.8.2 コネクションプールの削除

次に示すコマンドを使用して，必要に応じて，J2EE アプリケーションに含まれるリソースアダプタのコネクションプールを削除します。なお，未使用のコネクションプールはすべて削除されます。

実行形式

```
cjclearpool ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -type connector
［-test］ -name ＜J2EEアプリケーション名> -resname ＜リソースアダプタの表示名>
```

実行例

```
cjclearpool MyServer -type connector -name App1 -resname Rar1
```

cjclearpool コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjclearpool（コネクションプール内のコネクション削除）」を参照してください。

参考

Connector 1.5 の仕様に準拠するリソースアダプタの場合，-resname オプションの＜リソースアダプタの表示名＞は次の形式で指定します。
< リソースアダプタの表示名 >!< コネクション定義識別子 >

6 リソースアダプタ以外の J2EE リソースの設定

この章では，サーバ管理コマンドを使用した，リソースアダプタ以外の J2EE リソースの設定方法について説明します。

6.1 リソースアダプタ以外の J2EE リソースの設定概要

次の J2EE リソースについて，設定作業の概要を示します。

- JavaBeans リソースの設定
- メールコンフィグレーションの設定
- J2EE リソース共通の設定

6.1.1 JavaBeans リソースの設定の概要

JavaBeans リソースを利用する場合に必要な設定です。

表 6-1 JavaBeans リソースの設定の概要

設定項目	内容	参照先
JavaBeans リソースのインポート	JavaBeans リソースをインポートして，JavaBeans リソース属性の値を設定します。	6.2.1
JavaBeans リソースのプロパティの定義	JavaBeans リソースの実行に必要な情報を設定します。	6.2.2
JavaBeans リソースの開始	JavaBeans リソースを開始します。	6.2.3
JavaBeans リソースの停止	JavaBeans リソースを停止します。	6.2.4
JavaBeans リソースの削除	JavaBeans リソースを削除します。JavaBeans リソースを入れ替える場合は，必要に応じて実行してください。	6.2.5
JavaBeans リソースの状態表示	インポートされているすべての JavaBeans リソースの状態を表示します。	6.2.6

6.1.2 メールコンフィグレーションの設定の概要

JavaMail を使用して SMTP サーバに接続する場合に必要な設定です。メールサーバおよびメールのヘッダに入れる情報を設定します。

表 6-2 メールコンフィグレーションの設定の概要

設定項目	内容	参照先
メールコンフィグレーションの新規作成	メールコンフィグレーションを新規作成します。	6.3.1
メールコンフィグレーションのプロパティ定義	メールコンフィグレーションの次の情報を設定します。 • SMTP メールサーバのホスト名または IP アドレス • メールのヘッダに入れる情報	6.3.2

設定項目	内容	参照先
メールコンフィグレーションの接続テスト	メールコンフィグレーションに設定した内容が正しいかどうかを検証します。	6.3.3
メールコンフィグレーションの削除	作成したメールコンフィグレーションを削除します。	6.3.4
メールコンフィグレーションのコピー	メールコンフィグレーションのプロパティをコピーします。	6.3.5

6.1.3 J2EE リソース共通の設定の概要

J2EE リソースに共通の設定です。

表 6-3 J2EE リソースに共通な設定の概要

設定項目	内容	参照先
J2EE リソース一覧の参照	次の J2EE リソースについて一覧を参照します。 • EJB-JAR ファイルに含まれる J2EE リソース • WAR ファイルに含まれる J2EE リソース • メールコンフィグレーション	6.4
JNDI 名前空間に登録される J2EE リソース名の参照と変更	J2EE リソースの JNDI 名前空間への登録名を参照して，必要に応じて別名を付与します。	6.5

6.2 JavaBeans リソースの設定

JavaBeans リソースの設定とは JavaBeans リソースを利用できる状態にすることです。

JavaBeans リソースの設定をするには，あらかじめ，JavaBeans リソース属性ファイルのテンプレートファイルを参考に，JavaBeans リソース属性ファイルを編集しておいてください。JavaBeans リソース属性ファイルのテンプレートファイル（jb_template.xml）は，次のディレクトリに格納されています。

Windows の場合

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

UNIX の場合

/opt/Cosminexus/CC/admin/templates/

JavaBeans リソースは，次の手順で設定します。

JavaBeans リソースを新規に設定する場合

1. JavaBeans リソースをインポートします。
2. JavaBeans リソースを開始します。

JavaBeans リソースのプロパティ属性を変更する場合

1. JavaBeans リソースを停止します。
2. JavaBeans リソースのプロパティ定義を変更します。
3. JavaBeans リソースを開始します。

JavaBeans リソースを入れ替える場合

1. JavaBeans リソースを停止します。
2. J2EE サーバを再起動します。
3. JavaBeans リソースを削除します。
4. JavaBeans リソースをインポートします。
5. JavaBeans リソースを開始します。

6.2.1 JavaBeans リソースのインポート

次に示すコマンドを実行して JavaBeans リソースをインポートします。

実行形式

```
cjimportjb ［＜サーバ名称＞］ ［-nameserver ＜プロバイダURL＞］ -f ＜JARファイルパス＞
-c ＜JavaBeansリソース属性ファイルパス＞
```

実行例

```
cjimportjb MyServer -f Myjavabeans.jar -c Myjavabeansprop.xml
```

JavaBeans リソースは，cjimportjb コマンドの -d オプションで，ディレクトリパスを指定すると，次のようにインポートすることもできます。

- アーカイブの作成をしないで，ディレクトリ構成でインポートします。
- 指定したディレクトリ下のすべてのアーカイブされた JavaBeans リソースをインポートします。

コマンドの指定方法を次に示します。

実行形式

```
cjimportjb ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -d <ディレクトリパス>
-c <JavaBeansリソース属性ファイルパス>
```

実行例

```
cjimportjb MyServer -d MydirectoryPath -c Myjavabeansprop.xml
```

cjimportjb コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportjb（JavaBeans リソースのインポート）」を参照してください。

注意事項

cjimportjb コマンドの -d オプションには，インポートするディレクトリの最上位を指定してください。また，指定したディレクトリ下にあるすべてのファイルがインポートされるので，不要なファイルは指定したディレクトリ内に含めないようにしてください。

6.2.2　JavaBeans リソースのプロパティ定義

JavaBeans リソースのプロパティを定義します。

プロパティの設定手順の概要については，「3.3　属性ファイルによるプロパティの設定」を参照してください。次に JavaBeans リソースのプロパティ定義について説明します。

（1）編集する属性ファイル

JavaBeans リソース属性ファイル

（2）編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して，JavaBeans リソース属性ファイルを取得します。

実行形式

```
cjgetjbprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <JavaBeans
リソース表示名> -c <JavaBeansリソース属性ファイルパス>
```

実行例

```
cjgetjbprop MyServer -resname MyJavaBeansName -c
MyJavaBeansProp.xml
```

● 属性の設定

次に示すコマンドを実行して，JavaBeans リソース属性ファイルの値を反映します。

実行形式

```
cjsetjbprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <JavaBeans
リソース表示名> -c <JavaBeansリソース属性ファイルパス>
```

実行例

```
cjsetjbprop MyServer -resname MyJavaBeansName -c
MyJavaBeansProp.xml
```

(3) 編集する属性設定項目

JavaBeans リソースのプロパティ設定項目を次に示します。

- 実行時プロパティ
- 別名情報

(a) 実行時プロパティ

JavaBeans リソースの実行時プロパティ（<property> タグ）の設定項目を次に示します。

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

定義するプロパティの数だけ，上記の設定を繰り返してください。

プロパティ名（<property-name>）の設定項目に，JavaBeans リソースの set メソッドや get メソッドのメソッド名を指定します。プロパティ名（<property-name>）の設定項目については，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「4.1.13　<property> タグに指定できるプロ

パティ」を参照してください。

(b) 別名情報

JavaBeans リソースの別名情報（<resource-env-external-property> タグ）の別名（<optional-name>）を設定します。

JavaBeans リソースの別名の使用法については，「6.5　JNDI 名前空間に登録される J2EE リソース名の参照と変更」を参照してください。

6.2.3　JavaBeans リソースの開始

次に示すコマンドを実行して JavaBeans リソースを開始します。

実行形式

```
cjstartjb ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <JavaBeansリソース表示名>
```

実行例

```
cjstartjb MyServer -resname javabeansname
```

cjstartjb コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstartjb（JavaBeans リソースの開始）」を参照してください。

6.2.4　JavaBeans リソースの停止

次に示すコマンドを実行して JavaBeans リソースを停止します。

実行形式

```
cjstopjb ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <JavaBeansリソース表示名>
```

実行例

```
cjstopjb MyServer -resname javabeansname
```

cjstopjb コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstopjb（JavaBeans リソースの停止）」を参照してください。

注意事項

JavaBeans リソースを停止する前に，停止する JavaBeans リソースを使用してい

るアプリケーションはすべて停止させてください。

6.2.5 JavaBeans リソースの削除

インポートされている JavaBeans リソースを削除します。

準備

JavaBeans リソースを削除する前に，JavaBeans リソースを停止して J2EE サーバを再起動してください。

次に示すコマンドを実行して JavaBeans リソースを削除します。

実行形式

```
cjdeletejb ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -resname <JavaBeans
リソース表示名>
```

実行例

```
cjdeletejb MyServer -resname MyJavaBeans
```

cjdeletejb コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeletejb（JavaBeans リソースの削除）」を参照してください。

6.2.6 JavaBeans リソースの状態表示

すべての JavaBeans リソースの JavaBeans リソース名と状態（開始状態または停止状態）を表示します。

次に示すコマンドを実行して JavaBeans リソースの状態を表示します。

実行形式

```
cjlistjb ［<サーバ名称>］ ［-nameserver <プロバイダURL>］
```

実行例

```
cjlistjb MyServer
```

cjlistjb コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistjb（JavaBeans リソースの一覧表示）」を参照してください。

6.3　メールコンフィグレーションの設定

JavaMail を使用して SMTP サーバに接続する場合に必要な設定です。

6.3.1　メールコンフィグレーションの新規作成

メールコンフィグレーションを新規作成します。

次に示すコマンドを実行してメール属性ファイルを基にメールコンフィグレーションを新規作成します。

実行形式

```
cjsetresprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type mail -resname
<メールの表示名> -c <メール属性ファイルパス>
```

実行例

```
cjsetresprop MyServer -type mail -resname Mail -c MailProp.xml
```

cjsetresprop コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjsetresprop（リソースの属性設定)」を参照してください。属性ファイルについては，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「4.6　メール属性ファイル」を参照してください。

6.3.2　メールコンフィグレーションのプロパティ定義

メールコンフィグレーションのプロパティを定義します。新規作成時に定義した内容を変更できます。

プロパティの設定手順の概要については,「3.3　属性ファイルによるプロパティの設定」を参照してください。次にメールコンフィグレーションのプロパティ定義について説明します。

(1) 編集する属性ファイル

メール属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してメール属性ファイルを取得します。

実行形式

```
cjgetresprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type mail -resname
<メールの表示名> -c <メール属性ファイルパス>
```

実行例

```
cjgetresprop MyServer -type mail -resname Mail -c MailProp.xml
```

● 属性の設定

次に示すコマンドを実行して，メール属性ファイルを反映します。

実行形式

```
cjsetresprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type mail -resname
<メールの表示名> -c <メール属性ファイルパス>
```

実行例

```
cjsetresprop MyServer -type mail -resname Mail -c MailProp.xml
```

(3) 編集する属性設定項目

メールコンフィグレーションのプロパティ設定項目を次に示します。

項目	対応するタグ
E メール送信者の E メールアドレス	<from>
SMTP メールサーバのホスト名，または IP アドレス	<server>
別名情報※	<resource-external-property>

注※　別名情報（<resource-external-property>）には，次の項目を設定します。

別名情報の項目	必須	対応するタグ
リソースの別名	○	<optional-name>
リソース認証方式	△	<res-auth>
リソース共有の有無	△	<res-sharing-scope>

（凡例）○：必須　△：任意

プロパティの設定項目の説明については，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「4.6　メール属性ファイル」を参照してください。

6.3.3 メールコンフィグレーションの接続テスト

メールコンフィグレーションに設定した内容が正しいかどうか，メールサーバのコネクションをチェックします。

次に示すコマンドを実行してメールコンフィグレーションの接続テストを実施します。

実行形式

```
cjtestres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type mail -resname <メールコンフィグレーションの表示名>
```

実行例

```
cjtestres MyServer -type mail -resname Mymail1
```

cjtestres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjtestres（リソースの接続テスト）」を参照してください。

6.3.4 メールコンフィグレーションの削除

次に示すコマンドを実行してメールコンフィグレーションを削除します。

実行形式

```
cjdeleteres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type mail -resname <メールコンフィグレーションの表示名>
```

実行例

```
cjdeleteres MyServer -type mail -resname Mail
```

cjdeleteres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeleteres（リソースの削除）」を参照してください。

6.3.5 メールコンフィグレーションのコピー

次に示すコマンドを実行してメールコンフィグレーションのプロパティをコピーします。

実行形式

```
cjcopyres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type mail -src <コピー元のメールコンフィグレーションの表示名> -dst <コピー先のメールコンフィグレーションの表示名>
```

実行例

```
cjcopyres MyServer -type mail -src Mail -dst Mail2
```

cjcopyres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjcopyres（リソースのコピー）」を参照してください。

6.4　J2EE リソース一覧の参照

J2EE リソースについて，次の一覧を参照します。

- インポートされている EJB-JAR と，EJB-JAR に含まれる J2EE リソース
- インポートされている WAR と，WAR に含まれる J2EE リソース
- インポートされているメールコンフィグレーション

（1）EJB-JAR ファイルに含まれる J2EE リソースの一覧の参照

次に示すコマンドを実行して，EJB-JAR ファイルに含まれる J2EE リソースの一覧を参照します。

実行形式

```
cjlistres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type ejb ［-resname <
EJB-JARファイルの表示名>］
```

- -resname オプションを指定すると，特定の EJB-JAR ファイルに含まれるリソース（SessionBean，EntityBean，MessageDrivenBean）の一覧が表示されます。
- -resname オプションを指定しないと，インポートされている EJB-JAR ファイルの一覧が表示されます。

実行例

```
cjlistres MyServer -type ejb
```

（2）WAR ファイルに含まれる J2EE リソースの一覧の参照

次に示すコマンドを実行して，WAR ファイルに含まれる J2EE リソースの一覧を参照します。

実行形式

```
cjlistres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type war ［-resname <
WARファイルの表示名>］
```

- -resname オプションを指定すると，特定の WAR ファイルに含まれるリソース（サーブレット，JSP およびフィルタ）の一覧が表示されます。
- -resname オプションを指定しないと，インポートされている WAR ファイルの一覧が表示されます。

実行例

```
cjlistres MyServer -type war
```

(3) メールコンフィグレーションの一覧の参照

次に示すコマンドを実行して，インポートされているメールコンフィグレーションの一覧を参照します。

実行形式

```
cjlistres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type mail
```

実行例

```
cjlistres MyServer -type mail
```

cjlistres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistres（リソースの一覧表示）」を参照してください。

6.5　JNDI 名前空間に登録される J2EE リソース名の参照と変更

JNDI 名前空間に登録される J2EE リソース名を変更します。

J2EE リソース名には別名も付けられます。別名を付けることで，JNDI 名前空間から任意の名前で J2EE リソースを参照できるようになります。なお，この機能を**ユーザ指定名前空間機能**といいます。

なお，J2EE リソースの JNDI 名前空間については，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。ユーザ指定名前空間機能については，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「2.4　Enterprise Bean または J2EE リソースへの別名付与（ユーザ指定名前空間機能）」を参照してください。

別名は，次の属性ファイルに追加できます。

属性ファイル	対応するタグ名
メール属性ファイル	<resource-external-property>
JavaBeans リソース属性ファイル	<resource-env-external-property>

メール属性ファイルの <resource-external-property> タグの編集については，「6.3.2 メールコンフィグレーションのプロパティ定義」を，JavaBeans リソース属性ファイルの <resource-env-external-property> タグの編集については，「6.2.2　JavaBeans リソースのプロパティ定義」の別名情報の設定を参照してください。

注意事項

開始状態の J2EE アプリケーションがある場合，別名を設定している J2EE リソースの停止および削除はできません。J2EE サーバで開始されているすべての J2EE アプリケーションを停止させてください。

7 J2EE アプリケーションの作成

この章では，サーバ管理コマンドを使用した J2EE アプリケーションの作成方法について説明します。

7.1 J2EE アプリケーションの作成の概要

J2EE アプリケーションの作成とは，アプリケーション開発環境で作成した Enterprise Bean（EJB-JAR）およびサーブレットと JSP（WAR）を，一つの J2EE アプリケーション（EAR）にすることです。また，J2EE アプリケーションに，その J2EE アプリケーションで使用するリソースアダプタを含めることもできます。

J2EE アプリケーションの作成に必要な作業の概要を次の表に示します。

表 7-1 J2EE アプリケーションの作成で実施する作業

実施項目	内容	参照先
Enterprise Bean（EJB-JAR）のインポート	J2EE アプリケーションの構成に Enterprise Bean（EJB-JAR）が含まれる場合に必要な作業です。 J2EE サーバに Enterprise Bean をインポートします。	7.2
サーブレットと JSP（WAR）のインポート	J2EE アプリケーションの構成にサーブレット・JSP（WAR）が含まれる場合に必要な作業です。 J2EE サーバにサーブレット・JSP をインポートします。	7.3
リソースアダプタ（RAR）のインポート	J2EE アプリケーションにリソースアダプタ（RAR）を含める場合，含めるリソースアダプタを J2EE サーバにインポートします。	7.4
J2EE アプリケーションの新規作成	インポートした EJB-JAR，WAR を基に EAR を作成します。また，必要に応じて，J2EE アプリケーションに RAR を含めます。	7.5
J2EE アプリケーションへのライブラリ JAR ファイルの追加	J2EE アプリケーションにライブラリ JAR（ファイル拡張子は小文字の「.jar」）を追加します。	7.6
ライブラリ JAR ファイルの一覧の参照	J2EE アプリケーションに含まれているライブラリ JAR ファイルの一覧を参照します。	7.7
J2EE アプリケーションからのライブラリ JAR ファイルの削除	J2EE アプリケーションに含まれているライブラリ JAR ファイルを削除します。	7.8
J2EE アプリケーションへの参照ライブラリの設定	J2EE アプリケーションに参照ライブラリを設定します。	7.9
J2EE アプリケーションのプロパティの設定	作成した J2EE アプリケーションのプロパティを設定します。	9.
J2EE アプリケーションの実行	設定が完了した J2EE アプリケーションを実行します。また，運用に応じて，停止します。	10.

cosminexus.xml を使用した J2EE アプリケーションを作成する場合は，あらかじめ作成した cosminexus.xml を，EJB-JAR，WAR などと一緒に J2EE アプリケーションに含め

て使用してください。

ライブラリ JAR と参照ライブラリは，J2EE アプリケーション内の各モジュールから参照できる共通ライブラリです。

ライブラリ JAR と参照ライブラリの特長を次に示します。

● ライブラリ JAR

- J2EE アプリケーションに JAR（ファイル拡張子は小文字の「.jar」）を含めます。
- 複数の J2EE アプリケーションで使用する場合，それぞれの J2EE アプリケーションに追加する必要があります。
- class ファイルを含めることはできません。必ず JAR ファイル形式にまとめて使用します。

● 参照ライブラリ

- 参照するライブラリファイルの絶対パスを指定します。
- 複数の J2EE アプリケーションで使用する場合，同じ参照先を指定するだけで利用できます。
- JAR ファイル，class ファイルどちらも扱うことができます。

7.2 Enterprise Bean（EJB-JAR）のインポート

EJB コンポーネントを利用して J2EE アプリケーションを実行する前に，Enterprise Bean（EJB-JAR）をインポートします。インポートできるのは，EJB 1.1，2.0，2.1，または 3.0 の仕様に準拠した DD を持つ EJB-JAR，またはアノテーションを使用した EJB-JAR です。

なお，次のディレクトリに EJB の各種サンプルプログラムが格納されています。

Windows の場合
＜ Cosminexus のインストールディレクトリ＞ ¥CC¥examples¥

UNIX の場合
/opt/Cosminexus/CC/examples/

DD を持った EJB-JAR の場合，DD が DTD に従っていることを確認してから，インポートを実行してください。DD が DTD の仕様に従っていない EJB-JAR をインポートした場合，メッセージが出力されます。メッセージが出力された場合は，マニュアル「Cosminexus アプリケーションサーバ メッセージ 2」を参照してください。

次に示すコマンドを実行して EJB-JAR をインポートします。

実行形式

```
cjimportres ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -type ejb -f ＜
EJB-JARファイルパス＞
```

実行例

```
cjimportres MyServer -type ejb -f account.jar
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

注意事項

- DD のバージョンが 1.1 の EJB-JAR をインポートすると，J2EE サーバ内で DD のバージョンが 2.0 に変更されます。
- EJB 1.1，2.0，2.1，または 3.0 の仕様に準拠しない DD を持つ EJB-JAR はインポートできません。
- 「hitachi-runtime.jar」の名称の EJB-JAR はインポートしないでください。
- インポート時に指定した JAR ファイル名は作業ディレクトリ中のディレクトリ名として使用されます。また，JAR ファイル中に，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，usrconf.properties ファイルの定義内容に応じて，デプロイ・スタート時に作業ディレクトリ内にスタブが生成されます。デフォルトの設定の場合，スタブは，ejb-jar.xml の <remote> および <home> に記述されたインタフェースに対して作成されます。usrconf.properties ファイルの ejbserver.deploy.stub.generation.scope キーに app を指定した場合は，JAR ファイルに含まれるリモートインタフェースを実装したすべてのインタフェースのスタブが生成されます。
これらのことを考慮して，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 JAR ファイル名および 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，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「7.10.2　作業ディレクトリのパス長の見積もり式」を参照してください。

7.3 サーブレットと JSP（WAR）のインポート

J2EE サーバモードで Web コンポーネントを利用して J2EE アプリケーションを実行する前に，サーブレットおよび JSP（WAR）をインポートします。

なお，インポートできるのは，Servlet 2.2，2.3，2.4，または 2.5 の仕様に準拠した DD を持つ WAR，またはアノテーションを指定した WAR です。このため，DD が DTD に従っていることを確認してから，インポートを実行してください。DD が DTD の仕様に従っていない WAR をインポートした場合，メッセージが出力されます。メッセージが出力された場合は，マニュアル「Cosminexus アプリケーションサーバ メッセージ 2」を参照してください。

次に示すコマンドを実行して WAR をインポートします。

実行形式

```
cjimportres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type war -f <WAR
ファイルパス>
```

実行例

```
cjimportres MyServer -type war -f account.war
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

注意事項

- DD のバージョンが 2.2 の WAR をインポートすると J2EE サーバ内で DD のバージョンが 2.3 に変更されます。
- Servlet 2.2，2.3，2.4，または 2.5 の仕様に準拠しない DD を持つ WAR はインポートできません。
- インポート時に指定した WAR ファイル名は作業ディレクトリ中のディレクトリ名として使用されます。また，WAR ファイル内のファイルは，デプロイ・スタート時に作業ディレクトリ内に展開されます。WAR ファイル内に，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，usrconf.properties ファイルの定義内容に応じて，デプロイ・スタート時に作業ディレクトリ内にスタブが生成されます。usrconf.properties ファイルの ejbserver.deploy.stub.generation.scope キーに app を指定した場合は，WAR ファイルに含まれるリモートインタフェースを実装したすべてのインタフェースのスタブが生成されます。

これらのことを考慮して，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 WAR ファイル名，WAR ファイル中のパッケージ名やクラス名および 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，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「7.10.2　作業ディレクトリのパス長の見積もり式」を参照してください。

- 「hitachi-runtime.jar」の名称の WAR はインポートしないでください。

7.4 リソースアダプタ（RAR）のインポート

J2EE アプリケーションに含めるリソースアダプタ（RAR）をインポートします。J2EE アプリケーションにリソースアダプタを含めると，同じ J2EE アプリケーションの Enterprise Bean（EJB-JAR），サーブレットおよび JSP（WAR）からリソースアダプタが使用できます。

インポートできるリソースアダプタについては，「5.1.1　利用できるリソースアダプタ」を参照してください。

次に示すコマンドを実行して RAR をインポートします。

実行形式

```
cjimportres ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -f <RAR
ファイルパス>
```

実行例

```
cjimportres MyServer -type rar -f account.rar
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

注意事項

Connector 1.0 以降の仕様に準拠する DD を持つ RAR であることを確認してから，インポートを実行してください。J2EE Connector 1.0 以降の仕様に準拠しない DD を持つ RAR はインポートできません。

7.5　J2EE アプリケーションの新規作成

インポートした EJB-JAR，WAR を基に J2EE アプリケーションを作成します。また，EJB-JAR，WAR で使用する RAR を追加します。

(1) EJB-JAR ファイルの追加

次に示すコマンドを実行して EJB-JAR ファイルを追加します。J2EE アプリケーションがない場合は，J2EE アプリケーションが新規作成されます。

実行形式

```
cjaddapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type ejb -name <J2EE
アプリケーション名> -resname <EJB-JAR表示名>
```

実行例

```
cjaddapp MyServer -type ejb -name adder -resname adder
```

cjaddapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjaddapp（リソースの追加）」を参照してください。

(2) WAR ファイルの追加

次に示すコマンドを実行して WAR ファイルを追加します。J2EE アプリケーションがない場合は，J2EE アプリケーションが新規作成されます。

実行形式

```
cjaddapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type war -name <J2EE
アプリケーション名> -resname <WAR表示名>
```

実行例

```
cjaddapp MyServer -type war -name adder -resname adder_war
```

cjaddapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjaddapp（リソースの追加）」を参照してください。

(3) RAR ファイルの追加

次に示すコマンドを実行して RAR ファイルを追加します。J2EE アプリケーションがない場合は，J2EE アプリケーションが新規作成されます。

実行形式

```
cjaddapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type rar -name <J2EE
アプリケーション名> -resname <RAR表示名>
```

実行例

```
cjaddapp MyServer -type rar -name adder -resname account-rar
```

cjaddapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjaddapp（リソースの追加）」を参照してください。

（4） 注意事項

- J2EE アプリケーション作成時には，EJB ホームオブジェクトの JNDI 名前空間（HITACHI_EJB/SERVERS/ ＜サーバ名称＞ /EJB/ ＜ J2EE APP 名称＞ / ＜ Enterprise Bean 名称＞）の＜ J2EE APP 名称＞に該当する名前が自動的に割り当てられます。JNDI 名前空間については，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。
- 展開ディレクトリ形式のアプリケーションに対しては，EJB-JAR ファイル，WAR ファイルおよび RAR ファイルの追加はできません。
- 拡張子が ".jar" ではない EJB-JAR ファイル，拡張子が ".war" ではない WAR ファイル，および拡張子が ".rar" ではない RAR ファイルは，application.xml を省略したアプリケーションに追加できません。
- フィルタは，web.xml を省略した WAR ファイルに追加できません。
- J2EE アプリケーション名には，半角英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_）だけを指定してください。
- J2EE アプリケーションを作成したあとでは，J2EE アプリケーション名を変更できません。
- J2EE アプリケーションを含んだまま 06-50 よりも前のバージョンからアップグレードインストールした場合，Display name に指定した名前が作業ディレクトリ中のディレクトリ名として使用されます。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 Display name 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，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「7.10.2　作業ディレクトリのパス長の見積もり式」を参照してください。

7.6　J2EE アプリケーションへのライブラリ JAR ファイルの追加

J2EE アプリケーションにライブラリ JAR ファイルを追加します。

次に示すコマンドを実行して J2EE アプリケーションへライブラリ JAR ファイルを追加します。

実行形式

```
cjimportlibjar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -f <ライブラリJAR名>
```

実行例

```
cjimportlibjar MyServer -name App1 -f applib.jar
```

cjimportlibjar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportlibjar（ライブラリ JAR のインポート）」を参照してください。

注意事項

- インポート時に指定した JAR ファイル名は，作業ディレクトリ中のディレクトリ名として使用されます。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 JAR ファイル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，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「7.10.2　作業ディレクトリのパス長の見積もり式」を参照してください。
- ライブラリ JAR ファイルの拡張子には小文字の「.jar」を指定してください。
- 展開ディレクトリ形式の J2EE アプリケーションへのライブラリ JAR の追加・削除はできません。
- 「hitachi-runtime.jar」の名称のライブラリ JAR はインポートしないでください。

7.7 ライブラリ JAR ファイルの一覧の参照

J2EE アプリケーションにインポートされているライブラリ JAR ファイルの一覧を参照します。

次に示すコマンドを実行して J2EE アプリケーションにインポートされているライブラリ JAR ファイルの一覧を参照します。

実行形式

```
cjlistlibjar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名>
```

実行例

```
cjlistlibjar MyServer -name App1
```

cjlistlibjar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistlibjar（ライブラリ JAR の一覧表示）」を参照してください。

7.8 J2EE アプリケーションからのライブラリ JAR ファイルの削除

J2EE アプリケーションにインポートされているライブラリ JAR ファイルを削除します。

次に示すコマンドを実行して J2EE アプリケーションからライブラリ JAR ファイルを削除します。

実行形式

```
cjdeletelibjar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -f <ライブラリJARファイル名称>
```

実行例

```
cjdeletelibjar MyServer -name App1 -f applib.jar
```

cjdeletelibjar コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjdeletelibjar（ライブラリ JAR の削除)」を参照してください。

注意事項

展開ディレクトリ形式の J2EE アプリケーションへのライブラリ JAR の追加・削除はできません。

7.9 J2EE アプリケーションへの参照ライブラリの設定

J2EE アプリケーションに参照ライブラリを設定します。

J2EE アプリケーションへの参照ライブラリの設定は，J2EE アプリケーション属性ファイルで設定します。プロパティを設定する手順については，「3.3 属性ファイルによるプロパティの設定」を参照してください。

(1) 編集する属性ファイル

アプリケーション属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してアプリケーション属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -c <アプリケーション属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name App1 -c C:¥home¥app1.xml
```

● 属性の設定

次に示すコマンドを実行してアプリケーション属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -c <アプリケーション属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name App1 -c C:¥home¥app1.xml
```

(3) 編集する属性設定項目

参照するライブラリ JAR の指定（<ref-libraries>）のクラスパス（<classpath>）を設定します。

ここで設定した参照ライブラリのクラスを属性ファイルの中で指定する場合，サーバ管

理コマンドを実行するマシン上にその参照ライブラリファイルをコピーし，サーバ管理コマンド用オプション定義ファイルの USRCONF_JVM_CLASSPATH に，コピーした参照ライブラリファイルのパスを設定する必要があります。

8 J2EE アプリケーションのインポートとエクスポート

この章では，サーバ管理コマンドを使用した J2EE アプリケーションのインポートとエクスポートの方法について説明します。

8.1 J2EE アプリケーションのインポート

J2EE アプリケーションをインポートします。インポートできるのは，J2EE アプリケーション 1.2，1.3，1.4，および Java EE 5 の仕様に準拠した DD を持つ J2EE アプリケーション，またはアノテーションを使用した J2EE アプリケーションです。DD を持った J2EE アプリケーションの場合，EAR，EJB-JAR，WAR，および RAR の DD が DTD に従っていることを確認してから，インポートを実行してください。

インポートする J2EE アプリケーションには，次の形式があります。

- アーカイブ形式の J2EE アプリケーション
  J2EE サーバの作業ディレクトリ以下に，EJB-JAR/WAR/RAR ファイルのコンポーネントを持つ J2EE アプリケーションです。
- 展開ディレクトリ形式の J2EE アプリケーション
  J2EE サーバの外部に置かれている構成ファイルを，論理的な J2EE アプリケーションとして使用する J2EE アプリケーションです。また，リロードによる J2EE アプリケーションの入れ替えを実施する場合は，この形式でのインポートが必要になります。J2EE アプリケーションの入れ替えについては，「10.6 J2EE アプリケーションの入れ替え」を参照してください。

8.1.1 アーカイブ形式の J2EE アプリケーションのインポート

次に示すコマンドを実行して，アーカイブ形式の J2EE アプリケーションをインポートします。

実行形式

```
cjimportapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -f <EARファイルのパス
>
```

実行例

```
cjimportapp MyServer -f C:¥home¥bank.ear
```

cjimportapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportapp（J2EE アプリケーションのインポート）」を参照してください。

(1) 実行時情報に関する注意事項

実行時情報は Cosminexus の J2EE サーバ独自の情報です。次の情報が含まれています。

表 8-1　実行時情報一覧

項目	情報
J2EE アプリケーションの状態	開始／停止状態の情報
J2EE アプリケーションの各種設定	• Enterprise Bean の実行時プロパティ • 各種リファレンスと J2EE リソースのマッピング • CMP フィールドのデータベースのマッピング など
自動生成実装クラス	• EJB ホームオブジェクト • EJB オブジェクト
リモートインタフェースの stub クラス，tie クラス	• EJB ホームオブジェクト • EJB オブジェクト

実行時情報付き EAR ファイルをインポートした場合の注意事項を，次に説明します。

● エクスポートされたときの J2EE アプリケーションの状態が復元されます。したがって，開始済みの J2EE アプリケーションをインポートした場合，すぐに J2EE アプリケーションが開始されます。

● J2EE アプリケーションの各種設定が復元されます。インポートする J2EE サーバ上に，エクスポートした J2EE サーバと同じ名称で J2EE リソースアダプタ，JDBC データソース，または JavaMail コンフィグレーションを作成しておくと，インポート時に参照が解決されます。このため，開始済みの J2EE アプリケーションをインポートする場合は，エクスポートした J2EE サーバと同じ名称で J2EE リソースアダプタ，JDBC データソースまたは JavaMail コンフィグレーションを事前に作成しておいてください。また，J2EE リソースアダプタを開始させた状態で，J2EE アプリケーションをインポートしてください。

● 実行時情報付き EAR ファイルに含まれる，自動生成実行クラス，リモートインタフェースの stub クラス，tie クラスは再利用されます。ただし，旧バージョンでエクスポートした EAR ファイルをインポートした場合は，再作成されます。

● 実行環境情報ファイルは変更しないようにしてください。

(2) EAR の DD に関する注意事項

J2EE アプリケーション 1.2，1.3，1.4，および Java EE 5 の仕様に準拠しない DD を持つ EAR はインポートできません。

DD のバージョンが古い場合，バージョンを次のように変更します。

表 8-2　DD のバージョンの変更

DD	インポート前のバージョン	インポート後のバージョン
application.xml	1.2	1.4

DD	インポート前のバージョン	インポート後のバージョン
	1.3	
ejb-jar.xml	1.1	2.0
web.xml	2.2	2.3

(3) そのほかの注意事項

- インポートするファイルが，J2EE サーバからエクスポートした J2EE アプリケーションでない場合，次の名称のファイルを含めないでください。
  - hitachi-runtime.jar
  - rar.properties
  - fileinfo.properties
  - META-INF/hitachi-ra.xml
- J2EE アプリケーション開始時には，EJB ホームオブジェクトの JNDI 名前空間（HITACHI_EJB/SERVERS/ ＜サーバ名称＞ /EJB/ ＜ J2EE APP 名称＞ / ＜ Enterprise Bean 名称＞）の＜ J2EE APP 名称＞に該当する名前が自動的に割り当てられます。JNDI 名前空間については，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。
- EAR ファイル中の EJB-JAR ファイル名は作業ディレクトリ中のディレクトリ名として使用されます。また，EJB-JAR ファイル中に，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，usrconf.properties ファイルの定義内容に応じて，デプロイ・スタート時に作業ディレクトリ内にスタブが生成されます。デフォルトの設定の場合，スタブは，ejb-jar.xml の <remote> および <home> に記述されたインタフェースに対して作成されます。usrconf.properties ファイルの ejbserver.deploy.stub.generation.scope キーに app を指定した場合は，EJB-JAR ファイルに含まれるリモートインタフェースを実装したすべてのインタフェースのスタブが生成されます。
  これらのことを考慮して，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 EJB-JAR ファイル名，java.rmi.Remote を実装したクラスのパッケージ名，およびクラス名を指定してください。
  なお，作業ディレクトリのパス長の見積もりについては，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「7.10.2　作業ディレクトリのパス長の見積もり式」を参照してください。
- EAR ファイル中の WAR ファイル名は作業ディレクトリ中のディレクトリ名として使用されます。また，WAR ファイル内のファイルは作業ディレクトリ内に展開されます。WAR ファイル中に，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，usrconf.properties ファイルの定義内容に応じて，デプロイ・スタート時に作業ディレクトリ内にスタブが生成されます。usrconf.properties ファイルの

ejbserver.deploy.stub.generation.scope キーに app を指定した場合は，WAR ファイルに含まれるリモートインタフェースを実装したすべてのインタフェースのスタブが生成されます。
これらのことを考慮して，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 WAR ファイル名，WAR ファイル中のパッケージ名やクラス名，および 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。

- EAR ファイル中のライブラリ JAR ファイル名は，作業ディレクトリ中のディレクトリ名として使用されます。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 JAR ファイル名を指定してください。

- EAR ファイル中の EJB-JAR 同士，WAR 同士で表示名が重複している場合はインポートできません。表示名が空文字または指定されていない場合は，ファイル名を変換した文字列が指定されます。この値が重複した場合もインポートできません。

- インポートした実行時情報付きの J2EE アプリケーションの実行時情報にエラーがある場合，J2EE アプリケーションのインポートに失敗します。このとき，J2EE アプリケーションは自動開始されないで削除されます。J2EE アプリケーションが削除されないようにするには，cjimportapp コマンドの -nodelete オプションを指定して J2EE アプリケーションをインポートしてください。このとき，J2EE アプリケーションは削除されないでインポートできますが，自動開始されないので，必要に応じてエラーを取り除き，J2EE アプリケーションを開始してください。

- インポートするアプリケーションに application.xml がある場合の表示名，およびディレクトリ名の変換規則は次のとおりです。
  - application.xml の <display-name> タグの値が表示名となります。
  - <display-name> タグの値がない場合，-f オプションに指定した EAR ファイル名を変換した文字列が表示名になります。
    なお，ファイル名に汎用文字と数字以外の文字が含まれている場合，その文字はアンダースコア (_) に置き換えられます。
    置換対象の文字が続けて出現する場合は一つのアンダースコア (_) に纏められます。
  - application.xml の <module> タグに記述されたパス名から拡張子を除いた名称が EJB-JAR ディレクトリ名，または WAR ディレクトリ名となります。

- インポートするアプリケーションに application.xml がない場合の表示名，およびディレクトリ名の変換規則は次のとおりです。
  - -f オプションに指定したファイル名から拡張子を除き，変換した文字列が表示名になります。
    なお，半角英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア (_) 以外の文字はアンダースコア (_) に置き換えられます。
  - ピリオド（.）が最初に現れるファイル名の場合は，ピリオド（.）を除かないで変換した文字列が表示名になります。
  - EJB-JAR ディレクトリ名の最後は「_jar」，WAR ディレクトリ名の最後は「_war」となります。

● 06-50 より前のバージョンの Cosminexus Component Container からアップグレードインストールした場合，J2EE サーバ上にあるデータソースは，J2EE リソースアダプタに自動変換されます。
J2EE リソースアダプタに自動変換される前のデータソースとリファレンスを解決した J2EE アプリケーションを，実行時情報を含む状態でエクスポートし，06-50 以降のバージョンの Cosminexus Component Container にインポートする場合，インポートされた J2EE アプリケーションはリファレンス解決されていません。このため，次の 2 点に注意してください。
- インポートされた J2EE アプリケーションのリファレンス解決はされていないので，エクスポートした J2EE アプリケーションが開始状態の場合でも，J2EE アプリケーションは自動開始に失敗します。この場合は，cjimportapp コマンドの -nodelete オプションを指定して J2EE アプリケーションをインポートしてください。
- インポート後は，リファレンス解決を実施してから，J2EE アプリケーションを開始してください。

8.1.2 展開ディレクトリ形式の J2EE アプリケーションのインポート

展開ディレクトリ形式の J2EE アプリケーションは，J2EE サーバの作業ディレクトリ以下に各コンポーネントの class ファイルや JAR ファイルを持ちません。DD と展開ディレクトリのパス情報だけを持ち，EAR ファイルまたは ZIP ファイルが展開ディレクトリのパスに展開されている J2EE アプリケーションです。ただし，J2EE アプリケーションに含まれる RAR ファイルはアーカイブ形式で格納されています。

展開ディレクトリ形式の J2EE アプリケーションのインポートには，次の二つの方法があります。

- アプリケーションディレクトリを展開ディレクトリ形式のアプリケーションとしてインポートします。
- J2EE アプリケーションを展開ディレクトリ形式のアプリケーションとしてインポートします。

! 注意事項

展開ディレクトリ形式のアプリケーションをインポートした場合，J2EE サーバの起動処理中またはサーバ管理コマンド実行中に，アプリケーションディレクトリ以下のディレクトリおよびファイルの追加，削除および上書きはしないでください。

展開ディレクトリ形式のアプリケーションのインポート手順を，次に示します。

(1) アプリケーションディレクトリを展開ディレクトリ形式のアプリケーションとしてインポート

次の手順で，J2EE サーバにアプリケーションディレクトリを登録します。

1. **アプリケーションディレクトリを作成します。**
   展開ディレクトリ形式の J2EE アプリケーションを新規に作成する場合は，アプリケーションディレクトリを作成します。アプリケーションディレクトリの構成については，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「12.4.2　アプリケーションディレクトリの構成」を参照してください。
2. **アプリケーションディレクトリをインポートします。**
   上記 1. で作成した，アプリケーションディレクトリを J2EE サーバに登録します。
3. **次に示すコマンドを実行して，アプリケーションディレクトリをインポートします。**

実行形式

```
cjimportapp ［<サーバ名>］ -a <アプリケーションディレクトリパス>
```

実行例

```
cjimportapp MyServer -a AppDirPath
```

cjimportapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportapp（J2EE アプリケーションのインポート)」を参照してください。

注意事項

- リモート環境では実行できません。
- application.xml に <alt-dd> タグを指定している，アプリケーションディレクトリを展開ディレクトリ形式で使用できません。
- インポートするアプリケーションに application.xml がある場合の表示名，およびディレクトリ名の変換規則は次のとおりです。
  - application.xml の <display-name> タグの値が表示名となります。
  - <display-name> タグの値がない場合，-f オプションに指定した EAR ファイル名を変換した文字列が表示名になります。
    なお，ファイル名に汎用文字と数字以外の文字が含まれている場合，その文字はアンダースコア (_) に置き換えられます。
    置換対象の文字が続けて出現する場合は一つのアンダースコア (_) に纏められます。
  - application.xml の <module> タグに記述されたパス名から拡張子を除いた名称が EJB-JAR ディレクトリ名，または WAR ディレクトリ名となります。

- インポートするアプリケーションに application.xml がない場合の表示名，およびディレクトリ名の変換規則は次のとおりです。
  - -f オプションに指定したファイル名から拡張子を除き，変換した文字列が表示名になります。
    なお，半角英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア (_) 以外の文字はアンダースコア (_) に置き換えられます。
  - ピリオド（.）が最初に現れるファイル名の場合は，ピリオド（.）を除かないで変換した文字列が表示名になります。
  - EJB-JAR ディレクトリ名の最後は「_jar」，WAR ディレクトリ名の最後は「_war」となります。
- J2EE サーバ内で，同じディレクトリをアプリケーションディレクトリとする J2EE アプリケーションがすでにある場合，インポートできません。
- -d オプションを指定したインポートで展開ディレクトリに展開されたアプリケーションディレクトリが，次の場合，-a オプションを指定してインポートできません。
  - EJB-JAR のモジュール名が「.jar」で終わっていない。
  - WAR のモジュール名が「.war」で終わっていない。
  - 拡張子を含まないモジュール名称が，拡張子を含まないほかのモジュール名称と同一である。
  - 拡張子を含まないモジュールの名称が，EAR ファイル内のディレクトリと同一である。
- アプリケーションディレクトリ配下の各種コンポーネントの DD は UTF-8 エンコーディングになります。
- DD のバージョンが古い場合，バージョンを次のように変更します。

表 8-3 DD のバージョンの変更

DD	インポート前のバージョン	インポート後のバージョン
application.xml	1.2	1.4
	1.3	
ejb-jar.xml	1.1	2.0
web.xml	2.2	2.3

(2) J2EE アプリケーションを展開ディレクトリ形式のアプリケーションとしてインポート

EAR ファイルまたはエクスポートした実行時情報付き EAR ファイルを展開ディレクトリ形式でインポートします。

次に示すコマンドを実行して，アプリケーションを展開ディレクトリ形式でインポートします。

実行形式

```
cjimportapp ［<サーバ名>］-f <ファイルパス> -d <展開ディレクトリパス>
```

EAR ファイルまたは ZIP ファイルの内容は，展開ディレクトリパスに展開されます。

実行例

```
cjimportapp MyServer -f App1.zip -d ApplicationDir
```

cjimportapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportapp（J2EE アプリケーションのインポート）」を参照してください。

注意事項

- J2EE サーバを起動したユーザに対して，展開先ディレクトリパスに指定したディレクトリの書き込み権限が必要です。
- application.xml に <alt-dd> タグを指定している，J2EE アプリケーションを展開ディレクトリ形式で使用することはできません。
- J2EE サーバ内で，同じディレクトリを展開ディレクトリとする J2EE アプリケーションがすでにある場合，インポートできません。
- そのほかの注意事項は，アーカイブ形式の J2EE アプリケーションのインポートと同じです。アーカイブ形式の J2EE アプリケーションのインポートについては，「8.1.1 アーカイブ形式の J2EE アプリケーションのインポート」の注意事項を参照してください。
- アプリケーションディレクトリ配下の各種コンポーネントの DD は UTF-8 エンコーディングになります。

8.2 J2EE アプリケーションのエクスポート

次に示すコマンドを実行して J2EE アプリケーションをエクスポートします。

実行形式

```
cjexportapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -f <EARファイルパス> ［-raw|-normal］
```

実行時情報を含める場合

-normal を指定してください（デフォルト値）。

実行時情報を含めない場合

-raw を指定してください。

実行例

```
cjexportapp MyServer -name account -f C:¥home¥account.zip
```

この例の場合は，EAR ファイルの実行時情報が含まれます。

cjexportapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjexportapp（J2EE アプリケーションのエクスポート）」を参照してください。

J2EE アプリケーションのエクスポート時の注意事項を，次に示します。

- 実行時情報に関する注意事項
- EAR の DD に関する注意事項

(1) 実行時情報に関する注意事項

● J2EE アプリケーションをエクスポートする場合，次に示すどちらかを選択できます。デフォルトは実行時情報付きファイルの形式（ZIP 形式）になります。

- 実行時情報付きファイル（ZIP 形式）
  実行時情報を含んだ ZIP 形式のファイルです。なお，実行時情報付きファイルは，Cosminexus 独自のファイルです。
- EAR
  Java 2 Platform, Enterprise Edition で規定された EAR です。実行時情報を含みません。

実行時情報を含んだ形式でエクスポートした J2EE アプリケーションには，下位の互換性はありません。J2EE アプリケーションの内容は，変更しないでください。
複数の J2EE サーバから構成されるシステムで，J2EE サーバ間で JDBC リソースの設定，利用するデータベース，および J2EE アプリケーションの実行時プロパティを同じにする場合は，実行時情報付き EAR ファイルを使用します。このとき一方の

J2EE サーバで作成した J2EE アプリケーションをエクスポートし，他方の J2EE サーバ上にインポートさせることを推奨します。
J2EE サーバ上に作成した J2EE アプリケーションを別のシステムの J2EE サーバや同じシステムでも実行環境が異なる J2EE サーバにインポートする場合，またはほかの J2EE サーバ製品でも使用する場合には，EAR ファイル形式を使用して J2EE アプリケーションをエクスポートしてください。

- J2EE サーバに登録されている展開ディレクトリ形式のアプリケーションをエクスポートすると，EAR ファイルまたは実行時情報付き ZIP ファイルを生成します。
- アプリケーション開始時に JSP 事前コンパイルを実行したアプリケーションをエクスポートすると，エクスポートされた EAR ファイルには JSP コンパイル結果が含まれます。JSP コンパイル結果が含まれた EAR ファイルを，ほかの J2EE サーバにインポートすると，EAR ファイルに含まれる JSP ファイルのコンパイル結果を使用できます。
- インポートした J2EE アプリケーションが cosminexus.xml を含んでいる場合，cosminexus.xml を含んだ状態でエクスポートします。cosminexus.xml を含むアプリケーションのエクスポートについては，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「10.3.6　cosminexus.xml を含むアプリケーションの運用」を参照してください。

(2) EAR の DD に関する注意事項

- EAR の DD のバージョンは，J2EE アプリケーションのバージョンによって，1.4 以降になります。
- EAR に含まれる EJB-JAR の DD のバージョンは，EJB-JAR のバージョンによって，2.0 以降になります。
- EAR に含まれる WAR の DD のバージョンは，WAR のバージョンによって，2.3 以降になります。
- 07-00 より前のバージョンの Cosminexus Component Container の J2EE アプリケーションで，コンポーネントの追加，削除およびプロパティの変更を実施していない場合，EAR，EJB-JAR，WAR の DD はアップグレードインストールのときのバージョンとなります。
- EAR に含まれる各種コンポーネントの DD は UTF-8 エンコーディングになります。

9 J2EE アプリケーションのプロパティ設定

この章では，サーバ管理コマンドを使用した J2EE アプリケーションのプロパティの設定方法について説明します。

9.1　J2EE アプリケーションのプロパティ設定の概要

J2EE アプリケーションのプロパティ設定とは，インポートした J2EE アプリケーションの属性や動作を定義することです。この節では，次の説明をします。

- J2EE アプリケーションのプロパティ設定と属性ファイル
- Enterprise Bean のプロパティ設定項目
- サーブレットと JSP のプロパティ設定項目
- J2EE アプリケーション（共通）のプロパティ設定項目

注意事項

アノテーションで指定した項目は変更できません。また，cosminexus.xml を含むアプリケーションのプロパティを変更するには，サーバ管理コマンドを使って設定する方法と，MyEclipce 使って設定する方法があります。サーバ管理コマンドで cosminexus.xml にある定義情報を変更したあとでリデプロイ機能を使用すると，サーバ管理コマンドで変更した情報は失われます。
MyEclipce を使った cosminexus.xml を含むアプリケーションのプロパティの変更については，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「5.3.2　cosminexus.xml エディタの操作方法」を参照してください。また，cosminexus.xml を含むアプリケーションの運用については，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「10.3.6 cosminexus.xml を含むアプリケーションの運用」を参照してください。

J2EE アプリケーションに含まれるリソースアダプタのプロパティ設定と属性ファイルについては，「5.4　リソースアダプタのプロパティ定義」を参照してください。

(1) J2EE アプリケーションのプロパティ設定と属性ファイル

プロパティの設定手順の概要については，「3.3　属性ファイルによるプロパティの設定」を参照してください。

次の属性ファイルを利用して，J2EE アプリケーションのプロパティを設定できます。

- 固有の属性ファイルを利用します。
  J2EE アプリケーションを構成するコンポーネントの固有属性ファイルごとに，プロパティを設定します。
- アプリケーション統合属性ファイルを利用します。
  固有の属性ファイルの内容を一括して，プロパティを設定します。

J2EE アプリケーションのプロパティの設定では，固有の属性ファイルとアプリケーション統合属性ファイルを個々に利用することも，合わせて利用することもできます。

この章では，主に，固有の属性ファイルを使用してのプロパティ設定について説明しています。アプリケーション統合属性ファイルを使用してのプロパティ設定については，「9.2　アプリケーション統合属性ファイルによるプロパティ設定」を参照してください。

(2) Enterprise Bean のプロパティ設定項目

Enterprise Bean が含まれる J2EE アプリケーションを作成する場合およびカスタマイズする場合，必要に応じて設定してください。

Enterprise Bean のプロパティ設定項目を次に示します。

表 9-1　Enterprise Bean のプロパティ編集項目

編集項目	内容	必須／任意	固有属性ファイルの編集参照先
Enterprise Bean のリファレンス定義	J2EE アプリケーションを構成する，次の Enterprise Bean のリファレンスを定義します。 • Session Bean • Entity Bean • Message-driven Bean	○	9.3
ほかの Enterprise Bean のリファレンス定義	ほかの Enterprise Bean へのリファレンスについて定義します。	○	9.3.1
メールコンフィグレーションのリファレンス定義	メールサーバへのリファレンスについて定義します。	○	9.3.2
リソースアダプタのリファレンス定義	リソースアダプタへのリファレンスについて定義します。	○	9.3.3
リソース環境のリファレンス定義	リソース環境へのリファレンスを定義します。	○	9.3.4
Message-driven Bean のメッセージ参照定義	Message-driven Bean の Connection Factory と Destination を参照する個所に，実際の Connection Factory と Destination をマッピングします。	○	9.4
トランザクション属性の定義	トランザクション属性について定義します。	○	9.5
Entity Bean の CMP 定義	Entity Bean の永続性管理をコンテナに任せる場合に定義します。	○	9.6
CMP の設定	単一プライマリキーを使用して Entity Bean の永続性管理をコンテナに任せる場合に定義します。 複合プライマリキーを使用して Entity Bean の永続性管理をコンテナに任せる場合に定義します。	○	9.6.1
CMP1.x とデータベースのマッピング	CMP1.x Entity Bean のフィールドをデータベース上の表にマッピングします。	○	9.6.2

編集項目	内容	必須／任意	固有属性ファイルの編集参照先
CMP2.x とデータベースのマッピング	CMP2.x Entity Bean のフィールドをデータベース上の表にマッピングします。	○	9.6.3
Enterprise Bean の実行時属性の定義	Enterprise Bean の実行時の動作について設定します。必要に応じて設定してください。	◎	9.10
Stateful Session Bean の実行時プロパティの設定	Stateful Session Bean の実行時の動作についてのプロパティを設定します。	◎	9.10.1
Stateless Session Bean の実行時プロパティの設定	Stateless Bean の実行時の動作についてのプロパティを設定します。	◎	9.10.2
Entity Bean の実行時プロパティの設定	Entity Bean の実行時の動作についてのプロパティを設定します。	◎	9.10.3
Message-driven Bean の実行時プロパティの設定	Message-driven Bean の実行時の動作についてのプロパティを設定します。	◎	9.10.4
インターセプタの設定	インターセプタを使用する場合に定義します。	○	9.21

（凡例）

◎：必須。対応する Enterprise Bean が含まれる J2EE アプリケーションのカスタマイズで必ず設定する。

○：任意。J2EE アプリケーションに応じて設定する。

(3) サーブレットと JSP のプロパティ設定項目

サーブレットおよび JSP を含む J2EE アプリケーションを作成する場合に，必要に応じて設定してください。

サーブレットと JSP のプロパティ設定項目を次に示します。

表 9-2 サーブレットと JSP のプロパティ編集項目

編集項目	内容	必須／任意	固有属性ファイルの編集参照先
サーブレットと JSP のリファレンス定義	Enterprise Bean，セキュリティロール，リソース（データベースまたはメールサーバ）へのリファレンス（リソース参照）について定義します。	○	9.7
Enterprise Bean のリファレンス定義	Enterprise Bean へのリファレンスについて定義します。	○	9.7.1
メールコンフィグレーションのリファレンスの定義	メールサーバへのリファレンスについて定義します。	○	9.7.2

編集項目	内容	必須／任意	固有属性ファイルの編集参照先
リソースアダプタのリファレンス定義	リソースアダプタへのリファレンスについて定義します。	○	9.7.3
リソース環境のリファレンス定義	リソース環境リファレンスを定義します。	○	9.7.4
サーブレットと JSP のマッピング定義	サーブレットと JSP のマッピングを定義します。	○	9.8
フィルタの設定	フィルタを追加し，マッピングを定義します。	○	9.9
フィルタの追加	フィルタを追加します。	○	9.9.1
フィルタのマッピング定義	フィルタへのマッピングを定義します。	○	9.9.2
フィルタの削除	フィルタを削除します。	○	9.9.3
サーブレットと JSP の実行時属性の定義	サーブレットおよび JSP の実行時属性を定義します。	○	9.11
J2EE アプリケーションのコンテキストルート定義	J2EE アプリケーションのコンテキストルートを設定します。	◎	9.11.1
Web アプリケーション単位での同時実行スレッド数制御の定義	Web コンテナで Web アプリケーション単位での同時実行スレッド数を制御するかどうか，およびスレッド数などを設定します。	○	9.11.2
URL グループ単位での同時実行スレッド数制御の定義	URL グループ単位での同時実行スレッド数を制御するための定義をします。	○	9.11.3
URL グループ単位の実行待ちリクエスト数の監視の定義	URL グループ単位の実行待ちリクエスト数を監視するための定義をします。	○	9.11.4
デフォルトの文字エンコーディングの設定	Web アプリケーションのデフォルトの文字エンコーディングを設定します。	○	9.12
サーブレットと JSP のエラー通知の設定	サーブレット，JSP でエラーが発生した場合に，J2EE アプリケーションにエラーを通知するための設定をします。	○	9.16

（凡例）

◎：必須。サーブレットと JSP が含まれる J2EE アプリケーションのカスタマイズでは必ず設定する。

○：任意。J2EE アプリケーションに応じて設定する。

(4) J2EE アプリケーション（共通）のプロパティ設定項目

J2EE アプリケーションの構成に関係なく，必要に応じて設定してください。

表 9-3 J2EE アプリケーション（共通）のプロパティ編集項目

編集項目	内容	必須／任意	固有属性ファイルの編集参照先
JNDI 名前空間に登録される名称の参照と変更	J2EE アプリケーションの JNDI 名前空間への登録名を参照して，必要に応じて別名を付与します。	○	9.13
J2EE アプリケーション名の参照	J2EE アプリケーション名を参照するための設定をします。	○	9.13.1
Enterprise Bean 名の参照と変更	Enterprise Bean 名を参照するための設定および Enterprise Bean 名の変更をします。	○	9.13.2
CTM のスケジューリング	CTM を利用してキューのスケジューリングをするかどうかと，スケジューリング方法について設定します。	○	9.14
J2EE アプリケーション単位のスケジューリング	J2EE アプリケーション単位の CTM との連携にかかわる設定をします。	○	9.14.1
Stateless Session Bean 単位のスケジューリング	Stateless Session Bean 単位のスケジューリングの設定をします。	○	9.14.2
起動順序の設定	J2EE アプリケーションの起動順序，および J2EE アプリケーションに含まれる Enterprise Bean（EJB-JAR）やサーブレットまたは JSP（WAR）の起動順序を設定します。	○	9.15
J2EE アプリケーションの起動順序の設定	J2EE アプリケーションの起動順序を設定します。	○	9.15.1
Enterprise Bean の起動順序の設定	J2EE アプリケーションに含まれる Enterprise Bean（EJB-JAR）の起動順序を設定します。	○	9.15.2
サーブレットと JSP の起動順序の設定	J2EE アプリケーションに含まれるサーブレットまたは JSP（WAR）の起動順序を設定します。	○	9.15.3
セキュリティロールの設定	セキュリティロールを使用したユーザ管理をする場合に必要な設定です。 ユーザとロールの登録と対応づけができます。	○	9.17
ユーザの設定	ユーザを登録します。ロールとの対応づけもできます。	○	9.17.1
ロールの設定	ロールを登録します。ユーザとの対応づけもできます。	○	9.17.2
セキュリティロールのリファレンス定義	セキュリティロールへのリファレンスについて定義します。	○	9.18
Enterprise Bean のセキュリティロールリファレンスの定義	セキュリティロールを参照している個所に，実際に J2EE サーバが管理しているセキュリティロールをマッピングします。	○	9.18.1

編集項目	内容	必須／任意	固有属性ファイルの編集参照先
サーブレットと JSP のセキュリティロールリファレンスの定義	セキュリティロールへのリファレンスについて定義します。	○	9.18.2
セキュリティの定義（メソッドパーミッション）	セキュリティロールによるアクセス権を設定する場合のメソッドのパーミッションを定義します。 すべてのユーザに対してアクセス権限を設定する場合のメソッドのパーミッションを定義します。	○	9.19
セキュリティの定義（セキュリティアイデンティティ）	セキュリティの情報であるセキュリティアイデンティティとして UseCallerIdentity を設定します。 セキュリティの情報であるセキュリティアイデンティティとして Run as を設定します。	○	9.20
Enterprise Bean のセキュリティアイデンティティ	J2EE アプリケーション内で参照しているセキュリティアイデンティティ情報に，J2EE サーバが実際に管理しているユーザ ID をマッピングします。	○	9.20.1
サーブレットと JSP のセキュリティアイデンティティ	セキュリティアイデンティティを定義します。	○	9.20.2

（凡例）

○：任意。J2EE アプリケーションに応じて設定する。

9.2　アプリケーション統合属性ファイルによるプロパティ設定

サーバ管理コマンドでアプリケーション統合属性ファイルを取得して，固有の属性ファイルの内容を一括して編集できます。

アプリケーション統合属性ファイルに含まれる，固有の属性ファイルの要素を次に示します。

- アプリケーション属性ファイル
  （<hitachi-application-property></hitachi-application-property>）
- EJB-JAR 属性ファイル
  （<hitachi-ejb-jar-property></hitachi-ejb-jar-property>）
- Session Bean 属性ファイル
  <hitachi-session-bean-property></hitachi-session-bean-property>
- Entity Bean 属性ファイル
  <hitachi-entity-bean-property></hitachi-entity-bean-property>
- Message-driven Bean 属性ファイル
  <hitachi-message-bean-property></hitachi-message-bean-property>
- WAR 属性ファイル
  <hitachi-war-property></hitachi-war-property>
- フィルタ属性ファイル
  <hitachi-filter-property></hitachi-filter-property>
- サーブレット属性ファイル
  <hitachi-servlet-property></hitachi-servlet-property>
- Connector 属性ファイル
  <hitachi-connector-property></hitachi-connector-property>

アプリケーション統合属性ファイルの詳細については，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（アプリケーション／リソース定義）」の「3.1 アプリケーション統合属性ファイル」を参照してください。

アプリケーション統合属性ファイルによる，J2EE アプリケーションのプロパティ設定手順を次に示します。

（1） アプリケーション統合属性ファイルの取得

アプリケーション統合属性ファイルを取得するために，アプリケーション統合属性ファイルパスを指定して，cjgetappprop コマンドを実行します。

実行形式

```
cjgetappprop ＜サーバ名称＞ -name ＜J2EEアプリケーション名＞ -type all -c ＜アプリケーション統合属性ファイルパス＞
```

実行例

```
cjgetappprop Myserver -name App1 -type all -c App1prop.xml
```

(2) プロパティ設定項目の編集

上記 (1) で出力されたアプリケーション統合属性ファイルをテキストエディタで編集します。J2EE アプリケーションのプロパティ編集項目については，「9.1　J2EE アプリケーションのプロパティ設定の概要」を参照してください。アプリケーション統合属性ファイルの編集項目は，固有の属性ファイルの編集項目と同じです。

(3) アプリケーション属性の設定

アプリケーション統合属性ファイルで指定した項目をアプリケーション属性に反映するために，cjsetappprop コマンドを実行します。

実行形式

```
cjsetappprop ＜サーバ名称＞ -name ＜J2EEアプリケーション名＞ -type all -c ＜アプリケーション統合属性ファイルパス＞
```

実行例

```
cjsetappprop MyServer -name App1 -type all -c App1Prop.xml
```

9.3　Enterprise Bean のリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean のリファレンス（リソース参照）を定義します。Enterprise Bean には，次の種類があります。

- Session Bean
- Entity Bean
- Message-driven Bean

それぞれの Enterprise Bean には，次に示す 3 種類のリファレンスがあります。

- Enterprise Bean リファレンス
  ほかの Enterprise Bean 呼び出しのための参照です。
- リソースリファレンス
  リソースへの参照です。メールリファレンス，リソースアダプタリファレンス，およびリソース環境リファレンスがあります。
- セキュリティロールリファレンス
  Enterprise Bean のメソッド呼び出しのアクセス制御をするために使用するロール名称の参照です。

この節では，Enterprise Bean リファレンスとリソースリファレンスについて説明します。セキュリティロールのリファレンスについては，「9.18　セキュリティロールのリファレンス定義」を参照してください。

なお，プロパティの設定項目については，次の個所を参照してください。

- 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「3.4.1　Session Bean 属性ファイルの指定内容」
- 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「3.5.1　Entity Bean 属性ファイルの指定内容」
- 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「3.6.1　MessageDrivenBean 属性ファイルの指定内容」

9.3.1　ほかの Enterprise Bean のリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean が，ほかの Enterprise Bean を呼び出している場合，リファレンスを解決するためのプロパティを設定します。

（1）編集する属性ファイル

J2EE アプリケーションを構成する Enterprise Bean の種類に対応する属性ファイルを編集します。

- Session Bean 属性ファイル
- Entity Bean 属性ファイル

- Message-driven Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Enterprise Bean の属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -name ＜J2EEアプリケーション名> -type ejb -resname ＜EJB-JAR表示名>/＜Enterprise Bean表示名> -c ＜Enterprise Beanの属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder/adder_eb -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，Enterprise Bean の属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -name ＜J2EEアプリケーション名> -type ejb -resname ＜EJB-JAR表示名>/＜Enterprise Bean表示名> -c ＜Enterprise Beanの属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder/adder_eb -c C:¥home¥adder_ejb.xml
```

(3) 編集する属性設定項目

参照する Enterprise Bean のアクセスタイプには，次の 2 種類があります。

参照する Enterprise Bean のアクセスタイプ	対応するタグ名
リモートインタフェース	<ejb-ref>
ローカルインタフェース	<ejb-local-ref>

注　ビジネスインタフェース（<business-local> および <business-remote>）は，アノテーションで設定しているため変更できません。

それぞれの設定項目について説明します。

(a) リモートインタフェースの Enterprise Bean のリファレンス定義

リモート Enterprise Bean のリファレンス設定項目（<ejb-ref>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
Enterprise Bean 参照名	○	<ejb-ref-name>
参照する Enterprise Bean の種別	○	<ejb-ref-type>
リモートホームインタフェース	○	<home>
リモートインタフェース	○	<remote>
リンク先の <ejb-name>	△	<ejb-link>

（凡例） ○：必須 △：任意

(b) ローカルインタフェースの Enterprise Bean のリファレンス定義

ローカル Enterprise Bean のリファレンス設定項目（<ejb-local-ref>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
Enterprise Bean 参照名	○	<ejb-ref-name>
参照する Enterprise Bean の種別	○	<ejb-ref-type>
ローカルホームインタフェース	○	<local-home>
ローカルインタフェース	○	<local>
リンク先の <ejb-name>	△	<ejb-link>

（凡例） ○：必須 △：任意

(4) 注意事項

- リモートインタフェースの Enterprise Bean を呼び出す場合には，別アプリケーションにある Enterprise Bean をマッピング先として，「リンク先の <ejb-name>」（<ejb-link>）に設定できます。
  マッピング先の Enterprise Bean が Cosminexus 上で動作している場合，ネーミングの切り替え機能を利用して，次に示すような形式でマッピング先を指定できます。
  指定形式

```
corbaname:: ＜名前空間のホスト名＞：＜名前空間のポート番号＞# ＜ EJB ホームオブジェクトリファレンスの JNDI 名＞
```

＜名前空間のホスト名＞
マッピング先 Enterprise Bean が使用する名前空間が動作する，ホスト名称を指

定します。

＜名前空間のポート番号＞

マッピング先 Enterprise Bean が使用する名前空間が動作する，ポート番号を指定します。

＜ EJB ホームオブジェクトリファレンスの JNDI 名＞

次に示す形式に従って，JNDI 名を指定します。

```
HITACHI_EJB/SERVERS/＜サーバ名称＞/EJB/＜J2EE APP名称＞/＜Enterprise Bean名称＞
```

＜サーバ名称＞：J2EE サーバのサーバ名称

＜ J2EE APP 名称＞：J2EE アプリケーションのルックアップ名称

＜ Enterprise Bean 名称＞：Enterprise Bean のルックアップ名称

指定例

```
corbaname::MyHost:900#HITACHI_EJB/SERVERS/MyServer/EJB/
MyApplication/MyBean
```

- ネーミングサービスがローカルホスト上で動作する場合，J2EE サーバ用の usrconf.properties ファイルの ejbserver.naming.host キーには，「localhost」の文字列を使用しないで，マシン名または IP アドレスを指定してください。

9.3.2 メールコンフィグレーションのリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean が，メールコンフィグレーションを参照している場合，リファレンスを定義するためのプロパティを設定します。

(1) 編集する属性ファイル

編集する属性ファイルについては，「9.3.1 ほかの Enterprise Bean のリファレンス定義」の「(1) 編集する属性ファイル」を参照してください。

(2) 編集する属性ファイルの取得と属性の設定

編集する属性ファイルの取得と属性の設定については，「9.3.1 ほかの Enterprise Bean のリファレンス定義」の「(2) 編集する属性ファイルの取得と属性の設定」を参照してください。

(3) 編集する属性設定項目

メールコンフィグレーションのリファレンス設定項目（<resource-ref>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
リソース参照名	○	<res-ref-name>
リソースタイプ	○	<res-type>
リソース認証方式	○	<res-auth>
リソース共有可否	△	<res-sharing-scope>
リンク先のメールコンフィグレーション名	△	<linked-to>

（凡例） ○：必須 △：任意

9.3.3 リソースアダプタのリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean が，リソースアダプタを参照している場合，リファレンスを解決するためのプロパティを設定します。

(1) 編集する属性ファイル

編集する属性ファイルについては，「9.3.1 ほかの Enterprise Bean のリファレンス定義」の「(1) 編集する属性ファイル」を参照してください。

(2) 編集する属性ファイルの取得と属性の設定

編集する属性ファイルの取得と属性の設定については，「9.3.1 ほかの Enterprise Bean のリファレンス定義」の「(2) 編集する属性ファイルの取得と属性の設定」を参照してください。

(3) 編集する属性設定項目

リソースアダプタのリファレンス設定項目（<resource-ref>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
リソース参照名	○	<res-ref-name>
リソースタイプ	○	<res-type>
リソース認証方式	○	<res-auth>
リソース共有可否	△	<res-sharing-scope>
リンク先のリソースアダプタ表示名※	△	<linked-to>

（凡例） ○：必須 △：任意

注※ クラスタコネクションプールを使用している場合，ルートリソースアダプタの表示名を指定してください。メンバリソースアダプタは指定できません。

9.3.4 リソース環境のリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean のリソース環境のリファレンスを解決するためのプロパティを設定します。

(1) 編集する属性ファイル

編集する属性ファイルについては,「9.3.1 ほかの Enterprise Bean のリファレンス定義」の「(1) 編集する属性ファイル」を参照してください。

(2) 編集する属性ファイルの取得と属性の設定

編集する属性ファイルの取得と属性の設定については,「9.3.1 ほかの Enterprise Bean のリファレンス定義」の「(2) 編集する属性ファイルの取得と属性の設定」を参照してください。

(3) 編集する属性設定項目

リソース環境のリファレンス設定項目（<resource-env-ref>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
リソース環境変数名	○	<resource-env-ref-name>
リソース環境変数値のタイプ	○	<resource-env-ref-type>
リンク先キュー情報	△	<linked-queue>
リソースアダプタの表示名	△	<resource-adapter>
キュー名	△	<queue>
リンク先の JavaBeans リソース表示名	△	<linked-to>

（凡例） ○：必須 △：任意

9.4 Message-driven Bean のメッセージ参照定義

Message-driven Bean のメッセージ参照を定義します。

(1) 編集する属性ファイル

Message-driven Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Message-driven Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Message-driven Bean表示名> -c <Message-driven Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name message -type ejb -resname message/MyMessage -c C:¥home¥MyMessageBean.xml
```

● 属性の設定

次に示すコマンドを実行して，Message-driven Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Message-driven Bean表示名> -c <Message-driven Bean属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name message -type ejb -resname message/MyMessage -c C:¥home¥MyMessageBean.xml
```

(3) 編集する属性設定項目

メッセージ参照の設定項目（<message-ref>）を次に示します。

項目	必須	対応するタグ名
Connection Factory 名	○	<connection-factory>

項目	必須	対応するタグ名
リソースアダプタ表示名	○	<connction-destination> - <resource-adapter>
キューの表示名※	○	<connction-destination> - <queue>

（凡例） ○：必須
注※ JMS のキューを複数の異なる Message-driven Bean で共有することは推奨しません。

プロパティの設定項目については，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「3.6.1 MessageDrivenBean 属性ファイルの指定内容」を参照してください。

9.5　トランザクション属性の定義

コンテナによるトランザクションの管理方法を定義します。

トランザクション属性は Enterprise Bean 単位，インタフェース単位，メソッド単位にそれぞれ指定できます。指定がない場合は上位の単位で指定されたトランザクション属性が有効になります。

● Enterprise Bean ごとの適用
　割り当てたトランザクション属性を，Enterprise Bean 内のメソッドに適用します。

● インタフェースごとの適用
　割り当てたトランザクション属性を，インタフェース内のメソッドに適用します。

● メソッドごとの適用
　割り当てたトランザクション属性を，一つのメソッドに適用します。

(1) 編集する属性ファイル

次の属性ファイルのうち，トランザクション属性を設定する Enterprise Bean の種類に対応する属性ファイルを編集します。

- Session Bean 属性ファイル
- Entity Bean 属性ファイル
- Message-driven Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Enterprise Bean の属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Beanの属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder/
adder_eb -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，Enterprise Bean の属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -name ＜J2EEアプリケーション名＞ -type ejb -resname ＜EJB-JAR表示名>/＜Enterprise Bean表示名＞ -c ＜Enterprise Beanの属性ファイルパス＞
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder/adder_eb -c C:¥home¥adder_ejb.xml
```

(3) 編集する属性設定項目

コンテナトランザクション属性（<container-transaction>）の設定項目を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
メソッドの説明	△	<method> - <description>
インタフェース種別	△	<method> - <method-intf>
メソッド名	○	<method> - <method-name>
トランザクション属性	○	<trans-attribute>

（凡例） ○：必須 △：任意

プロパティの設定項目については，次の個所を参照してください。

- 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「3.4.1 Session Bean 属性ファイルの指定内容」
- 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「3.5.1 Entity Bean 属性ファイルの指定内容」
- 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「3.6.1 MessageDrivenBean 属性ファイルの指定内容」

トランザクション属性の指定とトランザクション管理の動作を，次に示します。

- NotSupported
  コンテナ管理トランザクション内でメソッドを実行しません。
- Supports
  メソッドのトランザクションの処理は場合によって異なります。トランザクション内およびトランザクション外で正しく実行するメソッドに対してだけ，この属性を使用してください。
- Required
  コンテナはメソッドを常にトランザクション内で実行します。
- RequiresNew

メソッドは，常に新しいトランザクションで実行します。

- Mandatory
  メソッドは，常にクライアントのトランザクションを使用しなければなりません。つまり，クライアントはトランザクション内でメソッドを呼び出さなければなりません。
- Never
  メソッドはトランザクション内で実行できません。つまり，クライアントは，トランザクションの外でメソッドを呼び出さなければなりません。

(4) 注意事項

● Enterprise Bean の種類によって設定できるトランザクション属性が異なります。設定できるトランザクション属性およびトランザクション管理の動作については，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)」を参照してください。

● Session Bean のプロパティ（<transaction-type>）で，Session Bean が自身のトランザクションを管理するよう指定されている場合，この設定でトランザクション属性を変更できません。

● Message-driven Bean の場合は，onMessage のメソッドだけ設定できます。

9.6 Entity Bean の CMP 定義

Entity Bean の CMP 定義をします。CMP を定義すると，Enterprise Bean である Entity Bean の永続性管理をコンテナに任せることができます。

9.6.1 CMP の設定

(1) 編集する属性ファイル

Entity Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Entity Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Entity Bean表示名> -c <Entity Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname account/MyAccoub -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，Entity Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Entity Bean表示名>-c <Entity Bean属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname account/MyAccoub -c C:¥home¥adder_ejb.xml
```

(3) 編集する属性設定項目

CMP の定義は，プライマリキーが単一プライマリキーの場合と複合プライマリキーの場合で，設定方法が異なります。

(a) 単一プライマリキーの場合の属性設定

プライマリキーが単一プライマリキーの場合のプロパティ項目を設定します。

永続性管理種別（<persistence-type>）を確認します。CMP Entity Bean の場合は「Container」が設定されています。

単一プライマリキーのプロパティ設定項目を次に示します。

項目	必須	対応するタグ名
プライマリキーのクラス※1	○	<prim-key-class>
リエントラント可否	○	<reentrant>
永続性管理フィールドの説明	△	<cmp-field> - <description>
永続性管理フィールドのフィールド名※2	△	<cmp-field> - <field-name>
プライマリキーフィールドのフィールド名	△	<primkey-field>

（凡例） ○：必須 △：任意
注※1 この Entity Bean のプライマリキーを含むクラスまたはインタフェースを入力します。プライマリキークラスは，java.lang.Object クラス，またはコンテナ管理フィールドと同じクラスもしくはインタフェースでなければなりません。
注※2 通常は不要です。追加する場合はあらかじめ Entity Bean のクラス定義にも追加しておく必要があります。

プロパティの設定項目については，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「3.5.1 Entity Bean 属性ファイルの指定内容」を参照してください。

(b) 複合プライマリキーの場合の属性設定

プライマリキーが複合プライマリキーの場合のプロパティ項目を設定します。

永続性管理種別（<persistence-type>）を確認します。CMP Entity Bean の場合は「Container」が設定されています。

複合プライマリキーのプロパティ設定項目を次に示します。

項目	必須	対応するタグ名
プライマリキーのクラス※1	○	<prim-key-class>
リエントラント可否	○	<reentrant>
永続性管理フィールドの説明	△	<cmp-field> - <description>
永続性管理フィールドのフィールド名※2	△	<cmp-field> - <field-name>

（凡例） ○：必須 △：任意
注※1 この Entity Bean のプライマリキーを含むクラスまたはインタフェースを入力します。プライマリキークラスは，java.lang.Object クラス，またはコンテナ管理フィールドと同じクラスもしくはインタフェースでなければなりません。
注※2 通常は不要です。追加する場合はあらかじめ Entity Bean のクラス定義にも追加しておく必要があります。

プロパティの設定項目については，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「3.5.1 Entity Bean 属性ファイルの指定内容」を参照してください。

9.6.2 CMP1.x とデータベースのマッピング

CMP1.x Entity Bean のフィールドをデータベース上の表にマッピングします。

(1) 編集する属性ファイル

Entity Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Entity Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Entity Bean表示名>-c <Entity Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname account/MyAccoub -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，Entity Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Entity Bean表示名>-c <Entity Bean属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname account/MyAccoub -c C:¥home¥adder_ejb.xml
```

(3) 編集する属性設定項目

CMP1.x Entity Bean のフィールドをデータベース上の表にマッピングする（<cmp-map>）プロパティ設定項目を次に示します。

項目	必須	対応するタグ名
リソースアダプタの表示名※1	○	<datasource-name>
データベースのカタログ名	△	<catalog-name>
データベースのスキーマ名	△	<schema-name>
データベースのテーブル名	○	<table-name>
データベースへの書き込み許可／禁止	○	<read-only-access>
トランザクション遮断レベル※2	△	<transaction-isolation>
データベース書き込みデータ照合方法	△	<concurrency-protection>
フィールドとテーブルのカラムとのマッピング情報※3	○	<field-impl>
finder メソッドの検索条件※4	○	<finder-impl>

（凡例） ○：必須　△：任意

注※1

クラスタコネクションプールを使用している場合，ルートリソースアダプタの表示名を指定してください。メンバリソースアダプタは指定できません。

注※2

使用できるトランザクション遮断レベル（<transaction-isolation>）の値は，データベースおよび JDBC ドライバでサポートされるオプションによって異なります。

注※3

プライマリキーに対してデータベーステーブルの列を設定します。同じデータベーステーブルのフィールドのマッピングにも使用されます。

EntityBean のフィールド名（<field-name>）の値は変更できません。EntityBean のフィールド名（<field-name>）のマッピング先のデータベースの列（<column-name>）を設定してください。

フィールドとテーブルのカラムとのマッピング情報（<field-impl>）は，次の項目で構成されています。

項目	対応するタグ名
EntityBean のフィールド名	<field-name>
テーブルのカラム名	<column-name>

注※4

finder メソッドのメソッド名（<method-name>）の値は変更できません。finder メソッドのメソッド名（<method-name>）のテーブルの検索条件（<where-clause>）を設定してください。

Enterprise Bean 内の finder メソッド情報（<finder-impl>）は，次の項目で構成されていま

す。

項目	対応するタグ名
finder メソッドのメソッド名	<method-name>
テーブルの検索条件	<where-clause>

プロパティの設定項目については，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「3.5.1　Entity Bean 属性ファイルの指定内容」を参照してください。

(4) 注意事項

- Entity Bean の CMP フィールドをデータベースの複数の表にマッピングする機能は提供していません。
- マッピングで利用する DB Connector の定義で指定されているユーザと，CMP のマッピング先となるデータベースの表の所有者が異なる場合，DB Connector の定義で指定されているユーザには次に示す権限が必要です。
  マッピング時：SELECT
  実行時：SELECT，INSERT，UPDATE，DELETE

9.6.3 CMP2.x とデータベースのマッピング

CMP2.x Entity Bean のフィールドをデータベース上のテーブルにマッピングします。

また，ここでは，CMP を定義した二つの CMP2.x Entity Bean の間に関連（CMR）を定義する方法についても説明します。

CMP2.x とデータベースのマッピング手順を，次に示します。

1. CMP2.x Entity Bean 間に CMR の関係がある場合，CMP2.x Entity Bean 間に CMR を定義します。
2. CMP2.x Entity Bean のフィールドをデータベース上のテーブルにマッピングします。
3. CMP2.x Entity Bean に対して cjgencmpsql コマンドを実行して SQL 文を生成します。

(1) CMR の定義

CMP を定義した二つの CMP2.x Entity Bean の間に CMR の関係がある場合，CMP2.x Entity Bean 間に CMR を定義します。

(a) 編集する属性ファイル

EJB-JAR 属性ファイル

(b) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して EJB-JAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名> -c <EJB-JAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，EJB-JAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名> -c <EJB-JAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder -c C:¥home¥adder_ejb.xml
```

注意事項

開始状態のアプリケーションに含まれる EJB-JAR 属性ファイルの取得はできますが，定義された関連情報の反映はできません。

(c) 編集する属性設定項目

二つの CMP2.x Entity Bean 間の関係を定義する（<relationships>・<ejb-relation>）プロパティ設定項目を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
関連定義名	○	<ejb-relation-name>
Enterprise Bean（EJB1）情報	○	<ejb1>
Enterprise Bean（EJB2）情報	○	<ejb2>

（凡例） ○：必須 △：任意

プロパティの設定項目については，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「3.3.1　EJB-JAR 属性ファイルの指定内容」を参照してください。

Enterprise Bean（EJB1）情報（<ejb1>）と Enterprise Bean（EJB2）情報（<ejb2>）について，それぞれ，設定側の Enterprise Bean と相手側の Enterprise Bean の関連についての項目を指定します。

項目	必須	対応するタグ名
説明	△	<description>
関連のロール名	△	<ejb-relationship-role-name>
設定側の多重度	○	<multiplicity>
カスケードデリートの設定	○	<cascade-delete>
設定側の <ejb-name>	○	<ejb-name>
CMR フィールドの名称	○	<cmr-field-name>
CMR フィールドの型	△	<cmr-field-type>

（凡例）　○：必須　△：任意

プロパティの設定項目については，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「3.3.1　EJB-JAR 属性ファイルの指定内容」を参照してください。

(2) CMP2.x とデータベースのマッピング

CMP2.x Entity Bean のフィールドをデータベース上のテーブルにマッピングします。

(a) 編集する属性ファイル

編集する属性ファイルについては，「9.6.2　CMP1.x とデータベースのマッピング」の「(1)　編集する属性ファイル」を参照してください。

(b) 編集する属性ファイルの取得と属性の設定

編集する属性ファイルの所得と属性の設定については，「9.6.2　CMP1.x とデータベースのマッピング」の「(2)　編集する属性ファイルの取得と属性の設定」を参照してください。

(c) 編集する属性設定項目

CMP2.x Entity Bean のフィールドをデータベース上の表にマッピングする（<cmp-map>）プロパティ設定項目を次に示します。

項目	必須	対応するタグ名
リソースアダプタの表示名※1	○	<datasource-name>

項目	必須	対応するタグ名
データベースのカタログ名	△	<catalog-name>
データベースのスキーマ名	△	<schema-name>
データベースのテーブル名	○	<table-name>
データベースへの書き込み許可／禁止	○	<read-only-access>
トランザクション遮断レベル※2	△	<transaction-isolation>
データベース書き込みデータ照合方法	△	<concurrency-protection>
フィールドとテーブルのカラムとのマッピング情報※3	○	<field-impl>

（凡例） ○：必須　△：任意

注※1

クラスタコネクションプールを使用している場合，ルートリソースアダプタの表示名を指定してください。メンバリソースアダプタは指定できません。

注※2

使用できるトランザクション遮断レベル（<transaction-isolation>）の値は，データベースおよびJDBCドライバでサポートされるオプションによって異なります。

注※3

プライマリキーに対してデータベーステーブルの列を設定します。同じデータベーステーブルのフィールドのマッピングにも使用されます。

EntityBeanのフィールド名（<field-name>）の値は変更できません。EntityBeanのフィールド名（<field-name>）のマッピング先のデータベースの列（<column-name>）を設定してください。

フィールドとテーブルのカラムとのマッピング情報（<field-impl>）は，次の項目で構成されています。

項目	対応するタグ名
EntityBeanのフィールド名	<field-name>
テーブルのカラム名	<column-name>

プロパティの設定項目については，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「3.5.1　Entity Bean属性ファイルの指定内容」を参照してください。

(d) 注意事項

- Entity BeanのCMPフィールドをデータベースの複数の表にマッピングする機能は提供していません。
- マッピングで利用するDB Connectorの定義で指定されているユーザと，CMPのマッピング先となるデータベースの表の所有者が異なる場合，DB Connectorの定義で指定されているユーザには次に示す権限が必要です。

  マッピング時：SELECT

実行時：SELECT，INSERT，UPDATE，DELETE

- CMP フィールドのマッピングが終わっていない場合，およびアプリケーションが開始状態の場合は，SQL を生成できません。
- CMR の関係を結ぶ二つの Bean のテーブルは同じデータベース製品のデータベース上になくてはいけません。

(3) SQL 文の生成

finder/select メソッドが実行されたときに使用される SQL 文を生成します。

CMP2.x Entity Bean 間に CMR の関係がない場合

定義された EJB QL（<query>）を基に SQL 文を生成します。

CMP2.x Entity Bean 間に CMR の関係がある場合

SQL 文の生成をする前に，まず，すべての CMP2.x Entity Bean についてフィールドを表の列にマッピングします。すべての Bean についてマッピング操作の終了後，すべての CMP2.x Entity Bean について SQL 文を生成してください。Entity Bean に finder/select メソッドがない場合でも同様に SQL 文を生成してください。

なお，Entity Bean 間に CMR の関係がある場合は，finder/select メソッドが実行されたときに使用される SQL 文のほかに，CMR 用の表を操作するための SQL 文も生成されます。一度 SQL を生成したあとに CMR の設定を変更した場合は，開始する前に，変更した CMR に関する Entity Bean について SQL 文を生成してください。CMR 用の表の詳細については，「(4) CMR 用の表」を参照してください。

次に示すコマンドを実行して SQL 文を生成します。

- J2EE アプリケーションに含まれるすべての CMP2.x Entity Bean の SQL 文を生成する場合

実行形式

```
cjgencmpsql ［<サーバ名称>］ -name J2EEアプリケーション名
```

実行例

```
cjgencmpsql MyServer -name App1
```

- J2EE アプリケーションに含まれる，特定の CMP2.x Entity Bean の SQL 文を生成する場合

実行形式

```
cjgencmpsql ［<サーバ名称>］ -name J2EEアプリケーション名 -resname ＜EJB-JARの表示名
/Entity Beanの表示名＞
```

実行例

```
cjgencmpsql MyServer -name App1 -resname EjbJar1/Ejb1
```

cjgencmpsql コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjgencmpsql（CMP2.x Entity Bean 用 SQL 文の生成）」を参照してください。

（4） CMR 用の表

ここでは，CMR 用の表について説明します。

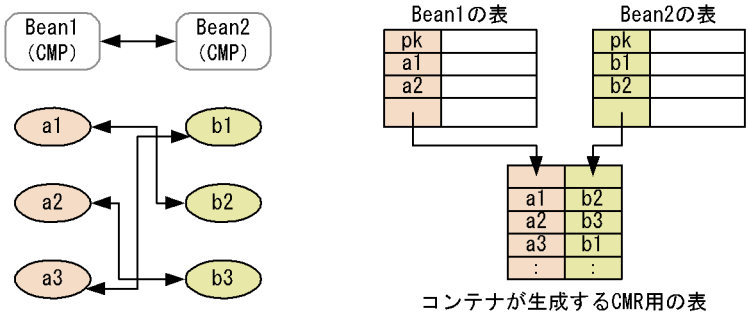
なお，以降の説明では，EJB-JAR の <ejb1> - <ejb-name> で設定されている Entity Bean を「Bean1」，<ejb2> - <ejb-name> で設定されている Entity Bean を「Bean2」と呼びます。

（a） CMR 用の表の概要

EJB コンテナでは，CMP2.x Entity Bean の間に CMR の関係があった場合，データベース上に表を作り，CMR の状態の保持のために利用します。

次の図に示すように CMR 用の表には関係を結ぶ二つの Entity Bean のプライマリキーが格納されます。

図 9-1　CMR 用の表（1 対 1，双方向の関係の場合）



（b） CMR 用の表の生成と削除

CMR を含むアプリケーションを開始すると，CMR 用の表が生成されます。生成される場所は Bean1 の表があるスキーマです。この表はアプリケーションの停止時に削除されます。表の生成，削除，操作のための SQL は，アプリケーションをカスタマイズするときのデータベースマッピングのあとで生成された SQL を使用します。

アプリケーションが開始状態のままサーバを停止させた場合は CMR 用の表はデータベース上に残ります。また，アプリケーションが開始状態のままサーバが再起動された場合は，CMR 用の表がデータベース上にあるかどうかをチェックします。アプリケーションの開始時およびサーバ起動時の CMR 用の表生成処理を次の表に示します。

表 9-4 アプリケーションの開始時およびサーバ起動時の CMR 用の表生成処理

状態	作成する表と同名の表がデータベース上にあるかどうか	同名の表がある場合，列の数，名前，JDBC 型が同じかどうか	動作	出力メッセージ ID
開始時※1	なし	該当しない	表を生成し，利用します。	KDJE43007-I
	あり	同じまたは同じでない	開始を中断します。※2	KDJE43003-E
J2EE サーバ起動時	なし	該当しない	開始を中断します。※3	KDJE43008-E
	あり	同じ	開始時に CMR 用として作ったものであると判断し，利用します。	KDJE43006-I
		同じでない	開始を中断します。※3	KDJE43008-E

注※ 1

CMR を含むアプリケーションの開始時のオプションを特に設定していないデフォルトの場合です。開始時のオプションについては，「(e) 障害発生時の CMR 用の表の回復」を参照してください。

注※ 2

同じ名前の表がすでにデータベース上にあるとアプリケーションの開始に失敗します。すでにある表が不要の場合，データベース製品で提供しているツールを使用して手動で削除するか，SQL 生成を再度実行してください。SQL 生成を再度実行すると，データベーススキーマ上の表と重複しない名前が表に付けられ，SQL が生成し直されます（SQL 生成は，EJB-JAR 内のすべての EJB のすべてのメソッドについて行ってください。これを行わないと表名の変更がすべての SQL に反映されないおそれがあります）。そのあと，再度アプリケーションを開始してください。

注※ 3

対象アプリケーションが開始されていない状態で J2EE サーバが起動します。

(c) CMR 用の表の命名規則

CMR 用の表の名前は次のようになります。

● 双方向の関係の場合

［Bean1 の <ejb-name>］_［Bean1 の <cmr-field-name>］_［Bean2 の <ejb-name>］_［Bean2 の <cmr-field-name>］

● 単方向の関係の場合（方向は Bean1 から Bean2 とします）

［Bean1 の <ejb-name>］_［Bean1 の <cmr-field-name>］_［Bean2 の <ejb-name>］

ただし，この方法で決めた名前が 29 文字以上になった場合は，28 文字になるように切ります。28 文字になるように切った名前がデータベース上の既存の表の名前や同じ

EJB-JAR 内のほかの CMR 用の表の名前と重なる場合は，0 ～ 99 の番号を末尾に付けて区別します。

(d) CMR についての注意事項

- ユーザは Bean1 の表があるスキーマで，CREATE TABLE を行う権限が必要です。
- データベースや OS の種別によってはプライマリキーのサイズの制限で，CMR 用の表の生成に失敗することがあります。図 9-1 に示すように，CMR 用の表には Entity Bean のプライマリキーを格納するので，Entity Bean のプライマリキーのサイズを調整してください。
- アプリケーションの停止時に CMR 用の表は削除されるため，それまでに結んだ relation の情報が失われてしまいます。また，CMR 用の表を手動で削除した場合も relation の情報は失われます。
- CMR 用の表の名前を付ける場合に 0 ～ 99 の 100 種類の番号を使い切ったときは，CMR 用の表が生成できません。データベース上の不要な表を削除してその名前を使用できるようにするか，または「(c)　CMR 用の表の命名規則」を参考にして異なる名前を付けられるようにしてください。
- relation を結ぶ Entity Bean のプライマリキーはデータベースの表のプライマリキーにできる型へマッピングしてください。
- CMR を含むアプリケーションの停止時に CMR 用の表は削除されますが，データベースへアクセスできないような障害発生などで CMR 用の表の削除に失敗する場合があります。この場合は，表がデータベース上に残るため，不要なときはデータベース製品で提供しているツールを使用して手動で削除してください。

(e) 障害発生時の CMR 用の表の回復

CMR を含むアプリケーションが開始され，運用していた状態で保守などのために J2EE サーバを停止，起動したときに障害が発生すると，CMR を含むアプリケーションが開始された状態で立ち上がらないことが考えられます。障害を解決し，再度 CMR を含むアプリケーションを開始しようとしても，表 9-4 に示されるように作成する表と同名の表がデータベース上にある場合，開始ができません（アプリケーション間で表の共有を避けるため）。

ここで，SQL 生成を再度実行すると，新しい CMR 用の表名を使用した SQL が生成され，新しい CMR 用の表を使用するように開始ができます。しかし，J2EE サーバを停止する以前に使用していた関係を継続させたい場合，データベースに残っている表を使うように開始する必要があります。

usrconf.properties ファイルの ejbserver.ejb.cmp20.cmr.use.existing_table キーは，アプリケーションの起動障害発生時，それまで使用していた関係の情報を回復させるためのオプションです。このキーに true を指定すると，データベース既存の表を使用するように開始できます。このキーに何も指定しない，または false を指定した場合は，表 9-4 の

動作をします。このオプションを使用して，既存の表を使用する場合の手順を次に示します。

1. **障害発生前に使用していた** CMR **用の表がデータベース上に残っていることを，データベース製品で提供しているツールなどを使用して確認してください。**
2. J2EE **サーバを停止します（アプリケーションが開始状態で立ち上がることに失敗した原因の対処をしてください）。**
3. usrconf.properties **ファイルに** ejbserver.ejb.cmp20.cmr.use.existing_table=true **を設定します。**
4. J2EE **サーバを起動します。**
5. **開始に失敗した** CMR **を含むアプリケーションを再度開始します。**

   注意事項

   ここで SQL 生成を再実行しないでください。再実行すると新しいテーブル名で SQL が生成され，以前の表が使用できなくなります。
6. J2EE **サーバを停止します。**
7. usrconf.properties **ファイルに** ejbserver.ejb.cmp20.cmr.use.existing_table=false **を設定するか，またはこのオプションを設定しない状態に戻します。**
8. **再度** J2EE **サーバを起動します。**

9.7 サーブレットと JSP のリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）のリファレンス（リソース参照）を定義します。次に示すリファレンスがあります。

- Enterprise Bean リファレンス
  Enterprise Bean 呼び出しのための参照です。
- リソースリファレンス
  データベースまたはメールサーバへの参照です。メールリファレンス，リソースアダプタリファレンス，およびリソース環境リファレンスがあります。

9.7.1 Enterprise Bean のリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）が Enterprise Bean を呼び出している場合，リファレンスを解決するためのプロパティを設定します。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder -c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder -c
C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

Enterprise Bean が，ほかの Enterprise Bean を呼び出している場合のプロパティ設定項目と同じです。「9.3.1　ほかの Enterprise Bean のリファレンス定義」の「(3)　編集する属性設定項目」を参照してください。

9.7.2 メールコンフィグレーションのリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）が，メールコンフィグレーションを参照している場合，リファレンスを定義するためのプロパティを設定します。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

編集する属性ファイルの所得と属性の設定については，「9.7.1　Enterprise Bean のリファレンス定義」の「(2)　編集する属性ファイルの取得と属性の設定」を参照してください。

(3) 編集する属性設定項目

Enterprise Bean がメールコンフィグレーションを参照している場合のプロパティ設定項目と同じです。「9.3.2　メールコンフィグレーションのリファレンス定義」の「(3)　編集する属性設定項目」を参照してください。

9.7.3 リソースアダプタのリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）が，リソースアダプタを参照している場合，リファレンスを定義するためのプロパティを設定します。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

編集する属性ファイルの所得と属性の設定については，「9.7.1　Enterprise Bean のリファレンス定義」の「(2)　編集する属性ファイルの取得と属性の設定」を参照してください。

(3) 編集する属性設定項目

Enterprise Bean がリソースアダプタを参照している場合のプロパティ設定項目と同じです。「9.3.3 リソースアダプタのリファレンス定義」の「(3) 編集する属性設定項目」を参照してください。

9.7.4 リソース環境のリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）がリソース環境を参照している場合，リファレンスを定義するためのプロパティを設定します。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

編集する属性ファイルの所得と属性の設定については，「9.7.1 Enterprise Bean のリファレンス定義」の「(2) 編集する属性ファイルの取得と属性の設定」を参照してください。

(3) 編集する属性設定項目

Enterprise Bean がリソース環境を参照している場合のプロパティ設定項目と同じです。「9.3.4 リソース環境のリファレンス定義」の「(3) 編集する属性設定項目」を参照してください。

9.8 サーブレットと JSP のマッピング定義

サーブレットおよび JSP のマッピングを定義します。

(1) 編集する属性ファイル

サーブレット属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してサーブレット属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名>/<サーブレットおよびJSPの表示名> -c <サーブレット属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder/adder_sv -c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，サーブレット属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名>/<サーブレットおよびJSPの表示名> -c <サーブレット属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder/adder_sv -c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

項目	必須	対応するタグ名
URL パターン	○	<url-pattern>

（凡例） ○：必須

プロパティの設定項目については，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「3.9.1 サーブレット属

性ファイルの指定内容」を参照してください。

9.9 フィルタの設定

フィルタの設定方法について説明します。フィルタを設定するには，フィルタを WAR ファイルに追加して，マッピングを定義します。

9.9.1 フィルタの追加

次の手順で，WAR ファイルにフィルタを追加します。

1. **フィルタ属性ファイルを作成します。**
   テキストエディタなどを使用して，フィルタ属性ファイルを作成します。フィルタ属性ファイルには，フィルタ名称およびフィルタのクラス名を指定します。また，必要に応じて，初期化パラメタの名称および初期化パラメタの値を指定します。
   フィルタ属性ファイルの詳細については，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(アプリケーション／リソース定義)」の「3.8 フィルタ属性ファイル」を参照してください。
2. **次に示すコマンドを実行して J2EE アプリケーション内の WAR ファイルにフィルタを登録します。**

実行形式

```
cjaddapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type filter -name <J2EEアプリケーション名> -warname <フィルタを追加するWARの表示名> -c <フィルタ属性ファイルパス>
```

実行例

```
cjaddapp MyServer -type filter -name App1 -warname account-war -c FilterProp.xml
```

cjaddapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjaddapp（リソースの追加)」を参照してください。

9.9.2 フィルタのマッピング定義

フィルタの追加後，フィルタのマッピングを定義します。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WARの表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WARの表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

Web アプリケーション（サーブレットおよび JSP）のフィルタマッピング定義（<filter-mapping>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
フィルタ名	○	<filter-name>
URL パターン	○※2	<url-pattern>
サーブレット名	○※2	<servlet-name>
フィルタの適用条件※1	△	<dispatcher>

（凡例） ○：必須 △：任意

注※1 Servlet2.3 以前の WAR には，この属性は設定できません。

注※2 フィルタのマッピングは，URL パターンまたはサーブレット名のどちらかを指定してください。

プロパティの設定項目については，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「3.7.1 WAR 属性ファイルの指定内容」を参照してください。

9.9.3 フィルタの削除

次に示すコマンドを実行してフィルタを削除します。

実行形式

```
cjdeleteapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type filter -resname <削除するWARの表示名>/<削除するフィルタの表示名>
```

WAR ファイルごとフィルタを削除する場合は，削除する WAR の表示名を指定します。フィルタだけを削除する場合は，削除する WAR の表示名とフィルタの表示名を指定します。

実行例

```
cjdeleteapp MyServer -name App1 -type filter -resname
account-war/account-filter
```

cjdeleteapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeleteapp（J2EE アプリケーションの削除）」を参照してください。

9.10　Enterprise Bean の実行時属性の定義

次に示す Enterprise Bean の実行時属性を定義します。

- Stateful Session Bean
- Stateless Session Bean
- Entity Bean
- Message-driven Bean

9.10.1　Stateful Session Bean の実行時プロパティの設定

アプリケーションを構成する個々の Stateful Session Bean に対して，アプリケーション実行時のプロパティを設定します。

(1) 編集する属性ファイル

Session Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Session Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Stateful Session Bean表示名> -c <Session Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder/MyAdder -c C:¥home¥MyAdder.xml
```

● 属性の設定

次に示すコマンドを実行して，Session Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Stateful Session Bean表示名> -c <Session Bean属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder/
```

```
MyAdder -c C:¥home¥MyAdder.xml
```

(3) 編集する属性設定項目

Stateful Session Bean の実行時プロパティは，次の二つに分けられます。

- Session Bean 共通の実行時属性
- Stateful Session Bean 固有の属性

ぞれぞれの設定項目を次に示します。

(a) Session Bean 共通の実行時属性

Session Bean 共通の実行時属性（<runtime>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
Enterprise Bean のルックアップ名※1	○	<lookup-name>
リモートインタフェースを持つ Enterprise Bean 名の別名※1	△	<optional-name>
ローカルインタフェースを持つ Enterprise Bean 名の別名※1	△	<local-optional-name>
セッション最大数※2	△	<maximum-sessions>
参照渡しの設定※3	△	<pass-by-reference>

（凡例） ○：必須　△：任意

注※1

JNDI 名前空間に登録される名称の参照と変更については，「9.13.2　Enterprise Bean 名の参照と変更」を参照してください。

注※2

Stateful Session Bean に指定された最大セッション数は，EJB クライアントアプリケーションから利用できる Stateful Session Bean の最大数です。「0」を指定した場合，同時に生成できる Stateful Session Bean のセッション（EJB オブジェクト）の数は無制限（実際の最大値は 2147483647）になります。

注※3

参照渡しの設定は，usrconf.properties ファイルでも実施できます。プロパティまたは usrconf.properties ファイルのどちらかで設定されていれば，参照渡しの設定は有効になります。

プロパティの設定項目については，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「3.4.1　Session Bean 属性ファイルの指定内容」を参照してください。

(b) Stateful Session Bean 固有の属性

Stateful Session Bean 固有の実行時属性（<runtime> - <stateful>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
アクティブセッションの最大数	○	<maximum-active-sessions>
再び活性化するまでに，非活性化状態に保持している時間	○	<inactivity-timeout>
セッションが削除されるまでに非活性化状態に保持している時間	○	<removal-timeout>

（凡例） ○：必須

プロパティの設定項目については，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「3.4.1 Session Bean 属性ファイルの指定内容」を参照してください。

プロパティ項目の設定と動作について，次に説明します。

アクティブセッションの最大数

生成された Stateful Session Bean の状態には，次の二つの状態があります。

- method-ready 状態
- passive 状態

method-ready 状態の Stateful Session Bean は，EJB クライアントアプリケーションからアクセスされた時点ですでに実行できるようになっています。
passive 状態の Stateful Session Bean は，活性化され method-ready 状態になってから実行できます。
アクティブセッションの最大数に，同時に method-ready 状態になることができる Stateful Session Bean の最大数を指定します。この値は，複数の EJB クライアントアプリケーションからアクセスされたときに，すぐに実行開始できる Stateful Session Bean の最大数です。ここで指定した数以上の Stateful Session Bean が生成された場合，method-ready 状態の Stateful Session Bean が幾つか非活性化され passive 状態になります。ただし，このとき，method-ready 状態の Stateful Session 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。セッション最大数，またはアクティブセッションの最大数に「0」を指定した場合は，非活性化の機能は動作しません（Stateful Session Bean が method-ready 状態から passive 状態に移りません）。

再び活性化するまでに，非活性化状態に保持している時間（デフォルト：0（分））

タイムアウト値を分単位で指定します。ここで指定された時間が経過するまでに passive 状態の Stateful Session Bean が活性化されない場合，この Bean は削除されます。再び活性化するまでに，非活性化状態に保持している時間に「0」を設定した場合は，タイムアウトは発生しません。また，Stateful Session Bean のインスタンスの削除は，指定した <inactivity-timeout> 値に対し，最大で ejbserver.container.passivate.scan.interval に指定した時間分の遅延が生じることがあります。この起動間隔を変更したい場合は，usrconf.properties ファイルのシス

テムプロパティ ejbserver.container.passivate.scan.interval に起動間隔を秒単位で指定してください。デフォルトでは 0 秒が設定されています。
ejbserver.container.passivate.scan.interval の指定例を次に示します。

10 分間隔で passive 状態の Stateful Session Bean の削除機能を起動したい場合

```
ejbserver.container.passivate.scan.interval=600
```

セッションが削除されるまでに非活性化状態に保持している時間（デフォルト：0（分））
タイムアウト値を分単位で指定します。method-ready 状態の Stateful Session Bean がここで指定された時間の間，一度も使用されない場合，この Bean は削除されます。ただし，このとき，method-ready 状態の Stateful Session Bean の中でトランザクション中のものについては，削除の対象になりません。
セッションが削除されるまでに非活性化状態に保持している時間に「0」を設定した場合は，タイムアウトは発生しません。また，Stateful Session Bean のインスタンスの削除は，指定した <removal-timeout> 値に対し，最大で ejbserver.container.remove.scan.interval に指定した時間分の遅延が生じることがあります。この起動間隔を変更したい場合は，J2EE サーバ用の usrconf.properties ファイルの ejbserver.container.remove.scan.interval に起動間隔を分位で指定してください。デフォルト値には 5 分が仮定されています。
ejbserver.container.remove.scan.interval の指定例を次に示します。

10 分間隔で method-ready 状態の Stateful Session Bean の削除機能を起動したい場合

```
ejbserver.container.remove.scan.interval=10
```

（4） 注意事項

- デフォルトのユーザ定義では，「ejbActivate，ejbPassivate を呼び出し，Stateful Session Bean の状態を保存，復元する機能」であり，デフォルトでは，Stateful Session Bean の活性化，非活性化は動作しません。したがって，Stateful Session Bean の Enterprise Bean 実行時プロパティとして設定できる，<maximum-active-sessions> の設定は有効になりません。また，非活性化された Stateful Session Bean をタイムアウトで削除する機能は，動作しません。したがって，Stateful Session Bean の Enterprise Bean 実行時プロパティとして設定できる，<inactivity-timeout> の設定は有効になりません。
- 活性化，非活性化の機能を利用する場合は，J2EE サーバ用の usrconf.properties ファイルで，ejbserver.stateful.passivate.switch キーに対し「true」を指定します。
- 非活性化が動作するとき，Stateful Session Bean の状態はファイルに書き込むことで

保存され，インスタンスは破棄されます。逆に活性化が動作するときはファイルから Stateful Session Bean の状態が読み込まれインスタンスが復元されます。活性化，非活性化が動作する場合，Stateful Session Bean（またはそれが参照するクラス）のメンバ変数に設定されているオブジェクトも保存，復元の対象になります。このとき，保存，復元ができるものを次に示します。

- 直列化できるオブジェクト
- EJB オブジェクトリファレンス
- EJB ホームオブジェクトリファレンス
- "java:comp/env" のルックアップで取得した javax.naming.Context
- javax.ejb.SessionContext

なお，javax.jta.UserTransaction は保存，復元できないため，メンバ変数に保持しないでください。保存，復元ができないオブジェクトをメンバ変数に保持している場合は，ejbPassivate で解放し（メンバ変数に null 代入），ejbActivate で再取得してください。

● 同時に生成できる Stateful Session Bean のセッションの最大数と method-ready 状態の Stateful Session Bean の最大数は次の条件を満たすように指定してください。

```
method-ready状態のStateful Session Beanの最大数≦同時に生成できる
Stateful Session Beanのセッションの最大数
```

また，同時に生成できる Stateful Session Bean のセッションの最大数を指定した場合，method-ready 状態の Stateful Session Bean の最大数も指定する必要があります。

9.10.2 Stateless Session Bean の実行時プロパティの設定

アプリケーションを構成する個々の Stateless Session Bean に対して，アプリケーション実行時のプロパティを設定します。

(1) 編集する属性ファイル

Session Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Session Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Stateless Session Bean表示名> -c <Session Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder/
MyAdder -c C:¥home¥MyAdder.xml
```

● 属性の設定

次に示すコマンドを実行して，Session Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Stateless Session Bean表示名> -c <Session Bean属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder/
MyAdder -c C:¥home¥MyAdder.xml
```

(3) 編集する属性設定項目

Stateless Session Bean の実行時プロパティは，次の三つに分けられます。

- Session Bean 共通の実行時属性
- Stateless Session Bean 固有の属性
- CTM 連携の実行時属性

ぞれぞれの設定項目を次に示します。

(a) Session Bean 共通の実行時属性

Session Bean 共通の実行時属性（<runtime>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
Enterprise Bean のルックアップ名※1	○	<lookup-name>
リモートインタフェースを持つ Enterprise Bean 名の別名※1	△	<optional-name>
ローカルインタフェースを持つ Enterprise Bean 名の別名※1	△	<local-optional-name>
セッション最大数※2	△	<maximum-sessions>
スケジューリングの設定※3	△	<enable-scheduling>
参照渡しの設定※4	△	<pass-by-reference>

（凡例） ○：必須 △：任意

注※1

JNDI 名前空間に登録される名称の参照と変更については，「9.13.2　Enterprise Bean 名の参照と変更」を参照してください。

注※ 2

Stateless Session Bean に指定された最大セッション数は，クライアントが使用できる Stateless Session Bean インスタンス数を定義します。ただし，この指定値は有効にはなりません。

注※ 3

CTM のスケジューリングについては，「9.14　CTM のスケジューリング」を参照してください。

注※ 4

参照渡しの設定は，usrconf.properties ファイルでも実施できます。プロパティまたは usrconf.properties ファイルのどちらかで設定されていれば，参照渡しの設定は有効になります。

プロパティの設定項目については，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「3.4.1　Session Bean 属性ファイルの指定内容」を参照してください。

(b) Stateless Session Bean 固有の属性

Stateless Session Bean 固有の実行時属性（<runtime> - <stateless>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
プール内のインスタンスの最小値	○	<pooled-instance> - <minimum>
プール内のインスタンスの最大値	○	<pooled-instance> - <maximum>

（凡例）　○：必須

プロパティの設定項目については，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「3.4.1　Session Bean 属性ファイルの指定内容」を参照してください。

プロパティ項目の設定と動作について，次に説明します。

プール内のインスタンスの最小値

J2EE アプリケーション起動時にここで指定した数の Stateless Session Bean が生成され，プーリングされます。プーリングされる Stateless Session Bean の数は，EJB クライアントアプリケーションからのアクセス量に応じて，最大数と最小数の間になります。プール内のインスタンスの最小値に「0」を指定した場合は，J2EE アプリケーション起動時に Stateless Session Bean が生成されません。
この値は，アプリケーションが開始されたときに作成する Stateless Session Bean インスタンスの数も指定します。

プール内のインスタンスの最大値

生成された Stateless Session Bean は，method-ready でプーリングされます。プーリングされた Stateless Session Bean は，EJB クライアントアプリケーションからアクセスされた時点ですでに実行できるようになっています。プール内のインスタンスの最大値に，プーリングされる Stateless Session Bean の最大数を指定します。この値は，複数の EJB クライアントアプリケーションからアクセスされたときに，すぐに実行開始できる Stateless Session Bean の最大数です。プール内のインスタンスの最大値に 0 を指定した場合は，プーリングされる Stateless Session Bean の数に制限はありません。

(c) CTM 連携の実行時属性

CTM 連携の実行時属性（<runtime> - <scheduling>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
キュー名	○	<queue-name>
スレッド数	○	<parallel-count>

（凡例） ○：必須

プロパティの設定項目については，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「3.4.1　Session Bean 属性ファイルの指定内容」を参照してください。

CTM のスケジューリングについては，「9.14　CTM のスケジューリング」を参照してください。

(4) 注意事項

- クライアント側からアクセスできる Stateless Session Bean の上限数をチェックする機能は動作しません。代わりに＜プール内のインスタンスの最大値＞で設定します。
- プーリングされる Stateless Session Bean の最小数とプーリングされる Stateless Session Bean の最大数は，次の条件を満たすように指定してください。

  プーリングされるStateless Session Beanの最小数≦プーリングされるStateless Session Beanの最大数

  また，プーリングされる Stateless Session Bean の最大数を指定した場合，プーリングされる Stateless Session Bean の最小数も指定する必要があります。

9.10.3　Entity Bean の実行時プロパティの設定

アプリケーションを構成する個々の Entity Bean に対して，アプリケーション実行時の

プロパティを設定します。

(1) 編集する属性ファイル

Entity Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Entity Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Entity Bean表示名> -c <Entity Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name account -type ejb -resname account/MyAccount -c C:¥home¥MyAccount.xml
```

● 属性の設定

次に示すコマンドを実行して，Entity Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Entity Bean表示名> -c <Entity 属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name account -type ejb -resname account/MyAccount -c C:¥home¥MyAccount.xml
```

(3) 編集する属性設定項目

Entity Bean の実行時属性（<runtime>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
Enterprise Bean のルックアップ名※1	○	<lookup-name>
リモートインタフェースを持つ Enterprise Bean 名の別名※1	△	<optional-name>
ローカルインタフェースを持つ Enterprise Bean 名の別名※1	△	<local-optional-name>

項目	必須	対応するタグ名
セッション最大数※2	△	<maximum-instances>
参照渡しの設定※3	△	<pass-by-reference>
プール内のインスタンスの最大値	○	<pooled-instance> - <maximum>
プール内のインスタンスの最小値	○	<pooled-instance> - <minimum>
データのキャッシュ方法	○	<caching-model>
EJB オブジェクトの存在期限	△	<entity-timeout>

（凡例） ○：必須　△：任意

注※ 1

JNDI 名前空間に登録される名称の参照と変更については，「9.13.2　Enterprise Bean 名の参照と変更」を参照してください。

注※ 2

Entity Bean に指定された最大セッション数は，EJB クライアントアプリケーションから利用できる Entity Bean の最大値です。0 を指定した場合は，同時に生成できる Entity Bean のリモートオブジェクトの数は無制限（実際の最大値は 2147483647）になります。

注※ 3

参照渡しの設定は，usrconf.properties ファイルでも実施できます。プロパティまたは usrconf.properties ファイルのどちらかで設定されていれば，参照渡しの設定は有効になります。

プロパティの設定項目については，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「3.5.1　Entity Bean 属性ファイルの指定内容」を参照してください。

プロパティ項目の設定と動作について，次に説明します。

プール内のインスタンスの最大値

メモリ上でプーリングされる Entity Bean には，次の二つの状態があります。

- ready 状態
  ready 状態の Entity Bean は，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 EJB クライアントアプリケーションからアクセスされた時点ですでに実行できるようになっています。
- pool 状態
  pool 状態の Entity Bean は，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できます。

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処理中のものについては，非活性化の対象になりません。
プール内のインスタンスの最大値（<pooled-instance> - <maximum>）には，プーリングされる ready 状態と pool 状態の Entity Bean の最大数を指定します。これは，メモリ上に展開される Entity Bean インスタンスの上限です。
プール内のインスタンスの最大値に「0」を指定した場合は，プーリングされる Entity Bean インスタンスの数は無制限になります。
プールされたインスタンスの数がこの値を超えると，J2EE サーバは，最も活性でないインスタンスを非活性化します。

プール内のインスタンスの最小値

プーリングされる ready 状態と pool 状態の Entity Bean の最小数を指定します。この値は，メモリ上に展開される Entity Bean インスタンスの下限です。J2EE アプリケーション起動時にここで指定した数の Entity Bean が生成され，pool 状態，または ready 状態になりプーリングされます。プーリングされる Entity Bean の数は，EJB クライアントアプリケーションからのアクセス量に応じて，最大数と最小数の間になります。
プール内のインスタンスの最小値（<pooled-instance> - <minimum>）に 0 を指定した場合は，J2EE アプリケーション起動時に Entity Bean が生成されません。クライアントに参照されない Entity Bean インスタンスを，最低幾つメモリに保持しておくかを指定します。この値は，プール内のインスタンスの最大値（<pooled-instance> - <maximum>）の値を超えてはいけません。

データのキャッシュ方法

Entity Bean のキャッシュモデル（コミットオプション）を指定します。

- Full caching（commit option A）
  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ないため，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）。Full caching は参照系の Entity Bean 用のキャッシュモデルです。
- Caching（commit cache option B）
  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るため，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。Caching は更新系の Entity Bean 用のキャッシュモデルです。
- No caching（commit cache option C）
  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。No caching は更新系の Entity Bean 用です。また，トランザクションコミット時に必ず非活性化されるため，多数の Entity Bean を利用する場合のキャッシュモデルです。

EJB オブジェクトの存在期限

タイムアウト値を秒単位で指定します。値は，0 または正の整数値で指定します。
パッシベイトされた Entity Bean の EJB オブジェクトは，最小でこのタイムアウト値に指定した時間，最大でこのタイムアウト値に指定した時間 +usrconf.properties ファイルの ejbserver.container.passivate.scan.interval に指定した値の時間だけ存在します。クライアントからタイムアウト値の時間を経過してもアクセスされない場合，該当する EJB オブジェクトは削除されます。
EJB オブジェクトの存在期限（<entity-timeout>）に「0」を指定した場合は，タイムアウトは発生しません。

(4) 注意事項

- 同時に生成できる Entity Bean の EJB オブジェクトの最大数に上限を指定した場合，プーリングされる Entity Bean の最大数にも上限を指定しなければいけません。
- 同時に生成できる Entity Bean の EJB オブジェクトの最大数，プーリングされる Entity Bean の最大数，プーリングされる Entity Bean の最小数の値は，次の条件を満たすように指定してください。

  プーリングされるEntity Beanの最小数≦プーリングされるEntity Beanの最大数≦同時に生成できるEntity BeanのEJBオブジェクトの最大数

9.10.4 Message-driven Bean の実行時プロパティの設定

アプリケーションを構成する個々の Message-driven Bean に対して，アプリケーション実行時のプロパティを設定します。

(1) 編集する属性ファイル

Message-driven Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Message-driven Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Message-driven Bean表示名> -c <Message-driven Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name message -type ejb -resname message/
MyMessageBean -c C:¥home¥MyMessageBean.xml
```

● 属性の設定

次に示すコマンドを実行して，Message-driven Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Message-driven Bean表示名> -c <Message-driven Bean属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name message -type ejb -resname message/
MyMessageBean -c C:¥home¥MyMessageBean.xml
```

(3) 編集する属性設定項目

Message-driven Bean の実行時属性（<runtime>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
プール内のインスタンスの最大値※	○	<pooled-instance> - <maximum>

（凡例） ○：必須

注※ Server Session プールの初期化時に指定値まで Bean を生成します。1 ～ 2147483647（int 型の最大値）を指定できます。デフォルト値は 1 です。指定数分の JMS セッションを生成するため，JMS プロバイダで生成できる JMS セッション数に合わせて定義します。

プロパティの設定項目については，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「3.6.1 MessageDrivenBean 属性ファイルの指定内容」を参照してください。

9.11 サーブレットと JSP の実行時属性の定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）の実行時属性を定義します。次に示す定義があります。

- J2EE アプリケーションのコンテキストルートの定義
- Web アプリケーション単位での同時実行スレッド数制御の定義
- URL グループ単位での同時実行スレッド数制御の定義
- URL グループ単位の実行待ちリクエストの監視の定義

それぞれの定義で設定するプロパティ項目を次に示します。

9.11.1 J2EE アプリケーションのコンテキストルート定義

J2EE アプリケーションのコンテキストルートを定義します。

コンテキストルートには，ルートコンテキストも定義できます。**ルートコンテキスト**とは，コンテキストルートが空文字 "" のコンテキストです。ルートコンテキストに welcome ファイルを作成すると，「http://www.cosminexus.com/」のようなドメイン名だけの URL から J2EE アプリケーションのトップページを表示できるようになります。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

J2EE アプリケーションのコンテキストルート定義（<runtime>）のプロパティ設定項目を，次に示します。

項目	必須	対応するタグ名
コンテキストルート	○	<context-root>

（凡例） ○：必須

プロパティの設定項目については，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「3.7.1 WAR 属性ファイルの指定内容」を参照してください。

(4) 注意事項

- コンテキストルートには，"ejb/"，または "web/" から始まる文字列を指定しないでください。
  複数の J2EE アプリケーションでコンテキストルートのパスの構成要素が包含関係（例："test" と "test/jsp"）にある場合で，包含する方のパスを含む URL（例："/test/jsp/test.jsp"）を指定してアクセスしたときは，包含する方のアプリケーション（例では，"test/jsp" をコンテキストルートに持つ方）が有効になり，包含される方のアプリケーション（例では，"test" をコンテキストルートに持つ方）にはアクセスできません。
- コンテキストルートは，作業ディレクトリ中のディレクトリ名として用いられます。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コンテキストルート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，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「7.10.2 作業ディレクトリのパス長の見積もり式」を参照してください。
- ルートコンテキストを使用して開始する Web アプリケーションでは，URL が "ejb" または "web" で始まる構成にしないでください。
- コンソールおよびログファイルに出力されるメッセージでは，ルートコンテキストのコンテキストルートは，空文字「""」で表示されます。

9.11.2 Web アプリケーション単位での同時実行スレッド数制御の定義

Web コンテナで同時実行スレッド数を制御するための定義をします。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

Web コンテナで同時実行スレッド数を制御するためのプロパティ設定項目（<thread-control>）を次に示します。

項目	必須	対応するタグ名
最大同時実行スレッド数	○	<thread-control-max-threads>
占有して使用するスレッド数	○	<thread-control-exclusive-threads>

項目	必須	対応するタグ名
Web アプリケーション単位の実行待ちキューサイズ	○	<thread-control-queue-size>

（凡例） ○：必須

プロパティの設定項目については，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「3.7.1 WAR 属性ファイルの指定内容」を参照してください。

（4） 注意事項

Web アプリケーション単位での同時実行スレッド数制御を有効にするためには，J2EE サーバ用の usrconf.properties ファイルの webserver.container.thread_control.enabled キーに「true」を指定してください。

9.11.3 URL グループ単位での同時実行スレッド数制御の定義

URL グループ単位での同時実行スレッド数を制御するための定義をします。

URL グループ単位での同時実行スレッド数制御を有効にするためは，Web アプリケーション単位での同時実行スレッド数制御の定義を設定してください。Web アプリケーション単位での同時実行スレッド数制御については，「9.11.2 Web アプリケーション単位での同時実行スレッド数制御の定義」を参照してください。

（1） 編集する属性ファイル

WAR 属性ファイル

（2） 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

URL グループ単位で同時実行スレッド数を制御するためのプロパティ設定項目（<urlgroup-thread-control>）を次に示します。

項目	必須	対応するタグ名
定義名	○	<urlgroup-thread-control-name>
最大同時実行スレッド数	○	<urlgroup-thread-control-max-threads>
占有スレッド数	○	<urlgroup-thread-control-exclusive-threads>
URL グループ単位の実行待ちキューサイズ	○	<urlgroup-thread-control-queue-size>
URL パターン	○	<urlgroup-thread-control-mapping> - <url-pattern>

（凡例） ○：必須

プロパティの設定項目については，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「3.7.1 WAR 属性ファイルの指定内容」を参照してください。

(4) 注意事項

● URL グループ単位での同時実行スレッド数制御で実行されるスレッド数は，Web アプリケーション単位で実行されるスレッド数に含まれています。したがって，URL グループ単位で一つスレッドが実行されている場合，この Web アプリケーションの Web アプリケーション単位でも一つスレッドが実行されていることになります。

● URL グループ単位での同時実行スレッド数は，次の範囲で設定します。

- 最大同時実行スレッド数
  最大同時実行スレッド数の指定範囲を次に示します。

1 ≦ URL グループ単位の最大同時実行スレッド数の総和 ≦ Web アプリケーション単位の最大同時実行スレッド数

- 占有スレッド数
  占有スレッド数の指定範囲を次に示します。
  条件 1 と条件 2 は，すべて満たしている必要があります。

条件 1：
- Web アプリケーション単位の最大同時実行スレッド数＝ Web アプリケーション単位の占有スレッド数の場合
  0 ≦ URL グループ単位の占有スレッド数 ≦ URL グループ単位の最大同時実行スレッド数
- Web アプリケーション単位の最大同時実行スレッド数≠ Web アプリケーション単位の占有スレッド数の場合
  0 ≦ URL グループ単位の占有スレッド数 < URL グループ単位の最大同時実行スレッド数

条件 2：
URL グループ単位の占有スレッド数の総和 ≦ Web アプリケーション単位の占有スレッド数

9.11.4 URL グループ単位の実行待ちリクエスト数の監視の定義

URL グループ単位の実行待ちリクエスト数を監視するための定義をします。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

URL グループ単位の実行待ちリクエスト数を監視するためのプロパティ設定項目（<urlgroup-thread-control> - <stats-monitor> - <waiting-request-count>）を次に示します。

項目	必須	対応するタグ名
しきい値イベント監視有無	○	<enabled>
しきい値イベントの出力上限しきい値※	○	<high-threshold>
しきい値イベントの出力下限しきい値※	○	<low-threshold>

（凡例） ○：必須
注※ 「しきい値イベントを出力する上限しきい値≧しきい値イベントを出力する下限しきい値」となるよう指定してください。

プロパティの設定項目については，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「3.7.1 WAR 属性ファイルの指定内容」を参照してください。

9.12 デフォルトの文字エンコーディングの設定

Web アプリケーション単位にデフォルトの文字エンコーディングを設定します。

次の文字エンコーディングについて，デフォルトのエンコーディングが設定できます。

- リクエストボディおよびクエリの文字エンコーディング
- レスポンスボディの文字エンコーディング
- JSP ファイルの文字エンコーディング

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

● 属性ファイルの設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder_war
-c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

デフォルトの文字エンコーディングのプロパティ設定項目を次に示します。

項目	必須	対応するタグ名
リクエストボディおよびクエリの文字エンコーディング	△	<http-request> - <encoding>
レスポンスボディの文字エンコーディング	△	<http-response> - <encoding>
JSP ファイルの文字エンコーディング	△	<jsp> - <page-encoding>

（凡例） △：任意

プロパティの設定項目については，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「3.7.1　WAR 属性ファイルの指定内容」を参照してください。

9.13　JNDI 名前空間に登録される名称の参照と変更

JNDI 名前空間に登録される J2EE アプリケーション名や Enterprise Bean 名を変更します。Enterprise Bean 名には，別名も付けられます。

EJB ホームオブジェクトは，J2EE アプリケーション開始時に JNDI 名前空間（HITACHI_EJB/SERVERS/ ＜サーバ名称＞ /EJB/ ＜ J2EE アプリケーション名＞ / ＜ Enterprise Bean 名＞）に割り当てられます。この名前は，EJB クライアントアプリケーションから参照する場合やネーミングの切り替え機能を使って参照する場合に使用します。

EJB ホームオブジェクトには，別名（Optional Name）が付けられます。別名を付けることで，EJB クライアントアプリケーションは JNDI 名前空間から任意の名前で EJB ホームオブジェクトを取得できるようになります。なお，この機能を**ユーザ指定名前空間機能**といいます。

なお，EJB ホームオブジェクトの JNDI 名前空間については，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。ユーザ指定名前空間機能については，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「2.4　Enterprise Bean または J2EE リソースへの別名付与（ユーザ指定名前空間機能）」を参照してください。

9.13.1　J2EE アプリケーション名の参照

J2EE アプリケーション名の参照は，次の手順で実行します。

1. **アプリケーション属性ファイルを取得します。**
   次に示すコマンドを実行して，アプリケーション属性ファイルを取得します。
   実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -c <アプリケーション属性ファイルパス>
```

   実行例

```
cjgetappprop MyServer -name account -c C:¥home¥accountAPP.xml
```

2. **テキストエディタなどでアプリケーション属性ファイルを開きます。**
   アプリケーション属性ファイルの J2EE アプリケーション名（<lookup-name>）を参照します。

cjgetappprop 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「cjgetappprop（アプリケーションの属性の取得）」を参照してください。属性ファイルの詳細については，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（アプリケーション／リソース定義）」の「3.2　アプリケーション属性ファイル」を参照してください。

注意事項

JNDI 名前空間に登録される J2EE アプリケーション名は J2EE アプリケーション名を基に自動的に割り当てられます。変更はできません。

9.13.2　Enterprise Bean 名の参照と変更

次の Enterprise Bean の Enterprise Bean 名を参照および変更します。

- Session Bean
- Entity Bean

(1) 編集する属性ファイル

- Session Bean 属性ファイル
- Entity Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次のコマンドを実行して，Enterprise Bean 属性ファイル（Session Bean 属性ファイルまたは Entity Bean 属性ファイル）を取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Bean属性ファイルパス※>
```

注※　Enterprise Bean 属性ファイルパスには，Session Bean 属性ファイルまたは Entity Bean 属性ファイルのファイルパスを指定します。

実行例

```
cjgetappprop MyServer -name account -type ejb -resname account/MyAccount -c C:¥home¥MyAccount.xml
```

● 属性ファイルの設定

次のコマンドを実行して，Enterprise Bean 属性ファイル（Session Bean 属性ファイルまたは Entity Bean 属性ファイル）の値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Bean属性ファイルパス※>
```

注※ Enterprise Bean 属性ファイルパスには，Session Bean 属性ファイルまたは Entity Bean 属性ファイルのファイルパスを指定します。

実行例

```
cjsetappprop MyServer -name account -type ejb -resname account/MyAccount -c C:¥home¥MyAccount.xml
```

(3) 編集する属性設定項目

Enterprise Bean 属性ファイル（Session Bean 属性ファイルまたは Entity Bean 属性ファイル）の Enterprise Bean 名を，実行時属性（<runtime>）で参照および変更します。

項目	必須	対応するタグ名
Enterprise Bean のルックアップ名	○	<lookup-name>
リモートインタフェースを持つ Enterprise Bean 名の別名	△	<optional-name>
ローカルインタフェースを持つ Enterprise Bean 名の別名	△	<local-optional-name>

（凡例） ○：必須 △：任意

プロパティの設定項目については，次の個所を参照してください。

- 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「3.4.1 Session Bean 属性ファイルの指定内容」
- 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「3.5.1 Entity Bean 属性ファイルの指定内容」

(4) 注意事項

- usrconf.properties ファイルの ejbserver.cui.optionalname.enabled キーに true が設定されていて，かつ EJB がリモートインタフェースを実装している場合にだけ指定できます。
- JNDI 名前空間に登録される Enterprise Bean 名の初期値は，Enterprise Bean の ejb-name が使われます。
- Message-driven Bean には JNDI 名前空間に登録される Enterprise Bean 名がありません。

9.14 CTM のスケジューリング

CTM を利用する場合の，スケジューリングについて設定します。CTM は，構成ソフトウェアに Cosminexus Component Transaction Monitor を含む製品だけで利用できます。利用できる製品については，マニュアル「Cosminexus アプリケーションサーバ 概説」の「2.3.1 製品と構成ソフトウェアの対応」を参照してください。

次の手順で，CTM のスケジューリングを設定します。

- アプリケーション単位のスケジューリングを設定します。
- スケジューリングの対象にしたい Stateless Session Bean のスケジューリングを設定します。

9.14.1 J2EE アプリケーション単位のスケジューリング

J2EE アプリケーション単位の CTM との連携にかかわる設定をします。

アプリケーションの属性と Session Bean の属性の両方を設定する必要があるので，アプリケーション統合属性ファイルを使用して設定することを推奨します。

(1) 編集する属性ファイル

アプリケーション統合属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してアプリケーション統合属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type all -c <アプリケーション統合属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name account -type all -c
C:¥home¥account.xml
```

● 属性ファイルの設定

次に示すコマンドを実行して，アプリケーション統合属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type all -c <アプリケーション統合属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name account -type all -c
C:¥home¥account.xml
```

(3) 編集する属性設定項目

編集する属性設定項目を次に示します。

- アプリケーションの属性
- Session Bean の属性

ぞれぞれの設定項目を次に示します。

(a) アプリケーションの属性

J2EE アプリケーション単位の CTM との連携にかかわる設定項目を次に示します。

項目	必須	対応するタグ名
CTM 連携有無	○	<managed-by-ctm>
キューの配置モデル	○	<scheduling-unit>
キュー名	○	<scheduling> - <queue-name>
スレッド数	○	<scheduling> - <parallel-count>
キューの長さ	△	<scheduling> - <queue-length>

（凡例） ○：必須 △：任意

プロパティの設定項目については，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「3.2.1 アプリケーション属性ファイルの指定内容」を参照してください。

(b) Session Bean の属性

J2EE アプリケーション単位の CTM との連携にかかわる実行時属性設定項目（<session-runtime>）を次に示します。

項目	必須	対応するタグ名
スケジューリングの設定	○	<enable-scheduling>

（凡例） ○：必須

プロパティの設定項目については，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「3.4.1 Session Bean 属

性ファイルの指定内容」を参照してください。

(4) 注意事項

アプリケーション統合属性ファイルを使用しないで，アプリケーション属性ファイルおよび Session Bean 属性ファイルを使用して設定する場合は，次の順に設定する必要があります。

1. 次に示すコマンドを実行して，Session Bean 属性ファイルの <enable-scheduling> タグの値を反映します。

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Stateless Session Bean表示名> -c <Session Bean属性ファイルパス>
```

2. 次に示すコマンドを実行して，アプリケーション属性ファイルの <managed-by-ctm> タグ，<scheduling-unit> タグ，および <scheduling> タグの下位のタグの値を反映します。

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -c <アプリケーション属性ファイルパス>
```

9.14.2 Stateless Session Bean 単位のスケジューリング

CTM によるスケジューリングの対象に設定した J2EE アプリケーションに含まれている Stateless Session Bean を CTM によるスケジューリングの対象にする場合，Stateless Session Bean 単位のスケジューリングの設定をします。

スケジューリングの対象となるのは，リモートホームインタフェースを実装している Stateless Session Bean だけです。

スケジューリング機能を利用する場合，Stateless Session Bean の実行時プロパティを設定します。Stateless Session Bean の実行時プロパティの設定については，「9.10.2 Stateless Session Bean の実行時プロパティの設定」を参照してください。

また，アプリケーションの属性で CTM に連携するための設定をします。

アプリケーションの属性と Session Bean の属性の両方を設定する必要があるので，アプリケーション統合属性ファイルを使用して設定することを推奨します。

(1) 編集する属性ファイル

アプリケーション統合属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してアプリケーション統合属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type all -c <アプリケーション統合属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type all -c C:¥home¥Adder.xml
```

● 属性の設定

次に示すコマンドを実行して，アプリケーション統合属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type all -c <アプリケーション統合属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type all -c C:¥home¥Adder.xml
```

(3) 編集する属性設定項目

編集する属性設定項目を次に示します。

- アプリケーションの属性
- Session Bean の属性

ぞれぞれの設定項目を次に示します。

(a) アプリケーションの属性

CTM に連携するための設定項目を次に示します。

項目	必須	対応するタグ名
CTM 連携有無	○	<managed-by-ctm>
キューの配置モデル	○	<scheduling-unit>

（凡例） ○：必須

プロパティの設定項目については，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「3.2.1 アプリケーショ

ン属性ファイルの指定内容」を参照してください。

(b) Session Bean の属性

CTM に連携するための実行時属性設定項目（<session-runtime>）を次に示します。

項目	必須	対応するタグ名
スケジューリングの設定	○	<enable-scheduling>
キュー名	○	<scheduling> - <queue-name>
スレッド数	○	<scheduling> - <parallel-count>
キューの長さ	△	<scheduling> - <queue-length>

（凡例） ○：必須　△：任意

プロパティの設定項目については，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「3.4.1　Session Bean 属性ファイルの指定内容」を参照してください。

(4) 注意事項

アプリケーション統合属性ファイルを使用しないで，アプリケーション属性ファイルおよび Session Bean 属性ファイルを使用して設定する場合は，次の順に設定する必要があります。

1. 次に示すコマンドを実行して，Session Bean 属性ファイルの <enable-scheduling> タグの値を反映します。

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -name ＜J2EEアプリケーション名＞ -type ejb -resname ＜EJB-JAR表示名＞/＜Stateless Session Bean表示名＞ -c ＜Session Bean属性ファイルパス＞
```

2. 次に示すコマンドを実行して，アプリケーション属性ファイルの <managed-by-ctm> タグおよび <scheduling-unit> タグの値を反映します。

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -name ＜J2EEアプリケーション名＞ -c ＜アプリケーション属性ファイルパス＞
```

3. 次に示すコマンドを実行して，Session Bean 属性ファイルの <scheduling> タグの下位のタグの値を反映します。

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -name ＜J2EEアプリケーション名＞ -type ejb -resname ＜EJB-JAR表示名＞/＜Stateless Session Bean表示名＞ -c ＜Session Bean属性ファイルパス＞
```

9.15　起動順序の設定

J2EE アプリケーションの起動順序，および J2EE アプリケーションに含まれる Enterprise Bean および WAR の起動順序を設定します。

9.15.1　J2EE アプリケーションの起動順序の設定

J2EE アプリケーションの起動順序を設定します。

(1) 編集する属性ファイル

アプリケーション属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してアプリケーション属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -c <アプリケーション属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name account -c C:¥home¥MyAccount.xml
```

● 属性の設定

次に示すコマンドを実行して，アプリケーション属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -c <アプリケーション属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name account -c C:¥home¥MyAccount.xml
```

(3) 編集する属性設定項目

J2EE アプリケーションの起動順序設定項目を次に示します。

項目	必須	対応するタグ名
開始・停止順※	△	<start-order>

（凡例） △：任意
注※ 値が小さいほど，先に起動します。また，値が大きいほど，先に停止します。なお，複数の J2EE アプリケーションに対して同じ値を指定した場合，それらの間の起動順序は保証されません。

プロパティの設定項目については，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「3.2.1 アプリケーション属性ファイルの指定内容」を参照してください。

9.15.2 Enterprise Bean の起動順序の設定

Enterprise Bean の起動順序を設定します。

（1） 編集する属性ファイル

- Session Bean 属性ファイル
- Entity Bean 属性ファイル
- Message-driven Bean 属性ファイル

（2） 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次のコマンドを実行して，編集する Enterprise Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Bean属性ファイルパス※>
```

注※ Enterprise Bean 属性ファイルパスには，編集する属性ファイルパスを指定します。

実行例

```
cjgetappprop MyServer -name account_eb -type ejb -resname account/account -c C:¥home¥MyAccount.xml
```

● 属性の設定

次のコマンドを実行して，編集した Enterprise Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Bean属性ファイルパス※>
```

注※ Enterprise Bean 属性ファイルパスには，値を反映する属性ファイルパスを指定します。

実行例

```
cjsetappprop MyServer -name account -type ejb -resname account/account_eb -c C:¥home¥MyAccount.xml
```

(3) 編集する属性設定項目

Enterprise Bean の起動順序設定項目を次に示します。

項目	必須	対応するタグ名
開始・停止順※	△	<start-order>

（凡例） △：任意
注※ 値が小さいほど，先に起動します。また，値が大きいほど，先に停止します。なお，Enterprise Bean に対して同じ値を指定した場合，それらの間の起動順序は保証されません。

プロパティの設定項目については，次の個所を参照してください。

- 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「3.4.1 Session Bean 属性ファイルの指定内容」
- 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「3.5.1 Entity Bean 属性ファイルの指定内容」
- 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「3.6.1 MessageDrivenBean 属性ファイルの指定内容」

9.15.3 サーブレットと JSP の起動順序の設定

サーブレット，JSP が含まれる WAR の起動順序を設定します。

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次のコマンドを実行して，次の WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name account -type war -resname account_war -c C:¥home¥account_war.xml
```

● 属性の設定

次のコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name account -type war -resname account_war -c C:¥home¥account_war.xml
```

(3) 編集する属性設定項目

サーブレット，JSP が含まれる WAR の起動順序設定項目を次に示します。

項目	必須	対応するタグ名
開始・停止順※	△	<start-order>

（凡例） △：任意
注※ 値が小さいほど，先に起動します。また，値が大きいほど，先に停止します。なお，サーブレットと JSP に対して同じ値を指定した場合，それらの間の起動順序は保証されません。

プロパティの設定項目については，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「3.7.1 WAR 属性ファイルの指定内容」を参照してください。

9.16　サーブレットと JSP のエラー通知の設定

次のタイミングでエラーが発生した場合に，エラーを通知して J2EE アプリケーションの開始処理を中止するかどうかを指定します。

- J2EE アプリケーションの開始時にロードするように設定されている（<load-on-startup> が設定されている）サーブレットまたは JSP の初期化処理中にエラーが発生した場合
- タグライブラリ解析時にエラーが発生した場合

(1) 編集する属性ファイル

WAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次のコマンドを実行して，次の WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjgetappprop MyServer -name account -type war -resname account_war -c C:¥home¥account_war.xml
```

● 属性の設定

次のコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL＞］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjsetappprop MyServer -name account -type war -resname account_war -c C:¥home¥account_war.xml
```

(3) 編集する属性設定項目

サーブレット，JSP が含まれる WAR のエラー通知の設定項目を次に示します。

項目	必須	対応するタグ名
開始時のエラー通知有無	△	<start-notify-error>

（凡例） △：任意

(4) 注意事項

● Web アプリケーション初期化失敗時のエラー通知の設定可否

J2EE サーバモードでは，Web アプリケーションの初期化で失敗した場合に，J2EE アプリケーションの開始処理を継続するか，エラーを通知して J2EE アプリケーションの開始処理を中止するかを設定できます。ただし，Web アプリケーションを利用不可とする必要があるエラーが発生した場合は，エラー通知の設定はできません。エラー通知の設定内容に関係なく，J2EE アプリケーションの開始処理が中止されます。ここでは，Web アプリケーションの初期化失敗の内容とエラー通知設定の可否について説明します。Web アプリケーションの初期化中に発生するエラーの内容と，J2EE アプリケーションの開始処理を継続するか，またはエラーとして中止するかのエラー通知の設定可否を次の表に示します。

表 9-5 Web アプリケーションの初期化失敗の内容とエラー通知の設定可否

項目	Web アプリケーション初期化失敗の内容	エラー通知の設定可否
JSP 用テンポラリディレクトリ	JSP 用テンポラリディレクトリのアクセス権がないため，ディレクトリの生成に失敗した場合。	不可
	JSP 用テンポラリディレクトリのアクセス権がないため，ディレクトリの削除に失敗した場合。	可
web.xml	web.xml の解析でエラーが発生した場合。	不可
フィルタ	web.xml の filter-class 要素に指定したフィルタクラスまたはそれが依存するクラスを見つけることができなかった場合。	不可
	web.xml の filter-class 要素に，次に示す不正なフィルタクラスが指定された場合。 1. javax.servlet.Filter インタフェースを実装していないクラス 2. 引数を取らないコンストラクタを持っていないクラス	不可
	フィルタクラスの初期化で例外が発生した場合。	不可

項目	Web アプリケーション初期化失敗の内容	エラー通知の設定可否
リスナ	web.xml の listener-class 要素に指定したリスナクラス，またはそれが依存するクラスを見つけることができなかった場合。	不可
	web.xml の listener-class 要素に，次に示す不正なリスナクラスが指定された場合。 1. 次に示すインタフェースを一つも実装していないクラス javax.servlet.ServletContextListener javax.servlet.ServletContextAttributeListener javax.servlet.ServletRequestListener javax.servlet.ServletRequestAttributeListener javax.servlet.http.HttpSessionListener javax.servlet.http.HttpSessionAttributeListener 2. 引数を取らないコンストラクタを持っていないクラス	不可
	リスナクラスの初期化で例外が発生した場合。	不可
web.xml で <load-on-startup> を指定したサーブレット	web.xml の servlet-class 要素に指定したサーブレットクラス，またはそれが依存するクラスを見つけることができなかった場合。	可
	サーブレットの初期化で例外が発生した場合。	可
web.xml で <load-on-startup> を指定した JSP	web.xml の jsp-file 要素に指定した JSP ファイルを見つけることができなかった場合。	可
	JSP 用テンポラリディレクトリにサブディレクトリを作成する権限がないため，JSP から Java ソースファイルを生成できなかった場合。	可
	JSP 用テンポラリディレクトリのサブディレクトリにアクセスする権限がないため，JSP から Java ソースファイルを生成できなかった場合。	可
	JSP から Java ソースファイルを生成する際にエラーが発生した場合。	可
	JSP から生成されたサーブレットのソースコードのコンパイルで，エラーが発生した場合。	可
	Servlet2.3 以前の Web アプリケーションで，web.xml の <taglib> タグで taglib をマッピングしていない，かつ JSP の taglib ディレクティブの uri 属性に絶対 URI を指定した場合。	可
	JSP ドキュメントの解析中にエラーが発生した場合。	可
	JSP の初期化で例外が発生した場合。	可
タグライブラリ（<load-on-startup> を指定した JSP の延長で実行される場合）	TLD ファイルの読み込みができなかった場合。	可
	TLD ファイルの解析中にエラーが発生した場合。	可
	タグライブラリバリデータクラスまたはそれが依存するクラスを見つけることができなかった場合。	可

項目	Web アプリケーション初期化失敗の内容	エラー通知の設定可否
	次に示す不正なタグライブラリバリデータクラスが指定された場合。 1. javax.servlet.jsp.tagext.TagLibraryValidator クラスを継承していないクラス 2. 引数を取らないコンストラクタを持っていないクラス	可
	タグライブラリバリデータクラスの初期化で例外が発生した場合。	可
	タグライブラリバリデータによる JSP の検証で例外が発生した場合。	可
	TagExtraInfo クラス，またはそれが依存するクラスを見つけることができなかった場合。	可
	不正な TagExtraInfo クラスが指定された場合。不正な TagExtraInfo クラスとは，Tei-class 要素または teiclass 要素に指定するクラスが，次のどれかの条件に一致するクラスを指す。 • TagExtraInfo クラスを継承していない • インタフェースや abstract クラスである • public なコンストラクタを持っていない	可
	TLD ファイルの tei-class 要素または teiclass 要素で定義したクラスの初期化で例外が発生した場合。	可
	タグライブラリバリデータクラスが JSP ページの検証で発見したエラーを報告した場合。	可
	TagExtraInfo クラスが属性の検証で発見したエラーを報告した場合。	可
	TagExtraInfo クラスによる属性の検証で例外が発生した場合。	可
	TLD ファイルの function-class 要素で定義したクラスを見つけることができなかった場合。	可
	TLD ファイルの function-class 要素で定義したクラスのロードで例外が発生した場合。	可
	TLD ファイルの function-signature 要素で定義した parameter クラスを見つけることができなかった場合。	可
	TLD ファイルの function-signature 要素で定義したメソッドを見つけることができなかった場合。	可
	TLD ファイルの function-signature 要素で定義した parameter クラスのロード，または function-class 要素で指定したクラスへのアクセス時に例外が発生した場合。	可
	TLD ファイルの type 要素で定義したクラスを見つけることができなかった場合。	可
	TLD ファイルの type 要素で定義したクラスのロードで例外が発生した場合。	可

9.17 セキュリティロールの設定

セキュリティロールを使用したユーザ管理をする場合に必要な設定です。

設定したユーザおよびロールの情報は，J2EE サーバごとに管理されます。

9.17.1 ユーザの設定

ユーザを設定します。

次に示すコマンドを実行して，J2EE サーバにユーザを登録します。

実行形式

```
cjaddsec ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type user -name <ユーザ名> -password <パスワード>
```

実行例

```
cjaddsec MyServer -type user -name cosmi -password tiger
```

cjaddsec コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjaddsec（ユーザとロールの追加）」を参照してください。

9.17.2 ロールの設定

ロールを設定して，ユーザと関連づけます。また，Enterprise Bean およびサーブレットと JSP に参照するセキュリティロールを設定します。

(1) ロールの登録

次に示すコマンドを実行して，J2EE サーバにロールを登録します。

実行形式

```
cjaddsec ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -type role -name <ロール名>
```

実行例

```
cjaddsec MyServer -type role -name manage
```

cjaddsec コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjaddsec（ユーザとロールの追加）」を参照してください。

(2) ロールにユーザを登録

次に示すコマンドを実行してロールにユーザを追加します。

実行形式

```
cjmapsec ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -role <ロール名> -user
<ユーザ名> ［-user <ユーザ名>］
```

実行例

```
cjmapsec MyServer -role manager -user cosmi
```

cjmapsec コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjmapsec（ユーザとロールのマッピング)」を参照してください。

(3) Enterprise Bean へのセキュリティロールの設定

Enterprise Bean へのセキュリティロールの設定を定義します。

(a) 編集する属性ファイル

EJB-JAR 属性ファイル

(b) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して EJB-JAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリ
ケーション名> -type ejb -resname <EJB-JAR表示名> -c <EJB-JAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder -c
C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，EJB-JAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリ
ケーション名> -type ejb -resname <EJB-JAR表示名> -c <EJB-JAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder -c
C:¥home¥adder_ejb.xml
```

(c) 編集する属性設定項目

Enterprise Bean のセキュリティロール（<security-role>）の設定項目を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
ロール名	○	<role-name>
セキュリティロール名	△	<linked-to>

（凡例） ○：必須 △：任意

プロパティの設定項目については，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「3.3.1 EJB-JAR 属性ファイルの指定内容」を参照してください。

(4) サーブレットおよび JSP へのセキュリティロールの設定

サーブレットおよび JSP へのセキュリティロールの設定を定義します。

(a) 編集する属性ファイル

WAR 属性ファイル

(b) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder -c
C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -name ＜J2EEアプリケーション名＞ -type war -resname ＜WAR表示名＞ -c ＜WAR属性ファイルパス＞
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder -c C:¥home¥adder_war.xml
```

(c) 編集する属性設定項目

Web アプリケーション（サーブレットおよび JSP）のセキュリティロールのリファレンス（<security-role>）の設定項目を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
ロール名	○	<role-name>
セキュリティロール名	△	<linked-to>

（凡例） ○：必須 △：任意

プロパティの設定項目については，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「3.7.1 WAR 属性ファイルの指定内容」を参照してください。

9.18 セキュリティロールのリファレンス定義

J2EE アプリケーションに含まれる Enterprise Bean および WAR の，一つまたは複数のメソッドに対するセキュリティロールチェックの参照を定義します。このセキュリティチェックは，コンテナで提供されるセキュリティサービスとは別のものです。

9.18.1 Enterprise Bean のセキュリティロールリファレンスの定義

Enterprise Bean のセキュリティロールのリファレンスを定義します。

(1) 編集する属性ファイル

Enterprise Bean の種類に対応する属性ファイルを編集します。

- Session Bean 属性ファイル
- Entity Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次のコマンドを実行して，Enterprise Bean 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Beanの属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder/adder_eb -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次のコマンドを実行して，Enterprise Bean 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Beanの属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder/
```

```
adder_eb -c C:¥home¥adder_ejb.xml
```

(3) 編集する属性設定項目

Enterprise Bean のセキュリティロールリファレンス（<security-role-ref>）の設定項目を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
セキュリティロール参照名	○	<role-name>
リンク先のセキュリティロール名※	△	<role-link>

（凡例） ○：必須 △：任意
注※ 設定したロール名を指定します。ロール名の設定については，「9.17.2 ロールの設定」を参照してください。また，<role-link> を設定後，EJB-JAR 属性ファイルの設定を行うと <role-link> の値がクリアされますので，再度，セキュリティロールリファレンスを設定してください。

プロパティの設定項目については，次の個所を参照してください。

- 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「3.4.1 Session Bean 属性ファイルの指定内容」
- 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「3.5.1 Entity Bean 属性ファイルの指定内容」

9.18.2 サーブレットと JSP のセキュリティロールリファレンスの定義

Web アプリケーション（サーブレットおよび JSP）のセキュリティロールのリファレンスを定義します。

(1) 編集する属性ファイル

サーブレット属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してサーブレット属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名>/<サーブレットおよびJSPの表示名> -c <サーブレット属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder/
adder_sv -c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名>/<サーブレットおよびJSPの表示名> -c <サーブレット属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder/
adder_sv -c C:¥home¥adder_war.xml
```

(3) 編集する属性設定項目

Web アプリケーション（サーブレットおよび JSP）のセキュリティロールのリファレンス（<security-role-ref>）の設定項目を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
セキュリティロール参照名	○	<role-name>
リンク先のセキュリティロール名※	△	<role-link>

（凡例） ○：必須 △：任意
注※ 設定したロール名を指定します。ロール名の設定については，「9.17.2 ロールの設定」を参照してください。

プロパティの設定項目については，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「3.9.1 サーブレット属性ファイルの指定内容」を参照してください。

9.19 セキュリティの定義（メソッドパーミッション）

メソッドパーミッションの設定方法について説明します。メソッドパーミッションの定義では，セキュリティロールによるアクセス制御を定義します。すべてのユーザにアクセス権限を与えることや，どのユーザにもアクセス権限を与えないこともできます。

メソッドパーミッションを設定できるメソッドは次のとおりです。

● Session Bean
- ホームインタフェースの create メソッド
- コンポーネントインタフェースのビジネスメソッド，remove メソッド

● Entity Bean
- ホームインタフェースの create メソッド，finder メソッド，home メソッド
- コンポーネントインタフェースのビジネスメソッド，remove メソッド

なお，次に示すメソッドにパーミッションを設定しても使用されません。これらのメソッドのアクセス権限チェックには，コンポーネントインタフェースの remove メソッドに設定したメソッドパーミッションが使用されます。

- javax.ejb.EJBHome の remove(javax.ejb.Handle handle) メソッド
- javax.ejb.EJBHome の remove(Object primaryKey) メソッド
- javax.ejb.EJBLocalHome の remove(Object primaryKey) メソッド

注意事項

CTM を使用するアプリケーションで <Enable Scheduling> プロパティが指定された Stateless Session Bean では，ホームインタフェースの create メソッドにセキュリティロールによるアクセス権限を設定しないでください。設定した場合，デプロイに失敗します。

(1) 編集する属性ファイル

次の Enterprise Bean の種類に対応する属性ファイルを編集します。

- Session Bean 属性ファイル
- Entity Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Enterprise Bean の属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Beanの属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder/adder-eb -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，Enterprise Bean の属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Beanの属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder/adder-eb -c C:¥home¥adder_ejb.xml
```

(3) 編集する属性設定項目

セキュリティの定義（メソッドパーミッション）の設定項目（<method-permission>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
ロール名	△※	<role-name>
メソッド認証有無	△※	<unchecked>
メソッドの説明	△	<method> - <description>
インタフェース種別	△	<method> - <intf>
メソッド名	△	<method> - <name>

（凡例） △：任意

注　セキュリティの定義（メソッドパーミッション）の設定項目（<method-permission>）は，アノテーションで設定している場合，変更できません。

注※　セキュリティ管理をする場合，ロール名とメソッド認証有無は，次のように，どちらか一つ設定します。

- セキュリティロールによるアクセス権限を設定する場合
  ロール名（<role-name>）を指定します。
- すべてのユーザにアクセス権限を与える場合

メソッド認証有無（<unchecked>）を指定します。

どのユーザにもアクセス権限を与えない場合は，<method-permission> で指定するのではなく，<exclude-list> の下の <method> にアクセス権を与えないメソッドの情報を指定します。

プロパティの設定項目については,，次の個所を参照してください。

- 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「3.4.1　Session Bean 属性ファイルの指定内容」
- 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「3.5.1　Entity Bean 属性ファイルの指定内容」

9.20 セキュリティの定義（セキュリティアイデンティティ）

セキュリティアイデンティティの設定には，次の二つの場合があります。

- Enterprise Bean が使用する実行時アイデンティティ情報を設定します。
- サーブレットが使用する実行時アイデンティティ情報を設定します。

9.20.1 Enterprise Bean のセキュリティアイデンティティ

Enterprise Bean のセキュリティアイデンティティを定義します。

セキュリティアイデンティティとしては，「UseCallerIdentity」と「RunAs」の二つのタイプを設定できます。

- UseCallerIdentity
  メソッド実行時に呼び出し元セキュリティアイデンティティを使用し，動作します。Enterprise Bean のホームインタフェースやコンポーネントインタフェースのメソッド実行時に，その実行スレッドに関連づけるセキュリティアイデンティティを設定します。
- RunAs
  Role name で指定したロールのアイデンティティに従い，動作します。

(1) 編集する属性ファイル

次の Enterprise Bean の種類に対応する属性ファイルを編集します。

- Session Bean 属性ファイル
- Entity Bean 属性ファイル
- Message-driven Bean 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Enterprise Bean の属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名>-c <Enterprise Bean属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname addr/adder_eb -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，Enterprise Bean の属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Bean属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder/adder_eb -c C:¥home¥adder_ejb.xml
```

(3) 編集する属性設定項目

Enterprise Bean のセキュリティの定義（セキュリティアイデンティティ）の設定項目（<security-identity>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
セキュリティアイデンティティ設定有無	△※	<use-caller-identity>
ロールのアイデンティティの説明	△	<run-as> - <description>
セキュリティロールの名称	△※	<run-as> - <role-name>
セキュリティロールに設定された名称	△	<run-as> - <user-name>

（凡例） △：任意

注※ セキュリティアイデンティティの設定をする場合，メソッド実行時に呼び出し元セキュリティアイデンティティを使用するかどうかで，次のように，どちらか一つ設定します。

- メソッド実行時に呼び出し元セキュリティアイデンティティを使用する場合
  セキュリティアイデンティティ設定有無（<use-caller-identity>）を設定します。
- メソッド実行時に呼び出し元セキュリティアイデンティティを使用しない場合
  ロールのアイデンティティ（<run-as>）情報を設定します。
- Message-driven Bean の場合は，ロールのアイデンティティ（<run-as>）情報だけ設定できます。

プロパティの設定項目については，次の個所を参照してください。

- 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「3.4.1 Session Bean 属性ファイルの指定内容」
- 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「3.5.1 Entity Bean 属性ファイルの指定内容」
- 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「3.6.1　MessageDrivenBean 属性ファイルの指定内容」

9.20.2　サーブレットと JSP のセキュリティアイデンティティ

サーブレットと JSP のセキュリティアイデンティティを定義します。

サーブレットが EJB 呼び出し時に使用する実行時アイデンティティ情報を設定します。

（1）編集する属性ファイル

サーブレット属性ファイル

（2）編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行してサーブレット属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名>/<サーブレットおよびJSPの表示名> -c <サーブレット属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder/adder_sv -c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，サーブレット属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名>/<サーブレットおよびJSPの表示名> -c <サーブレット属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder/adder_sv -c C:¥home¥adder_war.xml
```

（3）編集する属性設定項目

Web アプリケーション（サーブレットおよび JSP）のセキュリティの定義（セキュリティアイデンティティ）の設定項目（<security-identity>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
ロールのアイデンティティの説明	△	<run-as> - <description>
セキュリティロールの名称	○	<run-as> - <role-name>
セキュリティロールに設定された名称	△	<run-as> - <name>

（凡例） ○：必須　△：任意

プロパティの設定項目については，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「3.9.1　サーブレット属性ファイルの指定内容」を参照してください。

9.21 インターセプタの設定

インターセプタの設定方法について説明します。Cosminexus でインターセプタを使用する場合は，EJB-JAR の属性として設定してください。

(1) 編集する属性ファイル

EJB-JAR 属性ファイル

(2) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して EJB-JAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名> -c <EJB-JAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname adder -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，EJB-JAR 属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名> -c <EJB-JAR属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname adder -c C:¥home¥adder_ejb.xml
```

注意事項

開始状態のアプリケーションに含まれる EJB-JAR 属性ファイルの取得はできますが，定義された関連情報の反映はできません。

(3) 編集する属性設定項目

インターセプタの設定項目（<interceptor-binding>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
インターセプタの EJB 名	○	<ejb-name>
インターセプタクラスのクラス名	△	<interceptor-class>
インターセプタの実行順序の設定（インターセプタクラスのクラス名）	△	<interceptor-order>-<interceptor-class>
デフォルトインターセプタの呼び出し抑止の設定	△	<exclude-default-interceptors>
クラスレベルインターセプタの呼び出し抑止の設定	△	<exclude-class-interceptors>
EJB のビジネスメソッド名	△	<named-method>-<method-name>
メソッドの引数	△	<named-method>-<method-params>-<method-param>

（凡例） ○：必須　△：任意

プロパティの設定項目については，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「3.3.1　EJB-JAR 属性ファイルの指定内容」を参照してください。

インターセプタの設定方法や注意事項については，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（EJB コンテナ）」の「2.15　インターセプタの使用」を参照してください。

9.22　そのほかのプロパティの設定

J2EE アプリケーションの作成およびカスタマイズで設定する，そのほかのプロパティ項目については，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)」を参照してください。

10 J2EE アプリケーションの実行

この章では，サーバ管理コマンドを使用した J2EE アプリケーションの実行について説明します。

10.1 J2EE アプリケーションの実行の概要

J2EE アプリケーションとして必要なプロパティや動作の定義が終わったら，J2EE アプリケーションは実行できます。

J2EE アプリケーションの実行の概要について次に示します。

表 10-1 J2EE アプリケーションの実行の概要

設定項目	内容	J2EE アプリケーションの形式		参照先
		アーカイブ形式	展開ディレクトリ形式	
J2EE アプリケーションの開始と停止	J2EE アプリケーションを，通常開始または JSP をコンパイルして開始します。 J2EE アプリケーションを，通常停止または強制停止します。	○	○	10.2
J2EE アプリケーションの一覧の参照	インポートされているアプリケーションの一覧を参照します。	○	○	10.3
J2EE アプリケーションの削除	J2EE アプリケーションを削除します。	○	△※1	10.4
テストモードによる J2EE アプリケーションの実行	実行中のアプリケーションをテストモードに切り替えます。	○	○	10.5
J2EE アプリケーションの入れ替え	J2EE サーバ上の，次の J2EE アプリケーションを入れ替えます。 • アーカイブ形式のアプリケーション • 展開ディレクトリ形式のアプリケーション	△※2	△※2	10.6
J2EE アプリケーション名の変更	J2EE アプリケーションの名前を変更します。	○	○	10.7
RMI-IIOP スタブとインタフェースの取得	J2EE アプリケーションの RMI-IIOP スタブおよびインタフェースを取得します。	○	○	10.8

設定項目	内容	J2EE アプリケーションの形式		参照先
		アーカイブ形式	展開ディレクトリ形式	
トランザクション一覧の参照	トランザクションの情報の一覧を参照します。 J2EE サーバが稼働中の場合は，稼働状況や，開始してからの経過時間などのトランザクションの情報を表示します。 J2EE サーバが停止中の場合は，ステータスファイルにある，仕掛かり中のまま残っている未決着トランザクションの状況を表示します。	○	○	10.9

（凡例） ○：実行できる　△：実行条件がある

注※ 1　アプリケーションからコンポーネントの削除はできません。

なお，フィルタの追加は，アプリケーションの属性変更であるため実行できます。

注※ 2　J2EE アプリケーションの入れ替えは，次のコマンドを使用します。

- アーカイブ形式の場合　cjreplaceapp コマンド
- 展開ディレクトリ形式の場合　cjreloadapp コマンド

10.2 J2EE アプリケーションの開始と停止

ここでは，J2EE アプリケーションの開始と停止について説明します。

10.2.1 J2EE アプリケーションの開始

ここでは，次の J2EE アプリケーションの開始方法について説明します。

- 通常開始
- JSP をコンパイルして開始

(1) J2EE アプリケーションの通常開始

次に示すコマンドを実行して J2EE アプリケーションを開始します。J2EE アプリケーションにリソースアダプタが含まれている場合，J2EE アプリケーションに含まれるすべてのリソースアダプタも自動で開始されます。

実行形式

```
cjstartapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名>
```

実行例

```
cjstartapp MyServer -name account
```

cjstartapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstartapp（J2EE アプリケーションの開始）」を参照してください。

注意事項

- J2EE アプリケーションが，J2EE リソースアダプタを参照している場合，J2EE リソースアダプタを開始してから，J2EE アプリケーションを開始してください。
- J2EE アプリケーションに含まれるリソースアダプタは，開始順序を制御はできません。
- ほかの J2EE アプリケーションに含まれる Enterprise Bean を呼び出す場合，呼び出し先の J2EE アプリケーションを開始してから，呼び出し元の J2EE アプリケーションを開始してください。

(2) JSP をコンパイルして開始

J2EE アプリケーションに含まれるすべての JSP ファイルのコンパイルを実行して，J2EE アプリケーションを開始します。

次に示すコマンドを実行します。

実行形式

```
cjstartapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -jspc
```

実行例

```
cjstartapp MyServer -name App1 -jspc
```

cjstartapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstartapp（J2EE アプリケーションの開始）」を参照してください。

注意事項

アーカイブ形式のアプリケーションの場合，アプリケーションの開始時に JSP 事前コンパイルを実施して生成された JSP コンパイル結果は，アプリケーションの停止時に削除されます。

アプリケーション開始時の JSP 事前コンパイルで生成した JSP コンパイル結果をアプリケーションの停止後も利用する場合，次の手順を実行します。

1. JSP の事前コンパイルを指定してアプリケーションを開始します。
2. アプリケーションをエクスポートします。
3. リデプロイ機能などを使用して，JSP コンパイル結果を含むアプリケーションに入れ替えます。
   J2EE アプリケーションの入れ替えについては，「10.6　J2EE アプリケーションの入れ替え」を参照してください。

また，アプリケーション開始時の JSP 事前コンパイルを実行したあとに，アプリケーションの開始に失敗した場合，アプリケーションは停止し，JSP コンパイル結果は削除されます。アプリケーション開始時の JSP 事前コンパイルを実行する前に，アプリケーションが正常に開始できることを確認してください。

10.2.2　J2EE アプリケーションの停止

ここでは，J2EE アプリケーションの通常停止と強制停止について説明します。

(1) J2EE アプリケーションの通常停止

次に示すコマンドを実行して，J2EE アプリケーションを通常停止します。J2EE アプリケーションにリソースアダプタが含まれている場合，J2EE アプリケーションに含まれるすべてのリソースアダプタも自動で停止されます。

実行形式

```
cjstopapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名>
```

実行例

```
cjstopapp MyServer -name account
```

cjstopapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstopapp（J2EE アプリケーションの停止）」を参照してください。

注意事項

- J2EE アプリケーションが J2EE リソースアダプタとしてデプロイしたリソースアダプタを参照している場合，J2EE アプリケーションを停止してから，J2EE リソースアダプタを停止してください。
- J2EE アプリケーションに含まれるリソースアダプタは，停止順序を制御はできません。
- ほかの J2EE アプリケーションに含まれる Enterprise Bean を呼び出す場合，呼び出し元の J2EE アプリケーションを停止してから，呼び出し先の J2EE アプリケーションを停止してください。
- アーカイブ形式のアプリケーションの場合，アプリケーション開始時の JSP 事前コンパイルを実行して生成されたコンパイル結果は，アプリケーションの停止時に削除されます。

(2) J2EE アプリケーションの強制停止

J2EE アプリケーションを強制停止します。

J2EE アプリケーションの強制停止には，次の二つがあります。

- J2EE アプリケーションの通常停止を実行して，J2EE アプリケーションが停止しなかった場合に強制停止を実行します。
- 停止しなかった場合に自動に強制停止を実行するオプションを指定して，J2EE アプリケーションの通常停止を実行します。

それぞれの強制停止の手順について，次に示します。

(a) J2EE アプリケーションの通常停止を実行して J2EE アプリケーションが停止しなかった場合に強制停止を実行する

「(1) J2EE アプリケーションの通常停止」の手順で，J2EE アプリケーションの通常停止を実行しても J2EE アプリケーションが正常に停止しなかった場合，次の手順で J2EE アプリケーションを強制停止できます。

なお，J2EE アプリケーションの強制停止は，通常停止を実行しても J2EE アプリケーションが正常に停止しなかった場合にだけ実行してください。

次に示すコマンドを実行して J2EE アプリケーションを強制停止します。

実行形式

```
cjstopapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -cancel
```

実行例

```
cjstopapp MyServer -name account -cancel
```

cjstopapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstopapp（J2EE アプリケーションの停止）」を参照してください。

(b) J2EE アプリケーションの自動強制停止オプションを指定して，通常停止を実行する

J2EE アプリケーションの通常停止を実行しても J2EE アプリケーションが正常に停止しなかった場合，タイムアウト時間が経過すると自動的に強制停止を実行できます。

次のコマンドを実行して，通常停止しない J2EE アプリケーションを自動的に強制停止します。

実行形式

```
cjstopapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -t <タイムアウト時間（秒）> -force
```

実行例

```
cjstopapp MyServer -name account -t 120 -force
```

cjstopapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjstopapp（J2EE アプリケーションの停止）」を参照してください。

10.3　J2EE アプリケーションの一覧の参照

次のコマンドを実行して，インポートされている J2EE アプリケーションの一覧を参照します。

J2EE アプリケーションが展開ディレクトリ形式の場合，アプリケーションディレクトリのパスを表示します。

実行形式

```
cjlistapp ［<サーバ名称>］
```

実行例

```
cjlistapp MyServer
```

cjlistapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlistapp（アプリケーションの一覧表示）」を参照してください。

10.4　J2EE アプリケーションの削除

J2EE アプリケーションを削除します。

次に示すコマンドを実行して J2EE アプリケーションを削除します。

実行形式

```
cjdeleteapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名>
```

実行例

```
cjdeleteapp MyServer -name account
```

cjdeleteapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjdeleteapp（J2EE アプリケーションの削除）」を参照してください。

注意事項

展開ディレクトリ形式の J2EE アプリケーションを削除する場合，削除対象の J2EE アプリケーションのアプリケーションディレクトリは削除されません。J2EE サーバへの登録が解除されます。

10.5　テストモードによる J2EE アプリケーションの実行

テストモードで J2EE アプリケーションを実行することで，J2EE アプリケーションを本番環境で問題なく実行できるかを確認します。

次の二つの場合について説明します。

- 新規作成した J2EE アプリケーションをテストモードで実行します。
- J2EE アプリケーションを入れ替える場合にテストモードを使用します。

(1) 新規作成した J2EE アプリケーションをテストモードで実行する

新規作成した J2EE アプリケーションをテストモードで実行します。また，本番環境で問題なく実行できることを確認してから通常モードに切り替えます。

1. **J2EE アプリケーションをテストモードでインポートします。**
   J2EE アプリケーションをテストモードでインポートするには，-test オプションを指定した cjimportapp コマンドを実行します。J2EE アプリケーションのインポート方法については，「8.1　J2EE アプリケーションのインポート」を参照してください。

2. **J2EE アプリケーションをテストモードで開始します。**
   J2EE アプリケーションをテストモードで開始するには，-test オプションを指定した cjstartapp コマンドを実行します。J2EE アプリケーションの開始方法については「10.2.1　J2EE アプリケーションの開始」を参照してください。

3. **テストを実施します。**
   実際に J2EE アプリケーションを操作してテストします。問題がないことを確認したら次の手順に進みます。
   テスト結果に問題があった場合は，J2EE アプリケーションを修正して，手順 1. からもう一度実行してください。なお，同じ名称の J2EE アプリケーションは同じモードで登録できません。このため，テストが失敗したテストモードの J2EE アプリケーションは削除してください。テストモードの J2EE アプリケーションを削除するには，-test オプションを指定した cjdeleteapp コマンドを実行します。

4. **テストモードで実行している J2EE アプリケーションを停止します。**
   テストモードで実行中の J2EE アプリケーションを停止するには，-test オプションを指定した cjstopapp コマンドを実行します。J2EE アプリケーションの停止方法については「10.2.2　J2EE アプリケーションの停止」を参照してください。

5. **次に示すコマンドを実行して，J2EE アプリケーションのモードを，テストモードから通常モードへ切り替えます。**

   実行形式

```
cjchmodapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -test -name <アプリケーション名> -mode normal
```

実行例

```
cjchmodapp MyServer -test -name App1 -mode normal
```

cjchmodapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjchmodapp（アプリケーションのモード切り替え)」を参照してください。

6. **テストモードから通常モードへ切り替えた J2EE アプリケーションを開始します。**
J2EE アプリケーションの開始方法については「10.2.1　J2EE アプリケーションの開始」を参照してください。

(2) J2EE アプリケーションを入れ替える場合にテストモードを使用する

J2EE アプリケーションを入れ替える前に，入れ替える J2EE アプリケーションをテストモードで実行します。また，本番環境で問題なく実行できることを確認してから，既存の J2EE アプリケーションと入れ替えます。

注意事項

J2EE サーバ内で同じアプリケーションディレクトリを持つ複数の J2EE アプリケーションは共存できません。展開ディレクトリ形式のアプリケーションの場合はテストモードのアプリケーションと通常モードのアプリケーションでアプリケーションディレクトリを分けてください。

1. **J2EE アプリケーションをテストモードでインポートします。**
J2EE アプリケーションをテストモードでインポートするには，-test オプションを指定した cjimportapp コマンドを実行します。J2EE アプリケーションのインポート方法については，「8.1　J2EE アプリケーションのインポート」を参照してください。

2. **J2EE アプリケーションをテストモードで開始します。**
J2EE アプリケーションをテストモードで開始するには，-test オプションを指定した cjstartapp コマンドを実行します。J2EE アプリケーションの開始方法については「10.2.1　J2EE アプリケーションの開始」を参照してください。

3. **テストを実施します。**
実際にアプリケーションを操作してテストします。問題がないことを確認したら次の手順に進みます。
テスト結果に問題があった場合は，J2EE アプリケーションを修正して，手順 1. からもう一度実行してください。なお，同じ名称の J2EE アプリケーションは同じモードで登録できません。このため，テストが失敗したテストモードの J2EE アプリケー

ションは削除してください。テストモードの J2EE アプリケーションの削除は，-test オプションを指定した cjdeleteapp コマンドを実行します。

4. **テストモードで実行している J2EE アプリケーションを停止します。**
テストモードで実行中の J2EE アプリケーション停止するには，-test オプションを指定した cjstopapp コマンドを実行します。J2EE アプリケーションの停止方法については「10.2.2　J2EE アプリケーションの停止」を参照してください。

5. **通常モードで実行している J2EE アプリケーションを停止します。**
J2EE アプリケーションの停止方法については「10.2.2　J2EE アプリケーションの停止」を参照してください。

6. **通常モードで実行していた J2EE アプリケーションの名称を変更します。**
この操作は，インポートしたファイルの世代を管理したい場合などに，必要に応じて実行してください。例えば，「App1」という J2EE アプリケーションの場合，「App1_bak」などに変更します。
J2EE アプリケーション名の変更方法については，「10.7　J2EE アプリケーション名の変更」を参照してください。

7. **次に示すコマンドを実行して，テストモードで実行していた J2EE アプリケーションのモードを，テストモードから通常モードへ切り替えます。**

実行形式

```
cjchmodapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -test -name <アプリケーション名> -mode normal
```

実行例

```
cjchmodapp MyServer -test -name App1 -mode normal
```

cjchmodapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjchmodapp（アプリケーションのモード切り替え)」を参照してください。

8. **テストモードから通常モードへ切り替えた J2EE アプリケーションを開始します。**
J2EE アプリケーションの開始方法については「10.2.1　J2EE アプリケーションの開始」を参照してください。

10.6　J2EE アプリケーションの入れ替え

J2EE サーバモード上で動作する J2EE アプリケーションの入れ替えには，次の方法があります。

- J2EE アプリケーションを再デプロイします。
  J2EE アプリケーションを J2EE サーバから削除し，別の J2EE アプリケーションを入れ替え，開始します。
- リデプロイ機能を使用します。
  J2EE アプリケーションがアーカイブ形式の場合に使用できます。
- リロード機能を使用します。
  展開ディレクトリ形式の場合に使用できます。

この節では，次の J2EE アプリケーションの入れ替えについて説明します。

● リデプロイ機能を使用した，アーカイブ形式の J2EE アプリケーションの入れ替え

● リロード機能を使用した，展開ディレクトリ形式の J2EE アプリケーションの入れ替え

10.6.1　アーカイブ形式のアプリケーション

リデプロイ機能によって，J2EE サーバにインポートしたアーカイブ形式の J2EE アプリケーションを，別の J2EE アプリケーションに入れ替える手順を次に示します。

1. **更新する** Java **プログラムをコンパイルします。**
2. EJB-JAR，WAR **ファイルを再生成します。**
3. EAR **を再生成します。**
4. **リデプロイコマンドを実行します。**

次のリデプロイコマンドを実行して，アーカイブ形式の J2EE アプリケーション（EAR ファイル）を入れ替えます。

実行形式

```
cjreplaceapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <アプリケーション名> -f <EARファイルのパス> ［-t <タイムアウト時間（秒）>］ ［-replaceDD］
```

実行例

```
cjreplaceapp MyServer -name App1 -f App1.ear -t 120
```

J2EE アプリケーションの入れ替え時には，J2EE アプリケーションのすべての属性を引き継ぐことも，ランタイム属性（属性ファイルの独自の定義）だけを引き継ぐこともで

きます。ランタイム属性だけを引き継ぎたい場合は -replaceDD オプションを指定します。

cjreplaceapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjreplaceapp（アプリケーションの入れ替え）」を参照してください。

注意事項

- この操作は J2EE アプリケーションが開始，停止のどちらの状態であっても実行できます。開始状態の J2EE アプリケーションを入れ替える場合，cjreplaceapp コマンドを実行すると，処理の中で J2EE アプリケーションはいったん停止され，入れ替え後に再開始されます。
  なお，開始中状態の J2EE アプリケーションが正常に停止されなかった場合，J2EE アプリケーションは強制停止されます。-t オプションにタイムアウト時間を指定すると，J2EE アプリケーションの強制停止に移行するまでの時間を設定できます。-t オプションにタイムアウト時間を指定しなかった場合，強制停止に移行するまでの時間は 60 秒です。
- JSP 事前コンパイルを実行したアプリケーションの入れ替えで，入れ替え後も JSP 事前コンパイル機能を使用する場合，入れ替えるアプリケーションに JSP コンパイル結果が含まれていなければなりません。入れ替えるアプリケーションに JSP コンパイル結果が含まれていない場合，Web アプリケーション単位の JSP 事前コンパイルを実行し，入れ替えるアプリケーションに JSP コンパイル結果を含めてください。
- cosminexus.xml を含むアプリケーションにリデプロイ機能を使用した場合，入れ替え前の cosminexus.xml の定義情報を破棄してデフォルト値に戻したあと，入れ替え後の cosminexus.xml の定義情報を上書きします。そのため，実行環境でサーバ管理コマンドを使って変更した Cosminexus アプリケーションサーバ独自の定義情報は失われます。

10.6.2 展開ディレクトリ形式のアプリケーション

リロード機能を使用する場合は，アプリケーションの再生成（EAR ファイル作成）が不要となり，直接クラスファイルを置き換えるだけでよいため，アプリケーションの入れ替えが容易になります。

リロード機能は，展開ディレクトリ形式を使用し，開始状態およびリロードに失敗して停止状態にある J2EE アプリケーションに対してだけ実行できます。

リロード機能を実現するには，次の二つの手段があります。

- J2EE サーバが更新を自動的に検知することによるリロード機能
- コマンドの実行によるリロード機能

ここでは，コマンドの実行によるリロード機能について説明します。

リロード機能によって，J2EE アプリケーションに入れ替える手順を次に示します。

1. **更新する Java プログラムをコンパイルします。**

2. **リロードコマンドを実行します。**

次のリロードコマンドを実行して，展開ディレクトリ形式の J2EE アプリケーションを入れ替えます。

実行形式

```
cjreloadapp ［サーバ名］ -name アプリケーション名 ［-t 強制リロード開始までのタイムアウト時間］
```

実行例

```
cjreloadapp MyServer -name App1
```

cjreloadapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjreloadapp（アプリケーションのリロード）」を参照してください。

リロード機能の詳細および更新検知によるリロード機能については，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「12.8　J2EE アプリケーションの更新検知とリロード」を参照してください。

注意事項

- デフォルトの設定では，リロード機能は無効になっています。リロード機能を使用するためには，プロパティ設定ファイル（usrconf.properties）のキーを，次のように設定してください。
- リロード機能の適用範囲の設定
  ejbserver.deploy.context.reload_scope=app
- 更新検知インターバルの設定
  ejbserver.deploy.context.check_interval=1

J2EE アプリケーションの更新検知とリロードの設定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 基本・開発編 (コンテナ共通機能)」の「12.8.12　J2EE アプリケーションの更新検知とリロードの設定」を参照してください。

- リソースアダプタを含む J2EE アプリケーションの場合，RAR ファイルはアーカイブ形式で格納されています。アプリケーションディレクトリ下にある RAR ファイルが変更されてもリロードは実行されません。
- cosminexus.xml を含むアプリケーションにリロード機能を使った場合，cosminexus.xml の定義情報についてはリロードされません。

10.7　J2EE アプリケーション名の変更

J2EE サーバにインポートした J2EE アプリケーションの名称を変更します。

J2EE アプリケーションの名称を変更するには，J2EE アプリケーションを停止しておく必要があります。J2EE アプリケーションを停止する方法については「10.2.2　J2EE アプリケーションの停止」を参照してください。

次に示すコマンドを実行して J2EE アプリケーション名を変更します。

実行形式

```
cjrenameapp ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <アプリケーション名> -newname <変更後のアプリケーション名>
```

実行例

```
cjrenameapp MyServer -name App1 -newname App2
```

cjrenameapp コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjrenameapp（アプリケーション名の変更）」を参照してください。

10.8　RMI-IIOP スタブとインタフェースの取得

J2EE サーバにインポートした J2EE アプリケーションの RMI-IIOP スタブとインタフェースを取得します。

J2EE サーバにインポートした J2EE アプリケーションの RMI-IIOP スタブおよびインタフェースは，J2EE アプリケーションを一度も開始していない状態では取得できません。

また，J2EE アプリケーション内に，リモートインタフェースを持つ Enterprise Bean が存在しない場合，RMI-IIOP スタブおよびインタフェースの取得はエラーになります。

次に示すコマンドを実行して RMI-IIOP スタブおよびインタフェースを取得します。

実行形式

```
cjgetstubsjar ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <アプリケーション名> -d <RMI-IIOPスタブおよびインタフェースの格納パス>
```

実行例

```
cjgetstubsjar MyServer -name App1 -d temp
```

注意事項

● 次の条件を満たすときだけ，この機能を使用できます。
- J2EE サーバに旧バージョンで作成した J2EE アプリケーションが存在する状態で，06-50 以降のアプリケーションサーバをアップグレードインストールした場合
- cjrenameapp コマンドで名称を変更していない場合

● 次の場合は，index.html による RMI-IIOP スタブおよびインタフェースの取得はできません。
- 実行時情報を含む J2EE アプリケーションまたは実行時情報を含まない J2EE アプリケーションのどちらをインポートした場合も，実行できません。
- 06-50 以降の環境で新規に J2EE アプリケーションを作成した場合
- cjrenameapp コマンドで名称を変更した場合

● application.xml を省略したアプリケーションの名前を変更した場合，application.xml が作成されます。そのため，アプリケーション名を変更した J2EE アプリケーションは application.xml のあるアプリケーションとなります。

cjgetstubsjar コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjgetstubsjar（アプリケーションの RMI-IIOP スタブおよびインタフェースの取得）」を参照してください。

10.9 トランザクション一覧の参照

トランザクション情報の一覧を参照します。稼働中の J2EE サーバで動作しているトランザクション情報，および停止中の J2EE サーバに残っている未決着トランザクション情報を参照できます。

(1) 稼働中の J2EE サーバでのトランザクション情報の表示

稼働状況や開始してからの経過時間など，トランザクションの情報の一覧を参照します。

実行形式

```
cjlisttrn ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-gid <グローバルトランザクションID>］ ［-pending］ ［-time <経過時間>］ ［-bqual］
```

実行例

トランザクションに関連しているすべての Xid のブランチ ID を表示する例

```
cjlisttrn MyServer -bqual
```

cjlisttrn コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjlisttrn（稼働中の J2EE サーバのトランザクション情報の表示）」を参照してください。

(2) 停止中の J2EE サーバでのトランザクション情報の表示

停止中の J2EE サーバのステータスファイルに残っている，未決着トランザクション情報の一覧を参照します。

次に示すコマンドを実行してトランザクションの一覧を参照します。

実行形式

```
cjlisttrnfile ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ ［-gid <グローバルトランザクションID>］ ［-bqual］
```

実行例

指定のグローバルトランザクション ID に関する情報を表示する例

```
cjlisttrnfile MyServer -gid
d13800010000000000000000000000fefb57e648000000000000001
```

cjlisttrnfile コマンドの詳細については，マニュアル「Cosminexus アプリケーション サーバ リファレンス コマンド編」の「cjlisttrnfile（停止中の J2EE サーバのトランザクション情報の表示）」を参照してください。

11 Server Plug-in の基本操作

この章では，Server Plug-in で論理サーバ，J2EE アプリケーション，リソースアダプタを操作するための基本操作について説明します。

11.1　Server Plug-in パースペクティブ

Server Plug-in を使用してリソースアダプタや J2EE アプリケーションの設定をする場合，あらかじめ，Server Plug-in を Eclipse に組み込み，Server Plug-in を使用するための設定が必要です。Server Plug-in の Eclipse への組み込み手順については，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「7.8.2　Server Plug-in の組み込み手順」を参照してください。Server Plug-in を使用するための設定手順については，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「7.8.3　Server Plug-in を使用するための Management Server の設定」を参照してください。

Server Plug-in では，サーバの起動・停止，J2EE アプリケーションやリソースアダプタの操作を行う作業環境として，［Cosminexus Server Plug-in］パースペクティブ（Server Plug-in パースペクティブ）を提供しています。作業の目的によって必要なビューなどを初期設定した作業環境をパースペクティブといいます。

次の手順で，Server Plug-in パースペクティブを開きます。

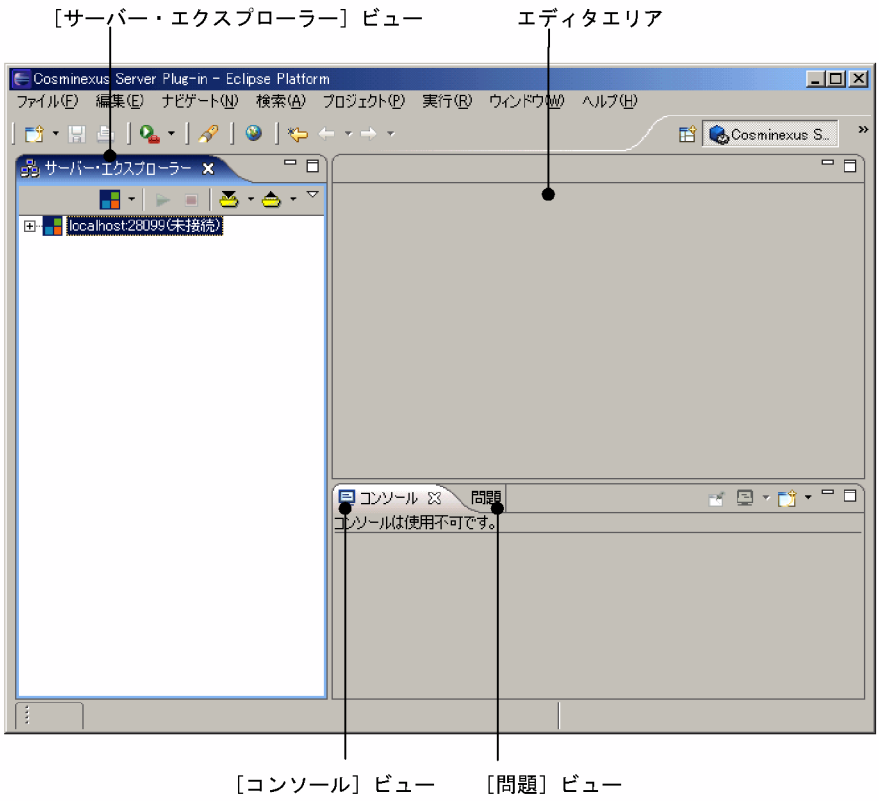
1. **Eclipse を起動して，Eclipse 画面を表示します。**
2. **次のどちらかの操作で，Server Plug-in パースペクティブを開きます。**
   - ［ショートカットバー］-［Cosminexus Server Plug-in］メニューを実行します。
   - ［ウィンドウ］-［パースペクティブを開く］-［Cosminexus Server Plug-in］メニューを実行します。

注意事項

Server Plug-in を起動するとワークスペース内に「.cad」というプロジェクトが作成されます。これは，Server Plug-in が作業用に使用するプロジェクトです。このプロジェクトおよびプロジェクト内のファイルの削除，追加，内容の変更およびプロジェクトを閉じるなどの操作をした場合，Server Plug-in の動作は保証できません。

［Cosminexus Server Plug-in］パースペクティブの構成を次に示します。

図 11-1　［Cosminexus Server Plug-in］パースペクティブの構成



［Cosminexus Server Plug-in］パースペクティブの構成要素について説明します。

［サーバー・エクスプローラー］ビュー

Server Plug-in の操作の中心となるビューです。論理サーバの起動・停止，J2EE アプリケーションやリソースアダプタの操作で使用します。

エディタエリア

エディタを表示する領域です。属性ファイル編集エディタで使用します。属性ファイル編集エディタについては，「11.3　プロパティの設定」を参照してください。

［コンソール］ビュー

サーバの起動・停止や J2EE アプリケーションの操作など，サーバと通信するプロセスの出力を表示します。

［問題］ビュー

属性ファイル編集エディタの構文エラーなどのエラー情報を表示します。

この節では，Server Plug-in パースペクティブと Server Plug-in の環境設定，および［サーバー・エクスプローラー］ビューの操作について説明します。

11.1.1　Server Plug-in の環境設定

Server Plug-in を使用するためには，次の設定が必要です。

- リモート管理機能に接続するための環境設定
- コンソールの共通の設定
- アプリケーション統合属性ファイルのエディタの設定

(1) 前提条件

Server Plug-in を使用するためには，次のことを確認してください。

Management Server がセットアップされているか

Management Server のセットアップについては，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「7.6.1　Management Server のセットアップ」を参照してください。

接続先の Management Server で Server Plug-in を使用するための設定がされているか

Server Plug-in を使用するための Management Server 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 システム構築・運用ガイド」の「7.8.3 Server Plug-in を使用するための Management Server の設定」を参照してください。

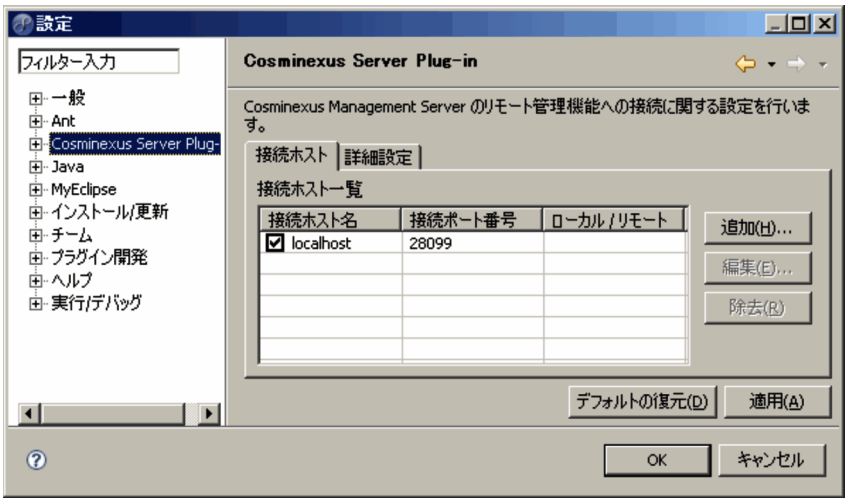
運用管理ドメイン内のすべてのホストの運用管理エージェント，および接続先の Management Server が起動されているか

運用管理エージェント，および Management Server の起動方法については，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「7.7　システムの起動と停止の設定」を参照してください。

(2) リモート管理機能に接続するための環境設定

Server Plug-in では，［設定］ダイアログの［Cosminexus Server Plug-in］ページでリモート管理機能に接続するための環境設定をします。［設定］ダイアログの［Cosminexus Server Plug-in］ページを次に示します。

図 11-2　［設定］ダイアログの［Cosminexus Server Plug-in］ページ



［Cosminexus Server Plug-in］ページは，次の手順で表示します。

1. **メニューから［ウィンドウ］-［設定］を選択します。**
   ［設定］ダイアログが表示されます。
2. **［設定］ダイアログの左ペインにあるツリービューで，［Cosminexus Server Plug-in］を選択します。**
   ［Cosminexus Server Plug-in］ページが表示されます。

［Cosminexus Server Plug-in］ページの構成は，次のとおりです。

```
［Cosminexus Server Plug-in］ページ
  ├─［接続ホスト］タブ
  │    ├─［接続ホストの追加］ダイアログ
  │    └─［接続ホストの編集］ダイアログ
  └─［詳細設定］タブ
```

［Cosminexus Server Plug-in］ページの各要素について説明します。

(a)［接続ホスト］タブ

［Cosminexus Server Plug-in］ページの［接続ホスト］タブについては，図 11-2 を参照してください。

［接続ホスト］タブの設定項目およびボタンについて説明します。

接続ホスト一覧

接続ホスト一覧には，複数のリモート管理機能に接続するための接続先ホストを登録できます。

接続ホスト名のチェックボックス
接続するホストを選択します。選択できるホストは一つです。同時に複数のホストを選択することはできません。

接続ホスト名
接続ホストのホスト名または IP アドレスが表示されています。ローカルホストの場合は，「localhost」が表示されています。

接続ポート番号
接続ホストのポート番号が表示されています。

ローカル / リモート
リモートホストとして設定されている場合に「リモート・ホスト」と表示されています。ローカルホストの場合は空欄です。

［追加］ボタン
［追加］ボタンをクリックすると，［接続ホストの追加］ダイアログが開きます。接続ホスト一覧に接続ホストを追加します。

［編集］ボタン
接続ホスト一覧でホストを選択して，［編集］ボタンをクリックすると，［接続ホストの編集］ダイアログが開きます。接続ホスト一覧で選択されたホストを編集します。
編集対象とするホストが選択されていない場合または複数のホストが選択されている場合は，［編集］ボタンは不活性になっています。

［除去］ボタン
接続ホスト一覧で一つまたは複数のホストを選択して，［除去］ボタンをクリックすると，接続ホスト一覧で選択されたホストは一覧から除去されます。
除去対象とするホストが選択されていない場合，またはすべてのホストが選択されている場合は，ボタンは不活性になっています。接続ホストとして選択されていたホストが削除された場合は，一覧の先頭にあるホストが選択された状態となります。

［デフォルトの復元］ボタン
［Cosminexus Server Plug-in］ページの各設定値をインストール時のデフォルト値に戻します。復元された値は［適用］ボタンをクリックすると保存されます。
［接続ホスト］タブのデフォルトは，次のとおりです。

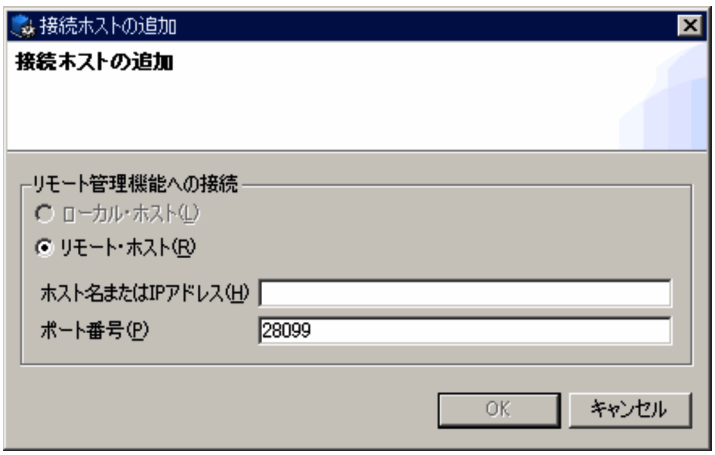
項目	デフォルト
接続ホストのチェックボックス	選択
接続ホスト名	localhost
接続ポート番号	28099
ローカル／リモート	（空欄）

［適用］ボタン
画面を終了しないで，入力値を保存します。

(b) [接続ホストの追加] ダイアログ

Management Server のリモート管理機能に接続するホストを追加します。追加された接続ホストは，[接続ホスト] タブの接続ホスト一覧の末尾に表示されます。

[接続ホストの追加] ダイアログを，次に示します。



[接続ホストの追加] ダイアログの設定項目とボタンについて次に説明します。

リモート管理機能への接続

[ローカル・ホスト] ／ [リモート・ホスト]

Management Server のリモート管理機能への接続が，ローカルホストなのか，リモートホストなのか選択します。

[ローカル・ホスト] または [リモート・ホスト] の選択によって，項目は次のように，活性または不活性になります。

- [接続ホスト] タブの接続ホスト一覧にローカルホストが設定済みの場合は，[ローカル・ホスト] は不活性になっていて，選択できません。
- 「ローカル・ホスト」を選択すると，「ホスト名または IP アドレス」に，「localhost」が表示され，不活性になります。
- 「リモート・ホスト」を選択すると，「ホスト名または IP アドレス」は活性になり，ホスト名または IP アドレスを指定できます。

[ホスト名または IP アドレス]

接続ホストのホスト名または IP アドレスを指定します。256 文字以内の文字を指定してください。

[ポート番号]

接続ホストのポート番号を指定します。接続先ホストの Management Server 環境設定ファイル（mserver.properties）の com.cosminexus.mngsvr.management.port に指定したポート番号を 5 文字以内の半角数字で指定してください。

［OK］ボタン
　接続ホストを追加して，ダイアログを閉じます。

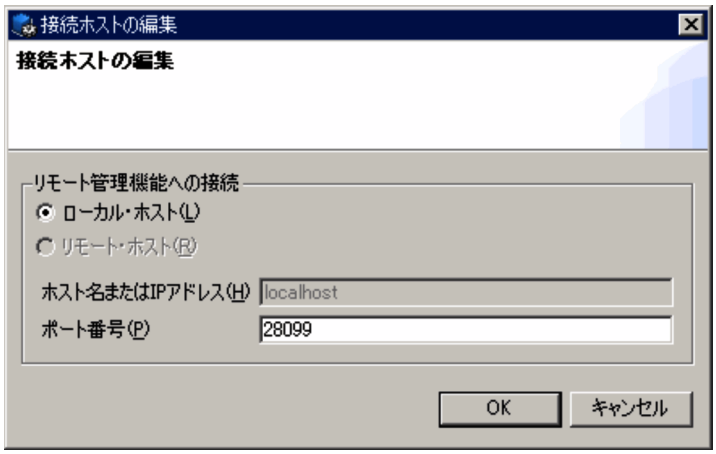
［キャンセル］ボタン
　接続ホストを追加しないで，ダイアログを閉じます。

エラーメッセージ表示領域
　「接続ホストの追加」の下に，エラーメッセージが表示されます。

(c) ［接続ホストの編集］ダイアログ

［接続ホストの編集］ダイアログで，Management Server のリモート管理機能に接続するホストの編集をします。

［接続ホストの編集］ダイアログを，次に示します。



［接続ホストの編集］ダイアログの設定項目およびボタンについて説明します。

リモート管理機能への接続

［ローカル・ホスト］／［リモート・ホスト］
　追加時の設定で，「ローカル・ホスト」または「リモート・ホスト」が活性になっています。変更することはできません。

［ホスト名または IP アドレス］
　「リモート・ホスト」が選択されている場合だけ，ホスト名または IP アドレスを指定できます。256 文字以内の文字を指定してください。

［ポート番号］
　接続ホストのポート番号を指定します。5 文字以内の半角数字を指定してください。

［OK］ボタン
　接続ホストを編集して，ダイアログを閉じます。

［キャンセル］ボタン
　接続ホストを編集しないで，ダイアログを閉じます。

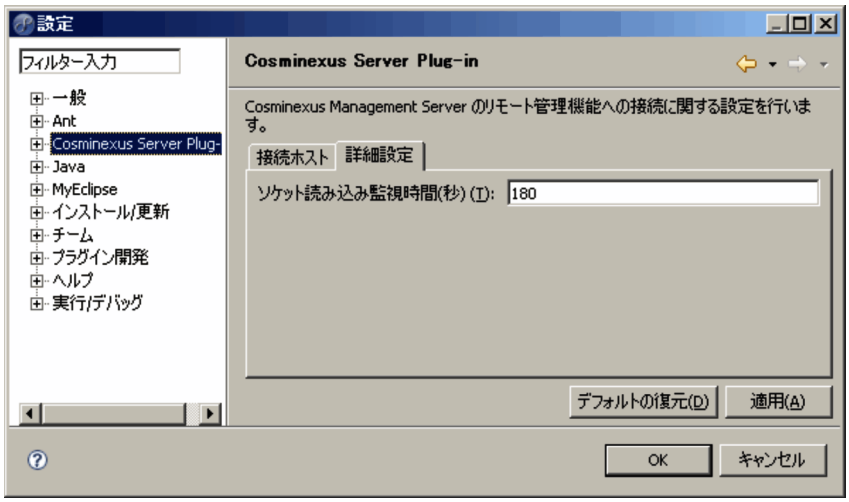
エラーメッセージ表示領域
　「接続ホストの編集」の下に，エラーメッセージが表示されます。

(d) ［詳細設定］タブ

［詳細設定］タブで，Management Server のリモート管理機能に接続するための詳細な設定をします。

［Cosminexus Server Plug-in］ページの［詳細設定］タブを，次に示します。

図 11-3　［詳細設定］タブ



［詳細設定］タブの設定項目およびボタンについて説明します。

［ソケット読み込み監視時間（秒）］
　Management Server のリモート管理機能への接続で，ソケットの読み込みを監視する時間（秒単位）に，0 または 180 ～ 99999 の半角数字を指定します。0（秒）を指定した場合は，監視は実施されません。ソケット読み込み監視時間の入力がない場合は，180（秒）が仮定されます。

注意事項

- ソケット読み込み監視時間は，Management Server のリモート管理機能にログインするときに設定されます。リモート管理機能に接続中に，ソケット読み込み監視時間を変更した場合，変更した監視時間は，次回，リモート管理機能にログインした時から有効になります。
- リモート管理機能を使用した処理の実行中に，「Read timed out」のメッセージが表示されたあとも，処理が続行されている場合があります。

［デフォルトの復元］ボタン
　［Cosminexus Server Plug-in］ページの各設定値をインストール時のデフォルト値に戻します。復元された値は［適用］ボタンをクリックすると保存されます。
　ソケット読み込み監視時間のデフォルト値は 180（秒）です。

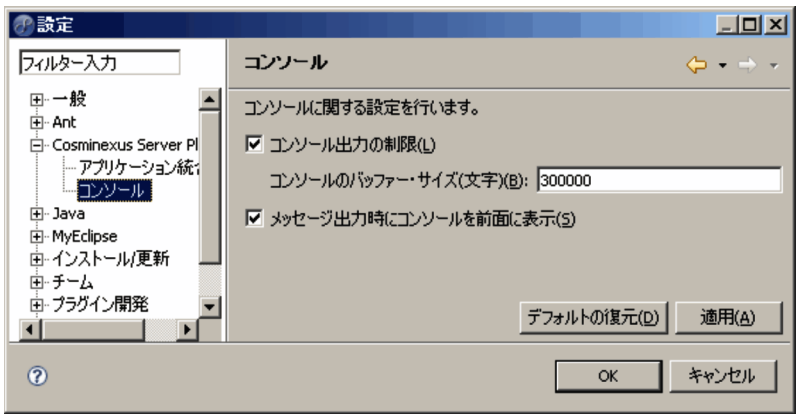
［適用］ボタン
　画面を終了しないで，入力値を保存します。

（3） コンソールの共通の設定

サーバー・エクスプローラー操作用のコンソールや J2EE アプリケーション実行用のコンソールなど，Server Plug-in では複数のコンソールを使用します。［設定］ダイアログの［コンソール］ページで，各種コンソールの共通の設定をします。

［設定］ダイアログの［コンソール］ページを次に示します。

図 11-4　［設定］ダイアログの［コンソール］ページ



［コンソール］ページは，次の手順で表示します。

1. **メニューから［ウィンドウ］-［設定］を選択します。**
　［設定］ダイアログが表示されます。
2. **［設定］ダイアログの左ペインにあるツリービューで，**［Cosminexus Server Plug-in］-**［コンソール］を選択します。**
　［コンソール］ページが表示されます。

［コンソール］ページの設定項目およびボタンについて説明します。

［コンソール出力の制限］のチェックボックス
　コンソール出力を制限するかどうかを選択します。出力を制限する場合，チェックを入れます。チェックを外すと出力制限は解除されます。
　出力制限が選択されている場合，出力された行までの文字数がバッファサイズを超えると，先頭から超えた分の出力行が消去されます。消去されたメッセージは次のファイルで確認できます。

<Eclipseワークスペースディレクトリ>/.metadata/.log

［コンソールのバッファー・サイズ（文字)］
［コンソール出力の制限］のチェックボックスにチェックを入れている場合，必ず入力してください。チェックを外している場合，入力値は無効です。
一つのコンソールのバッファサイズ（文字数）を，20000 ～ 1000000 の半角数字で指定します。
バッファサイズは，必要に応じて調整してください。

［メッセージ出力時にコンソールを前面に表示］のチェックボックス
メッセージを出力するときに，出力対象のコンソールを前面表示するかどうかを選択します。コンソールを前面に表示する場合，チェックを入れます。チェックを外すとコンソールは前面表示されません。ただし，コンソールに初めてメッセージが出力された場合は，そのコンソールは前面表示されます。

［デフォルトの復元］ボタン
［コンソール］ページの各設定値をデフォルト値に戻します。復元された値は［適用］ボタンをクリックすると保存されます。
［コンソール］ページのデフォルトは，次のとおりです。

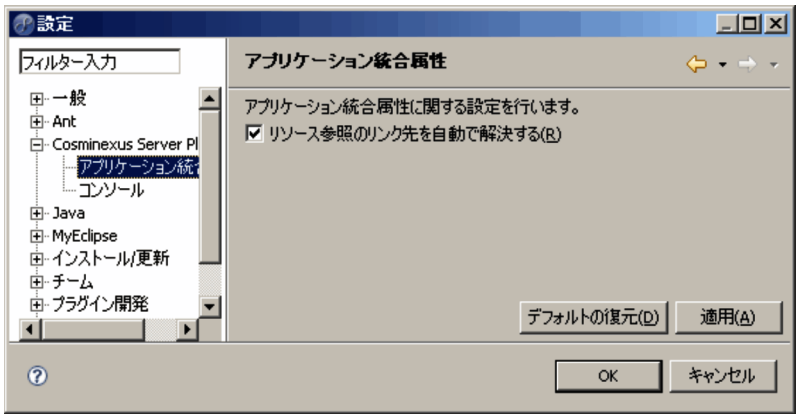
項目	デフォルト
［コンソール出力の制限］のチェックボックス	チェック済み
［コンソールのバッファー・サイズ（文字)］	300000
［メッセージ出力時にコンソールを前面に表示］のチェックボックス	チェック済み

［適用］ボタン
画面を終了しないで，入力値を保存します。

(4) アプリケーション統合属性ファイルのエディタの設定

Server Plug-in では，［設定］ダイアログの［アプリケーション統合属性］ページで，アプリケーション統合属性ファイルの属性ファイル編集エディタの設定をします。属性ファイル編集エディタについては，「11.3.2　属性ファイル編集エディタ」を参照してください。

［設定］ダイアログの［アプリケーション統合属性］ページを次に示します。



［アプリケーション統合属性］ページは，次の手順で表示します。

1. **メニューから［ウィンドウ］-［設定］を選択します。**
   ［設定］ダイアログが表示されます。
2. **［設定］ダイアログの左ペインにあるツリービューで，**［Cosminexus Server Plug-in］-**［アプリケーション統合属性］を選択します。**
   ［アプリケーション統合属性］ページが表示されます。

［アプリケーション統合属性］ページの設定項目およびボタンについて説明します。

［リソース参照のリンク先を自動で解決する］のチェックボックス

アプリケーション統合属性ファイルの属性編集エディタを起動したときに，J2EE アプリケーションのリソース参照を自動解決するかどうかを選択します。リソース参照を自動解決する場合，チェックを入れます。チェックを外すと，リソース参照は自動解決されません。

リソース参照の自動解決については，「11.3.3　リソース参照の自動解決」を参照してください。

［デフォルトの復元］ボタン

［アプリケーション統合属性］ページの設定値をデフォルト値に戻します。復元された値は［適用］ボタンをクリックすると保存されます。

［アプリケーション統合属性］ページのデフォルトは，次のとおりです。

項目	デフォルト
［リソース参照のリンク先を自動で解決する］のチェックボックス	チェック済み

［適用］ボタン

画面を終了しないで，入力値を保存します。

11.1.2　Management Server リモート管理機能への接続

Management Server リモート管理機能へ接続する場合，Management Server リモート管理機能へのログインを実行します。また，Management Server リモート管理機能への接続を切断する場合，Management Server リモート管理機能からのログアウトを実行します。

（1） Management Server リモート管理機能へのログイン

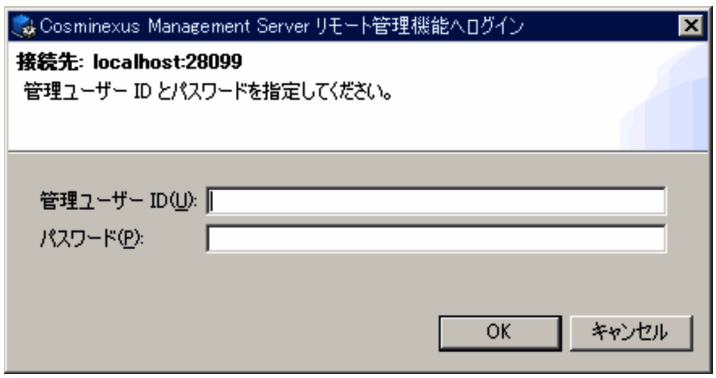
次のどちらかの操作で Management Server リモート管理機能へ接続します。

- ［サーバー・エクスプローラー］ビューの＜リモート管理機能と接続しているホスト名＞以下のツリーを展開します。
- ［サーバー・エクスプローラー］ビューで，［Cosminexus Management Server リモート管理機能へログイン］操作を実行します。
  ［Cosminexus Management Server リモート管理機能へログイン］操作については，「21.1.2　Management Server リモート管理機能へのログイン」を参照してください。

Management Server リモート管理機能に接続すると，［Cosminexus Management Server リモート管理機能へログイン］ダイアログが表示されます。［Cosminexus Management Server リモート管理機能へログイン］ダイアログで，管理ユーザ ID とパスワードを指定してログインします。

なお，管理ユーザアカウントの省略機能を有効にしている場合，［OK］ボタンをクリックするとログインできます。［管理ユーザー ID］および［パスワード］の入力は不要です。

図 11-5　［Cosminexus Management Server リモート管理機能へログイン］ダイアログ



［Cosminexus Management Server リモート管理機能へログイン］ダイアログの設定項目およびボタンについて説明します。

［管理ユーザー ID］

Management Server へログインするための管理ユーザ ID を指定します。管理ユー

ザアカウントの省略機能を有効にしている場合，この項目は省略できます。

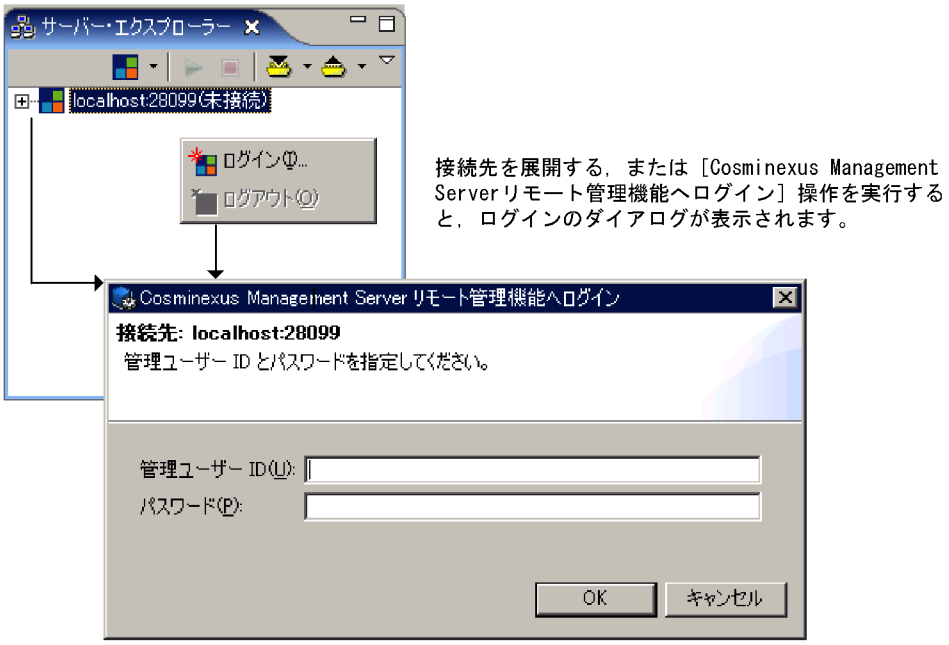
［パスワード］
Management Server へログインするためのパスワードを指定します。パスワードが設定されていない場合，または管理ユーザアカウントの省略機能を有効にしている場合，この項目は省略できます。

［OK］ボタン
ダイアログを閉じて，Management Server に接続します。

［キャンセル］ボタン
ダイアログを閉じて，Management Server には接続しません。

Management Server のリモート管理機能への接続手順を次に示します。

図 11-6　Management Server のリモート管理機能への接続手順



ログイン中は，「Cosminexus Management Server リモート管理機能へ接続中」のプログレスダイアログが表示されます。

Management Server リモート管理機能に接続すると，［サーバー・エクスプローラー］ビューの Management Server リモート管理機能に接続しているホストの状態が「(未接続)」から「(接続済)」に変更されます。＜リモート管理機能と接続しているホスト名＞以下のツリーは展開されません。［サーバー・エクスプローラー］ビューのツリービューの状態表示については，「11.2　表示の更新」を参照してください。

Management Server のリモート管理機能への接続に失敗した場合，エラーメッセージが

表示されます。

（2） Management Server リモート管理機能からのログアウト

Management Server リモート管理機能からログアウトするには，［サーバー・エクスプローラー］ビューで，［Cosminexus Management Server リモート管理機能からログアウト］操作を実行します。［Cosminexus Management Server リモート管理機能からログアウト］操作については，「21.1.3　Management Server リモート管理機能からのログアウト」を参照してください。

ログアウト中は，「Cosminexus Management Server リモート管理機能との接続を切断中」のプログレスダイアログが表示されます。

Management Server リモート管理機能との接続が切断されると，［サーバー・エクスプローラー］ビューは初期状態またはリモート管理機能への接続先変更後と同じ状態になります。［サーバー・エクスプローラー］ビューの Management Server リモート管理機能に接続しているホストの状態は，「(接続済)」から「(未接続)」に変更されます。［サーバー・エクスプローラー］ビューのツリーの状態表示については，「11.2　表示の更新」を参照してください。

また，リモート管理機能の接続先のホストを変更すると，Management Server リモート管理機能からログアウトされます。接続先のホストの変更については，「(3) Management Server リモート管理機能への接続先変更」を参照してください。

（3） Management Server リモート管理機能への接続先変更

［設定］ダイアログの［Cosminexus Server Plug-in］ページで，リモート管理機能への接続先を変更できます。リモート管理機能への接続先を変更した場合，パースペクティブを構成する各ウィンドウは，次の状態になります。

［サーバー・エクスプローラー］ビュー
　展開されているノードはすべて閉じられ，接続先のノードだけが表示されます。ノードについては，「11.1.3　サーバー・エクスプローラーの構成」を参照してください。

エディタエリア
　編集中の属性ファイル編集エディタは閉じます。属性ファイル編集エディタ以外はそのままの状態です。

［コンソール］ビュー
　リモート管理機能にログインするまで，変更前の接続先のログは表示されたままです。リモート管理機能にログインすると，変更前の接続先のログは破棄されます。

［問題］ビュー
　そのままの状態です。

［Cosminexus Server Plug-in］パースペクティブの構成については，図 11-1 を参照して

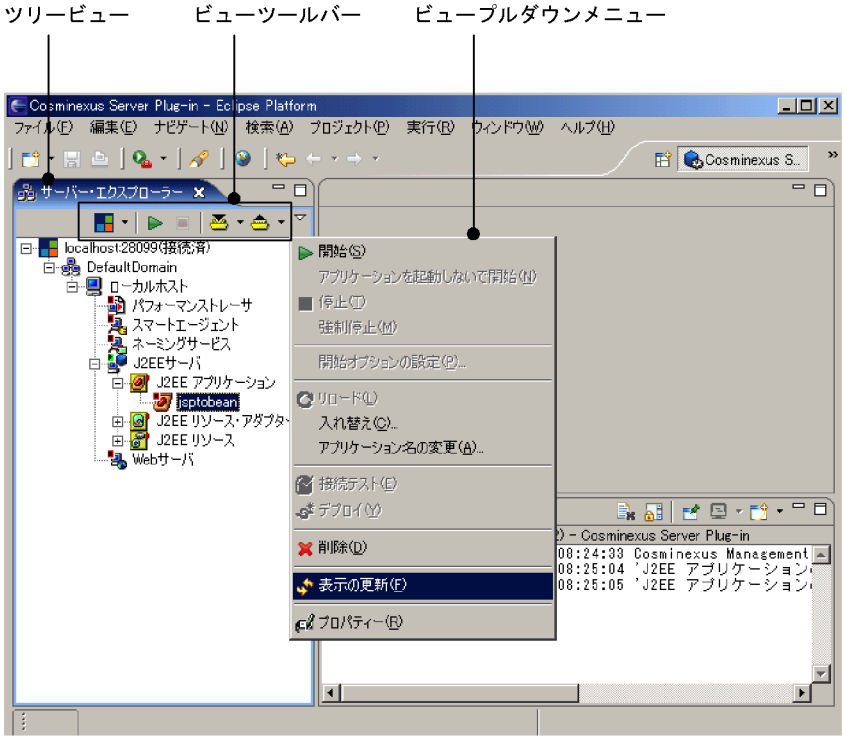
ください。

11.1.3　サーバー・エクスプローラーの構成

［Cosminexus Server Plug-in］パースペクティブの，［サーバー・エクスプローラー］ビューは，Server Plug-in の操作の中心となるビューです。Management Server リモート管理機能への接続，論理サーバの起動・停止，J2EE アプリケーションやリソースアダプタの起動・停止・実行属性の設定などの操作で使用します。

［サーバー・エクスプローラー］ビューの構成を次に示します。

図 11-7　［サーバー・エクスプローラー］ビューの構成



ツリービュー

論理サーバや J2EE アプリケーションやリソースアダプタをツリーで表示するビューです。ツリービューで表示されている項目をノードといいます。
ノードを選択して，右クリックで操作のコンテキストメニュー（ポップアップメニュー）が表示されます。

ビューツールバー

ツリービューで選択したノードの操作を表示するビューです。

ビュープルダウンメニュー
　ツリービューで選択したノードの操作を表示するメニューです。

11.1.4　サーバー・エクスプローラーの操作

Server Plug-in では，［サーバー・エクスプローラー］ビューを使用して，Management Server リモート管理機能への接続，論理サーバの起動・停止およびアプリケーション設定操作を実行します。

［サーバー・エクスプローラー］ビューの操作手順を次に示します。

1. **ツリービューで，操作の対象となるノードを選択します。**
   ツリービューで表示するノードの種類については，「21.1　Server Plug-in の操作で使用する画面」を参照してください。
2. **ビューツールバー，ビュープルダウンメニューまたは右クリックで表示されるコンテキストメニューで操作を選択します。**
   各操作の詳細については，「21.1.1　サーバー・エクスプローラーの基本操作」を参照してください。
   選択した操作の実行経過はコンソールビューに表示されます。
   操作の実行経過および実行結果の表示については，「11.1.5　コンソールおよび操作実行時のダイアログ表示」を参照してください。

ビューツールバー，ビュープルダウンメニューまたは右クリックで表示されるコンテキストメニューでの各操作は，上記操作の 1. で選択したノードによって，実行できるもの（活性）と実行できないもの（不活性）があります。また，ノードの状態によっても実行できるものと実行できないものがあります。ノードと操作の活性・不活性の関係については，「21.1.1　サーバー・エクスプローラーの基本操作」を参照してください。

ノードの状態は，ツリービューに表示されています。ツリービューの表示の確認については，「11.2　表示の更新」を参照してください。

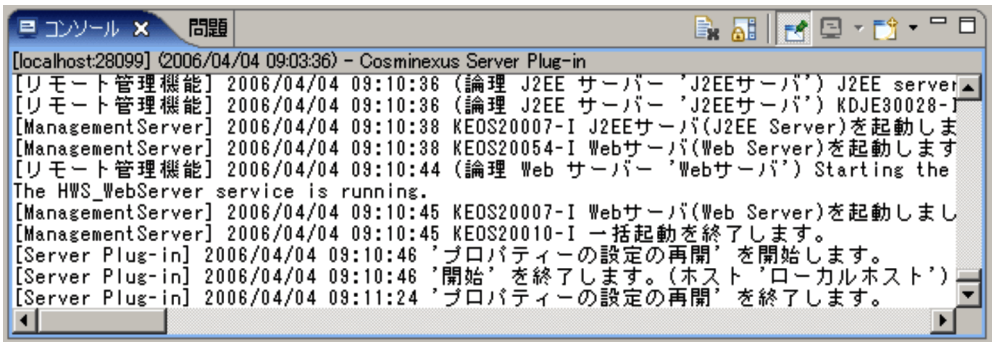
11.1.5　コンソールおよび操作実行時のダイアログ表示

操作が実行されると，［コンソール］ビューに実行経過が表示されます。また，操作の実行中にプログレスダイアログが表示されます。［コンソール］ビューおよびプログレスダイアログの表示について説明します。

(1)［コンソール］ビュー

操作の実行経過は，［Cosminexus Server Plug-in］パースペクティブの［コンソール］ビューに表示されます。コンソールは，接続している Management Server 単位に，一つだけ提供されています。また，コンソールには，Eclipse のエラーログも出力されています。

図 11-8 ［コンソール］ビューの実行経過表示



［コンソール］ビューには，Server Plug-in のコンソールであることが識別できるように，次の形式でタイトルが表示されています。

タイトルの形式

```
［＜リモート管理機能と接続しているホスト名＞］ （＜リモート管理機能への接続した日時＞） -
Cosminexus Server Plug-in
```

コンソールには，次の情報が表示されます。

- ［サーバー・エクスプローラー］ビューの操作メッセージ
  Server Plug-in から通知される，［サーバー・エクスプローラー］ビューで選択した操作が実行されたときのメッセージです。メッセージの形式を次に示します。

  メッセージの形式

```
［Server Plug-in］ ＜日時＞ ＜操作メッセージ＞ （＜操作対象＞）
```

  メッセージ例

  ［Server Plug-in］yyyy/mm/dd hh:MM:ss '開始' を開始します （ホスト 'MyHost'）
  ［Server Plug-in］yyyy/mm/dd hh:MM:ss '開始' を終了します （ホスト 'MyHost'）

- Management Server から通知されるメッセージ
  Management Server から通知される Management Server のメッセージです。メッセージの形式を次に示します。

  メッセージの形式

```
［＜出力コンポーネント＞］ ＜日時＞ ＜通知されたメッセージID＞ ＜通知されたメッセージ＞
```

メッセージ例

［Management Server］yyyy/mm/dd hh:MM:KEOS2009-I 一括起動を開始します。

● Management Server を経由して起動された CUI などのエラーメッセージ
Management Server を経由して起動された CUI などの標準出力／標準エラーのメッセージです。メッセージの形式を次に示します。

メッセージの形式

```
［リモート管理機能］　＜日時＞　＜投入されたエラーメッセージ＞
```

● Management Server から通知される論理サーバの標準出力／標準エラー出力
Management Server から通知される論理サーバの標準出力／標準エラーのメッセージです。メッセージの形式を次に示します。

メッセージの形式

```
［リモート管理機能］　＜日時＞　（＜論理サーバ名＞）＜通知されたエラーメッセージ＞
```

注意事項

Server Plug-in の環境設定の［コンソール］ページで，［メッセージ出力時にコンソールを前面に表示］がチェック済みのとき，メッセージを出力するたびにコンソールが前面に表示されます。［コンソール］ビューに複数のコンソールがある場合，閲覧中のコンソールが Server Plug-in のコンソールに切り替わります。自動的なコンソールの切り替えを抑止するには，［コンソール］ページの［メッセージ出力時にコンソールを前面に表示］のチェックを外すか，［コンソール］ビューの

（［コンソールのピン留め］）を ON にしてください。

(2) プログレスダイアログ

次の場合，プログレスダイアログが表示され，［サーバー・エクスプローラー］ビューの操作は抑止されます。

- 操作の実行で前処理が必要な場合，前処理実行中のプログレスダイアログが表示されます。前処理が完了するまで表示されています。
- 操作実行中は，実行が進行中であることを示すプログレスダイアログが表示されています。操作の実行が終了するまで表示されています。
  操作実行中にエラーが発生した場合，エラーメッセージダイアログが表示されます。エラーメッセージダイアログの［詳細］ボタンをクリックすると，エラーの状況が表示されます。エラー内容は［コンソール］ビューで確認してください。

11.2　表示の更新

この節では，リモート管理機能の接続先の状態表示および論理サーバや J2EE アプリケーションの状態表示について説明します。

［サーバー・エクスプローラー］ビューに，次のどちらかの形式でノードの状態は表示されます。

- ステータスアイコン
  ノードのアイコンの左上に表示されます。
- 文字列
  ノードの表示名の横に括弧で囲んで表示されます。
  文字列での状態表示の形式を次に示します。

＜アイコン＞＜表示名＞（＜状態文字列＞）

リモート管理機能に接続済みのとき，［サーバー・エクスプローラー］ビューでは，次のノードの状態を参照できます。

- リモート管理機能の接続先
- 論理サーバ
- J2EE アプリケーション
- J2EE リソースアダプタ
- リソースアダプタ

(1) リモート管理機能の接続先の状態表示

リモート管理機能の接続先への接続の状態は，文字列で表示されています。状態を表示する（＜状態文字列＞）は，次のとおりです。

- 未接続の場合：(未接続)
- 接続済みの場合：(接続済)

リモート管理機能の接続先に未接続の場合，［Cosminexus Management Server リモート管理機能へのログイン］ダイアログでログインして，リモート管理機能に接続してください。リモート管理機能への接続については，「11.1.2　Management Server リモート管理機能への接続」を参照してください。

(2) 論理サーバの状態表示

各論理サーバの稼働状態によって，［サーバー・エクスプローラー］ビューの［＜論理サーバ名＞］にステータスアイコンが付加されています。

ステータスアイコンは，次の場合に更新されます。

- Management Server の通知によって自動的に更新された場合

- ［サーバー・エクスプローラー］ビューのビュープルダウンメニューの［表示の更新］を選択して，手動で［サーバー・エクスプローラー］ビューを最新の状態に更新した場合

付加されるステータスアイコンを次に示します。

表 11-1　各論理サーバの稼働状態

稼働状態	ステータスアイコン
停止	
稼働中	
異常停止	
停止中	
起動中	
回復中	
自動停止中	
自動再起動中	
自動停止中（回復中）	
自動停止中（異常停止）	
自動停止中（再起動中）	
強制停止中	
計画停止中	
通信障害	
不明	

論理 J2EE サーバの稼働状態が［停止］の場合，［サーバー・エクスプローラー］ビューでは，［+］が付加されないで，下位ノード（J2EE アプリケーション，J2EE リソースアダプタ，J2EE リソース）が展開されません。

（3） J2EE アプリケーションの状態表示

J2EE アプリケーションの稼働状態によって，［サーバー・エクスプローラー］ビューの［＜ J2EE アプリケーション名＞］にステータスアイコンが付加されています。

［サーバー・エクスプローラー］ビューのビュープルダウンメニューの［表示の更新］を選択して，手動で［サーバー・エクスプローラー］ビューのステータスアイコンを最新の状態に更新します。自動的には更新されません。

付加されるステータスアイコンを次に示します。

表 11-2　J2EE アプリケーションの稼働状態

稼働状態	ステータスアイコン※
開始状態	
停止状態	
通常停止失敗状態	
強制停止失敗状態	
閉塞失敗状態	
閉塞中	
通常停止中	
強制停止中	
閉塞状態	

注※　次の場合，ステータスアイコンに加えて，その状態が表示されています。

- J2EE アプリケーションが展開ディレクトリ形式の場合，ステータスアイコンの右下に展開ディレクトリ形式を示すアイコン（  ）が付加されています。
- J2EE アプリケーションのプロパティを設定中（アプリケーション統合属性の設定中）の場合，「(プロパティー設定中)」の状態文字列が表示されています。

(4) J2EE リソースアダプタの状態表示

デプロイされた J2EE リソースアダプタの稼働状態によって，［サーバー・エクスプローラー］ビューの［＜ J2EE リソースアダプタ名＞］にステータスアイコンが付加されています。

［サーバー・エクスプローラー］ビューのビュープルダウンメニューの［表示の更新］を選択して，手動で［サーバー・エクスプローラー］ビューのステータスアイコンを最新の状態に更新します。自動的には更新されません。

付加されるステータスアイコンを次に示します。

表 11-3　J2EE リソースアダプタの稼働状態

稼働状態	ステータスアイコン※
開始状態	
停止状態	

注※ J2EE リソースアダプタのプロパティを設定中（Connector 属性の設定中）の場合，ステータスアイコンに加えて，「(プロパティー設定中)」の状態文字列が表示されています。

(5) リソースアダプタの状態表示

次のリソースアダプタのプロパティを設定中（Connector 属性の設定中）の場合，対応する［＜リソースアダプタ名＞］に，「(プロパティー設定中)」の状態文字列が付加されています。

- J2EE リソースアダプタとしてデプロイして使用するリソースアダプタ
  ［サーバー・エクスプローラー］ビューの［J2EE リソース］-［リソース・アダプター］以下の［＜リソースアダプタ名＞］
- J2EE アプリケーションに含めて使用するリソースアダプタ
  ［サーバー・エクスプローラー］ビューの［＜ J2EE アプリケーション名＞］下の［＜リソースアダプタ名＞］

11.3　プロパティの設定

J2EE アプリケーションやリソースアダプタの属性（プロパティ）を設定する操作について説明します。

11.3.1　プロパティの設定操作

Server Plug-in は，リソースアダプタや J2EE アプリケーションのプロパティを編集するためのエディタとして，**属性ファイル編集エディタ**を提供しています。

なお，Server Plug-in のバージョンと Cosminexus Component Container のバージョンが合致していない場合，属性ファイル編集エディタで属性ファイルの編集はできません。属性ファイル編集エディタを使用しないで，属性ファイルを編集する操作については，「11.3.2　属性ファイル編集エディタ」の「(5)　属性ファイル編集エディタを使用しない属性ファイルの編集」を参照してください。

(1) 手順

リソースアダプタや J2EE アプリケーションのプロパティを設定する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，プロパティを設定するリソースアダプタまたは J2EE アプリケーションを選択します。**
2. **次のどれかの方法で，属性ファイル編集エディタを起動します。**
   - ［サーバー・エクスプローラー］ビューで選択したノードをダブルクリックします。
   - ビュープルダウンメニューの［ プロパティ］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ プロパティ］を選択します。

   手順 1. で選択したノードによって，次の属性ファイル編集エディタが起動されます。
   - リソースアダプタを選択した場合
     Connector 属性ファイル編集エディタ
   - J2EE アプリケーションを選択した場合
     アプリケーション統合属性ファイル編集エディタ

   J2EE アプリケーションにリソースアダプタが含まれる場合，ツリービューで J2EE アプリケーション下のリソースアダプタを選択すると，Connector 属性ファイル編集エディタが起動されます。J2EE アプリケーションを選択すると，アプリケーション統合属性ファイル編集エディタが起動されます。
   属性ファイル編集エディタは，［Cosminexus Server Plug-in］パースペクティブのエディタエリアに表示されます。
3. **各属性ファイルに対応したエディタを開いて，プロパティを設定します。**
   属性ファイル編集エディタの操作については，「11.3.2　属性ファイル編集エディタ」

を参照してください。
プロパティの定義項目については，次の説明を参照してください。

- J2EE リソースアダプタとしてデプロイして使用するリソースアダプタのプロパティ定義
  「14.3.2　プロパティ設定手順（Connector 1.0 の場合）」，および「14.3.3　プロパティ設定手順（Connector 1.5 の場合）」
- J2EE アプリケーションに含めて使用するリソースアダプタのプロパティ定義
  「15.3　リソースアダプタのプロパティ定義」
- J2EE アプリケーションのプロパティ定義
  「18.　J2EE アプリケーションのプロパティ設定」

4. **プロパティの設定後，属性ファイルを保管します。**
   属性ファイルは，次のどれかの操作で保管されます。
   - Eclipse のメニューの［ファイル］-［保管］を選択
   - Eclipse の  （保管）をクリック
   - ソースページで右クリックして表示される，コンテキストメニューの［保管］を選択
   - 属性ファイル編集エディタを閉じる※

   注※　編集状態で，属性ファイル編集エディタを閉じると，変更を保管するかどうかの確認メッセージが表示されます。［はい］をクリックすると保管処理が実行されます。［いいえ］をクリックすると編集は破棄されます。
   属性ファイルを保管すると，属性ファイルの値は J2EE アプリケーションやリソースアダプタの属性に設定されます。
   属性ファイルの保管については，「11.3.2　属性ファイル編集エディタ」を参照してください。

(2) 画面表示

J2EE アプリケーションやリソースアダプタのプロパティ設定で使用する画面については，「21.1.14　プロパティの設定」を参照してください。

(3) 対応するサーバ管理コマンド

J2EE アプリケーションやリソースアダプタのプロパティを設定する場合のサーバ管理コマンドについては，「3.3　属性ファイルによるプロパティの設定」を参照してください。

11.3.2　属性ファイル編集エディタ

属性ファイル編集エディタは，Server Plug-in が提供する属性ファイル編集用のエディタです。

属性ファイル編集エディタには，次の 2 種類があります。

- アプリケーション統合属性ファイルエディタ

J2EE アプリケーションに含まれるコンポーネントの属性を一括して表示します。

- Connector 属性ファイルエディタ
  Connector 属性を表示します。

属性ファイル編集エディタは，フォームページおよびソースページから構成されています。次に，属性ファイル編集エディタの画面構成を示します。

図 11-9　属性ファイル編集エディタの画面構成

(1) フォームページ

フォームページは属性ファイルを編集するためのページです。属性ファイル編集エディタを起動すると，次のフォームページが初期表示されます。

- J2EE アプリケーションのプロパティ設定の場合
  アプリケーション統合属性ファイルのフォームページ
- リソースアダプタのプロパティ設定の場合
  Connector 属性ファイルのフォームページ

(a) フォームページでのプロパティ設定操作

フォームページは，ツリー（エレメント）と編集エリア（エレメント詳細）で構成されています。フォームページの構成については，図 11-9 を参照してください。

ツリー（エレメント）
　編集中の属性ファイルの構成がツリー形式で表示されています。属性ファイル（XML ファイル）の複合要素が一つのノードで表示されていますが，複数回出現する要素については，要素をまとめる仮想ノードが使用されています。仮想ノードにはアイコン  が付加されています。

編集エリア（エレメント詳細）
　ツリーでの選択に対応した編集項目を編集するためのページが表示されています。

ページの各項目は属性ファイル（XML ファイル）の一つの要素に対応しています。

フォームページで，プロパティを編集および設定するための基本操作を次に示します。

1. **ツリーで，属性ファイルの編集するノードを選択します。**
   ツリーでの選択に対応したプロパティ設定ページが編集エリアに表示されます。
2. **編集エリアのプロパティ設定ページで，プロパティを指定します。**
   指定したプロパティは，次の操作でソースページに反映されます。
   - ツリーで選択中のノードを切り替えた場合
   - ソースタブを選択して，ソースページの表示に切り替えた場合

ツリーを選択して右クリックするとコンテキストメニューが表示されます。コンテキストメニューで，ツリーのノードの追加および削除ができます。

ツリーおよび編集エリアの詳細については，「21.2　リソースアダプタの属性編集画面」および「21.3　J2EE アプリケーションの属性編集画面」を参照してください。

(b) アプリケーション統合属性ファイルのフォームページ

J2EE アプリケーションのプロパティ設定には，アプリケーション統合属性ファイル用エディタのフォームページを使用します。アプリケーション統合属性ファイル用エディタのフォームページは，次の属性ファイル種別タブに分けて表示されています。各タブをクリックすると，編集対象のフォームページが表示されます。

- アプリケーション属性
  アプリケーション属性のフォームページについては，「21.4　アプリケーション統合属性ファイル（アプリケーション属性）」を参照してください。
- EJB-JAR
  J2EE アプリケーションに EJB-JAR が含まれる場合，アプリケーション統合属性ファイルの「<ejb-jar>」タグ下の属性が表示されます。
  EJB-JAR 属性のフォームページについては，「21.5　アプリケーション統合属性ファイル（EJB-JAR 属性）」を参照してください。
- WAR
  J2EE アプリケーションに WAR が含まれる場合，アプリケーション統合属性ファイルの「<war>」タグ下の属性が表示されます。
  WAR 属性のフォームページについては，「21.9　アプリケーション統合属性ファイル（WAR 属性）」を参照してください。
- RAR
  J2EE アプリケーションに RAR が含まれる場合，アプリケーション統合属性ファイルの「<rar>」タグ下の属性が表示されます。
  RAR 属性のフォームページは，Connector 属性のフォームページと同じです。
  Connector 属性のフォームページについては，「21.2　リソースアダプタの属性編集画面」を参照してください。

(c) Connector 属性ファイルのフォームページ

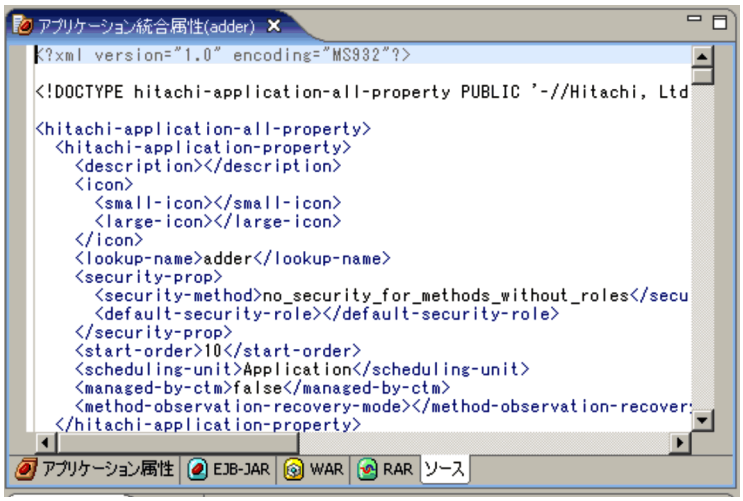
リソースアダプタのプロパティ設定には，Connector 属性ファイル用エディタのフォームページを使用します。Connector 属性ファイル用エディタのフォームページには Connector 属性ページが表示されています。

Connector 属性のフォームページについては，「21.2　リソースアダプタの属性編集画面」を参照してください。

(2) ソースページ

［ソース］タブをクリックすると，ソースページが表示されます。ソースページには，属性ファイル（XML ファイル）の内容が表示されています。アプリケーション統合属性ファイルおよび Connector 属性ファイルの内容を直接編集する場合に使用します。ソースページを次に示します。

図 11-10　属性ファイル編集エディタのソースページ



ソースページで属性ファイルを編集して，属性ファイルの定義（スキーマ）に従わない不正な構文となった場合，エラーとなります。この場合，属性ファイルの保管時に［問題］ビューに構文エラー個所が表示されるので，［問題］ビューを確認してソースページで修正してください。

(3) 属性ファイルの保管

属性ファイル編集エディタで属性ファイルを保管すると，属性ファイルの内容は，J2EE アプリケーションやリソースアダプタの属性が反映されます。

属性ファイル設定中は，「J2EE アプリケーション'＜J2EE アプリケーション名＞'を操作中」のプログレスダイアログが表示されています。

ただし，次の場合は J2EE アプリケーションやリソースアダプタの属性は設定されませ

ん。

- 属性ファイルにエラーがある場合
  ［問題］ビューでエラーが表示されます。
- 属性ファイル編集時にエラーが発生した場合
  属性ファイル編集中のエラーは，エラーダイアログおよび［コンソール］ビューに表示されます。また，サーバで属性ファイル（XML ファイル）にエラーが検知された場合も，［コンソール］ビューにエラーが表示されます。
- Eclipse が終了した場合
  次回，Eclipse を起動したときに属性ファイル編集エディタが再開されます。
- リモート管理機能への接続先を変更した場合
  次回，リモート管理機能への接続先を変更したときに属性ファイル編集エディタが再開されます。
- 論理 J2EE サーバが停止した場合
  次回，論理 J2EE サーバが開始したときに属性ファイル編集エディタが再開されます。

保管に失敗して，J2EE アプリケーションやリソースアダプタの属性が設定されなかった場合，属性ファイルの内容を修正して，保管し直すことができます。

（4） 属性ファイル編集エディタのエラー表示

属性ファイル編集エディタでは，属性ファイル編集エディタの起動時や属性ファイルの保管時などで，属性ファイル（XML ファイル）の正当性をチェックしています。エラーの内容は，エラーの種類に応じて，次のように表示されます。

- ダイアログが表示されます。
  XML ファイルとして不正なソースの場合，エラーメッセージダイアログが表示されます。
- ［問題］ビューにエラーの位置が表示されます。
  属性ファイルのタグの並びや出現回数など，XML スキーマ定義に一致していない場合，エラーの位置が［問題］ビューに表示されます。ソースページでマーキングされたエラー行を確認できます。
- 属性ファイル編集のエディタエリアに警告メッセージが表示されます。
  属性ファイルの編集時に妥当性エラーがある場合，エディタエリアの［エレメント詳細］の先頭行に，「このページにエラーがあります。」という警告メッセージが表示され，妥当性エラーとなっている項目にマウスポインタが合わされています。
- ［コンソール］ビューにエラーが表示されます。
  サーバから返されたエラーが表示されます。

エラーが表示された場合，エラー内容に応じて属性ファイルの定義内容を確認してください。属性ファイルの定義については，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）」を参照してください。

（5） 属性ファイル編集エディタを使用しない属性ファイルの編集

Server Plug-in のバージョンと Cosminexus Component Container のバージョンが合致

していない場合など，属性ファイル編集エディタで属性の編集ができない場合，属性を属性ファイルに取得（エクスポート）し，プロパティ設定項目を編集します。編集した属性ファイルの値を属性に設定（インポート）してください。

操作手順を次に示します。

1. **属性を取得します。**
   Connector 属性またはアプリケーション統合属性をエクスポートします。
   Connector 属性のエクスポートについては，「14.9.2　Connector 属性のエクスポート」を，アプリケーション統合属性のエクスポートについては，「20.3　アプリケーション統合属性のエクスポート」を参照してください。
2. **プロパティ設定項目を編集します。**
   エクスポートした属性ファイルを，テキストエディタなどで開き，編集します。
3. **属性を設定します。**
   Connector 属性またはアプリケーション統合属性をインポートします。
   Connector 属性のインポートについては，「14.9.1　Connector 属性のインポート」を，アプリケーション統合属性のインポートについては，「20.2　アプリケーション統合属性のインポート」を参照してください。

11.3.3　リソース参照の自動解決

Server Plug-in では，J2EE アプリケーションのリソース参照を自動解決できます。

リソース参照の自動解決とは，統合属性ファイルの属性ファイル編集エディタを起動したときに，編集対象の J2EE アプリケーションのリソース参照を自動で設定することです。

リソース参照の自動解決を実行するかどうかは，［設定］ダイアログの［アプリケーション統合属性］ページで，［リソース参照のリンク先を自動で解決する］のチェックボックスで設定します。［設定］ダイアログの［アプリケーション統合属性］ページについては，「11.1.1　Server Plug-in の環境設定」の「(4)　アプリケーション統合属性ファイルのエディタの設定」を参照してください。

(1) 自動解決の対象リソースアダプタ

リソース参照の自動解決は，次のリソースアダプタに対して実行されます。

- J2EE リソースアダプタ
- J2EE アプリケーションに含まれるリソースアダプタ

(2) 自動解決で設定される属性ファイルの定義項目

次の属性のリンク先（<resource-ref> - <linked-to> タグ）が自動で設定されます。

- Session Bean 属性
- Entity Bean 属性

- Message-driven Bean 属性
- WAR 属性

(3) リソース参照の自動解決の条件

次の順番で自動解決されます。

1. Connector 1.0 の仕様に準拠するリソースアダプタ
2. Connector 1.5 の仕様に準拠するリソースアダプタ

リソースの自動解決の条件は，リソースアダプタのバージョンによって，次のように異なります。

● Connector 1.0 の仕様に準拠するリソースアダプタの場合
次の条件を満たす場合，リンク先（<resource-ref> - <linked-to> タグ）に＜リソースアダプタの表示名＞が設定されます。

- Connector 1.0 の仕様に準拠するリソースアダプタが一つだけデプロイされている。
- リソース参照のタイプ（<resource-ref> - <res-type> タグ）の値が「javax.sql.DataSource」である場合に，「＜ Cosminexus のインストールディレクトリ＞ ¥CC¥DBConnector」で提供されるリソースアダプタがデプロイされている。
- リソース参照のタイプ（<resource-ref> - <res-type> タグ）の値が「javax.jms.QueueConnectionFactory」である場合に，「＜ Cosminexus のインストールディレクトリ＞ ¥RM¥lib」で提供されるリソースアダプタがデプロイされている。
- コネクション定義識別子が「javax.sql.DataSource」または「javax.jms.QueueConnectionFactory」である，Connector 1.5 の仕様に準拠するリソースアダプタがデプロイされていない。

リソース参照のタイプ（<resource-ref> - <res-type> タグ）の値と，自動解決の対象とするリソースアダプタの関係を次の表に示します。なお，リソースアダプタの判別には表示名が使用されます。

表 11-4 リソース参照のタイプの値と自動解決の対象とするリソースアダプタの関係

<resource-ref> - <res-type> タグの値	自動解決の対象とするリソースアダプタの表示名
javax.sql.DataSource	DB_Connector_for_Cosminexus_Driver
	DB_Connector_for_Cosminexus_Driver_XA
	DB_Connector_for_HiRDB_Type4
	DB_Connector_for_HiRDB_Type4_XA
	DB_Connector_for_Oracle
	DB_Connector_for_Oracle_XA
	DB_Connector_for_SQLServer
	DB_Connector_for_SQLServer2005

<resource-ref> - <res-type> タグの値	自動解決の対象とするリソースアダプタの表示名
	DB_Connector_for_ClusterPool_Root
	DB_Connector_for_Oracle_ClusterPool_Member
	DB_Connector_for_Cosminexus_Driver_Cosminexus_RM
	DB_Connector_for_Cosminexus_Driver_XA_Cosminexus_RM
	DB_Connector_for_HiRDB_Type4_Cosminexus_RM
	DB_Connector_for_HiRDB_Type4_XA_Cosminexus_RM
	DB_Connector_for_Oracle_Cosminexus_RM
	DB_Connector_for_Oracle_XA_Cosminexus_RM
javax.jms.QueueConnectionFactory	Cosminexus_Reliable_Messaging

● Connector 1.5 の仕様に準拠するリソースアダプタの場合
Connector 1.0 の仕様に準拠するリソースアダプタのリソース参照が解決できなかった場合に実行されます。
次の条件を満たす場合，リンク先（<resource-ref> - <linked-to> タグ）に，＜リソースアダプタ名＞！＜コネクション定義識別子＞が設定されます。

- Outbound リソースアダプタがデプロイされている。
- リソース参照のタイプ（<resource-ref> - <res-type> タグ）の値とコネクション定義識別子が一致するリソースアダプタが一つだけデプロイされている。
- リソース参照のタイプ（<resource-ref> - <res-type> タグ）の値が「javax.sql.DataSource」である場合に，Connector 1.0 の仕様に準拠するリソースアダプタがデプロイされていない。

! 注意事項

アノテーションを使用している場合でも，リンク先（<linked-to>）の設定値は自動で解決されます。アノテーションと属性ファイルで同時に指定した場合の実行は，属性ファイルの仕様に従います。

(4) リソース参照の自動解決例

次の J2EE アプリケーションの例で，リソース参照の自動解決の流れを示します。

前提

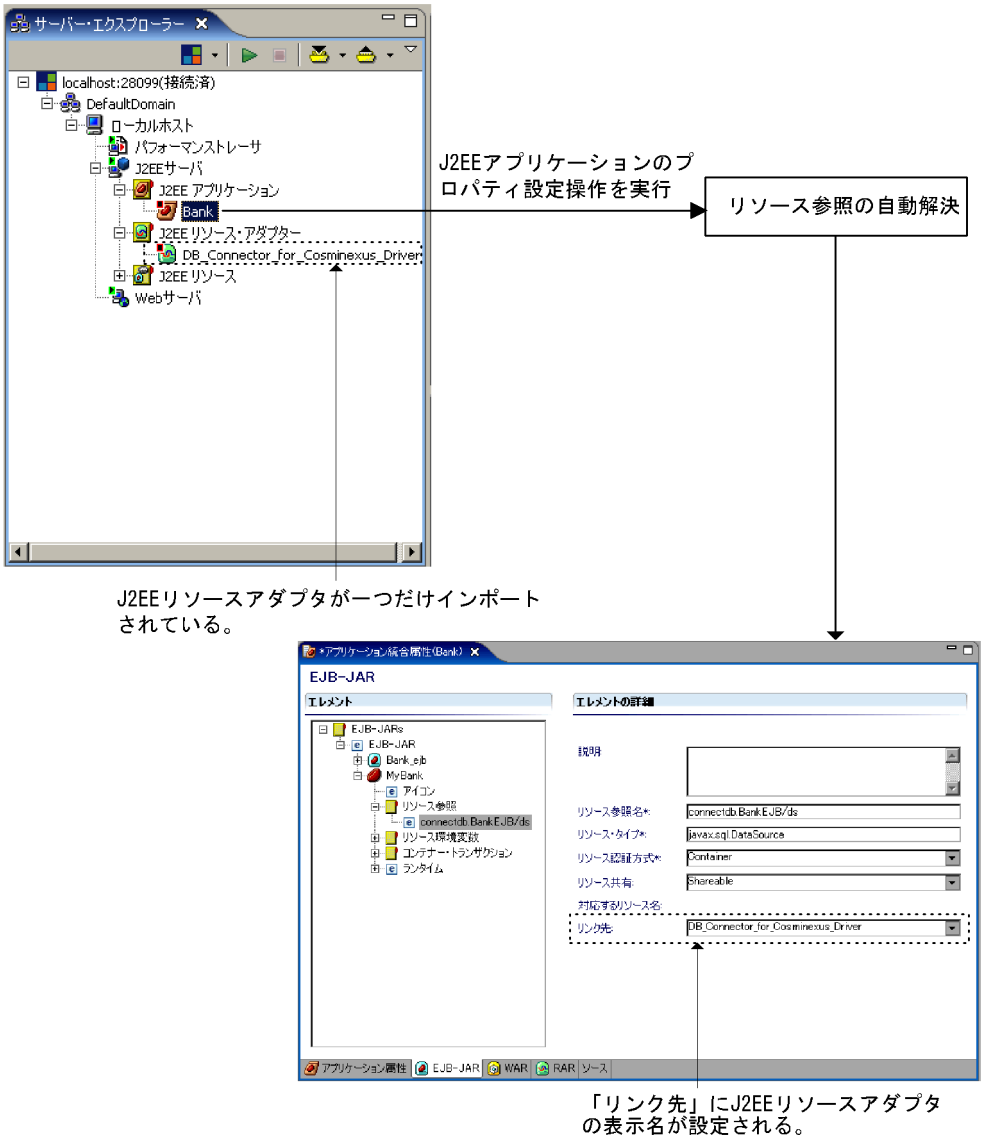
- Connector 1.0 の仕様に準拠するリソースアダプタ（DB_Connector_for_Cosminexus_Driver）だけが J2EE リソースアダプタとしてデプロイされています。J2EE アプリケーション（Bank）にはリソースアダプタは含まれていません。
- J2EE アプリケーション（Bank）の Session Bean 属性のリソース参照は，次の

ように設定されています。

- リソースタイプ（res-type）：「javax.sql.DataSource」
- リンク先（<linked-to>）：空タグ

［サーバー・エクスプローラー］ビューで J2EE アプリケーション（Bank）を選択し，プロパティ設定操作で属性ファイル編集エディタを起動すると，次のようにリンク先が自動設定されます。

図 11-11　リソース参照の自動解決の流れ



属性ファイル編集エディタで，［EJB-JAR］フォームページのリソース参照のプロパティページを表示すると，［リンク先］に，リソースアダプタの表示名が設定されています。

なお，アプリケーション統合属性ファイルは自動解決で変更されているため，属性ファイル編集エディタのステータスは「編集済み」となっています。

11.4　Server Plug-in の注意事項

Server Plug-in を使用する際の注意事項を次に示します。

（1） Windows 7 または Windows Vista 使用時の注意事項

Windows 7 または Windows Vista を使用している場合で，管理者特権で実行する必要がある操作について説明します。なお，システムドライブを C ドライブとして説明します。

C:¥Program Files 以下のディレクトリまたは C:¥windows 以下のディレクトリに対する操作には制限があります。これらの制限されたディレクトリに対して，ファイルをエクスポートしたり，Eclipse のワークスペースを保存したりすると，リダイレクション機能によって，操作対象のファイルはリダイレクトされ，次の場所に格納されます。

C:¥Users¥< ユーザ名 >¥AppData¥Local¥VirtualStore

エクスポートまたは保存したファイルの実体は上記の場所に格納されていますが，Eclipse 上からは C:¥Program Files 以下に実ファイルがあるかのように，ファイルを読み込んだり，書き込んだりできます。

（2） Server Plug-in を使用する場合の注意事項

EJB-JAR 属性の <interceptor-order> タグに複数の <interceptor-class> タグを指定した場合，または次に示すタグにハイフン（-）を指定した場合，エラーが発生します。この場合，Server Plug-in ではなく，サーバ管理コマンドを使用して属性ファイルを直接編集して対処してください。

- Session Bean 属性および Entity Bean 属性の <optional-name> タグ，<local-optional-name> タグ
- Connector 属性の <optional-name> タグ

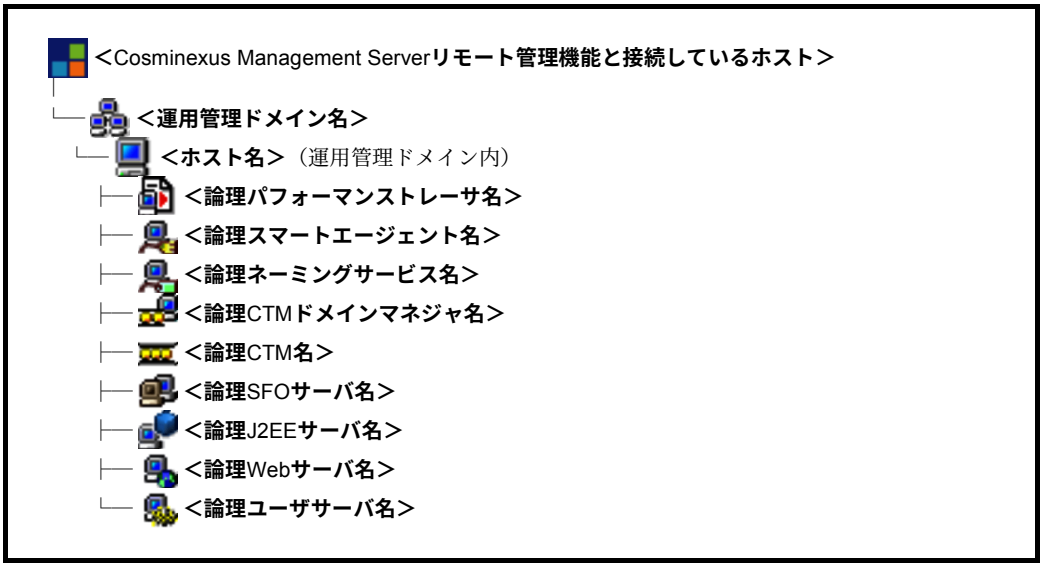
12 論理サーバの運用管理

この章では，Server Plug-in を使用した，論理サーバの起動や停止，稼働状況の確認などの操作について説明します。なお，この章の操作に対応するサーバ管理コマンドはありません。Management Server の運用管理コマンドなどが対応します。

12.1　論理サーバの運用管理の概要

この節では，Server Plug-in で管理できる論理サーバと，論理サーバの運用で使用する操作について説明します。

［サーバー・エクスプローラー］ビューには，次の論理サーバが表示されます。



［サーバー・エクスプローラー］ビューで表示される論理サーバは，起動順に並んでいます。論理サーバの起動順については，マニュアル「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」の「11.　論理サーバの起動／停止」を参照してください。

論理サーバの概要と論理サーバの運用で使用する操作を次の表に示します。

表 12-1　論理サーバの概要

論理サーバ	操作※		概要
	起動／停止	運用監視	
Cosminexus Management Server リモート管理機能と接続しているホスト	×	○	Management Server リモート管理機能と接続しているホストです。次のように表示されます。 ＜ホスト名または IP アドレス＞：＜ポート番号＞ 初期表示のときは，このノードだけが表示されています。
運用管理ドメイン	○	×	運用管理ドメイン名が表示されています。
ホスト名（運用管理ドメイン内）	○	×	運用管理ドメインに定義されているホスト名が表示されています。

論理サーバ	操作※		概要
	起動／停止	運用監視	
論理パフォーマンストレーサ	○	○	論理パフォーマンストレーサ名が表示されています。
論理スマートエージェント	○	○	論理スマートエージェント名が表示されています。
論理ネーミングサービス	○	○	論理ネーミングサービス名が表示されています。
論理 CTM ドメインマネジャ	○	○	論理 CTM ドメインマネジャ名が表示されています。
論理 CTM	○	○	論理 CTM 名が表示されています。
論理 SFO サーバ	○	○	論理 SFO サーバ名が表示されています。
論理 J2EE サーバ	○	○	論理 J2EE サーバ名が表示されています。
論理 Web サーバ	○	○	論理 Web サーバ名が表示されています。
論理ユーザサーバ	○	○	論理ユーザサーバ名が表示されています。

（凡例）　○：実行できる　×：実行できない

注　論理サーバについては，マニュアル「Cosminexus アプリケーションサーバ システム設計ガイド」を参照してください。

注※　論理サーバの起動／停止については，「12.2　論理サーバの起動」および「12.3　論理サーバの停止」を参照してください。論理サーバの運用監視については，「12.4　論理サーバの稼働状況表示」を参照してください。

12.2　論理サーバの起動

この節では，論理サーバを起動するときの，次の操作について説明します。

- 論理 J2EE サーバに開始オプションを設定する。
- 論理サーバを起動する。
- アプリケーションを起動しないで J2EE サーバを起動する。

12.2.1　開始オプションの設定

論理 J2EE サーバでは，デフォルト設定で，セキュリティマネジャが動作しています。展開ディレクトリ形式の J2EE アプリケーションを使用する場合，セキュリティに関する設定を変更して運用することが推奨されています。

セキュリティマネジャの使用有無を，［開始オプションの設定］ダイアログで設定します。

(1) 手順

開始オプションを設定する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，開始オプションを設定する［<論理 J2EE サーバ名>］を選択します。**
2. **次のどちらかの方法で，開始オプションの設定操作を実行します。**
   - ビュープルダウンメニューの［開始オプションの設定］を選択します。
   - 右クリックで表示されるコンテキストメニューの［開始オプションの設定］を選択します。

   開始オプションの設定操作が実行されると，［開始オプションの設定］ダイアログが表示されます。
3. **［セキュリティー・マネージャーの使用］について，セキュリティマネージャを使用する場合は，［する］を選択します。セキュリティマネージャを使用しない場合は，［しない］を選択します。**
4. **論理 J2EE サーバの開始オプションを更新する場合は，［OK］ボタンをクリックします。更新しない場合は，［キャンセル］ボタンをクリックします。**
   ［開始オプションの設定］ダイアログが閉じます。

(2) 画面表示

論理 J2EE サーバの開始オプションの設定で使用する画面については，「21.1.6　論理 J2EE サーバの開始オプションの設定」を参照してください。

(3) 注意事項

- 開始オプションの設定について

開始オプションの設定は，Management Server の設定情報として配布されます。運用管理ポータルなどで変更した場合，その設定が有効になります。セキュリティマネジャを使用しない設定にする場合，J2EE サーバ起動時のコマンドオプションに -nosecurity オプションを付加します。J2EE サーバ起動時のコマンドオプションについては，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「cjstartsv（J2EE サーバの開始）」を参照してください。

- usrconf.properties（J2EE サーバ用ユーザプロパティファイル）などの更新について
  Server Plug-in を使用して J2EE サーバの開始オプションを設定する場合，Management Server を使用して J2EE サーバの設定情報を配布します。
  このため，usrconf.cfg（J2EE サーバ用オプション定義ファイル），usrconf.properties（J2EE サーバ用ユーザプロパティファイル）などを直接編集して指定した設定情報については破棄されます。J2EE サーバの設定は，Management Server を使用して指定してください。
  設定情報の配布で設定されるファイルについては，マニュアル「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」の「10.11　論理サーバの設定ファイルの配布」を参照してください。

12.2.2　論理サーバの通常起動

Server Plug-in で論理サーバを通常起動するには，次の方法があります。

● ドメイン単位の論理サーバの一括起動
運用管理ドメイン内のすべての論理サーバを一括起動します。起動順序は，［サーバー・エクスプローラー］ビューで表示された順序に従います。なお，［サーバー・エクスプローラー］ビューに表示されていない論理サーバは一括起動の対象外になります。

● ホスト単位の論理サーバの一括起動
ホスト内のすべての論理サーバを一括起動します。起動順序は，［サーバー・エクスプローラー］ビューで表示された順序に従います。なお，［サーバー・エクスプローラー］ビューに表示されていない論理サーバは一括起動の対象外になります。

● 個別の論理サーバの起動
論理サーバを個別に起動します。J2EE アプリケーションを開始したあとに Web サーバを起動する場合や，異常終了した論理サーバだけ再起動する場合などに，論理サーバを個別に起動します。

（1）手順

論理サーバを起動する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，起動する論理サーバを選択します。**
   論理サーバの起動方法によって，次のノードを選択します。
   - ドメイン単位の論理サーバを一括起動する場合
     ［＜運用管理ドメイン名＞］を選択します。

- ホスト単位の論理サーバを一括起動する場合
  ［＜ホスト名＞］を選択します。
- 個別の論理サーバを起動する場合
  次の論理サーバのどれかを選択します。
  論理パフォーマンストレーサ
  論理スマートエージェント
  論理ネーミングサービス
  論理 CTM ドメインマネジャ
  論理 CTM
  論理 SFO サーバ
  論理 J2EE サーバ
  論理 Web サーバ
  論理ユーザサーバ

2. **次のどれかの方法で，開始操作を実行します。**
   - ビューツールバーの［ ▶ ］（開始）をクリックします。
   - ビュープルダウンメニューの［ ▶ 開始］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ ▶ 開始］を選択します。

   個別の論理サーバを起動する場合，上記 1. で選択した論理サーバが開始状態のときは，不活性になっていて選択できません。
   指定した論理サーバが，すべて起動されます。

（2） 画面表示

論理サーバの開始で使用する画面については，「21.1.4　開始」を参照してください。

12.2.3　アプリケーションを起動しないで J2EE サーバを起動

J2EE アプリケーションを起動しないで，論理 J2EE サーバを起動します。

（1） 手順

J2EE アプリケーションを起動しないで，論理 J2EE サーバを開始する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［＜論理 J2EE サーバ名＞］を選択します。**
2. **次のどちらかの方法で，アプリケーションを起動しないで開始する操作を実行します。**
   - ビュープルダウンメニューの［アプリケーションを起動しないで開始］を選択します。
   - 右クリックで表示されるコンテキストメニューの［アプリケーションを起動しない

で開始］を選択します。

アプリケーションを起動しないで，論理 J2EE サーバが起動されます。

（2） 画面表示

アプリケーションを起動しないで J2EE サーバを起動する画面については，「21.1.5　アプリケーションを起動しない場合の J2EE サーバの開始」を参照してください。

12.3　論理サーバの停止

この節では，論理サーバを停止するときの，次の操作について説明します。

- 論理サーバを通常停止する。
- 論理 J2EE サーバを強制停止する。

12.3.1　論理サーバの通常停止

Server Plug-in で論理サーバを通常停止するには，次の方法があります。

● ドメイン単位の論理サーバの一括停止
運用管理ドメイン内のすべての論理サーバを一括停止します。停止順序は，［サーバー・エクスプローラー］ビューで表示された順序の逆順になります。なお，［サーバー・エクスプローラー］ビューに表示されていない論理サーバは一括停止の対象外になります。

● ホスト単位の論理サーバの一括停止
ホスト内のすべての論理サーバを一括停止します。停止順序は，［サーバー・エクスプローラー］ビューで表示された順序の逆順になります。なお，［サーバー・エクスプローラー］ビューに表示されていない論理サーバは一括停止の対象外になります。

● 個別の論理サーバの停止
論理サーバを個別に停止します。J2EE アプリケーションを停止する前に Web サーバを停止する場合などに，論理サーバを個別に停止します。

(1) 手順

論理サーバを停止する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，停止する論理サーバを選択します。**
論理サーバの停止方法によって，次のノードを選択します。
   - ドメイン単位の論理サーバを一括停止する場合
   ［＜運用管理ドメイン名＞］を選択します。
   - ホスト単位の論理サーバを一括停止する場合
   ［＜ホスト名＞］を選択します。
   - 個別の論理サーバを停止する場合
   次の論理サーバのどれかを選択します。
   論理パフォーマンストレーサ
   論理スマートエージェント
   論理ネーミングサービス
   論理 CTM ドメインマネジャ
   論理 CTM
   論理 SFO サーバ
   論理 J2EE サーバ

論理 Web サーバ
論理ユーザサーバ

2. **次のどれかの方法で，停止操作を実行します。**
   - ビューツールバーの［  ］（停止）をクリックします。
   - ビュープルダウンメニューの［ 停止］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ 停止］を選択します。

   個別の論理サーバを停止する場合，上記 1. で選択した論理サーバが停止状態のときは，不活性になっていて選択できません。
   指定した論理サーバが，すべて停止されます。

(2) 画面表示

論理サーバの停止で使用する画面については，「21.1.7　停止」を参照してください。

12.3.2　論理 J2EE サーバの強制停止

論理 J2EE サーバを強制停止します。

(1) 手順

論理 J2EE サーバを強制停止する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，強制停止する［＜論理 J2EE サーバ名＞］を選択します。**
2. **次のどちらかの方法で，強制停止をする操作を実行します。**
   - ビュープルダウンメニューの［強制停止］を選択します。
   - 右クリックで表示されるコンテキストメニューの［強制停止］を選択します。

   論理 J2EE サーバが強制停止されます。

(2) 画面表示

論理 J2EE サーバを強制停止する画面については，「21.1.8　強制停止」を参照してください。

12.4　論理サーバの稼働状況表示

［サーバー・エクスプローラー］ビューのステータスアイコンで，次の論理サーバの稼働状態を確認できます。ステータスアイコンは，論理サーバのアイコンの左上に付加されます。

- Management Server リモート管理機能との接続状態
  Management Server リモート管理機能との接続状態の表示については，「11.2　表示の更新」の「(1)　リモート管理機能の接続先の状態表示」を参照してください。
- 運用管理ドメイン内のすべての論理サーバの稼働状態
  論理サーバの状態表示と付加されるステータスアイコンについては，「11.2　表示の更新」の「(2)　論理サーバの状態表示」を参照してください。

13 Server Plug-in による J2EE リソースの設定の概要

この章では，Server Plug-in を使用した，J2EE リソースの管理の概要について説明します。

13.1　J2EE リソース設定機能

リソースアダプタは，次のどちらかの方法で管理します。

- J2EE リソースアダプタとしてデプロイして，その J2EE サーバ上で動作するすべての J2EE アプリケーションで使用できるようにします。
- リソースアダプタをアプリケーションに含めて，その J2EE アプリケーションで使用できるようにします。

J2EE アプリケーションは，これらのどちらのリソースアダプタも使用できます。ただし，同じ表示名のリソースアダプタを同時に使用することはできません。同時に使用すると，J2EE アプリケーションは使用するリソースアダプタを区別できません。

J2EE リソースアダプタとしてデプロイして使用するリソースアダプタのアプリケーション設定操作については，「14.　リソースアダプタの設定」を参照してください。J2EE アプリケーションに含めて使用するリソースアダプタのアプリケーション設定操作については，「15.　J2EE アプリケーションに含まれるリソースアダプタの設定」を参照してください。

また，次の機能は，Server Plug-in では使用できません。

- JavaBeans リソースの設定
- メールコンフィグレーションの設定
- J2EE リソースのコピー

13.2　ビューツールバーでの J2EE リソース設定操作

Server Plug-in での J2EE リソースの設定操作を次に示します。

1. **［サーバー・エクスプローラー］ビューに表示されたツリー項目（ノード）を選択します。**
   選択するノードを次に示します。
   - J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの場合
     ［J2EE アプリケーション］フォルダ以下のノードを選択します。
   - J2EE アプリケーションに含めて使用するリソースアダプタの場合
     ［J2EE リソース・アダプター］フォルダ以下のノードまたは［J2EE リソース］フォルダ以下のノードを選択します。
2. **ビューツールバーなどで操作を指定します。**

［サーバー・エクスプローラー］ビューの構成については，「11.1.3　サーバー・エクスプローラーの構成」を参照してください。

［サーバー・エクスプローラー］ビューでの操作については，「11.1.4　サーバー・エクスプローラーの操作」を参照してください。

［サーバー・エクスプローラー］ビューに表示されたノードと，そのノードが実行できる操作について「21.1.1　サーバー・エクスプローラーの基本操作」を参照してください。

13.3　J2EE リソースアダプタの状態表示

J2EE リソースアダプタの稼働状態は，［サーバー・エクスプローラー］ビューのステータスアイコンで表示します。ステータスアイコンは，アイコンの左上に付加されます。

J2EE リソースアダプタの状態表示と付加されるステータスアイコンについては，「11.2　表示の更新」の「(4)　J2EE リソースアダプタの状態表示」を参照してください。

14 リソースアダプタの設定

この章では，Server Plug-in を使用したリソースアダプタの設定について説明します。
J2EE アプリケーションに含まれるリソースアダプタの設定については，「15. J2EE アプリケーションに含まれるリソースアダプタの設定」を参照してください。

14.1 リソースアダプタの設定の概要

J2EE リソースアダプタとしてデプロイして利用するリソースアダプタの種類と，アプリケーション設定操作の概要について説明します。

(1) 利用できるリソースアダプタ

次のリソースアダプタについて設定操作ができます。

- データベースと接続するための設定（DB Connector を使用する場合）
- データベース上のキューと接続するための設定（DB Connector for Cosminexus RM および Cosminexus RM を使用する場合）
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）

(2) リソースアダプタの設定操作

J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの設定に必要な作業は次のとおりです。

- リソースアダプタのインポート
  Server Plug-in 環境にあるリソースアダプタ，または J2EE サーバが稼働するホストにあるリソースアダプタをインポートします。
- リソースアダプタのプロパティ定義
- リソースアダプタのデプロイ
- J2EE リソースアダプタの接続テスト
- J2EE リソースアダプタの開始
- J2EE リソースアダプタの停止
- リソースアダプタの削除
- J2EE リソースアダプタのエクスポート

また，必要に応じて J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの Connector 属性をインポートおよびエクスポートします。

データベース上のキューと接続するための設定（DB Connector for Cosminexus RM および Cosminexus RM を使用する場合）については，マニュアル「Cosminexus Reliable Messaging」を参照してください。

Server Plug-in を使用した J2EE リソースの管理（GUI）の機能と，サーバ管理コマンド（CUI）の対応を次の表に示します。

表 14-1　Server Plug-in を使用した J2EE リソースの管理の機能とサーバ管理コマンドの対応

機能		対応するコマンド	参照先
リソースアダプタのインポート	Server Plug-in 環境にあるリソースアダプタ	cjimportres	14.2.1
	J2EE サーバ環境にあるリソースアダプタ		14.2.2
リソースアダプタのプロパティ定義		cjgetresprop, cjsetresprop または cjgetrarprop, cjsetrarprop	14.3
リソースアダプタのデプロイ		cjdeployrar	14.4
J2EE リソースアダプタの接続テスト		cjtestres	14.5
J2EE リソースアダプタの開始		cjstartrar	14.6.1
J2EE リソースアダプタの停止		cjstoprar	14.6.2
リソースアダプタの削除		cjundeployrar, cjdeleteres	14.7
J2EE リソースアダプタのエクスポート		cjexportrar	14.8
J2EE リソースアダプタの状態表示※1		（cjlistrar）	11.2
リソースアダプタの一覧表示※2		（cjlistres）	11.2
Connector 属性のインポートとエクスポート		cjgetresprop, cjsetresprop または cjgetrarprop, cjsetrarprop	14.9

（凡例）（　）：類似機能に対応するサーバ管理コマンド（CUI）
注※1 ［表示の更新］操作を実行すると，［サーバー・エクスプローラー］ビューの［J2EE リソース・アダプター］フォルダ内の［＜J2EE リソースアダプタ名＞］に，デプロイされているリソースアダプタの状態が表示されます。
注※2 ［サーバー・エクスプローラー］ビューの［J2EE リソース・アダプター］フォルダ内の［リソース・アダプター］フォルダに，インポートされている［＜リソースアダプタ名＞］が表示されています。

［サーバー・エクスプローラー］ビューについては，「11.1.3　サーバー・エクスプローラーの構成」を参照してください。

J2EE リソースアダプタの設定の注意事項については，サーバ管理コマンドによる J2EE リソースの設定（「4.　リソースアダプタの設定」）を参照してください。

14.2　リソースアダプタのインポート

次のリソースアダプタをインポートします。

- Server Plug-in 環境にあるリソースアダプタ
  Server Plug-in が実行されている環境にある RAR ファイルをインポートします。
- J2EE サーバが稼働するホストにあるリソースアダプタ
  J2EE サーバが稼働するホストにある，RAR ファイルをインポートします。

14.2.1　Server Plug-in 環境にあるリソースアダプタのインポート

Server Plug-in が実行されているホスト（ローカルホスト）にあるリソースアダプタをインポートします。

(1) 手順

ローカルホストのリソースアダプタをインポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース］フォルダ下の［リソース・アダプター］フォルダを選択します。**
2. **次の操作のどちらかで，リソースアダプタをインポートします。**
   - ビューツールバーの［  ］（インポート）をクリックして，サブメニューの［  リソース・アダプター］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  リソース・アダプターのインポート］を選択します。

   操作が実行されると，［リソース・アダプターのインポート］ダイアログが表示されます。
3. **インポートするリソースアダプタファイルを指定して，［開く］を実行します。**
   インポートが開始されます。

(2) 画面表示

リソースアダプタのインポートで使用する画面については，「21.1.20　リソースアダプタのインポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjimportres コマンドです。

リソースアダプタをインポートする場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースと接続するための設定（DB Connector を使用する場合）
  「4.2.1　DB Connector のインポート」
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）
  「4.4.1　リソースアダプタのインポート」

14.2.2　J2EE サーバ環境にあるリソースアダプタのインポート

J2EE サーバが稼働するホスト（接続先ホスト）にあるリソースアダプタをインポートします。

(1) 手順

接続先ホストのリソースアダプタをインポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース］フォルダ下の［リソース・アダプター］フォルダを選択します。**
2. **次の操作のどちらかで，接続先ホストのリソースアダプタをインポートします。**
   - ビューツールバーの［  ］（インポート）をクリックして，サブメニューの［  リソース・アダプター（Connector)］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  リソース・アダプター（Connector）のインポート］を選択します。

   操作が実行されると，［リソース・アダプター（Connector）のインポート］ダイアログが表示されます。
3. **接続先ホストのリソースアダプタファイルを絶対パスで指定して，［OK］を実行します。**
   インポートが開始されます。

(2) 画面表示

接続先ホストのリソースアダプタのインポートで使用する画面については，「21.1.19　リソースアダプタ（Connector）のインポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjimportres コマンドです。

リソースアダプタをインポートする場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースと接続するための設定（DB Connector を使用する場合）
  「4.2.1　DB Connector のインポート」
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）

「4.4.1　リソースアダプタのインポート」

14.3　リソースアダプタのプロパティ定義

属性ファイル編集エディタを起動して，次のリソースアダプタの属性や動作を定義します。

- インポートしたリソースアダプタ
- J2EE リソースアダプタとしてデプロイしたリソースアダプタ

14.3.1　属性ファイル編集エディタの起動

リソースアダプタのプロパティ定義には，Connector 属性ファイル編集エディタを使用します。

Connector 属性ファイル編集エディタは次の方法で起動します。

1. **［サーバー・エクスプローラー］ビューで，属性を編集する［< J2EE リソースアダプタ名>］または［<リソースアダプタ名>］（J2EE リソースフォルダ内）を選択します。**
2. **次のどれかの方法で，Connector 属性ファイル編集エディタを起動します。**
   - ［サーバー・エクスプローラー］ビューで選択したノードをダブルクリックします。
   - ビュープルダウンメニューの［  プロパティ］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  プロパティ］を選択します。

   Connector 属性ファイル編集エディタが起動されると，画面のエディタエリアに，Connector 属性ファイルを編集するためのフォームページが表示されます。
3. **エディタエリアのフォームページでプロパティを設定します。**
   設定手順はリソースアダプタが対応するコネクタのアーキテクチャのバージョンによって異なります。
   - Connector 1.0 の仕様に準拠するリソースアダプタ
     プロパティ設定手順については，「14.3.2　プロパティ設定手順（Connector 1.0 の場合）」を参照してください。
   - Connector 1.5 の仕様に準拠するリソースアダプタ
     プロパティ設定手順については，「14.3.3　プロパティ設定手順（Connector 1.5 の場合）」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されているリソースアダプタは，Connector 1.0 の仕様に準拠しています。

プロパティの設定操作および属性ファイル編集エディタについては，「11.3　プロパティの設定」を参照してください。

14.3.2 プロパティ設定手順（Connector 1.0 の場合）

プロパティ操作が実行されると，画面のエディタエリアに，Connector 属性ファイルを編集するためのフォームページが表示されます。エディタエリアのフォームページで，次のプロパティを設定します。

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

(1) 一般情報

1. **エディタエリアに表示された Connector 属性ツリーで，［リソース・アダプター］を選択します。**
   リソースアダプタのプロパティページが表示されます。
2. **表示されたプロパティページで，プロパティ項目を設定します。**
   一般情報のプロパティページについては，「21.2.2　Connector 属性設定ページ」の「(3)　リソースアダプタのプロパティページ」を参照してください。

(2) コンフィグレーションプロパティ

1. **エディタエリアに表示された Connector 属性ツリーで，［<コネクション定義識別子>］または［<コネクション定義識別子>］下の［コンフィグレーション・プロパティー］を選択して，マウスで右クリックします。**
   コンテキストメニューが表示されます。
2. **コンテキストメニューで［追加］を選択します。**
   コンフィグレーションプロパティのプロパティページが表示されます。
3. **コンフィグレーションプロパティを変更する場合は，表示されたツリーで，［<コンフィグレーションプロパティ名>］を選択します。**
   コンフィグレーションプロパティのプロパティページが表示されます。
4. **表示されたプロパティページで，プロパティ項目を設定します。**
   コンフィグレーションプロパティのページは，コンフィグレーションプロパティ名によって，次の 2 種類があります。
   - XAOpenString 以外のコンフィグレーションプロパティのページ
   - XAOpenString のコンフィグレーションプロパティのページ

   XAOpenString 以外のコンフィグレーションプロパティのページについては，「21.2.2　Connector 属性設定ページ」の「(8)　コンフィグレーションプロパティのプロパティページ（XAOpenString 以外）」を参照してください。XAOpenString のコンフィグレーションプロパティのページについては，「21.2.2　Connector 属性設定ページ」の「(9)　コンフィグレーションプロパティのプロパティページ（XAOpenString）」を参照してください。

表示されたプロパティページで，これらのコンフィグレーションプロパティ名を変更した場合は，手順 3. の操作を実行して画面を再表示してください。

(3) 実行時プロパティ

1. **エディタエリアに表示された Connector 属性ツリーで，[コネクター・ランタイム] 下の［プロパティー］を選択します。**
   実行時プロパティのプロパティページが表示されます。
2. **表示されたプロパティページで，プロパティ項目を設定します。**
   実行時プロパティのプロパティページについては，「21.2.2　Connector 属性設定ページ」の「(10)　プロパティのプロパティページ（Outbound リソースアダプタ）」を参照してください。

(4) 別名情報

1. **エディタエリアに表示された Connector 属性ツリーで，[コネクター・ランタイム] または［コネクター・ランタイム］下の［リソース外部プロパティー］をマウスで右クリックします。**
   コンテキストメニューが表示されます。
2. **コンテキストメニューで［追加］を選択します。**
   リソース外部プロパティのプロパティページが表示されます。
3. **別名を変更する場合は，表示されたツリーで，[コネクター・ランタイム］下の［<リソース外部プロパティ名>］を選択します。**
   リソース外部プロパティのプロパティページが表示されます。
4. **表示されたプロパティページで，プロパティ項目を設定します。**
   別名情報のプロパティページについては，「21.2.2　Connector 属性設定ページ」の「(12)　リソース外部プロパティのプロパティページ」を参照してください。

14.3.3　プロパティ設定手順（Connector 1.5 の場合）

プロパティ操作が実行されると，画面のエディタエリアに，Connector 属性ファイルを編集するためのフォームページが表示されます。エディタエリアのフォームページで，プロパティを設定します。

次のプロパティを設定します。

- コンフィグレーションプロパティ
- Outbound リソースアダプタのプロパティ
- Inbound リソースアダプタのプロパティ
- 管理対象オブジェクトのプロパティ
- 実行時プロパティ

(1) コンフィグレーションプロパティ

1. **エディタエリアに表示された Connector 属性ツリーで，[リソース・アダプター] または [リソース・アダプター] 下の [コンフィグレーション・プロパティー] を選択して，マウスで右クリックします。**
   コンテキストメニューが表示されます。
2. **コンテキストメニューで [追加] を選択します。**
   コンフィグレーションプロパティのプロパティページが表示されます。
3. **コンフィグレーションプロパティを変更する場合は，表示されたツリーで，[<コンフィグレーションプロパティ名>] を選択します。**
   コンフィグレーションプロパティのプロパティページが表示されます。
4. **表示されたプロパティページで，プロパティ項目を設定します。**
   コンフィグレーションプロパティのページは，コンフィグレーションプロパティ名によって，次の 2 種類があります。
   - XAOpenString 以外のコンフィグレーションプロパティのページ
   - XAOpenString のコンフィグレーションプロパティのページ

   XAOpenString 以外のコンフィグレーションプロパティのページについては，「21.2.2 Connector 属性設定ページ」の「(8) コンフィグレーションプロパティのプロパティページ（XAOpenString 以外）」を参照してください。XAOpenString のコンフィグレーションプロパティのページについては，「21.2.2 Connector 属性設定ページ」の「(9) コンフィグレーションプロパティのプロパティページ（XAOpenString）」を参照してください。
   表示されたプロパティページで，これらのコンフィグレーションプロパティ名を変更した場合は，手順 3. の操作を実行して画面を再表示してください。

(2) Outbound リソースアダプタのプロパティ

Connector 1.5 の仕様に準拠する Outbound リソースアダプタは，コネクション定義を複数持つ場合があります。プロパティ項目は，次のように分けて設定します。

- リソースアダプタ単位のプロパティ設定
- コネクション単位のプロパティ設定

(a) リソースアダプタ単位のプロパティ設定

1. **エディタエリアに表示された Connector 属性ツリーで，[アウトバウンド・リソース・アダプター] を選択します。**
   Outbound リソースアダプタのプロパティページが表示されます。
2. **表示されたプロパティページで，プロパティ項目を設定します。**
   Outbound リソースアダプタのプロパティページについては，「21.2.2 Connector 属性設定ページ」の「(4) Outbound リソースアダプタのプロパティページ」を参照してください。

(b) コネクション単位のプロパティ設定

次のプロパティを設定します。

- コンフィグレーションプロパティ
- 実行時プロパティ
- 別名情報

● コンフィグレーションプロパティ

1. **エディタエリアに表示された Connector 属性ツリーで，[<コネクション定義識別子>] または [<コネクション定義識別子>] 下の [コンフィグレーション・プロパティー] を選択して，マウスで右クリックします。**
   コンテキストメニューが表示されます。
2. **コンテキストメニューで [追加] を選択します。**
   コンフィグレーションプロパティのプロパティページが表示されます。
3. **コンフィグレーションプロパティを変更する場合は，表示されたツリーで，[<コンフィグレーションプロパティ名>] を選択します。**
   コンフィグレーションプロパティのプロパティページが表示されます。
4. **表示されたプロパティページで，プロパティ項目を設定します。**
   コンフィグレーションプロパティのページは，コンフィグレーションプロパティ名によって，次の 2 種類があります。
   - XAOpenString 以外のコンフィグレーションプロパティのページ
   - XAOpenString のコンフィグレーションプロパティのページ

   XAOpenString 以外のコンフィグレーションプロパティのページについては，「21.2.2 Connector 属性設定ページ」の「(8) コンフィグレーションプロパティのプロパティページ（XAOpenString 以外）」を参照してください。「21.2.2 Connector 属性設定ページ」の XAOpenString のコンフィグレーションプロパティのページについては，「(9) コンフィグレーションプロパティのプロパティページ（XAOpenString）」を参照してください。

表示されたプロパティページで，これらのコンフィグレーションプロパティ名を変更した場合は，手順 3. の操作を実行して画面を再表示してください。

● 実行時プロパティ

1. **エディタエリアに表示された Connector 属性ツリーで，[コネクター・ランタイム] 下の [プロパティー] を選択します。**
   実行時プロパティのプロパティページが表示されます。
2. **表示されたプロパティページで，プロパティ項目を設定します。**

実行時プロパティのプロパティページについては，「21.2.2 Connector 属性設定ページ」の「(10) プロパティのプロパティページ（Outbound リソースアダプタ）」を参照してください。

● 別名情報

1. **エディタエリアに表示された Connector 属性ツリーで，［コネクター・ランタイム］または［コネクター・ランタイム］下の［リソース外部プロパティー］をマウスで右クリックします。**
   コンテキストメニューが表示されます。
2. **コンテキストメニューで［追加］を選択します。**
   リソース外部プロパティのプロパティページが表示されます。
3. **別名を変更する場合は，表示されたツリーで，［コネクター・ランタイム］下の［＜リソース外部プロパティ名＞］を選択します。**
   リソース外部プロパティのプロパティページが表示されます。
4. **表示されたプロパティページで，プロパティ項目を設定します。**
   別名情報のプロパティページについては，「21.2.2　Connector 属性設定ページ」の「(12)　リソース外部プロパティのプロパティページ」を参照してください。

(3) Inbound リソースアダプタのプロパティ

Inbound リソースアダプタのプロパティは変更できません。プロパティページを表示して，設定されているプロパティ項目を参照してください。

1. **エディタエリアに表示された Connector 属性ツリーで，［インバウンド・リソース・アダプター］を選択します。**
2. **［インバウンド・リソース・アダプター］下の各ノードを選択します。**
   - メッセージリスナのプロパティを参照する場合
     ［＜メッセージリスナ名＞］を選択すると，メッセージリスナのプロパティページが表示されます。
   - ActivationSpec のプロパティを参照する場合
     ［アクティベーション・スペック］を選択すると，ActivationSpec のプロパティページが表示されます。
   - 必須コンフィグレーションプロパティを参照する場合
     ［＜必須コンフィグレーションプロパティ名＞］を選択すると，必須コンフィグレーションプロパティのプロパティページが表示されます。
3. **表示されたプロパティページで，プロパティ項目を参照します。**
   各プロパティページについては，「21.2.2　Connector 属性設定ページ」の「(14)　メッセージリスナのプロパティページ」，「(15)　ActivationSpec のプロパティページ」，または「(17)　必須コンフィグレーションプロパティのプロパティページ」を参照してください。

(4) 管理対象オブジェクトのプロパティ

1. **エディタエリアに表示された Connector 属性ツリーで，［リソース・アダプター］下の［管理対象オブジェクト］を選択して，マウスで右クリックします。**
   コンテキストメニューが表示されます。

2. **コンテキストメニューで［追加］を選択します。**
   管理対象オブジェクトのプロパティページが表示されます。
3. **表示されたプロパティページで，プロパティ項目を設定します。**
   管理対象オブジェクトのプロパティページについては，「21.2.2　Connector 属性設定ページ」の「(19)　管理対象オブジェクトのプロパティページ」を参照してください。

(5) 実行時プロパティ

1. **エディタエリアに表示された Connector 属性ツリーで，［リソース・アダプター・ランタイム］下の［プロパティー］を選択します。**
   実行時プロパティのプロパティページが表示されます。
2. **表示されたプロパティページで，プロパティ項目を設定します。**
   実行時プロパティのプロパティページについては，「21.2.2　Connector 属性設定ページ」の「(20)　プロパティのプロパティページ（リソースアダプタ）」を参照してください。

14.3.4　対応するサーバ管理コマンド

対応するサーバ管理コマンドは，次のコマンドです。

- cjgetresprop
- cjsetresprop
- cjgetrarprop
- cjsetrarprop

リソースアダプタのプロパティを設定する場合の，設定例，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースと接続するための設定（DB Connector を使用する場合）
  「4.2.2　DB Connector のプロパティ定義」
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）
  リソースアダプタのバージョンに対応して，次の説明を参照してください。
  - 「4.4.2　リソースアダプタのプロパティ定義（Connector 1.0 の場合）」
  - 「4.4.3　リソースアダプタのプロパティ定義（Connector 1.5 の場合）」

14.4　リソースアダプタのデプロイ

リソースアダプタをデプロイします。

(1) 手順

リソースアダプタをデプロイする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース］フォルダ内の［リソース・アダプター］フォルダ下の［<リソースアダプタ名>］を選択します。**
2. **次のどちらかの操作で，リソースアダプタをデプロイします。**
   - ビュープルダウンメニューの［ デプロイ］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ デプロイ］を選択します。

   デプロイが実行されます。

注意事項

実行時情報を含んだ J2EE リソースアダプタをインポートしてデプロイすると，［サーバー・エクスプローラー］ビューではデプロイされた J2EE リソースアダプタの稼働状態として，「停止」が仮定されて表示されます。このため，デプロイされた J2EE リソースアダプタの実際の状態と表示が不一致になる場合があります。実際の状態と一致させるためには，［表示の更新］を実行してください。［表示の更新］については，「11.2　表示の更新」の「(4)　J2EE リソースアダプタの状態表示」を参照してください。

(2) 画面表示

リソースアダプタのデプロイで使用する画面については，「21.1.29　リソースアダプタのデプロイ」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjdeployrar コマンドです。

リソースアダプタをデプロイする場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースと接続するための設定（DB Connector を使用する場合）
  「4.2.3　DB Connector のデプロイ」
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）
  「4.4.4　リソースアダプタのデプロイ」

14.5　J2EE リソースアダプタの接続テスト

J2EE リソースアダプタの接続テストをします。

(1) 手順

J2EE リソースアダプタの接続テストの手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダ下の［< J2EE リソースアダプタ名>］を選択します。**
2. **次のどちらかの操作で，J2EE リソースアダプタの接続テストをします。**
   - ビュープルダウンメニューの［  接続テスト］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  接続テスト］を選択します。

   操作が実行され，接続テストに成功した場合は，テスト成功のメッセージダイアログが表示されます。

(2) 画面表示

J2EE リソースアダプタの接続テストで使用する画面については，「21.1.28　接続テスト」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjtestres コマンドです。

リソースアダプタを接続テストする場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースと接続するための設定（DB Connector を使用する場合）
  「4.2.4　DB Connector の接続テスト」
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）
  「4.4.5　J2EE リソースアダプタの接続テスト」

14.6　J2EE リソースアダプタの開始と停止

J2EE アプリケーションを実行する前に，J2EE アプリケーションで利用するすべての J2EE リソースアダプタを開始してください。また，J2EE アプリケーションを停止する場合，J2EE アプリケーションが停止してから，J2EE リソースアダプタを停止してください。J2EE リソースアダプタの開始と停止の手順を，次に示します。

14.6.1　J2EE リソースアダプタの開始

リソースアダプタを開始します。

(1) 手順

リソースアダプタを開始する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダ下の［< J2EE リソースアダプタ名>］を選択します。**
2. **次の操作のどれかで，リソースアダプタを開始します。**
   - ビューツールバーの［  ］（開始）をクリックします。
   - ビュープルダウンメニューの［  開始］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  開始］を選択します。

   J2EE リソースアダプタの開始が実行されます。

(2) 画面表示

リソースアダプタの開始で使用する画面については，「21.1.4　開始」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjstartrar コマンドです。

リソースアダプタを開始する場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースを開始するための設定（DB Connector を使用する場合）
  「4.2.5　DB Connector の開始」
- そのほかのリソースを開始するための設定（ほかのリソースアダプタを使用する場合）
  「4.4.6　J2EE リソースアダプタの開始」

14.6.2　J2EE リソースアダプタの停止

J2EE リソースアダプタを停止します。

(1) 手順

J2EE リソースアダプタを停止する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダ下の［< J2EE リソースアダプタ名>］を選択します。**
2. **次の操作のどれかで，リソースアダプタを停止します。**
   - ビューツールバーの［ ］（停止）をクリックします。
   - ビュープルダウンメニューの［ 停止］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ 停止］を選択します。

   J2EE リソースアダプタの停止が実行されます。

(2) 画面表示

リソースアダプタの停止で使用する画面については，「21.1.7　停止」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjstoprar コマンドです。

リソースアダプタを停止する場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースを停止するための設定（DB Connector を使用する場合）
  「4.2.6　DB Connector の停止」
- そのほかのリソースを停止するための設定（ほかのリソースアダプタを使用する場合）
  「4.4.7　J2EE リソースアダプタの停止」

14.7　リソースアダプタの削除

J2EE リソースアダプタまたはリソースアダプタを削除します。

(1) 手順

J2EE リソースアダプタまたはリソースアダプタの削除手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダ下の［< J2EE リソースアダプタ名>］または［J2EE リソース］フォルダ下にある［リソース・アダプター］フォルダ下の［<リソース・アダプタ>］を選択します。**
2. **次の操作のどちらかで，リソースアダプタを削除します。**
   - ビュープルダウンメニューの［ 削除］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ 削除］を選択します。

   ［サーバー・エクスプローラー］ビューで，［< J2EE リソースアダプタ名>］を選択した場合，［< J2EE リソースアダプタ名>］が停止状態のときだけ操作は活性となっています。
   ［削除の確認］ダイアログが表示されます。
3. **［削除の確認］ダイアログで，［はい］または［いいえ］を選択します。**
   ［はい］を選択すると，削除は実行されます。［いいえ］を選択すると，削除は中止されます。

(2) 画面表示

リソースアダプタの削除で使用する画面については，「21.1.9　削除」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjundeployrar コマンドまたは cjdeleteres コマンドです。

リソースアダプタを削除する場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースを削除するための設定（DB Connector を使用する場合）
  「4.2.7　DB Connector のアンデプロイ」（J2EE リソースアダプタの削除の場合）
  「4.9　リソースアダプタの削除」（リソースアダプタの削除の場合）
- そのほかのリソースを削除するための設定
  「4.4.8　J2EE リソースアダプタのアンデプロイ」（J2EE リソースアダプタの削除の場合）
  「4.9　リソースアダプタの削除」（リソースアダプタの削除の場合）

14.8　J2EE リソースアダプタのエクスポート

J2EE リソースアダプタのエクスポートします。J2EE リソースアダプタのエクスポートには，実行時情報を含める場合と実行時情報を含めない場合があります。

注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

(1) 手順

● 実行時情報を含めて，J2EE リソースアダプタをエクスポートする場合

実行時情報を含んだ J2EE リソースアダプタをエクスポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダの［< J2EE リソースアダプタ名>］を選択します。**
2. **次のどちらかの方法で，J2EE リソースアダプタをエクスポートします。**
   - ビューツールバーの［  ］（エクスポート）をクリックすると表示される，サブメニューの［  Cosminexus リソース・アダプター］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  Cosminexus リソース・アダプターのエクスポート］を選択します。

   エクスポートが実行されると，［Cosminexus リソース・アダプターのエクスポート］ダイアログが表示されます。
3. **エクスポートする J2EE リソースアダプタ用ファイル（RAR ファイル）を指定して，［保存］を実行します。**
   エクスポートが開始されます。

● 実行時情報を含めないで，J2EE リソースアダプタをエクスポートする場合

実行時情報を含めないで，J2EE リソースアダプタをエクスポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダの［< J2EE リソースアダプタ名>］を選択します。**
2. **次のどちらかの方法で，J2EE リソースアダプタをエクスポートします。**
   - ビューツールバーの［  ］（エクスポート）をクリックすると表示される，サブメニューの［  J2EE リソース・アダプター］を選択します。

- 右クリックで表示されるコンテキストメニューの［  J2EE リソース・アダプターのエクスポート］を選択します。

エクスポートが実行されると，［J2EE リソース・アダプターのエクスポート］ダイアログが表示されます。

3. **エクスポートする** J2EE **リソースアダプタ用ファイル（**RAR **ファイル）を指定して，［保存］を実行します。**
エクスポートが開始されます。

（2） 画面表示

● 実行時情報を含めて，J2EE リソースアダプタをエクスポートする場合

実行時情報を含めて，J2EE リソースアダプタをエクスポートする場合，使用する画面については，「21.1.25　Cosminexus リソースアダプタのエクスポート」を参照してください。

● 実行時情報を含めないで，J2EE リソースアダプタをエクスポートする場合

実行時情報を含めないで，J2EE リソースアダプタをエクスポートする場合，使用する画面については，「21.1.27　J2EE リソースアダプタのエクスポート」を参照してください。

（3） 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjexportrar コマンドです。

リソースアダプタをエクスポートする場合の，注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースをエクスポートするための設定（DB Connector を使用する場合）
「4.2.8　DB Connector のエクスポート」
- そのほかのリソースをエクスポートするための設定（ほかのリソースアダプタを使用する場合）
「4.4.9　J2EE リソースアダプタのエクスポート」

14.9　Connector 属性のインポートとエクスポート

この節では，Server Plug-in を使用した Connector 属性のインポートとエクスポートについて説明します。

Server Plug-in では，J2EE リソースアダプタのプロパティを設定する方法として，次の二つの方法を提供しています。

- 属性ファイル編集エディタのフォームページでプロパティを設定します。属性ファイル編集エディタは，Server Plug-in のプロパティ操作で起動されます。
  Server Plug-in のプロパティ操作については，「11.3　プロパティの設定」を参照してください。
- Connector 属性を Connector 属性ファイルに取得し，プロパティ設定項目を編集します。編集した Connector 属性ファイルの値を，J2EE リソースアダプタの属性に反映します。

この節で，Connector 属性の取得（エクスポート）と Connector 属性の設定（インポート）の操作について説明します。

プロパティの設定項目については，「14.3　リソースアダプタのプロパティ定義」を参照してください。

14.9.1　Connector 属性のインポート

J2EE リソースアダプタの属性を，Connector 属性ファイルの値に変更します。

(1) 手順

Connector 属性をインポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダ下の［< J2EE リソースアダプタ名>］または［J2EE リソース］フォルダ下にある［リソース・アダプター］フォルダの［<リソースアダプタ名>］を選択します。**
2. **次の操作のどちらかで，Connector 属性をインポートします。**
   - ビューツールバーの［  ］（インポート）をクリックします。表示されるサブメニューの［  Connector 属性］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  Connector 属性のインポート］を選択します。

   ［サーバー・エクスプローラー］ビューで，［< J2EE リソースアダプタ名>］を選択した場合，［< J2EE リソースアダプタ名>］が停止状態のときだけ操作が活性と

なっています。
Connector 属性のインポートが実行されると，［Connector 属性のインポート］ダイアログが表示されます。

3. **インポートする Connector 属性ファイルを指定して，［開く］を実行します。**
インポートが開始されます。

(2) 画面表示

Connector 属性のインポートで使用する画面については，「21.1.16　Connector 属性のインポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，次のコマンドです。

- cjsetresprop コマンド
- cjsetrarprop コマンド

Connector 属性をインポートして，Connector 属性を設定する場合のサーバ管理コマンドについては，「3.3　属性ファイルによるプロパティの設定」を参照してください。

14.9.2　Connector 属性のエクスポート

J2EE リソースアダプタの属性を Connector 属性ファイルに取得します。

注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

(1) 手順

Connector 属性をエクスポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE リソース・アダプター］フォルダ下の［< J2EE リソースアダプタ名>］を選択します。**
2. **次の操作のどちらかで，Connector 属性を Connector 属性ファイルにエクスポートします。**
   - ビューツールバーの［ ］（エクスポート）をクリックして，表示されるサブメニューの［ Connector 属性］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ Connector 属性のエクス

ポート］を選択します。

エクスポートが実行されると，［Connector 属性のエクスポート］ダイアログが表示されます。

3. **エクスポートする属性ファイルを指定して，［保存］を実行します。**
エクスポートが開始されます。

（2） 画面表示

Connector 属性のエクスポートで使用する画面については，「21.1.22　Connector 属性のエクスポート」を参照してください。

（3） 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，次のコマンドです。

- cjgetresprop コマンド
- cjgetrarprop コマンド

Connector 属性をエクスポートして，Connector 属性を取得する場合のサーバ管理コマンドについては，「3.3　属性ファイルによるプロパティの設定」を参照してください。

15 J2EE アプリケーションに含まれるリソースアダプタの設定

この章では，Server Plug-in を使用した，J2EE アプリケーションに含まれるリソースアダプタの設定について説明します。

15.1　J2EE アプリケーションに含まれるリソースアダプタの設定の概要

J2EE アプリケーションに含めて使用するリソースアダプタの種類と，アプリケーション設定操作の概要について説明します。

(1) 利用できるリソースアダプタ

J2EE アプリケーションに含めて使用するリソースアダプタの種類は，サーバ管理コマンドを使用して，リソースアダプタを設定する場合と同じです。J2EE アプリケーションに含めて使用できるリソースアダプタの種類については，「5.1.1　利用できるリソースアダプタ」を参照してください。

(2) リソースアダプタの設定操作

J2EE アプリケーションに含めて使用するリソースアダプタの設定に必要な作業は次のとおりです。

- リソースアダプタを含む J2EE アプリケーションのインポート
- リソースアダプタのプロパティ定義
- リソースアダプタの接続テスト
- リソースアダプタを含む J2EE アプリケーションの開始
- リソースアダプタを含む J2EE アプリケーションの停止

また，必要に応じて J2EE アプリケーションに含まれるリソースアダプタの Connector 属性をインポートおよびエクスポートします。

Server Plug-in を使用した J2EE リソースの管理（GUI）の機能と，サーバ管理コマンド（CUI）の対応を次の表に示します。

表 15-1　Server Plug-in を使用した J2EE リソースの管理の機能とサーバ管理コマンドの対応

機能	対応するコマンド	参照先
リソースアダプタを含む J2EE アプリケーションのインポート	cjimportapp	15.2
リソースアダプタのプロパティ定義	cjgetappprop（-type rar 指定）， cjsetappprop（-type rar 指定）	15.3
リソースアダプタの接続テスト	cjtestres（-name ＜ J2EE アプリケーション名＞ -type rar 指定）	15.4
リソースアダプタを含む J2EE アプリケーションの開始	cjstartapp	15.5

機能	対応するコマンド	参照先
リソースアダプタを含む J2EE アプリケーションの停止	cjstopapp	15.5
Connector 属性のインポートとエクスポート	cjgetappprop（-type rar 指定）， cjsetappprop（-type rar 指定）	15.6

J2EE アプリケーションに含まれるリソースアダプタの設定の注意事項については，サーバ管理コマンドによるリソースアダプタの設定（「5.　J2EE アプリケーションに含まれるリソースアダプタの設定」）を参照してください。

15.2　リソースアダプタを含む J2EE アプリケーションのインポート

リソースアダプタを含む J2EE アプリケーションをインポートします。

インポートを実行する前に，J2EE アプリケーションに含まれるリソースアダプタと同じ表示名，または別名のリソースアダプタが J2EE リソースアダプタとしてデプロイされていないことを確認してください。デプロイされているリソースアダプタは，［サーバー・エクスプローラー］ビューの［J2EE リソース・アダプター］フォルダ内の［＜J2EE リソースアダプタ名＞］に表示されています。

次の J2EE アプリケーションをインポートできます。

- アーカイブ形式の J2EE アプリケーション
  J2EE サーバの作業ディレクトリ以下に，EJB-JAR/WAR/RAR ファイルのコンポーネントを持つ J2EE アプリケーションです。
  アーカイブ形式の J2EE アプリケーションのインポート操作については，「17.1.1 アーカイブ形式の J2EE アプリケーションのインポート」を参照してください。
- 展開ディレクトリ形式の J2EE アプリケーション
  EJB-JAR や WAR などのアプリケーションの実体を，J2EE サーバの外部にある一定のルールに従ったファイル／ディレクトリに持つ J2EE アプリケーションです。ただし，展開ディレクトリ形式の J2EE アプリケーションに含まれる RAR ファイルはアーカイブ形式で格納されます。
  展開ディレクトリ形式の J2EE アプリケーションのインポート操作については，「17.1.2　展開ディレクトリ形式の J2EE アプリケーションのインポート」を参照してください。

15.3　リソースアダプタのプロパティ定義

属性ファイル編集エディタを起動して，J2EE アプリケーションに含まれるリソースアダプタの属性や動作を定義します。

15.3.1　属性ファイル編集エディタの起動

リソースアダプタのプロパティ定義には，属性ファイル編集エディタを使用します。

属性ファイル編集エディタには，次の二つのエディタがあります。

- アプリケーション統合属性ファイル編集エディタ
- Connector 属性ファイル編集エディタ

J2EE アプリケーションに含まれるリソースアダプタは，どちらのエディタを使用してもリソースアダプタのプロパティ定義ができます。ただし，これらの属性ファイル編集エディタを，同時に起動することはできません。

(1) アプリケーション統合属性ファイル編集エディタの起動

アプリケーション統合属性ファイル編集エディタは，次の方法で起動します。

1. **［サーバー・エクスプローラー］ビューで，プロパティを設定するリソースアダプタが含まれている［< J2EE アプリケーション>］を選択します。**
2. **次のどれかの方法で，アプリケーション統合属性ファイル編集エディタを起動します。**
   - ［サーバー・エクスプローラー］ビューで選択したノードをダブルクリックします。
   - ビュープルダウンメニューの［ プロパティ］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ プロパティ］を選択します。

     アプリケーション統合属性ファイル編集エディタが起動されると，画面のエディタエリアに，アプリケーション統合属性ファイルを編集するためのフォームページが表示されます。
3. **フォームページの「RAR」タブをクリックしてください。**

   画面のエディタエリアに，J2EE アプリケーションに含まれるリソースアダプタの［Connector 属性］フォームページが表示されます。フォームページで，リソースアダプタのプロパティを設定してください。

(2) Connector 属性ファイル編集エディタの起動

Connector 属性ファイル編集エディタは，次の方法で起動します。

1. **［サーバー・エクスプローラー］ビューで，［< J2EE アプリケーション>］下の［< リソースアダプタ名>］を選択します。**

2. **次のどれかの方法で，Connector 属性ファイル編集エディタを起動します。**
   - ［サーバー・エクスプローラー］ビューで選択したノードをダブルクリックします。
   - ビュープルダウンメニューの［ プロパティ］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ プロパティ］を選択します。

     Connector 属性ファイル編集エディタが起動されると，画面のエディタエリアに，Connector 属性ファイルを編集するためのフォームページが表示されます。フォームページで，リソースアダプタのプロパティを設定してください。

15.3.2　プロパティ設定手順

属性ファイル編集エディタのフォームページで，プロパティを設定します。

なお，設定手順は，リソースアダプタが対応するコネクタアーキテクチャのバージョンによって異なります。

- Connector 1.0 の仕様に準拠するリソースアダプタ
  プロパティ設定手順については，「14.3.2　プロパティ設定手順（Connector 1.0 の場合）」を参照してください。
- Connector 1.5 の仕様に準拠するリソースアダプタ
  プロパティ設定手順については，「14.3.3　プロパティ設定手順（Connector 1.5 の場合）」を参照してください。

15.4　リソースアダプタの接続テスト

J2EE アプリケーションに含まれるリソースアダプタの接続テストをします。

(1) 手順

J2EE アプリケーションに含まれるリソースアダプタの接続テストの手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［< J2EE アプリケーション>］下の［< リソースアダプタ名>］を選択します。**
2. **次のどちらかの操作で，リソースアダプタの接続テストをします。**
   - ビュープルダウンメニューの［  接続テスト］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  接続テスト］を選択します。

   操作が実行され，接続テストに成功した場合は，テスト成功のメッセージダイアログが表示されます。

(2) 画面表示

リソースアダプタの接続テストで使用する画面については，「21.1.28　接続テスト」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjtestres（-name ＜ J2EE アプリケーション名＞ -type rar 指定）コマンドです。

J2EE アプリケーションに含まれるリソースアダプタを接続テストする場合の注意事項およびサーバ管理コマンドについては，次の説明を参照してください。

- データベースと接続するための設定（DB Connector を使用する場合）
  「4.2.4　DB Connector の接続テスト」
- そのほかのリソースと接続するための設定（ほかのリソースアダプタを使用する場合）
  「4.4.5　J2EE リソースアダプタの接続テスト」

15.5　リソースアダプタを含む J2EE アプリケーションの開始と停止

J2EE アプリケーションを開始すると，J2EE アプリケーションに含まれるすべてのリソースアダプタは開始されます。また，J2EE アプリケーションを停止すると，J2EE アプリケーションに含まれるすべてのリソースアダプタは停止されます。リソースアダプタの開始順序および停止順序の制御はできません。

J2EE アプリケーションの開始については，「19.2　J2EE アプリケーションの開始」を参照してください。

J2EE アプリケーションの停止については，「19.3　J2EE アプリケーションの停止」を参照してください。

注意事項

J2EE アプリケーションがリソースアダプタを含み，application.xml の <module> タグ以下で RAR を記述している場合，EJB-JAR や WAR が，リソースアダプタを参照していなくても，J2EE アプリケーション開始時にリソースアダプタは開始されます。

15.6　Connector 属性のインポートとエクスポート

J2EE アプリケーションに含まれるリソースアダプタの Connector 属性の取得（エクスポート）と Connector 属性の設定（インポート）の手順について説明します。

15.6.1　Connector 属性のインポート

J2EE アプリケーションに含まれるリソースアダプタの属性を，Connector 属性ファイルの値に変更します。

(1) 手順

Connector 属性をインポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［< J2EE アプリケーション>］下の［<リソースアダプタ名>］を選択します。**
2. **次の操作のどちらかで，Connector 属性をインポートします。**
   - ビューツールバーの［  ］（インポート）をクリックします。表示されるサブメニューの［  Connector 属性］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  Connector 属性のインポート］を選択します。

   ［サーバー・エクスプローラー］ビューで，［< J2EE アプリケーション>］が停止状態のときだけ操作が活性となっています。
   Connector 属性のインポートが実行されると，［Connector 属性のインポート］ダイアログが表示されます。
3. **インポートする Connector 属性ファイルを指定して，［開く］を実行します。**
   インポートが開始されます。

(2) 画面表示

Connector 属性のインポートで使用する画面については，「21.1.16　Connector 属性のインポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjsetappprop（-type rar 指定）コマンドです。

15.6.2　Connector 属性のエクスポート

J2EE アプリケーションに含まれるリソースアダプタの属性を，Connector 属性ファイル

に取得します。

注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

(1) 手順

Connector 属性をエクスポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［< J2EE アプリケーション>］下の［<リソースアダプタ名>］を選択します。**
2. **次の操作のどちらかで，Connector 属性を Connector 属性ファイルにエクスポートします。**
   - ビューツールバーの［  ］（エクスポート）をクリックして，表示されるサブメニューの［  Connector 属性］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  Connector 属性のエクスポート］を選択します。

   エクスポートが実行されると，［Connector 属性のエクスポート］ダイアログが表示されます。
3. **エクスポートする属性ファイルを指定して，［保存］を実行します。**
   エクスポートが開始されます。

(2) 画面表示

Connector 属性のエクスポートで使用する画面については，「21.1.22　Connector 属性のエクスポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjsetappprop（-type rar 指定）コマンドです。

16 Server Plug-in による J2EE アプリケーションの設定の概要

この章では，Server Plug-in を使用した，J2EE アプリケーションの設定の概要について説明します。

16.1 J2EE アプリケーション設定の機能一覧

サーバ管理コマンド（CUI）の機能と比較して，Server Plug-in を使用した J2EE アプリケーションの管理（GUI）で使用できる機能を次に示します。

表 16-1 J2EE アプリケーション設定操作の機能比較

機能		対応するコマンド	Server Plug-in での操作	参照先
J2EE アプリケーション／属性のインポート	アーカイブ形式の J2EE アプリケーション	cjimportapp	○	17.1.1
	展開ディレクトリ形式の J2EE アプリケーション（アプリケーションディレクトリパス指定）	cjimportapp（-a ＜アプリケーションディレクトリパス＞オプション指定）	○	17.1.2
	展開ディレクトリ形式の J2EE アプリケーション（展開ディレクトリパス指定）	cjimportapp（-d ＜展開ディレクトリパス＞オプション指定）	×	―
	アプリケーション統合属性のインポート	cjsetappprop	○	20.2
J2EE アプリケーション／属性のエクスポート	標準の J2EE アプリケーションのエクスポート	cjexportapp（-raw オプション指定）	○	17.2
	実行時情報付きの J2EE アプリケーションのエクスポート	cjexportapp（-normal オプション指定）	○	17.2
	アプリケーション統合属性のエクスポート	cjgetappprop	○	20.3

機能		対応するコマンド	Server Plug-in での操作	参照先
J2EE アプリケーションの実行	J2EE アプリケーションの通常開始	cjstartapp	○	19.2
	JSP をコンパイルして開始	cjstartapp（-jsp オプション指定）	×	－
	J2EE アプリケーションの通常停止	cjstopapp	○	19.3.1
	J2EE アプリケーションの通常停止実行後，自動強制停止	cjstopapp（-cancel オプション指定）	×	－
	J2EE アプリケーションの強制停止	cjstopapp（-force オプション指定）	○	19.3.2
	J2EE アプリケーションのプロパティの設定※1	cjgetappprop, cjsetappprop	○	18.1
	J2EE アプリケーションの一覧の参照	（cjlistapp）	△※2	－
	J2EE アプリケーションの削除	cjdeleteapp	○	19.4
	アーカイブ形式のアプリケーションの入れ替え	cjreplaceapp	○	19.5.1
	展開ディレクトリ形式のアプリケーションの入れ替え	cjreloadapp	○	19.5.2
	J2EE アプリケーション名の変更	cjrenameapp	○	19.6
	RMI-IIOP スタブとインタフェースの取得	cjgetstubsjar	○	19.7

（凡例）
　○：使用できる　△：類似機能　×：使用できない　－：該当なし
　（ ）：類似機能に対応するサーバ管理コマンド（CUI）
注※1　J2EE アプリケーションのプロパティ設定項目の比較については，「18.1　J2EE アプリケーションのプロパティ設定の概要」を参照してください。
注※2　［サーバー・エクスプローラー］ビューの［J2EE アプリケーション］フォルダに，インポートされている J2EE アプリケーションの状態が表示されています。［サーバー・エクスプローラー］ビューについては，「11.1.3　サーバー・エクスプローラーの構成」を参照してください。

次の機能は，Server Plug-in では使用できません。

- J2EE アプリケーションの作成
- Enterprise Bean（EJB-JAR）のインポート
- サーブレットと JSP のインポート
- J2EE アプリケーションの新規作成
- J2EE アプリケーションへのライブラリ JAR ファイルの追加

- ライブラリ JAR ファイルの一覧の参照
- J2EE アプリケーションからのライブラリ JAR ファイルの削除
- J2EE アプリケーションへの参照ライブラリの設定
- トランザクション一覧の参照

16.2　ビューツールバーでの J2EE アプリケーション設定操作

Server Plug-in での J2EE アプリケーション設定は，次のように操作します。

1. **［サーバー・エクスプローラー］ビューに表示されたツリー項目（ノード）を選択します。**
   J2EE アプリケーション設定操作で選択するノードを次に示します。
   - ［J2EE アプリケーション］フォルダ
   - ［＜ J2EE アプリケーション名＞］
2. **ビューツールバーで操作を指定します。**

［サーバー・エクスプローラー］ビューの構成については，「11.1.3　サーバー・エクスプローラーの構成」を参照してください。

［サーバー・エクスプローラー］ビューに表示されたノードと，そのノードが実行できる操作については，「21.1.1　サーバー・エクスプローラーの基本操作」を参照してください。

16.3　J2EE アプリケーションの状態表示

J2EE アプリケーションの稼働状態は，［サーバー・エクスプローラー］ビューのステータスアイコンで表示します。ステータスアイコンは，アイコンの左上に付加されます。

J2EE アプリケーションの状態表示と付加されるステータスアイコンについては，「11.2　表示の更新」の「(3)　J2EE アプリケーションの状態表示」を参照してください。

17 J2EE アプリケーションのインポートとエクスポート

この章では，Server Plug-in を使用した J2EE アプリケーションのインポートとエクスポートについて説明します。

17.1　J2EE アプリケーションのインポート

J2EE アプリケーションをインポートします。

インポートする J2EE アプリケーションには，次の形式があります。

- アーカイブ形式の J2EE アプリケーション
- 展開ディレクトリ形式の J2EE アプリケーション

インポートする J2EE アプリケーションの形式およびインポートを実行するときの注意事項については，サーバ管理コマンドの場合と同じです。詳細は「8.1　J2EE アプリケーションのインポート」を参照してください。

17.1.1　アーカイブ形式の J2EE アプリケーションのインポート

アーカイブ形式の J2EE アプリケーションをインポートします。

(1) 手順

アーカイブ形式の J2EE アプリケーションをインポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE アプリケーション］フォルダを選択します。**
2. **次のどちらかの方法で，アーカイブ形式の J2EE アプリケーションをインポートします。**
   - ビューツールバーの［  ］（インポート）をクリックします。表示されるサブメニューの［  J2EE アプリケーション（アーカイブ形式)］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  J2EE アプリケーション（アーカイブ形式）のインポート］を選択します。

   インポートが実行されると，［J2EE アプリケーション（アーカイブ形式）のインポート］ダイアログが表示されます。
3. **表示されたダイアログで，インポートするアーカイブ形式の J2EE アプリケーションファイルを指定して，［開く］を実行します。**
   インポートが開始されます。

(2) 画面表示

アーカイブ形式の J2EE アプリケーションのインポートで使用する画面については，「21.1.18　アーカイブ形式 J2EE アプリケーションのインポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjimportapp コマンドです。

アーカイブ形式の J2EE アプリケーションをインポートする場合の，サーバ管理コマンドについては，「8.1.1　アーカイブ形式の J2EE アプリケーションのインポート」を参照してください。

17.1.2　展開ディレクトリ形式の J2EE アプリケーションのインポート

展開ディレクトリ形式の J2EE アプリケーションをインポートします。

(1) 手順

展開ディレクトリ形式の J2EE アプリケーションをインポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，［J2EE アプリケーション］フォルダを選択します。**
2. **次のどちらかの方法で，展開ディレクトリ形式の J2EE アプリケーションをインポートします。**
   - ビューツールバーの［  ］（インポート）をクリックします。表示されるサブメニューの［  J2EE アプリケーション（展開ディレクトリー形式）］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  J2EE アプリケーション（展開ディレクトリー形式）のインポート］を選択します。

   インポートが実行されると，［J2EE アプリケーション（展開ディレクトリー形式）のインポート］ダイアログが表示されます。
3. **表示されたダイアログで，インポートする J2EE アプリケーションのアプリケーションディレクトリパスを指定して，［開く］を実行します。**
   インポートが開始されます。

注意事項

J2EE アプリケーションのディレクトリに，UNC 名を含むパスを指定した場合，展開ディレクトリ形式の J2EE アプリケーションのインポートはできません。

(2) 画面表示

展開ディレクトリ形式の J2EE アプリケーションのインポートで使用する画面については，「21.1.17　展開ディレクトリ形式の J2EE アプリケーションのインポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，-a オプションを指定した cjimportapp コマンドです。

展開ディレクトリ形式の J2EE アプリケーションをインポートする場合の，サーバ管理コマンドについては，「8.1.2　展開ディレクトリ形式の J2EE アプリケーションのインポート」の「(1)　アプリケーションディレクトリを展開ディレクトリ形式のアプリケーションとしてインポート」を参照してください。

17.2　J2EE アプリケーションのエクスポート

J2EE アプリケーションをエクスポートします。

注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

(1) 手順

J2EE アプリケーションのエクスポートには，実行時情報を含める場合と実行時情報を含めない場合があります。

● 実行時情報を含めて，J2EE アプリケーションをエクスポートする場合

実行時情報を含んだ J2EE アプリケーションをエクスポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，エクスポートする［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，J2EE アプリケーションをエクスポートします。**
   - ビューツールバーの［  ］（エクスポート）をクリックします。表示されるサブメニューの［  Cosminexus アプリケーション］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  Cosminexus アプリケーションのエクスポート］を選択します。

   エクスポートが実行されると，［Cosminexus アプリケーションのエクスポート］ダイアログが表示されます。
3. **エクスポートする J2EE アプリケーション用ファイル（ZIP ファイル）を指定して，［保存］を実行します。**
   エクスポートが開始されます。

● 実行時情報を含めないで，J2EE アプリケーションをエクスポートする場合

実行時情報を含めないで，J2EE アプリケーションをエクスポートする手順を次に示します。

1. **［サーバー・エクスプローラー］のビューで，エクスポートする［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，J2EE アプリケーションをエクスポートします。**
   - ビューツールバーの［  ］（エクスポート）をクリックします。表示されるサブ

メニューの［  J2EE アプリケーション］を選択します。

- 右クリックで表示されるコンテキストメニューの［  J2EE アプリケーションのエクスポート］を選択します。

エクスポートが実行されると，［J2EE アプリケーションのエクスポート］ダイアログが表示されます。

3. **エクスポートする J2EE アプリケーション用ファイル（EAR ファイル）を指定して，［保存］を実行します。**
エクスポートが開始されます。

（2）画面表示

● 実行時情報を含めて，J2EE アプリケーションをエクスポートする場合

実行時情報を含めて，J2EE アプリケーションをエクスポートする場合，使用する画面については，「21.1.23　Cosminexus アプリケーションのエクスポート」を参照してください。

● 実行時情報を含めないで，J2EE アプリケーションをエクスポートする場合

実行時情報を含めないで，J2EE アプリケーションをエクスポートする場合，使用する画面については，「21.1.26　J2EE アプリケーションのエクスポート」を参照してください。

（3）対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjexportapp コマンドです。

J2EE アプリケーションをエクスポートする場合の，サーバ管理コマンドについては，「8.2　J2EE アプリケーションのエクスポート」を参照してください。

18 J2EE アプリケーションのプロパティ設定

この章では，Server Plug-in を使用した J2EE アプリケーションのプロパティの設定方法について説明します。

18.1　J2EE アプリケーションのプロパティ設定の概要

J2EE アプリケーションのプロパティ設定とは，インポートした J2EE アプリケーションの属性や動作を定義することです。この節では，次の説明をします。

- 属性ファイル編集エディタによる J2EE アプリケーションのプロパティ設定操作
- Enterprise Bean のプロパティ設定項目
- サーブレットと JSP のプロパティ設定項目
- J2EE アプリケーション（共通）のプロパティ設定項目

（1）属性ファイル編集エディタによる J2EE アプリケーションのプロパティ設定操作

J2EE アプリケーションのプロパティの設定は，属性ファイル編集エディタを使用します。属性ファイル編集エディタは次の方法で起動します。

1. **［サーバー・エクスプローラー］ビューで，属性を編集する［< J2EE アプリケーション名>］を選択します。**
2. **プロパティ操作を実行します。**

プロパティの設定操作および属性ファイル編集エディタについては，「11.3　プロパティの設定」を参照してください。

プロパティ操作が実行されると，画面のエディタエリアに，属性ファイルを編集するためのフォームページが表示されます。エディタエリアのフォームページで，プロパティを設定します。

プロパティの設定操作に対応するサーバ管理コマンドは，cjgetappprop コマンドおよび cjsetappprop コマンドです。J2EE アプリケーションのプロパティ設定のサーバ管理コマンドについては，「3.3　属性ファイルによるプロパティの設定」を参照してください。

（2）Enterprise Bean のプロパティ設定項目

Enterprise Bean が含まれる J2EE アプリケーションを作成する場合およびカスタマイズする場合に，必要に応じて設定してください。

サーバ管理コマンド（CUI）のプロパティ設定と比較して，Server Plug-in を使用した J2EE アプリケーションの管理（GUI）で設定できる Enterprise Bean のプロパティ項目を次に示します。

表 18-1 Enterprise Bean のプロパティ編集項目

編集項目	内容	Server Plug-in でのプロパティ設定	参照先
Enterprise Bean のリファレンス定義	J2EE アプリケーションを構成する，次の Enterprise Bean のリファレンスを定義します。 • Session Bean • Entity Bean • Message-driven Bean	△	18.2
ほかの Enterprise Bean のリファレンス定義	ほかの Enterprise Bean へのリファレンスについて定義します。	○	18.2.1
メールコンフィグレーションのリファレンス定義	メールサーバへのリファレンスについて定義します。	×	－
リソースアダプタのリファレンス定義	リソースアダプタへのリファレンスについて定義します。	○	18.2.2
リソース環境のリファレンス定義	リソース環境へのリファレンスについて定義します。	○	18.2.3
Message-driven Bean のメッセージ参照定義	Message-driven Bean の Connection Factory と Destination を参照する個所に，実際の Connection Factory と Destination をマッピングします。	○	18.3
トランザクション属性の定義	トランザクション属性について定義します。	△	18.4
Entity Bean の CMP 定義	Entity Bean の永続性管理をコンテナに任せる場合に定義します。	×	－
CMP の設定	単一プライマリキーまたは 複合プライマリキーを使用して Entity Bean の永続性管理をコンテナに任せる場合に定義します。	×	－
CMP1.x とデータベースのマッピング	CMP1.x Entity Bean のフィールドをデータベース上の表にマッピングします。	×	－
CMP2.x とデータベースのマッピング	CMP2.x Entity Bean のフィールドをデータベース上の表にマッピングします。	×	－
Enterprise Bean の実行時属性の定義	Enterprise Bean の実行時の動作について設定します。必要に応じて設定してください。	○	18.8
Stateful Session Bean の実行時プロパティの設定	Stateful Session Bean の実行時の動作についてのプロパティを設定します。	○	18.8.1
Stateless Session Bean の実行時プロパティの設定	Stateless Bean の実行時の動作についてのプロパティを設定します。	○	18.8.2

編集項目	内容	Server Plug-in でのプロパティ設定	参照先
Entity Bean の実行時プロパティの設定	Entity Bean の実行時の動作についてのプロパティを設定します。	○	18.8.3
Message-driven Bean の実行時プロパティの設定	Message-driven Bean の実行時の動作についてのプロパティを設定します。	○	18.8.4

（凡例）
○：設定できる　△：一部設定できない　×：設定できない　－：該当なし

(3) サーブレットと JSP のプロパティ設定項目

サーブレットおよび JSP を含む J2EE アプリケーションを作成する場合に，必要に応じて設定してください。

サーバ管理コマンド（CUI）のプロパティ設定と比較して，Server Plug-in を使用した J2EE アプリケーションの管理（GUI）で設定できるサーブレットと JSP のプロパティ項目を次に示します。

表 18-2　サーブレットと JSP のプロパティ編集項目

編集項目	内容	Server Plug-in でのプロパティ設定	参照先
サーブレットと JSP のリファレンス定義	Enterprise Bean，リソース（データベースまたはメールサーバ）へのリファレンス（リソース参照）について定義します。	△	18.5
Enterprise Bean リファレンス定義	Enterprise Bean へのリファレンスについて定義します。	○	18.5.1
メールコンフィグレーションのリファレンスの定義	メールサーバへのリファレンスについて定義します。	×	－
リソースアダプタリファレンス定義	リソースアダプタへのリファレンスについて定義します。	○	18.5.2
リソース環境リファレンス定義	リソース環境リファレンスを定義します。	○	18.5.3
サーブレットと JSP のマッピング定義	サーブレットと JSP のマッピングを定義します。	○	18.6
フィルタの設定	フィルタを追加し，マッピングを定義します。	△	18.7
フィルタの追加	フィルタを追加します。	×	18.7.1

編集項目	内容	Server Plug-in でのプロパティ設定	参照先
フィルタのマッピング定義	フィルタへのマッピングを定義します。	○	18.7.2
フィルタの削除	フィルタを削除します。	×	18.7.1
サーブレットと JSP の実行時属性の定義	サーブレットおよび JSP の実行時属性を定義します。	○	18.9
J2EE アプリケーションのコンテキストルート定義	J2EE アプリケーションのコンテキストルートを設定します。	○	18.9.1
Web アプリケーション単位での同時実行スレッド数制御の定義	Web コンテナで Web アプリケーション単位での同時実行スレッド数を制御するための定義をします。	○	18.9.2
URL グループ単位での同時実行スレッド数制御の定義	URL グループ単位での同時実行スレッド数を制御するための定義をします。	○	18.9.3
URL グループ単位の実行待ちリクエスト数の監視の定義	URL グループ単位の実行待ちリクエスト数を監視するための定義をします。	○	18.9.4
デフォルトの文字エンコーディングの設定	次の文字エンコーディングを設定します。 • リクエストボディおよびクエリの文字エンコーディング • レスポンスボディの文字エンコーディング • JSP ファイルの文字エンコーディング	○	18.10
サーブレットと JSP のエラー通知の設定	サーブレット，JSP でエラーが発生した場合に，J2EE アプリケーションにエラーを通知するための設定をします。	○	18.13

（凡例）
○：設定できる　△：一部設定できない　×：設定できない　－：該当なし

（4） J2EE アプリケーション（共通）のプロパティ設定項目

J2EE アプリケーションの構成に関係なく，必要に応じて設定してください。

サーバ管理コマンド（CUI）のプロパティ設定と比較して，Server Plug-in を使用した J2EE アプリケーションの管理（GUI）で設定できる J2EE アプリケーション（共通）のプロパティ項目を次に示します。

表 18-3 J2EE アプリケーション（共通）のプロパティ編集項目

編集項目	内容	Server Plug-in でのプロパティ設定	参照先
JNDI 名前空間に登録される名称の参照と変更	J2EE アプリケーションの JNDI 名前空間への登録名を参照して，必要に応じて別名を付与します。	×	−
J2EE アプリケーション名の参照	J2EE アプリケーション名を参照するための設定をします。	×	−
Enterprise Bean 名の参照と変更	Enterprise Bean 名を参照するための設定および Enterprise Bean 名を変更します。	×	−
CTM のスケジューリング	CTM を利用してキューのスケジューリングをするかどうかと，スケジューリング方法について設定します。	○	18.11
J2EE アプリケーション単位のスケジューリング	J2EE アプリケーション単位の CTM との連携についての設定をします。	○	18.11.1
Stateless Session Bean 単位のスケジューリング	Stateless Session Bean 単位のスケジューリングの設定をします。	○	18.11.2
起動順序の設定	J2EE アプリケーションの起動順序，および J2EE アプリケーションに含まれる Enterprise Bean（EJB-JAR）やサーブレットまたは JSP（WAR）の起動順序を設定します。	○	18.12
J2EE アプリケーションの起動順序の設定	J2EE アプリケーションの起動順序を設定します。	○	18.12.1
Enterprise Bean の起動順序の設定	J2EE アプリケーションに含まれる Enterprise Bean（EJB-JAR）の起動順序を設定します。	○	18.12.2
サーブレットと JSP の起動順序の設定	J2EE アプリケーションに含まれるサーブレットまたは JSP（WAR）の起動順序を設定します。	○	18.12.3
セキュリティロールの設定	セキュリティロールを使用したユーザ管理をする場合に必要な設定です。ユーザとロールの登録と対応づけができます。	×	−
ユーザの設定	ユーザを登録します。ロールとの対応づけもできます。	×	−
ロールの設定	ロールを登録します。ユーザとの対応づけもできます。	×	−
セキュリティロールのリファレンス定義	セキュリティロールへのリファレンスについて定義します。	×	−
Enterprise Bean のセキュリティロールリファレンスの定義	セキュリティロールを参照している個所に，実際に J2EE サーバが管理しているセキュリティロールをマッピングします。	×	−

編集項目		内容	Server Plug-in でのプロパティ設定	参照先
	サーブレットと JSP のセキュリティロールリファレンスの定義	セキュリティロールへのリファレンスについて定義します。	×	−
セキュリティの定義（メソッドパーミッション）		セキュリティロールによるアクセス権を設定する場合のメソッドのパーミッションを定義します。また，すべてのユーザに対してアクセス権限を設定する場合のメソッドのパーミッションを定義します。	×	−
セキュリティの定義（セキュリティアイデンティティ）		セキュリティの情報であるセキュリティアイデンティティとして UseCallerIdentity または Run as を設定します。	×	−
	Enterprise Bean のセキュリティアイデンティティ	J2EE アプリケーション内で参照しているセキュリティアイデンティティ情報に，J2EE サーバが実際に管理しているユーザ ID をマッピングします。	×	−
	サーブレットと JSP のセキュリティアイデンティティ	セキュリティアイデンティティを定義します。	×	−

（凡例） ○：設定できる　×：設定できない　−：該当なし

18.2　Enterprise Bean のリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean のリファレンス（リソース参照）を定義します。Enterprise Bean には，次の種類があります。

- Session Bean
- Entity Bean
- Message-driven Bean

それぞれの Enterprise Bean には，次に示すリファレンスがあります。

- Enterprise Bean リファレンス
  ほかの Enterprise Bean 呼び出しのための参照です。
- リソースリファレンス
  データベースへの参照です。リソースアダプタリファレンス，およびリソース環境リファレンスがあります。

18.2.1　ほかの Enterprise Bean のリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean が，ほかの Enterprise Bean を呼び出している場合，リファレンスを解決するためのプロパティを設定します。

(1) 手順

ほかの Enterprise Bean のリファレンス定義を設定する手順を次に示します。

参照する Enterprise Bean のアクセスタイプには，次の 2 種類があります。

- リモートインタフェース
- ローカルインタフェース

● リモートインタフェースの場合

1. **エディタエリアに表示されている，アプリケーション統合属性（＜ J2EE アプリケーション名＞）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. EJB-JAR **属性フォームページで，**J2EE **アプリケーションを構成する** Enterprise Bean **の種類に対応する属性を選択します。**
   Enterprise Bean の種類に対応する属性を次に示します。
   - ［＜ Session Bean 属性名＞］
   - ［＜ Entity Bean 属性名＞］
   - ［＜ Message-driven Bean 属性名＞］

   選択した属性のツリーが表示されます。
3. **ほかの** Enterprise Bean **のリファレンス定義を追加する場合は，上記** 2. **で表示された**

ツリーで，［リモート EJB への参照］をマウスで右クリックします。
コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。リモート EJB への参照のプロパティページが表示されます。

4. **ほかの Enterprise Bean のリファレンス定義を変更する場合は，上記 2. で表示されたツリーで，［＜リモート EJB への参照名＞］を選択します。**
リモート EJB への参照のプロパティページが表示されます。

5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

● ローカルインタフェースの場合

1. **エディタエリアに表示されている，アプリケーション統合属性（＜ J2EE アプリケーション名＞）の［EJB-JAR］タブを選択します。**
EJB-JAR 属性フォームページが表示されます。

2. EJB-JAR **属性フォームページで，**J2EE **アプリケーションを構成する** Enterprise Bean **の種類に対応する属性を選択します。**
Enterprise Bean の種類に対応する属性を次に示します。
   - ［＜ Session Bean 属性名＞］
   - ［＜ Entity Bean 属性名＞］
   - ［＜ Message-driven Bean 属性名＞］

   選択した属性のツリーが表示されます。

3. **ほかの Enterprise Bean のリファレンス定義を追加する場合は，上記 2. で表示されたツリーで，［ローカル EJB への参照］をマウスで右クリックします。**
コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。ローカル EJB への参照のプロパティページが表示されます。

4. **ほかの Enterprise Bean のリファレンス定義を変更する場合は，上記 2. で表示されたツリーで，［＜ローカル EJB への参照名＞］を選択します。**
ローカル EJB への参照のプロパティページが表示されます。

5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

● リモート EJB への参照の場合

リモート EJB への参照のプロパティページに表示された項目を設定します。リモート EJB への参照のプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(6)　リモート EJB への参照のプロパティページ」を参照してください。

● ローカル EJB への参照の場合

ローカル EJB への参照のプロパティページに表示された項目を設定します。ローカル EJB への参照のプロパティページについては，「21.6.2　Session Bean 属性設定ページ」

の「(8)　ローカル EJB への参照のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティ設定

対応するサーバ管理コマンドでの，ほかの Enterprise Bean のリファレンス定義については，「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。

18.2.2　リソースアダプタのリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean が，リソースアダプタを参照している場合，リファレンスを解決するためのプロパティを設定します。

(1) 手順

リソースアダプタのリファレンス定義を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. **EJB-JAR 属性フォームページで，J2EE アプリケーションを構成する Enterprise Bean の種類に対応する属性を選択します。**
   Enterprise Bean の種類に対応する属性を次に示します。
   - ［< Session Bean 属性名>］
   - ［< Entity Bean 属性名>］
   - ［< Message-driven Bean 属性名>］

   選択した属性のツリーが表示されます。
3. **リソースアダプタのリファレンス定義を追加する場合は，上記 2. で表示されたツリーで，［リソース参照］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。リソース参照のプロパティページが表示されます。
4. **リソースアダプタのリファレンス定義を変更する場合は，上記 2. で表示されたツリーで，［<リソース参照名>］を選択します。**
   ［リソース参照］ページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

［リソース参照］ページに表示された項目を設定します。［リソース参照］ページについては，「21.6.2　Session Bean 属性設定ページ」の「(10)　リソース参照のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティ設定

対応するサーバ管理コマンドでの，リソースアダプタのリファレンス定義については，「9.3.3　リソースアダプタのリファレンス定義」を参照してください。

18.2.3　リソース環境のリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean のリソース環境のリファレンスを解決するためのプロパティを設定します。

(1) 手順

リソース環境のリファレンス定義を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. **EJB-JAR 属性フォームページで，J2EE アプリケーションを構成する Enterprise Bean の種類に対応する属性を選択します。**
   Enterprise Bean の種類に対応する属性を次に示します。
   - ［< Session Bean 属性名>］
   - ［< Entity Bean 属性名>］
   - ［< Message-driven Bean 属性名>］

   選択した属性のツリーが表示されます。
3. **リソース環境のリファレンス定義を追加する場合は，上記 2. で表示されたツリーで，［リソース環境変数］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。［リソース環境変数］のプロパティページが表示されます。
4. **リソース環境のリファレンス定義を変更する場合は，上記 2. で表示されたツリーで，［<リソース環境変数名>］を選択します。**
   リソース環境変数のプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**
6. **リソース環境のリファレンス定義にリンク先キューを追加する場合，上記 2. で表示されたツリーで，［<リソース環境変数名>］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。［リンク先キュー］のプロパティページが表示されます。
7. **リソース環境のリファレンス定義のリンク先キューを変更する場合，上記 2. で表示されたツリーで，［<リンク先キュー名>］を選択します。**
   リンク先キューのプロパティページが表示されます。
8. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

リソース環境変数のプロパティページおよびリンク先キューのプロパティページに表示された項目を設定します。

リソース環境変数のプロパティページについては，「21.6.2 Session Bean 属性設定ページ」の「(12) リソース環境変数のプロパティページ」を参照してください。

リンク先キューのプロパティページについては，「21.6.2 Session Bean 属性設定ページ」の「(13) リンク先キューのプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，リソース環境のリファレンス定義については，「9.3.4 リソース環境のリファレンス定義」を参照してください。

18.3　Message-driven Bean のメッセージ参照定義

Message-driven Bean のメッセージ参照を定義します。

(1) 手順

Message-driven Bean のメッセージ参照定義を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. EJB-JAR **属性フォームページで，［<** Message-driven Bean **属性名>］を選択します。**
   Message-driven Bean 属性ツリーが表示されます。
3. Message-driven Bean **のメッセージ参照定義を，新規に追加する場合は，上記** 2. **で表示されたツリーで，［<** Message-driven Bean **属性名>］を選択して，右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］-［メッセージ参照］を選択します。メッセージ参照のプロパティページが表示されます。
4. Message-driven Bean **のメッセージ参照定義を変更する場合，上記** 2. **で表示されたツリーで，［メッセージ参照］を選択します。**
   メッセージ参照のプロパティページが表示されます。
5. **表示されたプロパティページで，**(2) **のプロパティ項目を設定します。**

(2) プロパティ設定項目

メッセージ参照のプロパティページに表示された項目を設定します。メッセージ参照ページについては，「21.8.2　Message-driven Bean 属性設定ページ」の「(6)　メッセージ参照のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，Message-driven Bean のメッセージ参照定義については，「9.4　Message-driven Bean のメッセージ参照定義」を参照してください。

18.4　トランザクション属性の定義

コンテナのトランザクション属性のトランザクション種別を定義します。

(1) 手順

コンテナのトランザクション種別を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. EJB-JAR **属性フォームページで，**J2EE **アプリケーションを構成する** Enterprise Bean **の種類に対応する属性を選択します。**
   Enterprise Bean の種類に対応する属性を次に示します。
   - ［< Session Bean 属性名>］
   - ［< Entity Bean 属性名>］
   - ［< Message-driven Bean 属性名>］

   選択した属性のツリーが表示されます。
3. **トランザクションコンテナのトランザクション属性を追加する場合は，上記** 2. **で表示されたツリーで，［コンテナー・トランザクション］を右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。コンテナトランザクションのプロパティページが表示されます。
4. **トランザクションコンテナのトランザクション属性を変更する場合は，上記** 2. **で表示されたツリーで，［<コンテナトランザクション名>］を選択します。**
   コンテナトランザクションのプロパティページが表示されます。
5. **表示されたプロパティページで，**(2) **のプロパティ項目を設定します。**

(2) プロパティ設定項目

コンテナトランザクションのプロパティページに表示された項目を設定します。コンテナトランザクションのプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(16)　コンテナトランザクションのプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，トランザクション属性の定義については，「9.5　トランザクション属性の定義」を参照してください。

18.5　サーブレットと JSP のリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）のリファレンス（リソース参照）を定義します。次に示すリファレンスがあります。

- Enterprise Bean リファレンス
  Enterprise Bean 呼び出しのための参照です。
- リソースリファレンス
  データベースへの参照です。リソースアダプタリファレンス，およびリソース環境リファレンスがあります。

18.5.1　Enterprise Bean リファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）が Enterprise Bean を呼び出している場合，リファレンスを解決するためのプロパティを設定します。

(1) 手順

Enterprise Bean への参照定義を設定する手順を次に示します。

参照する Enterprise Bean のアクセスタイプには，次の 2 種類があります。

- リモートインタフェース
- ローカルインタフェース

● リモートインタフェースの場合

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **［WAR］フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **Enterprise Bean のリファレンス定義を追加する場合は，上記 2. で表示されたツリーで，［リモート EJB への参照］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。リモート EJB への参照のプロパティページが表示されます。
4. **Enterprise Bean のリファレンス定義を変更する場合は，上記 2. で表示されたツリーで，［<リモート EJB への参照名>］を選択します。**
   リモート EJB への参照のプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

● **ローカルインタフェースの場合**

1. **エディタエリアに表示されている，アプリケーション統合属性（＜** J2EE **アプリケーション名＞）の［**WAR**］タブを選択します。**
   WAR フォームページが表示されます。
2. **［**WAR**］フォームページで，［＜** WAR **属性名＞］を選択します。**
   WAR 属性ツリーが表示されます。
3. Enterprise Bean **のリファレンス定義を追加する場合は，上記** 2. **で表示されたツリーで，［ローカル** EJB **への参照］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。ローカル EJB への参照のプロパティページが表示されます。
4. Enterprise Bean **のリファレンス定義を変更する場合は，上記** 2. **で表示されたツリーで，［＜ローカル** EJB **への参照名＞］を選択します。**
   ローカル EJB への参照のプロパティページが表示されます。
5. **表示されたプロパティページで，**(2) **のプロパティ項目を設定します。**

(2) プロパティ設定項目

● **リモート** EJB **への参照の場合**

リモート EJB への参照のプロパティページに表示された項目を設定します。リモート EJB への参照のプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(6)　リモート EJB への参照のプロパティページ」を参照してください。

● **ローカル** EJB **への参照の場合**

ローカル EJB への参照のプロパティページに表示された項目を設定します。ローカル EJB への参照のプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(8)　ローカル EJB への参照のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，Enterprise Bean のリファレンス定義については，「9.7.1　Enterprise Bean のリファレンス定義」を参照してください。

18.5.2　リソースアダプタリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）が，リソースアダプタを参照している場合，リファレンスを定義するためのプロパティを設定します。

(1) 手順

リソースアダプタの参照定義を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **リソースアダプタのリファレンス定義を追加する場合は，上記 2. で表示されたツリーで，［リソースの参照］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。リソースの参照のプロパティページが表示されます。
4. **リソースアダプタのリファレンス定義を変更する場合は，上記 2. で表示されたツリーで，［<リソースの参照名>］を選択します。**
   リソースの参照のプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

リソースの参照のプロパティページに表示された項目を設定します。リソースの参照の参照ページについては，「21.6.2　Session Bean 属性設定ページ」の「(10)　リソース参照のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，リソースアダプタのリファレンス定義については，「9.7.3　リソースアダプタのリファレンス定義」を参照してください。

18.5.3　リソース環境リファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）のリソース環境のリファレンスを解決するためのプロパティを設定します。

(1) 手順

リソース環境のリファレンス定義を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **リソース環境のリファレンス定義を追加する場合は，上記 2. で表示されたツリーで，［リソース環境変数］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択しま

す。リソース環境変数のプロパティページが表示されます。

4. **リソース環境のリファレンス定義を変更する場合は，上記 2. で表示されたツリーで，［<リソース環境変数名>］を選択します。**
   リソース環境変数のプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**
6. **リソース環境のリファレンス定義にリンク先キューを追加する場合，上記 2. で表示されたツリーで，［<リソース環境変数名>］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。［リンク先キュー］のプロパティページが表示されます。
7. **リソース環境のリファレンス定義のリンク先キューを変更する場合，上記 2. で表示されたツリーで，［<リンク先キュー名>］を選択します。**
   リンク先キューのプロパティページが表示されます。
8. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

リソース環境変数のプロパティページおよびリンク先キューのプロパティページに表示された項目を設定します。

リソース環境変数のプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(12)　リソース環境変数のプロパティページ」を参照してください。

リンク先キューのプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(13)　リンク先キューのプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，リソースアダプタのリファレンス定義については，「9.7.4　リソース環境のリファレンス定義」を参照してください。

18.6　サーブレットと JSP のマッピング定義

サーブレットおよび JSP のマッピングを定義します。

(1) 手順

サーブレットおよび JSP のマッピング定義を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［<サーブレット属性名>］を選択します。**
   サーブレット属性ツリーが表示されます。
3. **サーブレットおよび JSP のマッピング定義を追加する場合，上記 2. で表示されたツリーで，［URL パターン］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。URL パターンのプロパティページが表示されます。
4. **サーブレットおよび JSP のマッピング定義を削除する場合，上記 2. で表示されたツリーで，［URL パターン］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［削除］を選択します。
5. **サーブレットおよび JSP のマッピング定義を変更する場合，上記 2. で表示されたツリーで，［< URL パターン名>］を選択します。**
   URL パターンのプロパティページが表示されます。
6. **サーブレットおよび JSP のマッピング定義を追加および変更する場合，表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

URL パターンのプロパティページに表示された項目を設定します。URL パターンのプロパティページについては，「21.11.2　サーブレット属性設定ページ」の「(5)　URL パターンのプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，サーブレットと JSP のマッピング定義については，「9.8　サーブレットと JSP のマッピング定義」を参照してください。

18.7　フィルタの設定

フィルタの設定方法について説明します。フィルタを設定するには，フィルタを WAR ファイルに追加して，マッピングを定義します。

18.7.1　フィルタの追加と削除

フィルタの作成および削除は，サーバ管理コマンドで実行します。フィルタの作成については，「9.9.1　フィルタの追加」を参照してください。フィルタの削除については，「9.9.3　フィルタの削除」を参照してください。

18.7.2　フィルタのマッピング

フィルタの追加後，フィルタのマッピングを定義します。

(1) 手順

フィルタのマッピング定義を設定する手順を，次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   ［WAR］フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **フィルタマッピングを追加する場合は，上記 2 で表示されたツリーで，［フィルター・マッピング］をマウスで右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］-［フィルターマッピング (URL パターン)］または［追加］- [フィルターマッピング (サーブレット名)] を選択します。フィルタマッピングのプロパティページが表示されます。
4. **フィルタマッピングを変更する場合は，上記 2. で表示されたツリーで，［<フィルタ名>］を選択します。**
   フィルタのプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

フィルタマッピングのプロパティページに表示された項目を設定します。フィルタマッピングのプロパティページについては，「21.9.2　WAR 属性設定ページ」の「(6)　フィルタマッピング（URL パターン）のプロパティページ」または「(7)　フィルタマッピング（サーブレット名）のプロパティページ」を参照してください。

（3）対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，フィルタのマッピング定義については，「9.9.2　フィルタのマッピング定義」を参照してください。

18.8　Enterprise Bean の実行時属性の定義

次に示す Enterprise Bean の実行時属性を定義します。

- Stateful Session Bean
- Stateless Session Bean
- Entity Bean
- Message-driven Bean

18.8.1　Stateful Session Bean の実行時プロパティの設定

アプリケーションを構成する個々の Stateful Session Bean に対して，アプリケーション実行時のプロパティを設定します。

(1) 手順

Stateful Session Bean の実行時のプロパティを設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. **EJB-JAR 属性フォームページで，［< Session Bean 属性名>］を選択します。**
   Session Bean 属性ツリーが表示されます。
3. **上記 2. で表示されたツリーで，［ランタイム］を選択します。**
   ランタイムのプロパティページが表示されます。
4. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**
5. **上記 2. で表示されたツリーで，［ランタイム］下の［ステートフル・セッション・ビーン］を選択します。**
   Stateful Session Bean のプロパティページが表示されます。
6. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

ランタイムのプロパティページおよび Stateful Session Bean のプロパティページに表示された項目を設定します。

ランタイムのプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(21)　ランタイムのプロパティページ」を参照してください。

Stateful Session Bean のプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(23)　Stateful Session Bean のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの Stateful Session Bean の実行時プロパティの設定については,「9.10.1　Stateful Session Bean の実行時プロパティの設定」を参照してください。

18.8.2　Stateless Session Bean の実行時プロパティの設定

アプリケーションを構成する個々の Stateless Session Bean に対して，アプリケーション実行時のプロパティを設定します。

(1) 手順

Stateless Session Bean の実行時のプロパティを設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. **EJB-JAR 属性フォームページで，［< Session Bean 属性名>］を選択します。**
   Session Bean 属性ツリーが表示されます。
3. **上記 2. で表示されたツリーで，［ランタイム］を選択します。**
   ランタイムのプロパティページが表示されます。
4. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**
5. **上記 2. で表示されたツリーで，［ランタイム］下の［ステートレス・セッション・ビーン］を選択します。**
   Stateless Session Bean のプロパティページが表示されます。
6. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

ランタイムのプロパティページおよび Stateless Session Bean のプロパティページに表示された項目を設定します。

ランタイムのプロパティページについては,「21.6.2　Session Bean 属性設定ページ」の「(21)　ランタイムのプロパティページ」を参照してください。

Stateless Session Bean のプロパティページについては,「21.6.2　Session Bean 属性設定ページ」の「(22)　Stateless Session Bean のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの Stateless Session Bean の実行時プロパティの設定に

ついては，「9.10.2　Stateless Session Bean の実行時プロパティの設定」を参照してください。

18.8.3　Entity Bean の実行時プロパティの設定

アプリケーションを構成する個々の Entity Bean に対して，アプリケーション実行時のプロパティを設定します。

(1) 手順

Entity Bean の実行時のプロパティを設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. **EJB-JAR 属性フォームページで，［< Entity Bean 属性名>］を選択します。**
   Entity Bean 属性ツリーが表示されます。
3. **上記 2. で表示されたツリーで，［ランタイム］を選択します。**
   ランタイムのプロパティページが表示されます。
4. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

ランタイムのプロパティページに表示された項目を設定します。ランタイムのプロパティページについては，「21.7.2　Entity Bean 属性設定ページ」の「(2)　ランタイムのプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，Entity Bean の実行時プロパティの設定については，「9.10.3　Entity Bean の実行時プロパティの設定」を参照してください。

18.8.4　Message-driven Bean の実行時プロパティの設定

アプリケーションを構成する個々の Message-driven Bean に対して，アプリケーション実行時のプロパティを設定します。

(1) 手順

Message-Driven Bean の実行時のプロパティを設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。

2. EJB-JAR **属性フォームページで，［＜** Message-driven Bean **属性名＞］を選択します。**
   Message-driven Bean 属性ツリーが表示されます。
3. **上記** 2. **で表示されたツリーで，［ランタイム］を選択します。**
   ランタイムのプロパティページが表示されます。
4. **表示されたプロパティページで，**(2) **のプロパティ項目を設定します。**

(2) プロパティ設定項目

ランタイムのプロパティページに表示された項目を設定します。ランタイムのプロパティページについては，「21.8.2　Message-driven Bean 属性設定ページ」の「(7)　ランタイムのプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，Message-driven Bean の実行時プロパティの設定については，「9.10.4　Message-driven Bean の実行時プロパティの設定」を参照してください。

18.9　サーブレットと JSP の実行時属性の定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）の実行時属性を定義します。次に示す定義があります。

- J2EE アプリケーションのコンテキストルートの定義
- Web アプリケーション単位での同時実行スレッド数制御の定義
- URL グループ単位での同時実行スレッド数制御の定義
- URL グループ単位の実行待ちリクエストの監視の定義

それぞれの定義で設定するプロパティ項目を次に示します。

18.9.1　J2EE アプリケーションのコンテキストルート定義

(1) 手順

J2EE アプリケーションのコンテキストルート定義を設定します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **上記 2. で表示されたツリーで，［< WAR 属性名>］を選択します。**
   WAR 属性のプロパティページが表示されます。
4. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

WAR 属性のプロパティページに表示された項目を設定します。WAR 属性のプロパティページについては，「21.9.2　WAR 属性設定ページ」の「(1)　WAR のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの J2EE アプリケーションのコンテキストルート定義については，「9.11.1　J2EE アプリケーションのコンテキストルート定義」を参照してください。

18.9.2　Web アプリケーション単位での同時実行スレッド数制御の定義

Web コンテナで同時実行スレッド数を制御するための定義をします。

なお，Web アプリケーション単位での同時実行スレッド数制御を有効にするためには，J2EE サーバ用の usrconf.properties ファイルの webserver.container.thread_control.enabled キーに「true」を指定してください。

(1) 手順

Web コンテナで同時実行スレッド数を制御するための定義を，設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **スレッド数制御情報を追加する場合は，上記 2. で表示されたツリーで，［< WAR 属性名>］を右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。スレッド数制御のプロパティページが表示されます。
4. **スレッド数制御情報を変更する場合は，上記 2. で表示されたツリーで，［スレッド数制御］を選択します。**
   スレッド数制御のプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

スレッド数制御のプロパティページに表示された項目を設定します。スレッド数制御のプロパティページについては，「21.9.2　WAR 属性設定ページ」の「(55)　スレッド数制御のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，Web アプリケーション単位での同時実行スレッド数制御の定義については，「9.11.2　Web アプリケーション単位での同時実行スレッド数制御の定義」を参照してください。

18.9.3　URL グループ単位での同時実行スレッド数制御の定義

URL グループ単位での同時実行スレッド数を制御するための定義をします。

URL グループ単位での同時実行スレッド数制御を有効にするためは，Web アプリケーション単位での同時実行スレッド数制御の定義を設定してください。Web アプリケーション単位での同時実行スレッド数制御については，「18.9.2　Web アプリケーション単位での同時実行スレッド数制御の定義」を参照してください。

(1) 手順

URL グループ単位での同時実行スレッド数を制御するための設定手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **URL グループ単位での同時実行スレッド数を制御する情報を，スレッド情報に新規に追加する場合は，上記 2. で表示されたツリーで，［スレッド数制御］を右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。URL グループのスレッド数制御のプロパティページが表示されます。
4. **すでにあるスレッド情報に，URL グループ単位での同時実行スレッド数を制御する情報を追加する場合は，上記 2. で表示されたツリーで，［URL グループのスレッド数制御］を右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。URL グループのスレッド数制御のプロパティページが表示されます。
5. **URL グループ単位での同時実行スレッド数を制御する情報を変更する場合は，上記 2. で表示されたツリーで，［< URL グループのスレッド数制御定義名>］を選択します。**
   URL グループのスレッド数制御のプロパティページが表示されます。
6. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

URL グループのスレッド数制御のプロパティページに表示された項目を設定します。
URL グループのスレッド数制御のプロパティページについては，「21.9.2　WAR 属性設定ページ」の「(57)　URL グループのスレッド数制御のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの URL グループ単位での同時実行スレッド数制御の定義については,「9.11.3　URL グループ単位での同時実行スレッド数制御の定義」を参照してください。

18.9.4　URL グループ単位の実行待ちリクエスト数の監視の定義

URL グループ単位の実行待ちリクエスト数を監視するための定義をします。

(1) 手順

URL グループ単位の実行待ちリクエスト数を監視するための定義を，設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **［< URL グループのスレッド数制御定義名>］，［状態モニター］の順で選択して，展開されたツリーで，［実行待ちリクエスト数の監視］を選択します。**
   実行待ちリクエスト数の監視のプロパティページが表示されます。
4. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

実行待ちリクエスト数の監視のプロパティページに表示された項目を設定します。［実行待ちリクエスト数の監視］ページについては,「21.9.2　WAR 属性設定ページ」の「(60)　実行待ちリクエスト数の監視のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，URL グループ単位の実行待ちリクエスト数の監視の定義については,「9.11.4　URL グループ単位の実行待ちリクエスト数の監視の定義」を参照してください。

18.10　デフォルトの文字エンコーディングの設定

Web アプリケーション単位にデフォルトの文字エンコーディングを設定します。

次の文字エンコーディングについて，デフォルトのエンコーディングが設定できます。

- リクエストボディおよびクエリの文字エンコーディング
- レスポンスボディの文字エンコーディング
- JSP ファイルの文字エンコーディング

(1) 手順

デフォルトの文字エンコーディングを設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR フォームページが表示されます。
2. **WAR フォームページで，［< WAR 属性名>］を選択します。**
   WAR 属性ツリーが表示されます。
3. **デフォルトの文字エンコーディングを追加する場合は，上記 2. で表示されたツリーで，［< WAR 属性名>］を右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューの［追加］で，次の文字エンコーディングのどれかを選択します。
   - リクエストボディおよびクエリの文字エンコーディング：［HTTP リクエスト］
   - レスポンスボディの文字エンコーディング：［HTTP レスポンス］
   - JSP ファイルの文字エンコーディング：［JSP］

   選択した文字エンコーディングのプロパティページが表示されます。
4. **デフォルトの文字エンコーディングを変更する場合は，上記 2. で表示されたツリーで，次のどれかを選択します。**
   - リクエストボディおよびクエリの文字エンコーディング：［HTTP リクエスト］
   - レスポンスボディの文字エンコーディング：［HTTP レスポンス］
   - JSP ファイルの文字エンコーディング：［JSP］

   選択した文字エンコーディングのプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

選択した文字エンコーディングのプロパティページに表示された項目を設定します。

選択した文字エンコーディングのプロパティページについては，「21.9.2　WAR 属性設

定ページ」の，次のページを参照してください。

- リクエストボディおよびクエリの文字エンコーディング
  「(61) HTTP リクエストのプロパティページ」
- レスポンスボディの文字エンコーディング
  「(62) HTTP レスポンスのプロパティページ」
- JSP ファイルの文字エンコーディング
  「(63) JSP のプロパティページ」

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，デフォルトの文字エンコーディングについては，「9.12 デフォルトの文字エンコーディングの設定」を参照してください。

18.11　CTM のスケジューリング

CTM を利用する場合の，スケジューリングについて設定します。CTM は，構成ソフトウェアに Cosminexus Component Transaction Monitor を含む製品だけで利用できます。利用できる製品については，マニュアル「Cosminexus アプリケーションサーバ 概説」の「2.3.1　製品と構成ソフトウェアの対応」を参照してください。

次の手順で，CTM のスケジューリングを設定します。

- アプリケーション単位のスケジューリングを設定します。
- スケジューリングの対象にしたい Stateless Session Bean のスケジューリングを設定します。

18.11.1　J2EE アプリケーション単位のスケジューリング

J2EE アプリケーション単位の CTM との連携にかかわる設定をします。

(1) 手順

J2EE アプリケーション単位の CTM との連携にかかわる設定の手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［アプリケーション属性］タブを選択します。**
   アプリケーション属性フォームページに，アプリケーション属性のツリーが表示されます。
2. **アプリケーション属性に CTM 連携を新規に設定する場合，<アプリケーション属性名>を右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。CTM 連携のプロパティページが表示されます。
3. **すでに設定した CTM 連携情報を変更する場合，上記 1. で表示されたツリーで［CTM 連携］を選択します。**
   CTM 連携のプロパティページが表示されます。
4. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

CTM 連携のプロパティページに表示された項目を設定します。CTM 連携のプロパティページについては，「21.4.2　アプリケーション属性設定ページ」の「(4)　CTM 連携のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，J2EE アプリケーション単位のスケジューリングにつ

いては，「9.14.1　J2EE アプリケーション単位のスケジューリング」を参照してください。

18.11.2　Stateless Session Bean 単位のスケジューリング

CTM によるスケジューリングの対象に設定した J2EE アプリケーションに含まれている Stateless Session Bean を CTM によるスケジューリングの対象にする場合，Stateless Session Bean 単位のスケジューリングの設定をします。

(1) 手順

Stateless Session Bean 単位の CTM との連携にかかわる設定の手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. **EJB-JAR 属性フォームページで，［< Session Bean 属性名>］を選択します。**
   Session Bean 属性ツリーが表示されます。
3. **ランタイム属性に CTM 連携情報を新規に設定する場合，［ランタイム］を右クリックします。**
   コンテキストメニューが表示されます。コンテキストメニューで［追加］を選択します。CTM 連携のプロパティページが表示されます。
4. **すでに設定した CTM 連携情報を変更する場合，上記 2. で表示されたツリーで，［ランタイム］，［CTM 連携］の順で選択します。**
   CTM 連携のプロパティページが表示されます。
5. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

CTM 連携のプロパティページに表示された項目を設定します。CTM 連携のプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(24)　CTM 連携のプロパティページ」を参照してください。

また，Stateless Session Bean 単位で CTM を利用するかどうかの設定は，実行時のプロパティで設定します。Stateless Session Bean の実行時プロパティの設定については，「18.8.2　Stateless Session Bean の実行時プロパティの設定」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，Stateless Session Bean 単位のスケジューリングについては，「9.14.2　Stateless Session Bean 単位のスケジューリング」を参照してください。

18.12　起動順序の設定

J2EE アプリケーションの起動順序，および J2EE アプリケーションに含まれる Enterprise Bean および WAR の起動順序を設定します。

18.12.1　J2EE アプリケーションの起動順序の設定

J2EE アプリケーションの起動順序を設定します。

(1) 手順

J2EE アプリケーションの起動順序を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［アプリケーション属性］タブを選択します。**
   アプリケーション属性フォームページが表示されます。
2. **アプリケーション属性フォームページで，［<アプリケーション属性名>］を選択します。**
   アプリケーション属性のプロパティページが表示されます。
3. **表示されたプロパティページで，**(2) **のプロパティ項目を設定します。**

(2) プロパティ設定項目

アプリケーション属性ページに表示された項目を設定します。アプリケーション属性ページについては，「21.4.2　アプリケーション属性設定ページ」の「(1)　アプリケーションのプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，J2EE アプリケーションの起動順序の設定については，「9.15.1　J2EE アプリケーションの起動順序の設定」を参照してください。

18.12.2　Enterprise Bean の起動順序の設定

Enterprise Bean の起動順序を設定します。

(1) 手順

Enterprise Bean の起動順序を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［EJB-JAR］タブを選択します。**
   EJB-JAR 属性フォームページが表示されます。
2. EJB-JAR **属性フォームページで，**J2EE **アプリケーションを構成する** Enterprise Bean

の種類に対応する属性を選択します。
Enterprise Bean の種類に対応する属性を次に示します。
- ［＜ Session Bean 属性名＞］
- ［＜ Entity Bean 属性名＞］
- ［＜ Message-driven Bean 属性名＞］

選択した Enterprise Bean の属性のツリーが表示されます。

3. **上記 2. で表示されたツリーで，Enterprise Bean 名を選択します。**
Enterprise Bean のプロパティページが表示されます。
4. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

Enterprise Bean プロパティページに表示された項目を設定します。

Enterprise Bean の種類によって，次の Enterprise Bean のプロパティページを参照してください。

- Session Bean の場合
「21.6.2　Session Bean 属性設定ページ」の「(1)　Session Bean のプロパティページ」
- Entity Bean の場合
「21.7.2　Entity Bean 属性設定ページ」の「(1)　Entity Bean のプロパティページ」
- Message-driven Bean の場合
「21.8.2　Message-driven Bean 属性設定ページ」の「(1)　Message-driven Bean のプロパティページ」

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，Enterprise Bean の起動順序の設定については，「9.15.2　Enterprise Bean の起動順序の設定」を参照してください。

18.12.3　サーブレットと JSP の起動順序の設定

サーブレット，JSP が含まれる WAR の起動順序を設定します。

(1) 手順

サーブレット，JSP が含まれる WAR の起動順序を設定する手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（＜ J2EE アプリケーション名＞）の［WAR］タブを選択します。**
WAR 属性フォームページが表示されます。
2. **WAR 属性フォームページで表示されたツリーで，［＜ WAR 属性名＞］を選択します。**

WAR のプロパティページが表示されます。

3. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

(2) プロパティ設定項目

WAR のプロパティページに表示された項目を設定します。WAR のプロパティページについては，「21.9.2　WAR 属性設定ページ」の「(1)　WAR のプロパティページ」を参照してください。

(3) 対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，サーブレットと JSP の起動順序の設定については，「9.15.3　サーブレットと JSP の起動順序の設定」を参照してください。

18.13　サーブレットと JSP のエラー通知の設定

次のタイミングでエラーが発生した場合に，エラーを通知して J2EE アプリケーションの開始処理を中止するかどうかを指定します。

- J2EE アプリケーションの開始時にロードするように設定されている（<load-on-startup> が設定されている）サーブレットまたは JSP の初期化処理中にエラーが発生した場合
- タグライブラリ解析時にエラーが発生した場合

（1）手順

サーブレット，JSP が含まれる WAR のエラー通知の設定手順を次に示します。

1. **エディタエリアに表示されている，アプリケーション統合属性（< J2EE アプリケーション名>）の［WAR］タブを選択します。**
   WAR 属性フォームページが表示されます。
2. **WAR 属性フォームページで表示されたツリーで，［< WAR 属性名>］を選択します。**
   WAR のプロパティページが表示されます。
3. **表示されたプロパティページで，(2) のプロパティ項目を設定します。**

（2）プロパティ設定項目

WAR のプロパティページに表示された項目を設定します。WAR のプロパティページについては，「21.9.2　WAR 属性設定ページ」の「(1)　WAR のプロパティページ」を参照してください。

（3）対応するサーバ管理コマンドでのプロパティの設定

対応するサーバ管理コマンドでの，サーブレットと JSP のエラー通知の設定については，「9.16　サーブレットと JSP のエラー通知の設定」を参照してください。

18.14　そのほかのプロパティの設定

J2EE アプリケーションの作成およびカスタマイズで設定する，そのほかのプロパティ項目については，「21.3　J2EE アプリケーションの属性編集画面」を参照してください。

19 J2EE アプリケーションの実行

この章では，Server Plug-in を使用した J2EE アプリケーションの実行について説明します。

19.1　J2EE アプリケーションの実行の概要

J2EE アプリケーションとして必要なプロパティや動作の定義が終わったら，J2EE アプリケーションを実行できます。

J2EE アプリケーションの形式と J2EE アプリケーションの実行の概要について次に示します。

表 19-1　J2EE アプリケーションの実行の概要

設定項目	内容	J2EE アプリケーションの形式		参照先
		アーカイブ形式	展開ディレクトリ形式	
J2EE アプリケーションの開始	J2EE アプリケーションを開始します。	○	○	19.2
J2EE アプリケーションの停止	J2EE アプリケーションを，通常停止または強制停止します。	○	○	19.3
J2EE アプリケーションの削除	J2EE アプリケーションを削除します。	○	○	19.4
J2EE アプリケーションの入れ替え	J2EE サーバ上の，次の J2EE アプリケーションを入れ替えます。 • アーカイブ形式のアプリケーション • 展開ディレクトリ形式のアプリケーション	△※	△※	19.5
J2EE アプリケーション名の変更	J2EE アプリケーションの名前を変更します。	○	○	19.6
RMI-IIOP スタブとインタフェースの取得	J2EE アプリケーションの RMI-IIOP スタブおよびインタフェースを取得します。	○	○	19.7

（凡例）　○：実行できる　△：実行条件がある
注※　J2EE アプリケーションの入れ替えは，入れ替え先の J2EE アプリケーションの形式と同じ形式の J2EE アプリケーションを指定します。

19.2　J2EE アプリケーションの開始

J2EE アプリケーションを開始します。

(1) 手順

J2EE アプリケーションを開始する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，開始する［< J2EE アプリケーション名 >］を選択します。**
2. **次のどれかの方法で，J2EE アプリケーションの開始を実行します。**
   - ビューツールバーの［ ］（開始）をクリックします。
   - ビュープルダウンメニューの［ 開始］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ 開始］を選択します。

   J2EE アプリケーションが開始されます。

(2) 画面表示

J2EE アプリケーションの開始で使用する画面については，「21.1.4　開始」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjstartapp コマンドです。

注意事項および J2EE アプリケーションを開始する場合のサーバ管理コマンドについては，「10.2.1　J2EE アプリケーションの開始」の「(1)　J2EE アプリケーションの通常開始」を参照してください。

19.3　J2EE アプリケーションの停止

J2EE アプリケーションを停止します。

J2EE アプリケーションの停止には，次の二つの方法があります。

- J2EE アプリケーションを通常停止させる方法
- J2EE アプリケーションを強制停止させる方法

19.3.1　J2EE アプリケーションの通常停止

J2EE アプリケーションを通常停止します。

(1) 手順

J2EE アプリケーションを通常停止する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，停止する［< J2EE アプリケーション名 >］を選択します。**
2. **次のどれかの方法で，J2EE アプリケーションの通常停止を実行します。**
   - ビューツールバーの［ ］（停止）をクリックします。
   - ビュープルダウンメニューの［ 停止］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ 停止］を選択します。

   J2EE アプリケーションが停止されます。

(2) 画面表示

J2EE アプリケーションの通常停止で使用する画面については，「21.1.7　停止」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjstopapp コマンドです。

注意事項および J2EE アプリケーションを停止する場合のサーバ管理コマンドについては，「10.2.2　J2EE アプリケーションの停止」の「(1)　J2EE アプリケーションの通常停止」を参照してください。

19.3.2　J2EE アプリケーションの強制停止

J2EE アプリケーションを強制停止します。

(1) 手順

J2EE アプリケーションを強制停止する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，強制停止する［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，J2EE アプリケーションの強制停止を実行します。**
   - ビュープルダウンメニューの［強制停止］を選択します。
   - 右クリックで表示されるコンテキストメニューの［強制停止］を選択します。

   J2EE アプリケーションが強制停止されます。

(2) 画面表示

J2EE アプリケーションの強制停止で使用する画面については，「21.1.8　強制停止」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，-force オプションを指定した cjstopapp コマンドです。

注意事項および J2EE アプリケーションを強制停止する場合のサーバ管理コマンドについては，「10.2.2　J2EE アプリケーションの停止」の「(2)　J2EE アプリケーションの強制停止」を参照してください。

19.4　J2EE アプリケーションの削除

J2EE アプリケーションを削除します。

(1) 手順

J2EE アプリケーションを削除する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，削除する［< J2EE アプリケーション名 >］を選択します。**
2. **次のどちらかの方法で，J2EE アプリケーションの削除を実行します。**
   - ビュープルダウンメニューの［ 削除］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ 削除］を選択します。

   J2EE アプリケーションの削除が実行されると，次のメッセージダイアログが表示されます。

```
J2EEアプリケーション‘ ＜J2EEアプリケーション名＞ ’を削除しますか？
```

3. **J2EE アプリケーションを削除する場合は，メッセージダイアログで［はい］をクリックします。**
   J2EE アプリケーションが削除されます。［いいえ］をクリックすると，J2EE アプリケーションの削除は中止されます。

(2) 画面表示

J2EE アプリケーションの削除で使用する画面については，「21.1.9　削除」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjdeleteapp コマンドです。

注意事項および J2EE アプリケーションを削除するサーバ管理コマンドについては，「10.4　J2EE アプリケーションの削除」を参照してください。

19.5　J2EE アプリケーションの入れ替え

J2EE アプリケーションを入れ替えます。

J2EE アプリケーションの入れ替えには，次の二つの場合があります。

- アーカイブ形式でインポートされた J2EE アプリケーションを入れ替える場合
- 展開ディレクトリ形式の J2EE アプリケーションを再ロードする場合

入れ替える J2EE アプリケーションの形式および注意事項については，サーバ管理コマンドによる J2EE アプリケーションの入れ替え（「10.6　J2EE アプリケーションの入れ替え」）を参照してください。

19.5.1　アーカイブ形式の J2EE アプリケーション

アーカイブ形式の J2EE アプリケーションを入れ替えます。

(1) 手順

アーカイブ形式でインポートされた J2EE アプリケーションを入れ替える手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，入れ替える［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，アーカイブ形式の J2EE アプリケーションを入れ替えます。**
   - ビュープルダウンメニューの［入れ替え］を選択します。
   - 右クリックで表示されるコンテキストメニューの［入れ替え］を選択します。

   J2EE アプリケーションの入れ替えが実行されると，［J2EE アプリケーションの入れ替え］ダイアログが表示されます。
3. **入れ替えるアーカイブ形式の J2EE アプリケーションを指定して，［開く］を実行します。**
   アーカイブ形式の J2EE アプリケーションの入れ替えが開始されます。

(2) 画面表示

アーカイブ形式の J2EE アプリケーションの入れ替えで使用する画面については，「21.1.11　J2EE アプリケーションの入れ替え」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjreplaceapp コマンドです。

アーカイブ形式の J2EE アプリケーションの入れ替えで使用するサーバ管理コマンド，および cosminexus.xml を含むアプリケーションの入れ替えの注意事項については，

「10.6.1　アーカイブ形式のアプリケーション」を参照してください。

19.5.2　展開ディレクトリ形式の J2EE アプリケーション

展開ディレクトリ形式の J2EE アプリケーションをリロードします。リロードする展開ディレクトリ形式の J2EE アプリケーションが開始状態の場合に操作できます。

(1) 手順

展開ディレクトリ形式でインポートされた J2EE アプリケーションをリロードする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，リロードする［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，展開ディレクトリ形式の J2EE アプリケーションをリロードします。**
   - ビュープルダウンメニューの［ リロード］を選択します。
   - 右クリックで表示されるコンテキストメニューの［ リロード］を選択します。

   展開ディレクトリ形式の J2EE アプリケーションがリロードされます。

(2) 画面表示

展開ディレクトリ形式の J2EE アプリケーションのリロードで使用する画面については，「21.1.10　リロード」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjreloadapp コマンドです。

展開ディレクトリ形式の J2EE アプリケーションをリロードするサーバ管理コマンド，および cosminexus.xml を含むアプリケーションの入れ替えの注意事項については，「10.6.2　展開ディレクトリ形式のアプリケーション」を参照してください。

19.6　J2EE アプリケーション名の変更

J2EE アプリケーションのアプリケーション名を変更します。

(1) 手順

J2EE アプリケーションのアプリケーション名を変更する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，アプリケーション名を変更する［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，J2EE アプリケーションのアプリケーション名を変更します。**
   - ビュープルダウンメニューの［アプリケーション名の変更］を選択します。
   - 右クリックで表示されるコンテキストメニューの［アプリケーション名の変更］を選択します。

   ［アプリケーション名の変更］ダイアログが表示されます。
3. **［新しい名前］に新しいアプリケーション名を入力し，［OK］をクリックします。**
   アプリケーション名を変更しない場合は，［キャンセル］をクリックしてください。

(2) 画面表示

J2EE アプリケーション名の変更で使用する画面については，「21.1.12　J2EE アプリケーション名の変更」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjrenameapp コマンドです。

注意事項および J2EE アプリケーション名を変更するサーバ管理コマンドについては，「10.7　J2EE アプリケーション名の変更」を参照してください。

19.7　RMI-IIOP スタブとインタフェースの取得

次の手順で，J2EE アプリケーションから RMI-IIOP インタフェースをエクスポートします。

- ［Cosminexus RMI-IIOP インターフェース］ページを表示します。
- RMI-IIOP インタフェースのエクスポートを実行します。

19.7.1　RMI-IIOP インタフェースページの表示

［Cosminexus RMI-IIOP インターフェース］ページを表示します。［Cosminexus RMI-IIOP インターフェース］ページは，J2EE アプリケーションの RMI-IIOP インタフェースをエクスポートするためのダイアログです。

Management Server リモート管理機能に未接続の状態でも，［Cosminexus RMI-IIOP インタフェース］ページは表示されます。Management Server リモート管理機能に未接続の場合，［Cosminexus RMI-IIOP インターフェース］ページが表示される前に，［Cosminexus Management Server リモート管理機能へログイン］ダイアログが表示されます。［Cosminexus Management Server リモート管理機能へログイン］ダイアログで，Management Server リモート管理機能に接続してください。Management Server リモート管理機能への接続については，「11.1.2　Management Server リモート管理機能への接続」を参照してください。

(1) 手順

次のどちらかの方法で，［Cosminexus RMI-IIOP インターフェース］ページを表示します。

- ［エクスポート］ダイアログから表示
- ［サーバー・エクスプローラー］ビューから表示

● ［エクスポート］ダイアログから表示

［エクスポート］ダイアログから［Cosminexus RMI-IIOP インターフェース］ページを表示する手順を次に示します。

1. **メニューから［ファイル］-［エクスポート］を選択します。**
   ［エクスポート］ダイアログが表示されます。
2. **［エクスポート先の選択］の［Cosminexus RMI-IIOP インターフェース］を選択し，［次へ］をクリックします。**
   ［Cosminexus RMI-IIOP インターフェース］ページが表示されます。

● ［サーバー・エクスプローラー］ビューから表示

［サーバー・エクスプローラー］ビューから［Cosminexus RMI-IIOP インターフェース］

ページを表示する手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，RMI-IIOP インタフェースをエクスポートする［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，［Cosminexus RMI-IIOP インターフェース］ページを表示します。**
   - ビューツールバーの［  ］（エクスポート）をクリックします。表示されるサブメニューの［  Cosminexus RMI-IIOP インターフェース］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  Cosminexus RMI-IIOP インターフェースのエクスポート］を選択します。

   ［Cosminexus RMI-IIOP インターフェース］ページが表示されます。

注意事項

［サーバー・エクスプローラー］ビューで，［< J2EE アプリケーション名>］を選択しないで，ビューツールバーの［  ］（エクスポート）をクリックして表示されるサブメニューの［  Cosminexus RMI-IIOP インターフェース］を選択しても，［Cosminexus RMI-IIOP インタフェース］ページは表示されます。

（2） 画面表示

［Cosminexus RMI-IIOP インターフェース］ページの表示で使用する画面については，「21.1.24　Cosminexus RMI-IIOP インタフェースのエクスポート」の「(1) ［Cosminexus RMI-IIOP インタフェース］ページの表示」を参照してください。

（3） 対応するサーバ管理コマンド

［Cosminexus RMI-IIOP インターフェース］ページを表示する操作のため，対応するサーバ管理コマンドはありません。

19.7.2　RMI-IIOP インタフェースのエクスポートの実行

［Cosminexus RMI-IIOP インターフェース］ページで，RMI-IIOP インタフェースのエクスポートを実行します。

（1） 手順

RMI-IIOP インタフェースをエクスポートする手順を次に示します。

1. **［エクスポート元アプリケーション］を選択します。**
   ［サーバー・エクスプローラー］ビューのノードが表示されます。J2EE アプリケーションが表示されていない場合，J2EE サーバのノードを展開して J2EE アプリケーションを表示してください。
   RMI-IIOP インタフェースをエクスポートする J2EE アプリケーションを選択しま

す。

2. **［エクスポート先］-［プロジェクト］または［エクスポート先］-［フォルダー］を指定します。**
RMI-IIOP インタフェースのエクスポート先のプロジェクトまたはフォルダを入力します。
［参照］をクリックすると，［プロジェクトの参照］ダイアログまたは［フォルダの参照］ダイアログが表示されます。［プロジェクトの参照］ダイアログまたは［フォルダの参照］ダイアログで，RMI-IIOP インタフェースのエクスポート先を選択してください。

注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

3. **［終了］をクリックしてください。**
［Cosminexus RMI-IIOP インターフェース］ページが閉じ，［エクスポート先］で指定したプロジェクトまたはフォルダに，RMI-IIOP インタフェースをエクスポートします。

(2) 画面表示

RMI-IIOP スタブとインタフェースの取得で使用する画面については，「21.1.24 Cosminexus RMI-IIOP インタフェースのエクスポート」の「(2) ［Cosminexus RMI-IIOP インタフェース］ページの操作」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjgetstubsjar コマンドです。

注意事項および RMI-IIOP スタブとインタフェースを取得するサーバ管理コマンドについては，「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。

20 アプリケーション統合属性のインポートとエクスポート

この章では，Server Plug-in を使用したアプリケーション統合属性のインポートとエクスポートについて説明します。

20.1　アプリケーション統合属性のインポートとエクスポートの概要

Server Plug-in では，J2EE アプリケーションのプロパティを設定する方法として，次の二つの方法を提供しています。

- 属性ファイル編集エディタのフォームページでプロパティを設定します。属性ファイル編集エディタは，Server Plug-in のプロパティ操作で起動されます。
  Server Plug-in のプロパティ操作については，「11.3　プロパティの設定」を参照してください。
- アプリケーション統合属性をアプリケーション統合属性ファイルに取得（エクスポート）し，プロパティ設定項目を編集します。編集したアプリケーション統合属性のファイルの値を，J2EE アプリケーションの属性に反映（インポート）します。

この章で，アプリケーション属性の取得（エクスポート）とアプリケーション属性の設定（インポート）の操作について説明します。

プロパティの設定項目については，「18.　J2EE アプリケーションのプロパティ設定」を参照してください。

20.2　アプリケーション統合属性のインポート

J2EE アプリケーションの属性を，アプリケーション統合属性ファイルの値に変更します。

(1) 手順

アプリケーション統合属性をインポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，アプリケーション属性ファイルの値に J2EE アプリケーションの属性を変更する［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，アプリケーション統合属性をインポートします。**
   - ビューツールバーの［  ］（インポート）をクリックして，表示されるサブメニューの［アプリケーション統合属性］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  アプリケーション統合属性のインポート］を選択します。

   アプリケーション統合属性のインポートが実行されると，［アプリケーション統合属性のインポート］ダイアログが表示されます。
3. **インポートするアプリケーション統合属性ファイルを指定して，［開く］を実行します。**
   インポートが開始されます。

(2) 画面表示

アプリケーション統合属性のインポートで使用する画面については，「21.1.15　アプリケーション統合属性のインポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjsetappprop コマンドに対応します。

アプリケーション統合属性をインポートする場合の，サーバ管理コマンドについては，「3.3　属性ファイルによるプロパティの設定」を参照してください。

20.3　アプリケーション統合属性のエクスポート

J2EE アプリケーションの属性をアプリケーション統合属性ファイルに取得します。

注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

(1) 手順

アプリケーション統合属性をエクスポートする手順を次に示します。

1. **［サーバー・エクスプローラー］ビューで，アプリケーション統合属性をエクスポートする［< J2EE アプリケーション名>］を選択します。**
2. **次のどちらかの方法で，アプリケーション統合属性をエクスポートします。**
   - ビューツールバーの［  ］（エクスポート）をクリックします。表示されるサブメニューの［  アプリケーション統合属性］を選択します。
   - 右クリックで表示されるコンテキストメニューの［  アプリケーション統合属性のエクスポート］を選択します。

   エクスポートが実行されると，［アプリケーション統合属性のエクスポート］ダイアログが表示されます。
3. **エクスポートするアプリケーション統合属性ファイルを指定して，［保存］を実行します。**
   エクスポートが開始されます。

(2) 画面表示

アプリケーション統合属性のエクスポートで使用する画面については，「21.1.21　アプリケーション統合属性のエクスポート」を参照してください。

(3) 対応するサーバ管理コマンド

対応するサーバ管理コマンドは，cjgetappprop コマンドに対応します。

アプリケーション統合属性をエクスポートする場合の，サーバ管理コマンドについては，「3.3　属性ファイルによるプロパティの設定」を参照してください。

21 Server Plug-in で使用する画面

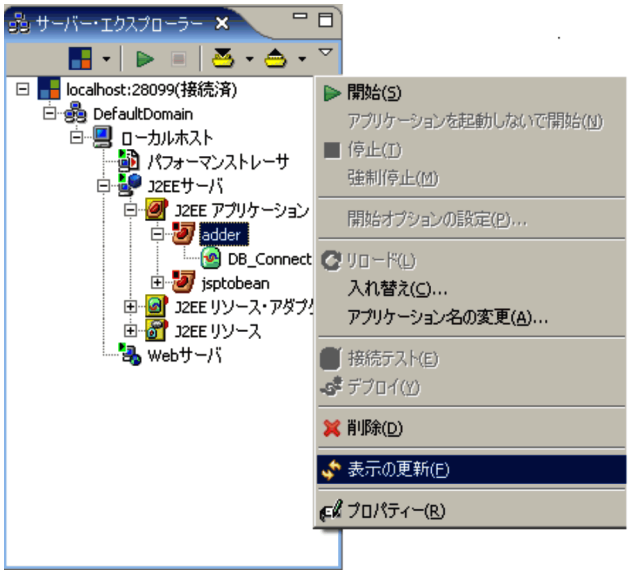
この章では，Server Plug-in の操作や属性の編集など，Server Plug-in で使用する画面について説明します。

21.1　Server Plug-in の操作で使用する画面

Server Plug-in では，サーバを起動・停止したり，J2EE アプリケーションや J2EE リソースアダプタの設定を行ったりするために，［サーバー・エクスプローラー］ビューを提供しています。［サーバー・エクスプローラー］ビューを使用して，サーバの起動・停止，J2EE アプリケーションや J2EE リソースアダプタの起動・停止・プロパティの設定などの操作を実行します。

［サーバー・エクスプローラー］ビューの構成例を次に示します。

図 21-1　［サーバー・エクスプローラー］ビューの構成例



［サーバー・エクスプローラー］ビューの構成について次に説明します。

ツリービュー

論理サーバ，J2EE アプリケーション，J2EE リソースアダプタなど，操作の対象がノードとして表示される，ツリー形式のビューです。
ツリービューに表示されるノードの種類を次に示します。なお，論理サーバは，起動する順で並べられています。それ以外のノードは，アルファベット順で並べられています。

ツリービューで表示するノードの種類

表 21-1　ツリービューで表示されるノード一覧

種類	説明
Management Server リモート管理機能と接続しているホスト	Management Server リモート管理機能に接続しているホストです。 名前は，「＜ホスト名または IP アドレス＞：＜ポート番号＞」で表示されます。初期表示は，このノードだけが表示されています。
運用管理ドメイン	運用管理ドメイン名（デフォルトドメイン）が表示されています。
ホスト（運用管理ドメイン内）	運用管理ドメイン名に定義されたホストが表示されています。
論理パフォーマンストレーサ	論理パフォーマンストレーサ名が表示されています。
論理スマートエージェント	論理スマートエージェント名が表示されています。
論理ネーミングサービス	論理ネーミングサービス名が表示されています。
論理 CTM ドメインマネジャ	論理 CTM ドメインマネジャ名が表示されています。

種類	説明
論理 CTM	論理 CTM 名が表示されています。
論理 SFO サーバ	SFO サーバ名が表示されています。
論理 J2EE サーバ	論理 J2EE サーバ名が表示されています。
［J2EE アプリケーション］フォルダ	デプロイされた J2EE アプリケーションをまとめるフォルダを表します。
J2EE アプリケーション	J2EE アプリケーション名が表示されています。
J2EE リソースアダプタ（J2EE アプリケーション内）	J2EE アプリケーションに含めてデプロイされるリソースアダプタ名が表示されています。
［J2EE リソース・アダプター］フォルダ	J2EE リソースアダプタをまとめるフォルダを表します。
J2EE リソースアダプタ	J2EE リソースアダプタとしてデプロイされているリソースアダプタ名が表示されています。
［J2EE リソース］フォルダ	J2EE リソースをまとめるフォルダを表します。
［リソース・アダプター］フォルダ（［J2EE リソース］フォルダ内）	J2EE リソース内の J2EE リソースアダプタをまとめるフォルダを表します。
リソースアダプタ	J2EE リソース内のリソースアダプタ名が表示されています。
論理 Web サーバ	論理 Web サーバ名が表示されています。
論理ユーザサーバ	論理ユーザサーバ名が表示されています。

ビューツールバー

ツリービューで選択した対象に対して実行できる操作がアイコンで表示されるビューです。

ビュープルダウンメニュー

ツリービューで選択した対象に対して実行できる操作がプルダウンメニューで表示されるビューです。

21.1.1 サーバー・エクスプローラーの基本操作

ここでは，［サーバー・エクスプローラー］ビューを使用した，操作の実行方法について説明します。［サーバー・エクスプローラー］ビューの操作は，次の手順で実行します。

1. **［サーバー・エクスプローラー］ビューのツリービューで，操作を実行したいノードを選択します。**
2. **ツリービューで，選択したノードに，起動，停止，実行時属性の設定などの操作を実行します。**

   次のどれかの方法で，選択したノードに対する操作を実行します。
   - ビューツールバーのアイコンをクリックする
   - ビュープルダウンメニューでメニュー項目を選択する
   - ツリービューのノードを右クリックして，表示されるコンテキストメニューのメ

ニュー項目を選択する

ビューツールバー，ビュープルダウンメニュー，および右クリックで表示されるコンテキストメニューで実行できる操作については，「(1)　実行できる操作の表示」を参照してください。

選択した処理が実行されている間は，処理状況を示すプログレスダイアログが表示されます。その間，［サーバー・エクスプローラー］ビューを使用したほかの操作は実行できません。

また，処理の実行経過は，パースペクティブウィンドウのコンソールビューにも表示されます。

(1) 実行できる操作の表示

ビューツールバー，ビュープルダウンメニュー，および右クリックで表示されるコンテキストメニューでは，実行できる操作が異なります。また，操作を実行する対象（ツリービューで，選択したノード）によっても実行できる操作が異なります。

ビューツールバー，ビュープルダウンメニュー，および右クリックで表示されるコンテキストメニューで実行できる操作の一覧を次に示します。

表 21-2　ビューツールバーでできる操作

アイコン	説明
	Management Server リモート管理機能と接続したり，接続を切断したりします。 次のサブメニューを選択します。 • **［ログイン］** Management Server リモート管理機能に接続します。 • **［ログアウト］** Management Server リモート管理機能との接続を切断します。
	論理サーバ，J2EE アプリケーション，J2EE リソースアダプタを開始します。
	論理サーバ，J2EE アプリケーション，J2EE リソースアダプタを停止します。
	サブメニューで選択した対象をインポートします。 サブメニューでは次の対象を選択できます。 • **［アプリケーション統合属性］** アプリケーション統合属性をインポートします。 • **［Connector 属性］** Connector 属性をインポートします。 • **［J2EE アプリケーション（展開ディレクトリ形式）］** 展開ディレクトリ形式の J2EE アプリケーションをインポートします。 • **［J2EE アプリケーション（アーカイブ形式）］** アーカイブ形式の J2EE アプリケーションをインポートします。 • **［リソース・アダプター（Connector）］** 接続先ホストにあるリソースアダプタをインポートします。 • **［リソース・アダプター］** Server Plug-in 環境にあるリソースアダプタをインポートします。

アイコン	説明
	サブメニューで選択した対象をエクスポートします。 サブメニューでは次の対象を選択できます。 • **［アプリケーション統合属性］** アプリケーション統合属性をエクスポートします。 • ［Connector **属性］** Connector 属性をエクスポートします。 • ［Cosminexus **アプリケーション］** 実行時情報を含めて J2EE アプリケーションをエクスポートします。 • ［Cosminexus **リソース・アダプター］** 実行時情報を含めてリソースアダプタをエクスポートします。 • ［J2EE **アプリケーション］** 実行時情報を含めないで J2EE アプリケーションをエクスポートします。 • ［J2EE **リソース・アダプター］** 実行時情報を含めないでリソースアダプタをエクスポートします。 • ［Cosminexus RMI-IIOP **インターフェース］** J2EE アプリケーションから RMI-IIOP インタフェースをエクスポートします。

表 21-3　ビュープルダウンメニューでできる操作

メニュー	説明
 開始	論理サーバ，J2EE アプリケーション，リソースアダプタを開始します。
アプリケーションを起動しないで開始	J2EE アプリケーションを起動しないで，論理 J2EE サーバを開始します。
 停止	論理サーバ，J2EE アプリケーション，J2EE リソースアダプタを停止します。
強制停止	論理サーバ，J2EE アプリケーションを強制停止します。
開始オプションの設定	論理 J2EE サーバの起動パラメタを設定します。
 リロード	展開ディレクトリ形式でインポートされた J2EE アプリケーションを手動でリロードします。
入れ替え	アーカイブ形式でインポートされた J2EE アプリケーションを入れ替えます。
アプリケーション名の変更	J2EE アプリケーション名を変更します。
 接続テスト	J2EE リソースアダプタの接続をテストします。
 デプロイ	リソースアダプタをデプロイします。
 削除	J2EE アプリケーション，リソースアダプタを削除します。
 表示の更新	［サーバー・エクスプローラー］ビューの表示を更新します。
 プロパティー	J2EE アプリケーションやリソースアダプタの属性を編集します。

表 21-4　右クリックで表示されるコンテキストメニューでできる操作

種類	コンテキストメニュー	説明
Cosminexus Management Server リモート管理機能と接続しているホスト	ログイン	Management Server リモート管理機能に接続します。
	ログアウト	Management Server リモート管理機能との接続を切断します。
運用管理ドメイン	開始	運用管理ドメイン下の論理サーバを一括起動します。
	停止	運用管理ドメイン下の論理サーバを一括停止します。
ホスト（運用管理ドメイン内）	開始	ホスト下の論理サーバを一括起動します。
	停止	ホスト下の論理サーバを一括停止します。
論理 J2EE サーバ	開始	論理 J2EE サーバを起動します。
	アプリケーションを起動しないで開始	アプリケーションを起動しないで，論理 J2EE サーバを起動します。
	停止	論理 J2EE サーバを停止します。
	強制停止	論理 J2EE サーバを強制停止します。
	開始オプションの設定	論理 J2EE サーバの起動パラメタを設定します。
［J2EE アプリケーション］フォルダ	J2EE アプリケーション（展開ディレクトリー形式）のインポート	展開ディレクトリ形式の J2EE アプリケーションをインポートします。
	J2EE アプリケーション（アーカイブ形式）のインポート	アーカイブ形式の J2EE アプリケーションをインポートします。
J2EE アプリケーション	開始	J2EE アプリケーションを開始します。
	停止	J2EE アプリケーションを停止します。
	強制停止	J2EE アプリケーションを強制停止します。
	リロード	展開ディレクトリ形式の J2EE アプリケーションをリロードします。
	入れ替え	アーカイブ形式の J2EE アプリケーションを入れ替えます。
	アプリケーション名の変更	J2EE アプリケーション名を変更します。

種類	コンテキストメニュー	説明
	 削除	J2EE アプリケーションを削除します。
	 アプリケーション統合属性のインポート	アプリケーション統合属性をインポートします。
	 アプリケーション統合属性のエクスポート	アプリケーション統合属性をエクスポートします。
	 Cosminexus アプリケーションのエクスポート	実行時情報を含めて，J2EE アプリケーションをエクスポートします。
	 J2EE アプリケーションのエクスポート	実行時情報を含めないで，J2EE アプリケーションをエクスポートします。
	 Cosminexus RMI-IIOP インターフェースのエクスポート	J2EE アプリケーションから RMI-IIOP インタフェースをエクスポートします。
	 プロパティー	アプリケーション統合属性を設定します。
J2EE リソースアダプタ（J2EE アプリケーション内）	 接続テスト	J2EE アプリケーションに含まれるリソースアダプタの接続をテストします。
	 Connector 属性のインポート	J2EE アプリケーションに含まれるリソースアダプタの Connector 属性をインポートします。
	 Connector 属性のエクスポート	J2EE アプリケーションに含まれるリソースアダプタの Connector 属性をエクスポートします。
	 プロパティー	J2EE アプリケーションに含まれるリソースアダプタの Connector 属性を設定します。
［J2EE リソース・アダプター］フォルダ	なし	コンテキストメニューは提供されていません。
J2EE リソースアダプタ	 開始	J2EE リソースアダプタとしてデプロイされたリソースアダプタを開始します。
	 停止	J2EE リソースアダプタとしてデプロイされたリソースアダプタを停止します。

種類	コンテキストメニュー	説明
	 接続テスト	J2EE リソースアダプタとしてデプロイされたリソースアダプタの接続をテストします。 なお，Connector 1.5 の仕様に準拠するリソースアダプタの場合は，Outbound リソースアダプタを含むときだけ，このコンテキストメニューが活性になります。
	 削除	J2EE リソースアダプタとしてデプロイされたリソースアダプタを削除します。
	 Connector 属性のインポート	J2EE リソースアダプタとしてデプロイされたリソースアダプタの Connector 属性をインポートします。
	 Connector 属性のエクスポート	J2EE リソースアダプタとしてデプロイされたリソースアダプタの Connector 属性をエクスポートします。
	 Cosminexus リソース・アダプターのエクスポート	実行時情報を含めて，デプロイされたリソースアダプタをエクスポートします。
	 J2EE リソース・アダプターのエクスポート	実行時情報を含めないで，J2EE リソースアダプタとしてデプロイされたリソースアダプタをエクスポートします。
	 プロパティー	J2EE リソースアダプタとしてデプロイされたリソースアダプタの Connector 属性を設定します。
［J2EE リソース］フォルダ	なし	コンテキストメニューは提供されていません。
［リソース・アダプター］フォルダ（［J2EE リソース］フォルダ内）	 リソース・アダプター（Connector）のインポート	J2EE サーバ環境にあるリソースアダプタをインポートします。
	 リソース・アダプターのインポート	Server Plug-in 環境にあるリソースアダプタをインポートします。
リソースアダプタ	 デプロイ	リソースアダプタを J2EE リソースアダプタとしてデプロイします。
	 削除	リソースアダプタを削除します。
	 Connector 属性のインポート	Connector 属性をインポートします。
	 Connector 属性のエクスポート	Connector 属性エクスポートします。

種類	コンテキストメニュー	説明
	 プロパティー	Connector 属性を設定します。
論理 J2EE サーバ以外の論理サーバ	 開始	論理サーバを起動します。
	 停止	論理サーバを停止します。
	強制停止	論理サーバを強制停止します。

(2) ツリービューに表示されるノードと実行できる操作

ビューツールバー，ビュープルダウンメニュー，および右クリックで表示されるコンテキストメニューで実行できる操作は，ツリービューで選択したノード（操作の対象）によって異なります。次に示す Server Plug-in の機能別に，ツリービューに表示されるノード，および選択したノードで実行できる操作を説明します。

- 論理サーバの運用管理
- J2EE サーバでのアプリケーション設定操作

また，どのノードも選択していない場合に実行できる操作についても説明します。

なお，処理の実行中は，［サーバー・エクスプローラ］ビューで，ほかの操作は実行できません。

(a) 論理サーバの運用管理

論理サーバのノード，および選択したノードごとに実行できる操作を次の表に示します。

表 21-5　ツリービューで選択するノードの種類（論理サーバの運用管理の場合）

項番	種類	説明
1	Management Server リモート管理機能と接続しているホスト	Management Server リモート管理機能に接続しているホスト
2	運用管理ドメイン	運用管理ドメイン（デフォルトドメイン）
3	ホスト（運用管理ドメイン内）	運用管理ドメインに定義されたホスト
4	論理 J2EE サーバ	論理 J2EE サーバ
5	論理 J2EE サーバ以外の論理サーバ	論理 J2EE サーバ以外の論理サーバ

表 21-6　ノードごとの操作の可否（論理サーバの運用管理の場合）

操作	操作手段の有無			ノード				
	ビューツールバー	ビュープルダウンメニュー	コンテキストメニュー	1	2	3	4	5
［Cosminexus Management Server リモート管理機能］	－	－	－	－	－	－	－	－
［ログイン］	○	－	○	○1	○1	○1	○1	○1
［ログアウト］	○	－	○	○2	○2	○2	○2	○2
［開始］	○	○	○	×	◎	◎	○6	○6
［停止］	○	○	○	×	◎	◎	○4	○4
［インポート］	－	－	－	－	－	－	－	－
［アプリケーション統合属性］	○	－	○	×	×	×	×	×
［Connector 属性］	○	－	○	×	×	×	×	×
［J2EE アプリケーション（展開ディレクトリー形式）］	○	－	○	×	×	×	×	×
［J2EE アプリケーション（アーカイブ形式）］	○	－	○	×	×	×	×	×
［リソース・アダプター(Connector)］	○	－	○	×	×	×	×	×
［リソース・アダプター］	○	－	○	×	×	×	×	×
［エクスポート］	－	－	－	－	－	－	－	－
［アプリケーション統合属性］	○	－	○	×	×	×	×	×
［Connector 属性］	○	－	○	×	×	×	×	×
［Cosminexus アプリケーション］	○	－	○	×	×	×	×	×
［Cosminexus リソース・アダプター］	○	－	○	×	×	×	×	×

操作	操作手段の有無			ノード				
	ビューツールバー	ビュープルダウンメニュー	コンテキストメニュー	1	2	3	4	5
［J2EE アプリケーション］	○	－	○	×	×	×	×	×
［J2EE リソース・アダプター］	○	－	○	×	×	×	×	×
［Cosminexus RMI-IIOP インターフェース］	○	－	○	◎	◎	◎	◎	◎
［強制停止］	－	○	○	×	×	×	○5	○5
［削除］	－	○	○	×	×	×	×	×
［表示の更新］	－	○	－	○2	○2	○2	○2	○2
［プロパティー］	－	○	○	×	×	×	×	×
［アプリケーションを起動しないで開始］	－	○	○	×	×	×	○3	×
［開始オプションの設定］	－	○	○	×	×	×	◎	×
［リロード］	－	○	○	×	×	×	×	×
［入れ替え］	－	○	○	×	×	×	×	×
［アプリケーション名の変更］	－	○	○	×	×	×	×	×
［接続テスト］	－	○	○	×	×	×	×	×
［デプロイ］	－	○	○	×	×	×	×	×

（凡例）

数字：表 21-5 の項番に対応するノードの種類を示す。

○：操作手段を提供する。

◎：常に実行できる。

○1：Management Server リモート管理機能への接続が切断されているときだけ実行できる。

○2：Management Server リモート管理機能に接続されているときだけ実行できる。

○3：停止状態のときだけ実行できる。

○4：開始状態のときだけ実行できる。

○5：停止状態および異常停止状態のどちらでもないときに実行できる。

○6：停止状態または異常停止状態のときだけ実行できる。

×：常に実行できない。

－：該当なし。

(b) J2EE サーバでのアプリケーション設定操作

論理 J2EE サーバ内のノード，および選択したノードごとに，実行できる操作を次の表

に示します。

表 21-7　ツリービューで選択するノードの種類（アプリケーション設定操作の場合）

項番	種類	説明
1	［J2EE アプリケーション］フォルダ	（固定値）
2	J2EE アプリケーション	J2EE アプリケーション
3	J2EE リソースアダプタ（J2EE アプリケーション内）	J2EE アプリケーションに含まれるリソースアダプタ
4	［J2EE リソース・アダプター］フォルダ	（固定値）
5	J2EE リソースアダプタ	デプロイされたリソースアダプタ
6	［J2EE リソース］フォルダ	（固定値）
7	［リソース・アダプター］フォルダ	（固定値）
8	リソースアダプタ	J2EE リソース内のリソースアダプタ

表 21-8　ノードごとの操作の可否（アプリケーション設定操作の場合）

操作	操作手段の有無			ノード							
	ビューツールバー	ビュープルダウンメニュー	コンテキストメニュー	1	2	3	4	5	6	7	8
［Cosminexus Management Server リモート管理機能］	－	－	－	－	－	－	－	－	－	－	－
［ログイン］	○	－	○	○ 1	○ 1	○ 1	○ 1	○ 1	○ 1	○ 1	○ 1
［ログアウト］	○	－	○	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2
［開始］	○	○	○	×	○ 3	×	×	○ 3	×	×	×
［停止］	○	○	○	×	○ 4	×	×	○ 4	×	×	×
［インポート］	－	－	－	－	－	－	－	－	－	－	－
［アプリケーション統合属性］	○	－	○	×	○ 3	×	×	×	×	×	×
［Connector 属性］	○	－	○	×	×	○ 3 ※5	×	○ 3	×	×	◎

操作	操作手段の有無			ノード							
	ビューツールバー	ビュープルダウンメニュー	コンテキストメニュー	1	2	3	4	5	6	7	8
［J2EE アプリケーション（展開ディレクトリー形式）］	○	－	○	◎ ※3	×	×	×	×	×	×	×
［J2EE アプリケーション（アーカイブ形式）］	○	－	○	◎	×	×	×	×	×	×	×
［リソース・アダプター(Connector)］	○	－	○	×	×	×	×	×	×	◎	×
［リソース・アダプター］	○	－	○	×	×	×	×	×	×	◎	×
［エクスポート］	－	－	－	－	－	－	－	－	－	－	－
［アプリケーション統合属性］	○	－	○	×	○ 5	×	×	×	×	×	×
［Connector 属性］	○	－	○	×	×	◎	×	○ 5	×	×	◎
［Cosminexus アプリケーション］	○	－	○	×	○ 5	×	×	×	×	×	×
［Cosminexus リソース・アダプター］	○	－	○	×	×	×	×	○ 5	×	×	×
［J2EE アプリケーション］	○	－	○	×	○ 5	×	×	×	×	×	×
［J2EE リソース・アダプター］	○	－	○	×	×	×	×	○ 5	×	×	×
［Cosminexus RMI-IIOP インターフェース］	○	－	○	◎	◎	◎	◎	◎	◎	◎	◎
［強制停止］	－	○	○	×	○ 6	×	×	×	×	×	×

操作	操作手段の有無			ノード							
	ビューツールバー	ビュープルダウンメニュー	コンテキストメニュー	1	2	3	4	5	6	7	8
［削除］	－	○	○	×	○ 3	×	×	○ 3	×	×	◎
［表示の更新］	－	○	－	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2
［プロパティー］	－	○	○	×	○ 3 ※6	○ 3 ※4	×	○ 3	×	×	○ 3
［アプリケーションを起動しないで開始］	－	○	○	×	×	×	×	×	×	×	×
［開始オプションの設定］	－	○	○	×	×	×	×	×	×	×	×
［リロード］	－	○	○	×	○ 4 ※1	×	×	×	×	×	×
［入れ替え］	－	○	○	×	○ 5 ※2	×	×	×	×	×	×
［アプリケーション名の変更］	－	○	○	×	○ 3 ※4, ※6	×	×	×	×	×	×
［接続テスト］	－	○	○	×	×	◎	×	○ 5	×	×	×
［デプロイ］	－	○	○	×	×	×	×	×	×	×	◎

（凡例）

数字：表 21-7 の項番に対応するノードの種類を示す。

○：操作手段を提供する。

◎：常に実行できる。

○ 1：Management Server リモート管理機能への接続が切断されているときだけ実行できる。

○ 2：Management Server リモート管理機能に接続されているときだけ実行できる。

○ 3：停止状態のときだけ実行できる。

○ 4：開始状態のときだけ実行できる。

○ 5：開始状態および停止状態のときだけ実行できる。

○ 6：通常停止中および閉塞状態のときだけ実行できる。

×：常に実行できない。

－：該当なし。

注※ 1　展開ディレクトリ形式の J2EE アプリケーションだけを対象とします。

注※ 2　アーカイブ形式の J2EE アプリケーションだけを対象とします。

注※3　ローカルホストの論理 J2EE サーバだけを対象とします。
注※4　J2EE アプリケーションの属性エディタが開いている場合は実行できません。
注※5　リソースアダプタが含まれる J2EE アプリケーションが停止状態の場合に実行できます。
注※6　J2EE アプリケーションに含まれるリソースアダプタの属性エディタが開いている場合は実行できません。

(c) ノードも選択されていない場合

ツリービューで，どのノードも選択されていない場合，実行できる操作を次の表に示します。

表 21-9　ノードが選択されていない場合の操作

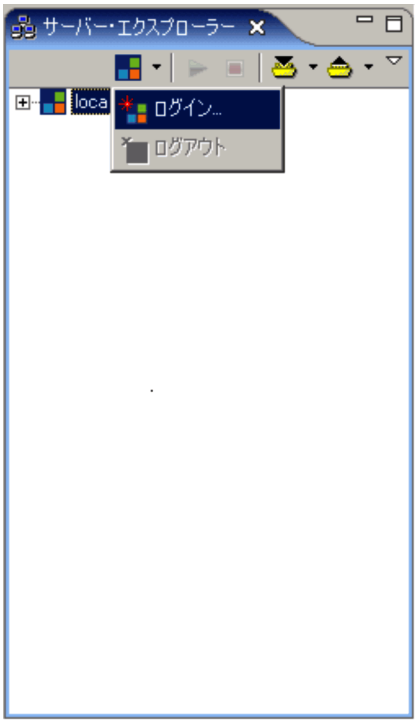
操作	操作手段の有無			ノードが選択されていない場合
	ビューツールバー	ビュープルダウンメニュー	コンテキストメニュー	
［Cosminexus Management Server リモート管理機能］	－	－	－	－
［ログイン］	○	－	○	○1
［ログアウト］	○	－	○	○2
［エクスポート］	－	－	－	－
［Cosminexus RMI-IIOP インターフェース］	○	－	○	◎
［表示の更新］	－	○	－	○2

（凡例）
○：操作手段を提供する。
◎：常に実行できる。
○1：Management Server リモート管理機能への接続が切断されているときだけ実行できる。
○2：Management Server リモート管理機能に接続されているときだけ実行できる
－：該当なし。

21.1.2　Management Server リモート管理機能へのログイン

Management Server リモート管理機能と接続する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-2　Management Server リモート管理機能と接続する場合の［サーバー・エクスプローラー］ビュー



次のどちらかの方法で，Management Server リモート管理機能に接続します。

● ビューツールバーを使用

ビューツールバーの［ ］（Management Server リモート管理機能）のサブメニュー［ ログイン］を選択します。ツリービューのノードの選択は不要です。

● コンテキストメニューを使用

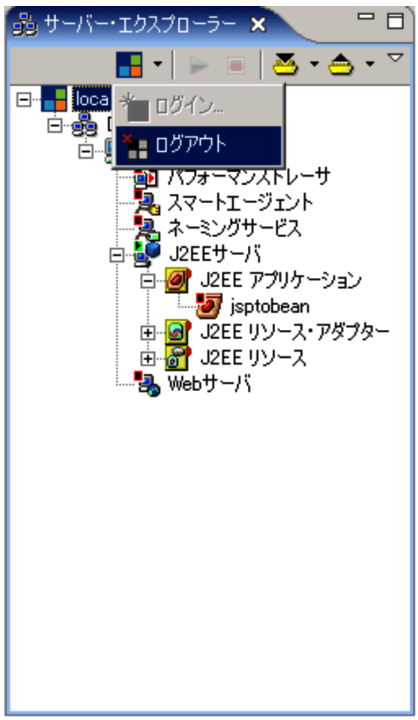
1. ツリービューで，Management Server リモート管理機能と接続するホストを選択します。
2. 右クリックで表示されるコンテキストメニューの［ ログイン］を選択します。

操作が実行されると，［Cosminexus Management Server リモート管理機能へログイン］ダイアログが表示されます。［Cosminexus Management Server リモート管理機能へログイン］ダイアログについては，「11.1.2　Management Server リモート管理機能への接続」を参照してください。

21.1.3　Management Server リモート管理機能からのログアウト

Management Server リモート管理機能との接続を切断する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-3　Management Server リモート管理機能との接続を切断する場合の［サーバー・エクスプローラー］ビュー



次のどちらかの方法で，Management Server リモート管理機能との接続を切断します。

● ビューツールバーを使用

ビューツールバーの［ ］（Management Server リモート管理機能）のサブメニュー［ ログアウト］を選択します。ツリービューのノードの選択は不要です。

● コンテキストメニューを使用

1. ツリービューで，Management Server リモート管理機能と接続しているホストを選択します。
2. 右クリックで表示されるコンテキストメニューの［ ログアウト］を選択します。

Management Server リモート管理機能からのログアウトが実行されると，確認メッセージが表示されます。ログアウトを続行する場合は，メッセージダイアログで［はい］を

クリックします。中止する場合は［いいえ］を選択します。

21.1.4　開始

論理サーバや J2EE アプリケーションを開始する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

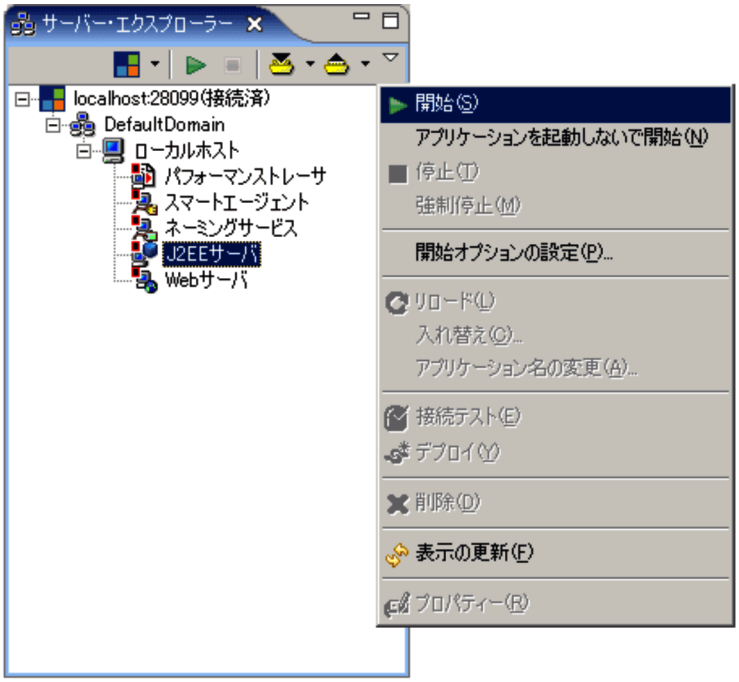
［サーバー・エクスプローラー］ビューで開始できる対象を次の表に示します。

表 21-10　［サーバー・エクスプローラー］ビューで開始できる対象

開始対象	開始動作
運用管理ドメイン	運用管理ドメインのすべての論理サーバを一括起動します。
ホスト	選択したホストのすべての論理サーバを一括起動します。
論理サーバ	選択した論理サーバを起動します。
J2EE アプリケーション	選択した J2EE アプリケーションを開始します。
［J2EE リソースアダプタ］フォルダ下のリソースアダプタ	選択されたリソースアダプタを開始します。

対象を開始する場合の［サーバー・エクスプローラー］ビューと操作手順を次に示します。

図 21-4　対象を開始する場合の［サーバー・エクスプローラー］ビュー



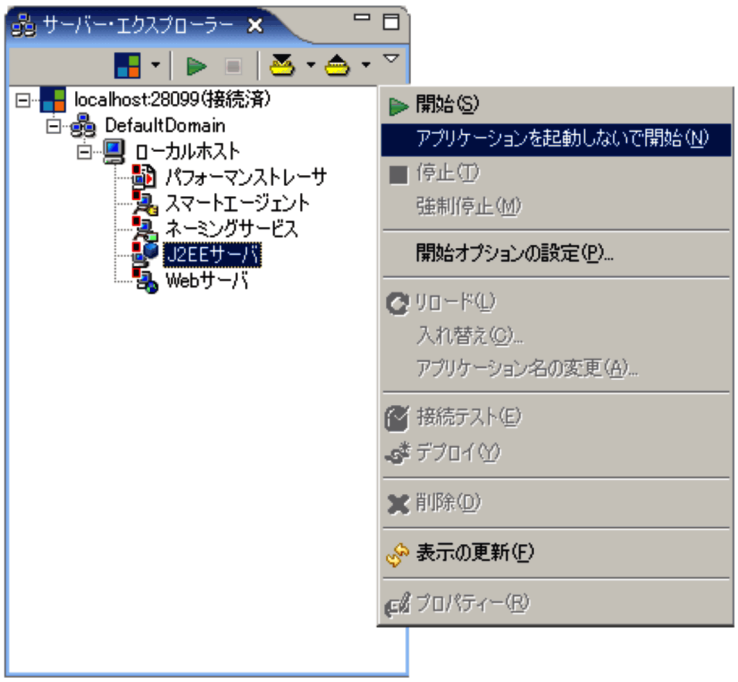
1. **ツリービューで，開始する対象を選択します。**

2. **選択した対象を開始します。**
   次のどれかの方法で，開始の操作を実行します。
   - ビューツールバーの［ ▶ ］を選択
   - ビュープルダウンメニューの［ ▶ 開始］を選択
   - 右クリックで表示されるコンテキストメニューの［ ▶ 開始］を選択

21.1.5　アプリケーションを起動しない場合の J2EE サーバの開始

J2EE アプリケーションを起動しないで論理 J2EE サーバを開始する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-5　アプリケーションを起動しないで J2EE サーバを開始する場合の［サーバー・エクスプローラー］ビュー

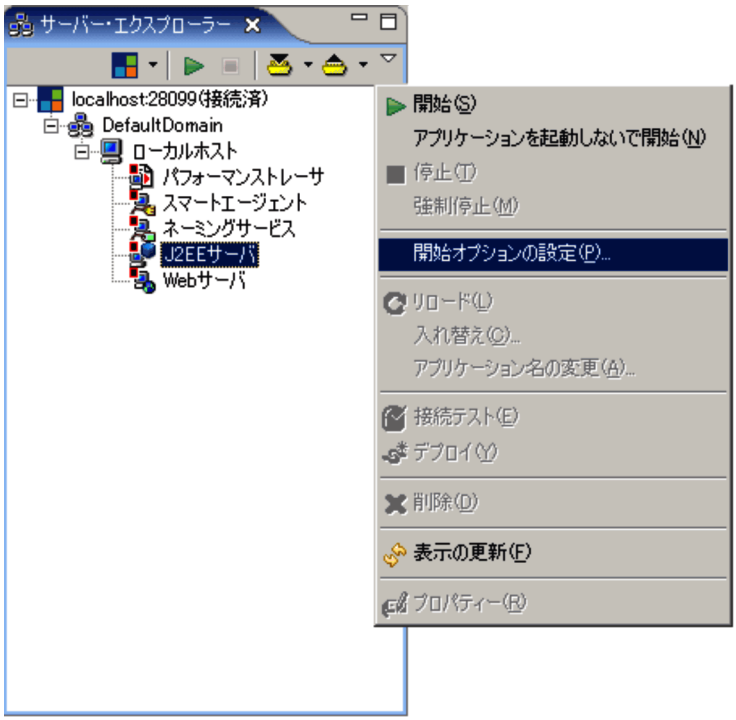


1. **ツリービューで，開始する J2EE サーバを選択します。**
2. **選択した J2EE サーバを開始します。**
   次のどちらかの方法で，開始の操作を実行します。
   - ビュープルダウンメニューの［アプリケーションを起動しないで開始］を選択
   - 右クリックで表示されるコンテキストメニューの［アプリケーションを起動しないで開始］を選択

21.1.6　論理 J2EE サーバの開始オプションの設定

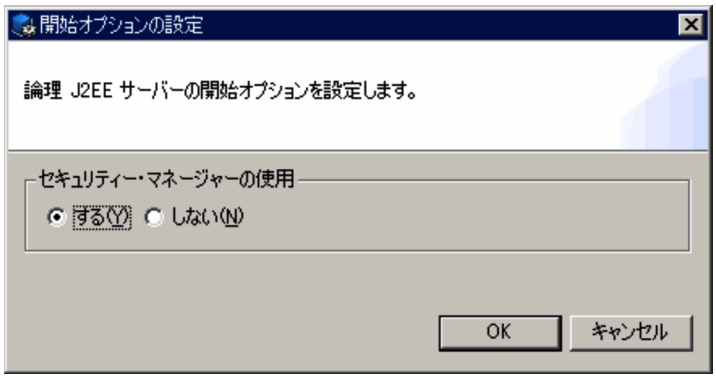
論理 J2EE サーバの開始オプションを設定する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-6　論理 J2EE サーバの開始オプションを設定する場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，開始オプションを設定する論理 J2EE サーバを選択します。**
2. **［開始オプションの設定］ダイアログを表示させます。**
   次のどちらかの方法で，［開始オプションの設定］ダイアログを表示させます。
   - 右クリックで表示されるコンテキストメニューの［開始オプションの設定］を選択
   - ビュープルダウンメニューの［開始オプションの設定］を選択
3. **表示された［開始オプションの設定］ダイアログで開始オプションを設定します。**

図 21-7　［開始オプションの設定］ダイアログ



［開始オプションの設定］ダイアログに表示される項目，およびボタンについて次に示します。

表 21-11　［開始オプションの設定］ダイアログに表示される項目，およびボタン

項目	説明
［セキュリティー・マネージャーの使用］ラジオボタン	セキュリティマネジャを使用するかどうかを，［する (Y)］，または［しない (N)］から選択してください。 ［しない (N)］を選択すると，論理 J2EE サーバの起動パラメタに -nosecurity オプションが設定されます。［する (Y)］を選択すると，論理 J2EE サーバの起動パラメタの -nosecurity オプションは解除されます。
［OK］ボタン	論理 J2EE サーバの開始パラメタが更新されて，［開始オプションの設定］ダイアログが閉じます。
［キャンセル］ボタン	論理 J2EE サーバの開始パラメタが更新されずに，［開始オプションの設定］ダイアログが閉じます。

21.1.7　停止

論理サーバや J2EE アプリケーションを停止する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

［サーバー・エクスプローラー］ビューで停止できる対象を次の表に示します。

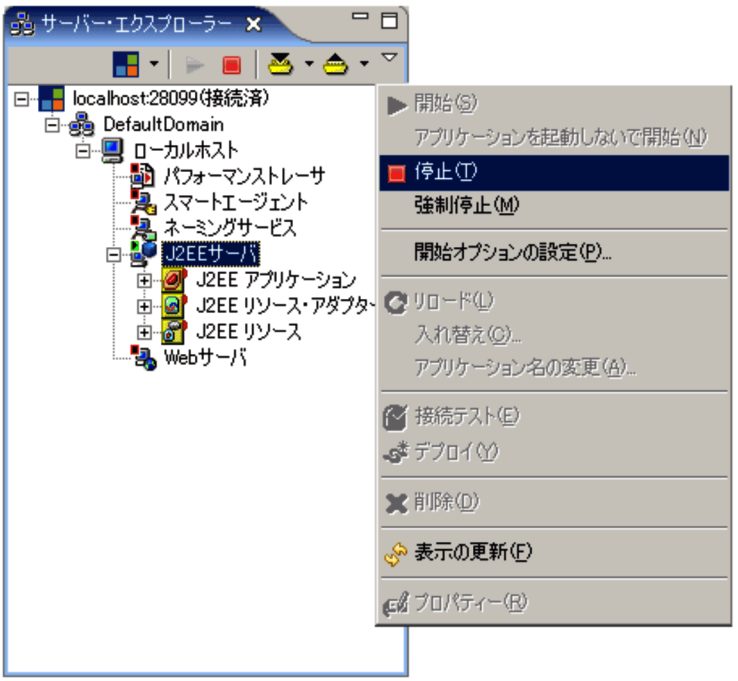
表 21-12　［サーバー・エクスプローラー］ビューで停止できる対象

停止対象	停止動作
運用管理ドメイン	運用管理ドメインのすべての論理サーバが一括停止します。
ホスト	選択したホストのすべての論理サーバが一括停止します。
論理サーバ	選択した論理サーバが停止します。
J2EE アプリケーション	選択した J2EE アプリケーションが停止します。

停止対象	停止動作
［J2EE リソースアダプタ］フォルダのリソースアダプタ	選択されたリソースアダプタが停止します。

対象を停止する場合の［サーバー・エクスプローラー］ビューと操作手順を次に示します。

図 21-8　［停止］する場合の［サーバー・エクスプローラー］ビュー

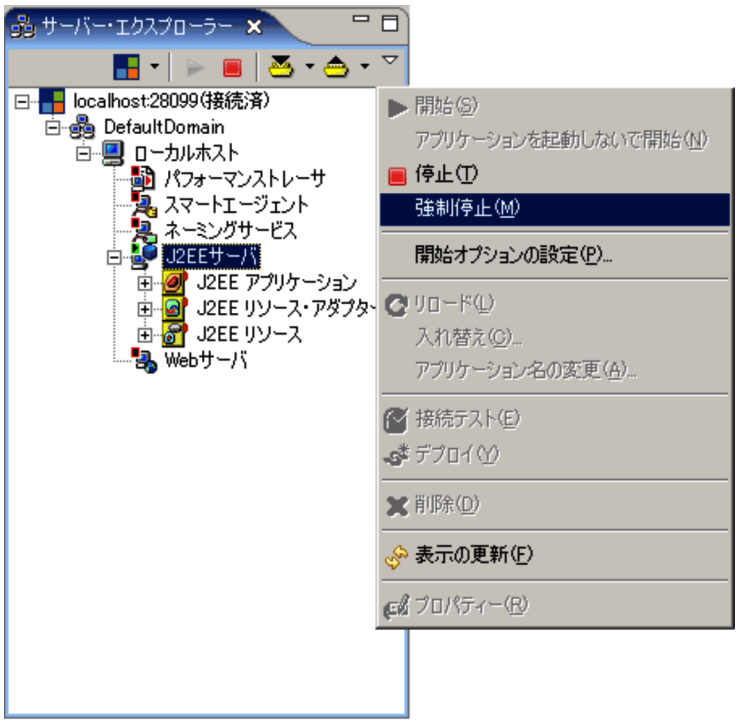


1. **ツリービューで，停止する対象を選択します。**
2. **選択した対象を停止します。**
   次のどれかの方法で，停止の操作を実行します。
   - ビューツールバーの［ ］を選択
   - ビュープルダウンメニューの［ 停止］を選択
   - 右クリックで表示されるコンテキストメニューの［ 停止］を選択

21.1.8　強制停止

論理サーバや J2EE アプリケーションを強制停止する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-9　対象を強制停止する場合の［サーバー・エクスプローラー］ビュー

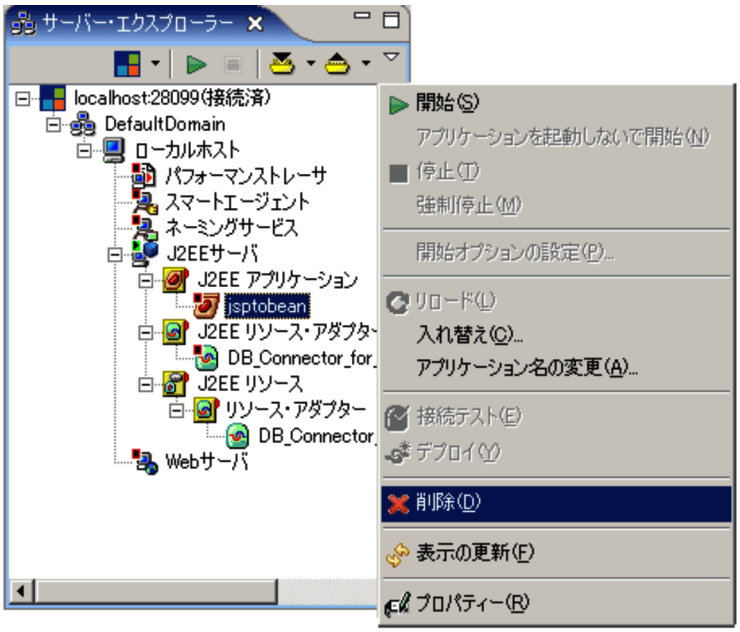


1. **ツリービューで，強制停止する論理サーバ，または J2EE アプリケーションを選択します。**
2. **選択した対象を強制停止します。**
   次のどちらかの方法で，強制停止の操作を実行します。
   - ビュープルダウンメニューの［強制停止］を選択
   - 右クリックで表示されるコンテキストメニューの［強制停止］を選択

21.1.9　削除

ツリービューで選択した対象を削除する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-10　対象を削除する場合の［サーバー・エクスプローラー］ビュー



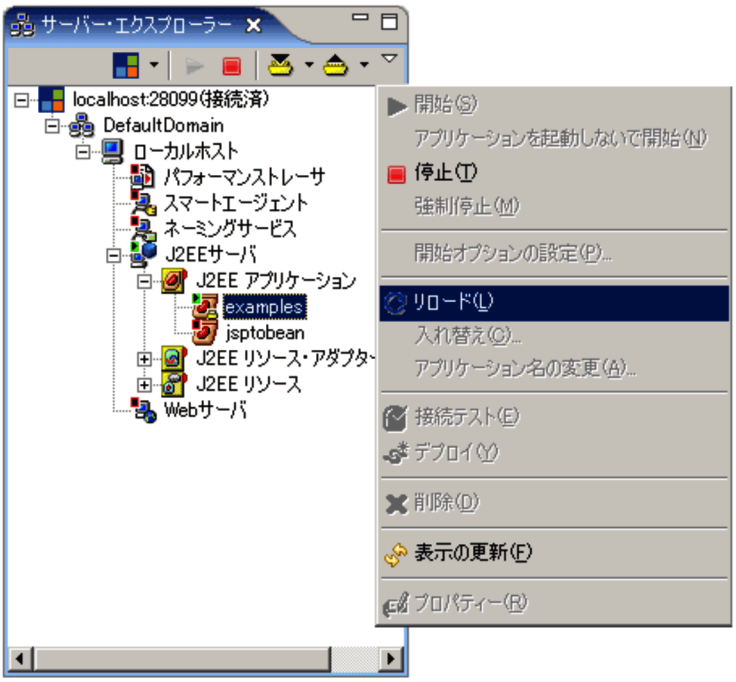
1. **ツリービューで，削除する対象を選択します。**
2. **選択した対象を削除します。**
   次のどちらかの方法で，削除の操作を実行します。
   - ビュープルダウンメニューの［削除］を選択
   - 右クリックで表示されるコンテキストメニューの［削除］を選択

   削除の操作を実行したあと，削除を確認するためのメッセージダイアログが表示されます。削除する場合は，［はい］を選択してください。削除を中止する場合は，［いいえ］を選択してください。

21.1.10　リロード

展開ディレクトリ形式でインポートされた J2EE アプリケーションをリロードする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-11　J2EE アプリケーションをリロードする場合の［サーバー・エクスプローラー］ビュー

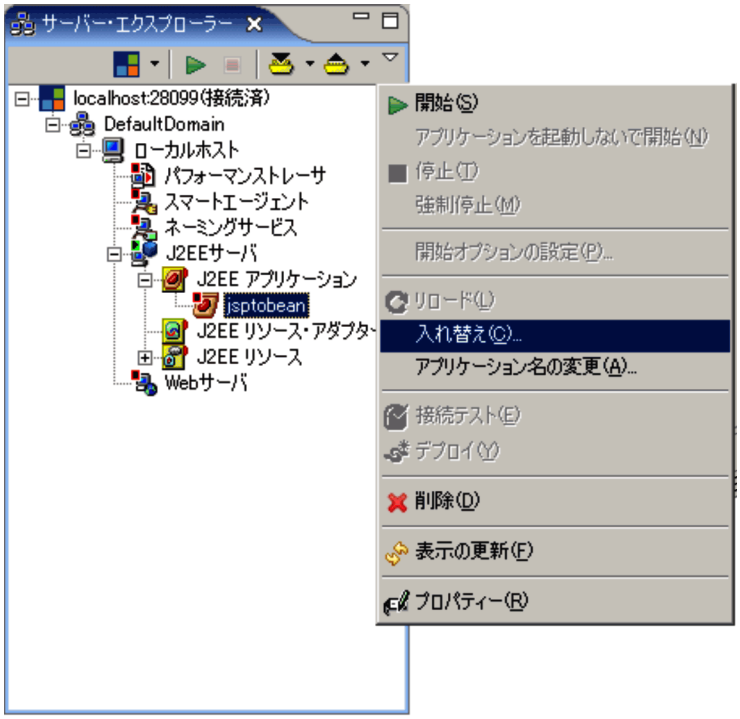


1. **ツリービューで，リロードする** J2EE **アプリケーションを選択します。**
2. **選択した** J2EE **アプリケーションをリロードします。**
   次のどちらかの方法で，リロードの操作を実行します。
   - ビュープルダウンメニューの［ リロード］を選択
   - 右クリックで表示されるコンテキストメニューの［ リロード］を選択

21.1.11　J2EE アプリケーションの入れ替え

アーカイブ形式でインポートされた J2EE アプリケーションを入れ替える場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-12　J2EE アプリケーションを入れ替える場合の［サーバー・エクスプローラー］ビュー



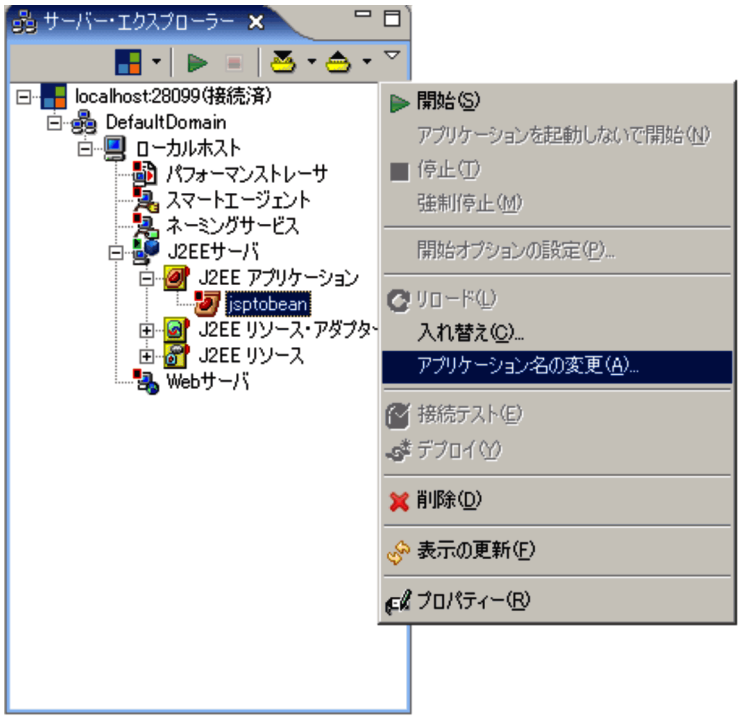
1. **ツリービューで，入れ替える** J2EE **アプリケーションを選択します。**
2. **選択した** J2EE **アプリケーションを入れ替えます。**
   次のどちらかの方法で，入れ替えの操作を実行します。
   - ビュープルダウンメニューの［入れ替え］を選択
   - 右クリックで表示されるコンテキストメニューの［入れ替え］を選択

   ［J2EE アプリケーションの入れ替え］ダイアログが表示されます。
3. **［**J2EE **アプリケーションの入れ替え］ダイアログで，入れ替えるアプリケーション（**EAR **ファイル）を選択します。**
4. **［**J2EE **アプリケーションの入れ替え］ダイアログで［開く］ボタンをクリックします。**
   アプリケーションの入れ替えが開始されます。

21.1.12　J2EE アプリケーション名の変更

J2EE アプリケーションのアプリケーション名を変更する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

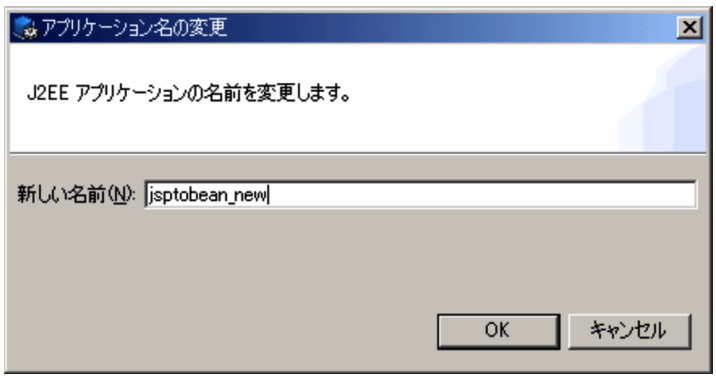
図 21-13 アプリケーション名を変更する場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，アプリケーション名を変更する J2EE アプリケーションを選択します。**
2. **選択した J2EE アプリケーションのアプリケーション名を変更します。**
   次のどちらかの方法で，アプリケーション名変更の操作を実行します。
   - ビュープルダウンメニューの［アプリケーション名の変更］を選択
   - 右クリックで表示されるコンテキストメニューの［アプリケーション名の変更］を選択

   ［アプリケーション名の変更］ダイアログが表示されます。

図 21-14 ［アプリケーション名の変更］ダイアログ



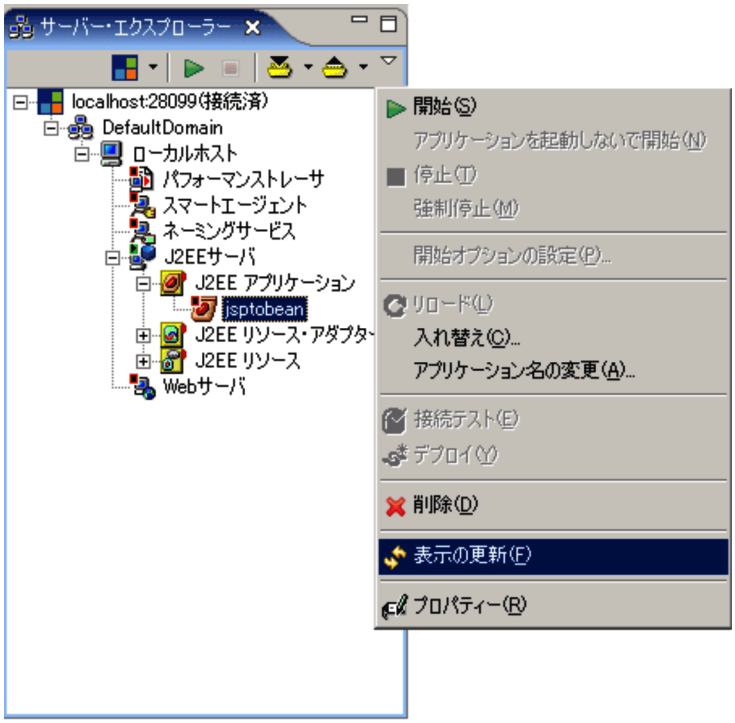
3. **［アプリケーション名の変更］ダイアログで，［新しい名前］に変更後のアプリケーション名を入力します。**
   デフォルトとして，変更前のアプリケーション名が表示され，［OK］ボタンは不活性となっています。
   新しいアプリケーション名には，英数字およびアンダースコアの文字が指定できます。また，同じ名前のアプリケーションがすでにある場合は，そのアプリケーション名は指定できません。
4. **［アプリケーション名の変更］ダイアログで［OK］ボタンをクリックします。**
   J2EE アプリケーションのアプリケーション名の変更が実行されます。
   アプリケーション名を変更しない場合，［キャンセル］ボタンをクリックしてください。
   ［アプリケーション名の変更］ダイアログが閉じます。

［新しい名前］で指定した値に誤りがある場合，［アプリケーション名の変更］ダイアログのタイトル行にエラーメッセージが表示されます。

21.1.13　表示の更新

ツリービューの表示を最新の状態に更新する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-15　表示を更新する場合の［サーバー・エクスプローラー］ビュー



1. **ビュープルダウンメニューから［ 表示の更新］を選択します。**
   ［サーバー・エクスプローラー］ビューの最新の状態が表示されます。

21.1.14　プロパティの設定

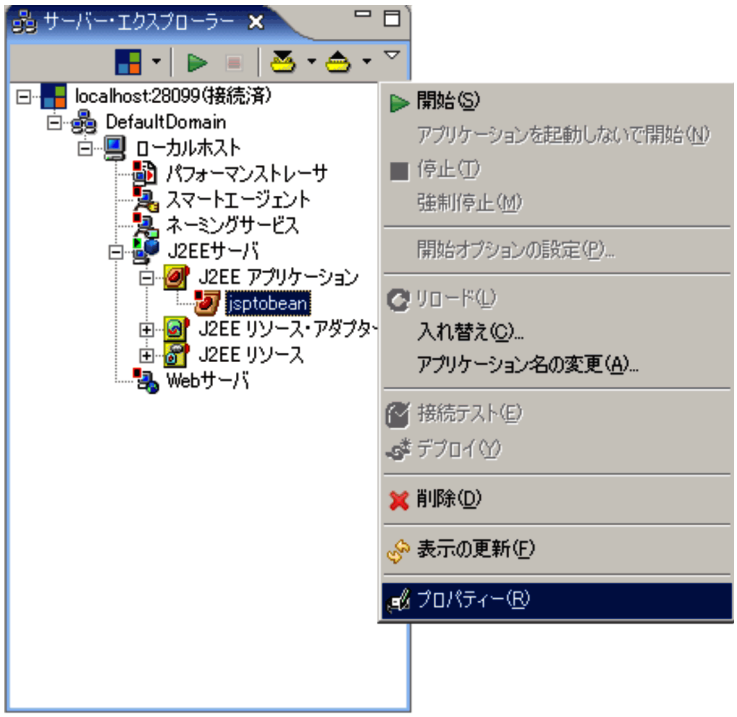
J2EE アプリケーションやリソースアダプタのプロパティを設定する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

プロパティの設定の操作を実行すると，次の処理が行われます。

1. アプリケーション統合属性，または Connector 属性が取得されます。
2. 属性ファイル編集エディタが起動されます。
3. プロパティ設定項目を編集します。
4. 属性ファイルを保管します。

プロパティを設定する場合の［サーバー・エクスプローラー］ビューと操作手順を次に示します。

図 21-16　プロパティを設定する場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，プロパティを設定する J2EE アプリケーション，またはリソースアダプタを選択します。**
2. **属性ファイル編集エディタを起動します。**
   次のどれかの方法で，属性ファイル編集エディタを起動します。

- ツリービューで選択したノードをダブルクリック
- ビュープルダウンメニューの［  プロパティー］を選択
- 右クリックで表示されるコンテキストメニューの［ プロパティー］を選択

上記 1. のツリービューで選択したプロパティ設定対象によって，次の属性ファイル編集エディタが起動されます。

ツリービューで選択したプロパティ設定対象	属性ファイル編集エディタ
［＜J2EE アプリケーション名＞］※1	アプリケーション統合属性ファイル編集エディタ
［＜リソースアダプタ名＞］（［＜J2EE アプリケーション名＞］内）※1	Connector 属性ファイル編集エディタ
［＜J2EE リソースアダプタ名＞］（［J2EE リソース・アダプター］フォルダ内）※2	Connector 属性ファイル編集エディタ
［＜リソースアダプタ名＞］（［リソース・アダプター］フォルダ内）※3	Connector 属性ファイル編集エディタ

注※ 1　J2EE アプリケーションに含まれるリソースアダプタのプロパティを設定します。

J2EE アプリケーションに含まれるリソースアダプタのプロパティを設定する場合，アプリケーション統合属性ファイル編集エディタと Connector 属性ファイル編集エディタのどちらも使用できます。ただし，どちらかの属性ファイル編集エディタが開いている場合，一方の属性ファイル編集エディタは開くことができません。属性ファイル編集エディタの選択方法については，「(1)　J2EE アプリケーションに含まれるリソースアダプタに使用する属性ファイル編集エディタ」を参照してください。

注※ 2　J2EE リソースアダプタとしてデプロイされているリソースアダプタのプロパティを設定します。

注※ 3　J2EE リソースアダプタとしてデプロイされていないリソースアダプタのプロパティを設定します。

3. **属性ファイル編集エディタで，属性ファイルを更新します。**
リソースアダプタの属性編集については，「21.2　リソースアダプタの属性編集画面」を参照してください。J2EE アプリケーションの属性編集については，「21.3　J2EE アプリケーションの属性編集画面」を参照してください。

4. **属性ファイル編集エディタで保存，または属性ファイル編集エディタを終了して，属性ファイルをサーバへアップロードします。**
属性ファイルを編集した状態で，属性ファイル編集エディタを閉じると，保管するかどうかの確認メッセージが表示されます。「はい」をクリックすると，保管処理が実行され，エラーがなければサーバへ更新情報が設定されます。「いいえ」をクリックすると，編集は破棄されます。「キャンセル」をクリックするとエディタはそのままの状態で，保管処理も実行されません。

属性ファイルの保管方法については，「11.3.1　プロパティの設定操作」を参照してください。
また，次の場合はメッセージダイアログが表示され，サーバへアップロードしません。

- 属性ファイルにエラーがある（XML 検証でエラー）
- サーバ側のコマンドの失敗

属性ファイル編集エディタについては，「11.3.2　属性ファイル編集エディタ」を参照してください。

注意事項

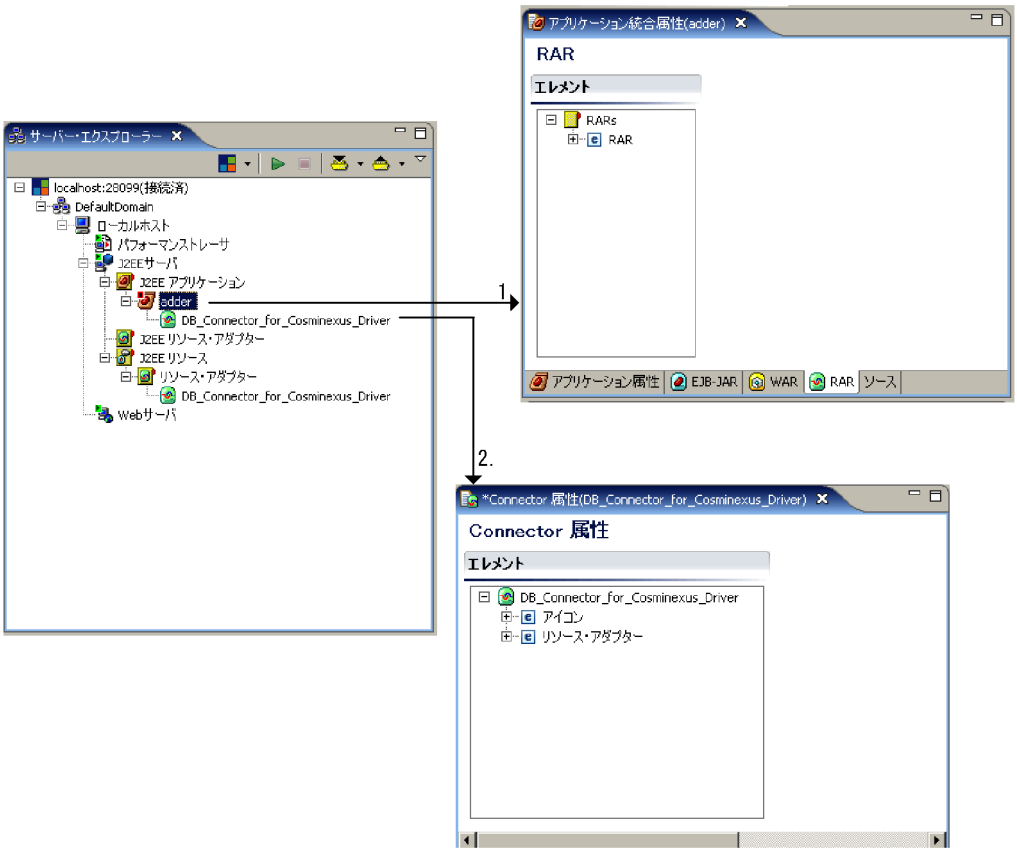
属性ファイル編集エディタのフォームページでは，属性ファイルで「xml:lang」属性が設定できる項目については，「xml:lang="en"」が設定されているか，または「xml:lang」が設定されていない場合だけ，表示，追加，更新ができます。
「xml:lang」属性が設定できる項目については，マニュアル「Cosminexus アプリケーションサーバ リファレンス 定義編（アプリケーション／リソース定義）」を参照してください。

（1）J2EE アプリケーションに含まれるリソースアダプタに使用する属性ファイル編集エディタ

J2EE アプリケーションに含まれるリソースアダプタのプロパティ設定操作では，アプリケーション統合属性ファイル編集エディタと Connector 属性ファイル編集エディタのどちらも使用できます。

アプリケーション統合属性ファイル編集エディタを使用する場合，アプリケーション統合属性ファイルの Connector 属性を編集します。また，Connector 属性ファイル編集エディタを使用する場合，リソースアダプタごとの Connector 属性を編集します。それぞれの属性ファイル編集エディタを起動する操作を，次の図に示します。

図 21-17　J2EE アプリケーションに含まれるリソースアダプタの属性ファイル編集エディタの起動



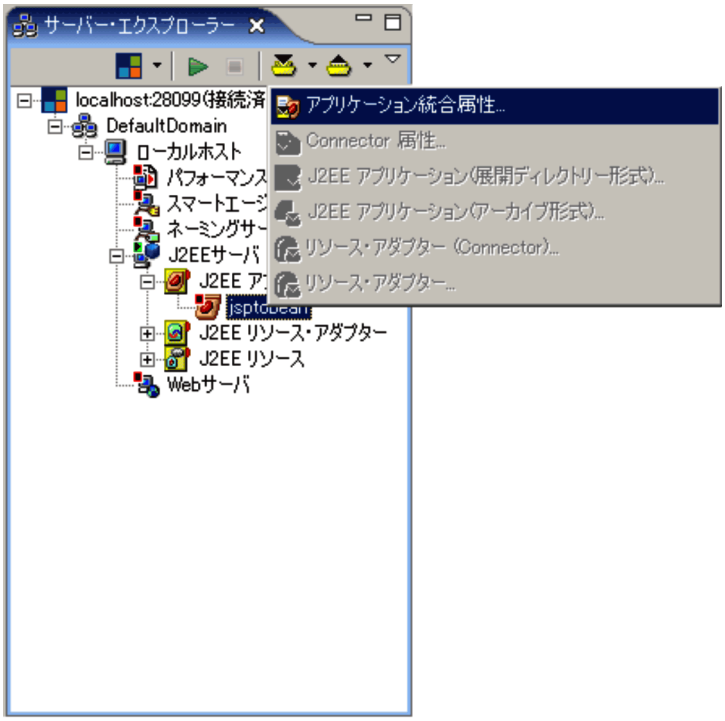
1. J2EEアプリケーションを選択して、プロパティ設定操作を実行すると，アプリケーション統合属性ファイル編集エディタが起動されます。

2. J2EEアプリケーションに含まれるリソースアダプタを選択して、プロパティ設定操作を実行すると，Connector属性ファイル編集エディタが起動されます。

21.1.15　アプリケーション統合属性のインポート

J2EE アプリケーションの属性を，アプリケーション統合属性ファイルの値に変更する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-18　J2EE アプリケーションの属性を，アプリケーション統合属性ファイルの値に変更する場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，属性を変更する J2EE アプリケーションを選択します。**
2. **［アプリケーション統合属性のインポート］ダイアログを表示させます。**
   次のどちらかの方法で，［アプリケーション統合属性のインポート］ダイアログを表示させます。
   - ビューツールバーの［ ］（インポート）のサブメニュー［ アプリケーション統合属性］を選択
   - 右クリックで表示されるコンテキストメニューの［ アプリケーション統合属性］を選択

   ［アプリケーション統合属性のインポート］ダイアログが表示されます。
3. **［アプリケーション統合属性のインポート］ダイアログでアプリケーション統合属性ファイルを選択します。**
4. **［アプリケーション統合属性のインポート］ダイアログで［開く］ボタンをクリックします。**
   アプリケーション統合属性のインポートが開始されます。

インポートダイアログで指定したファイルに，次のエラーがある場合，メッセージダイアログが表示されます。ダイアログの［OK］ボタンをクリックして，上記操作の 3. に戻ります。

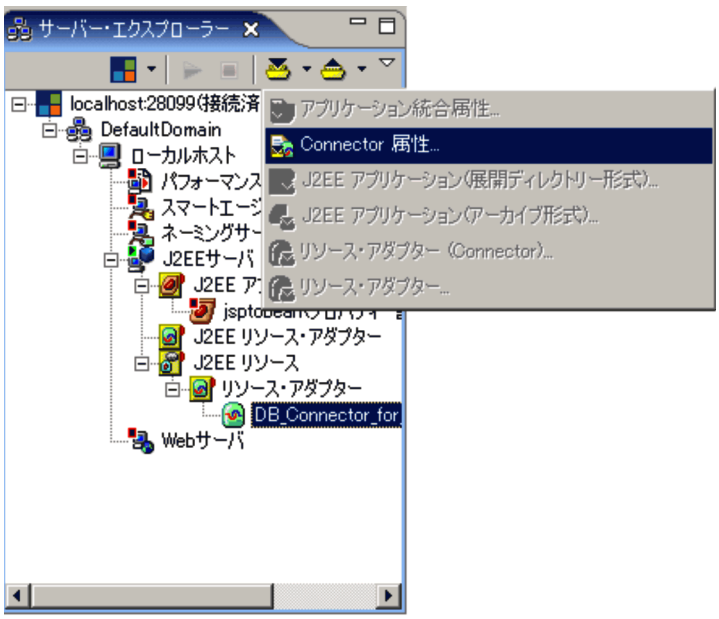
- インポートダイアログで指定したファイルが存在しない
- インポートダイアログで指定したファイルに読み取り権限がない

インポートダイアログで指定したファイルに読み取り権限がない場合，表示されたメッセージダイアログの［詳細］ボタンをクリックすると，発生した例外情報が表示されます。

21.1.16　Connector 属性のインポート

リソースアダプタの属性を Connector 属性ファイルの値に変更する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-19　リソースアダプタの属性を Connector 属性ファイルの値に変更する場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，属性を変更するリソースアダプタを選択します。**
2. **次のどちらかの方法で，［Connector 属性のインポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（インポート）のサブメニュー［ Connector 属性］を選択
   - 右クリックで表示されるコンテキストメニューの［ Connector 属性のインポート］を選択

   ［Connector 属性のインポート］ダイアログが表示されます。
3. **［Connector 属性のインポート］ダイアログで Connector 属性ファイルを選択します。**
4. **［Connector 属性のインポート］ダイアログで［開く］ボタンを選択します。**

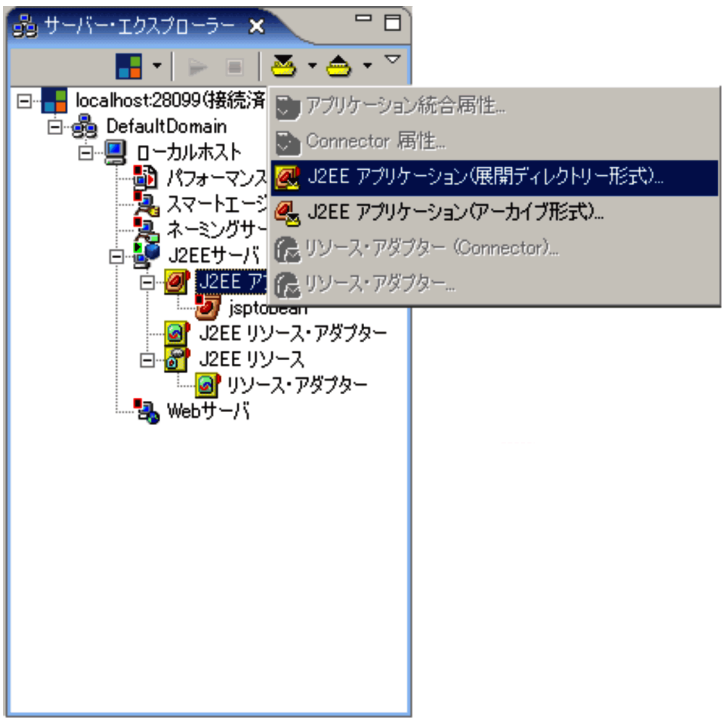
Connector 属性のインポートが開始されます。

インポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.15　アプリケーション統合属性のインポート」を参照してください。

21.1.17　展開ディレクトリ形式の J2EE アプリケーションのインポート

展開ディレクトリ形式の J2EE アプリケーションをインポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-20　展開ディレクトリ形式の J2EE アプリケーションをインポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，［J2EE アプリケーション］フォルダを選択します。**
2. **次のどちらかの方法で，［J2EE アプリケーション (展開ディレクトリー形式) のインポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（インポート）のサブメニュー［ J2EE アプリケーション (展開ディレクトリー形式)］を選択
   - 右クリックで表示されるコンテキストメニューの［ J2EE アプリケーション (展開ディレクトリー形式) のインポート］を選択

図 21-21　［J2EE アプリケーション (展開ディレクトリー形式) のインポート］ダイアログ

3. **［J2EE アプリケーション (展開ディレクトリー形式) のインポート］ダイアログでアプリケーションディレクトリを選択します。**
4. **［J2EE アプリケーション (展開ディレクトリー形式) のインポート］ダイアログで［OK］ボタンをクリックします。**
   展開ディレクトリ形式の J2EE アプリケーションのインポートが開始されます。

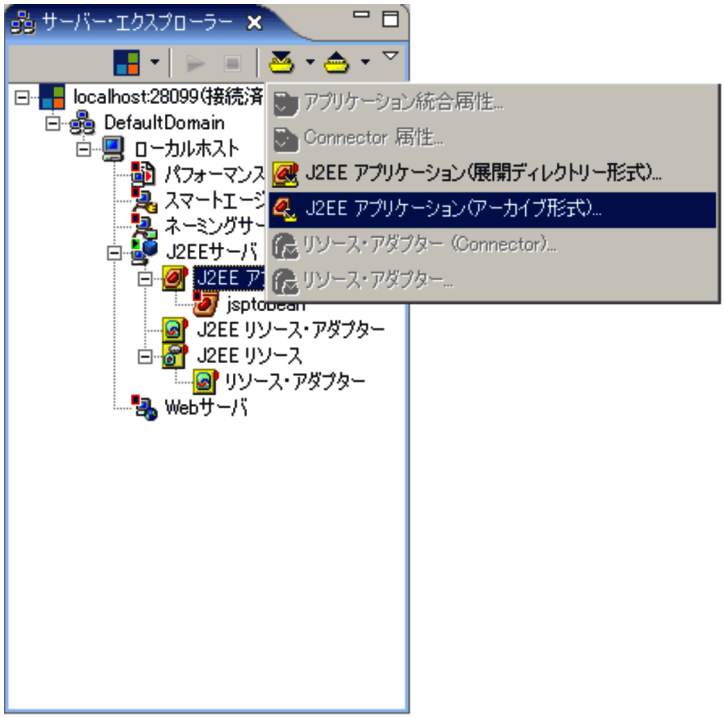
J2EE アプリケーションのディレクトリに，UNC 名を含むパスを指定した場合，メッセージダイアログが表示されます。メッセージダイアログで［OK］をクリックすると，上記の操作 3. に戻ります。

インポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.15　アプリケーション統合属性のインポート」を参照してください。

21.1.18　アーカイブ形式 J2EE アプリケーションのインポート

アーカイブ形式の J2EE アプリケーションをインポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-22　アーカイブ形式の J2EE アプリケーションをインポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，［J2EE アプリケーション］フォルダを選択します。**
2. **次のどちらかの方法で，［J2EE アプリケーション (アーカイブ形式) のインポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（インポート）のサブメニュー［ J2EE アプリケーション (アーカイブ形式)］を選択
   - 右クリックで表示されるコンテキストメニューの［ J2EE アプリケーション (アーカイブ形式) のインポート］を選択

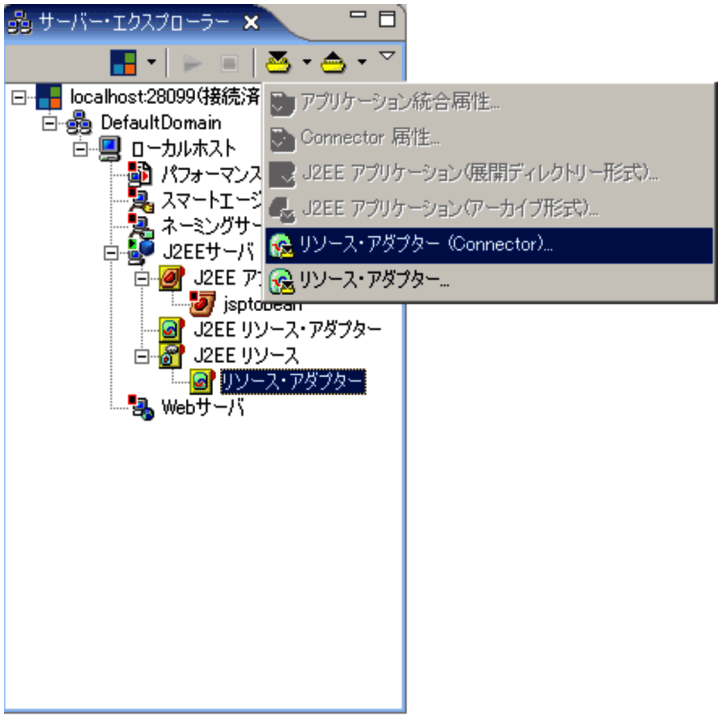
   ［J2EE アプリケーション (アーカイブ形式) のインポート］ダイアログが表示されます。
3. **［J2EE アプリケーション (アーカイブ形式) のインポート］ダイアログで，アプリケーションのアーカイブファイル (ZIP，または EAR) を選択します。**
4. **［J2EE アプリケーション (アーカイブ形式) のインポート］ダイアログで［開く］を選択します。**
   アーカイブ形式の J2EE アプリケーションのインポートが開始されます。

インポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.15　アプリケーション統合属性のインポート」を参照してください。

21.1.19　リソースアダプタ（Connector）のインポート

J2EE サーバが稼働するホスト（接続先ホスト）にあるリソースアダプタをインポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-23　接続先ホストにあるリソースアダプタをインポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，［リソース・アダプター］フォルダを選択します。**
2. **次のどちらかの方法で，［リソース・アダプター** (Connector) **のインポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（インポート）のサブメニュー［ リソース・アダプター (Connector)］を選択
   - 右クリックで表示されるコンテキストメニューの［ リソース・アダプター (Connector) のインポート］を選択

   ［リソース・アダプター (Connector) のインポート］ダイアログが表示されます。

図 21-24　［リソース・アダプター (Connector) のインポート］ダイアログ

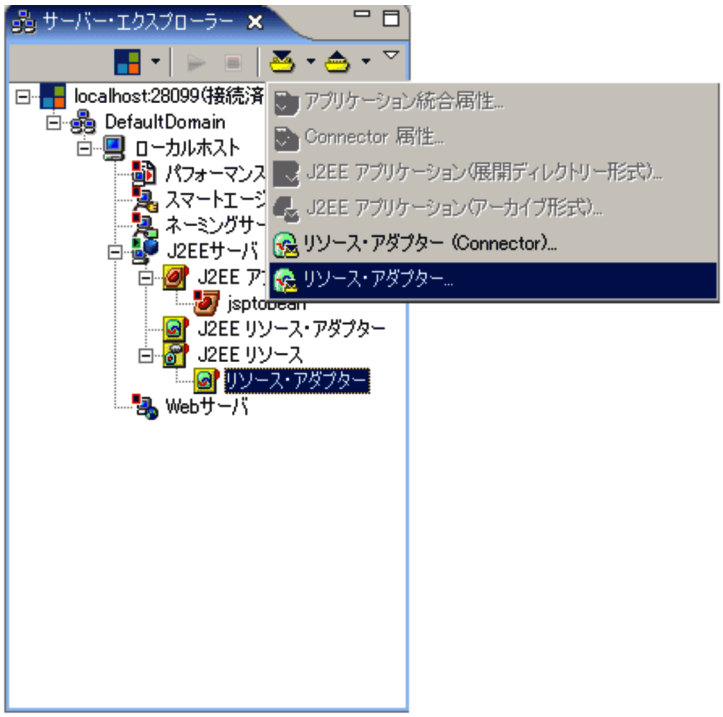
3. **［リソース・アダプター (Connector) のインポート］ダイアログで，接続先ホストにある RAR ファイルを指定します。**
   ［ファイル・パス］に RAR ファイルを直接入力するか，またはドロップダウンリストから選択します。
   RAR ファイルを直接入力する場合，格納ディレクトリを含めた絶対パスで指定してください。ドロップダウンリストに表示されている RAR ファイルが指定できます。
   なお，「${cosminexus.home}」は，Cosminexus インストールディレクトリを示します。
4. **［リソース・アダプター (Connector) のインポート］ダイアログで，［OK］を選択します。**
   接続先ホストのリソースアダプタのインポートが開始されます。

［ファイル・パス］で指定した値に誤りがある場合，［リソース・アダプター (Connector) のインポート］ダイアログのタイトル行にエラーメッセージが表示されます。

21.1.20　リソースアダプタのインポート

Server Plug-in 環境にあるリソースアダプタをインポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-25　リソースアダプタをインポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，［リソース・アダプター］フォルダを選択します。**
2. **次のどちらかの方法で，［リソース・アダプターのインポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（インポート）のサブメニュー［ リソース・アダプター］を選択
   - 右クリックで表示されるコンテキストメニューの［ リソース・アダプターのインポート］を選択

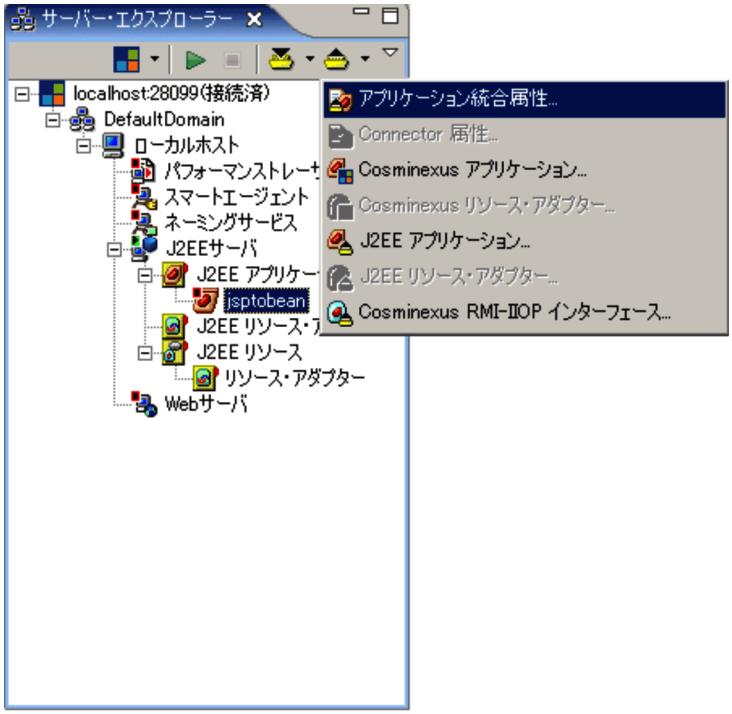
   ［リソース・アダプターのインポート］ダイアログが表示されます。
3. **［リソース・アダプターのインポート］ダイアログで，インポートするリソースアダプタを選択します。**
4. **［リソース・アダプターのインポート］ダイアログで［開く］を選択します。**
   Server Plug-in 環境のリソースアダプタのインポートが開始されます。

インポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.15　アプリケーション統合属性のインポート」を参照してください。

21.1.21　アプリケーション統合属性のエクスポート

J2EE アプリケーションの属性をアプリケーション統合属性ファイルとして出力する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-26　J2EE アプリケーションの属性をエクスポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，属性を出力する J2EE アプリケーションを選択します。**
2. **次のどちらかの方法で，［アプリケーション統合属性のエクスポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（エクスポート）のサブメニュー［ アプリケーション統合属性］を選択
   - 右クリックで表示されるコンテキストメニューの［ アプリケーション統合属性のエクスポート］を選択

   ［アプリケーション統合属性のエクスポート］ダイアログが表示されます。
3. **［アプリケーション統合属性のエクスポート］ダイアログで，出力するアプリケーション統合属性のファイル名を指定します。**
4. **［アプリケーション統合属性のエクスポート］ダイアログで，［保存］ボタンをクリックします。**
   アプリケーション統合属性のエクスポートが開始されます。

エクスポートダイアログで指定したファイルに，次のエラーがある場合，メッセージダイアログが表示されます。

- エクスポートダイアログで指定したファイルが，すでに存在する
- エクスポートダイアログで指定したファイルに書き込み権限がない
  エクスポートダイアログで指定したファイルに書き込み権限がない場合，［詳細］ボタンをクリックすると，発生した例外情報が表示されます。

［OK］ボタンまたは「はい」ボタンをクリックすると，アプリケーション統合属性のエクスポート操作の 3. に戻ります。

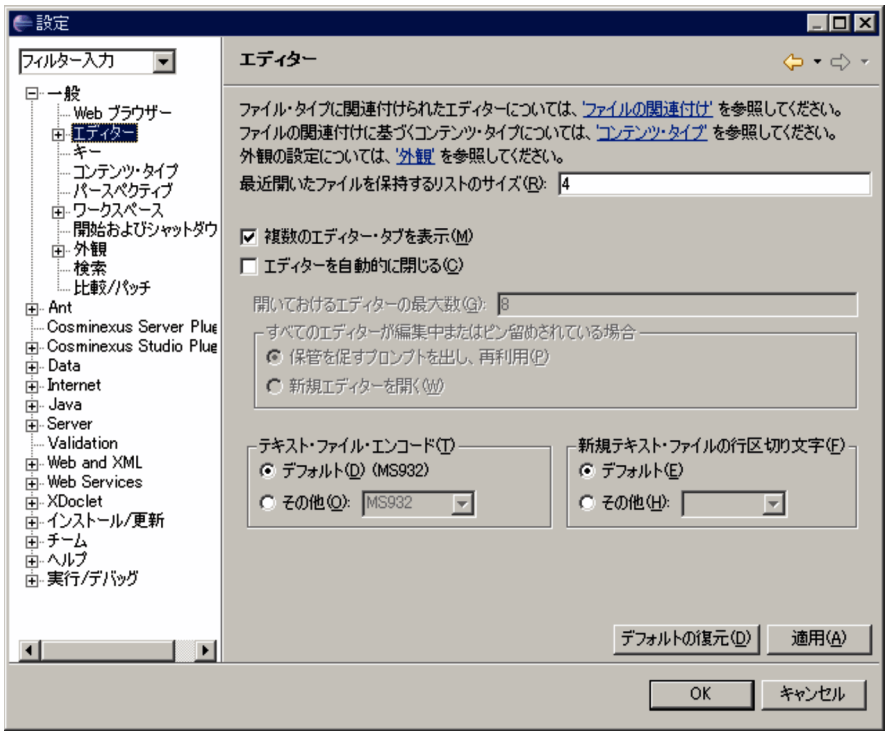
注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

参考

属性ファイルのエンコードは，Eclipse の環境設定の，［エディター］で設定します。設定画面を次に示します。

図 21-27　Eclipse のエンコード設定画面



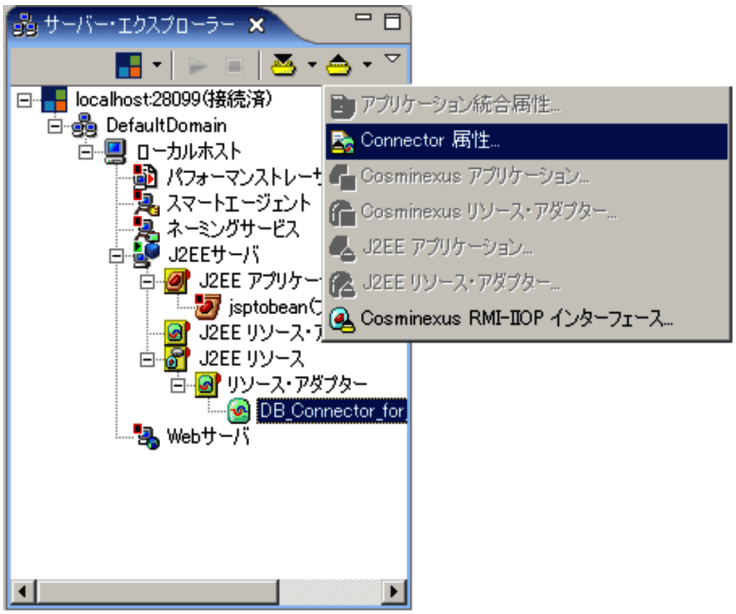
Eclipse のエンコード設定は，次の手順で表示します。

1. メニューから［ウィンドウ］-［設定］を選択します。
   ［設定］ダイアログが表示されます。
2. ［設定］ダイアログの左ペインにあるツリービューで，［一般］-［エディター］を選択します。
   ［エディター］ページが表示されます。
3. ［エディター］ページの［テキスト・ファイル・エンコード］から，エクスポートする属性ファイルのエンコードを選択します。

21.1.22　Connector 属性のエクスポート

リソースアダプタの属性を，Connector 属性ファイルに取得する場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-28　リソースアダプタの属性をエクスポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，属性を取得するリソースアダプタを選択します。**
2. **次のどちらかの方法で，［Connector 属性のエクスポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（エクスポート）のサブメニュー［ Connector 属性］を選択
   - 右クリックで表示されるコンテキストメニューの［ Connector 属性のエクスポート］を選択

   ［Connector 属性のエクスポート］ダイアログが表示されます。
3. **［Connector 属性のエクスポート］ダイアログで，Connector 属性を出力するファイル名を指定します。**
4. **［Connector 属性のエクスポート］ダイアログで，［保存］ボタンをクリックします。**
   Connector 属性のエクスポートが開始されます。

エクスポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.21　アプリケーション統合属性のエクスポート」を参照してください。

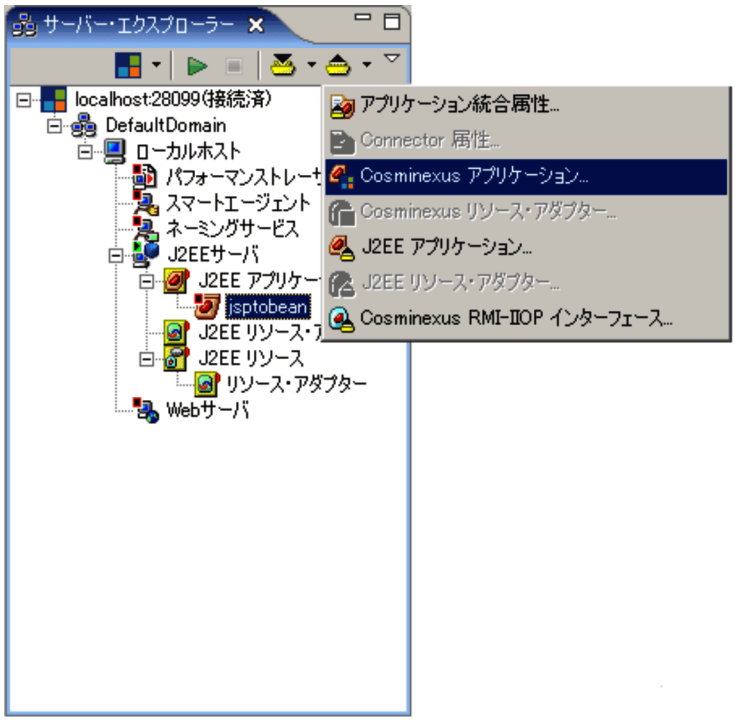
参考

属性ファイルのエンコードは，Eclipse の環境設定の，［エディター］で設定します。設定画面については，「21.1.21　アプリケーション統合属性のエクスポート」を参照してください。

21.1.23　Cosminexus アプリケーションのエクスポート

実行時情報を含んだ J2EE アプリケーションをエクスポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-29　実行時情報を含んだ J2EE アプリケーションをエクスポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，エクスポートする J2EE アプリケーションを選択します。**
2. **次のどちらかの方法で，［Cosminexus アプリケーションのエクスポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（エクスポート）のサブメニュー［ Cosminexus アプリケーション］を選択
   - 右クリックで表示されるコンテキストメニューの［ Cosminexus アプリケーションのエクスポート］を選択

［Cosminexus アプリケーションのエクスポート］ダイアログが表示されます。

3. **［Cosminexus アプリケーションのエクスポート］ダイアログで，エクスポートする J2EE アプリケーションのアーカイブファイルを選択します。**
4. **［Cosminexus アプリケーションのエクスポート］ダイアログで，［保存］ボタンをクリックします。**
   J2EE アプリケーションのエクスポートが開始されます。

エクスポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.21　アプリケーション統合属性のエクスポート」を参照してください。

21.1.24　Cosminexus RMI-IIOP インタフェースのエクスポート

［Cosminexus RMI-IIOP インタフェース］ページで，J2EE アプリケーションの RMI-IIOP インタフェースをエクスポートします。［Cosminexus RMI-IIOP インタフェース］ページの表示と操作について説明します。

（1）［Cosminexus RMI-IIOP インタフェース］ページの表示

［Cosminexus RMI-IIOP インタフェース］ページは，次のどちらかの方法で表示します。

- ［エクスポート］ダイアログから表示
- ［サーバー・エクスプローラー］ビューから表示

●［エクスポート］ダイアログから表示

［エクスポート］ダイアログを次に示します。

図 21-30 ［エクスポート］ダイアログ

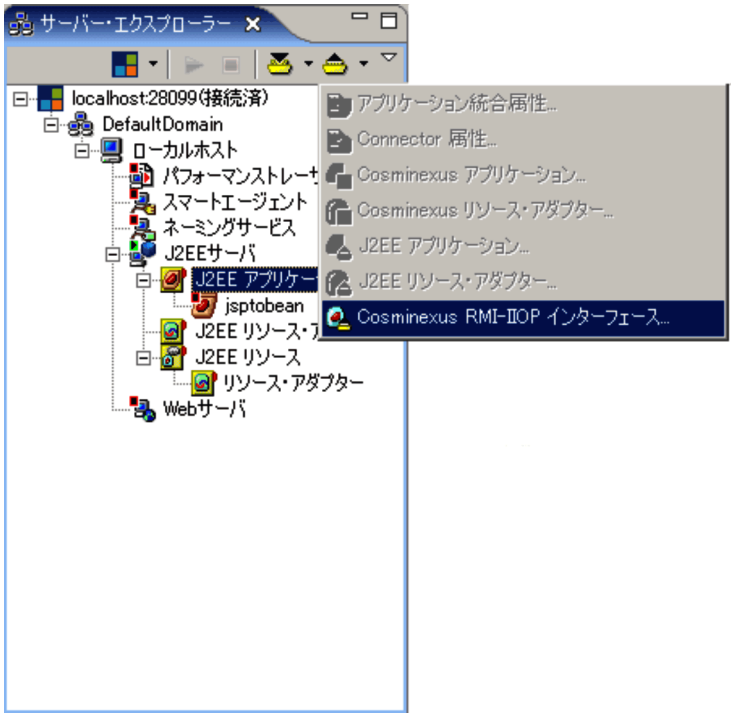
［Cosminexus RMI-IIOP インタフェース］ページは次の手順で表示します。

1. **メニューから［ファイル］-［エクスポート］を選択します。**
   ［エクスポート］ダイアログが表示されます。
2. **［エクスポート］ダイアログの［エクスポート先の選択］の［Cosminexus RMI-IIOP インターフェース］を選択して，［次へ］をクリックします。**
   ［Cosminexus RMI-IIOP インタフェース］ページが表示されます。

● ［サーバー・エクスプローラー］ビューから表示

［サーバー・エクスプローラー］ビューを次に示します。

図 21-31　［Cosminexus RMI-IIOP インタフェース］ページを表示する場合の［サーバー・エクスプローラー］ビュー



［Cosminexus RMI-IIOP インタフェース］ページは次の手順で表示します。

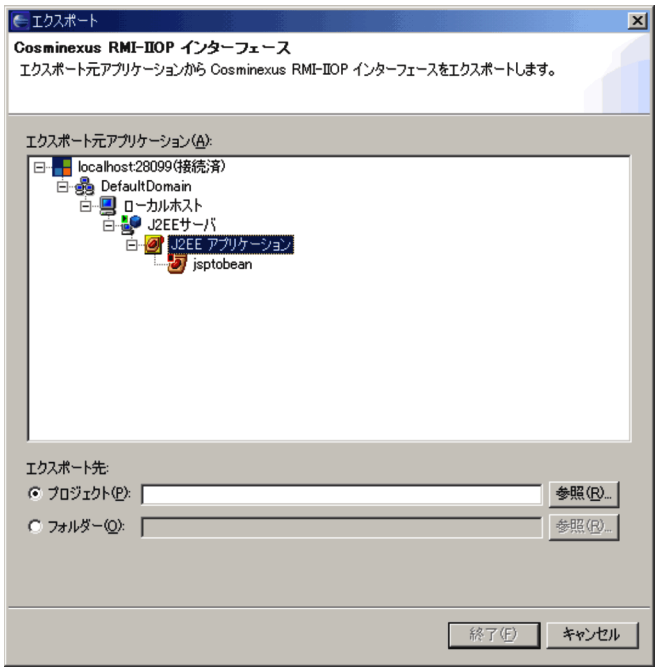
1. **ツリービューで，RMI-IIOP インタフェースをエクスポートする J2EE アプリケーションを選択します。**
2. **次のどちらかの方法で，［Cosminexus RMI-IIOP インタフェース］ページを表示させます。**
   - ビューツールバーの［ ］（エクスポート）のサブメニュー［ Cosminexus RMI-IIOP インターフェース］を選択
   - 右クリックで表示されるコンテキストメニューの［ Cosminexus RMI-IIOP インターフェースのエクスポート］を選択

   ［Cosminexus RMI-IIOP インタフェース］ページが表示されます。

(2) ［Cosminexus RMI-IIOP インタフェース］ページの操作

［Cosminexus RMI-IIOP インタフェース］ページを次に示します。

図 21-32 ［Cosminexus RMI-IIOP インタフェース］ページ



1. **［エクスポート元アプリケーション］のツリービューから RMI-IIOP インタフェースをエクスポートする J2EE アプリケーションを選択します。**
   J2EE アプリケーションが表示されていない場合，J2EE サーバのノードを展開して，J2EE アプリケーションを表示します。表示処理中はプログレスバーが表示され，キャンセルできません。
2. **RMI-IIOP インタフェースをプロジェクトにエクスポートする場合は，［プロジェクト］を選択します。**
   RMI-IIOP インタフェースのエクスポート先プロジェクトを次のように指定します。
   - ワークスペースからの相対パスで指定する。
   - パスのセパレータは，「¥」または「/」で指定する。
   - 大文字・小文字の区別はしない。
   - 入力値の長さに制限はない。

   ［参照］をクリックすると，［プロジェクトの参照］ダイアログが表示されます。［プロジェクトの参照］ダイアログには，ワークスペース内のプロジェクトおよびプロジェクト以下のフォルダが表示されています。RMI-IIOP インタフェースのエクスポート先プロジェクトを選択して，［OK］をクリックしてください。
3. **RMI-IIOP インタフェースをフォルダにエクスポートする場合は，［フォルダー］を選択します。**
   RMI-IIOP インタフェースのエクスポート先フォルダを次のように指定します。
   - ファイルシステムの絶対パスで指定する。

- パスのセパレータは，「¥」または「/」で指定する。
- 大文字・小文字の区別はしない。
- 最大文字数は 248 文字とする。

［参照］をクリックすると，［フォルダーの参照］ダイアログが表示されます。［フォルダーの参照］ダイアログで，RMI-IIOP インタフェースのエクスポート先フォルダを指定して，［OK］をクリックしてください。

4. **［終了］をクリックします。**
［エクスポート元アプリケーション］で J2EE アプリケーションが選択されている場合に活性となり，クリックできます。
RMI-IIOP インタフェースのエクスポートが開始されます。

［プロジェクト］または［フォルダー］で指定した値に誤りがある場合，［Cosminexus RMI-IIOP インタフェース］ページのタイトル行に，エラーメッセージが表示されます。

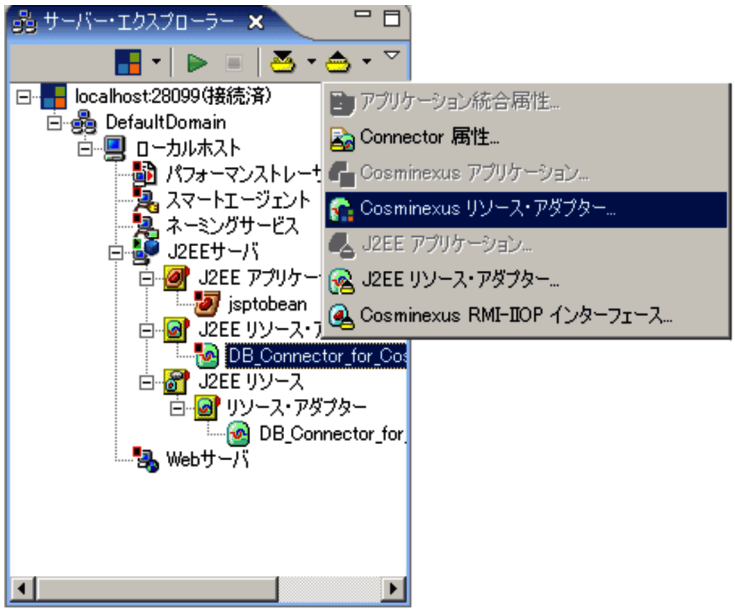
注意事項

Windows 7 または Windows Vista で管理者特権のないユーザが，エクスポート先に，C:¥Program Files 以下のディレクトリを指定した場合の対応については，「11.4　Server Plug-in の注意事項」の「(1)　Windows 7 または Windows Vista 使用時の注意事項」を参照してください。

21.1.25　Cosminexus リソースアダプタのエクスポート

実行時情報を含んだリソースアダプタをエクスポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-33　実行時情報を含んだリソースアダプタをエクスポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，エクスポートするリソースアダプタを選択します。**
2. **次のどちらかの方法で［Cosminexus リソース・アダプターのエクスポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（エクスポート）のサブメニュー［ Cosminexus リソース・アダプター］を選択
   - 右クリックで表示されるコンテキストメニューの［ Cosminexus リソース・アダプターのエクスポート］を選択

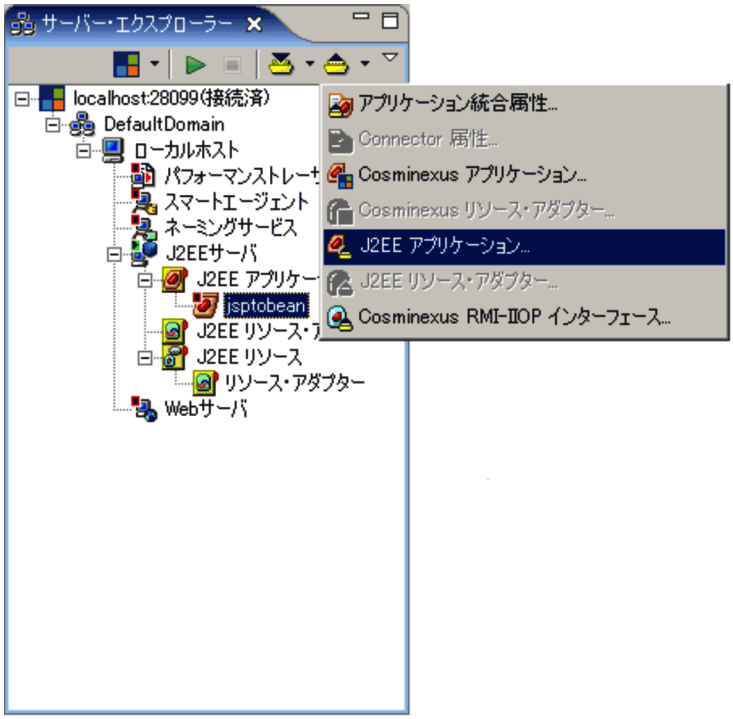
   ［Cosminexus リソース・アダプターのエクスポート］ダイアログが表示されます。
3. **［Cosminexus リソース・アダプターのエクスポート］ダイアログで，エクスポートするリソースアダプタ（RAR ファイル）を選択します。**
4. **［Cosminexus リソース・アダプターのエクスポート］ダイアログで，［保存］ボタンをクリックします。**
   リソースアダプタのエクスポートが開始されます。

エクスポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.21　アプリケーション統合属性のエクスポート」を参照してください。

21.1.26　J2EE アプリケーションのエクスポート

J2EE アプリケーションをエクスポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-34　J2EE アプリケーションをエクスポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，エクスポートする J2EE アプリケーションを選択します。**
2. **次のどちらかの方法で，［J2EE アプリケーションのエクスポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（エクスポート）のサブメニュー［ J2EE アプリケーション］を選択
   - 右クリックで表示されるコンテキストメニューの［ J2EE アプリケーションのエクスポート］を選択

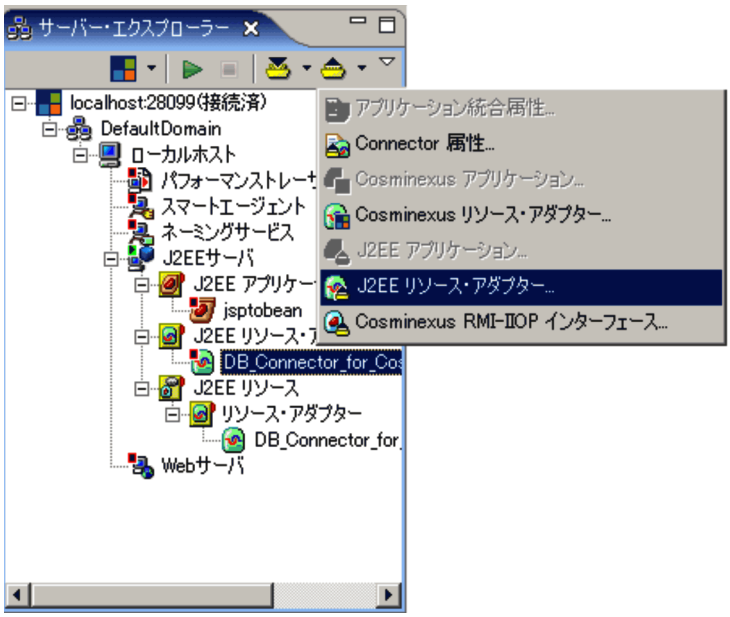
   ［J2EE アプリケーションのエクスポート］ダイアログが表示されます。
3. **［J2EE アプリケーションのエクスポート］ダイアログで，エクスポートする J2EE アプリケーションのアーカイブファイルを選択します。**
4. **［J2EE アプリケーションのエクスポート］ダイアログで，［保存］ボタンをクリックします。**
   J2EE アプリケーションのエクスポートが開始されます。

エクスポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.21　アプリケーション統合属性のエクスポート」を参照してください。

21.1.27　J2EE リソースアダプタのエクスポート

J2EE リソースアダプタをエクスポートする場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-35　J2EE リソースアダプタをエクスポートする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，エクスポートする J2EE リソースアダプタを選択します。**
2. **次のどちらかの方法で［J2EE リソース・アダプターのエクスポート］ダイアログを表示させます。**
   - ビューツールバーの［ ］（エクスポート）のサブメニュー［ J2EE リソース・アダプター］を選択
   - 右クリックで表示されるコンテキストメニューの［ J2EE リソース・アダプターのエクスポート］を選択

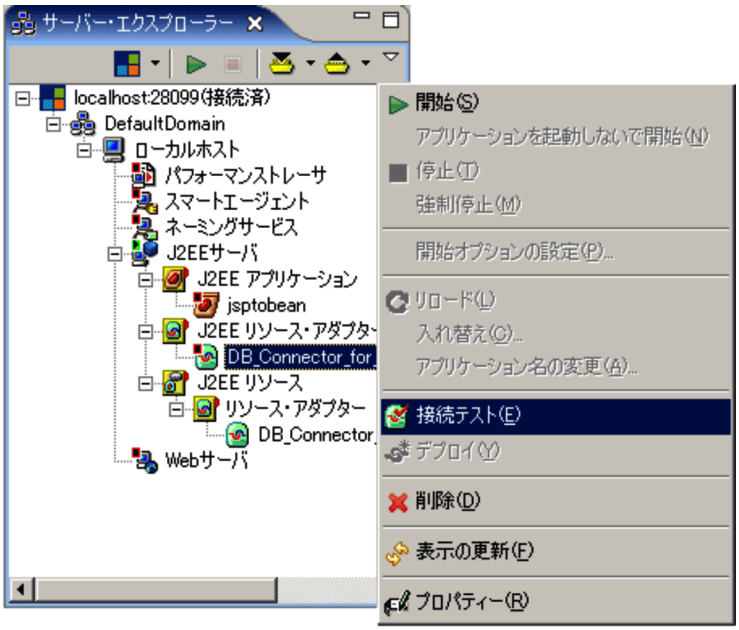
   ［J2EE リソース・アダプターのエクスポート］ダイアログが表示されます。
3. **［J2EE リソース・アダプターのエクスポート］ダイアログで，エクスポートするリソースアダプタ（RAR ファイル）を選択します。**
4. **［J2EE リソース・アダプターのエクスポート］ダイアログで，［保存］ボタンをクリックします。**
   リソースアダプタのエクスポートが開始されます。

エクスポートダイアログで指定したファイルにエラーがある場合の対応については，「21.1.21　アプリケーション統合属性のエクスポート」を参照してください。

21.1.28　接続テスト

J2EE リソースアダプタの接続テストを行う場合の［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-36　リソースアダプタの接続テストを行う場合の［サーバー・エクスプローラー］ビュー

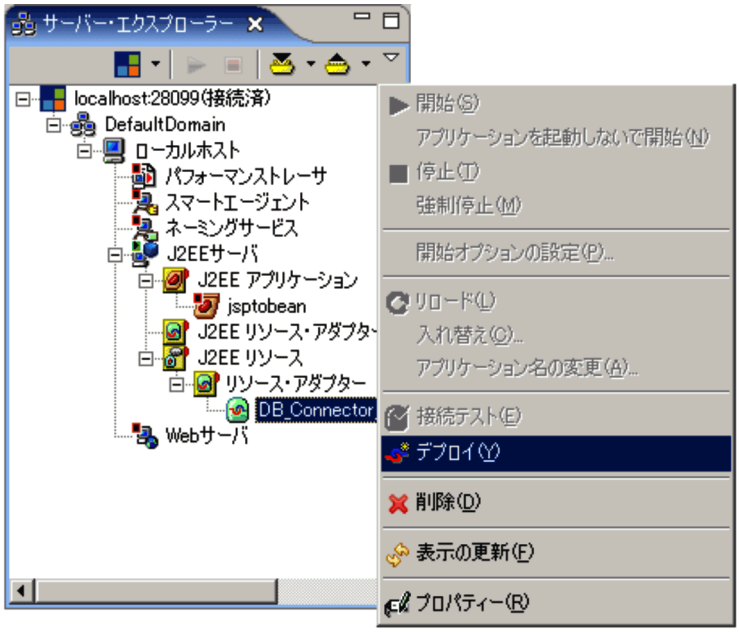


1. **ツリービューで，接続テストを行う J2EE リソースアダプタを選択します。**
2. **右クリックで表示されるコンテキストメニューの［接続テスト］を選択して，接続テストの操作を実行します。**
   接続テストが開始されます。
   接続テストに成功すると，接続テスト成功のメッセージダイアログが表示されます。

21.1.29　リソースアダプタのデプロイ

リソースアダプタを J2EE リソースアダプタとしてデプロイする場合の，［サーバー・エクスプローラー］ビューと操作手順について説明します。

図 21-37　リソースアダプタを J2EE リソースアダプタとしてデプロイする場合の［サーバー・エクスプローラー］ビュー



1. **ツリービューで，デプロイするリソースアダプタを選択します。**
2. **右クリックで表示されるコンテキストメニューの［ デプロイ］を選択して，デプロイの操作を実行します。**

21.2　リソースアダプタの属性編集画面

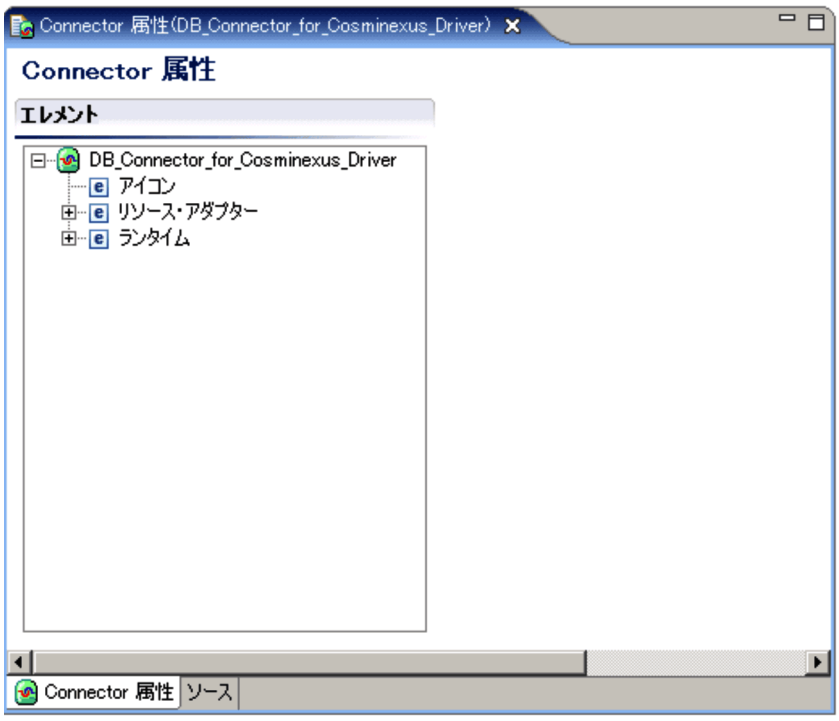
ここでは，リソースアダプタの属性編集画面について説明します。リソースアダプタの属性編集画面は，次の手順で表示します。

1. **［サーバー・エクスプローラー］ビューで，属性を編集するリソースアダプタのノードを選択します。**
   リソースアダプタの属性を編集する場合，次のノードを選択してください。
   - J2EE アプリケーションに含まれるリソースアダプタの場合
     ［J2EE アプリケーション］フォルダ - ［＜ J2EE アプリケーション名＞］内の［＜リソースアダプタ＞］
   - J2EE リソースアダプタとしてデプロイされるリソースアダプタの場合
     ［J2EE リソース・アダプター］フォルダ内の J2EE リソースアダプタ
   - J2EE リソースアダプタとしてデプロイしていないリソースアダプタの場合
     ［J2EE リソース］-［リソース・アダプター］フォルダ内のリソースアダプタ

2. **属性ファイル編集エディタを起動します。**
   次のどれかの方法で，属性ファイル編集エディタを起動します。
   - ツリービューで選択したノードをダブルクリック
   - ビュープルダウンメニューの［ プロパティー］を選択
   - 右クリックで表示されるコンテキストメニューの［ プロパティー］を選択

   属性ファイル編集エディタが起動されると，パースペクティブのエディタエリアに，Connector 属性のフォームページが表示されます。

図 21-38　Connector 属性のフォームページ例

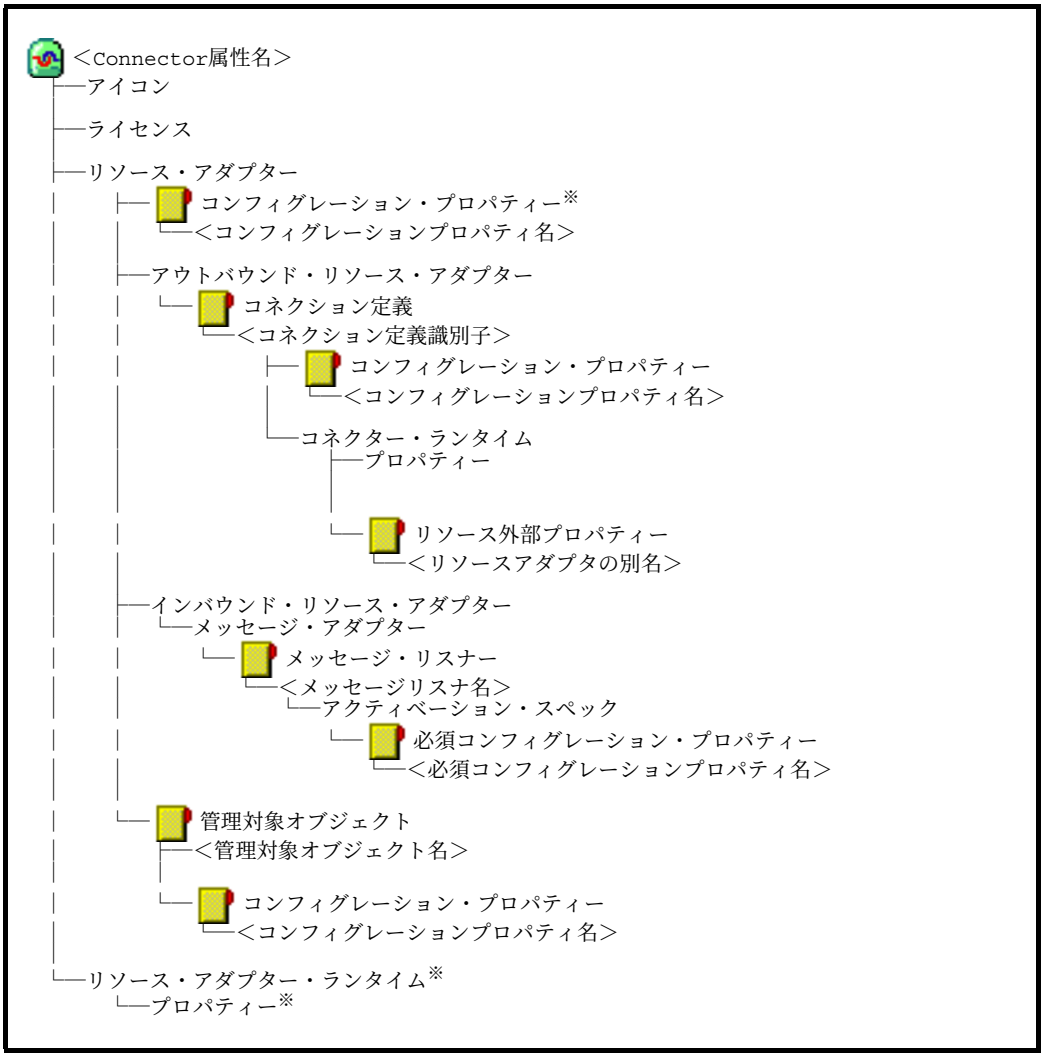


フォームページの［エレメント］の下に，Connector 属性の構成のツリーが展開されます。

21.2.1　Connector 属性のツリー

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。エディタエリアの［エレメント］の下に，Connector 属性の構成がツリー形式で表示されます。

Connector 属性のツリー構成を次に示します。



注※　このノード以下のノードは，Connector 1.5 の仕様に準拠するリソースアダプタの場合に指定できます。

Connector 属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。

次に，Connector 属性のツリーノードとコンテキストメニューについて説明します。

（1） Connector 属性のツリーノード

Connector 属性のツリービューに表示されるノードを次の表に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。

表 21-13 Connector 属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜Connector 属性名＞	Connector 属性を示すノードです。選択すると，Connector 属性のプロパティページが表示されます。	<hitachi-connector-property>	21.2.2 (1)
アイコン	アイコンを示すノードです。選択すると，アイコンのプロパティページが表示されます。	<icon>	21.4.2 (2)
ライセンス	ライセンスを示すノードです。選択すると，ライセンスのプロパティページが表示されます。	<license>	21.2.2 (2)
リソース・アダプター	リソースアダプタを示すノードです。選択すると，リソースアダプタのプロパティページが表示されます。	<resourceadapter>	21.2.2 (3)
コンフィグレーション・プロパティー（リソース・アダプター）	ResourceAdapter 実装クラスのインスタンスに対するコンフィグレーションプロパティをまとめるノードです。選択すると，コンフィグレーションプロパティの一覧ページが表示されます。	―	21.2.2 (7)
＜コンフィグレーションプロパティ名＞（リソース・アダプター）	ResourceAdapter 実装クラスのインスタンスに対するコンフィグレーションプロパティを示すノードです。選択すると，コンフィグレーションプロパティのプロパティページが表示されます。	<resourceadapter> - <config-property>	21.2.2 (8), 21.2.2 (9)
アウトバウンド・リソース・アダプター	Outbound リソースアダプタのプロパティを示すノードです。選択すると，Outbound リソースアダプタのプロパティページが表示されます。	<resourceadapter> - <outbound-resourceadapter>	21.2.2 (4)
コネクション定義	Outbound リソースアダプタのコネクション定義をまとめるノードです。選択すると，コネクション定義の一覧ページが表示されます。	―	21.2.2 (5)
＜コネクション定義識別子＞	コネクション定義を示すノードです。選択すると，コネクション定義のプロパティページが表示されます。	<resourceadapter> - <outbound-resourceadapter> - <connection-definition>	21.2.2 (6)
コンフィグレーション・プロパティー（コネクション定義）	Outbound リソースアダプタに対するコンフィグレーションプロパティをまとめるノードです。選択すると，コンフィグレーションプロパティの一覧ページが表示されます。	―	21.2.2 (7)

ノード	説明	対応するタグ名	プロパティページ
＜コンフィグレーションプロパティ名＞（コネクション定義）	Outbound リソースアダプタに対するコンフィグレーションプロパティを示すノードです。選択すると，コンフィグレーションプロパティのプロパティページが表示されます。	<resourceadapter> - <outbound-resourceadapter> - <connection-definition> - <config-property>	21.2.2 (8), 21.2.2 (9)
コネクター・ランタイム	Outbound リソースアダプタに対するランタイムのプロパティ，およびリソースの外部プロパティをまとめるノードです。	<resourceadapter> - <outbound-resourceadapter> - <connection-definition> -<connector-runtime>	—
プロパティー（コネクション定義）	Outbound リソースアダプタに対するランタイムのプロパティを示すノードです。選択すると，プロパティのプロパティページが表示されます。	<resourceadapter> - <outbound-resourceadapter> - <connection-definition> - <connector-runtime> - <property>	21.2.2 (10)
リソース外部プロパティー	Outbound リソースアダプタに対するリソース外部プロパティをまとめるノードです。選択すると，リソース外部プロパティの一覧ページが表示されます。	—	21.2.2 (11)
＜リソースアダプタの別名＞	Outbound リソースアダプタに対するリソースアダプタの別名を示すノードです。選択すると，リソース外部プロパティのプロパティページが表示されます。	<resourceadapter> - <outbound-resourceadapter> - <connection-definition> - <resource-external-property>	21.2.2 (12)
インバウンド・リソース・アダプター	Inbound リソースアダプタのプロパティを示すノードです。	<resourceadapter> - <inbound-resourceadapter>	—
メッセージ・アダプター	Inbound リソースアダプタに対するメッセージアダプタを示すノードです。	<resourceadapter> - <inbound-resourceadapter> - <messageadapter>	—
メッセージ・リスナー	Inbound リソースアダプタに対するメッセージリスナをまとめるノードです。選択すると，メッセージリスナの一覧ページが表示されます。	—	21.2.2 (13)
＜メッセージリスナ名＞	Inbound リソースアダプタに対するメッセージリスナを示すノードです。選択すると，メッセージリスナのプロパティページが表示されます。	<resourceadapter> - <inbound-resourceadapter> - <messageadapter> - <messagelistener> - <messagelistener-type>	21.2.2 (14)

ノード	説明	対応するタグ名	プロパティページ
アクティベーション・スペック	Inbound リソースアダプタに対する ActivationSpec を示すノードです。選択すると，ActivationSpec のプロパティページが表示されます。	<resourceadapter> - <inbound-resourceadapter> - <messageadapter> - <messagelistener> - <activationspec>	21.2.2 (15)
必須コンフィグレーション・プロパティー	Inbound リソースアダプタに対する必須コンフィグレーションプロパティをまとめるノードです。選択すると，必須コンフィグレーションプロパティの一覧ページが表示されます。	—	21.2.2 (16)
＜必須コンフィグレーションプロパティ名＞	Inbound リソースアダプタに対する必須コンフィグレーションプロパティを示すノードです。選択すると，必須コンフィグレーションプロパティのプロパティページが表示されます。	<resourceadapter> - <inbound-resourceadapter> - <messageadapter> - <messagelistener> - <activationspec> - <required-config-property> - <config-property-name>	21.2.2 (17)
管理対象オブジェクト	リソースアダプタに対する管理対象オブジェクトをまとめるノードです。選択すると，管理対象オブジェクトの一覧ページが表示されます。	—	21.2.2 (18)
＜管理対象オブジェクト名＞	リソースアダプタに対する管理対象オブジェクトを示すノードです。選択すると，管理対象オブジェクトのプロパティページが表示されます。	<resourceadapter> - <adminobject> - <adminobject-name>	21.2.2 (19)
コンフィグレーション・プロパティー（管理対象オブジェクト）	リソースアダプタに対する管理対象オブジェクトのコンフィグレーションプロパティをまとめるノードです。選択すると，コンフィグレーションプロパティの一覧ページが表示されます。	—	21.2.2 (7)
＜コンフィグレーションプロパティ名＞	リソースアダプタに対する管理対象オブジェクトのコンフィグレーションプロパティを示すノードです。選択すると，コンフィグレーションプロパティのプロパティページが表示されます。	<resourceadapter> - <adminobject> - <config-property> - <config-property-name>	21.2.2 (8), 21.2.2 (9)
リソース・アダプター・ランタイム	リソースアダプタに対するランタイムプロパティをまとめるノードです。	<resourceadapter-runtime>	—
プロパティー（リソース・アダプター）	リソースアダプタに対するランタイムプロパティを示すノードです。選択すると，プロパティのプロパティページが表示されます。	<resourceadapter-runtime> - <property>	21.2.2 (20)

（凡例）　－：該当なし

(2) コンテキストメニュー

Connector 属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-14　Connector 属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜ Connector 属性名＞	［追加］-［アイコン］	Connector 属性にアイコンが追加されます。アイコンのプロパティページが表示されます。すでにアイコンがある場合は選択できません。	21.4.2　(2)
アイコン	［削除］	Connector 属性からアイコンが削除されます。	－
リソース・アダプター	［追加］-［コンフィグレーション・プロパティー］	リソースアダプタにコンフィグレーションプロパティが追加されます。コンフィグレーションプロパティのプロパティページが表示されます。	21.2.2　(8), 21.2.2　(9)
	［追加］-［管理対象オブジェクト］	リソースアダプタに管理対象オブジェクトが追加されます。管理対象オブジェクトの一覧ページが表示されます。	21.2.2　(18)
コンフィグレーション・プロパティー（リソース・アダプター）	［追加］-［コンフィグレーション・プロパティー］	コンフィグレーションプロパティが追加されます。コンフィグレーションプロパティのプロパティページが表示されます。	21.2.2　(8), 21.2.2　(9)
	［削除］	リソースアダプタから，すべてのコンフィグレーションプロパティが削除されます。	－
＜コンフィグレーションプロパティ名＞（リソース・アダプター）	［削除］	リソースアダプタから，選択したコンフィグレーションプロパティが削除されます。	－
＜コネクション定義識別子＞	［追加］-［コンフィグレーション・プロパティー］	コンフィグレーションプロパティ（コネクション定義）が追加されます。コンフィグレーションプロパティのプロパティページが表示されます。	21.2.2　(8), 21.2.2　(9)
コンフィグレーション・プロパティー（コネクション定義）	［追加］-［コンフィグレーション・プロパティー］	コンフィグレーションプロパティ（コネクション定義）が追加されます。コンフィグレーションプロパティのプロパティページが表示されます。	21.2.2　(8), 21.2.2　(9)
	［削除］	＜コネクション定義識別子＞からすべてのコンフィグレーションプロパティが削除されます。	－

ノード	コンテキストメニュー	説明	プロパティページ
＜コンフィグレーションプロパティ名＞（コネクション定義）	［削除］	＜コネクション定義識別子＞から選択したコンフィグレーションプロパティが削除されます。	－
コネクター・ランタイム	［追加］-［リソース外部プロパティー］	リソースアダプタの別名が追加されます。リソース外部プロパティのプロパティページが表示されます。	21.2.2 (12)
リソース外部プロパティー	［追加］-［リソース外部プロパティー］	リソースアダプタの別名が追加されます。リソース外部プロパティのプロパティページが表示されます。	21.2.2 (12)
	［削除］	ランタイムから，すべてのリソースアダプタの別名が削除されます。	－
＜リソースアダプタの別名＞	［削除］	ランタイムから，選択したリソースアダプタの別名が削除されます。	－
管理対象オブジェクト	［追加］-［管理対象オブジェクト］	リソースアダプタに管理対象オブジェクトが追加されます。管理対象オブジェクトのプロパティページが表示されます。	21.2.2 (19)
	［削除］	リソースアダプタからすべての管理対象オブジェクトが削除されます。	－
＜管理対象オブジェクト名＞	［追加］-［コンフィグレーション・プロパティー］	管理対象オブジェクトのコンフィグレーションプロパティ（管理対象オブジェクト）が追加されます。コンフィグレーションプロパティのプロパティページが表示されます。	21.2.2 (8), 21.2.2 (9)
	［削除］	リソースアダプタから選択した管理対象オブジェクトが削除されます。	－
コンフィグレーション・プロパティー（管理対象オブジェクト）	［追加］-［コンフィグレーション・プロパティー］	管理対象オブジェクトのコンフィグレーションプロパティ（管理対象オブジェクト）が追加されます。コンフィグレーションプロパティのプロパティページが表示されます。	21.2.2 (8), 21.2.2 (9)
	［削除］	管理対象オブジェクトからすべてのコンフィグレーションプロパティ（管理対象オブジェクト）が削除されます。	－
＜コンフィグレーションプロパティ名＞	［削除］	管理対象オブジェクトから選択したコンフィグレーションプロパティ（管理対象オブジェクト）が削除されます。	－

（凡例）　－：該当なし

21.2.2　Connector 属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に，編集する Connector 属性のプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアに，選択したノードに対応するプロパティページが表示されます。

Connector 属性には，次のプロパティページがあります。

- Connector 属性のプロパティページ
- ライセンスのプロパティページ
- リソースアダプタのプロパティページ
- Outbound リソースアダプタのプロパティページ
- コネクション定義の一覧ページ
- コネクション定義のプロパティページ
- コンフィグレーションプロパティの一覧ページ
- コンフィグレーションプロパティのプロパティページ（XAOpenString 以外）
- コンフィグレーションプロパティのプロパティページ（XAOpenString）
- プロパティのプロパティページ（Outbound リソースアダプタ）
- リソース外部プロパティの一覧ページ
- リソース外部プロパティのプロパティページ
- メッセージリスナの一覧ページ
- メッセージリスナのプロパティページ
- ActivationSpec のプロパティページ
- 必須コンフィグレーションプロパティの一覧ページ
- 必須コンフィグレーションプロパティのプロパティページ
- 管理対象オブジェクトの一覧ページ
- 管理対象オブジェクトのプロパティページ
- プロパティのプロパティページ（リソースアダプタ）

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

（1）Connector 属性のプロパティページ

Connector 属性のプロパティページを次に示します。

図 21-39 Connector 属性のプロパティページ

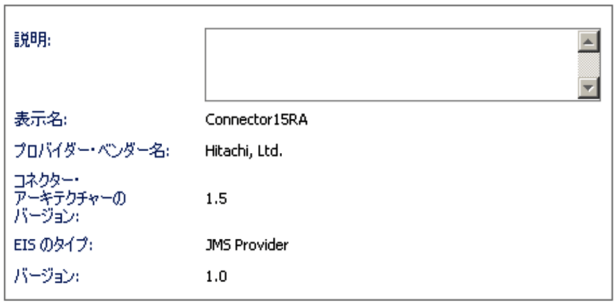


表 21-15 Connector 属性のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	リソースアダプタの説明を指定します。	<description> 説明 </description>
表示名	―	リソースアダプタ名が表示されます。	<display-name> 表示名 </display-name>
プロバイダー・ベンダー名	―	リソースアダプタのプロバイダベンダ名が表示されます。	<vender-name> プロバイダー・ベンダー名 </vender-name>
コネクター・アーキテクチャーのバージョン	―	リソースアダプタがサポートしているコネクタアーキテクチャ仕様のバージョンが表示されます。	<spec-version> コネクター・アーキテクチャーのバージョン </spec-version>
EIS のタイプ	―	EIS のタイプが表示されます。	<eis-type> EIS のタイプ </eis-type>
バージョン	―	リソースアダプタのバージョンが表示されます。	<version> バージョン </version>

（凡例） ×：任意 ―：変更不可

(2) ライセンスのプロパティページ

ライセンスのプロパティページを次に示します。

図 21-40 ライセンスのプロパティページ

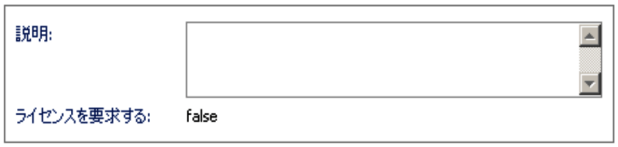


表 21-16　ライセンスのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	ライセンスの説明を指定します。	<description> 説明 </description>
ライセンスを要求する	－	リソースアダプタのデプロイ時にライセンスを要求する設定にしているかどうかが表示されます。 • true：ライセンスを要求する • false：ライセンスを要求しない	<license-required> ライセンスを要求する </license-required>

（凡例）　×：任意　－：変更不可

(3) リソースアダプタのプロパティページ

リソースアダプタのプロパティページを次に示します。

図 21-41　リソースアダプタのプロパティページ

表 21-17　リソースアダプタのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ResourceAdapter 実装クラス	－	javax.resource.spi.ResourceAdapter クラスを実装した Java クラス名が表示されます。	<resourceadapter-class> クラス名 </resourceadapter-class>

（凡例）　－：変更不可

(4) Outbound リソースアダプタのプロパティページ

Outbound リソースアダプタのプロパティページを次に示します。

図 21-42　Outbound リソースアダプタのプロパティページ

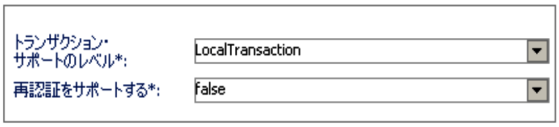


表 21-18　Outbound リソースアダプタのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
トランザクション・サポートのレベル	○	リソースアダプタで提供されるトランザクションサポートのレベルを指定します。 指定できる値を次に示します。 • NoTransaction • LocalTransaction • XATransaction	<transaction-support> トランザクションサポートのレベル </transaction-support>
再認証をサポートする	○	ManagedConnection インスタンスの再認証をサポートするかどうかを指定します。 指定できる値を次に示します。 • true：再認証をサポートする • false：再認証をサポートしない	<reauthentication-support> 再認証をサポートする </reauthentication-support>

（凡例）　○：必須

(5) コネクション定義の一覧ページ

コネクション定義の一覧ページには，設定されたコネクション定義の一覧が表示されます。

コネクション定義の一覧ページを次に示します。

図 21-43　コネクション定義の一覧ページ

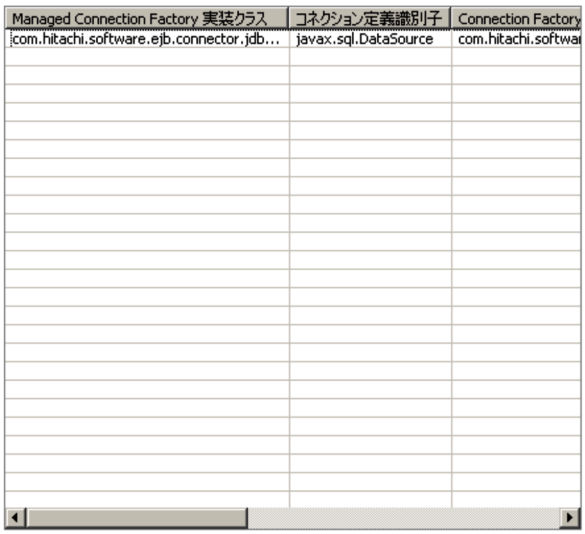


表 21-19　コネクション定義の一覧の表示項目

項目	説明	対応するタグ
Managed Connection Factory 実装クラス	javax.resource.spi.ManagedConnectionFactory インタフェースを実装したクラスが表示されます。選択すると，コネクション定義のプロパティページが表示されます。	<managedconnectionfactory-class> クラス名 </managedconnectionfactory-class>
コネクション定義識別子	リソースアダプタによってサポートされる ConnectionFactory インタフェースが表示されます。	<connectionfactory-interface> コネクション定義識別子 </connectionfactory-interface>
Connection Factory 実装クラス	ConnectionFactory インタフェースを実装したクラスが表示されます。	<connectionfactory-impl-class> クラス名 </connectionfactory-impl-class>
Connection インターフェース	リソースアダプタによってサポートされる Connection インタフェースが表示されます。	<connection-interface> インタフェース名 </connection-interface>
Connection 実装クラス	Connection インタフェースを実装したクラスが表示されます。	<connection-impl-class> クラス名 </connection-impl-class>

(6) コネクション定義のプロパティページ

コネクション定義のプロパティページを次に示します。

図 21-44　コネクション定義のプロパティページ

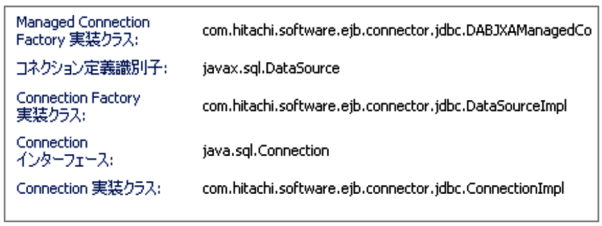


表 21-20　コネクション定義のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
Managed Connection Factory 実装クラス	—	javax.resource.spi.ManagedConnectionFactory インタフェースを実装した Java クラス名が表示されます。	<managedconnectionfactory-class> クラス名 </managedconnectionfactory-class>

項目	必須／任意	説明	対応するタグ
コネクション定義識別子	－	リソースでサポートされる ConnectionFactory インタフェースのクラス名が表示されます。	<connectionfactory-interface> コネクション定義識別子 </connectionfactory-interface>
Connection Factory 実装クラス	－	ConnectionFactory インタフェースを実装した Java クラス名が表示されます。	<connectionfactory-impl-class> クラス名 </connectionfactory-impl-class>
Connection インターフェース	－	リソースアダプタでサポートされる Connection インタフェースのクラス名	<connection-interface> インタフェース名 </connection-interface>
Connection 実装クラス	－	Connection インタフェースを実装した Java クラス名が表示されます。	<connection-impl-class> クラス名 </connection-impl-class>

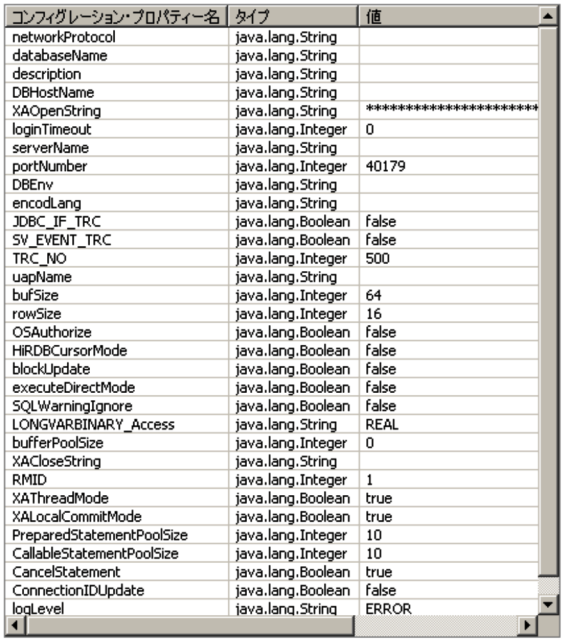
（凡例） －：変更不可

(7) コンフィグレーションプロパティの一覧ページ

コンフィグレーションプロパティの一覧ページには，設定されたコンフィグレーションのプロパティの一覧が表示されます。

コンフィグレーションプロパティの一覧ページを次に示します。

図 21-45　コンフィグレーションプロパティの一覧ページ



コンフィグレーション・プロパティー名	タイプ	値
networkProtocol	java.lang.String	
databaseName	java.lang.String	
description	java.lang.String	
DBHostName	java.lang.String	
XAOpenString	java.lang.String	**********
loginTimeout	java.lang.Integer	0
serverName	java.lang.String	
portNumber	java.lang.Integer	40179
DBEnv	java.lang.String	
encodLang	java.lang.String	
JDBC_IF_TRC	java.lang.Boolean	false
SV_EVENT_TRC	java.lang.Boolean	false
TRC_NO	java.lang.Integer	500
uapName	java.lang.String	
bufSize	java.lang.Integer	64
rowSize	java.lang.Integer	16
OSAuthorize	java.lang.Boolean	false
HiRDBCursorMode	java.lang.Boolean	false
blockUpdate	java.lang.Boolean	false
executeDirectMode	java.lang.Boolean	false
SQLWarningIgnore	java.lang.Boolean	false
LONGVARBINARY_Access	java.lang.String	REAL
bufferPoolSize	java.lang.Integer	0
XACloseString	java.lang.String	
RMID	java.lang.Integer	1
XAThreadMode	java.lang.Boolean	true
XALocalCommitMode	java.lang.Boolean	true
PreparedStatementPoolSize	java.lang.Integer	10
CallableStatementPoolSize	java.lang.Integer	10
CancelStatement	java.lang.Boolean	true
ConnectionIDUpdate	java.lang.Boolean	false
logLevel	java.lang.String	ERROR

表 21-21　コンフィグレーションプロパティの一覧ページの表示項目

項目	説明	対応するタグ
コンフィグレーション・プロパティー名	コンフィグレーションプロパティ名が表示されます。選択すると，コンフィグレーションプロパティのプロパティページが表示されます。	<config-property-name> コンフィグレーションプロパティ名 </config-property-name>
タイプ	コンフィグレーションプロパティの Java タイプが表示されます。	<config-property-type> タイプ </config-property-type>
値	コンフィグレーションのプロパティに設定されている値が表示されます。※	<config-property-value> 値 </config-property-value>

注※　J2EE サーバから取得した Connector 属性ファイルの「XAOpenString」に値が設定されている場合，一覧ではアスタリスク（「*」）が表示されます。

（8）コンフィグレーションプロパティのプロパティページ（XAOpenString 以外）

「XAOpenString」以外のコンフィグレーションプロパティのプロパティページについて説明します。「XAOpenString」のプロパティページについては，「(9)　コンフィグレーションプロパティのプロパティページ（XAOpenString)」を参照してください。

コンフィグレーションプロパティのプロパティページで，次の実装クラスのインスタン

スに対するコンフィグレーションプロパティを設定します。

- ResourceAdapter 実装クラス
- Managed Connection Factory 実装クラス

「XAOpenString」以外のコンフィグレーションプロパティのプロパティページを次に示します。

図 21-46 コンフィグレーションプロパティのプロパティページ（XAOpenString 以外の場合）

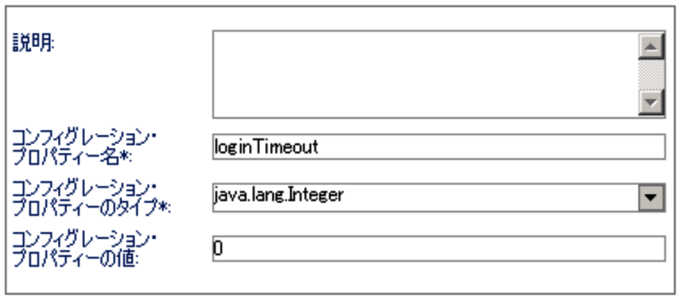


表 21-22 コンフィグレーションプロパティのプロパティページの設定項目（XAOpenString 以外の場合）

項目	必須／任意	説明	対応するタグ
説明	×	コンフィグレーションプロパティについての説明が表示されます。	<description> 説明 </description>
コンフィグレーション・プロパティー名	○	コンフィグレーションプロパティ名が表示されます。※1	<config-property-name> XAOpenString 以外※2 </config-property-name>
コンフィグレーション・プロパティーのタイプ	○	コンフィグレーションプロパティの Java タイプを指定します。 指定できる値を次に示します。※1 • java.lang.Boolean • java.lang.String • java.lang.Integer • java.lang.Double • java.lang.Byte • java.lang.Short • java.lang.Long • java.lang.Float • java.lang.Character	<config-property-type> コンフィグレーションプロパティのタイプ </config-property-type>

項目	必須／任意	説明	対応するタグ
コンフィグレーション・プロパティーの値	×	コンフィグレーションプロパティの値が表示されます。※1	<config-property-value> コンフィグレーションプロパティの値 </config-property-value>

（凡例）　○：必須　×：任意

注※ 1

指定できる値の詳細については，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（アプリケーション／リソース定義）」の「4.1.4　ResourceAdapter インスタンスに関するコンフィグレーションプロパティの定義」，「4.1.5　アウトバウンドリソースアダプタの定義」または「4.1.7　管理対象オブジェクトの定義」を参照してください。

注※ 2

「XAOpenString」以外のコンフィグレーションプロパティ名が表示されます。コンフィグレーションプロパティ名を「XAOpenString」に変更した場合，ページを再表示してください。

XAOpenString のコンフィグレーションプロパティのプロパティページが表示されます。

XAOpenString のコンフィグレーションプロパティのプロパティページについては，「(9)　コンフィグレーションプロパティのプロパティページ（XAOpenString）」を参照してください。

(9) コンフィグレーションプロパティのプロパティページ（XAOpenString）

「XAOpenString」のコンフィグレーションプロパティのプロパティページについて説明します。「XAOpenString」以外のプロパティページについては，「(8)　コンフィグレーションプロパティのプロパティページ（XAOpenString 以外）」を参照してください。

コンフィグレーションプロパティのプロパティページで，次の実装クラスのインスタンスに対するコンフィグレーションプロパティを設定します。

- ResourceAdapter 実装クラス
- Managed Connection Factory 実装クラス

「XAOpenString」のコンフィグレーションプロパティのプロパティページを次に示します。

図 21-47　コンフィグレーションプロパティのプロパティページ（XAOpenString の場合）

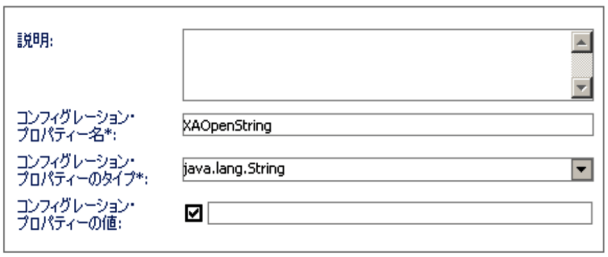


表 21-23　コンフィグレーションプロパティのプロパティページの設定項目（XAOpenString の場合）

項目	必須／任意	説明	対応するタグ
説明	×	コンフィグレーションプロパティについての説明が表示されます。	<description> 説明 </description>
コンフィグレーション・プロパティー名	○	コンフィグレーションプロパティ名（XAOpenString）が表示されます。※	<config-property-name> XAOpenString </config-property-name>
コンフィグレーション・プロパティーのタイプ	○	コンフィグレーションプロパティの Java タイプ（java.lang.String）を指定します。	<config-property-type> java.lang.String </config-property-type>
コンフィグレーション・プロパティーの値	×	チェックボックスで，コンフィグレーションプロパティ（XAOpenString）の値を変更するかどうかを指定します。変更する場合，チェックボックスにチェックを入れて，入力フィールドにコンフィグレーションプロパティ（XAOpenString）の値を指定します。	<config-property-value> XAOpenString の値 </config-property-value>

（凡例）　○：必須　×：任意

注※　コンフィグレーションプロパティ名を「XAOpenString」から「XAOpenString」以外に変更した場合，ページを再表示してください。XAOpenString 以外の場合のプロパティページが表示されます。「XAOpenString」以外のコンフィグレーションプロパティのプロパティページについては，「(8)　コンフィグレーションプロパティのプロパティページ（XAOpenString 以外）」を参照してください。

J2EE サーバから取得した Connector 属性ファイルの状態によって，コンフィグレーションプロパティ名（XAOpenString）の値は次のように表示されます。

Connector 属性ファイルの状態	コンフィグレーションプロパティの値（<config-property-value>）	チェックボックスのチェック	入力フィールド
コンフィグレーションプロパティの値が設定されていない	空タグ（<config-property-value></config-property-value>）	あり	入力可能。
コンフィグレーションプロパティの値が設定されている	<!--The config-property-value has already been set.-->	なし	入力不可。アスタリスク（「*」）で表示。

チェックボックスのチェックを変更した場合，コンフィグレーションプロパティページの入力フィールドおよび Connector 属性ファイルは次のように変更されます。

チェックボックスのチェック	コンフィグレーションプロパティページの入力フィールド	Connector 属性ファイル
「なし」から「あり」に変更	入力フィールドが表示されます。	空タグ（<config-property-value></config-property-value>）が設定されます。
「あり」から「なし」に変更	入力フィールドが削除されます。	<config-property-value> タグが削除されます。

（10）プロパティのプロパティページ（Outbound リソースアダプタ）

Outbound リソースアダプタのプロパティのプロパティページを次に示します。

図 21-48　プロパティのプロパティページ（Outbound リソースアダプタ）

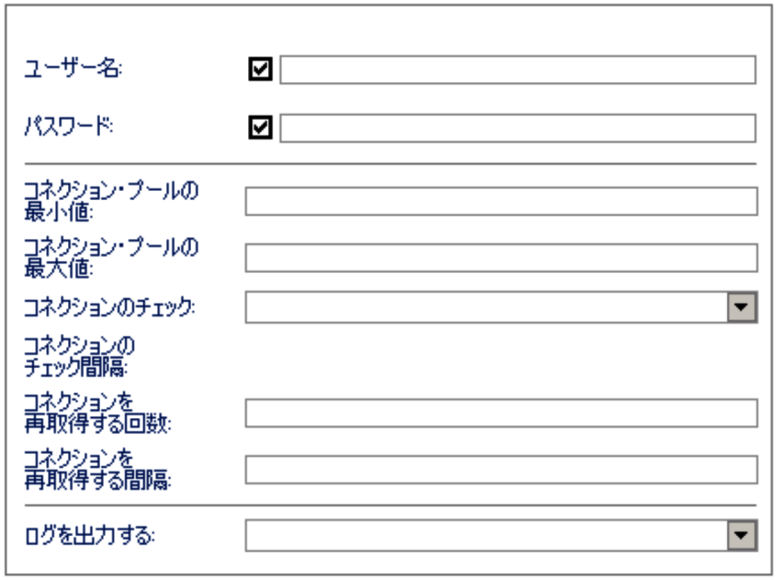


表 21-24 プロパティのプロパティページ（Outbound リソースアダプタ）の設定項目

項目	必須／任意	説明	対応するタグ
ユーザー名※1	×	チェックボックスでユーザ名を変更するかどうかを指定します。変更する場合，チェックを入れてユーザ名を指定します。※2	<property> <property-name> User </property-name> <property-type> String </property-type> <property-value> ユーザ名 </property-value> </property>
パスワード※1	×	チェックボックスでパスワードを変更するかどうかを指定します。変更する場合，チェックを入れてパスワードを指定します。※2	<property> <property-name> Password </property-name> <property-type> String </property-type> <property-value> パスワード </property-value> </property>
コネクション・プールの最小値	×	コネクションプールの最小値を，「0」から「2147483647」の整数で指定します。ただし，［コネクション・プールの最大値］に指定した値以下の値を指定してください。 指定がない場合，「10」が仮定されます。	<property> <property-name> MinPoolSize </property-name> <property-type> int </property-type> <property-value> コネクションプールの最小値 </property-value> </property>
コネクション・プールの最大値	×	コネクションプールの最大値を，「0」から「2147483647」の整数，または「-1」で指定します。「-1」を指定した場合，コネクションプールの最大値は無制限になります。 指定がない場合，「10」が仮定されます。	<property> <property-name> MaxPoolSize </property-name> <property-type> int </property-type> <property-value> コネクションプールの最大値 </property-value> </property>

項目	必須／任意	説明	対応するタグ
コネクションのチェック	×	リソースアダプタのコネクションをチェックするタイミングを選択します。 • （空白） • コネクションのチェックをしない • コネクションの取得時にチェックする • 一定の間隔でチェックする （空白）は，「コネクションの取得時にチェックする」が仮定されます。	<property> <property-name> ValidationType </property-name> <property-type> int </property-type> <property-value> コネクションのチェック </property-value> </property>
コネクションのチェック間隔	×	コネクションをチェックする間隔を，「1（秒）」から「2147483647（秒）」の整数で指定します。 指定がない場合，「3600」が仮定されます。 「コネクションのチェック」で「一定間隔でチェックする」を指定した場合にだけ指定できます。	<property> <property-name> ValidationInterval </property-name> <property-type> int </property-type> <property-value> コネクションのチェック間隔 </property-value> </property>
コネクションを再取得する回数	×	コネクションの取得に失敗したときに，何回取得をやり直すのかを，「0（回）」から「2147483647（回）」の整数で指定します。 指定がない場合，「0」が仮定されます。	<property> <property-name> RetryCount </property-name> <property-type> int </property-type> <property-value> コネクションを再取得する回数 </property-value> </property>
コネクションを再取得する間隔	×	コネクションの取得をやり直す間隔を，「1（秒）」から「2147483647I（秒）」の整数で指定します。 指定がない場合，「10」が仮定されます。	<property> <property-name> RetryInterval </property-name> <property-type> int </property-type> <property-value> コネクションを再取得する間隔 </property-value> </property>

項目	必須／任意	説明	対応するタグ
ログを出力する	×	LogWriter によるログ出力をするかどうかを指定します。 指定できる値を次に示します。 • true：出力する • false：出力しない 指定がない場合，「true」が仮定されます。	<property> <property-name> LogEnabled </property-name> <property-type> boolean </property-type> <property-value> ログを出力する </property-value> </property>

（凡例） ×：任意

注※1 J2EE サーバから取得した Connector 属性ファイルの状態によって，ユーザ名およびパスワードは次のように表示されます。

Connector 属性ファイルの状態	プロパティの値（<property-value>）	チェックボックスのチェック	入力フィールド
ユーザ名およびパスワードが設定されていない	空タグ（<property-value></property-value>）	あり	入力可能。
ユーザ名およびパスワードが設定されている	<!--The property-value has already been set.-->	なし	入力不可。 アスタリスク（「*」）で表示。

注※2 チェックボックスのチェックを変更した場合，プロパティページの入力フィールドおよび Connector 属性ファイルは次のように変更されます。

チェックボックスのチェック	プロパティページの入力フィールド	Connector 属性ファイル
「なし」から「あり」に変更	入力フィールドが表示される	空タグ（<property-value></property-value>）が設定される
「あり」から「なし」に変更	入力フィールドが削除される	<property-value> タグが削除される

（11）リソース外部プロパティの一覧ページ

リソース外部プロパティの一覧ページを次に示します。

リソース外部プロパティの一覧ページには，設定されたリソース外部プロパティの一覧が表示されます。

図 21-49　リソース外部プロパティの一覧ページ

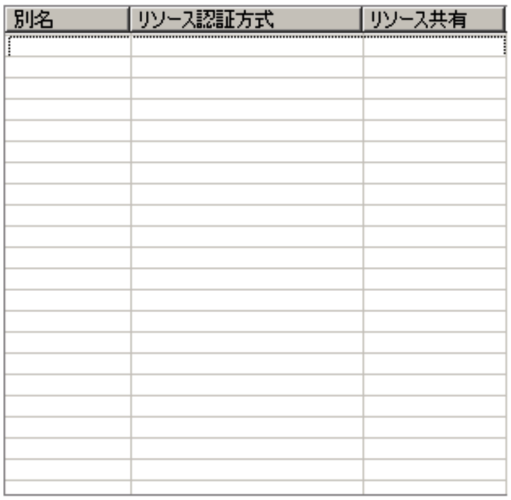


表 21-25　リソース外部プロパティの一覧ページの表示項目

項目	説明	対応するタグ
別名	リソースの別名が表示されます。選択すると，コンフィグレーションプロパティのプロパティページが表示されます。	<optional-name> 別名 </optional-name>
リソース認証方式	リソース外部プロパティに設定されている，リソース認証方式が表示されます。	<res-auth> リソース認証方式 </res-auth>
リソース共有	リソース外部プロパティに設定されている，リソース接続を共有できるようにするかどうかどうかが表示されます。	<res-sharing-scope> リソース共有 <res-sharing-scope>

(12) リソース外部プロパティのプロパティページ

リソース外部プロパティのプロパティページを次に示します。

図 21-50　リソース外部プロパティのプロパティページ

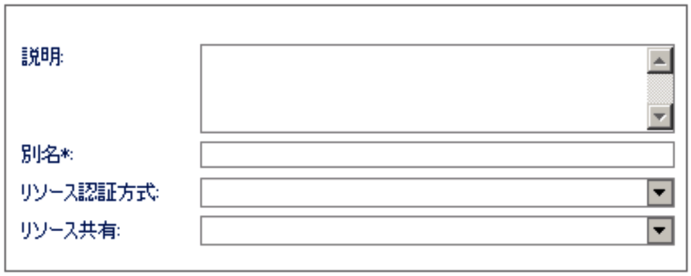


表 21-26 リソース外部プロパティのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	リソース外部プロパティの説明を指定します。	<description> 説明 </description>
別名	○	リソースの別名を指定します。 最大 255 文字で指定してください。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_），ピリオド（.），ハイフン（-），および区切り文字としてスラッシュ（/）です。ただし，予約語の「HITACHI_EJB」（大文字と小文字を区別する）で始まる文字列は指定できません。 名前の先頭や末尾にピリオドは使用できません。また，ピリオドだけの名前も使用できません。 指定値の前後に指定した空白，または改行は無視されます。 文字列の途中に空白，または改行を指定した場合，エラーになります。	<optional-name> 別名 </optional-name>
リソース認証方式	×	リソースを使用するための認証を，アプリケーションで行うか，コンテナで行うかを指定します。 指定できる値を次に示します。 •「Application」：アプリケーションで認証する •「Container」：コンテナで認証する	<res-auth> リソース認証方式 </res-auth> <res-auth> Application </res-auth> <res-auth> Container </res-auth>
リソース共有	×	リソース接続を共有するかどうかを指定します。 指定できる値を次に示します。 •「Shareable」：共有する •「Unshareable」：共有しない 指定がない場合，「Shareable」が仮定されます。	<res-sharing-scope> リソース共有 <res-sharing-scope> <res-sharing-scope> Shareable <res-sharing-scope> <res-sharing-scope> Unshareable <res-sharing-scope>

（凡例） ○：必須 ×：任意

（13）メッセージリスナの一覧ページ

メッセージリスナの一覧ページには，設定されたメッセージリスナのプロパティの一覧が表示されます。

メッセージリスナの一覧ページを次に示します。

図 21-51　メッセージリスナの一覧ページ

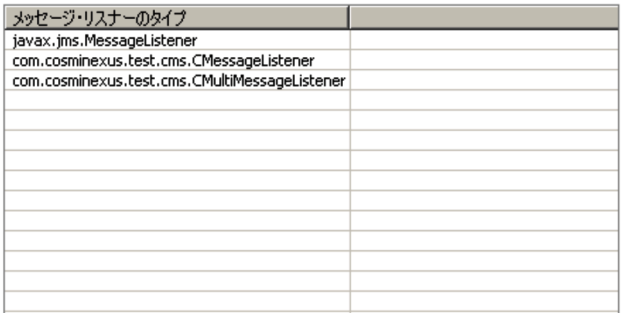


表 21-27　メッセージリスナの一覧ページの表示項目

項目	説明	対応するタグ
メッセージ・リスナーのタイプ	メッセージリスナのタイプが表示されます。	<messagelistener-type> メッセージリスナのタイプ </messagelistener-type>

（14）メッセージリスナのプロパティページ

メッセージリスナのプロパティページを次に示します。

図 21-52　メッセージリスナのプロパティページ

表 21-28　メッセージリスナのプロパティページの表示項目

項目	説明	対応するタグ
メッセージ・リスナーのタイプ	メッセージリスナのタイプが表示されます。	<messagelistener-type> メッセージリスナのタイプ </messagelistener-type>

（15）ActivationSpec のプロパティページ

ActivationSpec のプロパティページを次に示します。

図 21-53　ActivationSpec のプロパティページ

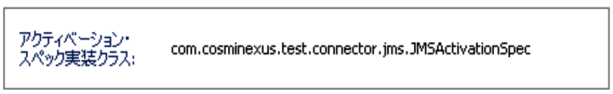


表 21-29 ActivationSpec のプロパティページの表示項目

項目	説明	対応するタグ
アクティベーション・スペック実装クラス	ActivationSpec のインタフェースを実装するクラスが表示されます。	<activationspec-class> ActivationSpec のインタフェースを実装するクラス </activationspec-class>

(16) 必須コンフィグレーションプロパティの一覧ページ

必須コンフィグレーションプロパティの一覧ページを次に示します。

図 21-54 必須コンフィグレーションプロパティの一覧ページ

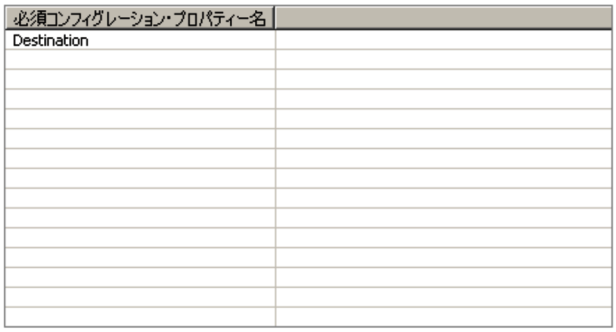


表 21-30 必須コンフィグレーションプロパティの一覧ページの表示項目

項目	説明	対応するタグ
必須コンフィグレーション・プロパティー名	必須コンフィグレーションプロパティ名が表示されます。	<config-property-name> 必須コンフィグレーションプロパティ名 </config-property-name>

(17) 必須コンフィグレーションプロパティのプロパティページ

必須コンフィグレーションプロパティのプロパティページを次に示します。

図 21-55 必須コンフィグレーションプロパティのプロパティページ

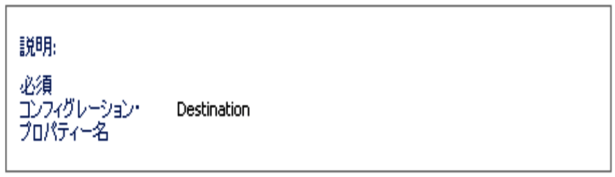


表 21-31　必須コンフィグレーションプロパティのプロパティページの表示項目

項目	説明	対応するタグ
説明	必須コンフィグレーションプロパティの説明が表示されます。	<description> 説明 </description>
必須コンフィグレーション・プロパティー名	必須コンフィグレーションプロパティ名が表示されます。	<config-property-name> 必須コンフィグレーションプロパティ名 </config-property-name>

(18) 管理対象オブジェクトの一覧ページ

管理対象オブジェクトの一覧ページを次に示します。

図 21-56　管理対象オブジェクトの一覧ページ

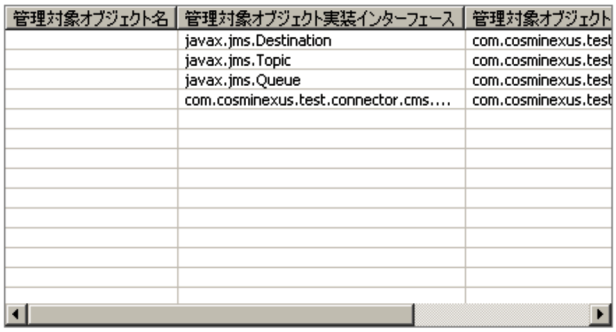


表 21-32　管理対象オブジェクトの一覧ページの表示項目

項目	説明	対応するタグ
管理対象オブジェクト名	管理対象オブジェクト名が表示されます。	<adminobject-name> 管理対象オブジェクト名 </adminobject-name>
管理対象オブジェクト実装インターフェース	管理対象オブジェクトを実装するインタフェースが表示されます。	<adminobject-interface> 管理対象オブジェクト実装インタフェース </adminobject-interface>
管理対象オブジェクト実装クラス	管理対象オブジェクトを実装するクラスが表示されます。	<adminobject-class> 管理対象オブジェクト実装クラス </adminobject-class>

(19) 管理対象オブジェクトのプロパティページ

管理対象オブジェクトのプロパティページを次に示します。

図 21-57　管理対象オブジェクトのプロパティページ

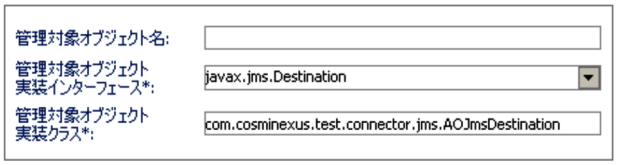


表 21-33　管理対象オブジェクトのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
管理対象オブジェクト名	×	管理対象オブジェクト名を指定します。リソースアダプタ内で一意な名称を指定します。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），およびアンダースコア（_）です。	<adminobject-name> 管理対象オブジェクト名 </adminobject-name>
管理対象オブジェクト実装インターフェース	○	管理対象オブジェクトを実装するインタフェースを指定します。 コンボボックスには，次のインタフェースが表示されます。 • javax.jms.Queue • javax.jms.Topic	<adminobject-interface> 管理対象オブジェクト実装インタフェース </adminobject-interface> <adminobject-interface> javax.jms.Queue </adminobject-interface> <adminobject-interface> javax.jms.Topic </adminobject-interface>
管理対象オブジェクト実装クラス	○	管理対象オブジェクトを実装するクラスを指定します。	<adminobject-class> 管理対象オブジェクト実装クラス </adminobject-class>

（凡例）　○：必須　×：任意

（20）プロパティのプロパティページ（リソースアダプタ）

リソースアダプタのプロパティのプロパティページを次に示します。

図 21-58　プロパティのプロパティページ（リソースアダプタ）

表 21-34　プロパティのプロパティページ（リソースアダプタ）の設定項目

項目	必須／任意	説明	対応するタグ
スレッド・プールの最小スレッド数	×	スレッドプールの最小スレッド数を「0」から「1024」の整数で指定します。指定がない場合，「0」が仮定されます。	<property> <property-name> MinTPoolSize </property-name> <property-type> int </property-type> <property-value> スレッドプールの最小スレッド数 </property-value> </property>
スレッド・プールの最大スレッド数	×	スレッドプールで同時に実行される最大スレッド数を「1」から「2147483647」の整数で指定します。指定がない場合，「10」が仮定されます。	<property> <property-name> MaxTPoolSize </property-name> <property-type> int </property-type> <property-value> スレッドプールの最大スレッド数 </property-value> </property>
スレッド・プールのタイムアウト値	×	スレッドプールのスレッド解放までのタイムアウト値（秒）を「1」から「2147483647」の整数で指定します。指定がない場合，「300」が仮定されます。	<property> <property-name> TPoolKeepalive </property-name> <property-type> int </property-type> <property-value> スレッドプールのタイムアウト値 </property-value> </property>

（凡例）　×：任意

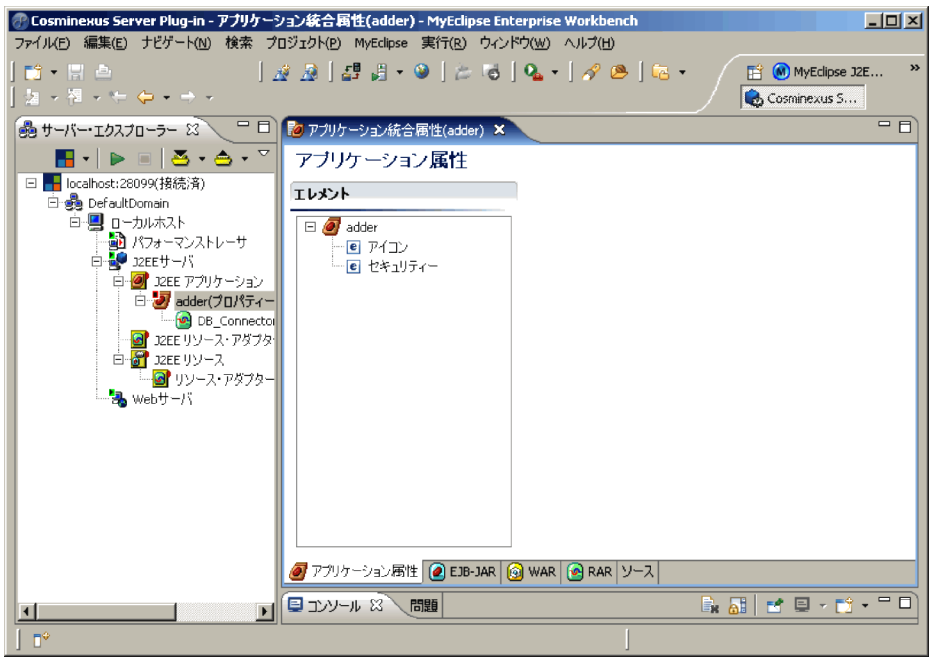
21.3　J2EE アプリケーションの属性編集画面

ここでは，J2EE アプリケーションの属性編集画面について説明します。

J2EE アプリケーションの属性編集画面は，次の手順で表示します。

1. **［サーバー・エクスプローラー］ビューで［J2EE アプリケーション］フォルダ内の J2EE アプリケーションを選択します。**
2. **属性ファイル編集エディタを起動します。**
   属性ファイル編集エディタを起動する方法については，「21.1.14　プロパティの設定」を参照してください。属性ファイル編集エディタが起動されると，パースペクティブのエディタエリアに，アプリケーション統合属性ファイルのフォームページが表示されます。

図 21-59　アプリケーション統合属性ファイルのフォームページ



アプリケーション統合属性ファイルのフォームタブで属性ファイルの種別を分けて，各ページが表示されています。アプリケーション統合属性ファイル（<hitachi-application-all-property>）のフォームページの種別を次に示します。

表 21-35　アプリケーション統合属性ファイルのフォームページの種別

フォームタブ	属性	対応するタグ名	参照先
アプリケーション属性	アプリケーション属性	<hitachi-application-property>	21.4

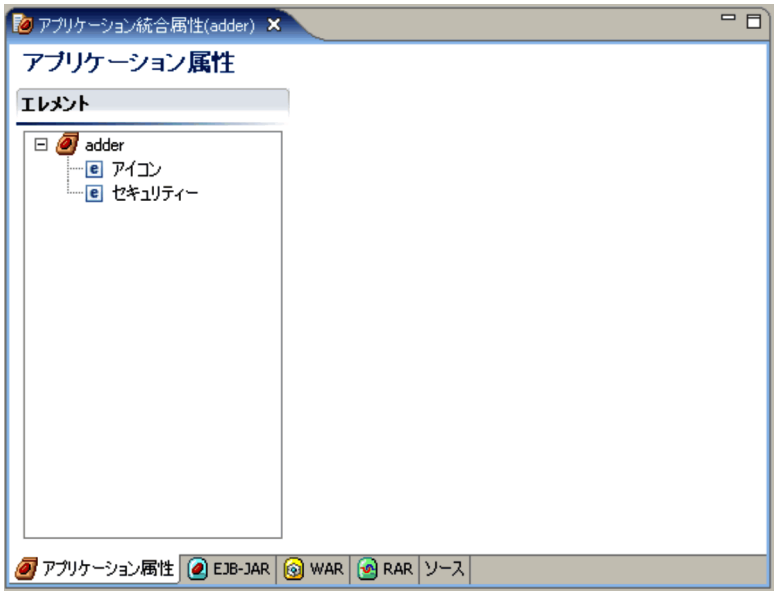
フォームタブ	属性	対応するタグ名	参照先
EJB-JAR	―	<ejb-jar>	―
	EJB-JAR 属性	<hitachi-ejb-jar-property>	21.5
	Session Bean 属性	<hitachi-session-bean-property>	21.6
	Entity Bean 属性	<hitachi-entity-bean-property>	21.7
	Message-driven Bean 属性	<hitachi-message-bean-property>	21.8
WAR	―	<war>	―
	WAR 属性	<hitachi-war-property>	21.9
	フィルタ属性	<hitachi-filter-property>	21.10
	サーブレット属性	<hitachi-servlet-property>	21.11
RAR	―	<rar>	―
	Connector 属性	<hitachi-Connector-property>	21.12

（凡例） ―：該当なし

21.4　アプリケーション統合属性ファイル（アプリケーション属性）

アプリケーション統合属性ファイルの［アプリケーション属性］タブを選択すると，［アプリケーション属性］フォームページが表示されます。

図 21-60　［アプリケーション属性］フォームページ例



フォームページの［エレメント］の下に，アプリケーション属性の構成のツリーが展開されます。

21.4.1　アプリケーション属性のツリー

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。エディタエリアの［エレメント］に，編集するアプリケーション属性の構成がツリー形式で表示されます。

アプリケーション属性のツリー構成を次に示します。

アプリケーション属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。アプリケーション属性のツリーノードとコンテキストメニューについて，次に説明します。

(1) アプリケーション属性のツリーノード

アプリケーション属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。

表 21-36　アプリケーション属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜アプリケーション属性名＞	アプリケーション属性を示すノードです。 選択すると，アプリケーションのプロパティページが表示されます。 アプリケーション属性がない場合，「アプリケーション属性はありません」が表示されます。	<hitachi-application-property>	21.4.2　(1)
アイコン	GUI ツール上に表示される EAR のアイコンの定義を示すノードです。 選択すると，アイコンのプロパティページが表示されます。	<icon>	21.4.2　(2)
セキュリティー	セキュリティの管理情報を示すノードです。 選択すると，セキュリティのプロパティページが表示されます。	<security-prop>	21.4.2　(3)
CTM 連携	CTM 連携の設定情報を示すノードです。 選択すると，CTM 連携のプロパティページが表示されます。	<scheduling>	21.4.2　(4)

(2) コンテキストメニュー操作

アプリケーション属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-37　アプリケーション属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜アプリケーション属性名＞	［追加］-［アイコン］	アプリケーション属性にアイコンが追加されます。選択すると，アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.4.2　(2)

ノード	コンテキストメニュー	説明	プロパティページ
	［追加］-［CTM 連携］	アプリケーション属性に CTM 連携が追加されます。選択すると，CTM 連携のプロパティページが表示されます。すでに CTM 連携がある場合は選択できません。	21.4.2 (4)
アイコン	［削除］	アプリケーション属性から，アイコンが削除されます。	－
CTM 連携	［削除］	アプリケーション属性から，CTM 連携が削除されます。	－

（凡例） －：該当なし

21.4.2 アプリケーション属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されます。［エレメントの詳細］に，編集するアプリケーション属性のプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアに，選択したノードに対応するプロパティページが表示されます。

アプリケーション属性には，次のプロパティページがあります。

- アプリケーションのプロパティページ
- アイコンのプロパティページ
- セキュリティのプロパティページ
- CTM 連携のプロパティページ

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

(1) アプリケーションのプロパティページ

アプリケーションのプロパティページを次に示します。

図 21-61 アプリケーションのプロパティページ

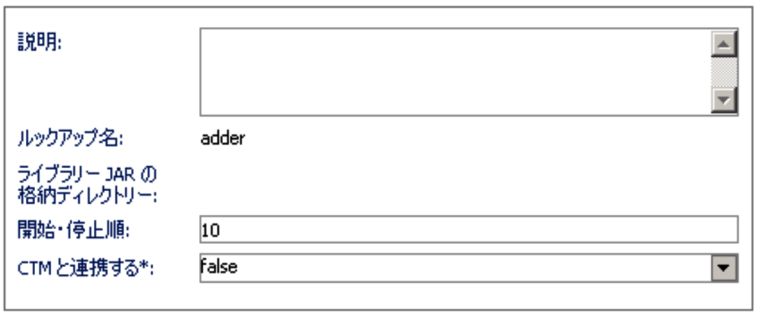


表 21-38　アプリケーションのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	アプリケーションの説明を指定します。	<description> 説明 </description>
ルックアップ名	－	アプリケーションのルックアップ名が表示されます。	<lookup-name> ルックアップ名 </lookup-name>
ライブラリー JAR の格納ディレクトリー	－	ライブラリ JAR を格納するディレクトリ名が表示されます。	<library-directory> ライブラリ JAR の格納ディレクトリ </library-directory>
開始・停止順	×	アプリケーションの開始および停止の順序を，「0」から「2147483647」の整数で指定してください。 指定がない場合，「10」が仮定されます。	<start-order> 開始・停止順 </start-order>
CTM と連携する	○	CTM との連携をするかどうかを指定してください。指定できる値を次に示します。 •「true」：CTM と連携する •「false」：CTM と連携しない	<managed-by-ctm> CTM と連携する </managed-by-ctm>

（凡例）　○：必須　×：任意　－：変更不可

(2) アイコンのプロパティページ

各属性ファイルの <icon> タグで囲まれたアイコンのプロパティを設定する共通のプロパティページを次に示します。

図 21-62　アイコンのプロパティページ

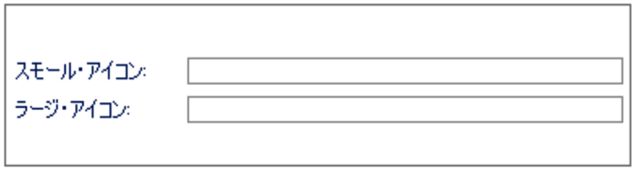


表 21-39　アイコンのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
スモール・アイコン	×	サイズが 16 × 16 ピクセル以下のアイコン (GIF ファイルまたは JPG ファイル) を指定します。	<small-icon> スモールアイコン </small-icon>

項目	必須／任意	説明	対応するタグ
ラージ・アイコン	×	サイズが 32 × 32 ピクセル以下のアイコン (GIF ファイルまたは JPG ファイル) を指定します。	<large-icon> ラージアイコン </large-icon>

（凡例） ×：任意

(3) セキュリティのプロパティページ

セキュリティのプロパティページを次に示します。

図 21-63 セキュリティのプロパティページ

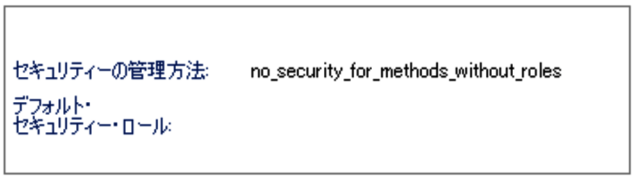


表 21-40 セキュリティのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
セキュリティーの管理方法	—	セキュリティの管理方法が表示されます。	<security-method> セキュリティの管理方法 <security-method>
デフォルト・セキュリティー・ロール	—	セキュリティの管理方法に「map_methods_without_roles」が指定されている場合のデフォルトのセキュリティロールが表示されます。	<default-security-role> デフォルトセキュリティロール </default-security-role>

（凡例） —：変更不可

(4) CTM 連携のプロパティページ

CTM 連携のプロパティページを次に示します。

図 21-64 CTM 連携のプロパティページ

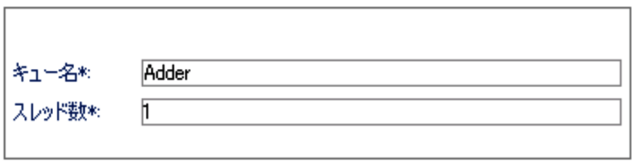


表 21-41　CTM 連携のプロパティページの設定項目

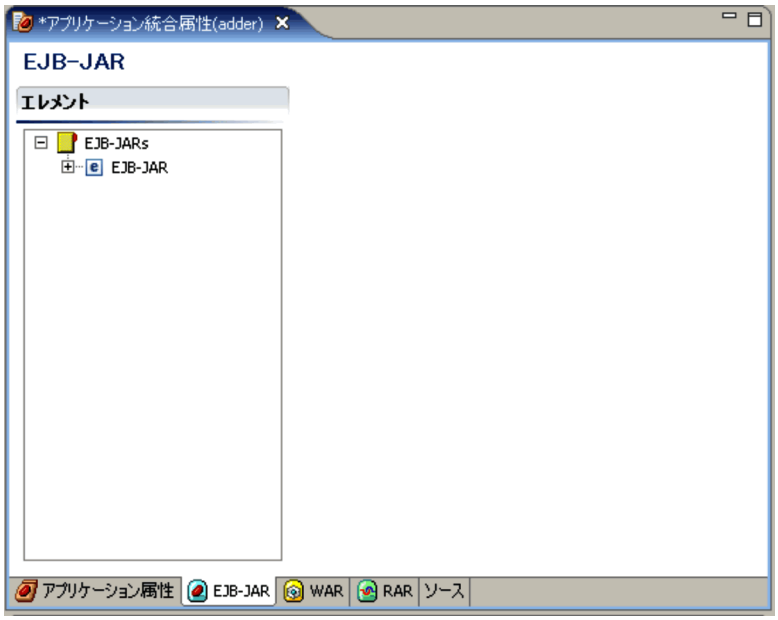
項目	必須／任意	説明	対応するタグ
キュー名	○	スケジューリングをするキューの名称を指定します。CTM 連携の追加で，CTM 連携のプロパティページが表示された場合，初期値としてアプリケーションの表示名が設定されています。 不活性の場合も，指定されている値がそのまま表示されます。	<queue-name> キュー名 </queue-name>
スレッド数	○	CTM がアプリケーションを呼び出すために用意するスレッド数を，「1」から「127」の範囲で指定します。CTM 連携の追加で，CTM 連携のプロパティページが表示された場合，初期値として「1」が設定されています。 不活性の場合も，指定されている値がそのまま表示されます。	<parallel-count> スレッド数 </parallel-count>

（凡例）　○：必須

21.5　アプリケーション統合属性ファイル（EJB-JAR 属性）

アプリケーション統合属性ファイルの［EJB-JAR］タブを選択すると，［EJB-JAR］フォームページが表示されます。

図 21-65　［EJB-JAR］フォームページ例



フォームページの［エレメント］の下に，EJB-JAR の構成によって，次のツリーが展開されます。

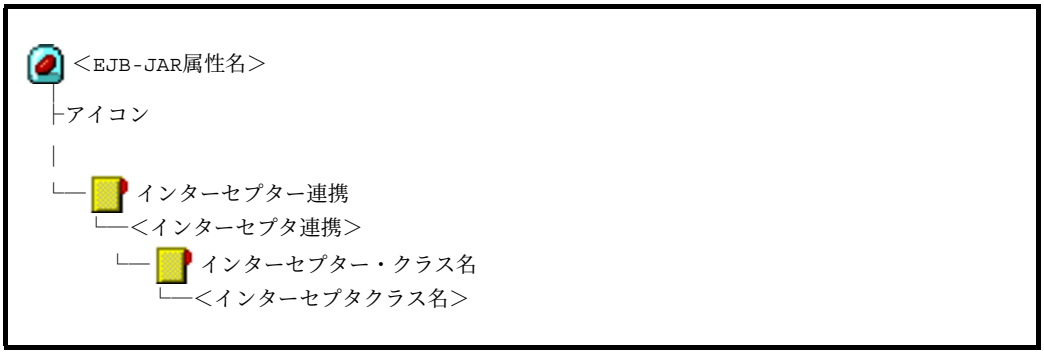
- EJB-JAR 属性
  ＜ EJB-JAR 属性名＞を選択すると，EJB-JAR 属性のツリーが表示されます。
  EJB-JAR 属性のツリーとプロパティの設定項目については，この節で説明しています。
- Session Bean 属性
  ＜ Session Bean 属性名＞を選択すると，Session Bean 属性のツリーが表示されます。
  Session Bean 属性のツリーとプロパティの設定項目については，「21.6　アプリケーション統合属性ファイル（Session Bean 属性）」を参照してください。
- Entity Bean 属性
  ＜ Entity Bean 属性名＞を選択すると，Entity Bean 属性のツリーが表示されます。
  Entity Bean 属性のツリーとプロパティの設定項目については，「21.7　アプリケーション統合属性ファイル（Entity Bean 属性）」を参照してください。
- Message-driven Bean 属性

＜Message-driven Bean 属性名＞を選択すると，Message-driven Bean 属性のツリーが表示されます。Message-driven Bean 属性のツリーとプロパティの設定項目については，「21.8　アプリケーション統合属性ファイル（Message-driven Bean 属性）」を参照してください。

21.5.1　EJB-JAR 属性のツリー

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。エディタエリアの［エレメント］に，編集する EJB-JAR 属性ファイルの構成がツリー形式で表示されます。

EJB-JAR 属性のツリー構成を次に示します。



EJB-JAR 属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。EJB-JAR 属性のツリーノードとコンテキストメニューについて，次に説明します。

（1）EJB-JAR 属性のツリーノード

EJB-JAR 属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。

表 21-42　EJB-JAR 属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜EJB-JAR 属性名＞	EJB-JAR 属性を示すノードです。 選択すると，EJB-JAR 属性のプロパティページが表示されます。 EJB-JAR 属性がない場合，「EJB-JAR はありません」が表示されます。	<hitachi-ejb-jar-property>	21.5.2　(1)

ノード	説明	対応するタグ名	プロパティページ
アイコン	アイコンを示すノードです。 選択すると，アイコンのプロパティページが表示されます。	<icon>	21.4.2 (2)
インターセプター連携	インターセプタ連携をまとめるノードです。インターセプタ連携は，クラスレベル・インターセプタおよびメソッドレベル・インターセプタに関する設定を表します。 選択すると，インターセプタ連携の一覧ページが表示されます。	−	21.5.2 (3)
＜インターセプタ連携＞	インターセプタ連携を示すノードです。 選択すると，インターセプタ連携のプロパティページが表示されます。	<interceptor-binding>	21.5.2 (4)
インターセプター・クラス名	インターセプタクラス名をまとめるノードです。 選択すると，インターセプタクラス名の一覧ページが表示されます。	−	21.5.2 (5)
＜インターセプタクラス名＞	インターセプタクラス名を示すノードです。 選択すると，インターセプタクラス名のプロパティページが表示されます。	<interceptor-class>	21.5.2 (6)

(2) コンテキストメニュー操作

EJB-JAR 属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-43 EJB-JAR 属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜ EJB-JAR 属性名 ＞	［追加］-［アイコン］	EJB-JAR 属性にアイコンが追加されます。アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.4.2 (2)
	［追加］-［インターセプター連携］	インターセプタ連携が追加されます。インターセプタ連携のプロパティページが表示されます。	21.5.2 (4)
アイコン	［削除］	EJB-JAR 属性からアイコンが削除されます。	−

ノード	コンテキストメニュー	説明	プロパティページ
インターセプター連携	［追加］-［インターセプター連携］	インターセプタ連携が追加されます。インターセプタ連携のプロパティページが表示されます。	21.5.2 (4)
	［削除］	選択しているノードが削除されます。 <ejb-name> 要素がワイルドカード（*）以外の場合，不活性となり削除できません。	－
＜インターセプタ連携＞	［追加］-［インターセプター・クラス名］	インターセプタクラス名が追加されます。インターセプタクラス名のプロパティページが表示されます。 <ejb-name> 要素がワイルドカード（*）以外の場合，不活性となり追加できません。	21.5.2 (6)
	［削除］	選択しているノードが削除されます。 <ejb-name> 要素がワイルドカード（*）以外の場合，不活性となり削除できません。	－
インターセプター・クラス名	［追加］-［インターセプター・クラス名］	インターセプタクラス名が追加されます。インターセプタクラス名のプロパティページが表示されます。 <ejb-name> 要素がワイルドカード（*）以外の場合，不活性となり追加できません。	21.5.2 (6)
	［削除］	選択しているノードが削除されます。 <ejb-name> 要素がワイルドカード（*）以外の場合，不活性となり削除できません。	－
＜インターセプタクラス名＞	［削除］	選択しているノードが削除されます。	－

21.5.2 EJB-JAR 属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に編集する EJB-JAR 属性ファイルのプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアの表示が切り替わります。

EJB-JAR 属性ファイルには，次のプロパティページがあります。

- EJB-JAR のプロパティページ
- アイコンのプロパティページ
- インターセプタ連携の一覧ページ
- インターセプタ連携のプロパティページ
- インターセプタクラス名の一覧ページ
- インターセプタクラス名のプロパティページ

(1) EJB-JAR のプロパティページ

EJB-JAR のプロパティページを次に示します。

図 21-66　EJB-JAR のプロパティページ

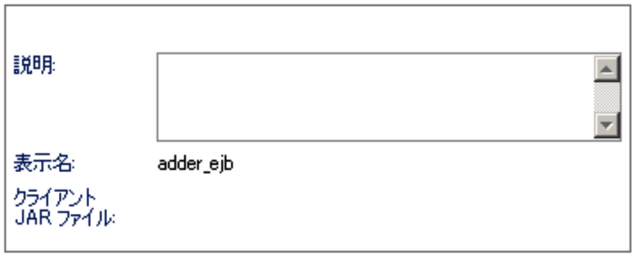


表 21-44　EJB-JAR のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
表示名	―	EJB-JAR に設定した表示名が表示されます。	<display-name> 表示名 </display-name>
クライアント JAR ファイル	―	クライアント JAR ファイル名が表示されます。	<ejb-client-jar> クライアント JAR ファイル </ejb-client-jar>

（凡例）　×：任意　―：変更不可

(2) アイコンのプロパティページ

アイコンのプロパティページの詳細については，「21.4.2　アプリケーション属性設定ページ」の「(2)　アイコンのプロパティページ」を参照してください。

(3) インターセプタ連携の一覧ページ

インターセプタ連携の一覧ページを次に示します。

図 21-67　インターセプタ連携の一覧ページ

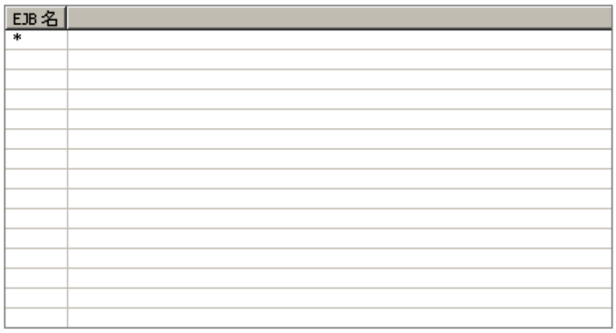


表 21-45　インターセプタ連携の一覧ページの表示項目

項目	説明	対応するタグ
EJB 名	インターセプタ連携の EJB 名が表示されます。	<ejb-name> EJB 名 </ejb-name>

(4) インターセプタ連携のプロパティページ

インターセプタ連携のプロパティページを次に示します。

図 21-68　インターセプタ連携のプロパティページ

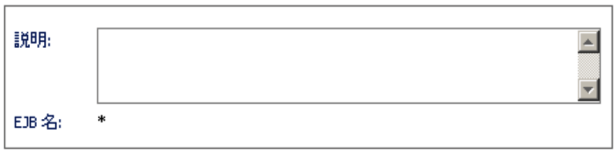


表 21-46　インターセプタ連携のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	インターセプタ連携の EJB 名の説明を指定します。 EJB 名がワイルドカード（*）以外の場合，説明は指定できません。	<description> 説明 </description>
EJB 名	－	インターセプタ連携の EJB 名が表示されます。	<ejb-name> EJB 名 </ejb-name>

（凡例）　×：任意　－：変更不可

(5) インターセプタクラス名の一覧ページ

インターセプタクラス名の一覧ページを次に示します。

図 21-69　インターセプタクラス名の一覧ページ

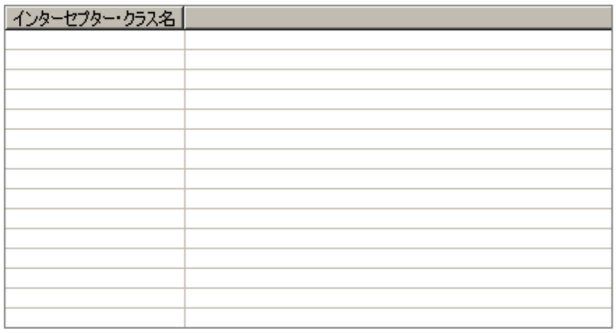


表 21-47　インターセプタクラス名の一覧ページの表示項目

項目	説明	対応するタグ
インターセプター・クラス名	インターセプタ連携のクラス名が表示されます。	<interceptor-class> インターセプタクラス名 </interceptor-class>

(6) インターセプタクラス名のプロパティページ

インターセプタクラス名のプロパティページを次に示します。

図 21-70　インターセプタクラス名のプロパティページ

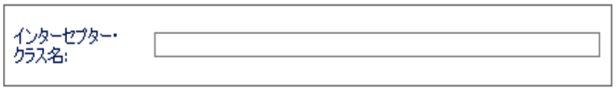


表 21-48　インターセプタクラス名のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
インターセプター・クラス名	×	インターセプタ連携のクラス名を指定します。 EJB 名がワイルドカード（*）以外の場合，クラス名は指定できません。	<interceptor-class> インターセプタクラス名 </interceptor-class>

（凡例）　×：任意

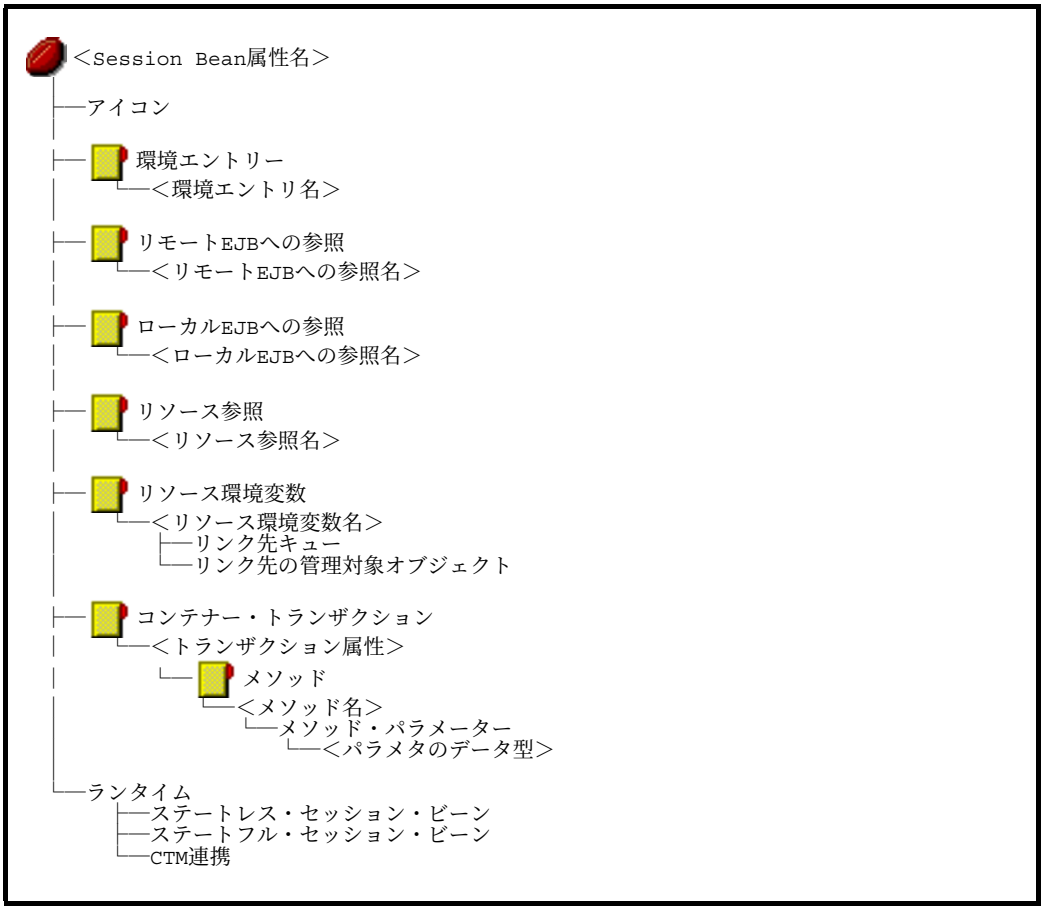
21.6　アプリケーション統合属性ファイル（Session Bean 属性）

アプリケーション統合属性ファイルの［EJB-JAR］タブを選択すると，［EJB-JAR］フォームページが表示されます。［EJB-JAR］フォームページの［エレメント］に表示されている，［＜ Session Bean 属性名＞］を選択して，Session Bean 属性のツリーを表示します。

21.6.1　Session Bean 属性のツリー

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。エディタエリアの［エレメント］に，編集する Session Bean 属性ファイルの構成がツリー形式で表示されます。

Session Bean 属性のツリー構成を次に示します。



Session Bean 属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。Session Bean 属性のツリーノードとコンテキストメニューについて，次に説明します。

（1） Session Bean 属性のツリーノード

Session Bean 属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。

表 21-49 Session Bean 属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜ Session Bean 属性名＞	Session Bean の属性を示すノードです。選択すると，Session Bean のプロパティページが表示されます。	<hitachi-session-bean-property>	21.6.2 (1)
アイコン	アイコンを示すノードです。選択すると，アイコンのプロパティページが表示されます。	<icon>	21.4.2 (2)
環境エントリー	環境エントリをまとめるノードです。 選択すると，環境エントリの一覧ページが表示されます。	－	21.6.2 (3)
＜環境エントリ名＞	環境エントリを示すノードです。選択すると，環境エントリのプロパティページが表示されます。	<env-entry>	21.6.2 (4)
リモート EJB への参照	リモート EJB への参照をまとめるノードです。 選択すると，リモート EJB への参照の一覧ページが表示されます。	－	21.6.2 (5)
＜リモート EJB への参照名＞	リモート EJB への参照を示すノードです。 選択すると，リモート EJB への参照のプロパティページが表示されます。	<ejb-ref>	21.6.2 (6)
ローカル EJB への参照	ローカル EJB への参照をまとめるノードです。 選択すると，ローカル EJB への参照の一覧ページが表示されます。	－	21.6.2 (7)
＜ローカル EJB への参照名＞	ローカル EJB への参照を示すノードです。 選択すると，ローカル EJB への参照のプロパティページが表示されます。	<ejb-local-ref>	21.6.2 (8)

ノード	説明	対応するタグ名	プロパティページ
リソース参照	リソース参照をまとめるノードです。 選択すると，リソース参照の一覧ページが表示されます。	－	21.6.2 (9)
＜リソース参照名＞	リソース参照を示すノードです。 選択すると，リソース参照のプロパティページが表示されます。	<resource-ref>	21.6.2 (10)
リソース環境変数	リソース環境変数をまとめるノードです。 選択すると，リソース環境変数の一覧ページが表示されます。	－	21.6.2 (11)
＜リソース環境変数名＞	リソース環境変数名を示すノードです。 選択すると，リソース環境変数のプロパティページが表示されます。	<resource-env-ref>	21.6.2 (12)
リンク先キュー	リンク先キューを示すノードです。 選択すると，リンク先キューのプロパティページが表示されます。	<linked-queue>	21.6.2 (13)
リンク先の管理対象オブジェクト	リンク先の管理対象オブジェクトを示すノードです。 選択すると，リンク先の管理対象オブジェクトのプロパティページが表示されます。	<linked-adminobject>	21.6.2 (14)
コンテナー・トランザクション	コンテナトランザクションをまとめるノードです。 選択すると，コンテナトランザクションの一覧ページが表示されます。	－	21.6.2 (15)
＜トランザクション属性＞	コンテナトランザクションを示すノードです。 選択すると，コンテナトランザクションのプロパティページが表示されます。	<container-transaction>	21.6.2 (16)
メソッド	メソッドをまとめるノードです。 選択すると，メソッドの一覧ページが表示されます。	－	21.6.2 (17)
＜メソッド名＞	メソッドを示すノードです。 選択すると，メソッドのプロパティページが表示されます。	<method>	21.6.2 (18)
メソッド・パラメーター	メソッドパラメタを表します。 選択すると，メソッドパラメタの一覧ページが表示されます。	<method-params>	21.6.2 (19)

ノード	説明	対応するタグ名	プロパティページ
＜パラメタのデータ型＞	パラメタのデータ型を示すノードです。 選択すると，パラメタのデータ型プロパティページが表示されます。	<method-param>	21.6.2 (20)
ランタイム	ランタイムを示すノードです。 選択すると，ランタイムのプロパティページが表示されます。	<session-runtime>	21.6.2 (21)
ステートレス・セッション・ビーン	Stateless Session Bean を示すノードです。 選択すると，Stateless Session Bean のプロパティページが表示されます。	<stateless>	21.6.2 (22)
ステートフル・セッション・ビーン	Stateful Session Bean を示すノードです。 選択すると，Stateful Session Bean のプロパティページが表示されます。	<stateful>	21.6.2 (23)
CTM 連携	CTM 連携を示すノードです。 選択すると，CTM 連携のプロパティページが表示されます。	<scheduling>	21.6.2 (24)

（凡例） －：該当なし

(2) コンテキストメニュー操作

Session Bean 属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-50 Session Bean 属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜ Session Bean 属性名＞	［追加］-［アイコン］	アイコンが追加されます。アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.4.2 (2)
	［追加］-［環境エントリー］	環境エントリが追加されます。環境エントリのプロパティページが表示されます。	21.6.2 (4)
	［追加］-［リモート EJB への参照］	リモート EJB への参照が追加されます。リモート EJB への参照のプロパティページが表示されます。	21.6.2 (6)

ノード	コンテキストメニュー	説明	プロパティページ
	［追加］-［ローカル EJB への参照］	ローカル EJB への参照が追加されます。ローカル EJB への参照のプロパティページが表示されます。	21.6.2 (8)
	［追加］-［リソース参照］	リソース参照が追加されます。リソース参照のプロパティページが表示されます。	21.6.2 (10)
	［追加］-［リソース環境変数］	リソース環境変数が追加されます。リソース環境変数のプロパティページが表示されます。	21.6.2 (12)
	［追加］-［コンテナー・トランザクション］	コンテナトランザクションが追加されます。コンテナトランザクションのプロパティページが表示されます。	21.6.2 (16)
アイコン	［削除］	選択しているノードが削除されます。	－
環境エントリー	［追加］-［環境エントリー］	環境エントリが追加されます。環境エントリのプロパティページが表示されます。	21.6.2 (4)
	［削除］	環境エントリがすべて削除されます。	－
＜環境エントリー名＞	［削除］	選択しているノードが削除されます。	－
リモート EJB への参照	［追加］-［リモート EJB への参照］	「リモート EJB への参照」が追加されます。 リモート EJB への参照のプロパティページが表示されます。	21.6.2 (6)
	［削除］	リモート EJB への参照がすべて削除されます。	－
＜リモート EJB への参照名＞	［削除］	選択しているノードが削除されます。	－
ローカル EJB への参照	［追加］-［ローカル EJB への参照］	「ローカル EJB への参照」が追加されます。 ローカル EJB への参照のプロパティページが表示されます。	21.6.2 (8)
	［削除］	ローカル EJB への参照がすべて削除されます。	－
＜ローカル EJB への参照名＞	［削除］	選択しているノードが削除されます。	－

ノード	コンテキストメニュー	説明	プロパティページ
リソース参照	［追加］-［リソース参照］	リソース参照が追加されます。 リソース参照のプロパティページが表示されます。	21.6.2 (10)
	［削除］	リソース参照がすべて削除されます。	－
＜リソース参照名＞	［削除］	選択しているノードが削除されます。	－
リソース環境変数	［追加］-［リソース環境変数］	リソース環境変数が追加されます。 リソース環境変数のプロパティページが表示されます。	21.6.2 (12)
	［削除］	リソース環境変数がすべて削除されます。	－
＜リソース環境変数＞	［追加］-［リンク先キュー］	リソース環境変数にリンク先キューが追加されます。 リンク先キューのプロパティページが表示されます。	21.6.2 (13)
	［追加］-［リンク先の管理対象オブジェクト］	リソース環境変数にリンク先の管理対象オブジェクトが追加されます。 リンク先の管理対象オブジェクトのプロパティページが表示されます。	21.6.2 (14)
	［削除］	選択しているノードが削除されます。	－
リンク先キュー	［削除］	選択しているノードが削除されます。	－
リンク先の管理対象オブジェクト	［削除］	選択しているノードが削除されます。	－
コンテナー・トランザクション	［追加］-［コンテナー・トランザクション］	コンテナトランザクションが追加されます。 コンテナトランザクションのプロパティページが表示されます。	21.6.2 (16)
	［削除］	コンテナトランザクションがすべて削除されます。	－
＜コンテナトランザクション＞	［追加］-［メソッド］	メソッドが追加されます。 メソッドのプロパティページが表示されます。	21.6.2 (18)
	［削除］	選択しているノードが削除されます。	－
メソッド	［追加］-［メソッド］	「メソッド」が追加されます。 メソッドのプロパティページが表示されます。	21.6.2 (18)

ノード	コンテキストメニュー	説明	プロパティページ
＜メソッド名＞	［追加］-［メソッド・パラメーター］	［メソッド］にメソッドパラメタが追加されます。 ［メソッド・パラメーター］のノードが作成され，メソッドパラメタの一覧ページが表示されます。 すでにメソッドパラメタがある場合は選択できません。	21.6.2 (19)
	［削除］	選択しているノードが削除されます。	－
メソッド・パラメーター	［追加］-［パラメーターのデータ型］	メソッドパラメタにパラメタのデータ型が追加されます。 パラメタのデータ型プロパティページが表示されます。	21.6.2 (20)
	［削除］	選択しているノードが削除されます。	－
＜パラメタのデータ型＞	［削除］	選択しているノードが削除されます。	－
ランタイム	［追加］-［CTM 連携］	ランタイムに CTM 連携が追加されます。 CTM 連携のプロパティページが表示されます。 すでに CTM 連携がある場合は選択できません。	21.6.2 (24)
ステートレス・セッション・ビーン	－	－	－
ステートフル・セッション・ビーン	－	－	－
CTM 連携	［削除］	選択しているノードが削除されます。	－

（凡例） －：該当なし

21.6.2 Session Bean 属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に編集する Session Bean 属性ファイルのプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアの表示が切り替わります。

Session Bean 属性ファイルには，次のプロパティページがあります。

- Session Bean のプロパティページ
- アイコンのプロパティページ

- 環境エントリの一覧ページ
- 環境エントリのプロパティページ
- リモート EJB への参照の一覧ページ
- リモート EJB への参照のプロパティページ
- ローカル EJB への参照の一覧ページ
- ローカル EJB への参照のプロパティページ
- リソース参照の一覧ページ
- リソース参照のプロパティページ
- リソース環境変数の一覧ページ
- リソース環境変数のプロパティページ
- リンク先キューのプロパティページ
- リンク先の管理対象オブジェクトのプロパティページ
- コンテナトランザクションの一覧ページ
- コンテナトランザクションのプロパティページ
- メソッドの一覧ページ
- メソッドのプロパティページ
- メソッドパラメタの一覧ページ
- パラメタのデータ型プロパティページ
- ランタイムのプロパティページ
- Stateless Session Bean のプロパティページ
- Stateful Session Bean のプロパティページ
- CTM 連携のプロパティページ

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

（1） Session Bean のプロパティページ

Session Bean のプロパティページを次に示します。

図 21-71 Session Bean のプロパティページ

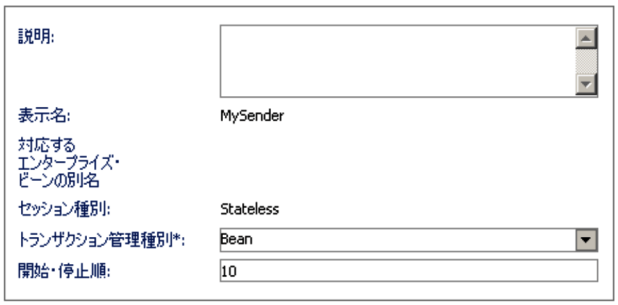


表 21-51　Session Bean のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	Session Bean 属性の説明を指定します。	<description> 説明 </ description>
表示名	－	Session Bean 属性の表示名が表示されます。	<display-nam e> 表示名 </ display-name >
対応するエンタープライズ・ビーンの別名	－	対応する Enterprise Bean の別名が表示されます。	<mapped-na me> 対応する Enterprise Bean の別名 </ mapped-name >
セッション種別	－	セッション種別が表示されます。 セッション種別が Stateful Session Bean の場合，「Stateful」が表示されます。 セッション種別が Stateless Session Bean の場合，「Stateless」が表示されます。	<session-type > セッション種別 </ session-type>
トランザクション管理種別	○	トランザクション管理種別を指定します。 指定できる値を次に示します。 • Bean • Container	<transaction-t ype> トランザクション管理種別 </ transaction-ty pe>
開始・停止順	×	開始・停止順を，「0」から「2147483647」の整数で指定します。 指定がない場合，「10」が仮定されます。	<start-order> 開始・停止順 </start-order>

（凡例）　○：必須　×：任意　－：変更不可

(2) アイコンのプロパティページ

Session Bean 属性のアイコンのプロパティページについては，「21.4.2　アプリケーション属性設定ページ」の「(2)　アイコンのプロパティページ」を参照してください。

(3) 環境エントリの一覧のページ

環境エントリの一覧のページを次に示します。

環境エントリの一覧には，設定された環境エントリがリストで表示されます。

図 21-72 環境エントリの一覧のページ

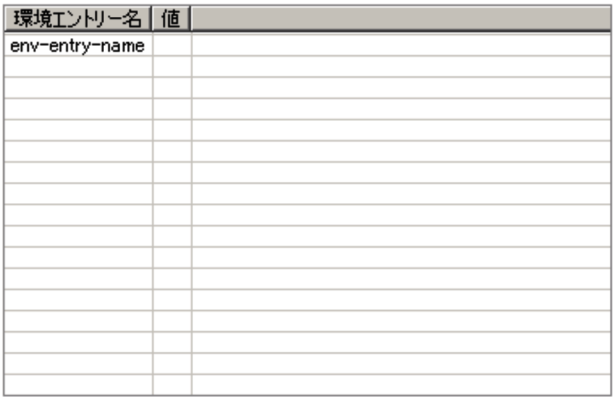


表 21-52 環境エントリの一覧の表示項目

項目	説明	対応するタグ
環境エントリー名	環境エントリ名が表示されます。	<env-entry-name> 環境エントリ名 </env-entry-name>
値	環境エントリの値が表示されます。	<env-entry-value> 値 </env-entry-value>

(4) 環境エントリのプロパティページ

環境エントリのプロパティページを次に示します。

図 21-73 環境エントリのプロパティページ

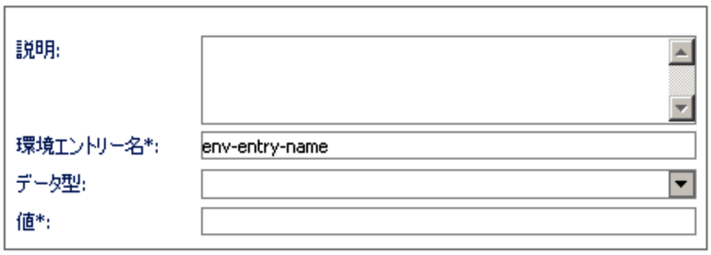


表 21-53　環境エントリのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
環境エントリー名	○	環境エントリ名を指定します。※1	<env-entry-name> 環境エントリ名 </env-entry-name>
データ型	○	環境エントリのデータ型を選択します。 選択できる値を次に示します。 • java.lang.Boolean • java.lang.String • java.lang.Integer • java.lang.Double • java.lang.Byte • java.lang.Short • java.lang.Long • java.lang.Float • java.lang.Character	<env-entry-type> データ型 </env-entry-type>
値	×※2	環境エントリの値を指定します。 データ型に指定した型に適合した値を指定します。	<env-entry-value> 値 </env-entry-value>

（凡例）　○：必須　×：任意

注※1　同一属性内に複数の環境エントリがある場合，同一の環境エントリ名（<env-entry-name>）は指定できません。

注※2　データ型が「java.lang.String」以外の場合は，必須です。

(5) リモート EJB への参照の一覧ページ

リモート EJB への参照の一覧ページを次に示します。

リモート EJB への参照の一覧には，設定されたリモート EJB への参照がリストで表示されます。

図 21-74 リモート EJB への参照の一覧ページ

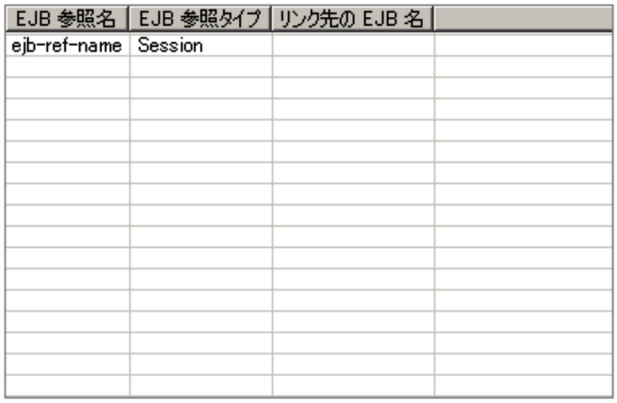


表 21-54 リモート EJB への参照の一覧の表示項目

項目	説明	対応するタグ
EJB 参照名	EJB 参照名が表示されます。	<ejb-ref-name> EJB 参照名 </ejb-ref-name>
EJB 参照タイプ	EJB 参照タイプが表示されます。	<ejb-ref-type> EJB 参照タイプ </ejb-ref-type>
リンク先の EJB 名	リンク先の EJB 名が表示されます。	<ejb-link> リンク先の EJB 名 </ejb-link>

(6) リモート EJB への参照のプロパティページ

リモート EJB への参照のプロパティページを次に示します。

図 21-75 リモート EJB への参照のプロパティページ

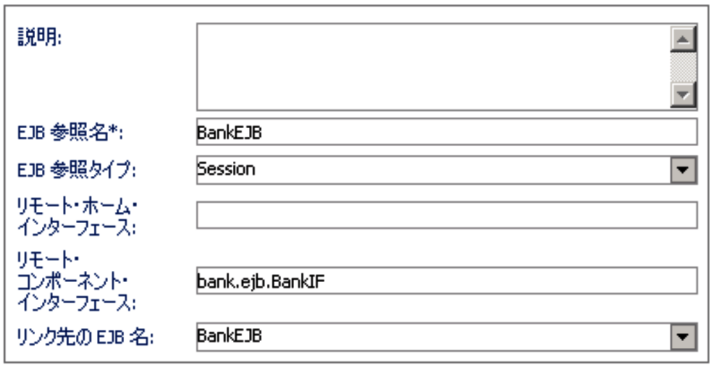


表 21-55　リモート EJB への参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
EJB 参照名	○	EJB 参照名を指定します。※1	<ejb-ref-name> EJB 参照名 </ejb-ref-name>
EJB 参照タイプ	○	EJB 参照タイプを指定します。 指定できる値を次に示します。 • Session • Entity	<ejb-ref-type> EJB 参照タイプ </ejb-ref-type> <ejb-ref-type> Entity </ejb-ref-type> <ejb-ref-type> Session </ejb-ref-type>
リモート・ホーム・インターフェース	○※2	リモートホームインタフェースを完全修飾クラス名で指定します。	<home> リモートホームインタフェース </home>
リモート・コンポーネント・インターフェース	○※2	リモートコンポーネントインタフェースを完全修飾クラス名で指定します。	<remote> リモートコンポーネントインタフェース </remote>
リンク先の EJB 名	×	参照する EJB の EJB 名を入力するか，またはリスト項目を選択します。 • EJB がほかの ejb-jar ファイルにある場合は，次の形式で入力します。「ファイル名 #EJB 名」 • ネーミングの切り替え機能でリンク先を設定する場合，次の形式で入力します。 corbaname::< 名前空間のホスト名 >:< 名前空間のポート番号 >#<EJB ホームオブジェクトリファレンスの JNDI 名 • アプリケーション統合属性ファイルにある Enterprise Bean 表示名をリスト項目から選択します。	<ejb-link> リンク先の EJB 名 </ejb-link>

（凡例）　○：必須　×：任意

注※ 1　同一属性内にリモート EJB への参照とローカル EJB への参照が両方ある場合や，リモート EJB への参照が複数ある場合，同じ EJB 参照名（<ejb-ref-name>）は指定できません。

注※ 2　アノテーションで値が空に設定されている場合があります。アノテーションでの設定以外は，値の設定が必要となります。

(7) ローカル EJB への参照の一覧ページ

ローカル EJB への参照の一覧ページを次に示します。

ローカル EJB への参照の一覧ページでは，設定されたローカル EJB への参照がリストで表示されます。

図 21-76　ローカル EJB への参照の一覧ページ

表 21-56　ローカル EJB への参照の一覧の表示項目

項目	説明	対応するタグ
EJB 参照名	EJB 参照名が表示されます。	<ejb-ref-name> EJB 参照名 </ejb-ref-name>
EJB 参照タイプ	EJB 参照タイプが表示されます。	<ejb-ref-type> EJB 参照タイプ </ejb-ref-type>
リンク先の EJB 名	リンク先の EJB 名が表示されます。	<ejb-link> リンク先の EJB 名 </ejb-link>

(8) ローカル EJB への参照のプロパティページ

ローカル EJB への参照のプロパティページを次に示します。

図 21-77　ローカル EJB への参照のプロパティページ

表 21-57　ローカル EJB への参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
EJB 参照名	○	EJB 参照名を指定します。※1	<ejb-ref-name> EJB 参照名 </ejb-ref-name>
EJB 参照タイプ	○	EJB 参照タイプを指定します。 指定できる値を次に示します。 • Session • Entity	<ejb-ref-type> EJB 参照タイプ </ejb-ref-type>
ローカル・ホーム・インターフェース	○※2	ローカルホームインタフェースを完全修飾クラス名で指定します。	<local-home> ローカルホームインタフェース </local-home>
ローカル・コンポーネント・インターフェース	○※2	ローカルコンポーネントインタフェースを完全修飾クラス名で指定します。	<local> ローカルコンポーネントインタフェース </local>
リンク先の EJB 名	×	参照する EJB の EJB 名を入力するか，またはリスト項目を選択します。 • EJB がほかの ejb-jar ファイルにある場合は，次の形式で入力します。 「ファイル名 #EJB 名」 • アプリケーション統合属性ファイルにある Enterprise Bean 表示名をリスト項目から選択します。	<ejb-link> リンク先の EJB 名 </ejb-link>

（凡例）　○：必須　×：任意

注※1　同一属性内にリモート EJB への参照とローカル EJB への参照が両方ある場合や，ローカル EJB への参照が複数ある場合，同じ EJB 参照名（<ejb-ref-name>）は指定できません。
注※2　アノテーションで値が空に設定されている場合があります。アノテーションでの設定以外は，値の設定が必要となります。

(9) リソース参照の一覧ページ

リソース参照の一覧ページを次に示します。

リソース参照の一覧では，設定されたリソース参照がリストで表示されます。

図 21-78　リソース参照の一覧ページ

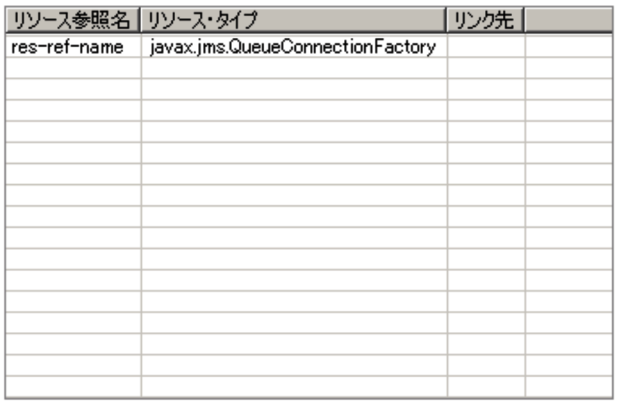


表 21-58　リソース参照の一覧の表示項目

項目	説明	対応するタグ
リソース参照名	リソース参照名が表示されます。	<res-ref-name> リソース参照名 </res-ref-name>
リソース・タイプ	リソースタイプが表示されます。	<res-type> リソースタイプ </res-type>
リンク先	リンク先が表示されます。	<linked-to> リンク先 </linked-to>

(10) リソース参照のプロパティページ

リソース参照のプロパティページを次に示します。

図 21-79　リソース参照のプロパティページ

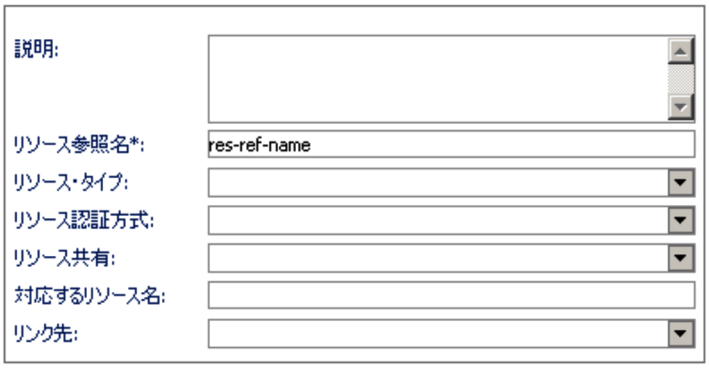


表 21-59　リソース参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
リソース参照名	○	リソース参照名を指定します。※1	<res-ref-name> リソース参照名 </res-ref-name>
リソース・タイプ	○	ソース種別を完全修飾クラス名で入力するか，または次のリスト項目を選択します。 • javax.jms.ConnectionFactory • javax.jms.QueueConnectionFactory • javax.jms.TopicConnectionFactory • javax.mail.Session • javax.resource.cci. ConnectionFactory • javax.sql.DataSource • org.omg.CORBA_2_3.ORB	<res-type> リソースタイプ </res-type>
リソース認証方式	○	リソース認証方式を指定します。 選択できる値を次に示します。 • Application • Container	<res-auth> リソース認証方式 </res-auth>
リソース共有	×	リソース共有を指定します。 選択できる値を次に示します。 •（空白） • Shareable • Unshareable	<res-sharing-scope> リソース共有 </res-sharing-scope>
対応するリソース名	－	対応するリソースアダプタ名，または メール表示名が表示されます。	<mapped-name> 対応するリソース名 </mapped-name>※2

項目	必須／任意	説明	対応するタグ
リンク先	×	リンク先を指定します。 リンク先として，対応する J2EE リソースアダプタの表示名を入力するか，またはリスト項目を選択します。 J2EE サーバにデプロイされている J2EE リソースアダプタの表示名がリスト項目※4 として表示されます。［サーバー・エクスプローラー］ビューでも J2EE リソースアダプタの表示名一覧を参照できます。	<linked-to> リンク先※3 </linked-to>

（凡例）　○：必須　×：任意　－：変更不可

注※1　同一属性内に複数のリソース参照名がある場合，同一のリソース参照名（<res-ref-name>）は指定できません。

注※2　<linked-to> タグに値が設定されている場合，<linked-to> タグの指定が有効になり，<mapped-name> タグの値は無効になります。

注※3　<linked-to> タグのリンク先の指定方法は，リソースアダプタのバージョンによって，次のように異なります。

- Connector 1.0 の仕様に準拠する場合：＜リソースアダプタの表示名＞
- Connector 1.5 の仕様に準拠する場合：＜リソースアダプタの表示名＞!＜コネクション定義識別子＞

注※4　リスト項目には，デプロイ済みのリソースアダプタが表示されます。なお，Connector1.5 の仕様に準拠するリソースアダプタの場合は，［リソース・タイプ］の値とコネクション定義識別子が一致する Outbound リソースアダプタが表示されます。

（11）リソース環境変数の一覧ページ

リソース環境変数の一覧ページを次に示します。

リソース環境変数の一覧には，設定されたリソース環境変数がリストで表示されます。

図 21-80　リソース環境変数の一覧ページ

表 21-60　リソース環境変数の一覧の表示項目

項目	説明	対応するタグ
リソース環境変数名	リソース環境変数名が表示されます。	<resource-env-ref-name> リソース環境変数名 </resource-env-ref-name>
リソース環境変数値のタイプ	リソース環境変数値のタイプが表示されます。	<resource-env-ref-type> リソース環境変数値のタイプ </resource-env-ref-type>
リソース・アダプター名	リソースアダプタ名が表示されます。	<resource-adapter> リソースアダプタ名 </resource-adapter>
キュー名	キュー名が表示されます。	<queue> キュー名 </queue>

(12) リソース環境変数のプロパティページ

リソース環境変数のプロパティページを次に示します。

図 21-81　リソース環境変数のプロパティページ

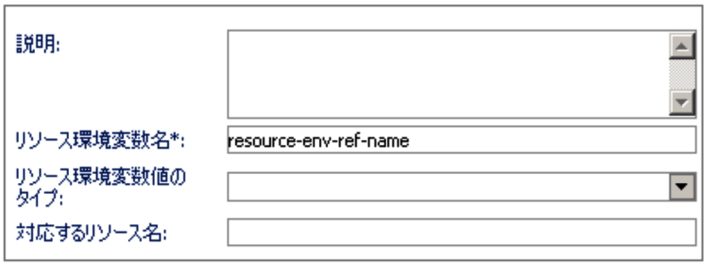


表 21-61　リソース環境変数のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
リソース環境変数名	○	リソース環境変数名を指定します。※1	<resource-env-ref-name> リソース環境変数名 </resource-env-ref-name>

項目	必須／任意	説明	対応するタグ
リソース環境変数値のタイプ	○	リソース環境変数値のタイプを指定します。 ＜JavaBeans リソースのクラス名＞を完全修飾クラス名で入力するか，または次のリスト項目を選択します。 • javax.jms.Queue • javax.jms.Topic • javax.transaction.UserTransaction ※2 • javax.ejb.TimerService ※2 • javax.ejb.EJBContext ※2	<resource-env-ref-type> リソース環境変数値のタイプ </resource-env-ref-type>
対応するリソース名	－	リソース環境変数値のタイプが「javax.jms.Queue」または「JavaBeans リソースのクラス名」の場合，対応するリソース名が表示されます。 •「javax.jms.Queue」の場合 対応するキューが「< リソースアダプタの表示名 >#<Queue 名称 >」の形式で表示されます。 •「JavaBeans リソースのクラス名」の場合 対応する JavaBeans リソースの表示名で表示されます。	<mapped-name> 対応するリソース名 </mapped-name>

（凡例） ○：必須　×：任意　－：変更不可

注※ 1　同一属性内に複数のリソース環境変数がある場合，同じリソース環境変数名（<resource-env-ref-name>）は指定できません。

注※ 2　Session Bean 属性の場合，アノテーションで設定された値です。リスト項目として表示されますが，アノテーションで設定された値は変更できないため，J2EE アプリケーションサーバへの保管時にサーバ側で変更前の設定値に戻されます。

（13）リンク先キューのプロパティページ

リンク先キューのプロパティページを次に示します。

図 21-82　リンク先キューのプロパティページ

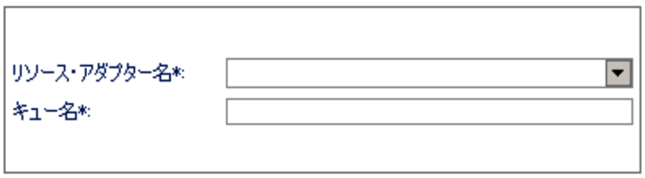


表 21-62　リンク先キューのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
リソース・アダプター名	○	J2EE リソースアダプタの表示名を入力※，または選択します。 J2EE サーバにデプロイされている J2EE リソースアダプタの表示名※がリスト項目として表示されます。［サーバー・エクスプローラー］ビューでも J2EE リソースアダプタの表示名一覧を参照できます。	<resource-adapter> リソースアダプタ名 </resource-adapter>
キュー名	○	キューの表示名を指定します。	<queue> キュー名 </queue>

（凡例）　○：必須

注※　Connector 1.5 の仕様に準拠するリソースアダプタは指定できません。また，リスト項目にも表示されません。

（14）リンク先の管理対象オブジェクトのプロパティページ

リンク先の管理対象オブジェクトのプロパティページを次に示します。

図 21-83　リンク先の管理対象オブジェクトのプロパティページ

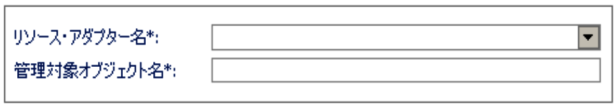


表 21-63　リンク先の管理対象オブジェクトのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
リソース・アダプター名	○	J2EE リソースアダプタの表示名を入力※，または選択します。 J2EE サーバにデプロイされている J2EE リソースアダプタの表示名※がリスト項目として表示されます。［サーバー・エクスプローラー］ビューでも J2EE リソースアダプタの表示名一覧を参照できます。	<resourceadapter-name> リソースアダプタ名 </resourceadapter-name>
管理対象オブジェクト名	○	管理対象オブジェクト名を指定します。指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），およびアンダースコア（_）です。	<adminobject-name> 管理対象オブジェクト名 </adminobject-name>

（凡例）　○：必須

注※　Connector 1.0 の仕様に準拠するリソースアダプタは指定できません。また，リスト項目にも表示されません。

（15）コンテナトランザクションの一覧ページ

コンテナトランザクションの一覧ページを次に示します。

コンテナトランザクションの一覧には，設定されたコンテナトランザクションがリストで表示されます。

図 21-84　コンテナトランザクションの一覧ページ

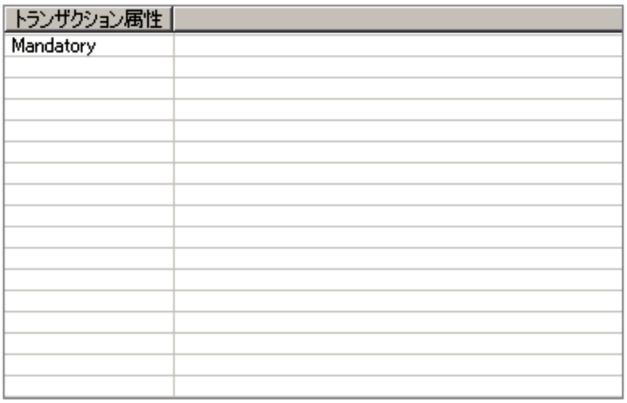


表 21-64　コンテナトランザクションの一覧の表示項目

項目	説明	対応するタグ
トランザクション属性	トランザクション属性が表示されます。	<trans-attribute> トランザクション属性 </trans-attribute>

（16）コンテナトランザクションのプロパティページ

コンテナトランザクションのプロパティページを次に示します。

図 21-85　コンテナトランザクションのプロパティページ

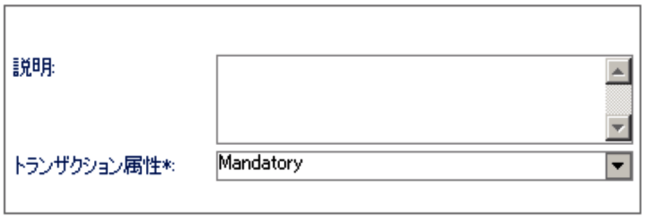


表 21-65　コンテナトランザクションのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
トランザクション属性	○	トランザクション属性を指定します。 指定できる値を次に示します。 • Mandatory • Never • NotSupported • Required • RequiresNew • Supports	<trans-attribute> トランザクション属性 </trans-attribute>

（凡例）　○：必須　×：任意

（17）メソッドの一覧ページ

メソッドの一覧ページを次に示します。

メソッドの一覧には，設定されたメソッドがリストで表示されます。

図 21-86　メソッドの一覧ページ

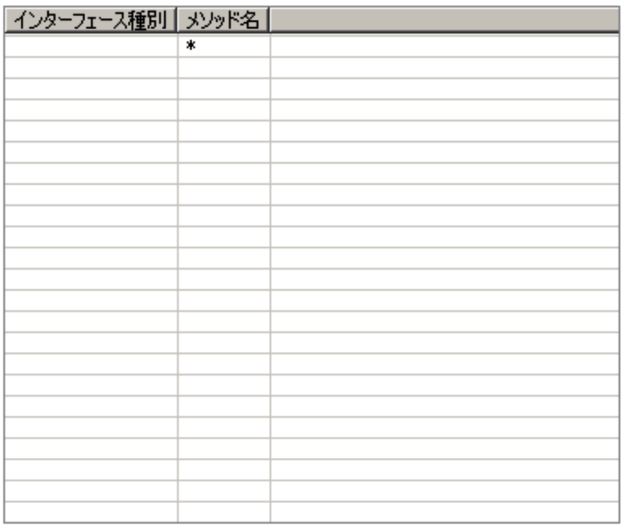


表 21-66　メソッドの一覧ページの表示項目

項目	説明	対応するタグ
インターフェース種別	インタフェース種別が表示されます。	<method-intf> インタフェース種別 </method-intf>

項目	説明	対応するタグ
メソッド名	メソッド名が表示されます。	<method-name> メソッド名 </method-name>

（18）メソッドのプロパティページ

メソッドのプロパティページを次に示します。

図 21-87　メソッドのプロパティページ

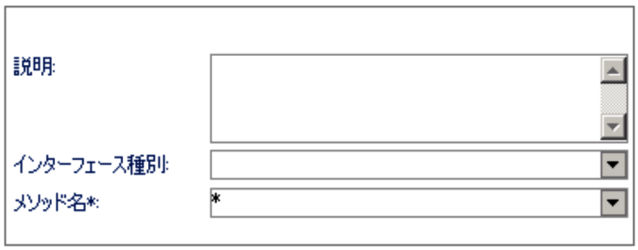


表 21-67　メソッドのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
インタフェース種別	×	インタフェース種別を指定します。 選択できる値を次に示します。 **Session Bean または Entity Bean の場合** • （空白） • Home • Remote • LocalHome • Local **Message-driven Bean の場合** • （空白） • Bean	<method-intf> インタフェース種別 </method-intf>
メソッド名	○	メソッド名を指定します。すべてのメソッドを指定する場合は，「*」を選択してください。 メソッド名に，「*」を指定した場合，［メソッド・パラメーター］の指定は無効になります。	<method-name> メソッド名 </method-name>

（凡例）　○：必須　×：任意

(19) メソッドパラメタの一覧ページ

メソッドパラメタの一覧ページを次に示します。

図 21-88 メソッドパラメタの一覧ページ

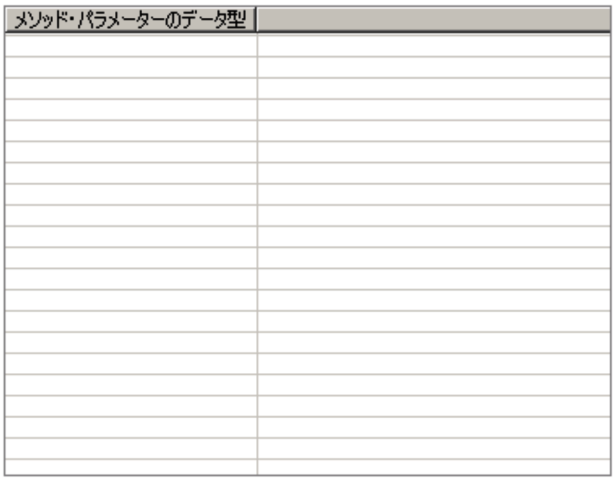


表 21-68 メソッドパラメタの一覧ページの表示項目

項目	説明	対応するタグ
メソッド・パラメーターのデータ型	パラメタのデータ型が表示されます。	<method-param> パラメタのデータ型 </method-param>

(20) パラメタのデータ型プロパティページ

パラメタのデータ型プロパティページを次に示します。

図 21-89 パラメタのデータ型プロパティページ

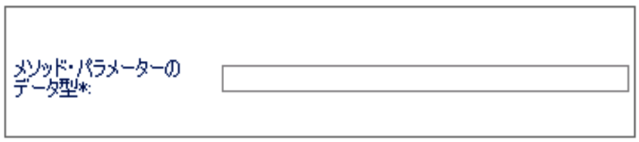


表 21-69　パラメタのデータ型プロパティページの設定項目

項目	必須／任意	説明	対応するタグ
メソッド・パラメーターのデータ型	○	パラメタのデータ型タイプを完全修飾クラス名で指定します。 配列の場合は，名称の後ろに [] を指定します。多次元配列は [] を続けて指定します。 例：short[][]	<method-param> パラメタのデータ型 </method-param>

（凡例）　○：必須

(21) ランタイムのプロパティページ

ランタイムのプロパティページを次に示します。

図 21-90　ランタイムのプロパティページ

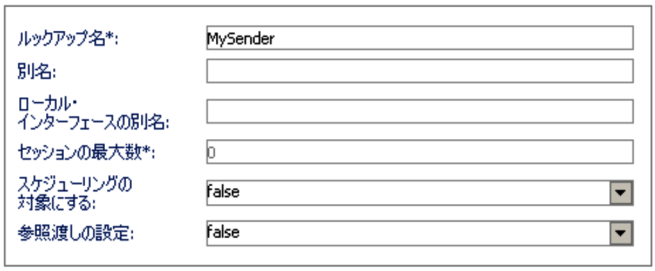


表 21-70　ランタイムのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ルックアップ名	○	クライアントから EJB をルックアップする場合に使う名前を指定します。 最大 255 文字で指定してください。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_），およびピリオド（.）です。 名前の先頭や末尾にピリオドは指定できません。	<lookup-name> ルックアップ名 </lookup-name>

項目	必須／任意	説明	対応するタグ
別名	×	別名を指定します。 最大 255 文字で指定してください。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_），ピリオド（.），ハイフン（-），および区切り文字としてスラッシュ（/）です。 サーバで，usrconf.properties ファイルの ejbserver.cui.optionalname.enabled キーが true に設定されていない場合，指定値は無効となります。 また，次の文字列は使用できません。 • 「/」だけ，または「/」が連続している文字列 • 先頭または末尾に「.」および「/」が指定されている文字列 • 「/」の前や後に「.」が指定されている文字 • 「HITACHI_EJB」で始まる文字列	<optional-name> 別名 </optional-name>
ローカル・インターフェースの別名	×	ローカルインタフェースの別名を指定します。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_），ピリオド（.），ハイフン（-），および区切り文字としてスラッシュ（/）です。 また，次の文字列は使用できません。 • 「/」だけ，または「/」が連続している文字列 • 先頭または末尾に「.」および「/」が指定されている文字列 • 「/」の前や後に「.」が指定されている文字 • 「HITACHI_EJB」で始まる文字列	<local-optional-name> ローカルインタフェースの別名 </local-optional-name>
セッションの最大数	○※1	セッションの最大数を，「0（無制限）」または「1」から「2147483647」の整数で指定します。 ただし，Stateless Session Bean の場合は指定できません。	<maximum-sessions> セッションの最大数 </maximum-sessions>
スケジューリングの対象にする	×	Bean をスケジューリングの対象にするかどうかを指定します。CTM を使用している場合にだけ指定できます。※2 指定できる値を次に示します。 • （空白） • true • false	<enable-scheduling> スケジューリングの対象にする </enable-scheduling>

項目	必須／任意	説明	対応するタグ
参照渡しの設定	×	Bean 単位でのデータの参照渡しを行うかどうかを指定します。 指定できる値を次に示します。 • （空白） • true • false	<pass-by-reference> 参照渡しの設定 </pass-by-reference>

（凡例）　○：必須　×：任意

注※ 1　Stateless Session Bean の場合は表示されるだけで，編集はできません。

注※ 2　CTM は，構成ソフトウェアに Cosminexus Component Transaction Monitor を含む製品だけで利用できます。利用できる製品については，マニュアル「Cosminexus アプリケーションサーバ 概説」の「2.3.1　製品と構成ソフトウェアの対応」を参照してください。

（22）Stateless Session Bean のプロパティページ

Stateless Session Bean のプロパティページを次に示します。

図 21-91　Stateless Session Bean のプロパティページ

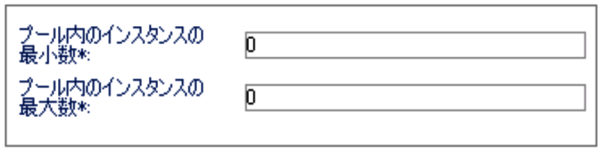


表 21-71　Stateless Session Bean のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
プール内のインスタンスの最小数	○	プール内のインスタンスの最小数を，「0（無制限）」または「1」以上の整数で指定します。 ただし，［プール内のインスタンスの最小数］は［プール内のインスタンスの最大数］に指定した値以下の値を設定してください。	<minimum> プール内のインスタンスの最小数 </minimum>
プール内のインスタンスの最大数	○	プール内のインスタンスの最大数を，「0（無制限）」または「1」以上の整数で指定します。 ただし，［プール内のインスタンスの最大数］は［プール内のインスタンスの最小数］に指定した値以上の値を設定してください。	<maximum> プール内のインスタンスの最大数 </maximum>

（凡例）　○：必須

（23）Stateful Session Bean のプロパティページ

Stateful Session Bean のプロパティページを次に示します。

図 21-92　Stateful Session Bean のプロパティページ

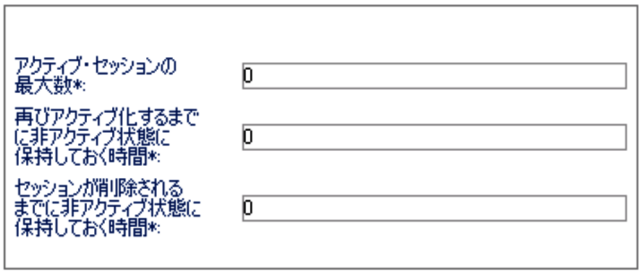


表 21-72　Stateful Session Bean のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
アクティブ・セッションの最大数	○	アクティブセッションの最大数を，「0（無制限）」または「1」以上の整数で指定します。 ただし，［アクティブ・セッションの最大数］は［セッションの最大数］に指定した値以下の値を設定してください。	<maximum-active-sessions> アクティブセッションの最大数 </maximum-active-sessions>
再びアクティブ化するまでに非アクティブ状態に保持しておく時間	○	再びアクティブ化するまでに非アクティブ状態に保持しておく時間を，「1（分）」から「2147483647（分）」の整数で指定します。 無制限の時間を設定する場合は，「0」を指定します。 ここで指定した時間が経過するまでに passive 状態の Stateful Session Bean が活性化されない場合，この Bean は削除されます。	<inactivity-timeout> 再びアクティブ化するまでに非アクティブ状態に保持しておく時間 </inactivity-timeout>
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	○	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，「1（分）」から「2147483647（分）」の整数で指定します。 無制限の時間を設定する場合は，「0」を指定します。 ここで指定された時間の間，method-ready 状態の Stateful Session Bean が一度も使用されない場合，この Bean は削除されます。	<removal-timeout> 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 </removal-timeout>

（凡例）　○：必須

（24）CTM 連携のプロパティページ

CTM 連携のプロパティページを次に示します。このページは，CTM を使用していない場合には不活性になります。CTM は，構成ソフトウェアに Cosminexus Component Transaction Monitor を含む製品だけで利用できます。利用できる製品については，マニュアル「Cosminexus アプリケーションサーバ 概説」の「2.3.1　製品と構成ソフトウェアの対応」を参照してください。

図 21-93　CTM 連携のプロパティページ

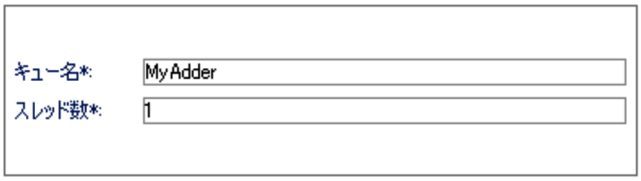


表 21-73　CTM 連携のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
キュー名	○	CTM 連携を行うキュー名称を指定します。 CTM 連携の追加で，CTM 連携のプロパティページが表示された場合，初期値として Session Bean の表示名（<display-name> タグの値）が設定されています。	<queue-name> キュー名 </queue-name>
スレッド数	○	CTM がアプリケーションを呼び出すために用意するスレッド数を，「1」から「127」の範囲で指定します。	<parallel-count> スレッド数 </parallel-count>

（凡例）　○：必須

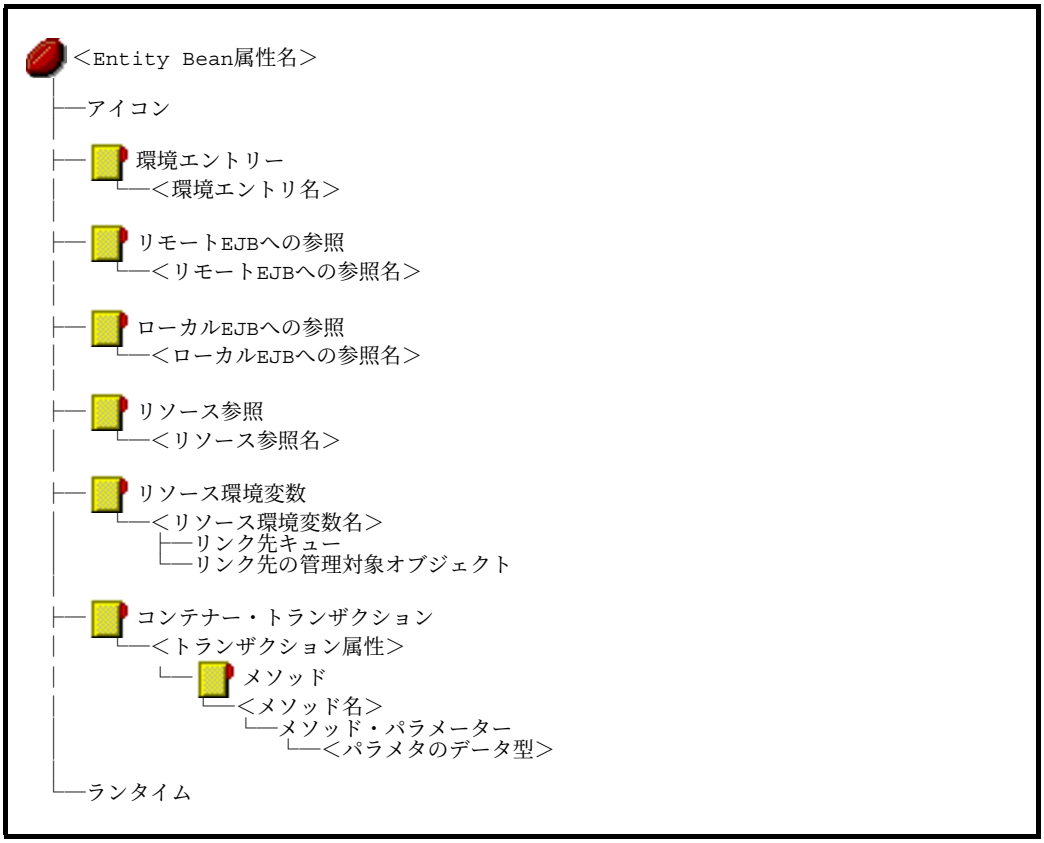
21.7　アプリケーション統合属性ファイル（Entity Bean 属性）

アプリケーション統合属性ファイルの［EJB-JAR］タブを選択すると，［EJB-JAR］フォームページが表示されます。［EJB-JAR］フォームページの［エレメント］に表示されている，［< Entity Bean 名>］を選択して，Entity Bean 属性のツリーを表示します。

21.7.1　Entity Bean 属性のツリー

エディタエリアの［エレメント］下に，編集する Entity Bean 属性の構成がツリー形式で表示されます。

Entity Bean 属性のツリー構成を次に示します。



Entity Bean 属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。Entity Bean 属性のツリーノードとコンテキストメニューについて，次に説明します。

(1) Entity Bean 属性のツリーノード

Entity Bean 属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。なお，下記以外のノードの詳細については，「21.6　アプリケーション統合属性ファイル（Session Bean 属性）」を参照してください。

表 21-74　Entity Bean 属性ファイルのツリーノード

ノード	説明	対応するタグ名	プロパティページ
＜ Entity Bean 属性名＞	Entity Bean 属性を示すノードです。 選択すると，Entity Bean のプロパティページが表示されます。	<hitachi-entity-bean-property>	21.7.2　(1)
ランタイム	ランタイムを示すノードです。 選択すると，ランタイムのプロパティページが表示されます。	<entity-runtime>	21.7.2　(2)

(2) コンテキストメニュー操作

Entity Bean 属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-75　Entity Bean 属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜ Entity Bean 属性名＞	［追加］-［アイコン］	Entity Bean 属性にアイコンが追加されます。 アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.4.2　(2)
	［追加］-［環境エントリー］	Entity Bean 属性に環境エントリが追加されます。 環境エントリのプロパティページが表示されます。	21.6.2　(4)
	［追加］-［リモート EJB］	Entity Bean 属性にリモート EJB への参照が追加されます。 リモート EJB への参照のプロパティページが表示されます。	21.6.2　(6)
	［追加］-［ローカル EJB］	Entity Bean 属性にローカル EJB への参照が追加されます。 ローカル EJB への参照のプロパティページが表示されます。	21.6.2　(8)

ノード	コンテキストメニュー	説明	プロパティページ
	［追加］-［リソース参照］	Entity Bean 属性にリソース参照が追加されます。 リソース参照のプロパティページが表示されます。	21.6.2 (10)
	［追加］-［リソース環境変数］	Entity Bean 属性にリソース環境変数が追加されます。 リソース環境変数のプロパティページが表示されます。	21.6.2 (12)
	［追加］-［コンテナー・トランザクション］	Entity Bean 属性にコンテナトランザクションが追加されます。 コンテナトランザクションのプロパティページが表示されます。	21.6.2 (16)
アイコン	［削除］	選択しているノードが削除されます。	－

（凡例） －：該当なし

21.7.2 Entity Bean 属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に，編集する Entity Bean 属性のプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアの表示が切り替わります。

Entity Bean 属性ファイルには，次のプロパティページがあります。

- Entity Bean のプロパティページ
- ランタイムのプロパティページ
- そのほかのプロパティページ

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

(1) Entity Bean のプロパティページ

Entity Bean のプロパティページを次に示します。

図 21-94　Entity Bean のプロパティページ

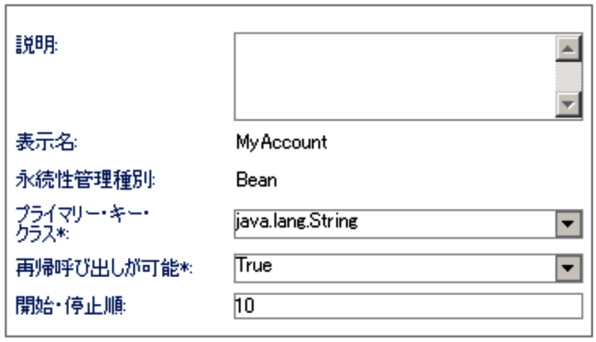


表 21-76　Entity Bean のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
表示名	－	表示名が表示されます。	<display-name> 表示名 </display-name>
永続性管理種別	－	永続性管理種別が表示されます。 永続性管理種別が「Bean」の場合は，［Bean］が表示されます。 永続性管理種別が「Container」の場合は，［Container］が表示されます。	<persistence-type> 永続性管理種別 </persistence-type>

項目	必須／任意	説明	対応するタグ
プライマリー・キー・クラス	○	単一プライマリキーまたは複合プライマリキーとして指定できるクラスまたはインタフェースを指定します。 永続性管理種別が「Bean」の場合は，具象クラス，抽象クラスおよびインタフェースが指定できます。 永続性管理種別が「Container」の場合は，具象クラスだけが指定できます。 • 単一プライマリキーの場合 次のクラスを選択します。 java.lang.Boolean java.lang.Byte java.lang.Character java.lang.Short java.lang.Integer java.lang.Long java.lang.Float java.lang.Double java.lang.String java.lang.Object • 複合プライマリキーの場合 次のクラスまたはインタフェースを指定します。 **クラスの場合** 「java.lang.Object」「java.io.Serializable」を実装し，「boolean equals(Object obj)」と「int hashCode()」をオーバーライドしたクラス（親クラスが「java.io.Serializable」を実装していてもよい） **インタフェースの場合** インタフェースを実装するクラスが「java.io.Serializable」を実装し，「boolean equals(Object obj)」と「int hashCode()」をオーバーライドしたクラス（親クラスがjava.io.Serializable を実装していてもよい）	<prim-key-class> プライマリキークラス </prim-key-class>
再帰呼び出しが可能	○	再帰呼び出しが可能かどうかを指定します。 指定できる値を次に示します。 • True • False	<reentrant> 再帰呼び出しが可能 </reentrant>
開始・停止順	×	開始・停止順を「0」から「2147483647」の整数で指定します。 指定がない場合，「10」が仮定されます。	<start-order> 開始・停止順 </start-order>

（凡例） ○：必須　×：任意　－：変更不可

(2) ランタイムのプロパティページ

ランタイムのプロパティページを次に示します。

図 21-95　ランタイムのプロパティページ

表 21-77　ランタイムのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ルックアップ名	○	クライアントから EJB をルックアップする場合に使う名前を指定します。 最大 255 文字で指定してください。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_），およびピリオド（.）です。 名前の先頭や末尾にピリオドは指定できません。	<lookup-name> ルックアップ名 </lookup-name>

項目	必須／任意	説明	対応するタグ
別名	×	別名を指定します。 最大 255 文字で指定してください。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_），ピリオド（.），ハイフン（-），および区切り文字としてスラッシュ（/）です。 サーバで，usrconf.properties ファイルの ejbserver.cui.optionalname.enabled キーに true に設定されていていない場合，指定した別名は無効となります。 また，次の文字列は使用できません。 •「/」だけ，または「/」が連続している文字列 • 先頭または末尾に「.」および「/」が指定されている文字列 •「/」の前や後に「.」が指定されている文字 •「HITACHI_EJB」で始まる文字列	<optional-name> 別名 </optional-name>
ローカル・インターフェースの別名	×	ローカルインタフェースの別名を指定します。 指定できる文字は，英数字（0 ～ 9，A ～ Z，a ～ z），アンダースコア（_），ピリオド（.），ハイフン（-），および区切り文字としてスラッシュ（/）です。 また，次の文字列は使用できません。 •「/」だけ，または「/」が連続している文字列 • 先頭または末尾に「.」および「/」が指定されている文字列 •「/」の前や後に「.」が指定されている文字 •「HITACHI_EJB」で始まる文字列	<local-optional-name> ローカルインタフェースの別名 </local-optional-name>
インスタンスの最大数	○	インスタンスの最大数を，「0（無制限）」または「1」から「2147483647」の整数で指定します。	<maximum-instances> インスタンスの最大数 </maximum-instances>
プール内のインスタンスの最小数	○	プール内のインスタンスの最小数を，「0（無制限）」または「1」以上の整数で指定します。ただし，［プール内のインスタンスの最小数］は［プール内のインスタンスの最大数］に指定した値以下の値を設定してください。	<minimum> プール内のインスタンスの最小数 </minimum>

項目	必須／任意	説明	対応するタグ
プール内のインスタンスの最大数	○	プール内のインスタンスの最大数を，「0（無制限）」または「1」以上の整数で指定します。ただし，［プール内のインスタンスの最大数］は［プール内のインスタンスの最小数］に指定した値以上の値を設定してください。 また，［プール内のインスタンスの最大数］は［インスタンスの最大数］に指定した値以下の値を設定してください。	<maximum> プール内のインスタンスの最大数 </maximum>
データのキャッシュ方法	○	データのキャッシュ方法を指定します。 指定できる値を次に示します。 • full-caching • caching • no-caching	<caching-model> データのキャッシュ方法 </caching-model>
EJB オブジェクトの存在時間	×	EJB オブジェクトの存在時間を「0（秒）」から「2147483647（秒）」の整数で指定します。 「0」を指定した場合，タイムアウトが実行されません。 指定がない場合，「0」が仮定されます。	<entity-timeout> EJB ノードの存在時間 </entity-timeout>
参照渡しの設定	×	Bean 単位でのデータの参照渡しを行うかどうかを指定します。 指定できる値を次に示します。 •（空白） • true • false	<pass-by-reference> 参照渡しの設定 </pass-by-reference>

（凡例）　○：必須　×：任意

(3) そのほかのプロパティページ

Entity Bean のプロパティページ，およびランタイムのプロパティページ以外のプロパティページは，Session Bean のプロパティの設定と同じです。プロパティページの詳細については，「21.6.2　Session Bean 属性設定ページ」を参照してください。

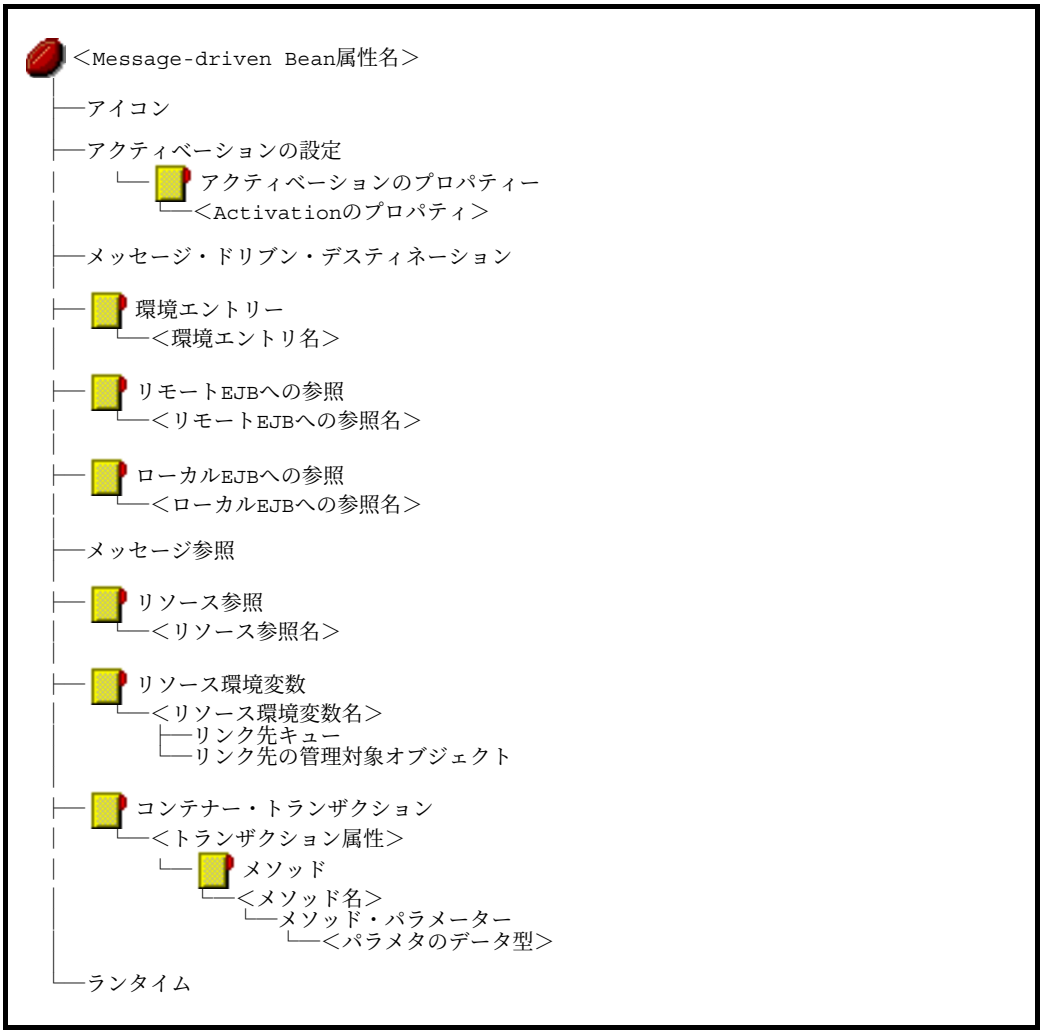
21.8　アプリケーション統合属性ファイル（Message-driven Bean 属性）

アプリケーション統合属性ファイルの［EJB-JAR］タブを選択すると，［EJB-JAR］フォームページが表示されます。［EJB-JAR］フォームページの［エレメント］に表示されている，［＜ Message-driven Bean 名＞］を選択して，Message-driven Bean 属性のツリーを表示します。

21.8.1　Message-driven Bean 属性のツリー

エディタエリアの［エレメント］下に編集する Message-driven Bean 属性の構成が，ツリー形式で表示されます。

Message-driven Bean 属性のツリー構成を次に示します。



Message-driven Bean 属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。Message-driven Bean 属性のツリーノードとコンテキストメニューについて，次に説明します。

（1） Message-driven Bean 属性のツリーノード

Message-driven Bean 属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。なお，下記以外のノードの詳細については，「21.6　アプリケーション統合属性ファイル（Session Bean 属性）」を参照してください。

表 21-78　Message-driven Bean 属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜Message-driven Bean 属性名＞	Message-driven Bean 属性を示すノードです。 選択すると，Message-driven Bean 属性のプロパティページが表示されます。	<hitachi-message-bean-property>	21.8.2　(1)
アクティベーションの設定	Activation の設定を示すノードです。 選択すると，Activation の設定のプロパティページが表示されます。	<activation-config>	21.8.2　(2)
アクティベーションのプロパティー	Activation のプロパティをまとめるノードです。 選択すると，Activation のプロパティの一覧ページが表示されます。	－	21.8.2　(3)
＜Activation のプロパティ＞	Activation のプロパティを示すノードです。 選択すると，Activation のプロパティのプロパティページが表示されます。	<activation-config-property>	21.8.2　(4)
メッセージ・ドリブン・デスティネーション	Message-driven Bean のデスティネーション定義を示すノードです。 選択すると，Message-driven Bean のデスティネーション定義のプロパティページが表示されます。	<message-driven-destination>	21.8.2　(5)
メッセージ参照	メッセージ参照を示すノードです。 選択すると，メッセージ参照のプロパティページが表示されます。	<message-ref>	21.8.2　(6)
ランタイム	ランタイムを示すノードです。選択すると，ランタイムのプロパティページが表示されます。	<message-runtime>	21.8.2　(7)

(2) コンテキストメニュー操作

Message-driven Bean 属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。なお，下記以外のノードの詳細については，「21.6　アプリケーション統合属性ファイル（Session Bean 属性）」を参照してください。

表 21-79 Message-driven Bean 属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜Message-driven Bean 属性名＞	［追加］-［アイコン］	Message-driven Bean 属性にアイコンが追加されます。 アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.4.2 (2)
	［追加］-［アクティベーションの設定］	Message-driven Bean 属性に Activation の設定が追加されます。 Activation の設定のプロパティページが表示されます。	21.8.2 (2)
	［追加］-［メッセージ・ドリブン・デスティネーション］	Message-driven Bean 属性に Message-driven Bean のデスティネーション定義のプロパティページが追加されます。 Message-driven Bean のデスティネーション定義のプロパティページが表示されます。すでに，Message-driven Bean のデスティネーション定義がある場合は選択できません。	21.8.2 (5)
	［追加］-［環境エントリー］	Message-driven Bean 属性に環境エントリが追加されます。 環境エントリのプロパティページが表示されます。	21.6.2 (4)
	［追加］-［リモート EJB への参照］	Message-driven Bean 属性にリモート EJB への参照が追加されます。 リモート EJB への参照のプロパティページが表示されます。	21.6.2 (6)
	［追加］-［ローカル EJB への参照］	Message-driven Bean 属性にローカル EJB への参照が追加されます。 ローカル EJB への参照のプロパティページが表示されます。	21.6.2 (8)
	［追加］-［メッセージ参照］	Message-driven Bean 属性にメッセージ参照が追加されます。 メッセージ参照のプロパティページが表示されます。 すでに，メッセージ参照がある場合は不活性となり追加できません。	21.8.2 (6)
	［追加］-［リソース参照］	Message-driven Bean 属性にリソース参照が追加されます。 リソース参照のプロパティページが表示されます。	21.6.2 (10)
	［追加］-［リソース環境変数］	Message-driven Bean 属性にリソース環境変数が追加されます。 リソース環境変数のプロパティページが表示されます。	21.6.2 (12)

ノード	コンテキストメニュー	説明	プロパティページ
	［追加］-［コンテナー・トランザクション］	Message-driven Bean 属性にコンテナトランザクションが追加されます。 コンテナトランザクションのプロパティページが表示されます。	21.6.2 (16)
アイコン	［削除］	選択しているノードが削除されます。	－
アクティベーションの設定	［追加］-［アクティベーションのプロパティー］	Message-driven Bean 属性に Activation のプロパティが追加されます。 Activation のプロパティのプロパティページが表示されます。	21.8.2 (4)
	［削除］	選択しているノードが削除されます。	－
アクティベーションのプロパティー	［追加］-［アクティベーションのプロパティー］	Message-driven Bean 属性に Activation のプロパティが追加されます。 Activation のプロパティのプロパティページが表示されます。	21.8.2 (4)
＜ Activation のプロパティ＞	［削除］	選択しているノードが削除されます。	－
メッセージ・ドリブン・デスティネーション	［削除］	選択しているノードが削除されます。	－
メッセージ参照	［削除］	選択しているノードが削除されます。	－
ランタイム	－	－	－

（凡例）　－：該当なし

21.8.2　Message-driven Bean 属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に編集する Message-driven Bean 属性ファイルのプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアの表示が切り替わります。

Message-driven Bean 属性には，次のプロパティページがあります。

- Message-driven Bean のプロパティページ
- Activation の設定のプロパティページ
- Activation のプロパティの一覧ページ

- Activation のプロパティのプロパティページ
- Message-driven Bean のデスティネーション定義のプロパティページ
- メッセージ参照のプロパティページ
- ランタイムのプロパティページ
- そのほかのプロパティページ

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

（1） Message-driven Bean のプロパティページ

Message-driven Bean のプロパティページを次に示します。

図 21-96　Message-driven Bean のプロパティページ

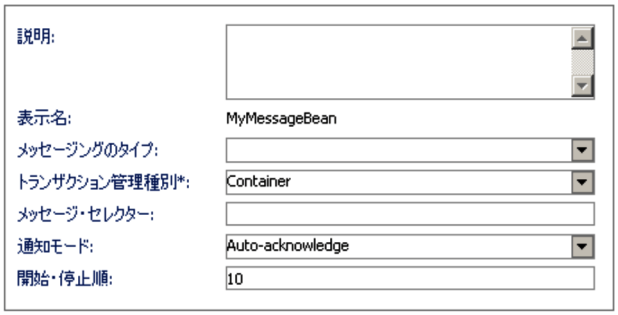


表 21-80　Message-driven Bean のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
表示名	－	表示名が表示されます。	<display-name> 表示名 </display-name>

項目	必須／任意	説明	対応するタグ
メッセージングのタイプ	×	Message-driven Bean のメッセージリスナインタフェースを指定します。 指定がない場合，サーバ側で実行時に「javax.jms.MessageListener」が仮定されます。 指定できる値は，<message-ref> - <connection-destination> - <resource-adapter> に指定された Inbound リソースアダプタに含まれるメッセージリスナのタイプです。 メッセージリスナのタイプは，Connector 属性の <inbound-resourceadapter> - <messageadapter> - <message-listener> - <message-listener-type> の値になります。	<messaging-type> メッセージングのタイプ <messaging-type>
トランザクション管理種別	○	トランザクション管理種別を指定します。 指定できる値を次に示します。 • Bean • Container	<transaction-type> トランザクション管理種別 </transaction-type>
メッセージ・セレクター	×	メッセージセレクタを指定します。	<message-selector> メッセージセレクタ </message-selector>
通知モード	×	通知モードを指定します。 デフォルトの場合，または指定を省略した場合，「Auto-acknowledge」が仮定されます。 指定できる値を次に示します。 •（空白） • Auto-acknowledge • Dups-ok-acknowledge	<acknowledge-mode> 通知モード </acknowledge-mode>
開始・停止順	×	開始・停止順を，「0」から「2147483647」の整数で指定します。 指定がない場合，「10」が仮定されます。	<start-order> 開始・停止順 </start-order>

（凡例） ○：必須　×：任意　－：変更不可

（2） Activation の設定のプロパティページ

Activation の設定のプロパティページを次に示します。

図 21-97　Activation の設定のプロパティページ

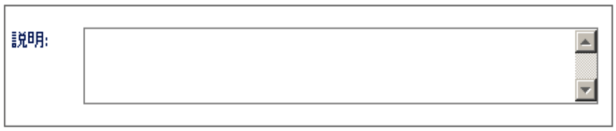


表 21-81　Activation の設定のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>

（凡例）　×：任意

（3） Activation のプロパティの一覧ページ

Activation のプロパティの一覧ページを次に示します。

図 21-98　Activation のプロパティの一覧ページ

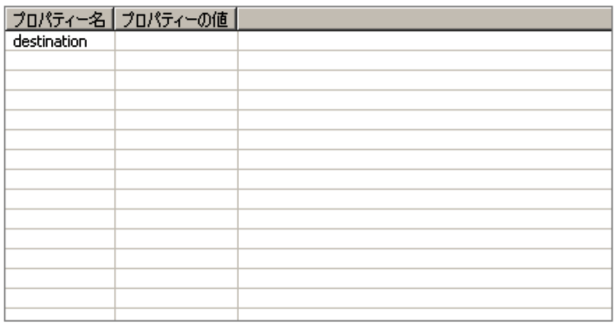


表 21-82　Activation のプロパティの一覧ページの表示項目

項目	説明	対応するタグ
プロパティー名	Activation のプロパティ名が表示されます。	<activation-config-property-name> Activation のプロパティ名 </activation-config-property-name>
プロパティーの値	Activation のプロパティの値が表示されます。	<activation-config-property-value> Activation のプロパティの値 </activation-config-property-value>

（4） Activation のプロパティのプロパティページ

Activation のプロパティのプロパティページを次に示します。

図 21-99　Activation のプロパティのプロパティページ

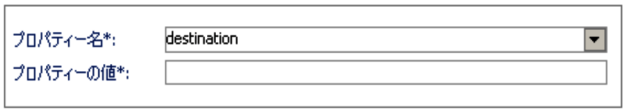


表 21-83　Activation のプロパティのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
プロパティー名	○	Activation のプロパティ名を指定します。 指定できる値を次に示します。 • destination • destinationType • 必須コンフィグレーションプロパティ名※	<activation-config-property-name> Activation のプロパティ名 </activation-config-property-name>
プロパティーの値	○	Activation のプロパティの値を指定します。	<activation-config-property-value> Activation のプロパティの値 </activation-config-property-value>

（凡例）　○：必須

注※　<messaging-type> に指定されたメッセージングのタイプに含まれる必須コンフィグレーションプロパティ名が表示されます。ただし，destination および destinationType と重複する値は表示されません。大文字と小文字が異なる値は，重複しない値として扱われます。

（5） Message-driven Bean のデスティネーション定義のプロパティページ

Message-driven Bean のデスティネーション定義のプロパティページを次に示します。

図 21-100　Message-driven Bean のデスティネーション定義のプロパティページ

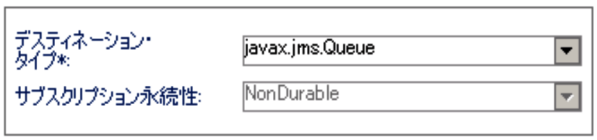


表 21-84　Message-driven Bean のデスティネーション定義のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
デスティネーション・タイプ	○	デスティネーションタイプを選択します。 選択できる値を次に示します。 • javax.jms.Queue • javax.jms.Topic	<destination-type> デスティネーションタイプ </destination-type>
サブスクリプション永続性	×※	デスティネーションタイプが「javax.jms.Topic」の場合，サブスクリプション永続性を選択します。「javax.jms.Topic」が指定されていない場合は，選択できません。 選択できる値を次に示します。 •（空白） • Durable • NonDurable	<subscription-durability> サブスクリプション永続性 </subscription-durability>

（凡例）　○：必須　×：任意
注※　デスティネーションタイプが「javax.jms.Topic」の場合は，設定は必須です。

(6) メッセージ参照のプロパティページ

メッセージ参照のプロパティページを次に示します。

図 21-101　メッセージ参照のプロパティページ

表 21-85　メッセージ参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
Connection Factory 名	○	Connection Factory 名を指定します。	<connection-factory> Connection Factory 名 </connection-factory>

項目	必須／任意	説明	対応するタグ
リソース・アダプター名	○	J2EE リソースアダプタの表示名を入力するか，またはリスト項目を選択します。 J2EE サーバにデプロイされている J2EE リソースアダプタの表示名がリスト項目として表示されます。［サーバー・エクスプローラー］ビューでも J2EE リソースアダプタの表示名一覧を参照できます。	<resource-adapter> リソースアダプタ名 </resource-adapter>
キュー名	○	キューの表示名を指定します。	<queue> キュー名 </queue>

（凡例）　○：必須

(7) ランタイムのプロパティページ

ランタイムのプロパティページを次に示します。

図 21-102　ランタイムのプロパティページ

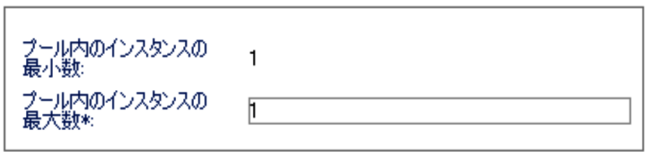


表 21-86　ランタイムのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
プール内のインスタンスの最小数	－	プール内のインスタンスの最小数「1」が表示されます。	<minimum> プール内のインスタンスの最小数 </minimum>
プール内のインスタンスの最大数	○	プール内のインスタンスの最大数を「1」から「2147483647」の整数で指定します。	<maximum> プール内のインスタンスの最大数 </maximum>

（凡例）　○：必須　－：変更不可

(8) そのほかのプロパティページ

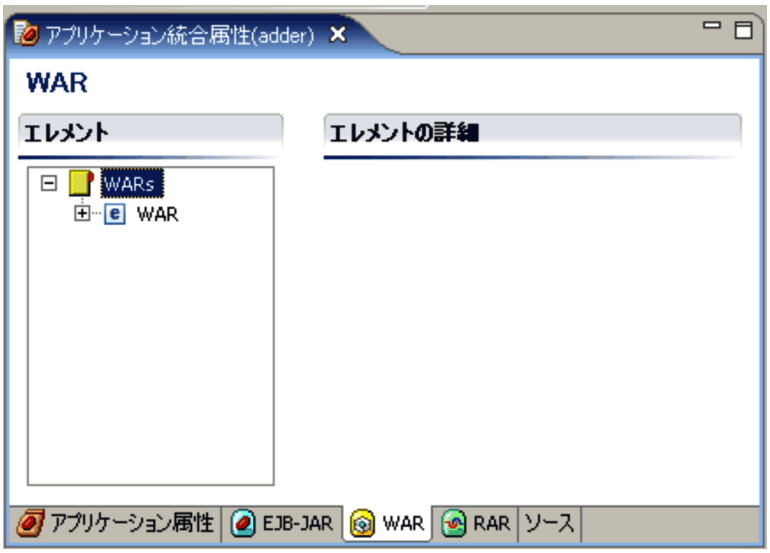
Message-driven Bean のプロパティページ，Message-driven Bean のデスティネーション定義のプロパティページ，メッセージ参照のプロパティページ，およびランタイムの

プロパティページ以外のプロパティページは，Session Bean 属性の場合と同じです。プロパティページの詳細については，「21.6.2　Session Bean 属性設定ページ」を参照してください。

21.9　アプリケーション統合属性ファイル（WAR 属性）

アプリケーション統合属性ファイルの［WAR］タブを選択すると，［WAR］フォームページが表示されます。

図 21-103　［WAR］フォームページ例



フォームページの［エレメント］の下に，WAR の構成によって，次のツリーが展開されます。

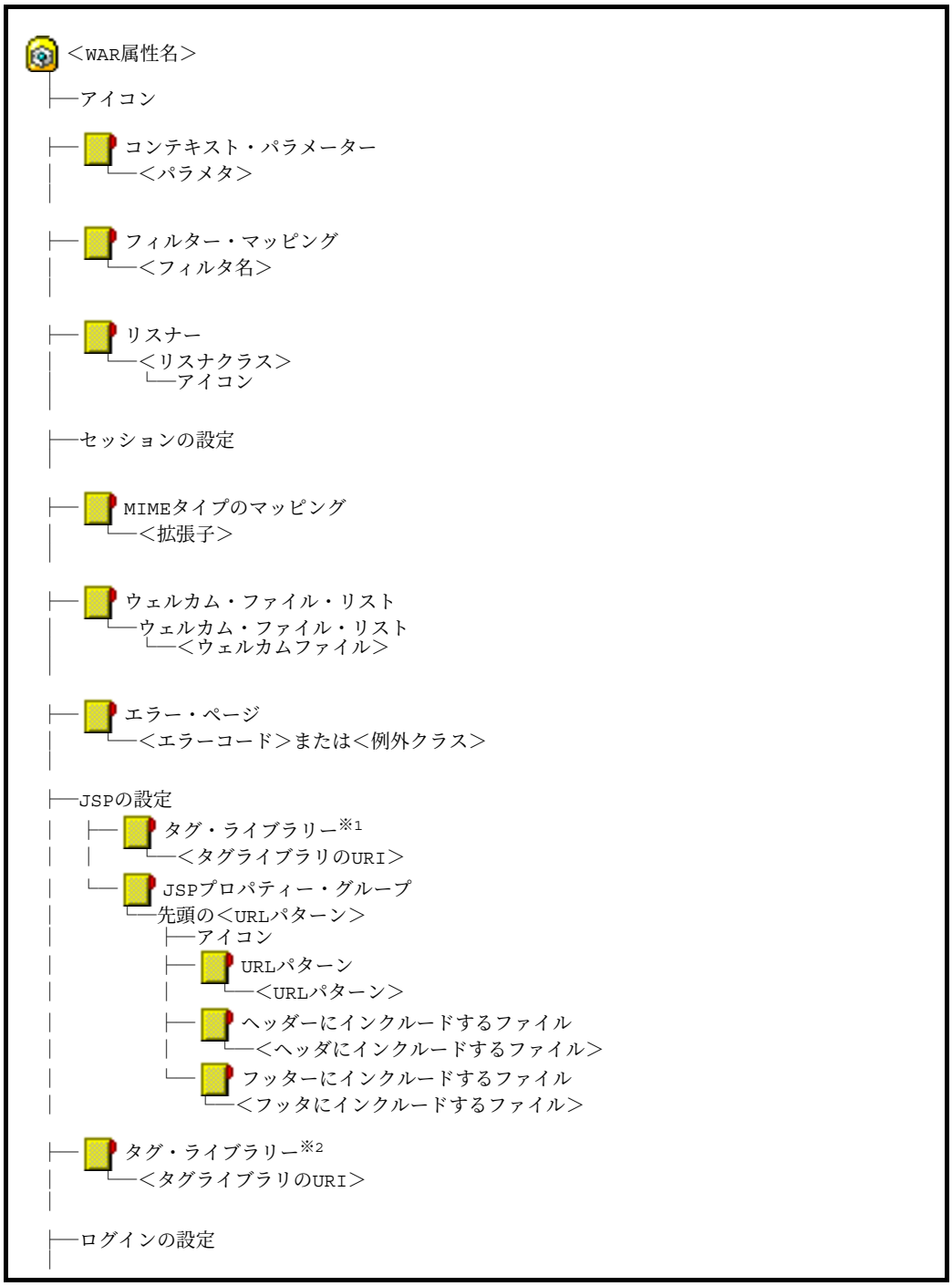
- WAR 属性
  ＜WAR 属性名＞を選択すると，WAR 属性のツリーが表示されます。WAR 属性のツリーとプロパティの設定項目については，この節で説明しています。
- フィルタ属性
  ＜フィルタ属性名＞を選択すると，フィルタ属性のツリーが表示されます。フィルタ属性のツリーとプロパティの設定項目については，「21.10　アプリケーション統合属性ファイル（フィルタ属性）」を参照してください。
- サーブレット属性
  ＜サーブレット属性名＞を選択すると，サーブレット属性のツリーが表示されます。サーブレット属性のツリーとプロパティの設定項目については，「21.11　アプリケーション統合属性ファイル（サーブレット属性）」を参照してください。

21.9.1　WAR 属性のツリー

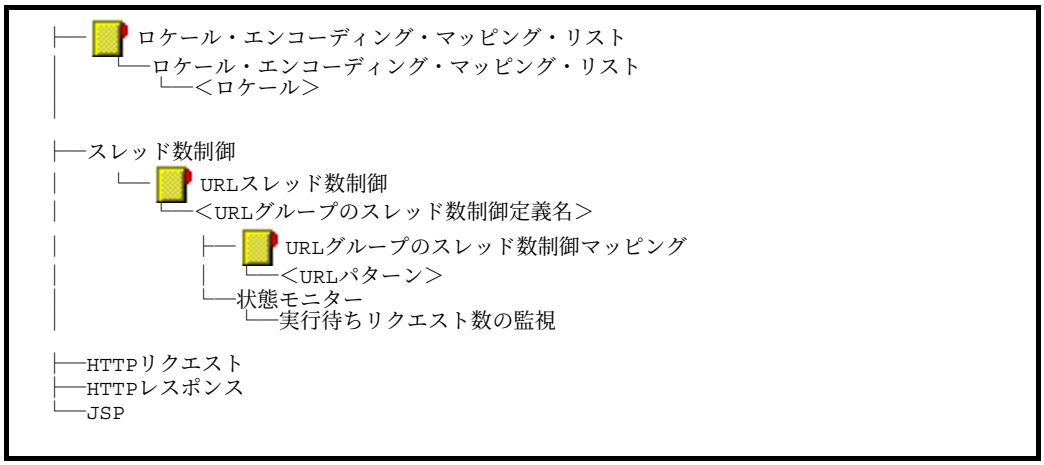
エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。エ

ディタエリアの［エレメント］下に編集する WAR 属性ファイルの構成がツリー形式で表示されます。

WAR 属性のツリー構成を次に示します。




環境エントリー
＜環境エントリ名＞
インジェクション・ターゲット※3
＜インジェクションターゲット名＞
リモートEJBへの参照
＜リモートEJBへの参照名＞
インジェクション・ターゲット※3
＜インジェクションターゲット名＞
ローカルEJBへの参照
＜ローカルEJBへの参照名＞
インジェクション・ターゲット※3
＜インジェクションターゲット名＞
リソース参照
＜リソース参照名＞
インジェクション・ターゲット※3
＜インジェクションターゲット名＞
リソース環境変数
＜リソース環境変数名＞
インジェクション・ターゲット※3
＜インジェクションターゲット名＞
リンク先キュー
リンク先の管理対象オブジェクト
永続化コンテキスト参照※3
＜永続化コンテキスト参照名＞
永続化のプロパティー
＜永続化のプロパティ名＞
インジェクション・ターゲット
＜インジェクションターゲット名＞
永続化ユニット参照※3
＜永続化ユニット参照名＞
インジェクション・ターゲット
＜インジェクションターゲット名＞
PostConstructメソッド※3
＜PostConstructメソッド名＞
PreDestroyメソッド※3
＜PreDestroyメソッド名＞



注※ 1　Servlet2.4 以降用のタグライブラリをまとめるノードです。
注※ 2　Servlet2.3 以前用のタグライブラリをまとめるノードです。
注※ 3　Servlet2.5 の場合に表示されます。

WAR 属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。WAR 属性のツリーノードとコンテキストメニューについて，次に説明します。

（1）WAR 属性のツリーノード

WAR 属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。

表 21-87　WAR 属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜ WAR 属性名＞	WAR 属性を示すノードです。 選択すると，WAR のプロパティページが表示されます。 WAR 属性がない場合，「WAR はありません」が表示されます。	<hitachi-war-property>	21.9.2　(1)
アイコン	WAR 属性のアイコンを示すノードです。 選択すると，アイコンのプロパティページが表示されます。	<icon>	21.9.2　(2)
コンテキスト・パラメーター	コンテキストパラメタをまとめるノードです。 選択すると，パラメタの一覧ページが表示されます。	—	21.9.2　(3)
＜パラメタ＞	コンテキストパラメタを示すノードです。 選択すると，パラメタのプロパティページが表示されます。	<context-param>	21.9.2　(4)

ノード	説明	対応するタグ名	プロパティページ
フィルター・マッピング	フィルタマッピングをまとめるノードです。 選択すると，フィルタマッピングの一覧ページが表示されます。	－	21.9.2 (5)
＜フィルタ名＞	フィルタマッピングを示すノードです。 選択すると，<url-pattern> が指定されている場合はフィルタマッピング（URL パターン）のプロパティページが，<servlet-name> が指定されている場合はフィルタマッピング（サーブレット名）のプロパティページが表示されます。	<filter-mapping>	21.9.2 (6), 21.9.2 (7)
リスナー	リスナをまとめるノードです。 選択すると，リスナの一覧ページが表示されます。	－	21.9.2 (8)
＜リスナクラス＞	リスナを示すノードです。 選択すると，リスナのプロパティページが表示されます。	<listener>	21.9.2 (9)
アイコン	リスナのアイコンを示すノードです。 選択すると，アイコンのプロパティページが表示されます。	<icon>	21.9.2 (2)
セッションの設定	セッションの設定を示すノードです。 選択すると，セッションの設定のプロパティページが表示されます。	<session-config>	21.9.2 (10)
MIME タイプのマッピング	MIME タイプのマッピングをまとめるノードです。 選択すると，MIME タイプのマッピングの一覧ページが表示されます。	－	21.9.2 (11)
＜拡張子＞	MIME タイプのマッピングを示すノードです。 選択すると，MIME タイプのマッピングのプロパティページが表示されます。	<mime-mapping>	21.9.2 (12)
ウェルカム・ファイル・リスト	ウェルカムファイルリストをまとめるノードです。	－	－
ウェルカム・ファイル・リスト	ウェルカムファイルリストを示すノードです。 選択すると，ウェルカムファイルリストのプロパティページが表示されます。	<welcome-file-list>	21.9.2 (13)
＜ウェルカムファイル＞	ウェルカムファイルを示すノードです。 選択すると，ウェルカムファイルのプロパティページが表示されます。	<welcome-file>	21.9.2 (14)
エラー・ページ	エラーページをまとめるノードです。 選択すると，エラーページの一覧ページが表示されます。	－	21.9.2 (15)

ノード	説明	対応するタグ名	プロパティページ
＜エラーコード＞または＜例外クラス＞	エラーページを示すノードです。 選択すると，<error-code> が指定されている場合はエラーページ (エラーコード) のプロパティページが，<exception-type> が指定されている場合は，エラーページ (例外クラス) のプロパティページが表示されます。	<error-page>	21.9.2 (16), 21.9.2 (17)
JSP の設定	JSP の設定を示すノードです。	<jsp-config>	―
タグ・ライブラリー※	タグライブラリをまとめるノードです。 選択すると，タグライブラリの一覧ページが表示されます。	―	21.9.2 (18)
＜タグライブラリの URI ＞	タグライブラリを示すノードです。 選択すると，タグライブラリのプロパティページが表示されます。	<taglib>	21.9.2 (19)
JSP プロパティー・グループ	JSP プロパティグループをまとめるノードです。 選択すると，JSP プロパティグループの一覧ページが表示されます。	―	21.9.2 (20)
先頭の＜ URL パターン＞	JSP プロパティグループを示すノードです。 選択すると，JSP プロパティグループのプロパティページが表示されます。	<jsp-property-group>	21.9.2 (21)
アイコン	JSP プロパティグループのアイコンを示すノードです。 選択すると，アイコンのプロパティページが表示されます。	<icon>	21.9.2 (2)
URL パターン	URL パターンをまとめるノードです。 選択すると，URL パターンの一覧ページが表示されます。	―	21.9.2 (22)
＜ URL パターン＞	URL パターンを示すノードです。 選択すると，URL パターンのプロパティページが表示されます。	<url-pattern>	21.9.2 (23)
ヘッダーにインクルードするファイル	ヘッダにインクルードするファイルをまとめるノードです。 選択すると，ヘッダにインクルードするファイルの一覧ページが表示されます。	―	21.9.2 (24)
＜ヘッダにインクルードするファイル＞	ヘッダにインクルードするファイルを示すノードです。 選択すると，ヘッダにインクルードするファイルのプロパティページが表示されます。	<include-prelude>	21.9.2 (25)

ノード	説明	対応するタグ名	プロパティページ
フッターにインクルードするファイル	フッタにインクルードするファイルをまとめるノードです。 選択すると，フッタにインクルードするファイルの一覧ページが表示されます。	－	21.9.2 (26)
＜フッタにインクルードするファイル＞	フッタにインクルードするファイルを示すノードです。 選択すると，フッターにインクルードするファイルのプロパティページが表示されます。	<include-coda>	21.9.2 (27)
タグ・ライブラリー※	タグライブラリをまとめるノードです。 選択すると，タグライブラリの一覧ページが表示されます。	－	21.9.2 (18)
＜タグライブラリの URI ＞	タグライブラリを示すノードです。 選択すると，タグライブラリのプロパティページが表示されます。	<taglib>	21.9.2 (19)
ログインの設定	ログインの設定を示すノードです。 選択すると，ログインの設定のプロパティページが表示されます。	<login-config>	21.9.2 (28)
環境エントリー	環境エントリをまとめるノードです。 選択すると，環境エントリの一覧のページが表示されます。	－	21.9.2 (29)
＜環境エントリ名＞	環境エントリを示すノードです。 選択すると，環境エントリのプロパティページが表示されます。	<env-entry>	21.9.2 (30)
インジェクション・ターゲット	インジェクションターゲットをまとめるノードです。 選択すると，インジェクションターゲットの一覧のページが表示されます。	－	21.9.2 (31)
＜インジェクションターゲット名＞	インジェクションターゲットを示すノードです。 選択すると，インジェクションターゲットのプロパティページが表示されます。	<injection-target>	21.9.2 (32)
リモート EJB への参照	リモート EJB への参照をまとめるノードです。 選択すると，リモート EJB への参照の一覧ページが表示されます。	－	21.9.2 (33)
＜リモート EJB への参照名＞	リモート EJB への参照を示すノードです。 <ejb-ref> 内の <ejb-ref-name> の内容が表示されます。 選択すると，リモート EJB への参照のプロパティページが表示されます。	<ejb-ref>	21.9.2 (34)

ノード	説明	対応するタグ名	プロパティページ
インジェクション・ターゲット	インジェクションターゲットをまとめるノードです。 選択すると，インジェクションターゲットの一覧のページが表示されます。	－	21.9.2 (31)
＜インジェクションターゲット名＞	インジェクションターゲットを示すノードです。 選択すると，インジェクションターゲットのプロパティページが表示されます。	<injection-target>	21.9.2 (32)
ローカル EJB への参照	ローカル EJB への参照をまとめるノードです。 選択すると，ローカル EJB への参照の一覧ページが表示されます。	－	21.9.2 (35)
＜ローカル EJB への参照名＞	ローカル EJB への参照を示すノードです。 選択すると，ローカル EJB への参照のプロパティページが表示されます。	<ejb-local-ref>	21.9.2 (36)
インジェクション・ターゲット	インジェクションターゲットをまとめるノードです。 選択すると，インジェクションターゲットの一覧のページが表示されます。	－	21.9.2 (31)
＜インジェクションターゲット名＞	インジェクションターゲットを示すノードです。 選択すると，インジェクションターゲットのプロパティページが表示されます。	<injection-target>	21.9.2 (32)
リソース参照	リソース参照をまとめるノードです。「リソース参照」が表示されます。 選択すると，リソース参照の一覧ページが表示されます。	－	21.9.2 (37)
＜リソース参照名＞	リソース参照を示すノードです。 選択すると，リソース参照のプロパティページが表示されます。	<resource-ref>	21.9.2 (38)
インジェクション・ターゲット	インジェクションターゲットをまとめるノードです。 選択すると，インジェクションターゲットの一覧のページが表示されます。	－	21.9.2 (31)
＜インジェクションターゲット名＞	インジェクションターゲットを示すノードです。 選択すると，インジェクションターゲットのプロパティページが表示されます。	<injection-target>	21.9.2 (32)
リソース環境変数	リソース環境変数をまとめるノードです。 選択すると，リソース環境変数の一覧ページが表示されます。	－	21.9.2 (39)

ノード	説明	対応するタグ名	プロパティページ
＜リソース環境変数名＞	リソース環境変数を示すノードです。 選択すると，リソース環境変数のプロパティページが表示されます。	<resource-env-ref>	21.9.2 (40)
インジェクション・ターゲット	インジェクションターゲットをまとめるノードです。 選択すると，インジェクションターゲットの一覧のページが表示されます。	―	21.9.2 (31)
＜インジェクションターゲット名＞	インジェクションターゲットを示すノードです。 選択すると，インジェクションターゲットのプロパティページが表示されます。	<injection-target>	21.9.2 (32)
リンク先キュー	リンク先キューを示すノードです。 選択すると，リンク先キューのプロパティページが表示されます。	<linked-queue>	21.9.2 (41)
リンク先の管理対象オブジェクト	リンク先の管理対象オブジェクトを示すノードです。 選択すると，リンク先の管理対象オブジェクトのプロパティページが表示されます。	<linked-adminobject>	21.9.2 (42)
永続化コンテキスト参照	永続化コンテキスト参照をまとめるノードです。 選択すると，永続化コンテキスト参照の一覧ページが表示されます。	―	21.9.2 (43)
＜永続化コンテキスト参照名＞	永続化コンテキスト参照を示すノードです。 選択すると，永続化コンテキスト参照のプロパティページが表示されます。	<persistence-context-ref>	21.9.2 (44)
永続化のプロパティー	永続化のプロパティをまとめるノードです。 選択すると，永続化のプロパティのプロパティページが表示されます。	―	21.9.2 (45)
＜永続化のプロパティ名＞	永続化のプロパティを示すノードです。 選択すると，永続化プロパティのプロパティページが表示されます。	<persistence-property>	21.9.2 (46)
インジェクション・ターゲット	インジェクションターゲットをまとめるノードです。 選択すると，インジェクションターゲットの一覧のページが表示されます。	―	21.9.2 (31)
＜インジェクションターゲット名＞	インジェクションターゲットを示すノードです。 選択すると，インジェクションターゲットのプロパティページが表示されます。	<injection-target>	21.9.2 (32)
永続化ユニット参照	永続化ユニットをまとめるノードです。 選択すると，永続化ユニット参照の一覧のページが表示されます。	―	21.9.2 (47)

ノード	説明	対応するタグ名	プロパティページ
＜永続化ユニット参照名＞	永続化ユニットを示すノードです。 選択すると，永続化ユニット参照のプロパティページが表示されます。	<persistence-unit-ref-name>	21.9.2 (48)
インジェクション・ターゲット	インジェクションターゲットをまとめるノードです。 選択すると，インジェクションターゲットの一覧のページが表示されます。	－	21.9.2 (31)
＜インジェクションターゲット名＞	インジェクションターゲットを示すノードです。 選択すると，インジェクションターゲットのプロパティページが表示されます。	<injection-target>	21.9.2 (32)
PostConstruct メソッド	PostConstruct メソッドをまとめるノードです。 選択すると，PostConstruct メソッドの一覧のページが表示されます。	－	21.9.2 (49)
＜ PostConstruct メソッド名＞	PostConstruct メソッドを示すノードです。 選択すると，PostConstruct メソッドのプロパティページが表示されます。	<post-construct>	21.9.2 (50)
PreDestroy メソッド	PreDestroy メソッドをまとめるノードです。 選択すると，PreDestroy メソッドの一覧のページが表示されます。	－	21.9.2 (51)
＜ PreDestroy メソッド名＞	PreDestroy メソッドを示すノードです。 選択すると，PreDestroy メソッドのプロパティページが表示されます。	<pre-destroy>	21.9.2 (52)
ロケール・エンコーディング・マッピング・リスト	ロケールエンコーディングマッピングリストをまとめるノードです。	－	－
ロケール・エンコーディング・マッピング・リスト	ロケールエンコーディングマッピングリストを示すノードです。 選択すると，ロケールエンコーディングマッピングの一覧ページが表示されます。	<locale-encoding-mapping-list>	21.9.2 (53)
＜ロケール＞	ロケールを示すノードです。 選択すると，ロケールエンコーディングマッピングのプロパティページが表示されます。	<local-encoding-mapping >	21.9.2 (54)
スレッド数制御	スレッド数制御を示すノードです。 選択すると，スレッド数制御のプロパティページが表示されます。	<thread-control>	21.9.2 (55)

ノード	説明	対応するタグ名	プロパティページ
URL グループのスレッド数制御	URL グループのスレッド数制御をまとめるノードです。 「URL グループのスレッド数制御」が表示されます。 選択すると，URL グループのスレッド数制御の一覧ページが表示されます。	－	21.9.2　(56)
＜URL グループのスレッド数制御定義名＞	URL グループのスレッド数制御を示すノードです。 選択すると，URL グループのスレッド数制御のプロパティページが表示されます。	<urlgroup-thread-control>	21.9.2　(57)
URL グループのスレッド数制御マッピング	URL グループのスレッド数制御マッピングをまとめるノードです。 選択すると，URL グループの スレッド数制御のマッピングの一覧ページが表示されます。	－	21.9.2　(58)
＜URL パターン＞	URL グループのスレッド数制御マッピングを示すノードです。 選択すると，URL グループのスレッド数制御マッピングのプロパティページが表示されます。	<urlgroup-thread-control-mapping>	21.9.2　(59)
状態モニター	状態モニタを示すノードです。	<stats-monitor>	－
実行待ちリクエスト数の監視	URL グループ単位の実行待ちリクエスト数の監視を示すノードです。 選択すると，実行待ちリクエスト数の監視のプロパティページが表示されます。	<waiting-request-count>	21.9.2　(60)
HTTP リクエスト	HTTP リクエストを示すノードです。 選択すると，HTTP リクエストのプロパティページが表示されます。	<http-request>	21.9.2　(61)
HTTP レスポンス	HTTP レスポンスを示すノードです。 選択すると，HTTP レスポンスのプロパティページが表示されます。	<http-response>	21.9.2　(62)
JSP	JSP を示すノードです。 選択すると，JSP のプロパティページが表示されます。	<jsp>	21.9.2　(63)

（凡例）　－：該当なし

注※　Servlet2.3 以前の場合は，WAR 属性直下の［タグ・ライブラリー］を設定してください。Servlet2.4 以降の場合は，［JSP の設定］下の［タグ・ライブラリー］などを設定してください。

(2) コンテキストメニュー操作

WAR 属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-88 WAR 属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜WAR 属性名＞	［追加］-［アイコン］	アイコンが追加されます。アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.9.2 (2)
	［追加］-［コンテキスト・パラメーター］	コンテキストパラメタが追加されます。パラメタのプロパティページが表示されます。	21.9.2 (4)
	［追加］-［フィルター・マッピング（URL パターン)］	フィルタマッピング（URL パターン）が追加されます。フィルタマッピング（URL パターン）のプロパティページが表示されます。 ただし，アプリケーション統合属性ファイルにフィルタ属性がない場合は選択できません。	21.9.2 (6)
	［追加］-［フィルター・マッピング（サーブレット名)］	フィルタマッピング（サーブレット名）が追加されます。フィルタマッピング（サーブレット名）のプロパティページが表示されます。 ただし，アプリケーション統合属性ファイルにフィルタ属性またはサーブレット属性がない場合は選択できません。	21.9.2 (7)
	［追加］-［リスナー］	リスナが追加されます。リスナのプロパティページが表示されます。	21.9.2 (9)
	［追加］-［セッションの設定］	セッションの設定が追加されます。セッションの設定のプロパティページが表示されます。 ただし，すでにセッションの設定がある場合は選択できません。	21.9.2 (10)
	［追加］-［MIME タイプのマッピング］	MIME タイプのマッピングが追加されます。MIME タイプのマッピングの一覧ページが表示されます。	21.9.2 (11)
	［追加］-［ウェルカム・ファイル・リスト］	ウェルカムファイルリスト，およびウェルカムファイルが追加されます。ウェルカムファイルのプロパティページが表示されます。	21.9.2 (14)
	［追加］-［エラー・ページ（エラー・コード)］	エラーページ（エラーコード）が追加されます。エラーページ（エラーコード）のプロパティページが表示されます。	21.9.2 (16)

ノード	コンテキストメニュー	説明	プロパティページ
	［追加］-［エラー・ページ（例外クラス）］	エラーページ（例外クラス）が追加されます。エラーページ（例外クラス）のプロパティページが表示されます。	21.9.2 (17)
	［追加］-［JSP の設定］	ツリービューに「JSP の設定」が追加されます。 ただし，すでに JSP の設定がある場合は選択できません。	－
	［追加］-［タグ・ライブラリー］※	タグライブラリが追加されます。タグライブラリのプロパティページが表示されます。 ただし，すでに＜ WAR 属性名＞，または「JSP の設定」の直下にタグライブラリがある場合は選択できません。	21.9.2 (19)
	［追加］-［ログインの設定］	ログインの設定が追加されます。ログインの設定のプロパティページが表示されます。 ただし，すでにログインの設定がある場合は選択できません。	21.9.2 (28)
	［追加］-［環境エントリー］	環境エントリが追加されます。環境エントリのプロパティページが表示されます。	21.9.2 (30)
	［追加］-［リモート EJB への参照］	リモート EJB への参照が追加されます。リモート EJB への参照のプロパティページが表示されます。	21.9.2 (34)
	［追加］-［ローカル EJB への参照］	ローカル EJB への参照が追加されます。ローカル EJB への参照のプロパティページが表示されます。	21.9.2 (36)
	［追加］-［リソース参照］	WAR 属性にリソース参照が追加されます。リソース参照のプロパティページが表示されます。	21.9.2 (38)
	［追加］-［リソース環境変数］	リソース環境変数が追加されます。リソース環境変数のプロパティページが表示されます。	21.9.2 (40)
	［追加］-［永続化コンテキスト参照］	永続化コンテキスト参照が追加されます。永続化コンテキスト参照のプロパティページが表示されます。	21.9.2 (44)
	［追加］-［永続化ユニット参照］	永続化ユニット参照が追加されます。永続化ユニット参照のプロパティページが表示されます。	21.9.2 (48)

ノード	コンテキストメニュー	説明	プロパティページ
	［追加］-［PostConstruct メソッド］	PostConstruct メソッドが追加されます。PostConstruct メソッドのプロパティページが表示されます。	21.9.2 (50)
	［追加］-［PreDestroy メソッド］	PreDestroy メソッドが追加されます。PreDestroy メソッドのプロパティページが表示されます。	21.9.2 (52)
	［追加］-［ロケール・エンコーディング・マッピング・リスト］	ロケールエンコーディングマッピングリストおよびロケールエンコーディングマッピングが追加されます。 ロケールエンコーディングマッピングのプロパティページが表示されます。	21.9.2 (54)
	［追加］-［スレッド数制御］	スレッド数制御が追加されます。スレッド数制御のプロパティページが表示されます。 ただし，すでにスレッド数制御がある場合は選択できません。	21.9.2 (55)
	［追加］-［HTTP リクエスト］	HTTP リクエストが追加されます。HTTP リクエストのプロパティページが表示されます。 ただし，すでに HTTP リクエストがある場合は選択できません。	21.9.2 (61)
	［追加］-［HTTP レスポンス］	HTTP レスポンスが追加されます。HTTP レスポンスのプロパティページが表示されます。 ただし，すでに HTTP レスポンスがある場合は選択できません。	21.9.2 (62)
	［追加］-［JSP］	JSP が追加されます。JSP のプロパティページが表示されます。 ただし，すでに JSP がある場合は選択できません。	21.9.2 (63)
アイコン	［削除］	WAR 属性からアイコンが削除されます。	－
コンテキスト・パラメーター	［追加］-［コンテキスト・パラメーター］	コンテキストパラメタが追加されます。パラメタのプロパティページが表示されます。	21.9.2 (4)
	［削除］	WAR 属性から，すべてのコンテキストパラメタが削除されます。	－
＜パラメタ＞	［削除］	WAR 属性から，選択したコンテキストパラメタが削除されます。	－

ノード	コンテキストメニュー	説明	プロパティページ
フィルター・マッピング	［追加］-［フィルター・マッピング（URL パターン）］	フィルタマッピング（URL パターン）が追加されます。フィルタマッピング（URL パターン）のプロパティページが表示されます。 ただし，アプリケーション統合属性ファイルにフィルタ属性がない場合は選択できません。	21.9.2 (6)
	［追加］-［フィルター・マッピング（サーブレット名）］	フィルタマッピング（サーブレット名）が追加されます。フィルタマッピング（サーブレット名）のプロパティページが表示されます。 ただし，アプリケーション統合属性ファイルにフィルタ属性がない場合は選択できません。	21.9.2 (7)
	［削除］	WAR 属性から，すべてのフィルタマッピングが削除されます。	－
＜フィルタ名＞	［削除］	WAR 属性から，選択したフィルタマッピングが削除されます。	－
リスナー	［追加］-［リスナー］	リスナが追加されます。リスナのプロパティページが表示されます。	21.9.2 (9)
	［削除］	WAR 属性から，すべてのリスナが削除されます。	－
＜リスナクラス＞	［追加］-［アイコン］	アイコンが追加されます。アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.9.2 (2)
	［削除］	WAR 属性から，選択したリスナが削除されます。	－
アイコン	［削除］	リスナからアイコンが削除されます。	－
セッションの設定	［削除］	WAR 属性からセッションの設定が削除されます。	－
MIME タイプのマッピング	［追加］-［MIME タイプのマッピング］	MIME タイプのマッピングが追加されます。MIME タイプのマッピングの一覧ページが表示されます。	21.9.2 (11)
	［削除］	WAR 属性から，すべての MIME タイプのマッピングが削除されます。	－
＜拡張子＞	［削除］	選択した MIME タイプのマッピングが削除されます。	－

ノード	コンテキストメニュー	説明	プロパティページ
ウェルカム・ファイル・リスト	［追加］-［ウェルカム・ファイル・リスト］	「ウェルカム・ファイル・リスト」と＜ウェルカムファイル＞が追加されます。 ウェルカムファイルのプロパティページが表示されます。	21.9.2 (14)
	［削除］	WAR 属性から，すべてのウェルカムファイルリストが削除されます。	―
ウェルカム・ファイル・リスト	［追加］-［ウェルカム・ファイル］	ウェルカムファイルが追加されます。ウェルカムファイルのプロパティページが表示されます。	21.9.2 (14)
	［削除］	WAR 属性から，ウェルカムファイルリストとリストに含まれるすべてのウェルカムファイルが削除されます。	―
＜ウェルカムファイル＞	［削除］	ウェルカムファイルリストから，選択したウェルカムファイルが削除されます。 ただし，ウェルカムファイルリストにウェルカムファイルが一つもない場合は選択できません。	―
エラー・ページ	［追加］-［エラー・ページ(エラー・コード)］	エラーページ（エラーコード）が追加されます。エラーページ（エラーコード）のプロパティページが表示されます。	21.9.2 (16)
	［追加］-［エラー・ページ(例外クラス)］	エラーページ（例外クラス）が追加されます。エラーページ（例外クラス）のプロパティページが表示されます。	21.9.2 (17)
	［削除］	WAR 属性から，すべてのエラーページが削除されます。	―
＜エラーコード＞または＜例外クラス＞	［削除］	WAR 属性から，選択したエラーページが削除されます。	―
JSP の設定	［追加］-［タグ・ライブラリー］※	Servlet2.4 以降用のタグライブラリが追加されます。 タグライブラリのプロパティページが表示されます。 ただし，すでに JSP の設定にタグライブラリがある場合は選択できません。	21.9.2 (19)
	［追加］-［JSP プロパティー・グループ］	JSP プロパティグループが追加されます。JSP プロパティグループのプロパティページが表示されます。	21.9.2 (21)
	［削除］	WAR 属性から JSP の設定が削除されます。	―

ノード	コンテキストメニュー	説明	プロパティページ
タグ・ライブラリー	［追加］-［タグ・ライブラリー］	タグライブラリが追加されます。タグライブラリのプロパティページが表示されます。	21.9.2 (19)
	［削除］	JSP の設定から，すべてのタグライブラリが削除されます。	―
＜タグライブラリの URI ＞	［削除］	JSP の設定から，選択したタグライブラリが削除されます。	―
JSP プロパティー・グループ	［追加］-［JSP プロパティー・グループ］	JSP プロパティグループが追加されます。JSP プロパティグループのプロパティページが表示されます。	21.9.2 (21)
	［削除］	JSP の設定から，すべての JSP プロパティグループが削除されます。	―
先頭の＜ URL パターン＞	［追加］-［アイコン］	アイコンが追加されます。アイコンのプロパティページが表示されます。すでにアイコンがある場合は選択できません。	21.9.2 (2)
	［追加］-［URL パターン］	URL パターンが追加されます。URL パターンのプロパティページが表示されます。	21.9.2 (23)
	［追加］-［ヘッダーにインクルードするファイル］	ヘッダにインクルードするファイルが追加されます。ヘッダにインクルードするファイルのプロパティページが表示されます。	21.9.2 (25)
	［追加］-［フッターにインクルードするファイル］	フッタにインクルードするファイルが追加されます。フッタにインクルードするファイルのプロパティページが表示されます。	21.9.2 (27)
	［削除］	JSP の設定から，選択した JSP プロパティグループが削除されます。	―
アイコン	［削除］	JSP プロパティグループからアイコンが削除されます。	―
URL パターン	［追加］-［URL パターン］	URL パターンが追加されます。URL パターンのプロパティページが表示されます。	21.9.2 (23)
＜ URL パターン＞	［削除］	JSP プロパティグループから，選択した URL パターンが削除されます。ただし，URL パターンが一つだけの場合は選択できません。	―

ノード	コンテキストメニュー	説明	プロパティページ
ヘッダーにインクルードするファイル	［追加］-［ヘッダーにインクルードするファイル］	ヘッダにインクルードするファイルが追加されます。ヘッダにインクルードするファイルのプロパティページが表示されます。	21.9.2 (25)
	［削除］	JSP プロパティグループから，ヘッダにインクルードするファイルがすべて削除されます。	－
＜ヘッダにインクルードするファイル＞	［削除］	JSP プロパティグループから，選択したヘッダにインクルードするファイルが削除されます。	－
フッターにインクルードするファイル	［追加］-［フッターにインクルードするファイル］	フッタにインクルードするファイルが追加されます。フッタにインクルードするファイルのプロパティページが表示されます。	21.9.2 (27)
	［削除］	JSP プロパティグループから，フッタにインクルードするファイルがすべて削除されます。	－
＜フッタにインクルードするファイル＞	［削除］	JSP プロパティグループから，選択したフッタにインクルードするファイルが削除されます。	－
タグ・ライブラリー	［追加］-［タグ・ライブラリー］	Servlet2.3 以前用のタグライブラリが追加されます。タグライブラリのプロパティページが表示されます。	21.9.2 (19)
	［削除］	WAR 属性から，すべてのタグライブラリ（Servlet2.3 以前用）が削除されます。	－
＜タグライブラリの URI＞	［削除］	タグライブラリの URI（Servlet2.3 以前用）から，選択したタグライブラリが削除されます。	－
ログインの設定	［削除］	WAR 属性からログインの設定が削除されます。	－
環境エントリー	［追加］-［環境エントリー］	環境エントリが追加されます。環境エントリのプロパティページが表示されます。	21.9.2 (30)
	［削除］	WAR 属性から，すべての環境エントリーが削除されます。	－
＜環境エントリ名＞	［追加］-［インジェクション・ターゲット］	＜環境エントリ名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)

ノード	コンテキストメニュー	説明	プロパティページ
	［削除］	WAR 属性から，選択した環境エントリが削除されます。	－
インジェクション・ターゲット	［追加］-［インジェクション・ターゲット］	＜環境エントリ名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	＜環境エントリ名＞から，すべてのインジェクションターゲットが削除されます。	－
＜インジェクションターゲット名＞	［削除］	＜環境エントリ名＞から，選択したインジェクションターゲットが削除されます。	－
リモート EJB への参照	［追加］-［リモート EJB への参照］	リモート EJB への参照が追加されます。 リモート EJB への参照のプロパティページが表示されます。	21.9.2 (34)
	［削除］	WAR 属性から，すべてのリモート EJB への参照が削除されます。	－
＜リモート EJB への参照＞	［追加］-［インジェクション・ターゲット］	＜リモート EJB への参照＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	WAR 属性から，選択したリモート EJB への参照が削除されます。	－
インジェクション・ターゲット	［追加］-［インジェクション・ターゲット］	＜リモート EJB への参照＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	＜リモート EJB への参照＞から，すべてのインジェクションターゲットが削除されます。	－
＜インジェクションターゲット名＞	［削除］	＜リモート EJB への参照＞から，選択したインジェクションターゲットが削除されます。	－
ローカル EJB への参照	［追加］-［ローカル EJB への参照］	ローカル EJB への参照が追加されます。 ローカル EJB への参照のプロパティページが表示されます。	21.9.2 (36)
	［削除］	WAR 属性から，すべてのローカル EJB への参照が削除されます。	－

ノード	コンテキストメニュー	説明	プロパティページ
＜ローカル EJB への参照名＞	［追加］-［インジェクション・ターゲット］	＜ローカル EJB への参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	WAR 属性から，選択したローカル EJB への参照が削除されます。	―
インジェクション・ターゲット	［追加］-［インジェクション・ターゲット］	＜ローカル EJB への参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	＜ローカル EJB への参照名＞から，すべてのインジェクションターゲットが削除されます。	―
＜インジェクションターゲット名＞	［削除］	＜ローカル EJB への参照名＞から，選択したインジェクションターゲットが削除されます。	―
リソース参照	［追加］-［リソース参照］	リソース参照が追加されます。リソース参照のプロパティページが表示されます。	21.9.2 (38)
	［削除］	WAR 属性から，すべてのリソース参照が削除されます。	―
＜リソース参照名＞	［追加］-［インジェクション・ターゲット］	＜リソース参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	WAR 属性から，選択したリソース参照が削除されます。	―
インジェクション・ターゲット	［追加］-［インジェクション・ターゲット］	＜リソース参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	＜リソース参照名＞から，すべてのインジェクションターゲットが削除されます。	―
＜インジェクションターゲット名＞	［削除］	＜リソース参照名＞から，選択したインジェクションターゲットが削除されます。	―
リソース環境変数	［追加］-［リソース環境変数］	リソース環境変数が追加されます。リソース環境変数のプロパティページが表示されます。	21.9.2 (40)

ノード	コンテキストメニュー	説明	プロパティページ
	［削除］	WAR 属性から，すべてのリソース環境変数が削除されます。	－
＜リソース環境変数名＞	［追加］-［インジェクション・ターゲット］	＜リソース環境変数名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［追加］-［リンク先キュー］	リンク先キューが追加されます。リンク先キューのプロパティページが表示されます。	21.9.2 (41)
	［追加］-［リンク先の管理対象オブジェクト］	リンク先の管理対象オブジェクトが追加されます。リンク先の管理対象オブジェクトのプロパティページが表示されます。	21.9.2 (42)
	［削除］	WAR 属性から，選択したリソース環境変数が削除されます。	－
インジェクション・ターゲット	［追加］-［インジェクション・ターゲット］	＜リソース環境変数名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	＜リソース環境変数名＞から，すべてのインジェクションターゲットが削除されます。	－
＜インジェクションターゲット名＞	［削除］	＜リソース環境変数名＞から，選択したインジェクションターゲットが削除されます。	－
リンク先キュー	［削除］	リソース環境変数からリンク先キューが削除されます。	－
リンク先の管理対象オブジェクト	［削除］	リソース環境変数から，選択したリンク先の管理対象オブジェクトが削除されます。	－
永続化コンテキスト参照	［追加］-［永続化コンテキスト参照］	永続化コンテキスト参照が追加されます。永続化コンテキスト参照のプロパティページが表示されます。	21.9.2 (44)
＜永続化コンテキスト参照名＞	［追加］-［永続化のプロパティー］	永続化のプロパティが追加されます。永続化のプロパティのプロパティページが表示されます。	21.9.2 (46)
	［追加］-［インジェクション・ターゲット］	＜永続化コンテキスト参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)

ノード	コンテキストメニュー	説明	プロパティページ
	［削除］	WAR 属性から，選択した永続化コンテキスト参照が削除されます。	－
永続化のプロパティー	［追加］-［永続化のプロパティー］	永続化のプロパティーが追加されます。永続化のプロパティのプロパティページが表示されます。	21.9.2 (46)
＜永続化のプロパティ名＞	［削除］	永続化コンテキスト参照から，選択した永続化のプロパティが削除されます。	－
インジェクション・ターゲット	［追加］-［インジェクション・ターゲット］	＜永続化コンテキスト参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	＜永続化コンテキスト参照名＞から，すべてのインジェクションターゲットが削除されます。	－
＜インジェクションターゲット名＞	［削除］	＜永続化コンテキスト参照名＞から，選択したインジェクションターゲットが削除されます。	－
永続化ユニット参照	［追加］-［永続化ユニット］	永続化ユニット参照が追加されます。永続化ユニット参照のプロパティページが表示されます。	21.9.2 (48)
＜永続化ユニット参照名＞	［追加］-［インジェクション・ターゲット］	＜永続化ユニット参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	WAR 属性から，選択した永続化ユニットが削除されます。	－
インジェクション・ターゲット	［追加］-［インジェクション・ターゲット］	＜永続化ユニット参照名＞にインジェクションターゲットが追加されます。インジェクションターゲットのプロパティページが表示されます。	21.9.2 (32)
	［削除］	＜永続化ユニット参照名＞から，すべてのインジェクションターゲットが削除されます。	－
＜インジェクションターゲット名＞	［削除］	＜永続化ユニット参照名＞から，選択したインジェクションターゲットが削除されます。	－
PostConstruct メソッド	［追加］-［PostConstruct メソッド］	PostConstruct メソッドが追加されます。PostConstruct メソッドのプロパティページが表示されます。	21.9.2 (50)

ノード	コンテキストメニュー	説明	プロパティページ
	［削除］	WAR 属性から，すべての PostConstruct メソッドが削除されます。	－
＜ PostConstruct メソッド名＞	［削除］	WAR 属性から，選択した PostConstruct メソッドが削除されます。	－
PreDestroy メソッド	［追加］-［PreDestroy メソッド］	PreDestroy メソッドが追加されます。PreDestroy メソッドのプロパティページが表示されます。	21.9.2 (52)
	［削除］	WAR 属性から，すべての PreDestroy メソッドが削除されます。	－
＜ PreDestroy メソッド名＞	［削除］	WAR 属性から，選択した PreDestroy メソッドが削除されます。	－
ロケール・エンコーディング・マッピング・リスト	［追加］-［ロケール・エンコーディング・マッピング・リスト］	ロケールエンコーディングマッピングリストとロケールエンコーディングマッピングを追加します。ロケールエンコーディングマッピングのプロパティページが表示されます。	21.9.2 (54)
	［削除］	WAR 属性から，すべてのロケールエンコーディングマッピングリストが削除されます。	－
ロケール・エンコーディング・マッピング・リスト	［追加］-［ロケール・エンコーディング・マッピング］	ロケールエンコーディングマッピングが追加されます。ロケールエンコーディングマッピングのプロパティページが表示されます。	21.9.2 (54)
	［削除］	WAR 属性から，選択したロケールエンコーディングマッピングリストが削除されます。	－
＜ロケール＞	［削除］	ロケール・エンコーディング・マッピング・リストから選択したロケールエンコーディングマッピングが削除されます。ただし，ロケールエンコーディングマッピングが一つしかない場合は選択できません。	－
スレッド数制御	［追加］-［URL グループのスレッド数制御］	URL グループのスレッド数制御が追加されます。URL グループのスレッド数制御のプロパティページが表示されます。	21.9.2 (57)
	［削除］	WAR 属性からスレッド数制御が削除されます。	－

ノード	コンテキストメニュー	説明	プロパティページ
URL グループのスレッド数制御	［追加］-［URL グループのスレッド数制御］	URL グループのスレッド数制御が追加されます。URL グループのスレッド数制御のプロパティページが表示されます。	21.9.2 （57）
	［削除］	URL グループのスレッド数制御から，すべての URL グループのスレッド数制御が削除されます。	－
＜URL グループのスレッド数制御定義名＞	［追加］-［URL グループのスレッド数制御マッピング］	URL グループのスレッド数制御マッピングが追加されます。URL グループのスレッド数制御マッピングのプロパティページが表示されます。	21.9.2 （59）
	［削除］	スレッド数制御から，選択した URL グループのスレッド数制御が削除されます。	－
URL グループのスレッド数制御マッピング	［追加］-［URL グループのスレッド数制御マッピング］	URL グループのスレッド数制御マッピングが追加されます。URL グループのスレッド数制御マッピングのプロパティページが表示されます。	21.9.2 （59）
	［削除］	URL グループのスレッド数制御から，すべての URL グループのスレッド数制御マッピングが削除されます。	－
＜URL パターン＞	［削除］	URL グループのスレッド数制御から，選択した URL グループのスレッド数制御御マッピングが削除されます。	－
HTTP リクエスト	［削除］	WAR 属性から HTTP リクエストが削除されます。	－
HTTP レスポンス	［削除］	WAR 属性から HTTP レスポンスが削除されます。	－
JSP	［削除］	WAR 属性から JSP が削除されます。	－

（凡例）　－：該当なし
注※　Servlet2.3 以前の場合は，WAR 属性直下の［タグ・ライブラリー］に追加してください。
Servlet2.4 以降の場合は，［JSP の設定］下の［タグ・ライブラリー］に追加してください。

21.9.2　WAR 属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に編集する WAR 属性ファイルのプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアの表

示が切り替わります。

WAR 属性ファイルには，次のプロパティページがあります。

- WAR のプロパティページ
- アイコンのプロパティページ
- パラメタの一覧ページ
- パラメタのプロパティページ
- フィルタマッピングの一覧ページ
- フィルタマッピング（URL パターン）のプロパティページ
- フィルタマッピング（サーブレット名）のプロパティページ
- リスナの一覧ページ
- リスナのプロパティページ
- セッションの設定のプロパティページ
- MIME タイプのマッピングの一覧ページ
- MIME タイプのマッピングのプロパティページ
- ウェルカムファイルリストのプロパティページ
- ウェルカムファイルのプロパティページ
- エラーページの一覧ページ
- エラーページ（エラーコード）のプロパティページ
- エラーページ（例外クラス）のプロパティページ
- タグライブラリの一覧ページ
- タグライブラリのプロパティページ
- JSP プロパティグループの一覧ページ
- JSP プロパティグループのプロパティページ
- URL パターンの一覧ページ
- URL パターンのプロパティページ
- ヘッダにインクルードするファイルの一覧ページ
- ヘッダにインクルードするファイルのプロパティページ
- フッタにインクルードするファイルの一覧ページ
- フッタにインクルードするファイルのプロパティページ
- ログインの設定のプロパティページ
- 環境エントリの一覧のページ
- 環境エントリのプロパティページ
- インジェクションターゲットの一覧のページ
- インジェクションターゲットのプロパティページ
- リモート EJB への参照の一覧ページ
- リモート EJB への参照のプロパティページ
- ローカル EJB への参照の一覧ページ
- ローカル EJB への参照のプロパティページ
- リソース参照の一覧のページ
- リソース参照のプロパティページ
- リソース環境変数の一覧ページ

- リソース環境変数のプロパティページ
- リンク先キューのプロパティページ
- リンク先の管理対象オブジェクトのプロパティページ
- 永続化コンテキスト参照の一覧ページ
- 永続化コンテキスト参照のプロパティページ
- 永続化プロパティの一覧のページ
- 永続化プロパティのプロパティページ
- 永続化ユニット参照の一覧のページ
- 永続化ユニット参照のプロパティページ
- PostConstruct メソッドの一覧のページ
- PostConstruct メソッドのプロパティページ
- PreDestroy メソッドの一覧のページ
- PreDestroy メソッドのプロパティページ
- ロケールエンコーディングマッピングの一覧ページ
- ロケールエンコーディングマッピングのプロパティページ
- スレッド数制御のプロパティページ
- URL グループのスレッド数制御の一覧ページ
- URL グループのスレッド数制御のプロパティページ
- URL グループのスレッド数制御マッピングの一覧ページ
- URL グループのスレッド数制御マッピングのプロパティページ
- 実行待ちリクエスト数の監視のプロパティページ
- HTTP リクエストのプロパティページ
- HTTP レスポンスのプロパティページ
- JSP のプロパティページ

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

（1） WAR のプロパティページ

WAR のプロパティページを次に示します。

図 21-104　WAR のプロパティページ

表 21-89　WAR のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	WAR ファイルについての説明を指定します。	<description> 説明 </description>
表示名	－	WAR ファイルの表示名が表示されます。	<display-name> 表示名 </display-name>
分散したサーブレットコンテナーへデプロイ可能	×	WAR ファイルが分散化されたサーブレットコンテナに配布できるかどうかを指定します。指定できる値を次に示します。 チェック済み：デプロイ可能 チェック解除：デプロイ不可	<distributable/> <distributable/> タグ自体削除
コンテキスト・ルート	○	コンテキストルートを，空文字，または URI で指定します。ルートは，空文，または「/」を指定します。	<war-runtime> <context-root> コンテキスト・ルート </context-root> </war-runtime>
開始時のエラー通知	×	J2EE アプリケーション開始時にスタートアップ時のロードが指定されているサーブレットの初期化処理中や，タグライブラリの解析中にエラーが発生した場合に，エラーを通知してアプリケーションの開始を中断するかどうかを指定します。指定できる値を次に示します。 • （空白） 指定なし • true 開始時のエラーを通知する • false 開始時のエラーを通知しない	<start-notify-error> </start-notify-error>
開始・停止順	×	J2EE アプリケーションの開始・停止順を「0」から「2147483647」の整数で指定します。 開始時は昇順，停止時は降順に処理が行われます。 指定がない場合，「10」が仮定されます。	<start-order> 開始・停止順 </start-order >

（凡例）　○：必須　×：任意　－：変更不可

(2) アイコンのプロパティページ

アイコンのプロパティページで次のアイコンを設定できます。

- WAR 属性のアイコン
- リスナのアイコン

- JSP プロパティグループのアイコン

アイコンのプロパティページについては，「21.4.2　アプリケーション属性設定ページ」の「(2)　アイコンのプロパティページ」を参照してください。

(3) パラメタの一覧ページ

パラメタの一覧には，設定されたコンテキストパラメタ (WAR 属性) または初期化パラメタ (フィルタ属性・サーブレット属性) がリストで表示されます。

パラメタの一覧ページを次に示します。

図 21-105　パラメタの一覧ページ

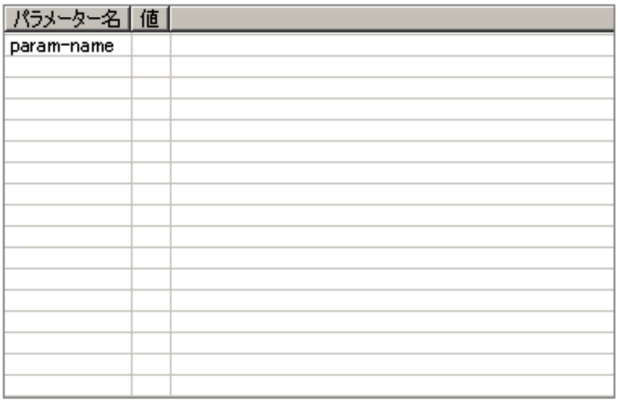


表 21-90　パラメタの一覧の表示項目

項目	説明	対応するタグ
パラメーター名	パラメタの名前が表示されます。	<context-param> <param-name> パラメタ名 </param-name> </context-param> ※
値	パラメタの値が表示されます。	<context-param> <param-value> 値 </param-value> </context-param> ※

注※　対応するタグは，初期化パラメタの場合 <context-param></context-param> の部分は，<init-param></init-param> になります。

(4) パラメタのプロパティページ

パラメタのプロパティページを次に示します。

パラメタのプロパティページで，コンテキストパラメタ (WAR 属性) または初期化パラ

メタ（フィルタ属性・サーブレット属性）を設定します。

図 21-106　パラメタのプロパティページ

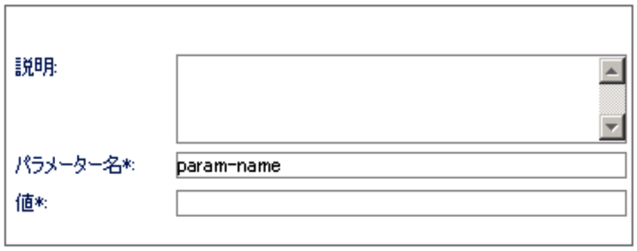


表 21-91　パラメタのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	パラメタついての説明を指定します。	<description> 説明 </description>
パラメーター名	○	パラメタの名前を指定します。※	<param-name> パラメタ名 </param-name>
値	○	パラメタの値を指定します。	<param-value> 値 </param-value>

（凡例）　○：必須　×：任意

注※　次の場合，同じパラメタの名前は指定できません。

- WAR 属性内に「コンテキストパラメタ」が複数ある場合のコンテキストパラメタ名
- フィルタ属性内に「初期化パラメタ」が複数ある場合の初期化パラメタ名
- サーブレット属性内に「初期化パラメタ」が複数ある場合の初期化パラメタ名

(5) フィルタマッピングの一覧ページ

フィルタマッピングの一覧には，設定されたフィルタマッピングがリストで表示されます。

フィルタマッピングの一覧ページを次に示します。

図 21-107 フィルタマッピングの一覧のページ

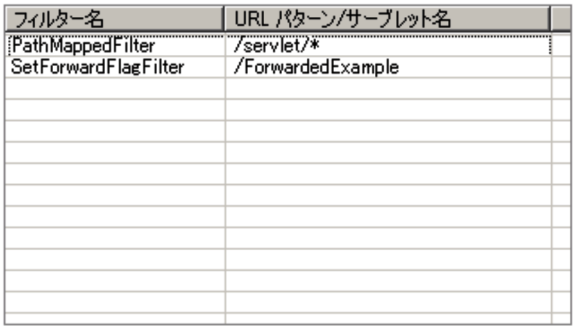


表 21-92 フィルタマッピングの一覧の表示項目

項目	説明	対応するタグ
フィルター名	フィルタの名前が表示されます。	<filter-mapping> <filter-name> フィルタ名 </filter-name> </filter-mapping>
URL パターン / サーブレット名	フィルタを対応づける URL パターンまたはサーブレット名が表示されます。	<filter-mapping> <url-pattern> URL パターン </url-pattern > </filter-mapping> または <filter-mapping> <servlet-name> サーブレット名 </servlet-name> </filter-mapping >

(6) フィルタマッピング（URL パターン）のプロパティページ

フィルタマッピング（URL パターン）のプロパティページを次に示します。

図 21-108 フィルタマッピング（URL パターン）のプロパティページ

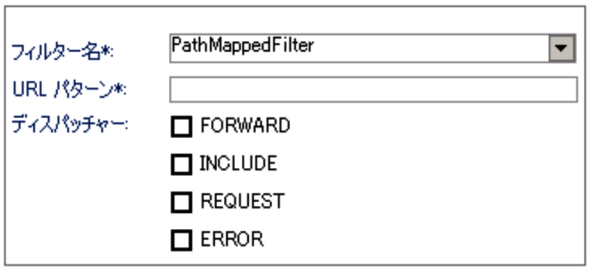


表 21-93　フィルタマッピング（URL パターン）のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
フィルター名	○	マッピングするフィルタの名称を選択します。 ドロップダウンリストには統合属性ファイルに定義済みのフィルタ名の一覧が表示されます。	<filter-name> フィルタ名 </filter-name >
URL パターン	○	指定したフィルタを適用する URL パターンを指定します。	<url-pattern> URL パターン </url-pattern >
ディスパッチャー	×	フィルタの適用ディスパッチャを選択します。 次の値から，複数選択できます。 • FORWARD RequestDispatcher.forward() が呼び出された場合だけにフィルタを適用する • INCLUDE RequestDispatcher.include() が呼び出された場合だけにフィルタを適用する • REQUEST 通常のクライアントからのアクセスのときだけにフィルタを適用する • ERROR 例外処理の場合だけにフィルタを適用する	<dispatcher> </dispatcher>

（凡例）　○：必須　×：任意

(7) フィルタマッピング（サーブレット名）のプロパティページ

フィルタマッピング（サーブレット名）のプロパティページを次に示します。

図 21-109　フィルタマッピング（サーブレット名）のプロパティページ

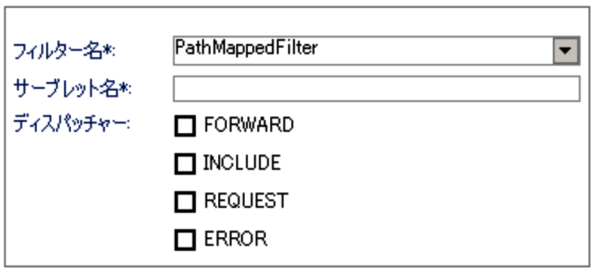


表 21-94　フィルタマッピング（サーブレット名）のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
フィルター名	○	マッピングするフィルタの名称を選択します。 ドロップダウンリストには統合属性ファイルに定義されているフィルタ名の一覧が表示されます。	<filter-name> フィルタ名 </filter-name >
サーブレット名	○	対応づけるサーブレットの名称を指定します。	<servlet-name> サーブレット名 </servlet-name>
ディスパッチャー	×	フィルタの適用ディスパッチャを次から選択します。複数選択できます。 • FORWARD RequestDispatcher.forward() が呼び出された場合だけにフィルタを適用する • INCLUDE RequestDispatcher.include() が呼び出された場合だけにフィルタを適用する • REQUEST 通常のクライアントがアクセスした場合だけにフィルタを適用する • ERROR 例外処理の場合だけにフィルタを適用する	<dispatcher> </dispatcher> <dispatcher> FORWARD </dispatcher> <dispatcher> INCLUDE </dispatcher> <dispatcher> REQUEST </dispatcher> <dispatcher> ERROR </dispatcher>

（凡例）　○：必須　×：任意

(8) リスナの一覧ページ

リスナの一覧には，設定されたリスナがリストで表示されます。

リスナの一覧ページを次に示します。

図 21-110　リスナの一覧ページ

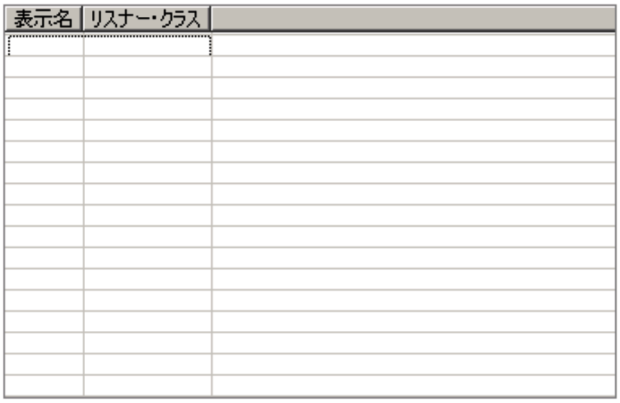


表 21-95　リスナの一覧の表示項目

項目	説明	対応するタグ
表示名	表示名が表示されます。	<listener> <display-name> 表示名 </display-name> </listener >
リスナー・クラス	リスナクラスが表示されます。	<listener> <listener-class> リスナクラス </listener-class> </listener >

(9) リスナのプロパティページ

リスナのプロパティページを次に示します。

図 21-111　リスナのプロパティページ

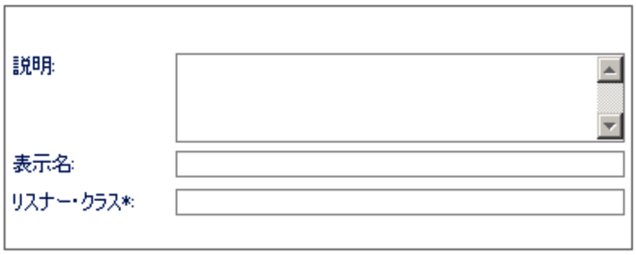


表 21-96　リスナのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	リスナついての説明を指定します。	<description> 説明 </description>
表示名	×	リスナの表示名を指定します。	<display-name> 表示名 </display-name>
リスナー・クラス	○	リスナの実装クラスを完全修飾名で指定します。 WAR 属性に複数のリスナがある場合，同じリスナクラスは指定できません。	<listener-class> リスナクラス </listener-class>

（凡例）　○：必須　×：任意

(10) セッションの設定のプロパティページ

セッションの設定のプロパティページを次に示します。

図 21-112　セッションの設定のプロパティページ

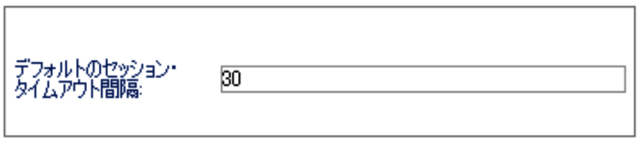


表 21-97　セッションの設定のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
デフォルトのセッション・タイムアウト間隔	×	セッションタイムアウトまでの時間を，「-1」，「0」，または「1（分）」から「35791394（分）」の整数で指定します。 セッションタイムアウトまでの時間に，「-1」，「0」を指定すると，タイムアウトしません。 指定がない場合，30（分）が仮定されます。	<session-config> <session-timeout> セッションタイムアウト間隔 </session-timeout> </session-config>

（凡例）×：任意

(11) MIME タイプのマッピングの一覧ページ

MIME タイプのマッピングの一覧には，設定された MIME タイプのマッピングがリスト

で表示されます。

MIME タイプのマッピングの一覧ページを次に示します。

図 21-113　MIME タイプのマッピングの一覧ページ

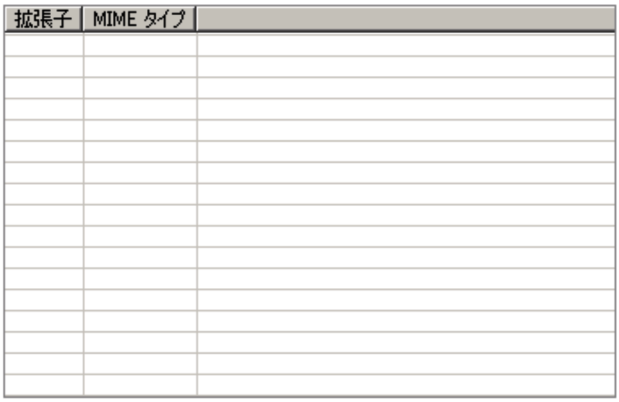


表 21-98　MIME タイプのマッピングの一覧の表示項目

項目	説明	対応するタグ
拡張子	拡張子が表示されます。	<mime-mapping> <extension> 拡張子 </extension> </mime -mapping>
MIME タイプ	拡張子に対応づける MIME タイプが表示されます。	<mime-mapping> <mime-type> MIME タイプ </mime-type> </mime-mapping>

(12) MIME タイプのマッピングのプロパティページ

MIME タイプのマッピングのプロパティページを次に示します。

図 21-114　MIME タイプのマッピングのプロパティページ

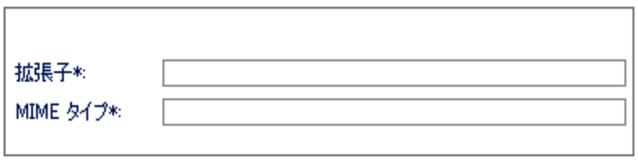


表 21-99 MIME タイプのマッピングのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
拡張子	○	マッピングするファイル拡張子を指定します。 WAR 属性に複数の MIME タイプのマッピングがある場合，同じ拡張子は指定できません。	<extension> 拡張子 </extension>
MIME タイプ	○	拡張子に対応するコンテンツのタイプを指定します。	<mime-type> MIME タイプ </mime-type>

（凡例） ○：必須

(13) ウェルカムファイルリストのプロパティページ

ウェルカムファイルリストのプロパティページを次に示します。

図 21-115 ウェルカムファイルリストのプロパティページ

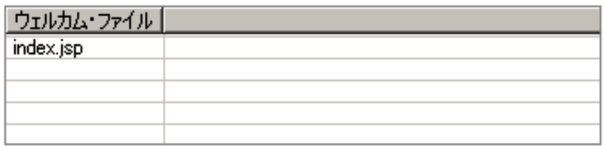


表 21-100 ウェルカムファイルリストのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ウェルカム・ファイル	○	ウェルカムファイルを指定します。 インライン入力ができます。※ WAR 属性のすべてのウェルカムファイルリストで，同じウェルカムファイルは指定できません。	<welcome-file-list> <welcome-file> ウェルカムファイル </welcome-file> <welcome-file-list>

（凡例） ○：必須

注※ ［ウェルカム・ファイル］は，インライン入力のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-101 インライン入力のコンテキストメニュー

コンテキストメニュー	説明
［追加］	リストの末尾および属性ファイルに項目を追加します。常に活性状態です。
［削除］	リストの選択行の項目をリストおよび属性ファイルから削除します。リストの表示項目が一つの場合は不活性状態です。

(14) ウェルカムファイルのプロパティページ

ウェルカムファイルのプロパティページを次に示します。

図 21-116　ウェルカムファイルのプロパティページ

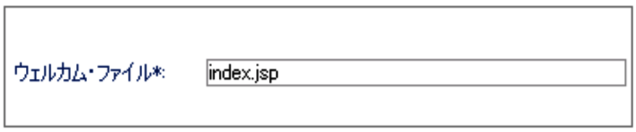


表 21-102　ウェルカムファイルのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ウェルカム・ファイル	○	ウェルカムファイルを指定します。WAR 属性のすべてのウェルカムファイルリストで，同じウェルカムファイルは指定できません。	<welcome-file> ウェルカムファイル </welcome-file>

（凡例）　○：必須

(15) エラーページの一覧ページ

エラーページの一覧には，設定されたエラーページがリストで表示されます。

エラーページの一覧ページを次に示します。

図 21-117　エラーページの一覧ページ

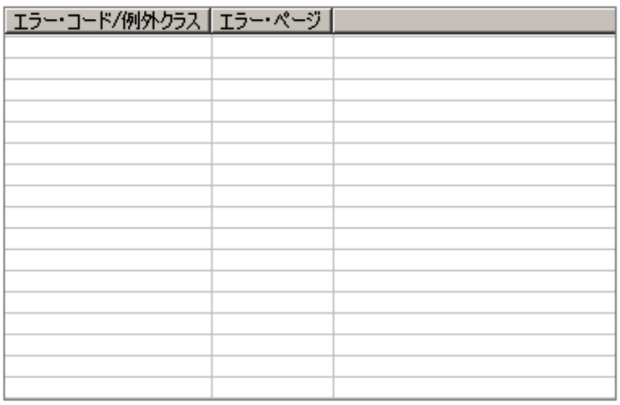


表 21-103 エラーページの一覧の表示項目

項目	説明	対応するタグ
エラー・コード／例外クラス	エラーページを設定する HTTP エラーコードまたは例外クラスが表示されます。	<error-page> <error-code> エラーコード </error-code> </error-page> または <error-page> <exception-type> 例外クラス </exception-type> </error-page>
エラー・ページ	HTTP エラーコードまたは例外クラスに対応づけられたエラーページが表示されます。	<error-page> <location> エラーページ </location> </error-page>

(16) エラーページ（エラーコード）のプロパティページ

エラーページ（エラーコード）のプロパティページを次に示します。

図 21-118 エラーページ（エラーコード）のプロパティページ

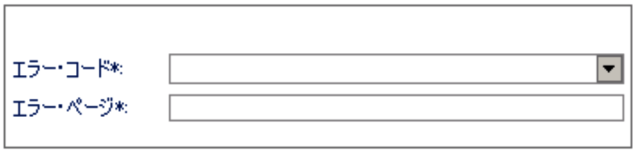


表 21-104　エラーページ（エラーコード）のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
エラー・コード	○	エラーページを設定する HTTP エラーコードを指定します。 「-2147483648」から「2147483647」の整数で指定するか，ドロップダウンリストから選択してください。ドロップダウンリストで選択できる値を次に示します。 • 400 • 401 • 403 • 404 • 500 • 501 • 502 WAR 属性に複数のエラーページがある場合，同じエラーコードは指定できません。	<error-code> エラーコード </error-code>
エラー・ページ	○	エラーコードに対応づけるページの WAR を指定します。 「/」で始まる値を指定します。	<location> エラーページ </location>

（凡例）　○：必須

（17）エラーページ（例外クラス）のプロパティページ

エラーページ（例外クラス）のプロパティページを次に示します。

図 21-119　エラーページ（例外クラス）のプロパティページ

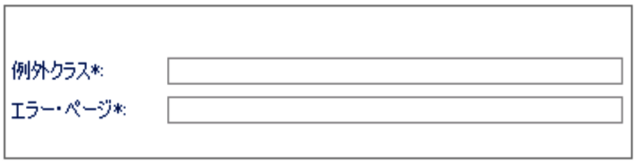


表 21-105　エラーページ（例外クラス）のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
例外クラス	○	エラーページを設定する例外クラスを完全修飾名で指定します。 WAR 属性に複数のエラーページがある場合，同じ例外クラスは指定できません。	<exception-type> 例外クラス </exception-type>

項目	必須／任意	説明	対応するタグ
エラー・ページ	○	指定した例外が発生した場合に表示するページを指定します。「/」で始まる値を指定します。	<location> エラーページ </location>

（凡例） ○：必須

(18) タグライブラリの一覧ページ

タグライブラリの一覧には，設定されたタグライブラリがリストで表示されます。

タグライブラリの一覧ページを次に示します。

図 21-120 タグライブラリの一覧ページ

表 21-106 タグライブラリの一覧の表示項目

項目	説明	対応するタグ
タグ・ライブラリーの URI	タグライブラリの URI が表示されます。	<taglib> <taglib-uri> タグライブラリの URI </taglib-uri> </taglib>
タグ・ライブラリーのファイル・パス	タグライブラリのファイルパスが表示されます。	<taglib> <taglib-location> タグライブラリのファイルパス </taglib-location> </taglib>

(19) タグライブラリのプロパティページ

タグライブラリのプロパティページを次に示します。

図 21-121　タグライブラリのプロパティページ

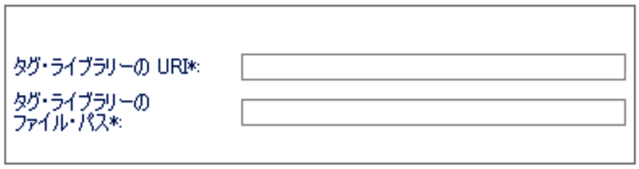


表 21-107　タグライブラリのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
タグ・ライブラリーの URI	○	タグライブラリの web.xml に対する相対的な URI を指定します。 WAR 属性に，タグライブラリ（[JSP の設定] 下，または [WAR 属性] 直下）が複数ある場合，同じタグライブラリの URI は指定できません。	<taglib-uri> タグライブラリーの URI </taglib-uri>
タグ・ライブラリーのファイル・パス	○	タグライブラリのファイルの場所を，WAR のルートからの相対位置の URL で指定します。	<taglib-location> タグライブラリーのファイルパス </taglib-location>

（凡例）　○：必須

(20) JSP プロパティグループの一覧ページ

JSP プロパティグループの一覧には，設定された JSP プロパティグループがリストで表示されます。

JSP プロパティグループの一覧ページを次に示します。

図 21-122　JSP プロパティグループの一覧ページ

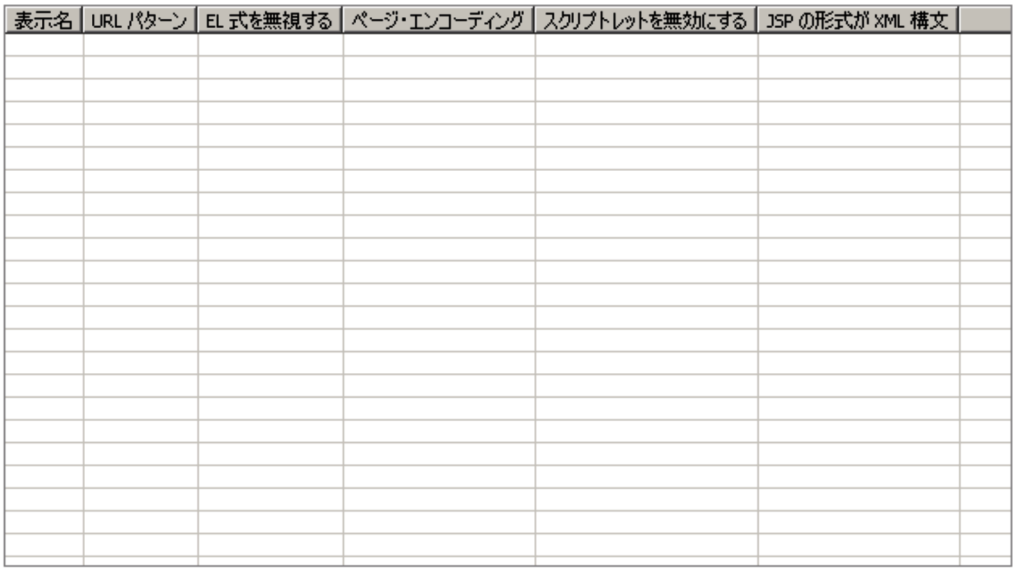


表 21-108　JSP プロパティグループの一覧の表示項目

項目	説明	対応するタグ
表示名	JSP プロパティグループの表示名が表示されます。	<jsp-property-group> <display-name> 表示名 </display-name> </jsp-property-group >
URL パターン	JSP プロパティグループを適用する URL パターンが表示されます。	<jsp-property-group> <url-pattern> URL パターン </url-pattern> </jsp-property-group >
EL 式を無視する	JSP 内の EL 式を無視するかどうかが，次の形式で表示されます。 • （空白） ［EL 式を無視する］が指定されていない • true EL 式を使用しない • false EL 式を使用する	<jsp-property-group> <el-ignored> EL 式を無視する </el-ignored> </jsp-property-group >
ページ・エンコーディング	JSP 内のページエンコーディングが表示されます。	<jsp-property-group> <page-encoding> ページエンコーディング </page-encoding> </jsp-property-group >

項目	説明	対応するタグ
スクリプトレットを無効にする	JSP 内のスクリプトレットを無効にするかどうかが，次の形式で表示されます。 • （空白） false が仮定される • true スクリプトを無効にする • false スクリプトを有効にする	<jsp-property-group> <scripting-invalid> スクリプトレットを無効にする </scripting-invalid> </jsp-property-group>
JSP の形式が XML 構文	JSP の形式が XML 構文かどうか，次の形式で表示されます。 • （空白） 「JSP の形式が XML 構文」の指定がされていない • true JSP ドキュメントである • false 従来の JSP である	<jsp-property-group> <is-xml> JSP の形式が XML 構文 </is-xml> </jsp-property-group>

(21) JSP プロパティグループのプロパティページ

JSP プロパティグループのプロパティページを次に示します。

図 21-123　JSP プロパティグループのプロパティページ

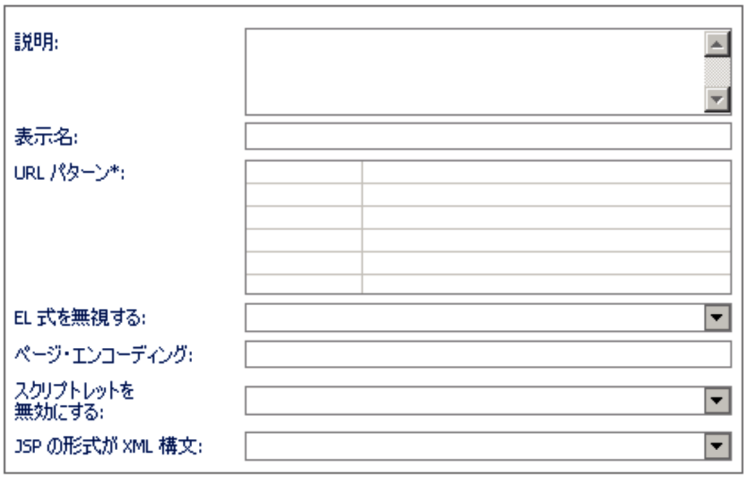


表 21-109　JSP プロパティグループのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	JSP プロパティグループの説明を指定します。	<description> 説明 </description>

項目	必須／任意	説明	対応するタグ
表示名	×	JSP プロパティグループの表示名を指定します。	<display-name> 表示名 </display-name>
URL パターン	○	JSP プロパティグループを適用する URL パターンを指定します。インライン入力ができます。※	<url-pattern> URL パターン </url-pattern>
EL 式を無視する	×	EL 式を無視するかどうかを指定します。指定できる値を次に示します。 • （空白） false が仮定される。 • true EL 式を使用しない。 • false EL 式を使用する。	<el-ignored> </el-ignored>
ページ・エンコーディング	×	ページエンコーディングを指定します。	<page-encoding> ページエンコーディング </page-encoding>
スクリプトレットを無効にする	×	JSP 内のスクリプトレットを無効にするかどうかを指定します。指定できる値を次に示します。 • true：スクリプトレットを無効にする • false，（空白）：スクリプトレットを有効にする	<scripting-invalid> </scripting-invalid>
JSP の形式が XML 構文	×	JSP が XML 構文の JSP（JSP ドキュメント形式）であるかどうかを指定します。指定できる値を次に示します。 •（空白）：JSP の形式の指定がない • true：JSP ドキュメントである • false：従来の JSP である	<is-xml> </is-xml> <is-xml> true </is-xml> <is-xml> false </is-xml>

（凡例） ○：必須　×：任意

注※ ［URL パターン］は，インライン入力のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

インライン入力のコンテキストメニューについては，「(13) ウェルカムファイルリストのプロパティページ」を参照してください。

(22) URL パターンの一覧ページ

URL パターンの一覧には，設定された URL パターンがリストで表示されます。

URL パターンの一覧ページを次に示します。

図 21-124　URL パターンの一覧ページ

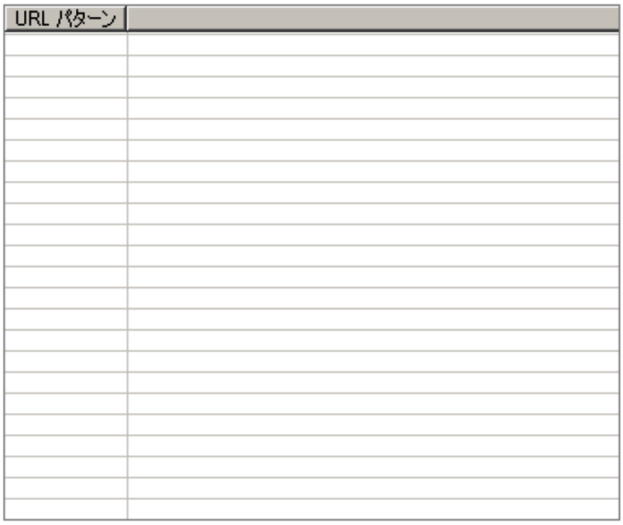


表 21-110　URL パターンの一覧の表示項目

項目	説明	対応するタグ
URL パターン	JSP プロパティグループを設定する URL パターンが表示されます。	<jsp-property-group> <url-pattern> URL パターン </url-pattern> </jsp-property-group>

(23) URL パターンのプロパティページ

URL パターンのプロパティページを次に示します。

図 21-125　URL パターンのプロパティページ

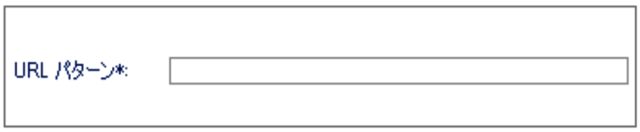


表 21-111　URL パターンのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
URL パターン	○	設定したプロパティグループに対応づける URL パターンを指定します。	<url-pattern> URL パターン </url-pattern>

（凡例） ○：必須

(24) ヘッダにインクルードするファイルの一覧ページ

ヘッダにインクルードするファイルの一覧には，設定されたヘッダにインクルードするファイルがリストで表示されます。

ヘッダにインクルードするファイルの一覧ページを次に示します。

図 21-126 ヘッダにインクルードするファイルの一覧ページ

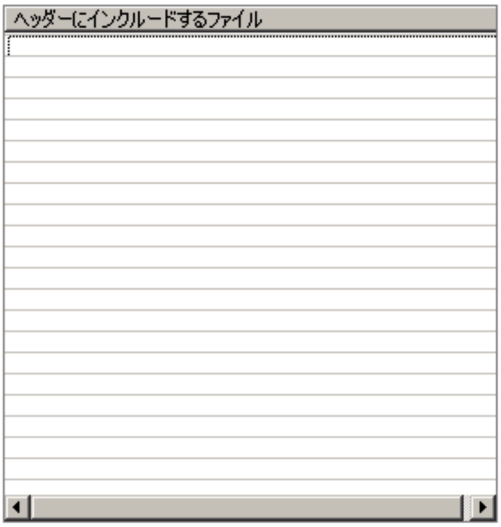


表 21-112 ヘッダにインクルードするファイルの一覧の表示項目

項目	説明	対応するタグ
ヘッダーにインクルードするファイル	ヘッダにインクルードするファイルが表示されます。	<include-prelude> ヘッダにインクルードするファイル </include-prelude>

(25) ヘッダにインクルードするファイルのプロパティページ

ヘッダにインクルードするファイルのプロパティページを次に示します。

図 21-127 ヘッダにインクルードするファイルのプロパティページ

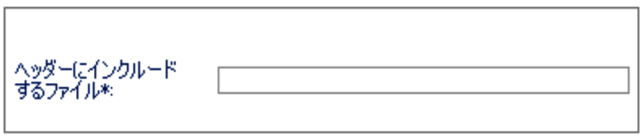


表 21-113　ヘッダにインクルードするファイルのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ヘッダーにインクルードするファイル	○	ページのヘッダにインクルードするファイルを相対パスで指定します。	<include-prelude> ヘッダにインクルードするファイル </include-prelude>

（凡例）　○：必須

(26) フッタにインクルードするファイルの一覧ページ

フッタにインクルードするファイルの一覧には，設定されたフッタにインクルードするファイルがリストで表示されます。

フッタにインクルードするファイルの一覧ページを次に示します。

図 21-128　フッタにインクルードするファイルの一覧ページ

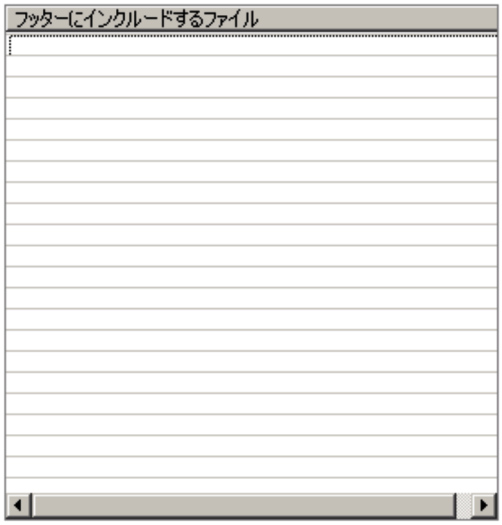


表 21-114　フッタにインクルードするファイルの一覧の表示項目

項目	説明	対応するタグ
フッターにインクルードするファイル	フッタにインクルードするファイルが表示されます。	<include-coda> フッタにインクルードするファイル </include-coda>

(27) フッタにインクルードするファイルのプロパティページ

フッターにインクルードするファイルのプロパティページを次に示します。

図 21-129　フッターにインクルードするファイルのプロパティページ

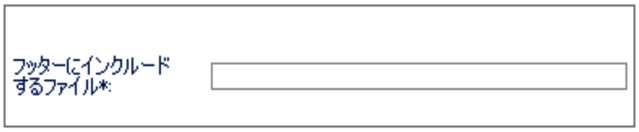


表 21-115　フッターにインクルードするファイルのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
フッターにインクルードするファイル	○	ページのフッタにインクルードするファイルを相対パスで指定します。	<include-coda> フッタにインクルードするファイル </include-coda>

（凡例）　○：必須

(28) ログインの設定のプロパティページ

ログインの設定のプロパティページを次に示します。

図 21-130　ログインの設定のプロパティページ

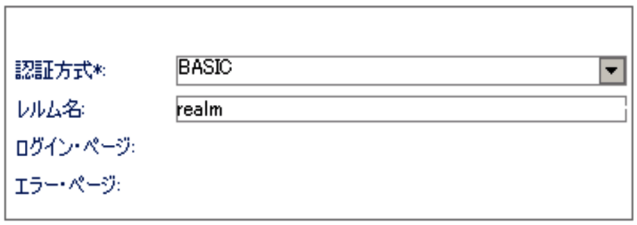


表 21-116　ログインの設定のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
認証方式	○	ログインの認証方式を指定します。 指定できる値を次に示します。 • BASIC HTTP Basic 認証を使用する • FORM フォーム認証を使用する • CLIENT － CERTSSL クライアント認証を要求する	<login-config> <auth-method> </auth-method> </login-config>

項目	必須／任意	説明	対応するタグ
レルム名	×	BASIC 認証で使用するレルム名を指定します。 ［認証方法］で「BASIC」を指定した場合は必ず指定してください。 ただし，認証方法で［BASIC］以外を指定した場合は指定できません。	<login-config> <realm-name> レルム名 </realm-name> </login-config>
ログイン・ページ	×	FORM 認証で使用するログインページの，WAR のルートからの URL を指定します。「/」で始まる値を指定します。 ［認証方法］で「FORM」を指定した場合は，必ず指定してください。 ただし，［認証方法］で「FORM」以外を指定した場合は指定できません。	<login-config> <form-login-config> <form-login-page> ログインページ </form-login-page> </form-login-config> </login-config>
エラー・ページ	×	FORM 認証でログインに失敗したときのエラーページの，WAR のルートからの URL を指定します。「/」で始まる値を指定します。 ［認証方法］で「FORM」を選択した場合は，必ず指定してください。 ただし，［認証方法］で「FORM」以外を指定した場合は指定できません。	<login-config> <form-login-config> <form-error-page> エラーページ </form-error-page> </form-login-config> </login-config>

（凡例）　○：必須　×：任意

(29) 環境エントリの一覧のページ

環境エントリの一覧には，設定された環境エントリがリストで表示されます。なお，表示されるリストの内容は Session Bean の場合と同じです。

環境エントリの一覧のページについては，「21.6.2　Session Bean 属性設定ページ」の「(3)　環境エントリの一覧のページ」を参照してください。

(30) 環境エントリのプロパティページ

環境エントリのプロパティページを次に示します。

図 21-131　環境エントリのプロパティページ

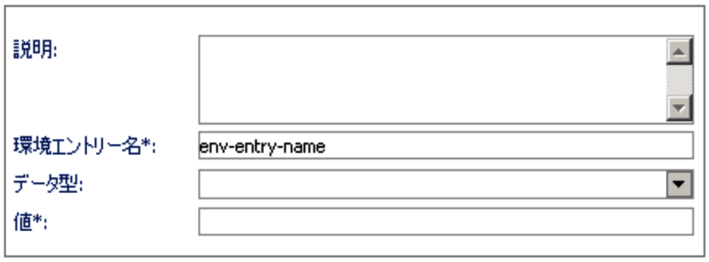


表 21-117 環境エントリのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
環境エントリー名	○	環境エントリ名を指定します。※1	<env-entry-name> 環境エントリ名 </env-entry-name>
データ型	×	環境エントリのデータ型を選択します。 選択できる値を次に示します。 • （空白） • java.lang.Boolean • java.lang.String • java.lang.Integer • java.lang.Double • java.lang.Byte • java.lang.Short • java.lang.Long • java.lang.Float • java.lang.Character	<env-entry-type> データ型 </env-entry-type>
値	×※2	環境エントリの値を指定します。 データ型に指定した型に適合した値を指定します。	<env-entry-value> 値 </env-entry-value>

（凡例） ○：必須 ×：任意

注※1 同一属性内に複数の環境エントリがある場合，同一の環境エントリ名（<env-entry-name>）は指定できません。

注※2 「java.lang.String」以外のデータ型を指定している場合は，必須です。

(31) インジェクションターゲットの一覧のページ

インジェクションターゲットの一覧には設定されたインジェクションターゲットがリストで表示されます。

インジェクションターゲットの一覧のページを次に示します。

図 21-132　インジェクションターゲットの一覧のページ

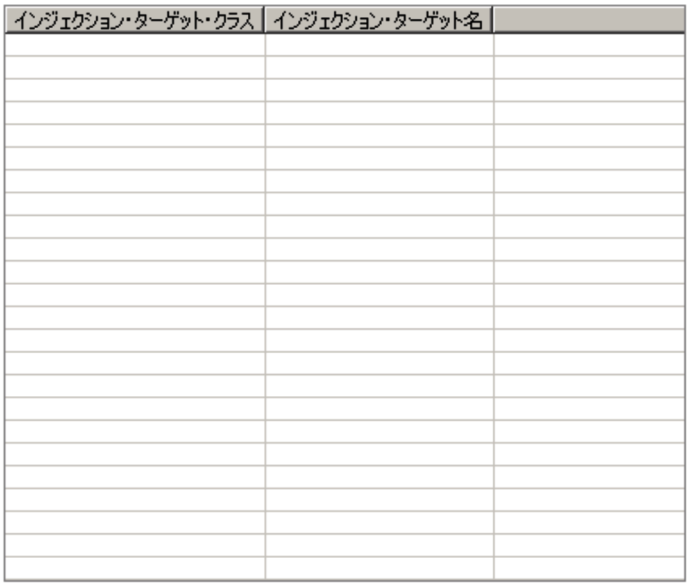


表 21-118　インジェクションターゲットの一覧の表示項目

項目	説明	対応するタグ
インジェクション・ターゲット・クラス	インジェクションターゲットクラスが表示されます。	<injection-target-class>インジェクションターゲットクラス</injection-target-class>
インジェクション・ターゲット名	インジェクションターゲット名が表示されます。	<injection-target-name>インジェクションターゲット名</injection-target-name>

(32) インジェクションターゲットのプロパティページ

インジェクションターゲットのプロパティページを次に示します。

図 21-133　インジェクションターゲットのプロパティページ

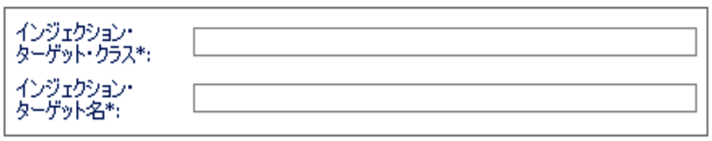


表 21-119　インジェクションターゲットのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
インジェクション・ターゲット・クラス	○	インジェクションターゲットクラスを完全修飾クラス名で指定します。	<injection-target-class> インジェクションターゲットクラス </injection-target-class>
インジェクション・ターゲット名	○	インジェクションターゲット名を指定します。	<injection-target-name> インジェクションターゲット名 </injection-target-name>

（凡例）　○：必須

(33) リモート EJB への参照の一覧ページ

リモート EJB への参照の一覧には，設定されたリモート EJB への参照がリストで表示されます。なお，表示されるリストの内容は Session Bean の場合と同じです。

リモート EJB への参照の一覧ページについては，「21.6.2　Session Bean 属性設定ページ」の「(5)　リモート EJB への参照の一覧ページ」を参照してください。

(34) リモート EJB への参照のプロパティページ

リモート EJB への参照のプロパティページを次に示します。

図 21-134　リモート EJB への参照のプロパティページ

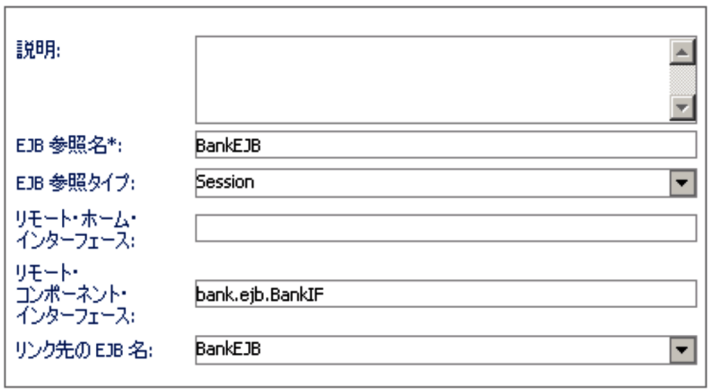


表 21-120　リモート EJB への参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>

項目	必須／任意	説明	対応するタグ
EJB 参照名	○	EJB 参照名を指定します。※	<ejb-ref-name> EJB 参照名 </ejb-ref-name>
EJB 参照タイプ	×	EJB 参照タイプを指定します。 指定できる値を次に示します。 • （空白） • Session • Entity	<ejb-ref-type> EJB 参照タイプ </ejb-ref-type>
リモート・ホーム・インターフェース	×	リモートホームインタフェースを完全修飾クラス名で指定します。	<home> リモートホームインタフェース </home>
リモート・コンポーネント・インターフェース	×	リモートコンポーネントインタフェースを完全修飾クラス名で指定します。	<remote> リモートコンポーネントインタフェース </remote>
リンク先の EJB 名	×	参照する EJB の EJB 名を入力するか，またはリスト項目から選択します。 • EJB がほかの ejb-jar ファイルにある場合は，次の形式で入力します。「ファイル名 #EJB 名」 • ネーミングの切り替え機能でリンク先を設定する場合，次の形式で入力します。 corbaname::< 名前空間のホスト名 >:< 名前空間のポート番号 >#<EJB ホームオブジェクトリファレンスの JNDI 名 • アプリケーション統合属性ファイルにある Enterprise Bean 表示名をリスト項目から選択します。	<ejb-link> リンク先の EJB 名 </ejb-link>

（凡例）　○：必須　×：任意

注※　同一属性内にリモート EJB への参照とローカル EJB への参照が両方ある場合や，リモート EJB への参照が複数ある場合，同じ EJB 参照名（<ejb-ref-name>）は指定できません。

(35) ローカル EJB への参照の一覧ページ

ローカル EJB への参照の一覧ページでは，設定されたローカル EJB への参照がリストで表示されます。なお，表示されるリストの内容は Session Bean の場合と同じです。

ローカル EJB への参照の一覧ページについては，「21.6.2　Session Bean 属性設定ページ」の「(7)　ローカル EJB への参照の一覧ページ」を参照してください。

(36) ローカル EJB への参照のプロパティページ

ローカル EJB への参照のプロパティページを次に示します。

図 21-135　ローカル EJB への参照のプロパティページ

表 21-121　ローカル EJB への参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
EJB 参照名	○	EJB 参照名を指定します。※	<ejb-ref-name> EJB 参照名 </ejb-ref-name>
EJB 参照タイプ	×	EJB 参照タイプを指定します。 指定できる値を次に示します。 • （空白） • Session • Entity	<ejb-ref-type> EJB 参照タイプ </ejb-ref-type>
ローカル・ホーム・インターフェース	×	ローカルホームインタフェースを完全修飾クラス名で指定します。	<local-home> ローカルホームインタフェース </local-home>
ローカル・コンポーネント・インターフェース	×	ローカルコンポーネントインタフェースを完全修飾クラス名で指定します。	<local> ローカルコンポーネントインタフェース </local>

項目	必須／任意	説明	対応するタグ
リンク先の EJB 名	×	参照する EJB の EJB 名を入力するか，またはリスト項目を選択します。 • EJB がほかの ejb-jar ファイルにある場合は，次の形式で入力します。 「ファイル名 #EJB 名」 • アプリケーション統合属性ファイルにある Enterprise Bean 表示名をリスト項目から選択します。	<ejb-link> リンク先の EJB 名 </ejb-link>

（凡例）　○：必須　×：任意

注※　同一属性内にリモート EJB への参照とローカル EJB への参照が両方ある場合や，ローカル EJB への参照が複数ある場合，同じ EJB 参照名（<ejb-ref-name>）は指定できません。

(37) リソース参照の一覧ページ

リソース参照の一覧では，設定されたリソース参照がリストで表示されます。なお，表示されるリストの内容は Session Bean の場合と同じです。

リソース参照の一覧ページについては，「21.6.2　Session Bean 属性設定ページ」の「(9)　リソース参照の一覧ページ」を参照してください。

(38) リソース参照のプロパティページ

リソース参照のプロパティページを次に示します。

図 21-136　リソース参照のプロパティページ

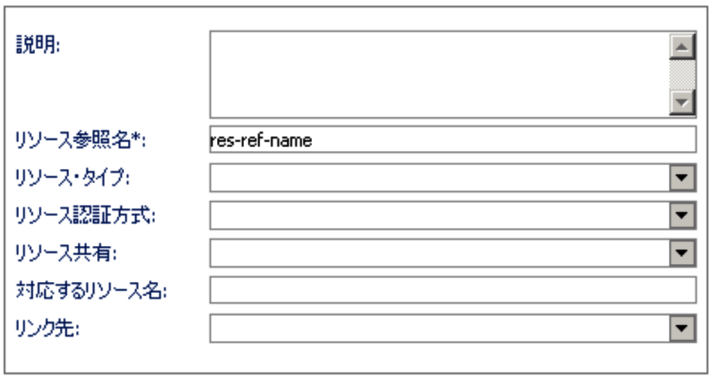


表 21-122 リソース参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
リソース参照名	○	リソース参照名を指定します。※1	<res-ref-name> リソース参照名 </res-ref-name>
リソース・タイプ	×	リソース種別を指定します。 完全修飾クラス名で入力するか，または次のリスト項目を選択します。 •（空白） • javax.jms.ConnectionFactory • javax.jms.QueueConnectionFactory • javax.jms.TopicConnectionFactory • javax.mail.Session • javax.resource.cci. ConnectionFactory • javax.sql.DataSource • org.omg.CORBA_2_3.ORB	<res-type> リソースタイプ </res-type>
リソース認証方式	×	リソース認証方式を指定します。 選択できる値を次に示します。 •（空白） • Application • Container	<res-auth> リソース認証方式 </res-auth>
リソース共有	×	リソース共有を指定します。 選択できる値を次に示します。 •（空白） • Shareable • Unshareable	<res-sharing-scope> リソース共有 </res-sharing-scope>
対応するリソース名	×	対応するリソースアダプタ名，またはメール表示名が表示されます。	<mapped-name> 対応するリソース名 </mapped-name>※2
リンク先	×	リンク先を指定します。 リンク先として，対応する J2EE リソースアダプタの表示名を入力するか，またはリスト項目を選択します。 J2EE サーバにデプロイされている J2EE リソースアダプタの表示名がリスト項目※4として表示されます。［サーバー・エクスプローラー］ビューでも J2EE リソースアダプタの表示名一覧を参照できます。	<linked-to> リンク先※3 </linked-to>

（凡例） ○：必須 ×：任意

注※1 同一属性内に複数のリソース参照名がある場合，同一のリソース参照名（<res-ref-name>）は指定できません。

注※2 <linked-to> タグに値が設定されている場合，<linked-to> タグの指定が有効になり，

<mapped-name> タグの値は無効になります。

注※ 3　<linked-to> タグのリンク先の指定方法は，リソースアダプタのバージョンによって，次のように異なります。

- Connector 1.0 の仕様に準拠する場合：＜リソースアダプタの表示名＞
- Connector 1.5 の仕様に準拠する場合：＜リソースアダプタの表示名＞！＜コネクション定義識別子＞

注※ 4　リスト項目には，デプロイ済みのリソースアダプタを表示されます。なお，Connector1.5 の仕様に準拠するリソースアダプタの場合は，［リソース・タイプ］の値とコネクション定義識別子が一致する Outbound リソースアダプタが表示されます。

(39) リソース環境変数の一覧ページ

リソース環境変数の一覧には，設定されたリソース環境変数がリストで表示されます。なお，表示されるリストの内容は Session Bean の場合と同じです。

リソース環境変数の一覧ページについては，「21.6.2　Session Bean 属性設定ページ」の「(11)　リソース環境変数の一覧ページ」を参照してください。

(40) リソース環境変数のプロパティページ

リソース環境変数のプロパティページを次に示します。

図 21-137　リソース環境変数のプロパティページ

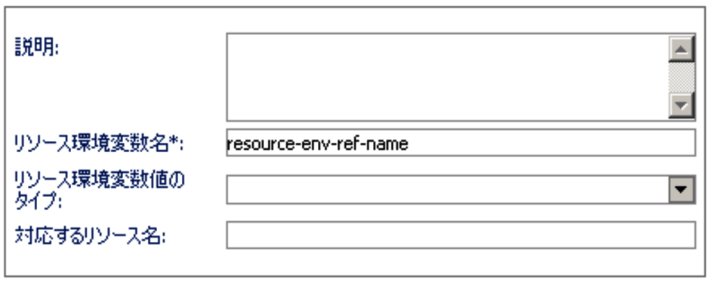


表 21-123　リソース環境変数のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
リソース環境変数名	○	リソース環境変数名を指定します。※ 1	<resource-env-ref-name> リソース環境変数名 </resource-env-ref-name>

項目	必須／任意	説明	対応するタグ
リソース環境変数値のタイプ	×	リソース環境変数値のタイプを指定します。 ＜JavaBeans リソースのクラス名＞を完全修飾クラス名で入力するか，または次のリスト項目を選択します。 • （空白） • javax.jms.Queue • javax.jms.Topic • javax.transaction.UserTransaction ※2 • javax.ejb.TimerService ※2 • javax.ejb.EJBContext ※2	<resource-env-ref-type> リソース環境変数値のタイプ </resource-env-ref-type>
対応するリソース名	×	リソース環境変数値のタイプが「javax.jms.Queue」または「JavaBeans リソースのクラス名」の場合，対応するリソース名を指定します。 •「javax.jms.Queue」の場合 対応するキューを「< リソースアダプタの表示名 >#<Queue 名称 >」の形式で指定します。 •「JavaBeans リソースのクラス名」の場合 対応する JavaBeans リソースの表示名を指定します。	<mapped-name> 対応するリソース名 </mapped-name>

（凡例）　○：必須　×：任意

注※ 1　同一属性内に複数のリソース環境変数がある場合，同じリソース環境変数名（<resource-env-ref-name>）は指定できません。

注※ 2　アノテーションで設定された値です。リスト項目として表示されますが，アノテーションで設定された値は変更できないため，J2EE アプリケーションサーバへの保管時にサーバ側で変更前の設定値に戻されます。

(41) リンク先キューのプロパティページ

リンク先キューの設定は Session Bean のプロパティの設定と同じです。

リンク先キューのプロパティページについては，「21.6.2　Session Bean 属性設定ページ」の「(13)　リンク先キューのプロパティページ」を参照してください。

(42) リンク先の管理対象オブジェクトのプロパティページ

リンク先の管理対象オブジェクトの設定は Session Bean のプロパティの設定と同じです。

リンク先キューのプロパティページについては，「21.6.2　Session Bean 属性設定ペー

ジ」の「(14)　リンク先の管理対象オブジェクトのプロパティページ」を参照してください。を参照してください。

(43) 永続化コンテキスト参照の一覧ページ

永続化コンテキスト参照の一覧には，設定された永続化コンテキスト参照がリストで表示されます。

永続化コンテキスト参照の一覧ページを次に示します。

図 21-138　永続化コンテキスト参照の一覧ページ

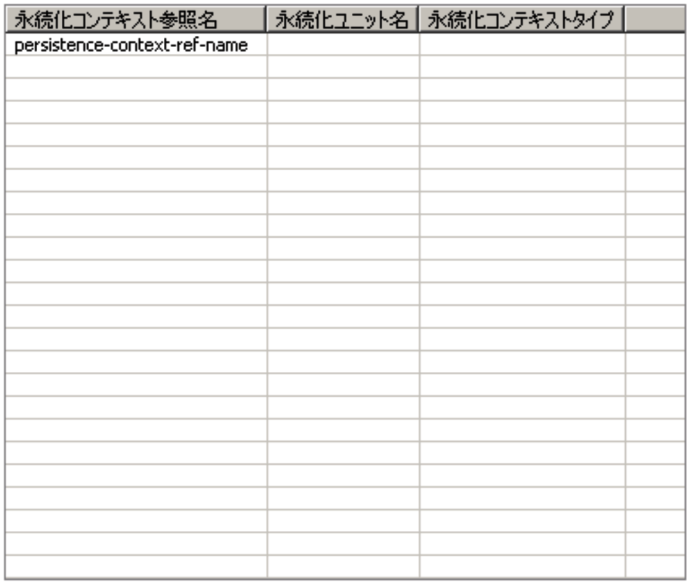


表 21-124　永続化コンテキスト参照の一覧の表示項目

項目	説明	対応するタグ
永続化コンテキスト参照名	永続化コンテキストの参照名が表示されます。	<persistence-context-ref-name> 永続化コンテキストの参照名 </persistence-context-ref-name>
永続化ユニット名	永続化ユニット名が表示されます。	<persistence-unit-name> 永続化ユニット名 </persistence-unit-name>
永続化コンテキストタイプ	永続化コンテキストのタイプが表示されます。	<persistence-context-type> 永続化コンテキストタイプ </persistence-context-type>

(44) 永続化コンテキスト参照のプロパティページ

永続化コンテキスト参照のプロパティページを次に示します。

図 21-139　永続化コンテキスト参照のプロパティページ

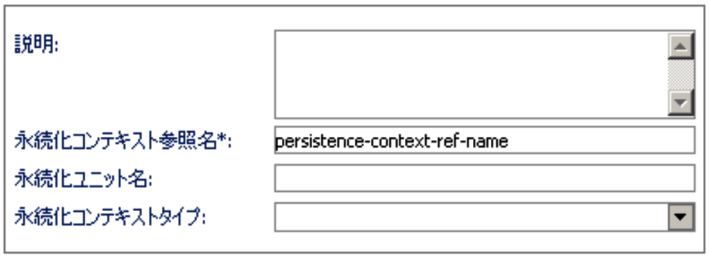


表 21-125　永続化コンテキスト参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
永続化コンテキスト参照名	○	永続化コンテキストの参照名を指定します。	<persistence-context-ref-name> 永続化コンテキストの参照名 </persistence-context-ref-name>
永続化ユニット名	×	永続化ユニットの名前を指定します。	<persistence-unit-name> 永続化ユニット名 </persistence-unit-name>
永続化コンテキストタイプ	×	永続化コンテキストのタイプを選択します。 • （空白） • Transaction • Extended	<persistence-context-type> 永続化コンテキストタイプ </persistence-context-type>

（凡例）　○：必須　×：任意

(45) 永続化プロパティの一覧のページ

永続化プロパティの一覧には，設定されたプロパティがリストで表示されます。

永続化プロパティの一覧ページを次に示します。

図 21-140　永続化プロパティの一覧ページ

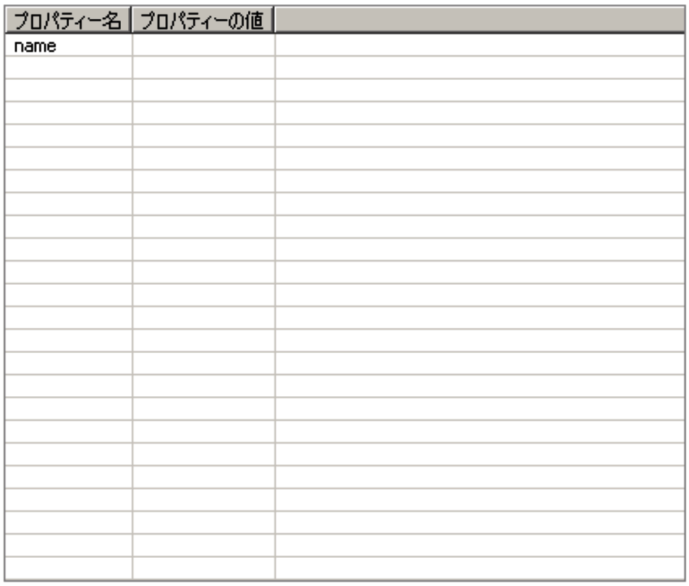


表 21-126　永続化プロパティの一覧の表示項目

項目	説明	対応するタグ
プロパティー名	永続化プロパティの名前が表示されます。	<name> 永続化プロパティ名 </name>
プロパティーの値	永続化プロパティの値が表示されます。	<value> 永続化プロパティの値 </value>

(46) 永続化プロパティのプロパティページ

永続化プロパティのプロパティページを次に示します。

図 21-141　永続化プロパティのプロパティページ

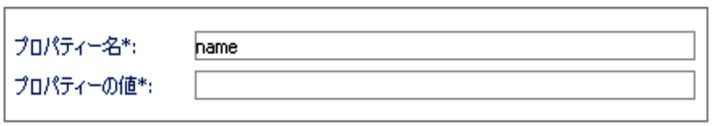


表 21-127　永続化プロパティのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
プロパティー名	○	永続化プロパティの名前を指定します。	<name> 永続化プロパティ名 </name>

項目	必須／任意	説明	対応するタグ
プロパティーの値	○	永続化プロパティの値を指定します。	<value> 永続化プロパティの値 </value>

（凡例） ○：必須

(47) 永続化ユニット参照の一覧のページ

永続化ユニット参照の一覧には，設定された永続化ユニット参照がリストで表示されます。

永続化ユニット参照の一覧ページを次に示します。

図 21-142 永続化ユニット参照の一覧ページ

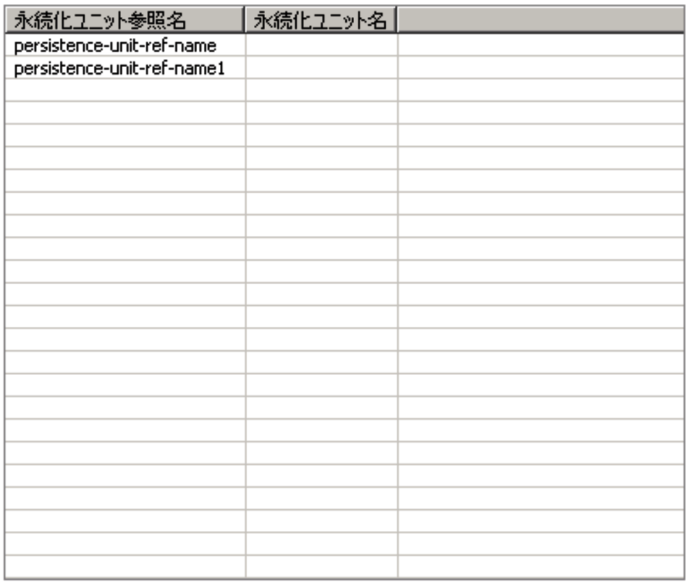


表 21-128 永続化ユニット参照の一覧の表示項目

項目	説明	対応するタグ
永続化ユニット参照名	永続化ユニットの参照名が表示されます。	<persistence-unit-ref-name> 永続化ユニットの参照名 </persistence-unit-ref-name>
永続化ユニット名	永続化ユニット名が表示されます。	<persistence-unit-name> 永続化ユニット名 </persistence-unit-name>

(48) 永続化ユニット参照のプロパティページ

永続化ユニット参照のプロパティページを次に示します。

図 21-143　永続化ユニット参照のプロパティページ

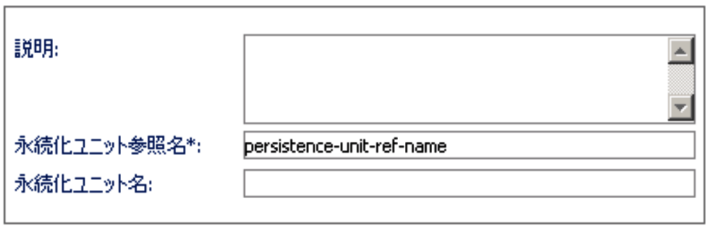


表 21-129　永続化ユニット参照のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	説明を指定します。	<description> 説明 </description>
永続化ユニット参照名	○	永続化ユニットの参照名を指定します。	<persistence-unit-ref-name> 永続化ユニットの参照名 </persistence-unit-ref-name>
永続化ユニット名	×	永続化ユニット名を指定します。	<persistence-unit-name> 永続化ユニット名 </persistence-unit-name>

（凡例）　○：必須　×：任意

(49) PostConstruct メソッドの一覧のページ

PostConstruct メソッドの一覧には，設定された PostConstruct メソッドがリストで表示されます。

PostConstruct メソッドの一覧ページを次に示します。

図 21-144　PostConstruct メソッドの一覧ページ

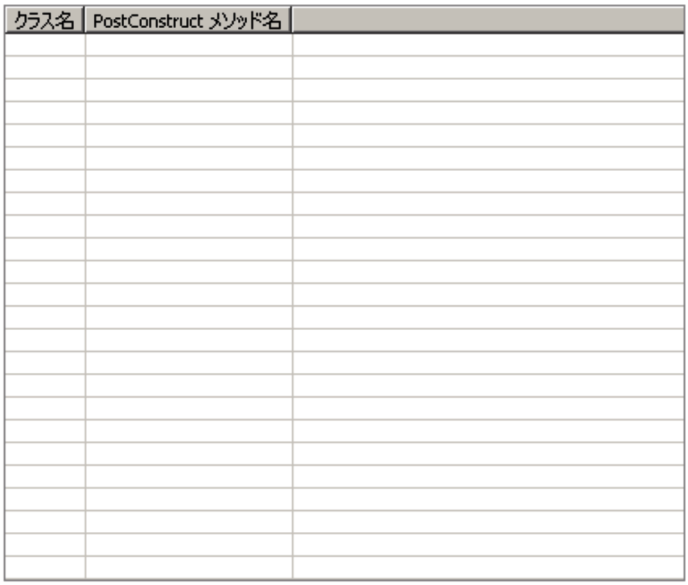


表 21-130　PostConstruct メソッドの一覧の表示項目

項目	説明	対応するタグ
クラス名	PostConstruct メソッドが存在するクラスが表示されます。	<lifecycle-callback-class> クラス名 </lifecycle-callback-class>
PostConstruct メソッド名	PostConstruct メソッド名が表示されます。	<lifecycle-callback-method> PostConstruct メソッド名 </lifecycle-callback-method>

(50) PostConstruct メソッドのプロパティページ

PostConstruct メソッドのプロパティページを次に示します。

図 21-145　PostConstruct メソッドのプロパティページ

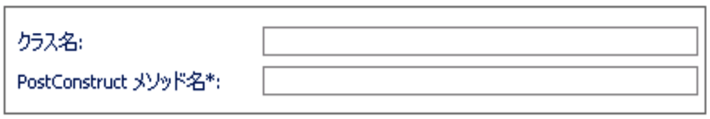


表 21-131　PostConstruct メソッドのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
クラス名	×	PostConstruct メソッドが存在するクラスを完全修飾名で指定します。	<lifecycle-callback-class> クラス名 </lifecycle-callback-class>
PostConstruct メソッド名	○	PostConstruct メソッド名を指定します。	<lifecycle-callback-method> PostConstruct メソッド名 </lifecycle-callback-method>

（凡例）　○：必須　×：任意

(51) PreDestroy メソッドの一覧のページ

PreDestroy メソッドの一覧には，設定された PreDestroy メソッドがリストで表示されます。

PreDestroy メソッドの一覧ページを次に示します。

図 21-146　PreDestroy メソッドの一覧ページ

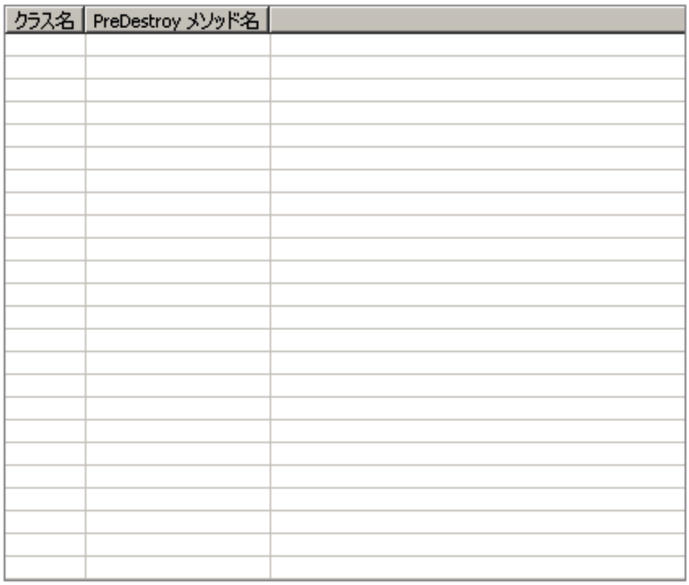


表 21-132　PreDestroy メソッドの一覧の表示項目

項目	説明	対応するタグ
クラス名	PreDestroy メソッドが存在するクラスが完全修飾名で表示されます。	<lifecycle-callback-class> クラス名 </lifecycle-callback-class>

項目	説明	対応するタグ
PreDestroy メソッド名	PreDestroy メソッド名が表示されます。	<lifecycle-callback-method> PreDestroy メソッド名 </lifecycle-callback-method>

(52) PreDestroy メソッドのプロパティページ

PreDestroy メソッドのプロパティページを次に示します。

図 21-147　PreDestroy メソッドのプロパティページ

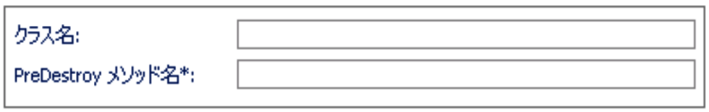


表 21-133　PreDestroy メソッドのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
クラス名	×	PreDestroy メソッドが存在するクラスを完全修飾名で指定します。	<lifecycle-callback-class> クラス名 </lifecycle-callback-class>
PreDestroy メソッド名	○	PreDestroy メソッド名を指定します。	<lifecycle-callback-method> PreDestroy メソッド名 </lifecycle-callback-method>

（凡例）　○：必須　×：任意

(53) ロケールエンコーディングマッピングの一覧ページ

ロケールエンコーディングマッピングの一覧ページを次に示します。

ロケールエンコーディングマッピングの一覧には，設定されたロケールエンコーディングマッピングがリストで表示されます。

図 21-148　ロケールエンコーディングマッピングの一覧ページ

表 21-134　ロケールエンコーディングマッピングの一覧の表示項目

項目	説明	対応するタグ
ロケール	ロケールが表示されます。	<locale-encoding-mapping-list> <locale-encoding-mapping> <locale> ロケール </locale> <locale-encoding-mapping> </local-encoding-mapping-list>
エンコーディング	エンコーディングが表示されます。	<locale-encoding-mapping -list> <locale-encoding-mapping> <encoding> エンコーディング </encoding> <locale-encoding-mapping> </local-encoding-mapping-list>

(54) ロケールエンコーディングマッピングのプロパティページ

ロケールエンコーディングマッピングのプロパティページを次に示します。

図 21-149　ロケールエンコーディングマッピングのプロパティページ

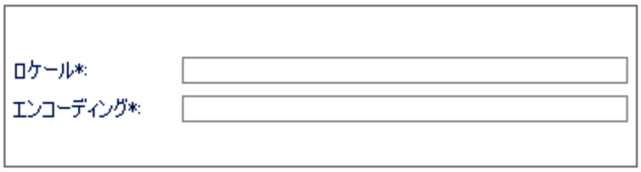


表 21-135 ロケールエンコーディングマッピングのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ロケール	○	ISO-639-1 および ISO-3166 で定義されているロケールを指定します。	<locale> ロケール </locale>
エンコーディング	○	ロケールに対応したエンコーディングを指定します。	<encoding> エンコーディング </encoding>

（凡例） ○：必須

(55) スレッド数制御のプロパティページ

スレッド数制御のプロパティページを次に示します。

図 21-150 スレッド数制御のプロパティページ

表 21-136 スレッド数制御のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
最大同時実行スレッド数	○	最大同時実行スレッド数を「1」から「1024」の整数で指定します。 ［占有スレッド数］に指定した値以上の値を指定してください。	<thread-control> <thread-control-max-threads> 最大同時実行スレッド数 </thread-control-max-threads> </thread-control>
占有スレッド数	○	占有スレッド数を「0」から「1024」の整数で指定します。 占有スレッド数を使用しない場合は「0」を指定します。 ［最大同時実行スレッド数］以下の値を指定てください。	<thread-control> <thread-control-exclusive-threads> 占有スレッド数 </thread-control- exclusive-threads> </thread-control>
実行待ちキュー・サイズ	○	Web アプリケーション単位の実行待ちキューサイズを「0」から「2147483647」の整数で指定します。	<thread-control> <thread-control-queue-size> Web アプリケーション単位の実行待ちキューサイズ </thread-control-queue-size> </thread-control>

（凡例）　○：必須

WAR 属性のツリービューで＜ WAR 属性名＞を選択して，右クリックでコンテキストメニュー［追加］-［スレッド数制御］を選択したときに，Web アプリケーション単位の実行待ちキューサイズ（<thread-control-queue-size> タグ）のあとに，リソース監視（<resource-watcher> タグ）が自動設定されます。スレッド数制御設定時のリソース監視の設定項目を次の表に示します。

表 21-137　スレッド数制御設定時のリソース監視の設定項目

設定項目	説明	対応するタグ
アラートメッセージを出力するしきい値	アラートメッセージを出力するしきい値（単位：%）を，「1」から「100」の整数で設定します。 指定がない場合，「80」が仮定されます。	<watcher-threshold> </watcher-threshold>
監視間隔	監視間隔を「1（秒）」から「2147483647（秒）」の整数で指定します。 指定がない場合，「30」が仮定されます。	<watcher-interval> </watcher-interval>
Web アプリケーション単位のリクエスト実行待ちキュー格納数の監視の有無	Web アプリケーション単位のリクエスト実行待ちキュー格納数の監視を有効にするかどうかを指定します。 指定できる値を次に示します。 • true：有効にする • false：無効にする ただし，実行待ちキュー・サイズの指定が 0 の場合は，「true」が指定されていても無効になります。 指定がない場合，「true」が仮定されます。	<watcher-enabled> </watcher-enabled>
リソース使用状況の出力有無	リソース使用状況をファイルに出力するかどうかを指定します。指定できる値を次に示します。 • true：出力する • false：出力しない 指定がない場合，「true」が仮定されます。	<watcher-writefile-enabled> </watcher-writefile-enabled>

(56) URL グループのスレッド数制御の一覧ページ

URL グループのスレッド数制御の一覧ページを次に示します。

URL グループのスレッド数制御の一覧には，設定された URL グループのスレッド数制御がリストで表示されます。

図 21-151　URL グループのスレッド数制御の一覧ページ

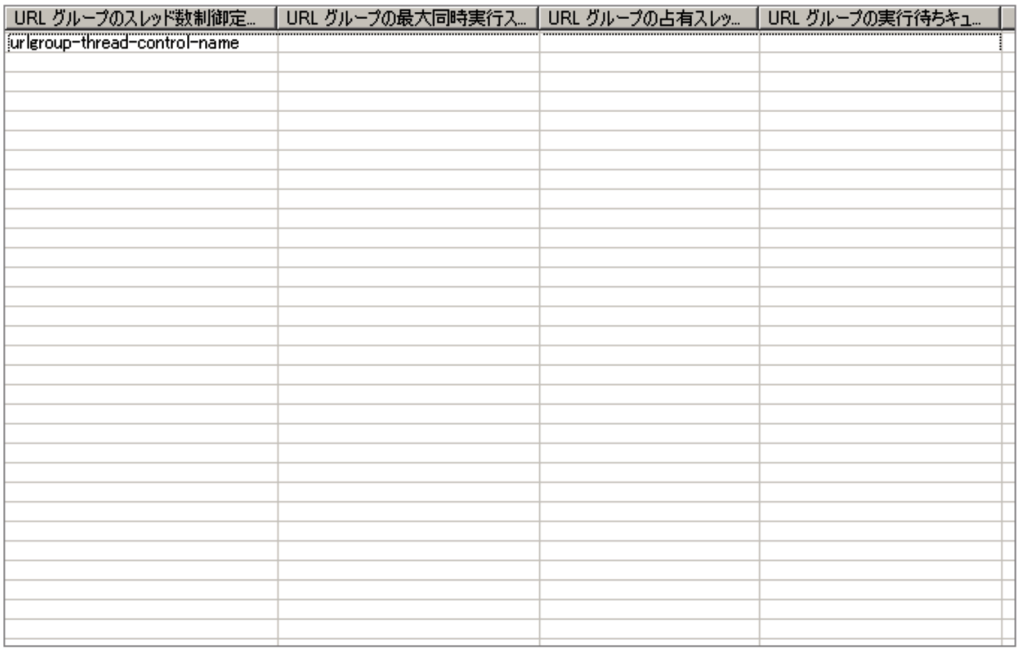


表 21-138　URL グループのスレッド数制御の一覧の表示項目

項目	説明	対応するタグ
URL グループのスレッド数制御定義名	URL グループのスレッド数制御定義名が表示されます。	<urlgroup-thread-control> <urlgroup-thread-control-name> URL グループのスレッド数制御定義名 </urlgroup-thread-control-name> </urlgroup-thread-control>
URL グループの最大同時実行スレッド数	URL グループ単位の最大同時実行スレッド数が表示されます。	<urlgroup-thread-control> <urlgroup-thread-control-max-threads> URL グループの最大同時実行スレッド数 </urlgroup-thread-control-max-threads> </thread-control>

項目	説明	対応するタグ
URL グループの占有スレッド数	URL グループ単位の占有スレッド数が表示されます。	<urlgroup-thread-control> <urlgroup-thread-control-exclusive-threads> URL グループの占有スレッド数 </urlgroup-thread-control-exclusive-threads> </urlgroup-thread-control>
URL グループの実行待ちキュー・サイズ	URL グループ単位の実行待ちキューのサイズが表示されます。	<urlgroup-thread-control> <urlgroup-thread-control-queue-size> URL グループの実行待ちキューサイズ </urlgroup-thread-control-queue-size> </urlgroup-thread-control>

(57) URL グループのスレッド数制御のプロパティページ

URL グループのスレッド数制御のプロパティページを次に示します。

図 21-152　URL グループのスレッド数制御のプロパティページ

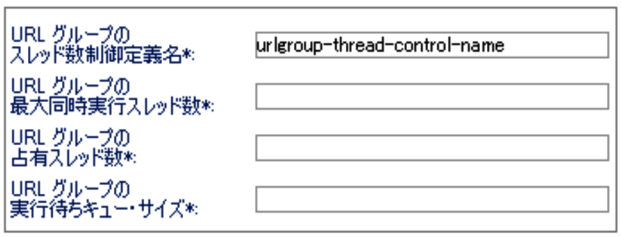


表 21-139　URL グループのスレッド数制御のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
URL グループのスレッド数制御定義名	○	URL グループのスレッド数制御の定義名を英数字 (0 ～ 9，A ～ Z，a ～ z)，ハイフン（-），アンダースコア（_）で指定します。 Web アプリケーション内で一意な名称を，最大 64 文字で指定します。 WAR 属性内に，複数の URL グループのスレッドがある場合，同じ URL グループのスレッド数制御定義名は指定できません。	<urlgroup-thread-control> <urlgroup-thread-control-name> URL グループのスレッド数制御定義名 </urlgroup-thread-control-name > </urlgroup-thread-control>

項目	必須／任意	説明	対応するタグ
URL グループの最大同時実行スレッド数	○	URL グループの最大同時実行スレッド数を，「1」から［最大同時実行スレッド数］までの値を指定します。	<urlgroup-thread-control> <urlgroup-thread-control-max-threads> URL グループの最大同時実行スレッド数 </urlgroup-thread-control-max-threads> </thread-control>
URL グループの占有スレッド数	○	URL グループの占有スレッド数を指定します。 URL グループのスレッド数制御の［URL グループの最大スレッド数］および［URL グループの占有スレッド数］に指定した値以下の値で指定してください。 なお，URL グループの占有スレッド数を使用しない場合は「0」を指定します。 複数指定する場合は，次の範囲の値を指定してください。 • WAR のスレッド数制御の最大同時実行スレッド数と WAR のスレッド数制御の占有スレッド数（<thread-control-exclusive-threads>）が等しくない場合 WAR のスレッド数制御の占有スレッド数 ≧ URL グループの占有スレッド数の総和 • WAR のスレッド数制御の最大同時実行スレッド数と WAR のスレッド数制御の占有スレッド数（<thread-control-exclusive-threads>）が等しい場合 WAR のスレッド数制御の占有スレッド数 > URL グループの占有スレッド数の総和	<urlgroup-thread-control> <urlgroup-thread-control-exclusive-threads> URL グループの占有スレッド数 </urlgroup-thread-control-exclusive-threads> </urlgroup-thread-control>
URL グループの実行待ちキュー・サイズ	○	URL グループの実行待ちキューのサイズを「0」から「2147483647」の数値で指定します。	<urlgroup-thread-control> <urlgroup-thread-control-queue-size> URL グループの実行待ちキュー・サイズ </urlgroup-thread-control-queue-size> </urlgroup-thread-control>

（凡例） ○：必須

(58) URL グループのスレッド数制御マッピングの一覧ページ

URL グループのスレッド数制御マッピングの一覧ページを次に示します。

URL グループのスレッド数制御マッピングの一覧には，設定された URL パターンがリストで表示されます。

図 21-153　URL グループのスレッド数制御マッピングの一覧ページ

表 21-140　URL グループのスレッド数制御マッピングの一覧の表示項目

項目	説明	対応するタグ
URL グループのスレッド数制御マッピング	URL グループのスレッド数制御の対象となる URL パターンが表示されます。	<urlgroup-thread-control-mapping> <url-pattern> URL グループのスレッド数制御マッピング </url-pattern> </urlgroup-thread-control-mapping>

(59) URL グループのスレッド数制御マッピングのプロパティページ

URL グループのスレッド数制御マッピングのプロパティページを次に示します。

図 21-154　URL グループのスレッド数制御マッピングのプロパティページ

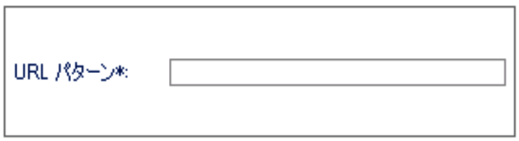


表 21-141　URL グループのスレッド数制御マッピングのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
URL パターン	○	スレッド数制御対象となる URL パターンを指定します。一つの Web アプリケーション内で同じ URL パターンを指定することはできません。Servlet2.4 以降の規定に従って指定してください。	<urlgroup-thread-control-mapping> <url-pattern> URL パターン </url-pattern> </urlgroup-thread-control-mapping>

（凡例）　○：必須

(60) 実行待ちリクエスト数の監視のプロパティページ

実行待ちリクエスト数の監視のプロパティページを次に示します。

図 21-155　実行待ちリクエスト数の監視のプロパティページ

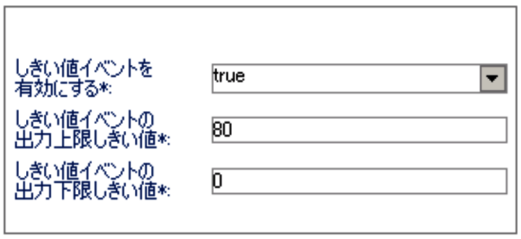


表 21-142　実行待ちリクエスト数の監視のプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
しきい値イベントを有効にする	○	URL グループ単位の実行待ちリクエスト数の監視のしきい値イベントを有効にするかどうかを指定します。 指定できる値を次に示します。 • true 有効にする • false 無効にする	<enabled> </enabled>
しきい値イベントの出力上限しきい値	○	しきい値イベントを出力する上限しきい値を，「1（%）」から「100（%）」の整数で指定します。 指定がない場合，「80（%）」が仮定されます。 ［しきい値イベントを出力する下限しきい値］に指定した値以上の値を指定してください。	<high-threshold> しきい値イベントの出力上限しきい値 </high-threshold>

項目	必須／任意	説明	対応するタグ
しきい値イベントの出力下限しきい値	○	しきい値イベントを出力する下限しきい値を「0（%）」から「99（%）」の整数で指定します。 指定がない場合，「0（%）」が仮定されます。 ［しきい値イベントを出力する上限しきい値］に指定した値以下の値を指定してください。	<low-threshold> しきい値イベントの出力下限しきい値 </low-threshold>

（凡例）　○：必須

(61) HTTP リクエストのプロパティページ

HTTP リクエストのプロパティページを次に示します。

図 21-156　HTTP リクエストのプロパティページ

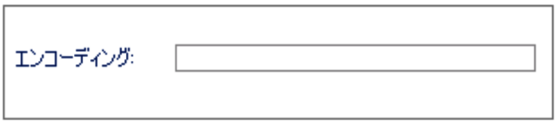


表 21-143　HTTP リクエストのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
エンコーディング	×	リクエストボディ，およびクエリのデコードに使用するデフォルトのエンコーディングを指定します。指定できる文字列は，java.nio API 用の正準名および java.io と java.lang API 用の正準名に記載されている文字エンコーディングです。 J2EE アプリケーションで文字エンコーディングが指定されていない場合に適用されます。	<encoding> エンコーディング </encoding>

（凡例）　×：任意

(62) HTTP レスポンスのプロパティページ

HTTP レスポンスのプロパティページを次に示します。

図 21-157　HTTP レスポンスのプロパティページ

表 21-144　HTTP レスポンスのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
エンコーディング	×	レスポンスボディに使用するデフォルトのエンコーディングを指定します。指定できる文字列は，java.nio API 用の正準名および java.io と java.lang API 用の正準名に記載されている文字エンコーディングです。J2EE アプリケーションで文字エンコーディングが指定されていない場合に適用されます。	<encoding> エンコーディング </encoding>

（凡例）　×：任意

(63) JSP のプロパティページ

JSP のプロパティページを次に示します。

図 21-158　JSP のプロパティページ

表 21-145　JSP レスポンスのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
ページ・エンコーディング	×	JSP ファイルの読み込みに使用するデフォルトのエンコーディングを指定します。指定できる文字列は，java.nio API 用の正準名および java.io と java.lang API 用の正準名に記載されている文字エンコーディングです。 J2EE アプリケーションで文字エンコーディングが指定されていない場合に適用されます。	<page-encoding> エンコーディング </page-encoding>

（凡例） ×：任意

21.10　アプリケーション統合属性ファイル（フィルタ属性）

アプリケーション統合属性ファイルの［WAR］タブを選択すると，［WAR 属性］フォームページが表示されます。［WAR］フォームページについては，「21.9　アプリケーション統合属性ファイル（WAR 属性）」を参照してください。

［WAR 属性］フォームページの［エレメント］に表示されている，［＜フィルタ属性名＞］を選択すると，フィルタ属性のツリーが表示されます。

21.10.1　フィルタ属性のツリー

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。エディタエリアの［エレメント］下に編集するフィルタ属性ファイルの構成がツリー形式で表示されます。

フィルタ属性のツリー構成を次に示します。

フィルタ属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。フィルタ属性のツリーノードとコンテキストメニューについて，次に説明します。

（1）フィルタ属性のツリーノード

フィルタ属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。

表 21-146　フィルタ属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜フィルタ属性名＞	フィルタ属性を示すノードです。 選択すると，フィルタのプロパティページが表示されます。	<hitachi-filter-property>	21.10.2　(3)
アイコン	アイコンを示すノードです。 選択すると，アイコンのプロパティページが表示されます。	<icon>	21.4.2　(2)

ノード	説明	対応するタグ名	プロパティページ
初期化パラメーター	初期化パラメタをまとめるノードです。 選択すると，パラメタの一覧ページが表示されます。	－	21.9.2 (3)
＜初期化パラメタ名＞	初期化パラメタを示すノードです。 選択すると，パラメタのプロパティページが表示されます。	<init-param>	21.9.2 (4)

（凡例） －：該当なし

(2) コンテキストメニュー操作

フィルタ属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-147 フィルタ属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
＜フィルタ属性名＞	［追加］-［アイコン］	フィルタ属性にアイコンが追加されます。 アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.4.2 (2)
	［追加］-［初期化パラメタの追加］	フィルタ属性に初期化パラメタが追加されます。 パラメタのプロパティページが表示されます。	21.9.2 (4)
アイコン	［削除］	フィルタ属性からアイコンが削除されます。	－
初期化パラメーター	［追加］-［初期化パラメーター］	［初期化パラメーター］に初期化パラメタが追加されます。 パラメタのプロパティページが表示されます。	21.9.2 (4)
	［削除］	フィルタ属性からすべての初期化パラメタが削除されます。	－
＜初期化パラメタ名＞	［削除］	フィルタ属性から選択した初期化パラメタが削除されます。	－

（凡例） －：該当なし

21.10.2 フィルタ属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に，編集するフィルタ属性のプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアの表示が切り替わります。

フィルタ属性には，次のプロパティページがあります。

- アイコンのプロパティページ
- フィルタの一覧ページ
- フィルタのプロパティページ

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

(1) アイコンのプロパティページ

フィルタ属性のアイコンのプロパティページについては，「21.4.2　アプリケーション属性設定ページ」の「(2)　アイコンのプロパティページ」を参照してください。

(2) フィルタの一覧ページ

フィルタの一覧ページを次に示します。

フィルタの一覧ページには，設定されたフィルタがリストで表示されます。

図 21-159　フィルタの一覧ページ

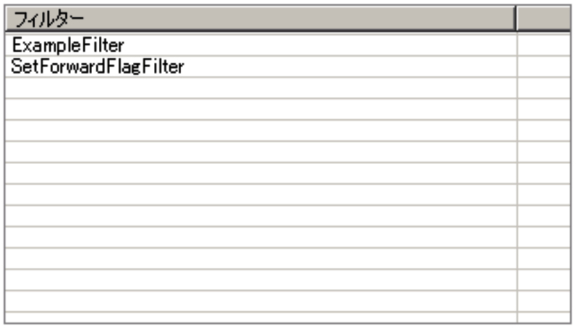


表 21-148　フィルタの一覧の表示項目

項目	説明	対応するタグ
フィルター	フィルタの表示名が表示されます。	<hitachi-filter-property> <display-name> フィルタ </display-name> </hitachi-filter-property>

(3) フィルタのプロパティページ

フィルタのプロパティページを次に示します。

図 21-160　フィルタのプロパティページ

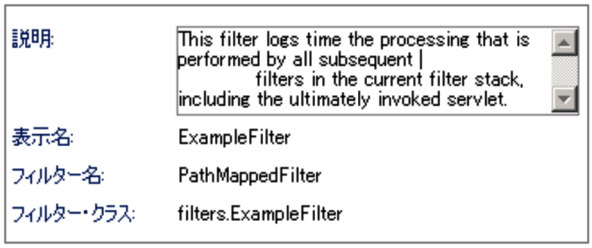


表 21-149　フィルタのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	フィルタについての説明を指定します。	<description> 説明 </description>
表示名	－	フィルタの表示名が表示されます。	<display-name> 表示名 </display-name>
フィルター名	－	フィルタの名前が表示されます。	<filter-name> フィルタ名 </filter-name>
フィルター・クラス	－	フィルタの実装クラスが表示されます。	<filter-class> フィルタクラス </filter-class>

（凡例）×：任意　－：変更不可

21.11　アプリケーション統合属性ファイル（サーブレット属性）

アプリケーション統合属性ファイルの［WAR］タブを選択すると，［WAR 属性］フォームページが表示されます。［WAR］フォームページについては，「21.9　アプリケーション統合属性ファイル（WAR 属性）」を参照してください。

［WAR 属性］フォームページの［エレメント］に表示されている，［＜サーブレット属性名＞］を選択すると，サーブレット属性のツリーが表示されます。

21.11.1　サーブレット属性のツリー

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。エディタエリアの［エレメント］下に編集するサーブレット属性ファイルの構成がツリー形式で表示されます。

サーブレット属性のツリー構成を次に示します。

サーブレット属性のツリー項目（ノード）を右クリックすると，コンテキストメニューが表示されます。サーブレット属性のツリーノードとコンテキストメニューについて，次に説明します。

(1) サーブレット属性のツリーノード

サーブレット属性のツリービューに表示されるノードを次に示します。表示されているノードを選択すると，対応するプロパティページが表示されます。

表 21-150　サーブレット属性ファイルのツリーのノード

ノード	説明	対応するタグ名	プロパティページ
＜サーブレット属性名＞	サーブレット属性を示すノードです。 選択すると，サーブレットのプロパティページが表示されます。	<hitachi-servlet-property>	21.11.2　(3)

ノード	説明	対応するタグ名	プロパティページ
アイコン	アイコンを示すノードです。 選択すると，アイコンのプロパティページが表示されます。	<icon>	21.4.2 (2)
初期化パラメーター	初期化パラメタをまとめるノードです。 選択すると，パラメタの一覧ページが表示されます。	－	21.9.2 (3)
＜初期化パラメタ名＞	初期化パラメタを示すノードです。 選択すると，パラメタのプロパティページが表示されます。	<init-param>	21.9.2 (4)
URL パターン	URL パターンをまとめるノードです。 選択すると，URL パターンの一覧ページが表示されます。	－	21.11.2 (4)
＜ URL パターン＞	URL パターンを示すノードです。 選択すると，URL パターンのプロパティページが表示されます。	<url-pattern>	21.11.2 (5)

（凡例）　－：該当なし

(2) コンテキストメニュー操作

サーブレット属性のツリービューに表示されるノードは，次のコンテキストメニューを持ちます。コンテキストメニューは，マウスの右クリックで表示します。

表 21-151　サーブレット属性のコンテキストメニュー

ノード	コンテキストメニュー	説明	プロパティページ
サーブレット属性	［追加］-［アイコン］	サーブレット属性にアイコンが追加されます。 アイコンのプロパティページが表示されます。 すでにアイコンがある場合は選択できません。	21.4.2 (2)
	［追加］-［初期化パラメーター］	サーブレット属性に初期化パラメタが追加されます。 パラメタのプロパティページが表示されます。	21.9.2 (4)
	［追加］-［URL パターン］	サーブレット属性に URL パターンが追加されます。 URL パターンのプロパティページが表示されます。	21.11.2 (5)
アイコン	［削除］	サーブレット属性からアイコンが削除されます。	－

ノード	コンテキストメニュー	説明	プロパティページ
初期化パラメーター	［追加］-［初期化パラメーター］	［初期化パラメーター］に初期化パラメタが追加されます。 パラメタのプロパティページが表示されます。	21.9.2 (4)
	削除	サーブレット属性からすべての初期化パラメタが削除されます。	－
＜初期化パラメーター名＞	削除	サーブレット属性から選択した初期化パラメタが削除されます。	－
URL パターン	［追加］-［URL パターン］	［URL パターン］に URL パターンが追加されます。 URL パターンのプロパティページが表示されます。	21.11.2 (5)
	削除	サーブレット属性からすべての URL パターンが削除されます。	－
＜ URL パターン＞	削除	サーブレット属性から選択した URL パターンが削除されます。	－

（凡例） －：該当なし

21.11.2 サーブレット属性設定ページ

エディタエリアには，［エレメント］と［エレメントの詳細］が表示されています。［エレメントの詳細］に編集するサーブレット属性ファイルのプロパティページが表示されます。

エディタエリアの［エレメント］のツリーのノードを選択すると，エディタエリアの表示が切り替わります。

サーブレット属性ファイルには，次のプロパティページがあります。

- アイコンのプロパティページ
- サーブレットの一覧ページ
- サーブレットのプロパティページ
- URL パターンの一覧ページ
- URL パターンのプロパティページ

注意事項

プロパティページで，設定が必須の項目には「*」が付与されています。

(1) アイコンのプロパティページ

サーブレット属性のアイコンのプロパティページについては，「21.4.2 アプリケーション属性設定ページ」の「(2) アイコンのプロパティページ」を参照してください。

(2) サーブレットの一覧ページ

サーブレットの一覧ページを次に示します。

サーブレットの一覧には，設定されたサーブレットがリストで表示されます。

図 21-161　サーブレットの一覧ページ

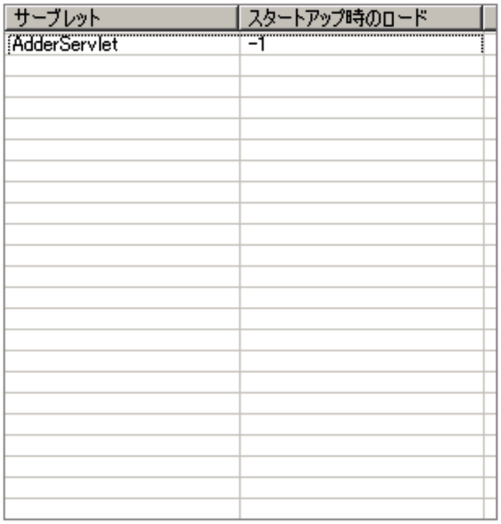


表 21-152　サーブレットの一覧の表示項目

項目	説明	対応するタグ
サーブレット	サーブレットの表示名が表示されます。	<hitachi-servlet-property> <display-name> サーブレット </display-name> </hitachi-servlet-property>
スタートアップ時のロード	スタートアップ時にロードするかどうかの指定が表示されます。	<hitachi-servlet-property> <load-on-startup> スタートアップ時のロード </load-on-startup> </hitachi-servlet-property>

(3) サーブレットのプロパティページ

サーブレットのプロパティページを次に示します。

図 21-162 サーブレットのプロパティページ

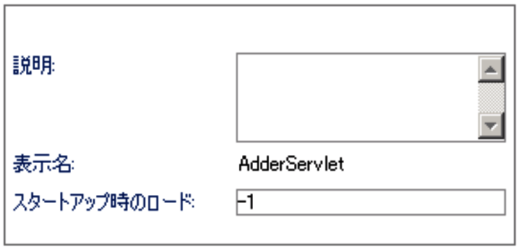


表 21-153 サーブレットのプロパティページの設定項目

項目	必須／任意	説明	対応するタグ
説明	×	サーブレットについての説明を指定します。	<description> 説明 </description>
表示名	－	サーブレットの表示名が表示されます。	<display-name> 表示名 </display-name>
スタートアップ時のロード	×	web アプリケーションのスタートアップ時にサーブレットをロードするかどうか，およびサーブレットのロード順を「-2147483648」から「2147483647」の整数で指定します。 正の整数を指定した場合は，その昇順でロードされます。 負の整数を指定した場合は，必要になった時にロードされます。 ロード順を指定しない場合，「-1」が仮定されます。	<load-on-startup> スタートアップ時のロード </load-on-startup>

（凡例） ×：任意 －：変更不可

(4) URL パターンの一覧ページ

URL パターンの一覧ページを次に示します。

URL パターンの一覧には，設定された URL パターンがリストで表示されます。

図 21-163　URL パターンの一覧ページ

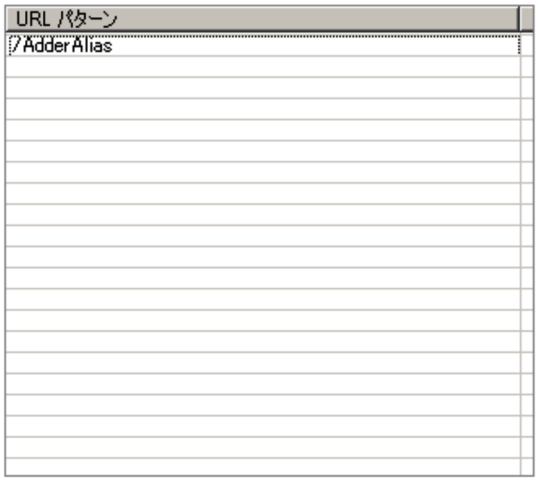


表 21-154　URL パターンの一覧の表示項目

項目	説明	対応するタグ
URL パターン	サーブレットをマッピングする URL パターンが表示されます。 サーブレット属性に，複数の URL パターンがある場合，同じ URL パターンは指定できません。	<url-pattern> URL パターン </url-pattern>

(5) URL パターンのプロパティページ

URL パターンのプロパティページを次に示します。

図 21-164　URL パターンのプロパティページ

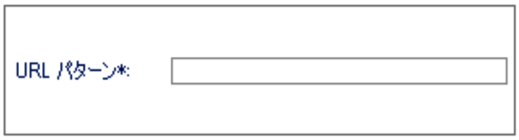


表 21-155　URL パターンのプロパティページの設定項目

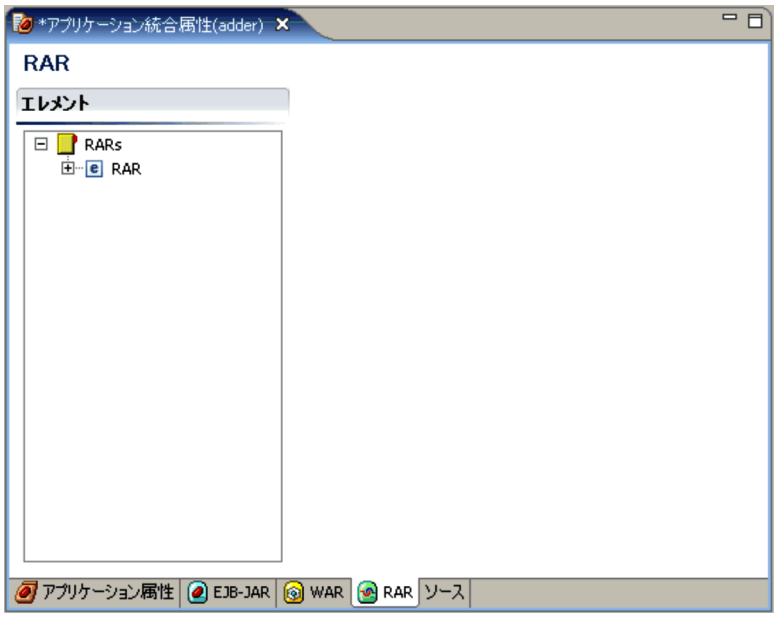
項目	必須／任意	説明	対応するタグ
URL パターン	○	サーブレットをマッピングする URL パターンを指定します。	<url-pattern> URL パターン </url-pattern>

（凡例）　○：必須

21.12　アプリケーション統合属性ファイル（RAR 属性）

アプリケーション統合属性ファイルの［RAR］タブを選択すると，エディタエリアに，J2EE アプリケーションに含まれるリソースアダプタの［RAR］フォームページが表示されます。

図 21-165　［RAR］フォームページ例



フォームページの［エレメント］には，Connector 属性ファイルの構成がツリーで表示されます。［エレメントの詳細］に編集する Connector 属性ファイルのプロパティページが表示されます。アプリケーション統合属性ファイルで編集できる Connector 属性は，Connector 属性ファイルと同じです。Connector 属性ファイルのツリーについては，「21.2.1　Connector 属性のツリー」を，Connector 属性ファイルのプロパティページについては，「21.2.2　Connector 属性設定ページ」を参照してください。

付録

付録 A　ベーシックモードの利用（互換用機能）

J2EE サーバの動作モードには，ベーシックモードと 1.4 モード（06-71 までのスタンダードモードまたはアドバンスドモード）があります。

ここでは，旧バージョンとの互換用機能としてのベーシックモードについて説明します。また，ベーシックモードで使用できる機能の一覧を示します。

付録 A.1　ベーシックモードの概要

ベーシックモードは，単一のデータベースだけのリソースをトランザクションで使用するシステムに適用できます。旧バージョンとの互換性を重視したモードです。

ベーシックモードでは，データソースを使用してデータベースに接続します。データソースを使用してデータベースと接続するための設定については，「付録 A.2　データベースと接続するための設定（データソースを使用する場合）」を参照してください。

J2EE アプリケーションがデータソースを呼び出す場合のリファレンスの定義については，「付録 A.4　データソースへのリファレンス定義」を参照してください。

なお，1.4 モードとベーシックモードでは使用できる機能に差異があります。ベーシックモードで使用できる機能については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編」の「9.2　使用できる機能の一覧」を参照してください。差異のない機能については，本文を参照してください。

付録 A.2　データベースと接続するための設定（データソースを使用する場合）

ここでは，ベーシックモードでのリソース接続について説明します。

データベースと接続するために必要なデータソースの設定をします。データソースの設定は，サーバ管理コマンドで実行します。

データソースを使用してデータベースと接続するには，J2EE サーバをベーシックモードで動作させる必要があります。

データソースの設定の流れを次に示します。

1. Cosminexus DABroker Library **の環境設定をします。**
2. **データソースをインポートします。**
3. **プロパティを定義します。**
4. **接続を確認します。**

インポートしたデータソースは，削除およびコピーができます。データソースの削除については，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「cjdeleteres（リソースの削除）」を参照してください。データソースのコピーについては，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjcopyres（リソースのコピー）」を参照してください。

（1） Cosminexus DABroker Library の環境設定

Cosminexus DABroker Library は，Cosminexus DABroker Library の環境設定ユティリティ（Windows の場合），または Cosminexus DABroker Library 動作環境定義ファイル（UNIX の場合）で設定します。設定方法については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編」の「11.　Cosminexus DABroker Library を使用したデータベース接続」を参照してください。

（2） データソースのインポート

データソースまたは JDBC ドライバを J2EE サーバにインポートします。

次に示すコマンドを実行してデータソースをインポートします。

実行形式

cjimportres［＜サーバ名称＞］［-nameserver ＜プロバイダ URL ＞］-type datasource -resname ＜データソースの表示名＞ -c ＜データソース設定ファイルパス＞ -f ＜ JDBC ドライバファイルパス＞

実行例

```
cjimportres MyServer -type datasource -resname JdbcDbpsv -c
JdbcDbpsvConf.xml -f JdbcDbpsv.jar
```

cjimportres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjimportres（リソースのインポート）」を参照してください。

注意事項

- JDBC のデータソースとしてインポートできる Cosminexus DABroker Library の JAR ファイルは，JdbcDbpsv.jar だけです。JdbcDbpsvEX.jar はサポートされていません。また，JdbcDbpsv.jar は，「＜ Cosminexus のインストールディレクトリ＞¥DABJ」ディレクトリ（Windows の場合）または「/opt/Cosminexus/DABJ」ディレクトリ（UNIX の場合）に配置されている必要があります。
- データソースの表示名には，すでにあるデータソースの名前やメールサーバ構成情報の名前を指定しないでください。
- インポート時に指定した JAR ファイル名は，作業ディレクトリ中のディレクトリ名として用いられます。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 JAR ファイル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，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「7.10.2　作業ディレクトリのパス長の見積もり式」を参照してください。

- データソースを削除するまで，J2EE サーバ用ユーザプロパティファイル（usrconf.properties）の ejbserver.http.port キーの値は変更しないでください。変更した場合は，データソースを再作成してください。

(3) データソースのプロパティ定義

データソースのプロパティを定義します。プロパティの設定手順については，「3.3　属性ファイルによるプロパティの設定」を参照してください。

(a) 編集する属性ファイル

データソース属性ファイル

(b) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して，データソースのデータソース属性ファイルを取得します。

実行形式

```
cjgetresprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -type datasource
-resname ＜データソースの表示名＞ -c ＜属性ファイルパス＞
```

実行例

```
cjgetresprop MyServer -type datasource -resname JdbcDbpsv -c
JdbcDbpsvProp.xml
```

● 属性の設定

次に示すコマンドを実行して，データソース属性ファイルの値を反映します。

実行形式

```
cjsetresprop ［<サーバ名称>］ ［-nameserver ＜プロバイダURL>］ -type datasource
-resname ＜データソースの表示名＞ -c ＜属性ファイルパス＞
```

実行例

```
cjsetresprop MyServer -type datasource -resname JdbcDbpsv -c
JdbcDbpsvProp.xml
```

(c) 編集する属性設定項目

データソースのプロパティ設定項目を次に示します。

- データソースの一般情報
- 実行時プロパティ
- コネクションプーリング情報

● データソースの一般情報

データソースの一般情報の設定項目を次に示します。

項目	対応するタグ
データソースの説明	<description>
データソース名	<display-name>

● 実行時プロパティ

データソースの実行時プロパティ（<property> タグ）の設定項目を次に示します。

項目	対応するタグ
プロパティ名	<name>
XADataSource インタフェースのプロパティ値	<XADataSource>

注　データソースについて設定する項目は，インポートしたデータベースの種類によって異なります。データベースに固有の項目については，そのデータベースのマニュアルを参照してください。

定義するプロパティの数だけ，<property> タグの設定を繰り返してください。

プロパティの設定項目については，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「4.4　データソース属性ファイル」を参照してください。

● コネクションプーリング情報

プーリングとはデータソースとデータベースのコネクションをあらかじめ確立しておき，接続時のオーバーヘッドを減らす手法です。データソースがデータベースコネクションプーリングをサポートする場合，データベースコネクションプールの管理方法を指定できます。コネクションプーリング情報は，プールにあらかじめどのくらいのコネクションを用意しておくのか，また，プールにある不要なコネクションを削除するタイミングなどを指定します。

コネクション取得時にエラーとなった場合に，アプリケーションにエラーを返さないで，繰り返し取得を試みることができます。プーリングしてあるすべてのコネクションが使用中で一時的に取得失敗になる場合など，本機能によってアプリケーションにエラーを返すことなくコネクションを再取得できます。

データソースのコネクションプーリング情報，およびコネクション再取得情報（<PoolConfiguration> タグ）の設定項目を次に示します。

項目	説明	対応するタグ
コネクションプールにプールするコネクションの最小値	データベースコネクションプールに割り当てるデータベースコネクションの最小数を指定します。この最小値に達すると，データベースコネクションはそれ以上削除されません。 定常状態の同時コネクション数を指定してください。	<MinimumSize>
コネクションプールにプールするコネクションの最大値	データベースコネクションプールで使用できるデータベースコネクションの最大数を指定します。この最大値に達すると，それ以上のデータベースコネクションは追加されません。-1 を指定すると，無制限にコネクションを使用できます。この場合，データベースで許容される最大コネクション数によって最大数は制限されます。 同時コネクション数の最大値を指定してください。	<MaximumSize>
コネクションプールにプールするコネクションのしきい値	データベースコネクションプールにデータベースコネクションを追加するしきい値を指定します。つまり，プールにある未使用コネクションの数がここで指定した値以下になった場合，データベースコネクションをプールに追加します。	<Threshold>
コネクションプールにプールするコネクションの増分値	しきい値に達した場合にデータベースコネクションプールに追加するデータベースコネクションの数を指定します。	<GrowthIncrement>
コネクション取得待ち時間（秒）	データベースコネクションプールからデータベースコネクションを取得しようとするときに，コネクション取得のリトライをするまでのアプリケーションの待ち時間を秒単位で指定します。指定した時間が経過してもリトライしない場合，エラーが返されます。コネクションが使用できない場合に待たないようにするときは 0 を指定し，コネクションを取得できるまで待つようにするときは -1 を指定します。	<WaitTimeout>
コネクション接続タイムアウト値（秒）	データベースコネクションの保持時間を秒単位で指定します。指定した時間が経過すると，次のスイーパサイクルでデータベースコネクションを削除します。コネクションタイムアウトの経過後，最小数よりも多く使用できるコネクションがプール内にある場合，そのコネクションは削除されます。プール内に最小数のコネクションだけがある場合，そのコネクションはリオープンされます。	<ConnectionTimeout>

項目	説明	対応するタグ
スイーパ起動時間	スイーパサイクルの間隔を秒単位で指定します。スイーパサイクルは，すべてのデータベースコネクションをチェックし，<ConnectionTimeout> で指定した時間よりも古いコネクションを削除します。	<SweeperInterval>
コネクション取得のリトライ回数	コネクションの取得に失敗して，再度取得を行う場合のリトライ回数を指定します。	<RetryCount>
コネクション取得をリトライするまでの時間（秒）	コネクションの取得に失敗して，再度取得を行う場合のリトライ間隔を指定します。	<RetryInterval>

プーリングを使用しない場合，すべてのフィールドに 0 を指定します。この場合，データソースは，非プーリングデータソースのように動作します。つまり，データベースコネクションを保持しません。また，データベースコネクションが必要な場合，要求側は，データソースがインタフェースを取得するまで待たなければなりません。したがって，性能にかなりの影響を与えることになります。

コネクションプーリング機能を使用する場合の，コネクションプーリングとコネクションスイーパの動作については，「付録 A.3　コネクションプーリングとコネクションスイーパの動作」を参照してください。

(4) データソースの接続テスト

データソースに設定した内容が正しいかどうか，接続テストによって検証します。

次に示すコマンドを実行してデータソースの接続を確認します。

実行形式

cjtestres ［＜サーバ名称＞］［-nameserver ＜プロバイダ URL ＞］-type datasource -resname ＜データソースの表示名＞ ［-resname ＜データソースの表示名＞］

実行例

```
cjtestres -type datasource -resname Myds1
```

cjtestres コマンドの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」の「cjtestres（リソースの接続テスト）」を参照してください。

付録 A.3　コネクションプーリングとコネクションスイーパの動作

データソースのプロパティで設定するコネクションプーリングとコネクションスイーパ

の動作，コネクションを共有する場合の留意点について説明します。

コネクションプーリングについては，マニュアル「Cosminexus アプリケーションサーバ 機能解説 基本・開発編 (コンテナ共通機能)」の「3.14.1　コネクションプーリング」を参照してください。コネクションスイーパについては，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「3.15.7　コネクションスイーパ」を参照してください。

(1) コネクションプーリングの動作

- サーバ上での最初のアクセス時（具体的には最初の javax.sql.DataSource#getConnection メソッド発行時），コネクションプールにプールするコネクションの最小値（<MinimumSize>）の値の JDBC コネクションを確立（接続）しプールします。データベースサーバの制限によって <MinimumSize> の値のコネクションが確立できない場合，<MinimumSize> の値以下の状態になります。
- ユーザアプリケーションプログラムが JDBC コネクションを取得（javax.sql.DataSource#getConnection メソッド）すると，プール内の未使用状態のコネクションが一つ選択され，ユーザアプリケーションプログラムに渡されます。選択された該当コネクションは，プール内で使用中状態となります。
- ユーザアプリケーションプログラムが JDBC コネクションを解放（java.sql.Connection#close メソッド）すると，解放した JDBC コネクションに異常がなく，再利用できる場合，該当する JDBC コネクションはプール内で未使用状態に戻ります。解放したコネクションが再利用できない場合，該当する JDBC コネクションは破棄されます。
- 未使用状態のコネクションが，コネクションプールにプールするコネクションのしきい値（<Threshold>）の値以下になると，プール内のコネクションを，コネクションプールにプールするコネクションの増分値（<GrowthIncrement>）の値ずつ増加させます。DB サーバの制限によって <GrowthIncrement> の値のコネクションが確立できない場合は，増分は <GrowthIncrement> の値以下になります。
- ユーザアプリケーションプログラムが JDBC コネクションを取得しようとしたときにプールに未使用コネクションがない場合，コネクション取得待ち時間（<WaitTimeout>）の値の間，待たされます。このあと，まだプールに未使用コネクションがない場合，ユーザアプリケーションプログラムに例外が通知されます。

(2) コネクションスイーパの動作

- 前回のコネクションスイーパの動作が終了してから，スイーパ起動時間（<SweeperInterval>）の値の経過後，コネクションスイーパが動作します。
- コネクションスイーパは，プール内の未使用状態コネクションを次の観点で監視します。
  - 該当コネクションの生成時点からの経過時間が，コネクション接続タイムアウト値（<ConnectionTimeout>）の値未満の場合は何もしません。

- 該当コネクションの生成時点からの経過時間が，<ConnectionTimeout> の値以上の場合はコネクションを破棄します。
- プールに残ったコネクション数が，<MinimumSize> の値以下の場合，コネクションスイーパは <MinimumSize> の値までコネクションを確立します。

（3） コネクションを共有する場合の留意点

コネクションシェア機能は，同一 JavaVM 内という条件で，同一トランザクション（ここでのトランザクションとは javax.transaction.UserTransaction によるトランザクションまたは EJB での CMT によるトランザクションだけ）中での同一データソース（javax.sql.DataSource）へのコネクション取得要求（getConnection メソッド）に対して，アプリケーションが同一の物理コネクションを共有できるようにするものです。これによって，アプリケーションがトランザクション中で同じデータソースから複数回コネクションを取得して SQL を発行した場合，トランザクションコミット時にそれぞれの SQL の処理結果をデータベースにまとめて反映できます。この機能を利用するには，コネクションプールの設定でコネクション数 <MinimumSize> を 1 以上に設定する必要があります。なお，コネクションシェア機能は，同一データソース（javax.sql.DataSource）に対し，パラメタなし getConnection メソッドと，パラメタあり getConnection（java.lang.String username，java.lang.String password）メソッド間では，同一トランザクション中でもコネクションを共有できません。また，パラメタありの getConnection（java.lang.String username，java.lang.String password）メソッド間で username が異なる場合，同一トランザクション中でもコネクションを共有できません。

（4） 注意事項

- データソースの表示名（<display-name>）には，すでにあるデータソースの名前やメールサーバ構成情報の名前を指定しないでください。
- コネクションプーリングの設定を変更した場合，設定内容を有効にするために J2EE サーバを再起動してください。
- コネクションプーリング機能を利用する場合には，データベースサーバ側からコネクションを強制的に切断する機能（タイムアウトなど）を利用しないでください。

付録 A.4　データソースへのリファレンス定義

J2EE アプリケーションを構成する，Enterprise Bean および Web アプリケーション（サーブレットおよび JSP）がデータソースを呼び出す場合，データソースのリファレンスを解決するためのプロパティを設定します。

（1） Enterprise Bean のリファレンス定義

J2EE アプリケーションを構成する Enterprise Bean が，データソースを参照している場合，リファレンスを解決するためのプロパティを設定します。

(a) 編集する属性ファイル

J2EE アプリケーションを構成する Enterprise Bean の種類に対応する属性ファイルを編集します。

- Session Bean 属性ファイル
- Entity Bean 属性ファイル
- Message-driven Bean 属性ファイル

(b) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して Enterprise Bean の属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type ejb -resname <EJB-JAR表示名>/<Enterprise Bean表示名> -c <Enterprise Beanの属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type ejb -resname addr/
cadder_eb -c C:¥home¥adder_ejb.xml
```

● 属性の設定

次に示すコマンドを実行して，Enterprise Bean の属性ファイルの値を反映します。

実行形式

```
cjsetappprop ［<サーバ名称>］［-nameserver <プロバイダ URL >］-name < J2EE アプリケーション名> -type ejb -resname < EJB-JAR 表示名> / < Enterprise Bean 表示名> -c < Enterprise Bean の属性ファイルパス>
```

実行例

```
cjsetappprop MyServer -name adder -type ejb -resname addr/
adder_eb -c C:¥home¥adder_ejb.xml
```

(c) 編集する属性設定項目

データソースのリファレンス設定項目（<resource-ref>）を次に示します。

項目	必須	対応するタグ名
説明	△	<description>
リソース参照名	○	<res-ref-name>
リソースタイプ	○	<res-type>

項目	必須	対応するタグ名
リソース認証方式	○	<res-auth>
リソース共有	△	<res-sharing-scope>
リンク先のデータソース名	△	<linked-to>

（凡例）　○：必須　△：任意

プロパティの設定項目については，次の個所を参照してください。

- 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「3.4.1　Session Bean 属性ファイルの指定内容」
- 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「3.5.1　Entity Bean 属性ファイルの指定内容」
- 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「3.6.1　MessageDrivenBean 属性ファイルの指定内容」

(2) サーブレットと JSP のリファレンス定義

J2EE アプリケーションを構成する Web アプリケーション（サーブレットおよび JSP）がデータソースを呼び出す場合，データソースのリファレンスを解決するためのプロパティを設定します。

(a) 編集する属性ファイル

WAR 属性ファイル

(b) 編集する属性ファイルの取得と属性の設定

● 属性ファイルの取得

次に示すコマンドを実行して WAR 属性ファイルを取得します。

実行形式

```
cjgetappprop ［<サーバ名称>］ ［-nameserver <プロバイダURL>］ -name <J2EEアプリケーション名> -type war -resname <WAR表示名> -c <WAR属性ファイルパス>
```

実行例

```
cjgetappprop MyServer -name adder -type war -resname adder -c C:¥home¥adder_war.xml
```

● 属性の設定

次に示すコマンドを実行して，WAR 属性ファイルの値を反映します。

実行形式

cjsetappprop ［＜サーバ名称＞］［-nameserver ＜プロバイダ URL ＞］-name ＜ J2EE アプリケーション名＞ -type war -resname ＜ WAR 表示名＞ -c ＜ WAR 属性ファイルパス＞

実行例

```
cjsetappprop MyServer -name adder -type war -resname adder -c
C:¥home¥adder_war.xml
```

(c) 編集する属性設定項目

Enterprise Bean が，データソースを参照している場合のプロパティ設定項目と同じです。「(1)　Enterprise Bean のリファレンス定義」の「(c)　編集する属性設定項目」を参照してください。

付録 B　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 B.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** (EJB **コンテナ**) **（**3020-3-U06**）**
  アプリケーションサーバで提供する EJB コンテナの機能，および EJB コンテナに関連する機能（EJB，EJB クライアントなど）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 **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。

- Cosminexus アプリケーションサーバ V8 Cosminexus Reliable Messaging

（3020-3-U21）
- VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　デベロッパーズガイド（3020-3-U28）
- 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）
- スケーラブルデータベースサーバ HiRDB Version 8 解説（UNIX(R) 用）（3000-6-351）
- スケーラブルデータベースサーバ HiRDB Version 8 システム定義（UNIX(R) 用）（3000-6-353）
- スケーラブルデータベースサーバ HiRDB Version 8 解説（Windows(R) 用）（3020-6-351）
- スケーラブルデータベースサーバ HiRDB Version 8 システム定義（Windows(R) 用）（3020-6-353）
- HiRDB Version 9 解説（UNIX(R) 用）（3000-6-451）
- HiRDB Version 9 システム定義（UNIX(R) 用）（3000-6-453）
- HiRDB Version 9 解説（Windows(R) 用）（3020-6-451）
- HiRDB Version 9 システム定義（Windows(R) 用）（3020-6-453）

なお，このマニュアルでは，次のマニュアルについて，対象 OS および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 概説	Cosminexus アプリケーションサーバ 概説
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 (EJB 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 (EJB 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

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 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 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)
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 アプリケーション開発ガイド	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
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
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	Cosminexus アプリケーションサーバ メッセージ 3
Cosminexus アプリケーションサーバ V8 メッセージ 4 監査ログ編	Cosminexus アプリケーションサーバ メッセージ 4
Cosminexus アプリケーションサーバ V8 Cosminexus Reliable Messaging	Cosminexus Reliable Messaging
スケーラブルデータベースサーバ HiRDB Version 8 システム定義（UNIX(R) 用）	HiRDB システム定義
スケーラブルデータベースサーバ HiRDB Version 8 システム定義（Windows(R) 用）	
HiRDB Version 9 システム定義（UNIX(R) 用）	
HiRDB Version 9 システム定義（Windows(R) 用	
スケーラブルデータベースサーバ HiRDB Version 8 解説（UNIX(R) 用）	HiRDB 解説
スケーラブルデータベースサーバ HiRDB Version 8 解説（Windows(R) 用）	
HiRDB Version 9 解説（UNIX(R) 用）	
HiRDB Version 9 解説（Windows(R) 用）	

付録 B.2　このマニュアルでの表記

このマニュアルでは，製品名を次のように表記しています。

表記			製品名
Application Server	Application Server Enterprise		uCosminexus Application Server Enterprise
	Application Server Standard		uCosminexus Application Server Standard
Cosminexus Developer's Kit for Java			Cosminexus Developer's Kit for JavaTM
Developer	Developer Professional		uCosminexus Developer Professional
	Developer Standard		uCosminexus Developer Standard
Eclipse			Eclipse 3.6.1
HiRDB または HiRDB サーバ	HiRDB/Parallel Server		HiRDB/Parallel Server Version 8
	HiRDB/Single Server		HiRDB/Single Server Version 8
	HiRDB Server		HiRDB Server Version 9
HiRDB Run Time または HiRDB クライアント			HiRDB/Developer's Kit Version 8
			HiRDB/Developer's Kit Version 9
			HiRDB/Run Time Version 8
			HiRDB/Run Time Version 9
IPF			Itanium(R) Processor Family
Oracle	Oracle10g		Oracle 10g
			Oracle 10g R2
			Oracle Database 10g
	Oracle11g		Oracle Database 11g
			Oracle Database 11g R2
	Oracle9i		Oracle9i
			Oracle9i R2
UNIX	AIX		AIX 5L V5.3
			AIX V6.1
			AIXV7.1
	HP-UX または HP-UX（IPF）		HP-UX 11i V2（IPF）
			HP-UX 11i V3（IPF）
	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）

表記			製品名
		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）
XDM/RD E2			VOS3 XDM/RD E2

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）
	Red Hat Enterprise Linux(R) 5（Intel Itanium）
	Red Hat Enterprise Linux(R) 5 Advanced Platform（x86）

表記	OS 名
	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
Server Plug-in	Cosminexus Server Plug-in
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
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
JAAS	Java™ Authentication and Authorization Service
JAR	Java™ Archive

表記	Java 関連用語
Java	Java™
JavaBeans	JavaBeans™
Java EE または Java Platform, Enterprise Edition	Java™ Platform, Enterprise Edition
JavaMail	JavaMail™
JavaVM	Java™ Virtual Machine
JCA	J2EE™ Connector Architecture
JDBC	JDBC™
	Java™ Database Connectivity
JDK	JDK™
	Java™ Development Kit
JMS	Java™ Message Service
JNDI	Java Naming and Directory Interface™
JSP	JSP™
	JavaServer Pages™
JSTL	JavaServer Pages™ Standard Tag Library
JTA	Java™ Transaction API
Servlet またはサーブレット	Java™ Servlet
WAR	Web ARchive

付録 B.3　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
CMP	Container-Managed Persistence
CMR	Container-Managed Relationship
CMT	Container-Managed Transaction
CORBA	Common Object Request Broker Architecture
CSV	Comma Separated Value
CUI	Character User Interface

英略語	英字での表記
DB	Database
DBMS	Database Management System
DD	Deployment Descriptor
DIT	Directory Information Tree
DMZ	Demilitarized Zone
DN	Distinguished Name
DNS	Domain Name System
DTD	Document Type Definition
EIS	Enterprise Information System
EL	Expression Language
EUC	Extended UNIX Code
GUI	Graphical User Interface
HA	High Availability
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Security
IIOP	Internet Inter-Orb Protocol
JNI	Java Native Interface
LAN	Local Area Network
MDA	Model Driven Architecture
MIB	Management Information Base
OID	Object Identifier
OMG	Object Management Group
ORB	Object Request Broker
OS	Operating System
OTM	Object Transaction Monitor
OTS	Object Transaction Service
RAC	Real Application Clusters
RDB	Relational Database
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SFO	Session Fail Over
SHA	Secure Hash Algorithm
SNMP	Simple Network Management Protocol
SPI	Service Provider Interface

英略語	英字での表記
SPP	Service Providing Program
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

付録 B.4　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。

索引

記号

C

D

E

H

J

M

P

R

S

U

W

あ

い

う

え

か

き

く

こ

さ

し

す

せ

そ

た

つ

て

と

は

ひ

ふ

へ

ほ

み

め

ゆ

ら

り

る

ろ