HITACHI
Inspire the Next

Cosminexus アプリケーションサーバ V8

リファレンス API 編

文法書

3020-3-U26-40

■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

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

■輸出時の注意

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

■商標類

Active Directory は，米国 Microsoft Corporation の，米国およびその他の国における登録商標または商標です。
AIX は，米国およびその他の国における International Business Machines Corporation の商標です。
AIX 5L は，米国およびその他の国における International Business Machines Corporation の商標です。
AMD は，Advanced Micro Devices, Inc. の商標です。
CORBA は，Object Management Group が提唱する分散処理環境アーキテクチャの名称です。
HP-UX は，Hewlett-Packard Company のオペレーティングシステムの名称です。
IIOP は，OMG 仕様による ORB(Object Request Broker) 間通信のネットワークプロトコルの名称です。
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。
JavaScript は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
JDK は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
JSP は，Oracle Corporation 及びその子会社，関連会社の米国 及びその他の国における登録商標または商標です。
Linux は，Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。
Microsoft は，米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
Microsoft Internet Information Services は，米国 Microsoft Corporation の商品名称です。
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。
Oracle 及び Oracle9i 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 Database 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 Oracle Database 11g 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。
Red Hat は，米国およびその他の国で Red Hat, Inc. の登録商標もしくは商標です。
SOAP（Simple Object Access Protocol）は，分散ネットワーク環境において XML ベースの情報を交換するための通信プロトコルの名称です。
Solaris 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。
すべての SPARC 商標は，米国 SPARC International, Inc. のライセンスを受けて使用している同社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。SPARC 商標がついた製品は，米国 Sun Microsystems, Inc. が開発したアーキテクチャに基づくものです。
SQL Server 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。
Sun 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。
Sun Microsystems 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。
UNIX は，The Open Group の米国ならびに他の国における登録商標です。
Windows 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。
Windows Server 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。
Windows Vista 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。
This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

■マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

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

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

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

■発行

2011 年 7 月 3020-3-U26-40

■著作権

All Rights Reserved. Copyright (C) 2008, 2011, Hitachi, Ltd.

変更内容

変更内容（3020-3-U26-40）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
DatabaseAccessException クラス，および SessionOperationException クラスの例外がスローされる条件を追加した。	3.1
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
JavaVM 内部で暗黙にスレッドを生成する JavaAPI クラスについての説明を追加した。	付録 B
HiRDB Version 9 に対応した。	－
SQL Server 2008 に対応した。これに伴い，使用できる JDBC ドライバに SQL Server JDBC Driver 2.0，および SQL Server JDBC Driver 3.0 を追加した。	－
Microsoft IIS 7.0 および Microsoft IIS 7.5 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-U26-20）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
TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API を追加した。
スレッドの非同期並行処理で使用する API を追加した。
サポートする javax.xml.ws パッケージのアノテーションに，@WebServiceProvider を追加した。
URI によるデータベースセッションフェイルオーバ機能の抑止ができるようになったことに伴い，SessionOperationException クラスの説明を変更した。
日立固有の JavaVM 拡張オプションに -XX:+HitachiExplicitMemoryAutoReclaim（明示管理ヒープ機能の自動解放機能 ON/OFF 指定オプション）を追加した。
Explicit メモリブロックの自動解放明示予約の説明を追加した。
次の製品の適用 OS に Windows Server 2008 R2 を追加した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise • uCosminexus Web Redirector • uCosminexus Service Platform • uCosminexus Client
次の製品の適用 OS から Solaris を削除した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise
次の製品の適用 OS に Windows 7 を追加した。 • uCosminexus Developer Standard • uCosminexus Developer Professional • uCosminexus Service Architect • uCosminexus Client

はじめに

このマニュアルは，Cosminexus（コズミネクサス）のアプリケーションサーバのアプリケーション開発で使用する API ／タグライブラリ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 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が提供する API またはタグライブラリを使用してアプリケーションを開発する方を対象としています。
次の内容を理解されていることを前提としています。

- Windows またはご使用の UNIX の基本操作に関する知識
- Java によるプログラム開発に関する基本的な知識
- CORBA に関する基本的な知識
- J2EE 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
- 使用する IDE 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。

■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。

記述記号

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。
…	記述が省略されていることを示します。 （例）　ABC… 　ABC の後ろに記述があり，その記述が省略されていることを示します。

記号	意　味
＜　＞	この記号で囲まれている項目は，該当する要素を指定することを示します。 （例）＜プロパティ＞ 　プロパティを記述します。
…	この記号の直前に示す記号を繰り返し，複数個指定できることを示します。 （例）＜プロパティ＞… 　プロパティは複数個，繰り返して指定できます。

構文要素

構文要素	定義
英字	A～Z　a～z
英小文字	a～z
英大文字	A～Z
数字	0～9
英数字	A～Z　a～z　0～9
記号	！　"　#　$　%　&　'　（　）　+　，　_　・　．　/　：　；　＜　＝　＞　@　［　］　^　―　｛　｝　～　タブ　空白

注　すべて半角文字を使用してください。

目次

1 API とタグライブラリの概要

この章では，アプリケーションサーバで使用する API とタグライブラリの種類，およびこのマニュアルでの記述形式について説明します。

1.1 API とタグライブラリの種類

アプリケーションサーバで使用する API とタグライブラリの種類について説明します。

このマニュアルでは，アプリケーションごとに使用できる API とタグライブラリを二つに分類して説明します。

● J2EE アプリケーションで使用できる API とタグライブラリ

● バッチアプリケーションまたは EJB クライアントアプリケーションで使用できる API

J2EE アプリケーションで使用できる API とタグライブラリを次の表に示します。

表 1-1 J2EE アプリケーションで使用できる API

API とタグライブラリの種類	API とタグライブラリの説明	参照先
Web コンテナで使用する API	Web コンテナで使用する API です。	3 章
EJB クライアントアプリケーションで使用する API	EJB クライアントのセキュリティや通信タイムアウトなどを設定するための API です。	4 章
TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API	TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API です。	5 章
スレッドの非同期並行処理で使用する API	スレッドの非同期並行処理で使用する API です。	6 章
クライアント性能モニタ機能で使用する API	クライアント性能モニタ機能で使用する API です。	7 章
統合ユーザ管理フレームワークで使用する API	統合ユーザ管理機能を使用する場合に，ユーザ認証を実装するために使用する，統合ユーザ管理フレームワークのライブラリです。	8 章
統合ユーザ管理フレームワークで使用するタグライブラリ	統合ユーザ管理機能を使用する場合に，ユーザ認証を実装するために使用する，統合ユーザ管理フレームワークの JSP タグライブラリです。	9 章
ユーザログ機能で使用する API	J2EE アプリケーションが出力するログ（ユーザログ）を日立トレース共通ライブラリ形式で出力する場合に，ユーザログ出力を実装するための API です。	10 章
監査ログ出力で使用する API	J2EE アプリケーションで監査ログを出力するための API です。	11 章
性能解析トレースで使用する API	性能解析トレースでアプリケーションサーバの処理性能を解析する場合に，ルートアプリケーション情報を文字列表現で取得するための API です。	12 章
JavaVM で使用する API	Java プログラムから直接ガーベージコレクションのメモリ情報を取得するための API です。	13 章
Cosminexus DABroker Library で使用する API	Cosminexus DABroker Library を使用してデータベースに接続する場合に，データベースの情報などを設定するための API です。	14 章

なお，API とタグライブラリのほかに，アノテーションと Dependency Injection も使用できます。アノテーションと Dependency Injection については，「2. アプリケーションサーバが対応しているアノテーションおよび Dependency Injection」を参照してください。

バッチアプリケーションまたは EJB クライアントアプリケーションで使用できる API を次の表に示します。

表 1-2 バッチアプリケーションまたは EJB クライアントアプリケーションで使用できる API

API とタグライブラリの種類	API とタグライブラリの説明	参照先
EJB クライアントアプリケーションで使用する API	EJB クライアントアプリケーションのセキュリティや通信タイムアウトなどを設定するための API です。	4 章
ユーザログ機能で使用する API	バッチアプリケーションまたは EJB クライアントアプリケーションが出力するログ（ユーザログ）を日立トレース共通ライブラリ形式で出力する場合に，ユーザログ出力を実装するための API です。	10 章
監査ログ出力で使用する API	バッチアプリケーションまたは EJB クライアントアプリケーションで監査ログを出力するための API です。	11 章
性能解析トレースで使用する API	性能解析トレースでアプリケーションサーバの処理性能を解析する場合に，ルートアプリケーション情報を文字列表現で取得するための API です。	12 章
JavaVM で使用する API	Java プログラムから直接ガーベージコレクションのメモリ情報を取得するための API です。	13 章
Cosminexus DABroker Library で使用する API	Cosminexus DABroker Library を使用してデータベースに接続する場合に，データベースの情報などを設定するための API です。	14 章

参考

サーブレットエンジンモードで使用する API の種類を次に示します。

- 統合ユーザ管理フレームワークで使用する API
- 統合ユーザ管理フレームワークで使用するタグライブラリ
- ユーザログ機能で使用する API
- 性能解析トレースで使用する API
- JavaVM で使用する API
- Cosminexus DABroker Library で使用する API

1.2 アノテーションの記述形式

2 章では，アノテーションについて次の形式で説明します。なお，各アノテーションはアルファベットの順に説明します。

(1) 説明

アノテーションの機能について説明します。

(2) 属性

アノテーションに含まれる属性について説明します。各属性については，次の形式で説明します。

(a) 属性名

型
　属性の型を示します。

説明
　属性の機能について説明します。

デフォルト値
　属性のデフォルト値を示します。

1.3 API の記述形式

3 章から 6 章，および 8 章から 12 章では，API について次の形式で説明します。なお，各 API は，アルファベットの順に説明します。

説明
API の機能について説明します。

形式
API の記述形式を示します。

パラメタ
API のパラメタについて説明します。

例外
API を利用する際に発生する例外について説明します。

戻り値
API の戻り値について説明しています。

注意事項
API を利用する上での注意事項について説明します。

1.4 タグライブラリの記述形式

7 章では，タグライブラリについて次の形式で説明します。なお，各タグライブラリは，アルファベットの順に説明します。

説明

タグライブラリの機能について説明します。

タグ属性

タグライブラリの属性について，表で説明します。

2 アプリケーションサーバが対応しているアノテーションおよび Dependency Injection

この章では，アプリケーションサーバが対応しているアノテーションおよび Dependency Injection について説明します。
なお，アノテーション参照抑止機能を使用している場合，アノテーションの指定は参照されません。アノテーション参照抑止機能については，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「11.5 アノテーションの参照抑止」を参照してください。

2.1 対応するアノテーションのサポート範囲

アノテーションは，ソースコードに注釈を付けることができる言語仕様です。

アプリケーションサーバが対応しているアノテーションの一覧を次に示します。

2.1.1 javax.annotation パッケージに含まれるアノテーションのサポート範囲

javax.annotation パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

（1） WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-1 WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@PostConstruct	○	○	○	－	○	○	×	－
@PreDestroy	○	○	○	－	○	○	×	－
@Resource	○	○	○	－	○	○	×	－
@Resources	○	○	○	－	○	○	×	－

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

（2） WAR ファイル（Servlet 2.4 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-2 WAR ファイル（Servlet 2.4 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@Resource ※1	○	○	○	－	○	○※2	×	－
@Resources ※1	○	○	○	－	○	○	×	－

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

注※ 1
アプリケーションサーバでは，EJB2.1 仕様，または Servlet2.4 仕様のアプリケーションでもサポートしています。

注※ 2
Servlet2.4 仕様の Web アプリケーションではメソッドおよびフィールドに対する DI ができません。

（3） EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-3 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@PostConstruct	—	○	—	×	○	○	—	—
@PreDestroy	—	○	—	×	○	○	—	—
@Resource	—	○	—	×	○	○	—	—
@Resources	—	○	—	×	○	○	—	—

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
—：標準仕様で対応していない。

（4） EJB-JAR ファイル（EJB2.1 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-4 EJB-JAR ファイル（EJB2.1 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Enterprise Bean				例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean		
@Resource ※	—	○	—	×	—	—
@Resources ※	—	○	—	×	—	—

（凡例）
○：対応する。

×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

注※
アプリケーションサーバでは，EJB2.1 仕様，または Servlet2.4 仕様のアプリケーションでもサポートしています。

(5) ライブラリ JAR ファイル（サーブレット /JSP）

ライブラリ JAR のサーブレットまたは JSP に記述できるアノテーションの一覧を示します。

表 2-5 ライブラリ JAR（サーブレット /JSP）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様			
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
					クラシックタグハンドラ	シンプルタグハンドラ	
@PostConstruct	－	○	○	－	○	○	×
@PreDestroy	－	○	○	－	○	○	×
@Resource	－	○	○	－	○	○	×
@Resources	－	○	○	－	○	○	×

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

(6) ライブラリ JAR ファイル（Enterprise Bean/ 例外クラス / その他のクラス）

ライブラリ JAR の Enterprise Bean，例外クラス，またはその他のクラスに記述できるアノテーションの一覧を示します。

表 2-6 ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Enterprise Bean					例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ		
@PostConstruct	－	－	－	×	○	－	－

アノテーション名	Enterprise Bean					例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ		
@PreDestroy	―	―	―	×	○	―	―
@Resource	―	―	―	×	○	―	―
@Resources	―	―	―	×	○	―	―

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
―：標準仕様で対応していない。

2.1.2 javax.annotation.security パッケージに含まれるアノテーションのサポート範囲

javax.annotation.security パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

（1） WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-7 WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.annotation.security パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@DeclareRoles	○	○	○	―	―	―	―	―
@RunAs	○	―	―	―	―	―	―	―

（凡例）
○：対応する。
―：標準仕様で対応していない。

（2） EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-8 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.annotation.security パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@DeclareRoles	－	○	－	×	－	－	－	－
@DenyAll	－	○	－	×	－	－	－	－
@PermitAll	－	○	－	×	－	－	－	－
@RolesAllowed	－	○	－	×	－	－	－	－
@RunAs	－	○	－	×	－	－	－	－

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

(3) ライブラリ JAR（サーブレット /JSP）

ライブラリ JAR のサーブレットまたは JSP に記述できるアノテーションの一覧を示します。

表 2-9 ライブラリ JAR（サーブレット /JSP）に記述できるアノテーション（javax.annotation.security パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様			
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
					クラシックタグハンドラ	シンプルタグハンドラ	
@DeclareRoles	－	○	○	－	－	－	－

（凡例）
○：対応する。
－：標準仕様で対応していない。

（4） ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）

ライブラリ JAR の Enterprise Bean，例外クラス，またはその他のクラスに記述できるアノテーションはありません。

2.1.3 javax.ejb パッケージに含まれるアノテーションのサポート範囲

javax.ejb パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

（1） WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-10 WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@EJB	○	○	○	－	○	○	×	－
@EJBs	○	○	○	－	○	○	×	－

（凡例）
○：対応する。

×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

（2） WAR ファイル（Servlet 2.4 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-11　WAR ファイル（Servlet 2.4 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@EJB※1	○	○	○	－	○	○※2	×	－
@EJBs※1	○	○	○	－	○	○	×	－

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

注※ 1
アプリケーションサーバでは，EJB2.1 仕様，または Servlet2.4 仕様のアプリケーションでもサポートしています。

注※ 2
Servlet2.4 仕様の Web アプリケーションではメソッドおよびフィールドに対する DI ができません。

（3） EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-12 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@ApplicationException	－	－	－	－	－	－	○	－
@EJB	－	○	－	×	○	○	－	－
@EJBs	－	○	－	×	○	○	－	－
@Init ※2	－	○	－	－	－	－	－	－
@Local	○	○	－	－	－	－	－	－
@LocalHome	－	○	－	－	－	－	－	－
@Remote	○	○	－	－	－	－	－	－
@RemoteHome	－	○	－	－	－	－	－	－
@Remove ※2	－	○	－	－	－	－	－	－
@Stateful ※2	－	○	－	－	－	－	－	－
@Stateless ※1	－	○	－	－	－	－	－	－
@Timeout ※1	－	○	－	×	－	－	－	－
@TransactionAttribute	－	○	－	×	－	－	－	－
@TransactionManagement	－	○	－	×	－	－	－	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※ 1

Stateless Session Bean の場合にだけ使用できます。

注※ 2

Stateful Session Bean の場合にだけ使用できます。

(4) EJB-JAR ファイル（EJB2.1 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-13 EJB-JAR ファイル（EJB2.1 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Enterprise Bean				例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean		
@EJB※	－	○	－	×	－	－
@EJBs※	－	○	－	×	－	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※

アプリケーションサーバでは，EJB2.1 仕様，または Servlet2.4 仕様のアプリケーションでもサポートしています。

(5) ライブラリ JAR（サーブレット /JSP）

ライブラリ JAR のサーブレットまたは JSP に記述できるアノテーションの一覧を示します。

表 2-14 ライブラリ JAR（サーブレット /JSP）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様			
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
					クラシックタグハンドラ	シンプルタグハンドラ	
@EJB	－	○	○	－	○	○	×
@EJBs	－	○	○	－	○	○	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

（6） ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）

ライブラリ JAR の Enterprise Bean，例外クラス，またはその他のクラスに記述できるアノテーションの一覧を示します。

表 2-15　ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Enterprise Bean					例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ		
@ApplicationException	－	－	－	－	－	○	－
@EJB	－	－	－	－	○	－	－
@EJBs	－	－	－	－	○	－	－
@Local	○	－	－	－	－	－	－
@Remote	○	－	－	－	－	－	－

（凡例）
○：対応する。
－：標準仕様で対応していない。

2.1.4　javax.interceptor パッケージに含まれるアノテーションのサポート一覧

javax.interceptor パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

（1） WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションはありません。

（2） EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-16 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.interceptor パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@AroundInvoke	－	○	－	×	○	○	－	－
@ExcludeClassInterceptors	－	○	－	×	－	－	－	－
@ExcludeDefaultInterceptors	－	○	－	×	－	－	－	－
@Interceptors	－	○	－	×	－	－	－	－

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

(3) ライブラリ JAR（サーブレット /JSP）

ライブラリ JAR のサーブレットまたは JSP に記述できるアノテーションはありません。

(4) ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）

ライブラリ JAR の Enterprise Bean，例外クラス，およびその他のクラスに記述できるアノテーションの一覧を示します。

表 2-17 ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）に記述できるアノテーション（javax.interceptor パッケージ）

アノテーション名	Enterprise Bean					例外クラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ	
@AroundInvoke	―	―	―	×	○	―

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
―：標準仕様で対応していない。

2.1.5 javax.jws パッケージに含まれるアノテーションのサポート範囲

javax.jws パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

なお，各アノテーションの詳細は，マニュアル「Cosminexus アプリケーションサーバ Web サービス開発の手引」の「13.2　Java から WSDL へのマッピングのカスタマイズ」を参照してください。

（1） WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-18 WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.jws パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@HandlerChain	―	―	―	―	―	―	―	○
@OneWay	―	―	―	―	―	―	―	×
@SOAPBinding	―	―	―	―	―	―	―	○
@WebMethod	―	―	―	―	―	―	―	○
@WebParam	―	―	―	―	―	―	―	○
@WebResult	―	―	―	―	―	―	―	○

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@WebService	―	―	―	―	―	―	―	○

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
―：標準仕様で対応していない。

(2) EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。表に記載されているアノテーションは Stateless Session Bean にだけ記述できます。

表 2-19 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@HandlerChain	―	○	―	―	―	―	―	―
@OneWay	―	○	―	―	―	―	―	―
@SOAPBinding	―	○	―	―	―	―	―	―
@WebMethod	―	○	―	―	―	―	―	―
@WebParam	―	○	―	―	―	―	―	―
@WebResult	―	○	―	―	―	―	―	―

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@WebService	―	○	―	―	―	―	―	―

（凡例）
○：対応する。
―：標準仕様で対応していない。

(3) ライブラリ JAR（サーブレット /JSP）

ライブラリ JAR のサーブレットまたは JSP に記述できるアノテーションはありません。

(4) ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）

ライブラリ JAR の Enterprise Bean，例外クラス，およびその他のクラスに記述できるアノテーションはありません。

2.1.6 javax.persistence パッケージに含まれるアノテーションのサポート範囲

javax.persistence パッケージのアノテーションは，JPA プロバイダに依存する場合としない場合で記述できるコンポーネントが異なります。ここでは，JPA プロバイダに依存するアノテーションと JPA プロバイダに依存しないアノテーションに分けて説明します。

(1) JPA プロバイダに依存するアノテーションの場合

JPA プロバイダに依存するアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

(a) WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-20 WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@PersistenceContext	○	○	○	－	○	○	×	－
@PersistenceContexts	○	○	○	－	○	○	×	－
@PersistenceProperty	○	○	○	－	○	○	×	－
@PersistenceUnit	○	○	○	－	○	○	×	－
@PersistenceUnits	○	○	○	－	○	○	×	－

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

(b) EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-21 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@PersistenceContext	—	○	—	×	○	○	—	—
@PersistenceContexts	—	○	—	×	○	○	—	—
@PersistenceProperty	—	○	—	×	○	○	—	—
@PersistenceUnit	—	○	—	×	○	○	—	—
@PersistenceUnits	—	○	—	×	○	○	—	—

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
—：標準仕様で対応していない。

(c) ライブラリ JAR（サーブレット /JSP）

ライブラリ JAR のサーブレットまたは JSP に記述できるアノテーションの一覧を示します。

表 2-22 ライブラリ JAR（サーブレット /JSP）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様			
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
					クラシックタグハンドラ	シンプルタグハンドラ	
@PersistenceContext	—	○	○	—	○	○	×
@PersistenceContexts	—	○	○	—	○	○	×
@PersistenceProperty	—	○	○	—	○	○	×
@PersistenceUnit	—	○	○	—	○	○	×
@PersistenceUnits	—	○	○	—	○	○	×

（凡例）
○：対応する。
×：アプリケーションサーバでは対応しない。
—：標準仕様で対応していない。

(d) ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）

ライブラリ JAR の Enterprise Bean，例外クラス，またはその他のクラスに記述できるアノテーションの一覧を示します。

表 2-23 ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Enterprise Bean					例外クラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ	
@PersistenceContext	—	—	—	×	○	—
@PersistenceContexts	—	—	—	×	○	—
@PersistenceProperty	—	—	—	×	○	—
@PersistenceUnit	—	—	—	×	○	—
@PersistenceUnits	—	—	—	×	○	—

（凡例）
○：対応する。

×：アプリケーションサーバでは対応しない。
－：標準仕様で対応していない。

（2） JPA プロバイダに依存しないアノテーションの場合

JPA プロバイダに依存しないアノテーションは，ファイルの種類に関係なく，エンティティクラス内に記述できます。

javax.persistence パッケージに含まれるアノテーションの一覧については，「2.6 javax.persistence パッケージ」のアノテーション一覧を参照してください。

2.1.7 javax.xml.ws パッケージに含まれるアノテーションのサポート範囲

javax.xml.ws パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

なお，各アノテーションの詳細は，マニュアル「Cosminexus アプリケーションサーバ Web サービス開発の手引」の「13.2　Java から WSDL へのマッピングのカスタマイズ」を参照してください。

（1） WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-24　WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.xml.ws パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@BindingType	－	－	－	－	－	－	－	○
@RequestWrapper	－	－	－	－	－	－	－	○
@RespectBinding	－	－	－	－	－	－	－	○
@ResponseWrapper	－	－	－	－	－	－	－	○
@WebEndpoint	－	－	－	－	－	－	－	○
@WebFault	－	－	－	－	－	－	－	○
@WebServiceClient	－	－	－	－	－	－	－	○

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@WebServiceProvider	―	―	―	―	―	―	―	○

（凡例）
○：対応する。
―：標準仕様で対応していない。

(2) EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-25 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.xml.ws パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@WebEndpoint	―	―	―	―	―	―	―	○
@WebFault	―	―	―	―	―	―	―	○
@WebServiceClient	―	―	―	―	―	―	―	○

（凡例）
○：対応する。
―：標準仕様で対応していない。

（3） ライブラリ JAR（サーブレット /JSP）

ライブラリ JAR のサーブレットまたは JSP に記述できるアノテーションはありません。

（4） ライブラリ JAR（Enterprise Bean/ 例外クラス / その他のクラス）

ライブラリ JAR の Enterprise Bean，例外クラス，およびその他のクラスに記述できるアノテーションはありません。

2.2 javax.annotation パッケージ

javax.annotation パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@PostConstruct	Servlet，Enterprise Bean インスタンスなどが生成された直後にコールバックするメソッドを設定します。
@PreDestroy	Servlet，Enterprise Bean インスタンスなどが削除される直前にコールバックするメソッドを設定します。
@Resource	リソースへの参照を宣言します。
@Resources	@Resource を複数設定します。

それぞれのアノテーションの詳細について，次に説明します。

2.2.1 @PostConstruct

(1) 説明

Servlet，Enterprise Bean インスタンスなどが生成された直後にコールバックするメソッドを設定します。

(2) 属性

@PostConstruct の属性はありません。

2.2.2 @PreDestroy

(1) 説明

Servlet，Enterprise Bean インスタンスなどが削除される直前にコールバックするメソッドを設定します。

(2) 属性

@PreDestroy の属性はありません。

2.2.3 @Resource

(1) 説明

リソースへの参照を宣言します。クラス，メソッド，およびフィールドに設定できます。メソッドやフィールドに設定した場合，Dependency Injection の対象となります。ただ

し，メソッドは set メソッドである必要があります。

(2) 属性

@Resource の属性の一覧を次の表に示します。

属性名	機能
name	リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
type	リソースの Java タイプを設定します。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
authenticationType	リソースに使用する認証タイプを設定します。
shareable	リソースを共用するかどうかを設定します。
mappedName	参照先リソースを特定するためにリソース表示名やキュー名を設定します。
description	リソースの説明を設定します。

各属性の詳細を次に示します。

(a) name 属性

型

String

説明

リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
なお，リソースの別名を指定することもできます。J2EE リソースの別名の設定については，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「2.4.6　J2EE リソースの別名の設定」を参照してください。

デフォルト値

- メソッドに設定した場合
  アノテーションを設定したクラス名 /set メソッドのプロパティ
- フィールドに設定した場合
  アノテーションを設定したクラス名 / フィールド名

(b) type 属性

型

Class

説明

リソースの Java タイプを設定します。アノテーションをメソッドまたはフィールド

に設定する場合，省略できます。

デフォルト値

- メソッド設定した場合
  メソッドの引数の型
- フィールドに設定した場合
  フィールドの型

type 属性と DD の対応

type 属性は J2EE 仕様と異なり，設定値（Java Type）によって対応する DD が変わります。Java Type によって異なる DD の対応を次の表に示します。

表 2-26　type 属性による DD の対応表

type 属性	J2EE 仕様で対応する DD のタグ	Cosminexus 仕様で対応する DD のタグ※1
java.lang.String※2	env-entry	env-entry
java.lang.Character※2	env-entry	env-entry
java.lang.Integer※2	env-entry	env-entry
java.lang.Boolean※2	env-entry	env-entry
java.lang.Double※2	env-entry	env-entry
java.lang.Byte※2	env-entry	env-entry
java.lang.Short※2	env-entry	env-entry
java.lang.Long※2	env-entry	env-entry
java.lang.Float※2	env-entry	env-entry
javax.xml.rpc.Service	service-ref	例外※3
javax.xml.ws.Service	service-ref	例外※3
javax.jws.WebService	service-ref	例外※3
javax.sql.DataSource	resource-ref	resource-ref
javax.jms.ConnectionFactory	resource-ref	resource-ref
javax.jms.QueueConnectionFactory	resource-ref	resource-ref
javax.jms.TopicConnectionFactory	resource-ref	resource-ref
javax.mail.Session	resource-ref	resource-ref
java.net.URL	resource-ref	例外※3
javax.resource.cci.ConnectionFactory	resource-ref	resource-ref
org.omg.CORBA_2_3.ORB	resource-ref	resource-ref
リソースアダプタによって定義されるほかのコネクションファクトリ	resource-ref	resource-env-ref

type 属性	J2EE 仕様で対応する DD のタグ	Cosminexus 仕様で対応する DD のタグ※1
javax.jms.Queue	message-destination-ref	resource-env-ref
javax.jms.Topic	message-destination-ref	resource-env-ref
javax.resource.cci.InteractionSpec	resource-env-ref	例外※3
javax.transaction.UserTransaction	resource-env-ref	resource-env-ref
上記以外のすべてのタイプ	resource-env-ref	resource-env-ref

注※1
mappedName 要素に「!#」が含まれていた場合は，Java Type とは関係なく，
<resource-env-ref> に対応づけます。

注※2
標準 DD から値を取得できないため，属性ファイルには表示しますが，DI は行いません。

注※3
インポート時に例外となります。

(c) authenticationType 属性

型
AuthenticationType

説明
リソースに使用する認証タイプを設定します。

デフォルト値
CONTAINER

(d) shareable 属性

型
boolean

説明
リソースを共用するかどうかを設定します。

デフォルト値
true

(e) mappedName 属性

型
String

説明
参照先リソースを特定するためにリソース表示名やキュー名を設定します。
リソース表示名に半角英数字およびアンダースコア (_) 以外の文字を含む場合，半角

英数字およびアンダースコア (_) 以外の文字をアンダースコア (_) に置き換えて設定してください。

デフォルト値

""

mappedName 属性の設定条件

mappedName 属性は type 属性によって設定条件が変わります。@Resource での mappedName 属性の設定条件を次の表に示します。

表 2-27 @Resource の mappedName() の設定条件

設定条件（Java Type，リソースなど）	使用可否※1
java.lang.String	×
java.lang.Character	×
java.lang.Integer	×
java.lang.Boolean	×
java.lang.Double	×
java.lang.Byte	×
java.lang.Short	×
java.lang.Long	×
java.lang.Float	×
javax.xml.rpc.Service	×
javax.sql.DataSource	○
javax.jms.ConnectionFactory	○
javax.jms.QueueConnectionFactory	○
javax.jms.TopicConnectionFactory	○
javax.mail.Session	○
java.net.URL	×
javax.resource.cci.ConnectionFactory	○
org.omg.CORBA_2_3.ORB	×
javax.jms.Queue※2	○
javax.jms.Topic	○
javax.resource.cci.InteractionSpec	×
javax.transaction.UserTransaction	×
javax.ejb.EjbContext	×
javax.ejb.SessionContext	×
javax.ejb.TimerService	×
JavaBeans リソース	○

（凡例）
○：使用できます。
×：使用できません。

注※ 1
管理対象オブジェクトへの対応づけは，Java Type に関係なく，mappedName 要素で対応づけます。リソースアダプタの表示名と管理対象オブジェクト名の区切り文字には，「!#」を使用してください。

注※ 2
uCosminexus TP1/Message Queue - Access または Cosminexus Reliable Messaging を使用時に javax.jms.Queue を使用する場合，リソースアダプタの表示名とキューの表示名の区切り文字には，「#」を使用してください。

(f) description 属性

型
String

説明
リソースの説明を設定します。

デフォルト値
""

2.2.4 @Resources

(1) 説明

@Resource を複数設定します。なお，クラスにだけ設定できます。

(2) 属性

@Resources の属性の一覧を次の表に示します。

属性名	機能
value	複数のリソース（@Resource）を定義します。

各属性の詳細を次に示します。

(a) value 属性

型
Resource[]

説明
複数のリソース（@Resource）を定義します。

デフォルト値

なし

2.3 javax.annotation.security パッケージ

javax.annotation.security パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@DeclareRoles	セキュリティロールの参照を設定します。
@DenyAll	すべてのセキュリティロールに対して，アクセスを拒否するメソッドに設定します。
@PermitAll	すべてのセキュリティロールに対して，アクセスを許可するクラスまたはメソッドに設定します。
@RolesAllowed	クラスまたはメソッドに対してアクセスを許可するセキュリティロールを設定します。
@RunAs	Servlet，または Enterprise Bean を実行する際に適用するセキュリティロールを設定します。

2.3.1 @DeclareRoles

(1) 説明

セキュリティロールの参照を設定します。なお，クラスにだけ設定できます。

(2) 属性

@DeclareRoles の属性の一覧を次の表に示します。

属性名	機能
value	参照するセキュリティロール名を設定します。

各属性の詳細を次に示します。

(a) value 属性

型
　String[]

説明
　参照するセキュリティロール名を設定します。

デフォルト値
　なし

2.3.2 @DenyAll

(1) 説明

すべてのセキュリティロールに対して，アクセスを拒否するメソッドに設定します。

(2) 属性

@DenyAll の属性はありません。

2.3.3 @PermitAll

(1) 説明

すべてのセキュリティロールに対して，アクセスを許可するクラスまたはメソッドに設定します。

(2) 属性

@PermitAll の属性はありません。

2.3.4 @RolesAllowed

(1) 説明

クラスまたはメソッドに対してアクセスを許可するセキュリティロールを設定します。

(2) 属性

@RolesAllowed の属性の一覧を次の表に示します。

属性名	機能
value	アプリケーションでのメソッドにアクセスするのが許可されたロールリストを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

String[]

説明

アプリケーションでのメソッドにアクセスするのが許可されたロールリストを設定します。

デフォルト値

なし

2.3.5 @RunAs

(1) 説明

Servlet，または Enterprise Bean を実行する際に適用するセキュリティロールを設定します。なお，クラスにだけ設定できます。

(2) 属性

@RunAs の属性の一覧を次の表に示します。

属性名	機能
value	Enterprise Bean を実行する際に適用するセキュリティロール名を設定します。

各属性の詳細を次に示します。

(a) value 属性

型
String

説明
Servlet，または Enterprise Bean を実行する際に適用するセキュリティロール名を設定します。

デフォルト値
なし

2.4 javax.ejb パッケージ

javax.ejb パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@ApplicationException	アプリケーション例外とする例外クラスに設定します。
@EJB	EJB のビジネスインタフェースまたはホームインタフェースへの参照を設定します。
@EJBs	@EJB を複数設定します。
@Init	Stateful Session Bean の Home インタフェースで定義した create<METHOD>() を実行した際，コールバックするメソッドに設定します。
@Local	Enterprise Bean のローカルビジネスインタフェースを設定します。
@LocalHome	ローカルホームインタフェース，およびローカルコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。
@PostActivate	Stateful Session Bean が活性化された直後にコールバックするメソッドに設定します。
@PrePassivate	Stateful Session Bean が非活性化される直前にコールバックするメソッドに設定します。
@Remote	Enterprise Bean のリモートビジネスインタフェースを設定します。アノテーションをインタフェースに設定した場合，そのインタフェースがリモートビジネスインタフェースとなります。
@RemoteHome	リモートホームインタフェース，およびリモートコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。
@Remove	Stateful Session Bean を削除する働きを持つビジネスメソッドに設定します。
@Stateful	Stateful Session Bean のクラスに設定します。
@Stateless	Stateless Session Bean のクラスに設定します。
@Timeout	TimerService 使用時にコールバックするタイムアウトメソッドに設定します。
@TransactionAttribute	Enterprise Bean が CMT で動作する場合のトランザクション属性を設定します。
@TransactionManagement	Enterprise Bean のトランザクション管理種別を設定します。

2.4.1 @ApplicationException

(1) 説明

アプリケーション例外とする例外クラスに設定します。

(2) 属性

@ApplicationException の属性の一覧を次の表に示します。

属性名	機能
rollback	例外発生時にコンテナがトランザクションをロールバックするかどうかを設定します。

各属性の詳細を次に示します。

(a) rollback 属性

型
boolean

説明
例外発生時にコンテナがトランザクションをロールバックするかどうかを設定します。

デフォルト値
false

2.4.2 @EJB

(1) 説明

EJB のビジネスインタフェースまたはホームインタフェースへの参照を設定します。クラス，メソッド，およびフィールドに設定できます。メソッドやフィールドに設定した場合，Dependency Injection の対象となります。ただし，メソッドは set メソッドである必要があります。

(2) 属性

@EJB の属性の一覧を次の表に示します。

属性名	機能
name	リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
beanInterface	ビジネスインタフェースまたはホームインタフェースのクラスを設定します。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。

属性名	機能
beanName	参照する EJB のパッケージなしクラス名を設定します。ただし，参照する EJB クラスを定義するアノテーション（@Stateless, @Stateful）に name 属性が設定されている場合，name 属性の値を設定します。また，DD による定義をサポートする EJB の場合，DD の <ejb-name> タグの値を設定します。
mappedName	属性を指定できますが，サポートしないため，動作しません。
description	参照する EJB の説明を設定します。

各属性の詳細を次に示します。

(a) name 属性

型

String

説明

リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。

デフォルト値

- メソッドに設定した場合
  アノテーションを設定したクラス名 /set メソッドのプロパティ
- フィールドに設定した場合
  アノテーションを設定したクラス名 / フィールド名

(b) beanInterface 属性

型

Class

説明

ビジネスインタフェースまたはホームインタフェースのクラスを設定します。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。

デフォルト値

- メソッド設定した場合
  メソッドの引数の型
- フィールドに設定した場合
  フィールドの型

(c) beanName 属性

型

String

説明

参照する EJB のパッケージなしクラス名を設定します。ただし，参照する EJB クラスを定義するアノテーション（@Stateless，@Stateful）に name 属性が設定されている場合，name 属性の値を設定します。また，DD による定義をサポートする EJB の場合，DD の <ejb-name> タグの値を設定します。

デフォルト値
""

(d) mappedName 属性

型
String

説明
属性を指定できますが，サポートしないため，動作しません。

デフォルト値
なし

(e) description 属性

型
String

説明
参照する EJB の説明を設定します。

デフォルト値
""

2.4.3 @EJBs

(1) 説明

@EJB を複数設定します。なお，クラスにだけ設定できます。

(2) 属性

@EJBs の属性の一覧を次の表に示します。

属性名	機能
value	@EJB を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

EJB[]

説明
@EJB を設定します。

デフォルト値
なし

2.4.4 @Init

(1) 説明

Stateful Session Bean の Home インタフェースで定義した create<METHOD>() を実行した際，コールバックするメソッドに設定します。

(2) 属性

@Init の属性の一覧を次の表に示します。

属性名	機能
value	対応する create<METHOD>() 名を設定します。

各属性の詳細を次に示します。

(a) value 属性

型
String

説明
対応する create<METHOD>() 名を設定します。

デフォルト値
""

2.4.5 @Local

(1) 説明

Enterprise Bean のローカルビジネスインタフェースを設定します。アノテーションをインタフェースに設定した場合，そのインタフェースがローカルビジネスインタフェースとなります。Bean クラスに設定した場合，value 属性にローカルビジネスインタフェースを設定する必要があります。

(2) 属性

@Local の属性の一覧を次の表に示します。

属性名	機能
value	アノテーションを Bean クラスに設定した場合，ローカルビジネスインタフェースのクラスを設定します。

各属性の詳細を次に示します。

(a) value 属性

型
Class[]

説明
アノテーションを Bean クラスに設定した場合，ローカルビジネスインタフェースのクラスを設定します。

デフォルト値
{}

2.4.6 @LocalHome

(1) 説明

ローカルホームインタフェース，およびローカルコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。

(2) 属性

@LocalHome の属性の一覧を次の表に示します。

属性名	機能
value	ローカルホームインタフェースを設定します。

各属性の詳細を次に示します。

(a) value 属性

型
Class

説明
ローカルホームインタフェースを設定します。

デフォルト値

なし

2.4.7 @PostActivate

(1) 説明

Stateful Session Bean が活性化された直後にコールバックするメソッドに設定します。アノテーションを設定できますが，活性化，非活性化の状態変化をサポートしないため，動作しません。

(2) 属性

@PostActivate の属性はありません。

2.4.8 @PrePassivate

(1) 説明

Stateful Session Bean が非活性化される直前にコールバックするメソッドに設定します。アノテーションを設定できますが，活性化，非活性化の状態変化をサポートしないため，動作しません。

(2) 属性

@PrePassivate の属性はありません。

2.4.9 @Remote

(1) 説明

Enterprise Bean のリモートビジネスインタフェースを設定します。アノテーションをインタフェースに設定した場合，そのインタフェースがリモートビジネスインタフェースとなります。Bean クラスに設定した場合，value 属性にリモートビジネスインタフェースを設定する必要があります。

(2) 属性

@Remote の属性の一覧を次の表に示します。

属性名	機能
value	アノテーションを Bean クラスに設定した場合，リモートビジネスインタフェースのクラスを設定します。

各属性の詳細を次に示します。

(a) value 属性

型
Class[]

説明
アノテーションを Bean クラスに設定した場合，リモートビジネスインタフェースのクラスを設定します。

デフォルト値
{}

2.4.10 @RemoteHome

(1) 説明

リモートホームインタフェース，およびリモートコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。

(2) 属性

@RemoteHome の属性の一覧を次の表に示します。

属性名	機能
value	リモートホームインタフェースを設定します。

各属性の詳細を次に示します。

(a) value 属性

型
Class

説明
リモートホームインタフェースを設定します。

デフォルト値
なし

2.4.11 @Remove

(1) 説明

Stateful Session Bean を削除する働きを持つビジネスメソッドに設定します。

(2) 属性

@Remove の属性の一覧を次の表に示します。

属性名	機能
retainIfException	メソッドがアプリケーション例外で異常終了した場合に削除するかどうかを設定します。

各属性の詳細を次に示します。

(a) retainIfException 属性

型
boolean

説明
メソッドがアプリケーション例外で異常終了した場合に削除するかどうかを設定します。

デフォルト値
false

2.4.12 @Stateful

(1) 説明

Stateful Session Bean のクラスに設定します。

(2) 属性

@Stateful の属性の一覧を次の表に示します。

属性名	機能
name	Stateful Session Bean の名称を設定します。
mappedName	属性を指定できますが，サポートしないため，動作しません。
description	Stateful Session Bean の説明を設定します。

各属性の詳細を次に示します。

(a) name 属性

型
String

説明
Stateful Session Bean の名称を設定します。

デフォルト値
Stateful Session Bean のパッケージを除いたクラス名

(b) mappedName 属性

型
String

説明
Stateful Session Bean の別名を設定します。

デフォルト値
なし

(c) description 属性

型
String

説明
Stateful Session Bean の説明を設定します。

デフォルト値
""

2.4.13 @Stateless

(1) 説明

Stateless Session Bean のクラスに設定します。

(2) 属性

@Stateless の属性の一覧を次の表に示します。

属性名	機能
name	Stateless Session Bean の名称を設定します。
mappedName	属性を指定できますが，サポートしないため，動作しません。
description	Stateless Session Bean の説明を設定します。

各属性の詳細を次に示します。

(a) name 属性

型
String

説明
Stateless Session Bean の名称を設定します。

デフォルト値
Stateless Session Bean のパッケージを除いたクラス名

(b) mappedName 属性

型
String

説明
Stateless Session Bean の別名を設定します。

デフォルト値
なし

(c) description 属性

型
String

説明
Stateless Session Bean の説明を設定します。

デフォルト値
""

2.4.14 @Timeout

(1) 説明

TimerService 使用時にコールバックするタイムアウトメソッドに設定します。

(2) 属性

@Timeout の属性はありません。

2.4.15 @TransactionAttribute

(1) 説明

Enterprise Bean が CMT で動作する場合のトランザクション属性を設定します。クラス，およびメソッドに設定できます。

(2) 属性

@TransactionAttribute の属性の一覧を次の表に示します。

属性名	機能
value	トランザクション属性を設定します。

各属性の詳細を次に示します。

(a) value 属性

型
TransactionAttributeType

説明
トランザクション属性を設定します。

デフォルト値
REQUIRED

2.4.16 @TransactionManagement

(1) 説明

Enterprise Bean のトランザクション管理種別を設定します。なお，クラスにだけ設定できます。

(2) 属性

@TransactionManagement の属性の一覧を次の表に示します。

属性名	機能
value	トランザクション管理種別を設定します。

各属性の詳細を次に示します。

(a) value 属性

型
TransactionManagementType

説明
トランザクション管理種別を設定します。

デフォルト値
CONTAINER

2.5 javax.interceptor パッケージ

javax.interceptor パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@AroundInvoke	ビジネスメソッドの呼び出しをインターセプトするメソッドに設定します。
@ExcludeClassInterceptors	クラスインターセプタを適用しないメソッドに設定します。
@ExcludeDefaultInterceptors	デフォルトインターセプタを適用しないクラス，およびメソッドに設定します。
@Interceptors	適用するインターセプタクラスを設定します。クラス，およびメソッドに設定できます。

2.5.1 @AroundInvoke

(1) 説明

ビジネスメソッドの呼び出しをインターセプトするメソッドに設定します。

(2) 属性

@AroundInvoke の属性はありません。

2.5.2 @ExcludeClassInterceptors

(1) 説明

クラスインターセプタを適用しないメソッドに設定します。

(2) 属性

@ExcludeClassInterceptors の属性はありません。

2.5.3 @ExcludeDefaultInterceptors

(1) 説明

デフォルトインターセプタを適用しないクラス，およびメソッドに設定します。

(2) 属性

@ExcludeDefaultInterceptors の属性はありません。

2.5.4 @Interceptors

(1) 説明

適用するインターセプタクラスを設定します。クラス，およびメソッドに設定できます。

(2) 属性

@Interceptors の属性の一覧を次の表に示します。

属性名	機能
value	適用するインターセプタクラスを設定します。

各属性の詳細を次に示します。

(a) value 属性

型
Class[]

説明
適用するインターセプタクラスを設定します。

デフォルト値
なし

2.6 javax.persistence パッケージ

javax.persistence パッケージに含まれるアノテーションの一覧およびアノテーションを指定する際の注意事項について説明します。

なお，マッピング情報はアノテーションの代わりに O/R マッピングファイルで指定することもできます。アノテーションと O/R マッピングファイルとの対応については，「2.6.64　アノテーションと O/R マッピングとの対応」を参照してください。

アノテーションを指定する際の注意事項

- Cosminexus JPA では，DDL 出力機能に関連する属性に対応していません。
- アノテーションで同じカラム名を複数指定する場合は，大文字および小文字をそろえて指定してください。
- フィールド名またはメソッド名をカラム名に割り当てた場合，Cosminexus JPA では文字列を大文字として扱います。対応するアノテーションでカラム名を指定する場合は，大文字にしてください。
- アクセスタイプは，アノテーションを付与する場所によって決まります。ただし，アクセスタイプがフィールドとプロパティで混在した場合は，フィールドの設定が有効になります。
- プロパティ名は，アクセサメソッドの get または set（is）を除いた文字列によって次のように決まります。
  ・最初の二文字が大文字の場合，そのままの文字列になります。
  ・最初の二文字が大文字ではない場合，最初の文字を小文字に変換した文字列になります。
  ・一文字の場合，最初の文字を小文字に変換した文字列になります。

アノテーション一覧

アノテーションの区分	アノテーション名	概要
エンティティのアノテーション	@Entity	クラスがエンティティであることを示します。
テーブル・カラム関連のアノテーション	@Column	永続化フィールドまたは永続化プロパティと，データベース上のカラムとのマッピングを指定します。
	@JoinColumn	エンティティクラス間の関連づけで，結合表のための外部キーカラムまたは外部キーカラムから参照された，結合先テーブルのカラムを指定します。
	@JoinColumns	@JoinColumn を複数同時に記述する場合に使用します。

アノテーションの区分	アノテーション名	概要
	@JoinTable	次のクラスに設定する結合表を指定するアノテーションです。 • ManyToMany リレーションシップを指定する場合の所有者側のクラス • 片方向の OneToMany リレーションシップを持つクラス
	@PrimaryKeyJoinColumn	ほかのテーブルと結合する場合に，外部キーとして使われるカラムを指定します。
	@PrimaryKeyJoinColumns	@PrimaryKeyJoinColumn を複数同時に記述する場合に使用します。
	@SecondaryTable	エンティティクラスにセカンダリテーブルを指定します。
	@SecondaryTables	@SecondaryTable を複数同時に記述する場合に使用します。
	@Table	エンティティクラスにプライマリテーブルを指定します。
	@UniqueConstraint	プライマリテーブルまたはセカンダリテーブルに対して，CREATE 文を生成する場合にユニーク制約を含めることを指定します。 なお，このアノテーションは，Cosminexus JPA プロバイダには対応していません。
ID 関連のアノテーション	@EmbeddedId	埋め込み可能クラスの複合プライマリキーであることを指定します。
	@GeneratedValue	プライマリキーカラムにユニークな値を自動で生成，付与する方法を指定します。
	@Id	エンティティクラスのプライマリキーのプロパティ，またはフィールドであることを指定します。
	@IdClass	エンティティクラスの複数のフィールドまたはプロパティへマップされた複合プライマリキークラスを指定します。
	@SequenceGenerator	プライマリキーを作成するシーケンスジェネレータの設定を指定します。
	@TableGenerator	プライマリキーを作成するジェネレータの設定を指定します。
ロックのアノテーション	@Version	楽観的ロック機能を使用するために用いる version フィールドまたは version プロパティを指定します。
マッピング関連のアノテーション	@Basic	最も単純なデータベースのカラムへのマッピングの型を示します。
	@Embeddable	埋め込みクラスであることを指定します。

アノテーションの区分	アノテーション名	概要
	@Embedded	埋め込み先のエンティティクラス内で，埋め込みクラスのインスタンス値を示す永続化プロパティまたは永続化フィールドであることを指定します。
	@Enumerated	永続化フィールドまたは永続化プロパティを列挙型として指定します。
	@Lob	データベースがサポートしている large オブジェクト型の永続化フィールドまたは永続化プロパティであることを指定します。
	@MapKey	OneToMany リレーションシップ，または ManyToMany リレーションシップで，被所有者側のエンティティクラスが java.util.Map 型で示される場合にマップ内のオブジェクト識別に用いられるマップキーを指定します。
	@OrderBy	エンティティの情報を取得するとき，コレクションが評価した順番を指定します。
	@Temporal	時刻を表す型（java.util.Date および java.util.Calendar）を持つ永続化プロパティまたは永続化フィールドに指定します。
	@Transient	永続化しないエンティティクラス，マップドスーパークラス，または埋め込みクラスのフィールドまたはプロパティであることを指定します。
リレーション関連のアノテーション	@ManyToMany	指定したクラスが ManyToMany リレーションシップであることを示し，所有者側のエンティティクラスから被所有者側のエンティティクラスへの複数の関連を指定します。
	@ManyToOne	指定したクラスが ManyToOne リレーションシップであることを示し，被所有者側のエンティティクラスへの関連を指定します。
	@OneToMany	指定したクラスが OneToMany リレーションシップであることを示し，所有者側のエンティティクラスから被所有者側のエンティティクラスへの複数の関連を指定します。
	@OneToOne	指定したクラスが OneToOne リレーションシップであることを示し，エンティティクラス間の一つの関連を指定します。

アノテーションの区分	アノテーション名	概要
継承・オーバーライド関連のアノテーション	@AssociationOverride	マップドスーパークラスや埋め込みクラスで指定された，ManyToOne リレーションシップまたは OneToOne リレーションシップで使用する設定をオーバーライドします。
	@AssociationOverrides	@AssociationOverride を複数同時に記述する場合に使用します。
	@AttributeOverride	次に示すマッピング情報をオーバーライドします。 • @Basic が指定された（またはデフォルトで適用された）プロパティ，フィールド • @Id で指定されたプロパティ，フィールド
	@AttributeOverrides	@AttributeOverride を複数同時に記述する場合に使用します。
	@DiscriminatorColumn	SINGLE_TABLE 戦略または JOINED 戦略で使用する識別用カラムを指定します。 エンティティクラスの継承で，スーパークラスとなるエンティティクラスに付与します。
	@DiscriminatorValue	SINGLE_TABLE 戦略または JOINED 戦略で使用する識別用カラムの値を指定します。
	@Inheritance	エンティティクラス階層で使われる継承マッピング戦略を指定します。
	@MappedSuperclass	マップドスーパークラスであることを指定します。
クエリ関連のアノテーション	@ColumnResult	SQL のクエリ結果をエンティティクラスとマッピングするためのカラムを指定します。
	@EntityResult	SQL のクエリ結果をマッピングするエンティティクラスを指定します。
	@FieldResult	SQL のクエリ結果をマッピングするフィールドを指定します。
	@NamedNativeQueries	@NamedNativeQuery を複数同時に記述する場合に使用します。
	@NamedNativeQuery	SQL で名前付きクエリを指定します。
	@NamedQueries	@NamedQuery を複数同時に記述する場合に使用します。
	@NamedQuery	JPQL の名前付きクエリを指定します。
	@QueryHint	データベース固有のクエリのヒントを指定します。

アノテーションの区分	アノテーション名	概要
	@SqlResultSetMapping	SQL のクエリの結果セットマッピングを指定します。
	@SqlResultSetMappings	@SqlResultSetMapping を複数同時に記述する場合に使用します。
イベント・コールバック関連のアノテーション※	@EntityListeners	エンティティクラスまたはマップドスーパークラスで使用されるコールバックリスナクラスを指定します。
	@ExcludeDefaultListeners	次に示すクラスに対して，デフォルトリスナを抑止するアノテーションです。 • エンティティクラス • マップドスーパークラス • エンティティクラスまたはマップドスーパークラスのサブクラス
	@ExcludeSuperclassListeners	次に示すクラスに対して，スーパークラスリスナを抑止するアノテーションです。 • エンティティクラス • マップドスーパークラス • エンティティクラスまたはマップドスーパークラスのサブクラス
	@PostLoad	データベースに SELECT 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。
	@PostPersist	データベースに INSERT 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。
	@PostRemove	データベースに DELETE 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。
	@PostUpdate	データベースに UPDATE 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。
	@PrePersist	データベースに INSERT 文を発行する前に呼び出されるコールバックメソッドであることを示すアノテーションです。
	@PreRemove	データベースに DELETE 文を発行する前に呼び出されるコールバックメソッドであることを示すアノテーションです。
	@PreUpdate	データベースに UPDATE 文を発行する前に呼び出されるコールバックメソッドであることを示すアノテーションです。

アノテーションの区分	アノテーション名	概要
EntityManager と EntityManagerFactory のリファレンス関連のアノテーション	@PersistenceContext	コンテナ管理の EntityManager を定義します。
	@PersistenceContexts	@PersistenceContext を複数同時に記述する場合に使用します。
	@PersistenceProperty	コンテナ管理の EntityManager にプロパティを設定します。
	@PersistenceUnit	EntityManagerFactory への永続化ユニットを定義します。
	@PersistenceUnits	@PersistenceUnit を複数同時に記述する場合に使用します。

注※　コールバックメソッ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 機能解説 基本・開発編 (コンテナ共通機能)」の「6.15　コールバックメソッドの指定方法」を参照してください。

2.6.1　@AssociationOverride

(1) 説明

マップドスーパークラスや埋め込みクラスで指定された ManyToOne リレーションシップまたは OneToOne リレーションシップで使用する設定をオーバーライドするアノテーションです。

@AssociationOverride が指定されていない場合，外部キーカラムはオリジナルマッピングと同様にマッピングされます。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@AssociationOverride の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	オーバーライドする関連マッピングを持つフィールドまたはプロパティの名前を指定する属性です。
joinColumns	必須	@ JoinColumn の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明
オーバーライドする関連マッピングを持つフィールドまたはプロパティの名前を指定する属性です。

デフォルト値
なし

(b) joinColumns 属性

型
JoinColumn[]

説明
@ JoinColumn の配列を指定する属性です。
マッピングの型はマップドスーパークラスまたは埋め込みクラスの定義が適用されます。
指定できる値は，@JoinColumn の配列で指定できる範囲です。詳細は，「2.6.24 @JoinColumn」を参照してください。

デフォルト値
なし

2.6.2 @AssociationOverrides

(1) 説明

@AssociationOverride を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@AssociationOverrides の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@AssociationOverride の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
AssociationOverride[]

説明

@AssociationOverride の配列を指定する属性です。
指定できる値は，@AssociationOverride の配列で指定できる範囲です。詳細は，「2.6.1 @AssociationOverride」を参照してください。

デフォルト値
なし

2.6.3 @AttributeOverride

(1) 説明

次に示すマッピング情報をオーバーライドするアノテーションです。

- @Basic で指定した（デフォルトが適用された）プロパティまたはフィールド
- デフォルトが適用されたプロパティまたはフィールド
- @Id で指定されたプロパティまたはフィールド

マップドスーパークラスや埋め込みクラスに定義された @Column の設定をオーバーライドするために，マップドスーパークラスを継承したエンティティクラスや埋め込みクラスのフィールドまたはプロパティに適用します。

@AttributeOverride が指定されていない場合，カラムはオーバーライド前のオリジナルマッピングでマップされます。

継承関係を持たない単体のエンティティクラスに @AttributeOverride を定義した場合，動作はしますが動作の保証はできません。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@AttributeOverride の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	マッピングがオーバーライドされるフィールドまたはプロパティの名前を指定する属性です。
column	必須	オーバーライドする@ Column を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
マッピングがオーバーライドされるフィールドまたはプロパティの名前を指定する

属性です。

デフォルト値
なし

(b) column 属性

型
Column

説明
オーバーライドする@ Column を指定する属性です。
マッピングの型は埋め込み可能クラス，またはマップドスーパークラスの定義が適用されます。
指定できる値は，@Column で指定できる範囲です。詳細は，「2.6.6　@Column」を参照してください。

デフォルト値
なし

2.6.4　@AttributeOverrides

(1) 説明

@AttributeOverride を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@AttributeOverrides の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@AttributeOverride の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
AttributeOverride[]

説明
@AttributeOverride の配列を指定する属性です。
指定できる値は，@AttributeOverride の配列で指定できる範囲です。詳細は，「2.6.3　@AttributeOverride」を参照してください。

デフォルト値
なし

2.6.5 @Basic

(1) 説明

最も単純なデータベースのカラムへのマッピングの型を示すアノテーションです。

次に示す永続化する型のプロパティまたはインスタンス変数に適用できます。

- Java のプリミティブ型
- プリミティブ型のラッパークラス
- java.lang.String
- java.math.BigInteger
- java.math.BigDecimal
- java.util.Date
- java.util.Calendar
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp
- byte[]
- Byte[]
- char[]
- Character[]
- enums
- ユーザが定義するシリアライズ型

適用可能要素は，メソッドとフィールドです。

(2) 属性

@Basic の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
fetch	任意	フェッチ戦略の指定値を指定する属性です。
optional	任意	フィールドまたはプロパティに null 値を使用できるかどうかを指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) fetch 属性

型
FetchType

説明
フェッチ戦略の指定値を指定する属性です。
指定できる値は，FetchType.EAGER または FetchType.LAZY です。
なお，Cosminexus JPA プロバイダでは，fetch 属性は無視され，デフォルトの FetchType.EAGER が常に適用されます。fetch 属性の詳細は，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「6.4.5　データベースとの同期」を参照してください。

デフォルト値
FetchType.EAGER

2.6.6 @Column

(1) 説明

永続化フィールドまたは永続化プロパティと，データベース上のカラムとのマッピングを指定するアノテーションです。

永続化プロパティまたは永続化フィールドに明示的に @Column を指定しない場合でも，@Column が指定されたように永続化フィールドまたは永続化プロパティは扱われます。この場合 @Column の各属性値にはデフォルト値が適用されます。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@Column の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	カラム名を指定する属性です。
unique	任意	プロパティがユニークキーであるかどうかを指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
nullable	任意	データベースのカラムに null 値を指定できるかどうかを指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
insertable	任意	@Column で指定したカラムを SQL の INSERT 文に含むかどうかを指定する属性です。
updatable	任意	@Column で指定したカラムを SQL の UPDATE 文に含むかどうかを指定する属性です。

属性名	任意 / 必須	属性の説明
columnDefinition	任意	CREATE 文を出力するとき，カラムに付加する制約を DDL で記載する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
table	任意	カラムを含むテーブル名を指定する属性です。
length	任意	カラムの長さを指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
precision	任意	カラムの精度を指定する属性です。カラムが数値型の場合に指定します。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
scale	任意	カラムのスケールを指定する属性です。カラムが数値型の場合に指定します。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

カラム名を指定する属性です。
指定できるカラム名は，データベースの仕様に依存します。

デフォルト値

このアノテーションを指定したプロパティ名またはフィールド名

(b) insertable 属性

型

boolean

説明

@Column で指定したカラムを SQL の INSERT 文に含むかどうかを指定する属性です。指定できる値は，true または false です。
それぞれの値の意味は次のとおりです。
true：@Column で指定したカラムを SQL の INSERT 文に含みます。
false：@Column で指定したカラムを SQL の INSERT 文に含みません。

デフォルト値

true

(c) updatable 属性

型

boolean

説明

@Column で指定したカラムを SQL の UPDATE 文に含むかどうかを指定する属性です。指定できる値は，true または false です。
それぞれの値の意味は次のとおりです。
true：@Column で指定したカラムを SQL の UPDATE 文に含みます。
false：@Column で指定したカラムを SQL の UPDATE 文に含みません。

デフォルト値

true

(d) table 属性

型

String

説明

カラムを含むテーブル名を指定します。
指定できるテーブル名は，データベースの仕様に依存します。

デフォルト値

プライマリテーブル名

2.6.7 @ColumnResult

(1) 説明

SQL のクエリ結果をエンティティクラスとマッピングするためのカラムを指定するアノテーションです。

適用可能要素は，@SqlResultSetMapping の columns です。

(2) 属性

@ColumnResult の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	SELECT 節のカラムの名またはカラムの別名を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

SELECT 節のカラム名またはカラムの別名を指定する属性です。
指定できるカラム名は，データベースの仕様に依存します。

デフォルト値
なし

2.6.8 @DiscriminatorColumn

(1) 説明

SINGLE_TABLE 戦略または JOINED 戦略で使用する識別用カラムを指定するアノテーションです。エンティティクラスの継承で，スーパークラスとなるエンティティクラスに付与します。

適用可能要素は，クラスです。

(2) 属性

@DiscriminatorColumn の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	識別用カラム名を指定する属性です。
discriminatorType	任意	識別用カラムの型を指定する属性です。
columnDefinition	任意	CREATE 文を出力するとき，識別用カラムに付加する制約を DDL で記載する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
length	任意	識別用カラムが文字列の場合，その長さを指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
識別用カラム名を指定する属性です。
指定できるカラム名は，データベースの仕様に依存します。
name 属性の値は，@Column の name 属性と同じ値（大文字および小文字を合わせた値）を指定してください。

デフォルト値

"DTYPE"

(b) discriminatorType 属性

型
DiscriminatorType

説明
識別用カラムの型を指定する属性です。
指定できる値は，次のとおりです。
- DiscriminatorType.STRING
- DiscriminatorType.CHAR
- DiscriminatorType.INTEGER

デフォルト値
DiscriminatorType.STRING

2.6.9 @DiscriminatorValue

(1) 説明

SINGLE_TABLE 戦略または JOINED 戦略で使用する識別用カラムの値を指定するアノテーションです。スーパークラスまたはサブクラスに指定できます。

適用可能要素は，クラスです。

なお，次の点に注意して指定してください。

- @DiscriminatorValue の設定は継承されません。@DiscriminatorValue をエンティティクラスごとに設定する必要があります。
- @DiscriminatorValue の設定は，@DiscriminatorColumn の discriminatorType の型と length の長さに一致させる必要があります。
- @DiscriminatorColumn の discriminatorType が INTEGER の場合は，次の点に注意してください。
  - @DiscriminatorValue には，先頭に 0 や空白を含まない整数だけを指定してください。
  - @DiscriminatorValue は省略できません。省略した場合の動作は保証しません。
- @DiscriminatorColumn の discriminatorType が INTEGER 以外の場合は，@DiscriminatorValue を省略できます。このとき，value の値がエンティティのクラス名であるものとして動作します。

(2) 属性

@DiscriminatorValue の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	識別用のカラムに設定する値を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
String

説明
識別用カラムに設定する値を指定する属性です。
指定できる値は，識別用カラムの型に依存します。

デフォルト値
エンティティ名

2.6.10 @Embeddable

(1) 説明

埋め込みクラスであることを示すアノテーションです。

埋め込みクラスとは，エンティティクラス内にフィールドとして埋め込むことができるクラスのことです。

適用可能要素は，クラスです。

(2) 属性

なし

2.6.11 @Embedded

(1) 説明

エンティティクラス内で，埋め込みクラスのインスタンス値を示す永続化プロパティまたは永続化フィールドであることを示すアノテーションです。

埋め込みクラス内で宣言されたカラムマッピングをオーバーライドしたい場合は，@AttributeOverride または @AttributeOverrides を使用します。

適用可能要素は，メソッドとフィールドです。

(2) 属性

なし

2.6.12 @EmbeddedId

(1) 説明

埋め込みクラスの複合プライマリキーであることを示すアノテーションです。

エンティティが所有する埋め込み可能クラスの永続化プロパティまたは永続化フィールドに付与します。

@EmbeddedId を利用する場合，@EmbeddedId を複数指定したり，@EmbeddedId 以外に @Id を指定したりしてはいけません。

@Transient を埋め込みクラスのフィールドに付与した場合，そのフィールドは複合プライマリキーの対象になりません。

適用可能要素は，メソッドとフィールドです。

(2) 属性

なし

2.6.13 @Entity

(1) 説明

クラスがエンティティであることを示すアノテーションです。

エンティティクラスのクラス名は，パッケージ名を含まないクラス名になります。指定するときは，次の内容に注意してください。

- エンティティ名は永続化ユニット内でユニークな名称にしてください。
- JPQL の予約文字を設定してはいけません。設定した場合の動作は保証しません。

適用可能要素は，クラスです。

(2) 属性

@Entity の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	エンティティクラスに対する論理的な名前を指定する属性です。なお，JPQL では，抽象スキーマ名となります。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明
エンティティクラスに対する論理的な名前を指定する属性です。なお，JPQL では，抽象スキーマ名となります。
指定できる値は，JPQL の仕様に依存します。

デフォルト値
@Entity を指定したクラスのクラス名

2.6.14 @EntityListeners

(1) 説明

エンティティクラスまたはマップドスーパークラスで使用されるコールバックリスナクラスを指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@EntityListeners の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	コールバックリスナクラスを指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
Class[]

説明
コールバックリスナクラスを指定する属性です。
指定できる値は，クラスです。

デフォルト値
なし

2.6.15 @EntityResult

(1) 説明

SQL のクエリ結果をマッピングするエンティティクラスを指定するアノテーションです。

適用可能要素は，@SqlResultSetMapping の entities 属性です。

(2) 属性

@EntityResult の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
entityClass	必須	結果のクラスを指定する属性です。
fields	任意	@FieldResult の配列を指定する属性です。
discriminatorColumn	任意	エンティティインスタンスの型を決定する SELECT 節の中で，識別用カラムの名前または別名を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) entityClass 属性

型

Class

説明

結果のクラスを指定する属性です。
指定できる値は，クラス名です。

デフォルト値

なし

(b) fields 属性

型

FieldResult[]

説明

@FieldResult の配列を指定する属性です。
指定できる値は，@FieldResult の配列で指定できる範囲です。詳細は，「2.6.19 @FieldResult」を参照してください。

デフォルト値

空の配列

(c) discriminatorColumn 属性

型

String

説明

エンティティインスタンスの型を決定するための SELECT 節の中で，識別用カラムの名前または別名を指定する属性です。
指定できる値は，テーブルに指定されたカラムの名前または別名です。

デフォルト値
空の文字列

2.6.16 @Enumerated

(1) 説明

永続化フィールドまたは永続化プロパティを列挙型として指定するアノテーションです。

@Basic とともに使用できます。列挙型には ORDINAL（数値型）と STRING（文字列型）が指定できます。

次の場合，列挙型は ORDINAL（数値型）が指定されます。

- value 属性に列挙される型が指定されていない場合
- @Enumerated が指定されていない場合

適用可能要素は，メソッドとフィールドです。

(2) 属性

@Enumerated の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	任意	列挙型をマッピングするのに使われる型を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
EnumType

説明
列挙型をマッピングするのに使われる型を指定する属性です。
指定できる値は，次のどちらかの値です。
- EnumType.ORDINAL：数値型
- EnumType.STRING：文字列型

デフォルト値
EnumType.ORDINAL

2.6.17 @ExcludeDefaultListeners

(1) 説明

次に示すクラスに対して，デフォルトリスナを抑止するアノテーションです。

- エンティティクラス
- マップドスーパークラス
- エンティティクラスまたはマップドスーパークラスのサブクラス

なお，デフォルトリスナを指定できるのは，XML ディスクリプタだけです。

適用可能要素は，クラスです。

(2) 属性

なし

2.6.18 @ExcludeSuperclassListeners

(1) 説明

次に示すクラスに対して，スーパークラスリスナを抑止するアノテーションです。

- エンティティクラス
- マップドスーパークラス
- エンティティクラスまたはマップドスーパークラスのサブクラス

適用可能要素は，クラスです。

(2) 属性

なし

2.6.19 @FieldResult

(1) 説明

SQL のクエリ結果をマッピングするフィールドを指定するアノテーションです。

適用可能要素は，@EntityResult の field 属性です。

(2) 属性

@FieldResult の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	クラスの永続化フィールドまたは永続化プロパティの名前を指定する属性です。
column	必須	SELECT 節のカラム名またはカラムの別名を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
クラスの永続化フィールドまたは永続化プロパティの名前を指定する属性です。

デフォルト値
なし

(b) column 属性

型
String

説明
SELECT 節のカラム名またはカラムの別名を指定する属性です。
指定できるカラム名または別名は，データベースの仕様に依存します。

デフォルト値
なし

2.6.20 @GeneratedValue

(1) 説明

プライマリキーカラムにユニークな値を自動で生成，付与する方法を指定するアノテーションです。@Id を持つエンティティクラスまたはマップドスーパークラスのプライマリキーのフィールドまたはプロパティに適用します。

プライマリキー値の生成方法には，次の 4 種類の方法があります。なお，選択する生成方法に基づいて，あらかじめ基礎テーブルやデータベースシーケンスオブジェクトを用意しておく必要があります。それぞれの生成方法の詳細は，strategy 属性の説明を参照してください。

- GenerationType.AUTO
- GenerationType.IDENTITY
- GenerationType.SEQUENCE
- GenerationType.TABLE

適用可能要素は，メソッドとフィールドです。

(2) 属性

@GeneratedValue の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
strategy	任意	エンティティクラスのプライマリキー値を生成する方法を指定する属性です。
generator	任意	使用する @SequenceGenerator または @TableGenerator で設定される name 属性を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) strategy 属性

型

GenerationType

説明

エンティティクラスのプライマリキー値を生成する方法を指定する属性です。
指定できる値は，次の 4 種類です。

- GenerationType.AUTO
  データベースごとに最も適切な手順を選択して，プライマリキー値を生成します。
  データベースが Oracle または HiRDB の場合は，GenerationType.TABLE と同じ処理をします。
- GenerationType.IDENTITY
  データベースの identity 列を利用して，プライマリキー値を生成します。
  データベースが Oracle の場合は，GenerationType.SEQUENCE と同じ処理をします。
  データベースが HiRDB の場合は，GenerationType.TABLE と同じ処理をします。
- GenerationType.SEQUENCE
  データベースのシーケンスオブジェクトを使用して，プライマリキー値を生成します。
  データベースが HiRDB の場合は，GenerationType.TABLE と同じ処理をします。
- GenerationType.TABLE
  プライマリキー値を保持しておくためのテーブルを使用して，プライマリキー値を生成します。

デフォルト値

GenerationType.AUTO

(b) generator 属性

型

String

説明

使用する @SequenceGenerator または @TableGenerator で設定される name 属性を指定する属性です。

デフォルト値

strategy 属性の値によって，次の名前が仮定されます。

- GenerationType.AUTO **の場合**
  "SEQ_GEN"
- GenerationType.SEQUENCE **の場合**
  "SEQ_GEN_SEQUENCE"
- GenerationType.TABLE **の場合**
  "SEQ_GEN_TABLE"

2.6.21 @Id

(1) 説明

エンティティクラスのプライマリキーのプロパティまたはフィールドであることを示すアノテーションです。

@Id は，エンティティクラスまたはマップドスーパークラスで適用されます。

@Id を指定したフィールドまたはプロパティに対してマップされたデータベース上のカラムは，プライマリテーブルのプライマリキーカラムであると仮定されます。プライマリキーカラムのカラム名を @Column を用いて指定していない場合，プライマリキーカラムのカラム名は @Id を指定したフィールドまたはプロパティの名前になります。

なお，@Id を指定したフィールドに @Version を指定した場合は，@Id が無効になります。

適用可能要素は，メソッドとフィールドです。

(2) 属性

なし

2.6.22 @IdClass

(1) 説明

エンティティクラスの複数のフィールドまたはプロパティへマップされた複合プライマリキークラスを指定するアノテーションです。

このアノテーションは，マップドスーパークラスまたはエンティティクラスに適用されます。

エンティティクラスのプライマリキーのフィールドまたはプロパティと，複合プライマリキークラスのフィールドまたはプロパティの名前と型は一致させる必要があります。このアノテーションで指定した名前および型は，@Id が付いたエンティティのプライマリキーのプロパティまたはフィールドの型および名前に対応させてください。

適用可能要素は，クラスです。

(2) 属性

@IdClass の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	複合プライマリキークラスを指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
Class

説明
複合プライマリキークラスを指定する属性です。
指定できる値は，クラス名です。

デフォルト値
なし

2.6.23 @Inheritance

(1) 説明

エンティティの継承階層で使われる継承マッピング戦略を指定するアノテーションです。

@Inheritance は継承階層の親であるエンティティクラスに指定されます。

Cosminexus JPA プロバイダで使用できる継承マッピング戦略には，次の 2 種類があります。

- SINGLE_TABLE（クラス階層ごとの単体テーブル）
- JOINED（結合サブクラス戦略）

継承マッピング戦略については，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「6.13.2　継承マッピング戦略」を参照してください。

適用可能要素は，クラスです。

(2) 属性

@Inheritance の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
strategy	任意	継承マッピング戦略の種類を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) strategy 属性

型

InheritanceType

説明

エンティティで使用する継承マッピング戦略の種類を指定する属性です。
指定できる値は，次の 2 種類です。

- InheritanceType.SINGLE_TABLE：継承階層にあるすべてのクラスを一つのテーブルにマッピングする戦略です。
- InheritanceType.JOINED：継承階層の最上位（親クラス）は単一の表にマッピングされ，各サブクラスはサブクラス特有のマッピングをする戦略です。

デフォルト値

InheritanceType.SINGLE_TABLE

2.6.24 @JoinColumn

(1) 説明

エンティティクラス間の関連づけをする場合に，結合表のための外部キーカラムまたは外部キーカラムによって参照された，結合先テーブルのカラム名を指定するアノテーションです。必ず結合先テーブルのプライマリキーとなっているカラムを指定してください。

複数の外部キーカラムがある場合は @JoinColumns を使用し，それぞれの関連に対して @JoinColumn を指定してください。なお，複数の @JoinColumn を指定した場合は，name 属性および referencedColumnName 属性をそれぞれのアノテーションで指定してください。

@JoinColumn を明示的に指定しない場合，関連を示す永続化プロパティまたは永続化フィールドに単体の外部キーカラムを指定しているように扱われます。また，@JoinColumn の各属性値にデフォルト値が適用されます。

また，一つのカラムがフィールドに対して行った変更と，カスケード操作の関連づけによる変更が同時に実施されたことによって，整合性が取れなくなる場合があります。このため，name 属性および referencedColumnName 属性に指定したカラムをエンティティのフィールドに定義する場合は，insertable 属性および updatable 属性を false に指定する必要があります。この指定によって，フィールドに対して行った変更だけがデータベースに反映されますが，カスケード操作の関連づけによる変更はデータベースに反

映されません。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@JoinColumn の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	対象テーブルを結合するために使用する外部キーカラム名を指定する属性です。
referencedColumnName	任意	name 属性で指定した外部キーカラムによって参照された結合先テーブルのカラム名を指定する属性です。
unique	任意	プロパティがユニークキーであるかどうかを指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダでは対応していません。
nullable	任意	データベースのカラムに null 値を指定できるかどうかを指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダでは対応していません。
insertable	任意	@JoinColumn で指定したカラムを SQL の INSERT 文に含むかどうか指定する属性です。
updatable	任意	@JoinColumn で指定したカラムを SQL の UPDATE 文に含むかどうか指定する属性です。
columnDefinition	任意	CREATE 文を出力する場合，外部カラムに追加する規約を DDL で記載する属性です。 なお，この属性は，Cosminexus JPA プロバイダでは対応していません。
table	任意	外部キーカラムを含むテーブル名を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

対象テーブルを結合するために使用する外部キーカラム名を指定する属性です。

外部キーカラムは，エンティティのリレーションシップの種別ごとに存在する個所が異なります。エンティティのリレーションシップの種別ごとに外部キーカラムの存在個所を示します。

- **OneToOne リレーションシップまたは** ManyToOne **リレーションシップの場合**
  自エンティティのテーブル内
- **ManyToMany リレーションシップの場合**

@JoinTable の結合表内

なお，指定できる値は，データベースのカラム名の仕様に依存します。

デフォルト値

- **自エンティティ内に単体の外部キーカラムが指定され，name 要素の値に何も指定しない場合**
  < 自エンティティ内の関連プロパティまたはフィールドの名前 >_< 参照されるプライマリキー列の名前 >
- **参照している関連プロパティやフィールドがない場合（例：@JoinTable を使用している場合）**
  < 参照しているエンティティの名前 >_< 参照されるプライマリキー列の名前 >

(b) referencedColumnName 属性

型

String

説明

name 属性で指定した外部キーカラムによって参照された結合先テーブルのカラム名を指定する属性です。
結合先テーブルのカラム名は次の個所に存在します。

- **リレーションシップのアノテーションを使った場合**
  参照されるテーブル内
- **@JoinTable で使った場合**
  所有者側エンティティのエンティティテーブル内

注

結合が逆結合の一部で定義される場合は，被所有者側のエンティティクラスのテーブル内になります。

なお，指定できるカラム名は，データベースの仕様に依存します。

デフォルト値

外部キーによって参照されるテーブルのプライマリキーのカラム名

注

単体の外部キーカラムが指定される場合は，デフォルトを適用します。

(c) insertable 属性

型

boolean

説明

@JoinColumn で指定したカラムを SQL の INSERT 文に含むかどうかを指定する属性です。指定できる値は，true または false です。
それぞれの値の意味は次のとおりです。
true：@JoinColumn で指定したカラムを SQL の INSERT 文に含みます。

false：@JoinColumn で指定したカラムを SQL の INSERT 文に含みません。

デフォルト値
true

(d) updatable 属性

型
boolean

説明
@JoinColumn で指定したカラムを SQL の UPDATE 文に含むかどうかを指定する属性です。指定できる値は，true または false です。
それぞれの値の意味は次のとおりです。
true：@JoinColumn で指定したカラムを SQL の UPDATE 文に含みます。
false：@JoinColumn で指定したカラムを SQL の UPDATE 文に含みません。

デフォルト値
true

(e) table 属性

型
String

説明
外部キーカラムを含むテーブル名を指定する属性です。
指定できるテーブル名は，データベースの仕様に依存します。

デフォルト値
プライマリテーブル名

2.6.25 @JoinColumns

(1) 説明

同じ関連を示す @JoinColumn を複数同時に記述する場合に指定するアノテーションです。また，@JoinColumns は複合外部キーのマッピングを定義します。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@JoinColumns の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@JoinColumn の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
JoinColumn[]

説明
@JoinColumn の配列を指定する属性です。
指定できる値は，@JoinColumn の配列で指定できる範囲です。

デフォルト値
なし

2.6.26 @JoinTable

(1) 説明

次のクラスに設定する結合表を指定するアノテーションです。

- ManyToMany リレーションシップを指定する場合の所有者側のクラス
- 片方向の OneToMany リレーションシップを持つクラス

name 属性が指定されない場合，結合表の名前は次の名称になります。

< 所有者側テーブル名 >_< 被所有者側テーブル名 >

適用可能要素は，メソッドとフィールドです。

(2) 属性

@JoinTable の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	結合表のテーブル名を指定する属性です。
catalog	任意	結合表のテーブルのカタログ名を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
schema	任意	結合表のテーブルのスキーマ名を指定する属性です。
joinColumns	任意	所有者側エンティティのプライマリテーブルを参照する，結合表の外部キーカラムを指定する属性です。
inverseJoinColumns	任意	被所有者側エンティティのプライマリテーブルを参照する，結合表の外部キーカラムを指定する属性です。
uniqueConstraints	任意	テーブルのユニーク規制を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
結合表のテーブル名を指定する属性です。
指定できるテーブル名は，データベースの仕様に依存します。

デフォルト値
< 所有者側テーブル名 >_< 被所有者側テーブル名 >

(b) schema 属性

型
String

説明
テーブルのスキーマ名を指定する属性です。
指定できる値は，データベースのスキーマ名の仕様に依存します。

デフォルト値
使用するデータベースのデフォルトのスキーマ

(c) joinColumns 属性

型
JoinColumn[]

説明
所有者側エンティティのプライマリテーブルを参照する，結合表の外部キーカラムを指定する属性です。@JoinColumn の配列を指定します。@JoinColumn の name 属性には結合表の外部キーカラム名，referencedColumnName 属性には所有者側の参照カラム名をそれぞれ指定します。
指定できるカラム名は，データベースの仕様に依存します。

デフォルト値
@JoinColumn の外部キー

(d) inverseJoinColumns 属性

型
JoinColumn[]

説明
被所有者側エンティティのプライマリテーブルを参照する，結合表の外部キーカラムを指定する属性です。@JoinColumn の配列を指定します。@JoinColumn の

name 属性には結合表の外部キーカラム名，referencedColumnName 属性には外部キーカラムで参照された結合先テーブルのカラムをそれぞれ指定します。
指定できる値は，データベースのカラム名の仕様に依存します。

デフォルト値
@JoinColumn の外部キーカラム

2.6.27 @Lob

(1) 説明

データベースがサポートしている large オブジェクト型の永続化フィールドまたは永続化プロパティであることを指定するアノテーションです。@Basic とともに使用できます。

@Lob にはバイナリ型（Blob）とキャラクタ型（Clob）があります。@Lob の型は，永続化フィールドまたは永続化プロパティの型によって決まります。文字列およびキャラクタ型の場合は Clob になり，そのほかの場合は Blob になります。

適用可能要素は，メソッドとフィールドです。

(2) 属性

なし

2.6.28 @ManyToMany

(1) 説明

ManyToMany リレーションシップの関係を持つ所有者側のエンティティクラスから被所有者側のエンティティクラスへの複数の関連を指定するアノテーションです。

ManyToMany リレーションシップは，双方向，単方向に関係なく，所有者側と被所有者側を持ちます。関係が双方向の場合，結合表の指定はどちらの側でも指定できます。

なお，Generics を使用して Collection 要素型が指定されている場合，被所有者側のエンティティクラスを指定する必要はありません。そのほかの場合は，必ず指定してください。

また，@ManyToMany を指定した場合は，次に示すアノテーションの設定に注意してください。

- @OneToMany のための同じアノテーションの属性は，@ManyToMany と同じ属性を持ちます。
- 所有者側と被所有者側のクラスで @ManyToMany を定義したプロパティまたはフィールドが同じ名前の場合は，@JoinTable のデフォルト設定（joinColumns 属性，

inverseJoinColumns 属性の指定がない状態）を使用しないでください。
- 双方向関係の場合には，所有者側の情報を基に結合表の値が更新されます。mappedBy 属性を指定したエンティティクラスでマッピング情報を変更しても，その情報は結合表には反映されません。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@ManyToMany の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
targetEntity	任意	被所有者側のエンティティクラスを指定する属性です。
cascade	任意	カスケード対象となるオペレーションを指定する属性です。
fetch	任意	フェッチ戦略の指定値を指定する属性です。
mappedBy	任意	被所有者側のエンティティクラスの要素に付与して，所有者側のエンティティクラスで関係を保持しているフィールドまたはプロパティの名前を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) targetEntity 属性

型

Class

説明

被所有者側のエンティティクラスを指定する属性です。
Generics を使って定義されているコレクションプロパティの場合は，任意で指定します。それ以外の場合は必ず指定してください。

デフォルト値

コレクションのパラメタ化された型

注 Generics を使って定義されているときにだけ設定されます。

(b) cascade 属性

型

CascadeType[]

説明

カスケード対象となるオペレーションを指定する属性です。
指定できる値を次に示します。
- CascadeType.ALL：所有者側のエンティティクラスの persist，remove，merge，refresh の操作が関連先にカスケードされます。
- CascadeType.MERGE：所有者側のエンティティクラスの merge 操作が関連先に

カスケードされます。

- CascadeType.PERSIST：所有者側のエンティティクラスの persist 操作が関連先にカスケードされます。
- CascadeType.REFRESH：所有者側のエンティティクラスの refresh 操作が関連先にカスケードされます。
- CascadeType.REMOVE：所有者側のエンティティクラスの remove 操作が関連先にカスケードされます。

デフォルト値
カスケード対象なし

(c) fetch 属性

型
FetchType

説明
データベースからのデータのフェッチ戦略を定義する属性です。フェッチ戦略については，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「6.4.5　データベースとの同期」を参照してください。
指定できる値は，次の 2 種類です。

- EAGER 戦略：データが EAGER にフェッチされなければならない要求
- LAZY 戦略：データが最初にアクセスされるときに，LAZY にフェッチされる要求

デフォルト値
FetchType.LAZY

(d) mappedBy 属性

型
String

説明
被所有者側のエンティティクラスの要素に付与し，所有者側のエンティティクラスで関係を保持しているフィールドまたはプロパティの名前を指定する属性です。
この属性を指定した場合，関係は双方向になります。双方向関係の場合には所有者側の情報を基に結合表の値は更新されます。被所有者側のエンティティクラス（mappedBy 属性を指定したエンティティクラス）でマッピング情報を変更しても，その情報は結合表には反映されません。

デフォルト値
なし

2.6.29 @ManyToOne

(1) 説明

@ManayToOne が指定されたクラスが ManyToOne リレーションシップであることを示し，所有者側のエンティティクラスから被所有者側のエンティティクラスへの関連を指定するアノテーションです。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@ManyToOne の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
targetEntity	任意	被所有者側のエンティティクラスを指定する属性です。
cascade	任意	カスケード対象となるオペレーションを指定する属性です。
fetch	任意	フェッチ戦略の指定値を指定する属性です。
optional	任意	すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を設定できるかどうかを指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) targetEntity 属性

型
Class

説明
被所有者側のエンティティクラスを指定する属性です。

デフォルト値
アノテーションが付与されているフィールドやプロパティの型

(b) cascade 属性

型
CascadeType[]

説明
カスケード対象となるオペレーションを指定する属性です。
指定できる値を次に示します。

- CascadeType.ALL：所有者側のエンティティクラスの persist，remove，merge，および refresh の操作が関連先にカスケードされます。
- CascadeType.MERGE：所有者側のエンティティクラスの merge 操作が関連先にカスケードされます。

- CascadeType.PERSIST：所有者側のエンティティクラスの persist 操作が関連先にカスケードされます。
- CascadeType.REFRESH：所有者側のエンティティクラスの refresh 操作が関連先にカスケードされます。
- CascadeType.REMOVE：所有者側のエンティティクラスの remove 操作が関連先にカスケードされます。

デフォルト値
カスケード対象なし

(c) fetch 属性

型
FetchType

説明
データベースからのデータのフェッチ戦略を定義する属性です。フェッチ戦略については，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「6.4.5　データベースとの同期」を参照してください。
指定できる値は，次の 2 種類です。

- EAGER 戦略：データが EAGER にフェッチされなければならない要求
- LAZY 戦略：データが最初にアクセスされるときに，LAZY にフェッチされる要求

デフォルト値
FetchType.EAGER

(d) optional 属性

型
boolean

説明
すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を指定できるかどうかを指定する属性です。指定できる値は，次のとおりです。

- true：すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を設定できます。
- false：すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を指定できません。

デフォルト値
true

2.6.30 @MapKey

(1) 説明

OneToMany リレーションシップまたは ManyToMany リレーションシップで被所有者側のエンティティクラスが java.util.Map 型で示される場合に，マップ内のオブジェクト識別に用いられるマップキーを指定するアノテーションです。

name 属性が指定されない場合，関連づけられたエンティティのプライマリキーはマップキーとして使われます。

プライマリキーが複合プライマリキーである場合に @IdClass としてマップされるときは，マップキーに複合プライマリキーを使います。

プライマリキー以外の永続化フィールドまたは永続化プロパティがマップキーとして使われる場合は，そのマップキーと関連したユニークキー制約を持つことができます。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@MapKey の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	マップキーとして使われる被所有者側のエンティティクラスの永続化フィールドまたは永続化プロパティの名前を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
マップキーとして使われる被所有者側のエンティティクラスの永続化フィールドまたは永続化プロパティの名前を指定する属性です。

デフォルト値
被所有者側のエンティティクラスのプライマリキーフィールドまたはプロパティの名前

2.6.31 @MappedSuperclass

(1) 説明

マップドスーパークラスであることを指定するアノテーションです。

マップドスーパークラスは，継承するためのクラスであるため，このクラスに対応するテーブルを持ちません。マップドスーパークラスは，サブクラスへのマッピングと関連マッピング情報が継承されることを除いて，エンティティと同じ方法でテーブルにマップされます。

@AttributeOverride を使うことでマッピング情報をサブクラスでオーバーライドできます。

適用可能要素は，クラスです。

(2) 属性

なし

2.6.32 @NamedNativeQueries

(1) 説明

@NamedNativeQuery を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@NamedNativeQueries の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@NamedNativeQuery の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型

NamedNativeQuery[]

説明

@NamedNativeQuery の配列を指定する属性です。
指定できる値は，@NamedNativeQuery 配列で指定できる範囲です。詳細は，「2.6.33 @NamedNativeQuery」を参照してください。

デフォルト値

なし

2.6.33 @NamedNativeQuery

(1) 説明

SQL で名前付きクエリを指定するアノテーションです。エンティティクラスとマップドスーパークラスに適用できます。

適用可能要素は，クラスです。

(2) 属性

@NamedNativeQuery の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	名前付きクエリの名前を指定する属性です。
query	必須	SQL の文字列を指定する属性です。
hints	任意	@QueryHint の配列を指定する属性です。
resultClass	任意	SQL の結果を反映するクラスを指定する属性です。
resultSetMapping	任意	@SqlResultSetMapping の name 属性で指定した名前を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
名前付きクエリの名前を指定する属性です。
指定できる値は，文字列です。

デフォルト値
なし

(b) query 属性

型
String

説明
SQL の文字列を指定する属性です。
指定できる SQL は，使用するデータベースの仕様に依存します。

デフォルト値
なし

(c) hints 属性

型
QueryHint[]

説明
@QueryHint の配列を指定する属性です。
指定できる値は，@QueryHint の配列で指定できる範囲です。詳細は，「2.6.53 @QueryHint」を参照してください。

デフォルト値
空の配列

(d) resultClass 属性

型
Class

説明
SQL の結果を反映するクラスを指定する属性です。
resultClass 属性は，クエリの実行結果をマッピングしたいクラスがあるときに指定します。resultClass 属性と resultSetMapping 属性は同時に指定しないでください。
指定できる値は，クラス名です。

デフォルト値
void.class

(e) resultSetMapping 属性

型
String

説明
結果セットを定義した @SqlResultSetMapping の name 属性で指定した名前を指定する属性です。
任意の結果セットに SQL の結果をマッピングしたいときに指定します。
resultClass 属性と resultSetMapping 属性は同時に指定しないでください。
指定できる範囲は，@SqlResultSetMapping の name 属性で指定できる範囲です。
詳細は，「2.6.57(2)(a)　name 属性」を参照してください。

デフォルト値
空の文字列

2.6.34 @NamedQueries

(1) 説明

@NamedQuery を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@NamedQueries の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@NamedQuery の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型

NamedQuery[]

説明

@NamedQuery の配列を指定する属性です。
指定できる値は，@NamedQuery の配列で指定できる範囲です。詳細は，「2.6.35 @NamedQuery」を参照してください。

デフォルト値

なし

2.6.35 @NamedQuery

(1) 説明

JPQL の名前付きクエリを指定するアノテーションです。エンティティクラスとマップドスーパークラスに適用できます。

適用可能要素は，クラスです。

(2) 属性

@NamedQuery の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	名前付きクエリ名を指定する属性です。
query	必須	JPQL のクエリ文字列を指定する属性です。

属性名	任意 / 必須	属性の説明
hints	任意	@QueryHint の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
名前付きクエリ名を指定する属性です。
指定できる値は，文字列です。

デフォルト値
なし

(b) query 属性

型
String

説明
JPQL のクエリ文字列を指定する属性です。
指定できる値は，JPQL の仕様に依存します。

デフォルト値
なし

(c) hints 属性

型
QueryHint[]

説明
@QueryHint の配列を指定する属性です。
指定できる値は，@QueryHint の配列で指定できる範囲です。詳細は，「2.6.53 @QueryHint」を参照してください。

デフォルト値
空の配列

2.6.36 @OneToMany

(1) 説明

OneToMany リレーションシップの関係を持つ所有者側のエンティティクラスから被所

有者側のエンティティクラスへの複数の関連を指定するアノテーションです。

@OneToMany のための同じアノテーションの属性は，@ManyToMany と同じ属性を持ちます。

なお，Generics を使用して Collection 要素型が指定されている場合，被所有者側のエンティティクラスを指定する必要はありません。そのほかの場合は，必ず指定してください。

また，双方向の関係にする場合は，非所有者側に必ず mappedBy 属性を指定してください。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@OneToMany の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
targetEntity	任意	被所有者側のエンティティクラスを指定する属性です。
cascade	任意	カスケード対象となるオペレーションを指定する属性です。
fetch	任意	フェッチ戦略の指定値を指定する属性です。
mappedBy	任意	被所有者側のエンティティクラスの要素に付与し，所有者側のエンティティクラスで関係を保持しているフィールドまたはプロパティの名前を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) targetEntity 属性

型

Class

説明

被所有者側のエンティティクラスを指定する属性です。
Generics を使って定義されているコレクションプロパティの場合は，任意で指定します。それ以外の場合は必ず指定してください。

デフォルト値

コレクションのパラメタ化された型

注 Generics を使って定義されているときにだけ設定されます。

(b) cascade 属性

型

CascadeType[]

説明

カスケード対象となるオペレーションを指定する属性です。
指定できる値を次に示します。

- CascadeType.ALL：所有者側のエンティティクラスの persist，remove，merge，および refresh の操作が関連先にカスケードされます。
- CascadeType.MERGE：所有者側のエンティティクラスの merge 操作が関連先にカスケードされます。
- CascadeType.PERSIST：所有者側のエンティティクラスの persist 操作が関連先にカスケードされます。
- CascadeType.REFRESH：所有者側のエンティティクラスの refresh 操作が関連先にカスケードされます。
- CascadeType.REMOVE：所有者側のエンティティクラスの remove 操作が関連先にカスケードされます。

デフォルト値
カスケード対象なし

(c) fetch 属性

型
FetchType

説明
データベースからのデータのフェッチ戦略を定義する属性です。フェッチ戦略については，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「6.4.5　データベースとの同期」を参照してください。
指定できる値は，次の 2 種類です。

- EAGER 戦略：データが EAGER にフェッチされなければならない要求
- LAZY 戦略：データが最初にアクセスされるときに，LAZY にフェッチされる要求

デフォルト値
FetchType.LAZY

(d) mappedBy 属性

型
String

説明
被所有者側のエンティティクラスの要素に付与し，所有者側のエンティティクラスで関係を保持しているフィールドまたはプロパティの名前を指定する属性です。
この属性を指定した場合，関係は双方向になります。

デフォルト値
なし

2.6.37 @OneToOne

(1) 説明

指定されたクラスが OneToOne リレーションシップであることを示し，エンティティクラス間の一つの関連を指定するアノテーションです。

双方向の関係にする場合は，非所有者側に必ず mappedBy 属性を指定してください。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@OneToOne の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
targetEntity	任意	被所有者側のエンティティクラスを指定する属性です。
cascade	任意	カスケード対象となるオペレーションを指定する属性です。
fetch	任意	フェッチ戦略の指定値を指定する属性です。
optional	任意	すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を設定できるかどうかを指定する属性です。
mappedBy	任意	被所有者側のエンティティクラスの要素に付与し，所有者側のエンティティクラスで関係を保持しているフィールド名を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) targetEntity 属性

型
Class

説明
被所有者側のエンティティクラスを指定する属性です。

デフォルト値
アノテーションが付与されているフィールドやプロパティの型

(b) cascade 属性

型
CascadeType[]

説明
カスケード対象となるオペレーションを指定する属性です。
指定できる値を次に示します。
- CascadeType.ALL：所有者側のエンティティクラスの persist，remove，merge，

refresh の操作が関連先にカスケードされます。
- CascadeType.MERGE：所有者側のエンティティクラスの merge 操作が関連先にカスケードされます。
- CascadeType.PERSIST：所有者側のエンティティクラスの persist 操作が関連先にカスケードされます。
- CascadeType.REFRESH：所有者側のエンティティクラスの refresh 操作が関連先にカスケードされます。
- CascadeType.REMOVE：所有者側のエンティティクラスの remove 操作が関連先にカスケードされます。

デフォルト値
カスケード対象なし

(c) fetch 属性

型
FetchType

説明
データベースからのデータのフェッチ戦略を定義する属性です。フェッチ戦略については，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「6.4.5　データベースとの同期」を参照してください。
指定できる値は，次の 2 種類です。
- EAGER 戦略：データが EAGER にフェッチされなければならない要求
- LAZY 戦略：データが最初にアクセスされるときに，LAZY にフェッチされる要求

デフォルト値
FetchType.EAGER

(d) optional 属性

型
boolean

説明
すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を指定できるかどうかを指定する属性です。指定できる値は，次のとおりです。
- true：すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を設定できます。
- false：すべての非プリミティブ型のフィールドおよびプロパティの値に null 値を指定できません。

デフォルト値
true

(e) mappedBy 属性

型
String

説明
被所有者側のエンティティクラスの要素に付与し，所有者側のエンティティクラスで関係を保持しているフィールド名を指定します。指定した場合，関係は双方向になります。

デフォルト値
なし

2.6.38 @OrderBy

(1) 説明

エンティティの情報を取得するとき，コレクションに保持される順番を指定するアノテーションです。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@OrderBy の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	任意	プライマリキー以外のフィールドまたはプロパティを基にした順番でエンティティを取得したい場合に指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
String

説明
プライマリキー以外のフィールドまたはプロパティを基にした順番でエンティティクラスを取得するときに指定する属性です。コンマ区切りで，順番を指定したいフィールドまたはプロパティを指定します。
取得する順番は，フィールドまたはプロパティのあとに指定します。指定できる値は次のとおりです。指定しなかった場合は，昇順になります。
- ASC：昇順
- DESC：降順

value 属性内で指定されるフィールドまたはプロパティには，比較演算できる値が格

納されているカラムを指定します。

デフォルト値
エンティティクラスのプライマリキーによる昇順

2.6.39 @PersistenceContext

(1) 説明

コンテナ管理の EntityManager のリファレンスを定義するアノテーションです。ルックアップをするクラスに付加します。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@PersistenceContext の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	EntityManager のルックアップ名を指定する属性です。
unitName	任意	persistence.xml ファイルで定義されている永続化ユニットの名前を指定する属性です。
type	任意	永続化コンテキストのライフサイクルの種類を指定する属性です。
properties	任意	@PersistenceProperty で指定するベンダ依存のプロパティを指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
EntityManager のルックアップ名を指定する属性です。
DI を使用する場合は指定不要です。

デフォルト値
空の文字列

(b) unitName 属性

型
String

説明

persistence.xml ファイルで定義された永続化ユニットの名前を指定する属性です。
unitName 属性を指定した場合，JNDI 名前空間でアクセスできる
EntityManagerFactory が使用する永続化ユニットを同じ名前にしてください。

デフォルト値
空の文字列

(c) type 属性

型
PersistenceContextType

説明
永続化コンテキストのライフサイクルの種類を指定する属性です。
指定できる値は，次の 2 種類です。
- TRANSACTION：トランザクションスコープの永続化コンテキスト
- EXTENDED：拡張永続化コンテキスト

デフォルト値
TRANSACTION

(d) properties 属性

型
PersistenceProperty[]

説明
@PersistenceProperty で指定する JPA プロバイダのベンダに依存するプロパティを指定する属性です。
指定できる値は，@PersistenceProperty の配列で指定できる範囲です。詳細は，「2.6.41　@PersistenceProperty」を参照してください。
properties 属性を指定した場合，認識できないプロパティは無視します。

デフォルト値
空の配列

2.6.40　@PersistenceContexts

(1) 説明

@PersistenceContext を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@PersistenceContexts の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@PersistenceContext の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型

PersistenceContext[]

説明

@PersistenceContext の配列を指定する属性です。
指定できる値は，@PersistenceContext の配列で指定できる範囲です。詳細は，「2.6.39　@PersistenceContext」を参照してください。

デフォルト値

なし

2.6.41　@PersistenceProperty

(1) 説明

コンテナ管理の EntityManager にプロパティを設定するアノテーションです。

現状，使用できるプロパティはありません。

適用可能要素は，@PersistenceContext の properties 属性です。

(2) 属性

@PersistenceProperty の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	プロパティの名前を指定する属性です。
value	必須	プロパティの値を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

プロパティの名前を指定する属性です。

デフォルト値
なし

(b) value 属性

型
String

説明
プロパティの値を指定する属性です。
指定できる値は，name 属性に指定したプロパティの仕様に依存します。

デフォルト値
なし

2.6.42 @PersistenceUnit

(1) 説明

EntityManagerFactory のリファレンスを定義するアノテーションです。ルックアップするクラスに付加します。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@PersistenceUnit の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	EntityManagerFactory のルックアップ名を指定する属性です。
unitName	任意	persistence.xml ファイルで定義されている永続化ユニットの名前を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
EntityManagerFactory のルックアップ名を指定する属性です。JNDI 名前空間に登録する EntityManagerFactory の名前を指定します。
指定できる値は，文字列です。
DI を使用する場合は指定不要です。

デフォルト値
空の文字列

(b) unitName 属性

型
String

説明
persistence.xml ファイルで定義された永続化ユニットの名前を指定する属性です。
unitName 属性を指定した場合，JNDI 名前空間でアクセスできる EntityManagerFactory が使用する永続化ユニットを同じ名前にしてください。

デフォルト値
空の文字列

2.6.43 @PersistenceUnits

(1) 説明

@PersistenceUnit を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@PersistenceUnits の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@PersistenceUnit の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
PersistenceUnit[]

説明
@PersistenceUnit の配列を指定する属性です。
指定できる値は，@PersistenceUnit の配列で指定できる範囲です。詳細は，「2.6.42 @PersistenceUnit」を参照してください。

デフォルト値
なし

2.6.44 @PostLoad

(1) 説明

キャッシュからエンティティを読み込んだあと，またはデータベースに SELECT 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。エンティティクラス，マップドスーパークラス，またはエンティティリスナクラスのメソッドに適用されます。

適用可能要素は，メソッドです。

(2) 属性

なし

2.6.45 @PostPersist

(1) 説明

データベースに INSERT 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。エンティティクラス，マップドスーパークラス，またはエンティティリスナクラスのメソッドに適用されます。

適用可能要素は，メソッドです。

(2) 属性

なし

2.6.46 @PostRemove

(1) 説明

データベースに DELETE 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。エンティティクラス，マップドスーパークラス，またはエンティティリスナクラスのメソッドに適用されます。

適用可能要素は，メソッドです。

(2) 属性

なし

2.6.47 @PostUpdate

(1) 説明

データベースに UPDATE 文を発行したあとに呼び出されるコールバックメソッドであることを示すアノテーションです。エンティティクラス，マップドスーパークラス，またはエンティティリスナクラスのメソッドに適用されます。

適用可能要素は，メソッドです。

(2) 属性

なし

2.6.48 @PrePersist

(1) 説明

データベースに INSERT 文を発行する前に呼び出されるコールバックメソッドであることを示すアノテーションです。エンティティクラス，マップドスーパークラス，またはエンティティリスナクラスのメソッドに適用されます。

適用可能要素は，メソッドです。

(2) 属性

なし

2.6.49 @PreRemove

(1) 説明

データベースに DELETE 文を発行する前に呼び出されるコールバックメソッドであることを示すアノテーションです。エンティティクラス，マップドスーパークラス，またはエンティティリスナクラスのメソッドに適用されます。

適用可能要素は，メソッドです。

(2) 属性

なし

2.6.50 @PreUpdate

(1) 説明

データベースに UPDATE 文を発行する前に呼び出されるコールバックメソッドであるこ

とを示すアノテーションです。エンティティクラス，マップドスーパークラス，またはエンティティリスナクラスのメソッドに適用されます。

適用可能要素は，メソッドです。

(2) 属性

なし

2.6.51 @PrimaryKeyJoinColumn

(1) 説明

ほかのテーブルと結合する場合に外部キーとして使われるカラムを指定するアノテーションです。次の場合に使用します。

- 継承マッピング戦略の JOINED 戦略で，エンティティのスーパークラスのプライマリキーとサブクラスのプライマリキーが異なる名前の場合
- @SecondaryTable で，プライマリテーブルとセカンダリテーブルを結合する場合※
- OneToOne リレーションシップで，被所有者側のエンティティクラスのプライマリキーが外部キーとして使用されている場合

注※

この場合は，@SecondaryTable 内で使用します。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@PrimaryKeyJoinColumn の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	対象テーブルを結合するためのカラム名を指定する属性です。
referencedColumnName	任意	name 属性で指定したカラムによって参照される結合先テーブルのプライマリキーのカラム名を指定する属性です。
columnDefinition	任意	CREATE 文を出力するときにカラムに付加する制約を DDL で記載する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

対象テーブルを結合するためのカラム名を指定する属性です。
指定できるカラム名は，データベースの仕様に依存します。

デフォルト値

- JOINED **戦略を使用した場合**
  スーパークラスのプライマリテーブルのプライマリキーのカラム名
- @SecondaryTable **を使用した場合**
  プライマリテーブルのプライマリキーのカラム名
- OneToOne **リレーションシップを使用した場合**
  対象となるエンティティのテーブルのプライマリキーのカラム名

（b）referencedColumnName 属性

型

String

説明

name 属性で指定したカラムによって参照される結合先テーブルのプライマリキーのカラム名を指定する属性です。@Column の name 属性の文字列と同じ値を指定してください。なお，指定する文字列は大文字，小文字もそろえてください。
指定できるカラム名は，データベースの仕様に依存します。
OneToOne リレーションシップでカラムの指定をプライマリキーにしなくてもユニークキー制約があれば動作してしまいますが，動作の保証はしません。

デフォルト値

- JOINED **戦略を使用した場合**
  スーパークラスのプライマリテーブルのプライマリキーのカラム名
- @SecondaryTable **を使用した場合**
  プライマリテーブルのプライマリキーのカラム名
- OneToOne **リレーションシップを使用した場合**
  対象となるエンティティのテーブルのプライマリキーのカラム名

2.6.52 @PrimaryKeyJoinColumns

（1）説明

@PrimaryKeyJoinColumn を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラス，メソッド，およびフィールドです。

（2）属性

@PrimaryKeyJoinColumns の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@PrimaryKeyJoinColumn の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
PrimaryKeyJoinColumn[]

説明
@PrimaryKeyJoinColumn の配列を指定する属性です。
指定できる値は，@PrimaryKeyJoinColumn で指定できる範囲です。詳細は，「2.6.51　@PrimaryKeyJoinColumn」を参照してください。

デフォルト値
なし

2.6.53　@QueryHint

(1) 説明

データベース固有のクエリのヒントを指定するアノテーションです。

悲観的ロックやエンティティのキャッシュ機能の設定ができます。

適用可能要素は，@NamedQuery，または @NamedNativeQuery の hints 要素です。

(2) 属性

@QueryHint の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	ヒントの名前を指定する属性です。
value	必須	ヒントの値を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
使用するヒントの名前を指定する属性です。指定できる値は次のとおりです。

cosminexus.jpa.pessimistic-lock
悲観的ロックを使用するかどうかを指定するヒントの名前です。

デフォルト値
なし

(b) value 属性

型
String

説明
ヒントの値を指定する属性です。name 属性で指定したヒントの名前に合わせて，次の値を指定します。

name 属性で cosminexus.jpa.pessimistic-lock を指定した場合の指定値

- NoLock：悲観的ロックを使用しない場合に指定します。
- Lock：悲観的ロックを使用する場合に指定します。対象となるテーブルがすでにロックされている場合は，解放されるまで待ちます。このとき発行される SQL を使用するデータベースごとに示します。
  Oracle の場合：SELECT.... FOR UPDATE
  HiRDB の場合：SELECT....WITH EXCLUSIVE LOCK
- LockNoWait：悲観的ロックを使用する場合に指定します。対象となるテーブルがすでにロックされている場合は，例外が発生します。このとき発行される SQL を使用するデータベースごとに示します。
  Oracle の場合：SELECT.... FOR UPDATE NO WAIT
  HiRDB の場合：SELECT....WITH EXCLUSIVE LOCK NO WAIT

デフォルト値

name 属性で cosminexus.jpa.pessimistic-lock を指定した場合
NoLock

2.6.54 @SecondaryTable

(1) 説明

エンティティクラスにセカンダリテーブルを指定するアノテーションです。

エンティティクラスがデータベース上の複数のテーブルにわたってマッピングされる場合に指定します。

@SecondaryTable をエンティティクラス内で指定しない場合，エンティティクラスのすべての永続化プロパティまたは永続化フィールドは，プライマリテーブルで指定されたテーブルとマッピングします。

適用可能要素は，クラスです。

(2) 属性

@SecondaryTable の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	セカンダリテーブル名を指定する属性です。
catalog	任意	セカンダリテーブルのカタログ名を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
schema	任意	セカンダリテーブルのスキーマ名を指定する属性です。
pkJoinColumns	任意	セカンダリテーブルがプライマリテーブルと結合するために使用する外部キーカラムを指定する属性です。
uniqueConstraints	任意	テーブルでのユニークキー制約を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

セカンダリテーブル名を指定する属性です。
指定できるテーブル名は，データベースの仕様に依存します。

デフォルト値

なし

(b) schema 属性

型

String

説明

セカンダリテーブルのスキーマ名を指定する属性です。
指定できるスキーマ名は，データベースの仕様に依存します。

デフォルト値

使用するデータベースのデフォルトのスキーマ名

(c) pkJoinColumns 属性

型

PrimaryKeyJoinColumn[]

説明

セカンダリテーブルの外部キーカラムを指定する属性です。
@PrimaryKeyJoinColumn の配列で指定します。
この要素が指定されない場合，セカンダリテーブルの外部キーカラムはプライマリテーブルのプライマリキーカラムと同じ名前と型を持ちます。このため，セカンダリテーブルはプライマリテーブルのプライマリキーカラムを参照します。

デフォルト値
指定できる値は，@PrimaryKeyJoinColumn の配列で指定できる範囲です。詳細は，「2.6.51　@PrimaryKeyJoinColumn」を参照してください。

2.6.55　@SecondaryTables

(1) 説明

@SecondaryTable を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@SecondaryTables の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@SecondaryTable の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
SecondaryTable[]

説明
@SecondaryTable の配列を指定する属性です。
指定できる値は，@SecondaryTable の配列で指定できる範囲です。詳細は，「2.6.54　@SecondaryTable」を参照してください。

デフォルト値
なし

2.6.56　@SequenceGenerator

(1) 説明

プライマリキーを作成するシーケンスジェネレータの設定を指定するアノテーションで

す。@SequenceGenerator を使用する場合，次の設定が必要です。

- @GeneratedValue の strategy 属性に GenerationType.SEQUENCE を指定します。
- @GeneratedValue の generator 属性で指定された名前を @SequenceGenerator の name 属性に設定します。

シーケンスジェネレータは，エンティティクラス，またはプライマリキーのフィールドもしくはプロパティで指定されます。シーケンスジェネレータ名のスコープは永続化ユニットで有効です。

シーケンスオブジェクトを作成する場合，順序の番号間の増分間隔（INCREMENT BY）と生成する順序番号の初期値（START WITH）には，正の整数を指定します。初期値（START WITH）に 1 を指定した場合，プライマリキーは 1 から生成されます。負の値を指定した場合の動作は保証しません。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@SequenceGenerator の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	@GeneratedValue アノテーションの generator 属性で指定された名前を指定する属性です。
sequenceName	任意	既存のプライマリキー値または事前に定義したプライマリキー値を取得するためのデータベースシーケンスオブジェクトの名前を指定する属性です。
initialValue	任意	シーケンスオブジェクトが，プライマリキー値生成を開始する初期値を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。値を指定した場合は無視されます。
allocationSize	任意	シーケンスからプライマリキー値を割り当てるサイズを指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

@GeneratedValue アノテーションの generator 属性で指定された名前を指定する属性です。
指定できる値は，文字列です。

デフォルト値

なし

(b) sequenceName 属性

型

String

説明

既存のプライマリキー値または事前に定義したプライマリキー値を取得するためのデータベースシーケンスオブジェクトの名前を指定する属性です。
指定できるシーケンスオブジェクト名は，データベースの仕様に依存します。

デフォルト値

@GeneratedValue の generator 属性の指定値

(c) allocationSize 属性

型

int

説明

シーケンスからプライマリキー値を割り当てるサイズを指定する属性です。指定できるシーケンスオブジェクト名は，データベースの仕様に依存します。
指定できるサイズは，int 型の 1 以上の数値です。シーケンスオブジェクトの増分間隔と同じ値を指定します。異なる値を指定した場合の動作は保証しません。
なお，この属性は，実行時に利用する最大値を指定できます。シーケンス番号の管理領域として取得するため，大きな値を指定した場合は，実行時に java.lang.OutOfMemoryError 例外が発生します。

デフォルト値

50

2.6.57 @SqlResultSetMapping

(1) 説明

SQL のクエリの結果セットマッピングを指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@SqlResultSetMapping の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	結果セットマッピングの名前を指定する属性です。
entities	任意	@EntityResult の配列を指定する属性です。

属性名	任意 / 必須	属性の説明
columns	任意	@ColumnResult の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
結果セットマッピングの名前を指定する属性です。
指定できる値は，文字列です。

デフォルト値
なし

(b) entities 属性

型
EntityResult[]

説明
@EntityResult の配列を指定する属性です。
指定できる値は，@EntityResult の配列で指定できる範囲です。詳細は，「2.6.15 @EntityResult」を参照してください。

デフォルト値
空の配列

(c) columns 属性

型
ColumnResult[]

説明
@ColumnResult の配列を指定する属性です。
指定できる値は，@ColumnResult の配列で指定できる範囲です。詳細は，「2.6.7 @ColumnResult」を参照してください。

デフォルト値
空の配列

2.6.58 @SqlResultSetMappings

(1) 説明

@SqlResultSetMapping を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@SqlResultSetMappings の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	@SqlResultSetMapping の配列を指定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型
SqlResultSetMapping[]

説明
@SqlResultSetMapping の配列を指定する属性です。
指定できる値は，@SqlResultSetMapping の配列で指定できる範囲です。詳細は，「2.6.57 @SqlResultSetMapping」を参照してください。

デフォルト値
なし

2.6.59 @Table

(1) 説明

エンティティクラスに，プライマリテーブルを指定するアノテーションです。

明示的にエンティティクラスに @Table を指定しない場合でも，@Table が指定されたようにエンティティクラスは扱われます。その場合，@Table の各属性値にはデフォルト値が適用されます。

エンティティがマッピングするテーブルを複数指定する場合は，@SecondaryTable または @SecondaryTables を使用してください。

適用可能要素は，クラスです。

(2) 属性

@Table の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	任意	テーブル名を指定する属性です。
catalog	任意	テーブルのカタログ名を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
schema	任意	テーブルのスキーマ名を指定する属性です。
uniqueConstraints	任意	テーブルでのユニークキー制約を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型

String

説明

テーブル名を指定する属性です。
指定できるテーブル名は，データベースの仕様に依存します。

デフォルト値

エンティティ名

(b) schema 属性

型

String

説明

テーブルのスキーマ名を指定する属性です。
指定できるスキーマ名は，データベースの仕様に依存します。

デフォルト値

使用するデータベースのデフォルトのスキーマ名

2.6.60 @TableGenerator

(1) 説明

プライマリキーを作成するジェネレータの設定を指定するアノテーションです。

@TableGenerator を使用する場合，次の設定が必要です。

- @GeneratedValue の strategy 属性に GenerationType.TABLE を指定します。
- @GeneratedValue の generator 属性で指定された名前を @TableGenerator の name 属性に設定します。

テーブルジェネレータは，エンティティクラス，またはプライマリキーのフィールドまたはプロパティに指定されます。ジェネレータ名のスコープは永続化ユニットで有効です。

エンティティはプライマリキー値の生成をするために，ジェネレータテーブルの行を使用します。

シーケンスを管理するテーブルを作成する場合，初期値には正の整数を指定します。初期値に 0 を指定した場合，プライマリキーは 1 から生成されます。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@TableGenerator の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
name	必須	プライマリキー値のためのジェネレータ名を指定する属性です。
table	任意	生成されるプライマリキー値を確保するテーブルの名前を指定する属性です。
catalog	任意	生成されるプライマリキー値を確保するテーブルのカタログ名を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。
schema	任意	生成されるプライマリキー値を確保するテーブルのスキーマ名を指定する属性です。
pkColumnName	任意	生成されるプライマリキー値を確保するテーブルのプライマリキーカラム名を指定する属性です。
valueColumnName	任意	生成された最終値を確保するカラム名を指定する属性です。
pkColumnValue	任意	生成されるプライマリキー値を確保するテーブルのプライマリキー値を指定する属性です。
initialValue	任意	生成された最新の値を確保するカラムを初期化するために使う値を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。値を指定した場合は無視されます。
allocationSize	任意	ジェネレータからプライマリキー値を割り当てるサイズを指定する属性です。
uniqueConstraints	任意	生成されるプライマリキー値を確保するテーブル上でのユニークキー制約を指定する属性です。 なお，この属性は，Cosminexus JPA プロバイダには対応していません。値を指定した場合は無視されます。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) name 属性

型
String

説明
プライマリキー値のためのジェネレータ名を指定する属性です。
指定できる値は，文字列です。

デフォルト値
なし

(b) table 属性

型
String

説明
生成されるプライマリキー値を確保するテーブルの名前を指定する属性です。
指定できるテーブル名は，データベースの仕様に依存します。

デフォルト値
"SEQUENCE"

(c) schema 属性

型
String

説明
生成されるプライマリキー値を確保するテーブルのスキーマ名を指定する属性です。
指定できるスキーマ名は，データベースの仕様に依存します。

デフォルト値
使用するデータベースのデフォルトのスキーマ名

(d) pkColumnName 属性

型
String

説明
生成されるプライマリキー値を確保するテーブルのプライマリキーカラム名を指定する属性です。
指定できるカラム名は，データベースの仕様に依存します。

デフォルト値
"SEQ_NAME"

(e) valueColumnName 属性

型
String

説明
生成された最終値を確保するカラム名を指定する属性です。
指定できるカラム名は，データベースの仕様に依存します。

デフォルト値
"SEQ_COUNT"

(f) pkColumnValue 属性

型
String

説明
生成されるプライマリキー値を確保するテーブルのプライマリキー値を指定する属性です。
指定できる値は，生成されたプライマリキーのカラムの型に依存します。

デフォルト値
name 属性で指定された文字列

(g) allocationSize 属性

型
int

説明
ジェネレータからプライマリキー値を割り当てるサイズを指定する属性です。
指定できる値は，int 型の 1 以上の数値です。
なお，この属性は，実行時に利用する最大値を指定できます。シーケンス番号の管理領域として取得するため，大きな値を指定した場合は，実行時に java.lang.OutOfMemoryError 例外が発生します。

デフォルト値
50

2.6.61 @Temporal

(1) 説明

時刻を表す型（java.util.Date および java.util.Calendar）を持つ永続化プロパティまたは永続化フィールドに指定するアノテーションです。@Basic とともに使用できます。

ただし，@Version と @Temporal は同時に指定できません。どちらかのアノテーション

だけを指定してください。

適用可能要素は，メソッドとフィールドです。

(2) 属性

@Temporal の属性の一覧を次の表に示します。

属性名	任意 / 必須	属性の説明
value	必須	データベース型に対応する TemporalType 列挙型に設定する属性です。

Cosminexus JPA プロバイダで対応する属性の詳細を次に示します。

(a) value 属性

型

TemporalType

説明

データベース型に対応する TemporalType 列挙型に設定する属性です。

指定できる値は，次の 3 種類です。

- TemporalType.DATE：java.sql.Data と同じです。
- TemporalType.TIME：java.sql.Time と同じです。
- TemporalType.TIMESTAMP：java.sql.Timestamp と同じです。

デフォルト値

なし

2.6.62 @Transient

(1) 説明

次に示す永続化しないクラスのフィールドまたはプロパティであることを示すアノテーションです。

- エンティティクラス
- マップドスーパークラス
- 埋め込みクラス

適用可能要素は，メソッドとフィールドです。

@Transient を定義したフィールドの値は永続化されませんが，永続化コンテキストに保持されるため，設定した値を取得できます。ただし，別の EntityManager からは値を取得できません。

(2) 属性

なし

2.6.63 @Version

(1) 説明

楽観的ロック機能を使用するために用いる version フィールドまたは version プロパティを指定するアノテーションです。

version フィールドまたは version プロパティでサポートする型を次に示します。

- int
- java.lang.Integer
- short
- java.lang Short
- long
- java.lang Long
- java.sql.Timestamp

このアノテーションを使用する場合は，次のことに注意してください。

- @Version と @Temporal は同時に指定できません。どちらかのアノテーションだけを指定してください。
- version プロパティをプライマリテーブル以外のテーブルには指定しないでください。
- @Version で指定されたフィールドまたはプロパティは，アプリケーションによって更新してはいけません。
- SQL を使用して複数のレコードを一度に更新するバルク更新の場合は，version フィールドまたは version プロパティの自動更新をしません。このため，バルク更新をする場合に楽観的ロックを使用するときは，手動で参照および更新をする必要があります。
- エンティティクラスに対しては，version フィールドまたは version プロパティは一つだけ設定できます。複数の version フィールドまたは version プロパティを設定した場合，一つだけ有効となります。設定が有効になる順番は不定です。

適用可能要素は，メソッドとフィールドです。

(2) 属性

なし

2.6.64 アノテーションと O/R マッピングとの対応

アノテーションと O/R マッピングファイルとの対応を次の表に示します。

表 2-28 アノテーションと O/R マッピングファイルの対応

アノテーション	O/R マッピングの要素
@AssociationOverride	<association-override>
@AssociationOverrides	—
@AttributeOverride	<attribute-override>
@AttributeOverrides	—
@Basic	<basic>
@Column	<column>
@ColumnResult	<column-result>
@DiscriminatorColumn	<discriminator-column>
@DiscriminatorValue	<discriminator-value>
@Embeddable	<embeddable>
@Embedded	<embedded>
@EmbeddedId	<embedded-id>
@Entity	<entity>
@EntityListeners	<entity-listeners>
@EntityResult	<entity-result>
@Enumerated	<enumerated>
@ExcludeDefaultListeners	<exclude-default-listeners>
@ExcludeSuperclassListeners	<exclude-superclass-listeners>
@FieldResult	<field-result>
@GeneratedValue	<generated-value>
@Id	<id>
@IdClass	<id-class>
@Inheritance	<inheritance>
@JoinColumn	<join-column>
@JoinColumns	—
@JoinTable	<join-table>
@Lob	<lob>
@ManyToMany	<many-to-many>
@ManyToOne	<many-to-one>
@MapKey	<map-key>
@MappedSuperclass	<mapped-superclass>
@NamedNativeQueries	—
@NamedNativeQuery	<named-native-query>
@NamedQueries	—

アノテーション	O/R マッピングの要素
@NamedQuery	<named-query>
@OneToMany	<one-to-many>
@OneToOne	<one-to-one>
@OrderBy	<order-by>
@PersistenceContext	―※
@PersistenceContexts	―※
@PersistenceProperty	―※
@PostLoad	<post-load>
@PostPersist	<post-persist>
@PostRemove	<post-remove>
@PostUpdate	<post-update>
@PrePersist	<pre-persist>
@PreRemove	<pre-remove>
@PreUpdate	<pre-update>
@PrimaryKeyJoinColumn	<primary-key-join-column>
@PrimaryKeyJoinColumns	―
@QueryHint	<hint>
@SecondaryTable	<secondary-table>
@SecondaryTables	―
@SequenceGenerator	<sequence-generator>
@SqlResultSetMapping	<sql-result-set-mapping>
@SqlResultSetMappings	―
@Table	<table>
@TableGenerator	<table-generator>
@Temporal	<temporal>
@Transient	<transient>
@UniqueConstraint	<unique-constraint>
@Version	<version>

（凡例）

－：該当しません。

注※

O/R マッピングのためのアノテーションではありません。

2.7 Cosminexus が対応する Dependency Injection

Dependency Injection（DI）とは，ターゲットクラスのフィールドや set メソッドにアノテーション（@EJB や @Resource）を設定することで，EJB やリソースへの参照を EJB コンテナが自動的にセットする機能です。

EJB コンテナ上で動作するクラスの中で，ターゲットクラスとなるクラスを次に示します。

- Enterprise Bean
- インターセプタ

また，Web コンテナ上で動作するクラスの中で，ターゲットクラスとなるクラスを次に示します。

- サーブレット
- フィルタ
- リスナ
- タグハンドラ

Enterprise Bean のホームインタフェース，またはビジネスインタフェースへの参照を DI する場合は，@EJB を設定します。

@Resource を設定した場合は，次の表に示すリソースのタイプを DI できます。

表 2-29 @Resource で DI できるリソースのタイプ

リソースのタイプ	DI の可否※1
java.lang.String※2	×
java.lang.Character※2	×
java.lang.Integer※2	×
java.lang.Boolean※2	×
java.lang.Double※2	×
java.lang.Byte※2	×
java.lang.Short※2	×
java.lang.Long※2	×
java.lang.Float※2	×
javax.xml.rpc.Service	×
javax.xml.ws.Service	×

リソースのタイプ	DI の可否※1
javax.jws.WebService	×
javax.sql.DataSource ※3	○
javax.jms.ConnectionFactory	○
javax.jms.QueueConnectionFactory ※4	○
javax.jms.TopicConnectionFactory	○
javax.mail.Session	○
java.net.URL	×
javax.resource.cci.ConnectionFactory ※5	○
org.omg.CORBA_2_3.ORB	○※6
javax.jms.Queue ※3, ※7	○
javax.jms.Topic ※7	○
javax.resource.cci.InteractionSpec	×
javax.transaction.UserTransaction	○※8
javax.ejb.EJBContext	○※9
javax.ejb.SessionContext	○※9
javax.ejb.TimerService	○※9, ※10
JavaBeans リソース	○
管理対象オブジェクトの独自のインタフェース	○

（凡例）

○：使用できます。

×：使用できません。

注※1

管理対象オブジェクトへの対応づけは，Java Type に関係なく，mappedName 要素で対応づけます。リソースアダプタの表示名と管理対象オブジェクト名の区切り文字には，「!#」を使用してください。

注※2

<env-entry-value> に値を設定できないので，DI，lookup で得られる値を設定できません。

注※3

DB Connector が該当します。

注※4

TP1/Message Queue - Access，Cosminexus RM が該当します。

注※5

uCosminexus TP1 Connector が該当します。

注※6

ORB の shareable 属性は true が設定されているものとして動作します。なお，注入される ORB オブジェクトは，ほかのコンポーネントでも使用される共有のインスタンスです。

注※ 7

Connector 1.5 に準拠したリソースアダプタを使用する場合は，JMS で定義する管理対象オブジェクト（javax.jms.Destination インタフェースまたはサブインタフェース）をリソースアダプタの標準 DD（ra.xml）の <connector>-<resourceadapter>-<adminobject>-<adminobject-interface> タグに指定してください。

注※ 8

CMT で動作する Enterprise Bean またはインターセプタでは使用できません。

注※ 9

Web コンテナ上で動作するクラスでは使用できません。

注※ 10

Stateful SessionBean や，Stateful SessionBean に適用されたインターセプタでは使用できません。

3 Web コンテナで使用する API

この章では，Web コンテナで使用する API について説明します。ここでは，アプリケーションサーバの Web コンテナ独自の例外クラスについて説明します。

3.1 例外クラス

3.1 例外クラス

Web コンテナの API のうち，アプリケーションサーバが提供している例外クラスについて説明します。

Web コンテナの例外クラスを次の表に示します。

表 3-1 Web コンテナの例外クラス

例外名	内容
DatabaseAccessException クラス	データベースセッションフェイルオーバ機能でデータベースへのアクセスに失敗したことを通知する例外です。 この例外が出力された場合，データベースが正常に稼働しているか，データベースと J2EE サーバの通信路に問題が発生していないか，およびデータベースセッションフェイルオーバ機能が無効となっていないか確認し，次の対策をしてください。 データベースに障害が発生している場合 データベースの回復手順に従い原因を対策してください。 データベースと J2EE サーバの通信路に問題が発生している場合 通信路の問題を解決してください。通信路に問題が発生した場合，データベース上の排他が未解放となっていることがあります。業務を再開する前に無効な接続を確認して未解放となっている排他を解放してください。 データベースセッションフェイルオーバ機能が無効の場合 拡張子または URI によるデータベースセッションフェイルオーバ機能の抑止によってデータベースセッションフェイルオーバ機能が無効となったリクエスト処理内では，HttpSession オブジェクトの操作はしないでください。 また，J2EE サーバの webserver.dbsfo.exception_type_backcompat プロパティに true を指定している場合，データベースセッションフェイルオーバ機能の抑止の対象となる URL で HTTP セッションの操作をすると，この例外がスローされます。データベースや通信路に問題がない場合にこの例外が発生したときは，データベースセッションフェイルオーバ機能の抑止の対象となる URL で HTTP セッションの操作をしていないかを確認してください。 DatabaseAccessException クラスは java.lang.IllegalStateException クラスを継承しています。

例外名	内容
HttpSessionLimitExceededException クラス	HttpSession オブジェクトが上限値を超えたことを通知する例外です。 この例外は，HttpSession オブジェクト数の上限が設定できる，J2EE サーバモードの場合に適用されます。HttpSession オブジェクト数の上限が設定できないサーブレットエンジンモードには適用されません。また，SFO サーバでグローバルセッション数が上限を超えた場合の例外には，適用されません。 J2EE アプリケーションを構成するプログラム（サーブレットなど）で com.hitachi.software.web.session.HttpSessionLimitExceededException クラスを使用する場合は，<Cosminexus のインストールディレクトリ >¥CC¥lib¥ejbserver.jar をクラスパスに追加して，Java プログラムをコンパイルしてください。 HttpSessionLimitExceededException クラスは java.lang.IllegalStateException クラスを継承しています。

例外名	内容
SessionOperationException クラス	HttpSession の操作ができない状態であることを通知する例外です。 この例外がスローされる場合を次に示します。 • 拡張子または URI によるデータベースセッションフェイルオーバ機能の抑止を使用した場合，データベースセッションフェイルオーバ機能が無効となったリクエスト処理内では HttpSession オブジェクトの操作はできません。HttpSession オブジェクトを取得するために javax.servlet.http.HttpServletRequest#getSession() または getSession(boolean create) を呼び出した場合，この例外がスローされます。 • HTTP セッションの参照専用リクエストでは，HTTP セッションの無効化はできません。参照専用リクエストで javax.servlet.http.HttpSession#invalidate() を呼び出した場合，この例外がスローされます。 • Web アプリケーション単位の同時実行スレッド数制御の実行待ちキューを使用して 503 エラーを返す設定をした場合，DD（web.xml）で指定するエラーページでは HTTP セッションの作成および無効化はできません。DD（web.xml）で指定したエラーページで HTTP セッションを作成したり，javax.servlet.http.HttpSession#invalidate() を呼び出したりすると，この例外がスローされます。 この例外がスローされた場合は，次の点を確認してください。 • データベースセッションフェイルオーバ機能の抑止を使用している場合は，抑止する拡張子，または URI の設定に問題がないかを確認してください。設定に問題がない場合は，Web アプリケーションを確認して，データベースセッションフェイルオーバ機能の抑止の対象となる URL で HTTP セッションの操作をしていないかどうか確認してください。 • HTTP セッションの参照専用リクエスト定義機能を使用している場合は，HTTP セッションの参照専用リクエストの URI の設定に問題がないかを確認してください。設定に問題がない場合は，Web アプリケーションを確認して，参照専用リクエストで HTTP セッションを無効化していないか確認してください。 • 実行待ちキューを使用して 503 エラーを返す設定をしている場合は，DD（web.xml）で指定したエラーページで HTTP セッションの作成または無効化していないか確認してください。 SessionOperationException クラスは java.lang.IllegalStateException クラスを継承しています。

4 EJBクライアントアプリケーションで使用するAPI

この章では，EJBクライアントアプリケーションで使用するAPIおよび例外クラスについて説明します。

4.1 EJB クライアントアプリケーションで使用する API の一覧

EJB クライアントアプリケーションで使用する API には，セキュリティ機能および通信タイムアウトを設定する API があります。API の一覧を次の表に示します。

表 4-1 EJB クライアントアプリケーションで使用する API の一覧

クラス名	機能
EJBClientInitializer クラス	EJB クライアント用の J2EE サービスを初期化します。
LoginInfoManager クラス	セキュリティ機能を設定します。
RequestTimeoutConfigFactory クラス	RMI-IIOP タイムアウトを設定するために必要な RequestTimeoutConfig オブジェクトを取得します。
RequestTimeoutConfig クラス	RMI-IIOP タイムアウトを設定します。
UserTransactionFactory クラス	EJB クライアントでトランザクションを使用するための UserTransaction オブジェクトを取得します。

4.2 EJBClientInitializer クラス

説明

EJB クライアント用の J2EE サービスを初期化します。
EJBClientInitializer クラスのパッケージ名は，
com.hitachi.software.ejb.ejbclient.EJBClientInitializer です。

メソッド一覧

メソッド名	機能
initialize メソッド	EJB クライアント用の J2EE サービスを初期化します。

initialize メソッド

説明

EJB クライアントアプリケーション用の J2EE サービスを初期化します。また，トランザクション処理中に EJB クライアントが停止した場合，EJB クライアントを再起動したあとに，グローバルトランザクションのリカバリ処理を開始します。

EJB クライアントプロセスの開始直後に，EJB クライアントのユーザコードから，initialize メソッドを呼び出してください。

なお，initialize メソッドを呼び出す前に，javax.naming.InitialContext を生成した場合，または UserTransactionFactory クラスの getUserTransaction メソッドを呼び出した場合，その時点で初期化処理が行われます。

形式

```
public static void initialize()
  throws InitializeFailedException;
```

パラメタ

なし

例外

com.hitachi.software.ejb.ejbclient.InitializeFailedException：
サービスの初期化に失敗しました。

戻り値

なし

注意事項

サービスの初期化処理で例外が発生した場合は，EJB クライアント実行時のシステムプロパティが正しく設定されていないおそれがあります。例外のメッセージに従って対処してください。

4.3　LoginInfoManager クラス

説明

J2EE サーバに設定したユーザとパスワードを使って，セキュリティ認証を実行します。

セキュリティ認証を実行する J2EE サーバについて説明します。

- ejbserver.security.service.url **プロパティを指定している場合**

  セキュリティ認証は，ejbserver.security.service.url プロパティに指定した CORBA ネーミングサービスに接続している J2EE サーバのうち，ejbserver.serverName プロパティで指定したサーバ名称と一致する J2EE サーバで実行されます。

  ejbserver.security.service.url プロパティは，java.naming.provider.url プロパティに指定した CORBA ネーミングサービスに接続していない J2EE サーバでセキュリティ認証を実行する場合に，指定してください。
- ejbserver.security.service.url **プロパティを指定していない場合**

  セキュリティ認証は，java.naming.provider.url プロパティに指定した CORBA ネーミングサービスに接続している J2EE サーバのうち，ejbserver.serverName プロパティで指定したサーバ名称と一致する J2EE サーバで実行されます。

JNDI ラウンドロビン検索機能や CTM 連携機能などを使った負荷分散構成の場合は，セキュリティ認証を実行できる J2EE サーバが複数存在することになります。この場合，構成する J2EE サーバすべてに同じユーザ，および同じロールの情報を設定した上で，セキュリティ認証用の J2EE サーバを一つ決定してください。

各プロパティの詳細については，マニュアル「Cosminexus アプリケーションサーバ リファレンス 定義編 (サーバ定義)」の「15.　Java アプリケーションで使用するファイル」を参照してください。EJB クライアントアプリケーションでのセキュリティの実装方法については，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 (EJB コンテナ)」の「3.6　EJB クライアントアプリケーションでのセキュリティの実装」を参照してください。

LoginInfoManager クラスのパッケージ名は，com.hitachi.software.ejb.security.base.authentication です。

メソッド一覧

メソッド名	機能
getLoginInfoManager メソッド	LoginInfoManager オブジェクトを取得します。
login メソッド	J2EE サーバにログインします。
logout メソッド	J2EE サーバからログアウトします。

注意事項

LoginInfoManager クラスのメソッドを使用する場合は次の点に注意してください。

- LoginInfoManager クラスのメソッドは，EJB クライアントアプリケーションから発行することを推奨しています。JSP，サーブレット，または EJB 内から発行した場合，RunAs 機能によって設定した情報が，リクエスト単位で削除されます。
- login メソッドを発行して J2EE サーバを呼び出したあとは，必ず logout メソッドを発行してください。
- login メソッドおよび logout メソッドを入れ子で発行しないでください。logout メソッドを発行しないで login メソッドを連続で発行すると，1 回目の login メソッドで指定した情報が 2 回目の login メソッドによって上書きされます。

getLoginInfoManager メソッド

説明

LoginInfoManager オブジェクトを取得します。

形式

```
public static LoginInfoManager getLoginInfoManager();
```

パラメタ

なし

例外

なし

戻り値

LoginInfoManager オブジェクトを返却します。

login メソッド

説明

J2EE サーバにログインします。

形式

```
public final boolean login(String username,
                           String password)
  throws NotFoundServerException,InvalidUserNameException,
         InvalidPasswordException;
```

パラメタ

username：

ユーザ名（plain text）を指定します。

password：
パスワード（plain text）を指定します。

例外

com.hitachi.software.ejb.security.base.authentication.NotFoundServerException：
J2EE サーバが見つかりません。

com.hitachi.software.ejb.security.base.authentication.InvalidUserNameException：
指定されたユーザ名が見つかりません。

com.hitachi.software.ejb.security.base.authentication.InvalidPasswordException：
指定したパスワードが不正です。

戻り値

true：
ログインに成功しました。

false：
ログインに失敗しました。

logout メソッド

説明

J2EE サーバからログアウトします。

形式

```
public final void logout();
```

パラメタ

なし

例外

なし

戻り値

なし

4.4 RequestTimeoutConfigFactory クラス

説明

RMI-IIOP 通信タイムアウトを設定するオブジェクトである RequestTimeoutConfig を取得するためのファクトリです。getRequestTimeoutConfig メソッドで RequestTimeoutConfig を取得したあと，RequestTimeoutConfig のメソッドでタイムアウトを設定します。
RequestTimeoutConfigFactory クラスのパッケージ名は，com.hitachi.software.ejb.ejbclient です。

メソッド一覧

メソッド名	機能
getRequestTimeoutConfig メソッド	RequestTimeoutConfig オブジェクトを取得します。

getRequestTimeoutConfig メソッド

説明

RequestTimeoutConfig オブジェクトを取得します。

形式

```
public static RequestTimeoutConfig getRequestTimeoutConfig();
```

パラメタ

なし

例外

なし

戻り値

RequestTimeoutConfig：
RequestTimeoutConfig オブジェクトを返却します。

4.5 RequestTimeoutConfig クラス

説明

RMI-IIOP 通信タイムアウトを設定するオブジェクトです。
RequestTimeoutConfig クラスのパッケージ名は，
com.hitachi.software.ejb.ejbclient です。

メソッド一覧

メソッド名	機能
setRequestTimeout メソッド（形式 1）	RMI-IIOP 通信タイムアウトを設定します。 オブジェクトにタイムアウトを設定します。
setRequestTimeout メソッド（形式 2）	RMI-IIOP 通信タイムアウトを設定します。 スレッドにタイムアウトを設定します。
unsetRequestTimeout メソッド	setRequestTimeout メソッド（形式 2）で設定した RMI-IIOP 通信タイムアウトの設定をデフォルト設定に戻します。

setRequestTimeout メソッド（形式 1）

説明

RMI-IIOP 通信タイムアウトを設定します。obj パラメタのコピーを生成し，sec パラメタをタイムアウト値として設定したオブジェクトを返却します。このメソッドで設定したタイムアウトは，返却されたオブジェクトに対して有効です。

形式

```
public java.rmi.Remote setRequestTimeout(java.rmi.Remote obj,
                                         int sec)
  throws IllegalArgumentException,
         IllegalStateException;
```

パラメタ

obj：

タイムアウトを設定するオブジェクト（EJBHome または EJBObject）を指定します。

sec：

0 ～ 86400 の整数でタイムアウト時間（単位：秒）を指定します。0 を指定した場合，タイムアウトを設定しません。

例外

java.lang.IllegalArgumentException：

タイムアウト設定対象として不正なオブジェクト，またはタイムアウト時間として不正な値を指定しました。

java.lang.IllegalStateException：
タイムアウトの設定に失敗しました。

戻り値

タイムアウト設定済みのオブジェクトを返却します。

注意事項

このメソッドでタイムアウトを設定する場合，setRequestTimeout メソッド（形式 2）を使用してタイムアウトを設定する場合に比べて，処理に時間が掛かります。

setRequestTimeout メソッド（形式 2）

説明

RMI-IIOP 通信タイムアウトを設定します。実行中のスレッドに対し，パラメタ sec をタイムアウト値として設定します。このメソッドで設定したタイムアウトは，現在実行中のスレッドに対して有効です。なお，処理の終了時には，unset メソッドを使用して必ずタイムアウトの設定を解除してください。同一スレッド内でこのメソッドを複数回呼び出した場合，タイムアウトの設定値が上書きされます。

形式

```
public void setRequestTimeout(int sec)
  throws IllegalArgumentException,
         IllegalStateException;
```

パラメタ

sec：
0 ～ 86400 の整数でタイムアウト時間（単位：秒）を指定します。0 を指定した場合，タイムアウトを設定しません。

例外

java.lang.IllegalArgumentException：
タイムアウト設定対象として不正なオブジェクト，またはタイムアウト時間として不正な値を指定しました。

java.lang.IllegalStateException：
タイムアウトの設定に失敗しました。

戻り値

なし

注意事項

このメソッドでタイムアウトを設定する場合は，処理が終わった時点で必ずunsetRequestTimeout メソッドを呼び出してタイムアウトの設定を解除してください。解除しないと，ほかのクライアントからの呼び出しに対して該当スレッドが使用された場合に，そのクライアントにとって意図しない通信タイムアウトが発生するおそれがあります。

unsetRequestTimeout メソッド

説明

RMI-IIOP 通信タイムアウトの設定を解除します。実行中のスレッドに対し，setRequestTimeout（形式 2）で設定したタイムアウトを解除します。なお，setRequestTimeout（形式 2）でスレッドにタイムアウトを設定した場合は，処理の終了時に必ずこのメソッドを使用してタイムアウトの設定を解除してください。setRequestTimeout（形式 2）を呼び出さないでこのメソッドを呼び出した場合や，同一スレッド内でこのメソッドを複数回呼び出した場合でも，例外は発生しません。

形式

```
public void unsetRequestTimeout()
  throws IllegalStateException;
```

パラメタ

なし

例外

java.lang.IllegalStateException：
　タイムアウトの解除に失敗しました。

戻り値

なし

4.6 UserTransactionFactory クラス

説明

EJB クライアントでトランザクションを使用するためのオブジェクトである UserTransaction オブジェクトを取得するためのファクトリです。
UserTransactionFactory クラスのパッケージ名は，com.hitachi.software.ejb.ejbclient.UserTransactionFactory です。

メソッド一覧

メソッド名	機能
getUserTransaction メソッド	UserTransaction オブジェクトを取得します。

getUserTransaction メソッド

説明

UserTransaction オブジェクトを取得します。

形式

```
public static UserTransaction getUserTransaction();
```

例外

java.lang.IllegalStateException：

EJB クライアント以外から API を発行しました。または，UserTransaction オブジェクトの取得に失敗しました。

戻り値

javax.transaction.UserTransaction オブジェクト

4.7 例外クラス

EJB クライアントアプリケーションの API で使用する例外クラスのうち，アプリケーションサーバが提供しているクラスについて説明します。

EJB クライアントアプリケーションの API で使用する例外クラスを次の表に示します。

表 4-2 EJB クライアントアプリケーションの API で使用する例外クラス

例外名	内容
com.hitachi.software.ejb.security.base.authentication.NotFoundServerException	LoginInfoManager クラスの login メソッドでログインしようとした場合に，ログイン先の J2EE サーバに接続できなかったときに送出されます。 ejbserver.serverName プロパティに指定する J2EE サーバ名が，ログイン先の J2EE サーバ名と同じになっていることを確認してください。また，ログイン先の J2EE サーバが起動していることを確認してください。
com.hitachi.software.ejb.security.base.authentication.InvalidUserNameException	LoginInfoManager クラスの login メソッドでログインしようとした場合に，ユーザ名が不正だったときに送出されます。 ユーザ名が正しいかを確認してください。
com.hitachi.software.ejb.security.base.authentication.InvalidPasswordException	LoginInfoManager クラスの login メソッドでログインしようとした場合に，パスワードが不正だったときに送出されます。 パスワードが正しいかを確認してください。

5 TP1インバウンドアダプタによってOpenTP1と連携する場合に使用するAPI

この章では，TP1インバウンドアダプタによってOpenTP1と連携する場合に使用するAPIについて説明します。

5.1 TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API の一覧

TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API の一覧を次の表に示します。

表 5-1 TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API の一覧

インタフェース名	機能
TP1InMessage インタフェース	OpenTP1 の RPC に指定された入力メッセージを取得したり，応答用の出力メッセージを生成したりするためのインタフェースです。
TP1MessageListener インタフェース	TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出す onMessage メソッドを提供するインタフェースです。
TP1OutMessage インタフェース	OpenTP1 からの RPC の応答時に返す出力メッセージを保持するためのインタフェースです。

5.2 TP1InMessage インタフェース

説明

OpenTP1 の RPC に指定された入力メッセージオブジェクトを取得したり，応答用の出力メッセージオブジェクトを生成したりするためのインタフェースです。
TP1InMessage インタフェースのパッケージ名は，
com.hitachi.software.ejb.adapter.tp1 です。

形式

```
public interface TP1InMessage
{
  public byte[] getInputData();
  public TP1OutMessage createOutMessage();
}
```

メソッド一覧

メソッド名	機能
getInputData メソッド	OpenTP1 の RPC に指定された入力メッセージオブジェクトを取得するメソッドです。
createOutMessage メソッド	OpenTP1 の RPC の応答用の出力メッセージオブジェクトを生成するメソッドです。

getInputData メソッド

説明

OpenTP1 の RPC に指定された入力メッセージオブジェクトを取得するメソッドです。

形式

```
public byte[] getInputData();
```

パラメタ

なし

例外

なし

戻り値

OpenTP1 の RPC に指定された入力メッセージオブジェクトです。サイズは，RPC の in_len で指定された値です。

createOutMessage メソッド

説明

OpenTP1 の RPC の応答用の出力メッセージオブジェクトを生成するメソッドです。

形式

```
public TP1OutMessage createOutMessage();
```

パラメタ

なし

例外

なし

戻り値

サービスの出力メッセージオブジェクトです。OpenTP1 の RPC の応答時に返す出力メッセージを保持します。

5.3 TP1MessageListener インタフェース

説明

TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出す onMessage メソッドを提供するインタフェースです。TP1 インバウンドアダプタから呼び出すサービスでビジネスロジックを実装する必要があります。

TP1MessageListener インタフェースのパッケージ名は,
com.hitachi.software.ejb.adapter.tp1 です。

形式

```
public interface TP1MessageListener
{
  public TP1OutMessage onMessage(TP1InMessage in);
}
```

メソッド一覧

メソッド名	機能
onMessage メソッド	TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出されるメソッドです。

onMessage メソッド

説明

TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出すメソッドです。

形式

```
public TP1OutMessage onMessage(TP1InMessage in);
```

パラメタ

in：

サービスの入力メッセージオブジェクトを指定します。

例外

なし

戻り値

サービスの出力メッセージオブジェクトです。OpenTP1 の RPC の応答時に返す出力パラメタを保持します。

5.4 TP1OutMessage インタフェース

説明

OpenTP1 からの RPC の応答時に返す出力メッセージを保持するためのインタフェースです。

TP1OutMessage インタフェースのパッケージ名は，com.hitachi.software.ejb.adapter.tp1 です。

形式

```
public interface TP1OutMessage
{
public byte[] getOutputData(int outLen) throws
IllegalArgumentException;
public int getMaxOutputLength();
}
```

メソッド一覧

メソッド名	機能
getOutputData メソッド	OpenTP1 の RPC の応答を格納する byte 配列を取得するメソッドです。
getMaxOutputLength メソッド	OpenTP1 の RPC の応答の長さを返すメソッドです。

getOutputData メソッド

説明

OpenTP1 の RPC の応答を格納する byte 配列を取得するメソッドです。取得した byte 配列に出力データを格納することで，RPC の応答データを設定できます。

getOutputData メソッドが複数回呼び出された場合，最後に呼び出された getOutputData メソッドが取得した byte 配列が，OpenTP1 への応答として使用されます。それ以前に呼び出された getOutputData メソッドが取得した byte 配列は使用されません。

形式

```
public byte[] getOutputData(int outLen) throws
IllegalArgumentException;
```

パラメタ

outLen：

応答の長さ（バイト数）を，0 ～ <getMaxOutputLength メソッドで得られる長さ> で指定します。

例外

IllegalArgumentException：

パラメタの outLen に 0 ～ <getMaxOutputLength メソッドで得られる長さ > 以外の値を指定しました。

戻り値

OpenTP1 の RPC の応答を格納する byte 配列です。サイズは，パラメタの out_Len で指定された値です。

getMaxOutputLength メソッド

説明

OpenTP1 の RPC の要求で指定された応答の長さを返します。getMaxOutputLength メソッドが返す値が，getOutputData メソッドのパラメタの outLen に指定できる最大の長さとなります。getMaxOutputLength メソッドが返す値には，getOutputData メソッドのパラメタの outLen に指定した値は反映されません。

形式

```
public int getMaxOutputLength();
```

パラメタ

なし

例外

なし

戻り値

OpenTP1 の RPC の要求で指定された応答の長さです。

6 スレッドの非同期並行処理で使用する API

この章では，スレッドの非同期並行処理で使用する API について説明します。ここでは，Timer and Work Manager for Application Servers 仕様が定義する API と動作が異なる Cosminexus の API について説明します。

6.1 Timer and Work Manager for Application Servers 仕様と動作が異なる Cosminexus の API の一覧

Timer and Work Manager for Application Servers 仕様が定義する API と動作が異なる Cosminexus の API の名称および動作を次の表に示します。

表 6-1 Timer and Work Manager for Application Servers 仕様と動作が異なる Cosminexus の API の一覧

クラス名	メソッド名	Cosminexus での動作
commonj.timers.TimerManager クラス	schedule(TimerListener listener,Date time) メソッド	listener が javax.ejb.EnterpriseBean を継承している場合，IllegalArgumentException を返します。
	schedule(TimerListener listener,long delay) メソッド	
	schedule(TimerListener listener,Date firstTime,long period) メソッド	
	schedule(TimerListener listener,long delay,long period) メソッド	
	scheduleAtFixedRate(TimerListener listener,Date firstTime,long period) メソッド	
	scheduleAtFixedRate (TimerListener listener,long delay,long period) メソッド	
commonj.work.WorkManager クラス	schedule(Work work) メソッド	work が null の場合，WorkException をスローします。
	schedule(Work work,WorkListener wl) メソッド	work が null の場合，WorkException を返します。 WorkListener が javax.ejb.EnterpriseBean を継承している場合，IllegalArgumentException を返します。

7 クライアント性能モニタ機能で使用する API

この章では，クライアント性能モニタ機能で使用する API について説明します。

7.1 クライアント性能モニタ機能で使用する API の一覧

クライアント性能モニタ機能で使用する API の一覧を次の表に示します。

表 7-1 クライアント性能モニタ機能で使用する API

メソッド名	機能
ClientPerformance.getAllLog メソッド	Web ブラウザ上に保存されている性能データを JavaScript の配列として参照可能にします。

7.2 ClientPerformance.getAllLog メソッド

説明

ClientPerformance.getAllLog メソッドは JavaScript の API です。Web ブラウザ上に保存されている性能データを JavaScript の配列として参照可能にします。

ClientPerformance.getAllLog メソッドを使用することで，性能データをサーバ側で収集するアプリケーションが実装できます。アプリケーションの作成方法については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 拡張編」の「9.2.6 Web ブラウザから性能データを読み出すアプリケーションの作成」を参照してください。

なお，ClientPerformance.getAllLog メソッドを使用して参照できる性能データは，モニタページで表示できるすべての性能データです。次に示す性能データはモニタページに表示されないため，参照できません。

- すでに削除されている性能データ
- 有効期限が切れている性能データ

形式

ClientPerformance.getAllLog()

パラメタ

なし

例外

なし

戻り値

参照できるすべての性能データを文字列の配列として返却します。参照できる性能データがなかった場合は，[]（空の配列）を返却します。また，API を実行するために必要なオブジェクトが見つからなかった場合，そのほかの例外が発生した場合など，エラーが発生した場合は null を返却します。

配列の各要素が一つのログシーケンスを表した文字列になっています。文字列の形式を次に示します。

```
（性能データ），（性能データ），（性能データ），…
```

各性能データの文字列の形式を次に示します。

```
（ 問い合わせID，ロード時間，通信時間，描画時間，操作時間，パス，（子フレームの問い合わせID，子フレームの問い合わせID,…） ）
```

各要素について次に示します。

問い合わせ ID

問い合わせ ID を示します。

ロード時間

ロード時間（万国標準時（UTC）の 1970 年 1 月 1 日の 00:00:00 からの経過時間）をミリ秒単位で示します。

通信時間
通信時間をミリ秒単位で示します。

描画時間
描画時間をミリ秒単位で示します。

操作時間
操作時間をミリ秒単位で示します。

パス
パスを「ハッシュ値 `::` コンテキストルート以下のパス」の形式で示します。
ディレクトリ部分を除いたファイル名は，クライアント性能フィルタの初期化パラメタ urlMaxLength に指定した値以下です。指定値を超える文字列は省略されます。ハッシュ値はファイル名が省略される前のパスを識別するための値です。

子フレームの問い合わせ ID
子フレームがある場合に子フレームの問い合わせ ID を示します。子フレームがない場合は，0 になります。

注意事項

- ClientPerformance.getAllLog メソッドを呼び出す場合，次に示すページを IFRAME タグで読み込み，フレーム越しに API を呼び出してください。
  < クライアント性能フィルタ適用アプリケーションのコンテキストルート >/`cpmonitor_a_LoadAllUserData.html`
- ClientPerformance.getAllLog メソッドは，IFRAME タグのロードが完了したことを確認してから呼び出してください。
- 性能データのパス情報の中で，ディレクトリ部分は Web ブラウザ上に保存されていない場合があります。この場合，各要素に示されるパスは次の形式になります。
  ハッシュ値 `::[dir]/` ファイル名

8 統合ユーザ管理フレームワークで使用する API

この章では，統合ユーザ管理フレームワークで使用する API および例外クラスについて説明します。

8.1　統合ユーザ管理フレームワークで使用する API の一覧

統合ユーザ管理フレームワークのライブラリを利用してユーザ認証を実装する場合に使用する API および例外クラスの一覧を次の表に示します。

表 8-1　統合ユーザ管理フレームワークで使用する API および例外クラスの一覧

クラス・インタフェース名	機能	API の種別
AttributeEntry クラス	属性名と Alias を対で管理します。	ユーザ認証ライブラリ
ChangeDataFailedException クラス	SSODataListener インタフェースの実装クラスが呼び出す例外クラスです。	シングルサインオンライブラリ（例外クラス）
DelegationLoginModule クラス	JAAS のログインモジュールの実装クラスです。カスタムログインモジュールを呼び出します。	Cosminexus 標準ログインモジュール
LdapSSODataManager クラス	LDAP ディレクトリサーバのシングルサインオン情報リポジトリの情報を参照または更新します。	シングルサインオンライブラリ
LdapUserDataManager クラス	LDAP ディレクトリサーバの，ユーザ情報リポジトリの情報を参照または更新します。	ユーザ認証ライブラリ
LdapUserEnumeration インタフェース	ユーザ ID の一覧を参照します。	ユーザ認証ライブラリ
LoginUtil クラス	統合ユーザ管理のセッション内でログインしているユーザの有無を調べます。	ユーザ認証ライブラリ
ObjectClassEntry クラス	LDAP ディレクトリサーバのエントリのオブジェクトクラスを格納します。	ユーザ認証ライブラリ
PasswordCryptography インタフェース	ユーザが入力したパスワードを暗号化します。	ユーザ認証ライブラリ
PasswordUtil クラス	ユーザが入力したパスワードを変更します。	ユーザ認証ライブラリ
Principal インタフェース	WebPasswordLoginModule が認証したときのユーザ ID を参照します。	ユーザ認証ライブラリ
SSOData クラス	シングルサインオン用認証情報を格納します。	シングルサインオンライブラリ
SSODataEvent クラス	シングルサインオン用認証情報の更新内容を格納します。	シングルサインオンライブラリ
SSODataListener インタフェース	シングルサインオン用認証情報の更新を通知します。	シングルサインオンライブラリ
SSODataListenerException クラス	シングルサインオン用認証情報リスナクラスで例外が発生した場合に呼び出される例外クラスです。	シングルサインオンライブラリ（例外クラス）

クラス・インタフェース名	機能	API の種別
UserAttributes インタフェース	WebPasswordLoginModule が認証したときに作成した Credential を参照します。	ユーザ認証ライブラリ
UserData クラス	ユーザ情報を格納します。	ユーザ認証ライブラリ
WebCertificateCallback クラス	JAAS の Callback の実装クラスです。Web サーバの SSL 認証した結果の情報を格納します。	ユーザ認証ライブラリ
WebCertificateHandler クラス	JAAS の CallbackHandler の実装クラスです。Web サーバの SSL 認証した結果で必要な情報を読み込みます。	ユーザ認証ライブラリ
WebCertificateLoginModule クラス	JAAS のログインモジュールの実装クラスです。Web サーバで認証された証明書からユーザ属性を求めます。	Cosminexus 標準ログインモジュール
WebLogoutCallback クラス	JAAS の Callback の実装クラスです。ログアウトするユーザを格納します。	ユーザ認証ライブラリ
WebLogoutHandler クラス	JAAS の CallbackHandler の実装クラスです。ログアウトに必要なユーザを読み込みます。	ユーザ認証ライブラリ
WebPasswordCallback クラス	JAAS の Callback の実装クラスです。パスワードなどの認証情報を格納します。	ユーザ認証ライブラリ
WebPasswordHandler クラス	JAAS の CallbackHandler の実装クラスです。パスワード認証に必要な情報を読み込みます。	ユーザ認証ライブラリ
WebPasswordJDBCLoginModule クラス	JAAS のログインモジュールの実装クラスです。JDBC を使ってデータベースにアクセスし，パスワード認証をします。	Cosminexus 標準ログインモジュール
WebPasswordLDAPLoginModule クラス	JAAS のログインモジュールの実装クラスです。LDAP ディレクトリサーバにバインドした結果で認証をします。	Cosminexus 標準ログインモジュール
WebPasswordLoginModule クラス	JAAS のログインモジュールの実装クラスです。Web アプリケーションのためのパスワード認証をします。	Cosminexus 標準ログインモジュール
WebSSOCallback クラス	シングルサインオンライブラリが提供する JAAS の Callback の実装クラスです。WebSSOLoginModule で必要な情報を知るために使用します。	シングルサインオンライブラリ
WebSSOHandler クラス	シングルサインオンライブラリが提供する JAAS の CallbackHandler の実装クラスです。WebSSOLoginModule で必要な情報を読み込みます。	シングルサインオンライブラリ
WebSSOLoginModule クラス	JAAS のログインモジュールの実装クラスです。シングルサインオンをするためにほかのログインモジュールを呼び出します。	Cosminexus 標準ログインモジュール

クラス・インタフェース名	機能	API の種別
例外クラス	統合ユーザ管理で使用する API の例外クラスです。	例外クラス

8.2 AttributeEntry クラス

説明

ユーザ管理リポジトリから取得する属性の属性名，別名（Alias），およびユーザ管理コンテキストからのサブコンテキストのタプルを表すクラスです。ユーザ認証後，指定した属性は Subject の Public Credential に別名で関連づけられます。別名を指定しなかった場合は，属性名で関連づけられます。

AttributeEntry クラスのパッケージ名は，com.cosminexus.admin.auth です。

形式

```
class AttributeEntry
{
public AttributeEntry(String attr,
                      String alias,
                      String subcontext);
public AttributeEntry(String attr,
                      String alias);
public AttributeEntry(String attr);
public AttributeEntry();

public String getAlias();
public String getAttributeName();
public String getSubcontext();
public void setAlias(String alias);
public void setAttributeName(String attr);
public void setSubcontext(String subcontext);
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
AttributeEntry コンストラクタ	AttributeEntry クラスのインスタンスを生成します。
getAlias メソッド	setAlias メソッドまたはコンストラクタで指定した別名を取得します。
getAttributeName メソッド	setAttributeName メソッドまたはコンストラクタで指定した属性名を取得します。
getSubcontext メソッド	setSubcontext メソッドまたはコンストラクタで指定したサブコンテキストを取得します。
setAlias メソッド	パラメタに指定された別名をオブジェクトに保管します。
setAttributeName メソッド	パラメタに指定された属性名をオブジェクトに保管します。
setSubcontext メソッド	パラメタに指定されたサブコンテキストをオブジェクトに保管します。

AttributeEntry コンストラクタ

説明

AttributeEntry クラスのインスタンスを作るためのコンストラクタです。

形式

```
public AttributeEntry(String attr,
                      String alias,
                      String subcontext);

public AttributeEntry(String attr,
                      String alias);

public AttributeEntry(String attr);

public AttributeEntry();
```

パラメタ

attr：

リポジトリに格納されている属性名を指定します。

alias：

属性名に対応する別名（Alias）を指定します。

subcontext：

サブコンテキストを指定します。

例外

なし

getAlias メソッド

説明

setAlias メソッドまたはコンストラクタで指定した内容を取得します。値を保管していないときに getAlias メソッドを呼び出すと null が返却されます。

形式

```
public String getAlias();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getAttributeName メソッド

説明

setAttributeName メソッドまたはコンストラクタで指定した内容を取得します。値を保管していないときに getAttributeName メソッドを呼び出すと null が返却されます。

形式

```
public String getAttributeName();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getSubcontext メソッド

説明

setSubcontext メソッドまたはコンストラクタで指定した内容を取得します。値を保管していないときに getSubcontext メソッドを呼び出すと null が返却されます。

形式

```
public String getSubcontext();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

setAlias メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合に setAlias メソッドを呼び出したときは上書きされます。

形式

```
public void setAlias(String alias);
```

パラメタ

alias：
　属性名に対応する別名（Alias）を指定します。

例外

なし

戻り値

なし

setAttributeName メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合に setAttributeName メソッドを呼び出したときは上書きされます。

形式

```
public void setAttributeName(String attr);
```

パラメタ

attr：
　属性名を指定します。

例外

なし

戻り値

なし

setSubcontext メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合に setSubcontext メソッドを呼び出したときは上書きされます。

形式

```
public void setSubcontext(String subcontext);
```

パラメタ

subcontext：

サブコンテキストを指定します。

例外

なし

戻り値

なし

8.3　ChangeDataFailedException クラス

説明

SSODataListener インタフェースの実装クラスが，データの追加，修正，または削除に失敗したときに呼び出す例外クラスです。
ChangeDataFailedException クラスのパッケージ名は，
com.cosminexus.admin.auth.api.repository.event です。

形式

```
class ChangeDataFailedException extends UAException
{
public ChangeDataFailedException();
public ChangeDataFailedException(String msg);
}
```

コンストラクター一覧

コンストラクタ名	機能
ChangeDataFailedException コンストラクタ	ChangeDataFailedException クラスのインスタンスを生成します。

ChangeDataFailedException コンストラクタ

説明

パラメタに指定されたエラーメッセージを使用してインスタンスを生成します。

形式

```
public ChangeDataFailedException();

public ChangeDataFailedException(String msg);
```

パラメタ

msg：
　エラーメッセージを指定します。

例外

なし

8.4 DelegationLoginModule クラス

説明

ユーザ認証ライブラリが提供する JAAS のログインモジュールの実装クラスです。
カスタムログインモジュールを呼び出します。
パッケージ名は com.cosminexus.admin.auth.login です。

8.5　LdapSSODataManager クラス

説明

LDAP ディレクトリサーバのシングルサインオン情報リポジトリに格納されている情報を参照または更新するクラスです。
LdapSSODataManager クラスのパッケージ名は,
com.cosminexus.admin.auth.api.repository.ldap です。

形式

```
class LdapSSODataManager
{
  public LdapSSODataManager(String realm);

  public LdapUserEnumeration listUsers()
    throws NamingException;
  public LdapUserEnumeration listUsers(String uid)
    throws NamingException;
  public SSOData getSSOData(String uid)
    throws NamingException;
  public void addSSOData(String uid,
                         SSOData ssoData)
    throws SSODataListenerException, NamingException,
    CryptoException, UnsatisfiedLinkError, SecurityException;
  public void removeSSOData(String uid)
    throws SSODataListenerException, NamingException,
    CryptoException, UnsatisfiedLinkError, SecurityException;
  public void modifySSOData(String uid,
                            SSOData ssoData)
    throws SSODataListenerException, NamingException,
    CryptoException, UnsatisfiedLinkError, SecurityException;
  public SSODataListener[] getSSODataListeners();
  public void addSSODataListener(SSODataListener listener);
  public void removeSSODataListener(SSODataListener listener);
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
LdapSSODataManager コンストラクタ	LdapSSODataManager クラスのインスタンスを生成します。
addSSOData メソッド	シングルサインオン用認証情報を追加します。
addSSODataListener メソッド	シングルサインオン用認証情報リスナを登録します。
getSSOData メソッド	シングルサインオン用認証情報を取得します。
getSSODataListeners メソッド	SSODataListener オブジェクトの配列を取得します。
listUsers メソッド（形式 1）	すべてのユーザ ID の一覧を取得します。
listUsers メソッド（形式 2）	ユーザ ID の一覧を取得します。
modifySSOData メソッド	シングルサインオン用認証情報を修正します。
removeSSOData メソッド	シングルサインオン用認証情報を削除します。
removeSSODataListener メソッド	SSODataListener オブジェクトを削除します。

LdapSSODataManager コンストラクタ

説明

インスタンスを生成します。

形式

```
public LdapSSODataManager(String realm);
```

パラメタ

realm：
　生成したインスタンスがアクセス対象にするレルム名を指定します。

例外

なし

addSSOData メソッド

説明

指定したユーザのシングルサインオン用認証情報を追加します。指定したユーザのシングルサインオン用認証情報がすでにある場合は，例外が発生します。

このオブジェクトに登録されているすべてのシングルサインオン用認証情報リスナの ssoDataAdded メソッドが呼び出されます。

形式

```
public void addSSOData(String uid,
                       SSOData ssoData)
  throws SSODataListenerException, NamingException,
  CryptoException, UnsatisfiedLinkError, SecurityException;
```

パラメタ

uid：
　ユーザ ID を指定します。

ssoData：
　シングルサインオン用認証情報を格納した SSOData オブジェクトを指定します。

例外

com.cosminexus.admin.auth.api.repository.event.SSODataListenerException：
　他システムの認証情報更新に失敗しました。

com.cosminexus.admin.auth.CryptoException：

暗号鍵ファイルの読み込みに失敗しました。または，誤った暗号鍵ファイルを使用したため SecretData の復号化に失敗しました。

java.lang.UnsatisfiedLinkError：
シングルサインオンライブラリの読み込みに失敗しました。

java.lang.SecurityException：
SecurityManager が存在し，SecurityManager の checkRead メソッドでファイルへの読み込みアクセスが拒否されました。

javax.naming.CommunicationException：
LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameAlreadyBoundException：
指定したユーザのシングルサインオン用認証情報がすでにあります。

その他 JNDI の例外：
バインド DN の指定ミスなどです。

戻り値

なし

addSSODataListener メソッド

説明

シングルサインオン用認証情報を追加，修正，または削除したとき，他システムに変更を通知するためのシングルサインオン用認証情報リスナをこのオブジェクトに登録します。

形式

```
public void addSSODataListener(SSODataListener listener);
```

パラメタ

listener：
SSODataListener オブジェクトを指定します。null を指定した場合は何もしません。

例外

なし

戻り値

なし

getSSOData メソッド

説明

シングルサインオン用認証情報を取得します。

形式

```
public SSOData getSSOData(String uid)
  throws NamingException;
```

パラメタ

uid：
　ユーザ ID を指定します。

例外

javax.naming.CommunicationException：
　LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameNotFoundException：
　指定したユーザ ID がありません。

その他 JNDI の例外：
　バインド DN の指定ミスなどです。

戻り値

シングルサインオン用認証情報を格納した SSOData オブジェクトを返却します。

getSSODataListeners メソッド

説明

このオブジェクトに登録されている SSODataListener オブジェクトの配列を取得します。登録されていない場合は大きさ 0 の配列を返却します。

形式

```
public SSODataListener[] getSSODataListeners();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトに登録されている SSODataListener オブジェクトの配列を返却します。

listUsers メソッド（形式 1）

説明

すべてのユーザ ID の一覧を取得します。addSSOData メソッドまたは removeSSOData メソッドを呼び出した場合，以前に返却された LdapUserEnumeration オブジェクトにその結果が反映されるかどうかは不定です。

形式

```
public LdapUserEnumeration listUsers()
  throws NamingException;
```

パラメタ

なし

例外

javax.naming.CommunicationException：
　LDAP ディレクトリサーバへの接続に失敗しました。

その他 JNDI の例外：
　バインド DN の指定ミスなどです。

戻り値

ユーザ ID の一覧を格納した LdapUserEnumeration オブジェクトを返却します。

listUsers メソッド（形式 2）

説明

ユーザ ID の一覧を取得します。addSSOData メソッドまたは removeSSOData メソッドを呼び出した場合，以前に返却された LdapUserEnumeration オブジェクトにその結果が反映されるかどうかは不定です。

形式

```
public LdapUserEnumeration listUsers(String uid)
  throws NamingException;
```

パラメタ

uid：

ユーザ ID を指定します。ユーザ ID にはワイルドカード（*）を含めることができます。このパラメタを省略した場合や null を指定した場合は，すべてのユーザ ID の一覧を取得します。

例外

javax.naming.CommunicationException：

LDAP ディレクトリサーバへの接続に失敗しました。

その他 JNDI の例外：

バインド DN の指定ミスなどです。

戻り値

ユーザ ID の一覧を格納した LdapUserEnumeration オブジェクトを返却します。

modifySSOData メソッド

説明

シングルサインオン用認証情報を修正します。指定したユーザがない場合は例外が発生します。

このオブジェクトに登録されているすべてのシングルサインオン用認証情報リスナの ssoDataModified メソッドが呼び出されます。

このメソッドでは，SSOData オブジェクトの生成後，変更された認証情報だけが既存のものに上書きされます。

例えば，リポジトリの既存のシングルサインオン用認証情報が次に示す要素を持っていたとします。

認証情報名	SecretData	PublicData	マッピング	
			レルム	ユーザ ID
値	secret	public	RealmA	user1
			RealmB	admin

このとき，次のコードで生成した SSOData オブジェクトをこのメソッドのパラメタに指定します。

```
SSOData data = new SSOData();
```

```
data.setMapping("RealmA", "user2");
```

すると，リポジトリのシングルサインオン用認証情報は次のように変更されます。

認証情報名	SecretData	PublicData	マッピング	
			レルム	ユーザ ID
値	secret	public	RealmA	user2
	―	―	―	―

（凡例）―：情報がないことを示します。

形式

```
public void modifySSOData(String uid,
                          SSOData ssoData)
  throws SSODataListenerException, NamingException,
  CryptoException, UnsatisfiedLinkError, SecurityException;
```

パラメタ

uid：

ユーザ ID を指定します。

ssoData：

シングルサインオン用認証情報を格納した SSOData オブジェクトを指定します。

例外

com.cosminexus.admin.auth.api.repository.event.SSODataListenerException：

他システムの認証情報更新に失敗しました。

com.cosminexus.admin.auth.CryptoException：

暗号鍵ファイルの読み込みに失敗しました。または，誤った暗号鍵ファイルを使用したため SecretData の復号化に失敗しました。

java.lang.UnsatisfiedLinkError：

シングルサインオンライブラリの読み込みに失敗しました。

java.lang.SecurityException：

SecurityManager が存在し，SecurityManager の checkRead メソッドでファイルへの読み込みアクセスが拒否されました。

javax.naming.CommunicationException：

LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameNotFoundException：

指定したユーザ ID がありません。

その他 JNDI の例外：
　バインド DN の指定ミスなどです。

戻り値

なし

removeSSOData メソッド

説明

シングルサインオン用認証情報を削除します。指定したユーザがない場合は例外が発生します。

このオブジェクトに登録されているすべてのシングルサインオン用認証情報リスナの ssoDataRemoved メソッドが呼び出されます。

形式

```
public void removeSSOData(String uid)
  throws SSODataListenerException, NamingException,
  CryptoException, UnsatisfiedLinkError, SecurityException;
```

パラメタ

uid：
　ユーザ ID を指定します。

例外

com.cosminexus.admin.auth.api.repository.event.SSODataListenerException：
　他システムの認証情報更新に失敗しました。

com.cosminexus.admin.auth.CryptoException：
　暗号鍵ファイルの読み込みに失敗しました。または誤った暗号鍵ファイルを使用したため SecretData の復号化に失敗しました。

java.lang.UnsatisfiedLinkError：
　シングルサインオンライブラリの読み込みに失敗しました。

java.lang.SecurityException：
　SecurityManager が存在し，SecurityManager の checkRead メソッドでファイルへの読み込みアクセスが拒否されました。

javax.naming.CommunicationException：
　LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameNotFoundException：

指定したユーザ ID がありません。

その他 JNDI の例外：
バインド DN の指定ミスなどです。

戻り値

なし

removeSSODataListener メソッド

説明

指定した SSODataListener オブジェクトをこのオブジェクトから削除します。指定したオブジェクトが登録されていない場合は何もしません。

形式

```
public void removeSSODataListener(SSODataListener listener);
```

パラメタ

listener：
SSODataListener オブジェクトを指定します。

例外

なし

戻り値

なし

8.6 LdapUserDataManager クラス

説明

LDAP ディレクトリサーバのユーザ情報リポジトリに格納されている情報を参照または更新するクラスです。
このクラスのオブジェクトごとに，addUserData メソッド，modifyUserData メソッド，removeUserData メソッド，および getUserData メソッドで排他制御をします。異なるオブジェクトで同時に同じリポジトリを操作しないでください。
LdapUserDataManager クラスのパッケージ名は，
com.cosminexus.admin.auth.api.repository.ldap です。

形式

```
class LdapUserDataManager
{
  public LdapUserDataManager(String name)
    throws ConfigError;
  public LdapUserDataManager(String name,
                             AttributeEntry[] aliases)
    throws ConfigError, FormatError;
  public LdapUserDataManager(String name,
                             String aliasesFile)
    throws ConfigError, FormatError, IOException,
           FileNotFoundException,
    SecurityException;
  public LdapUserDataManager(String name,
                             AttributeEntry[] aliases,
                             ObjectClassEntry[] ocEntries)
    throws ConfigError, FormatError;
  public LdapUserDataManager(String name,
                             AttributeEntry[] aliases,
                             String objclassesFile)
    throws ConfigError, FormatError, IOException,
           FileNotFoundException,SecurityException;
  public LdapUserDataManager(String name,
                             String aliasesFile,
                             ObjectClassEntry[] ocEntries)
    throws ConfigError, FormatError, IOException,
           FileNotFoundException,SecurityException;
  public LdapUserDataManager(String name,
                             String aliasesFile,
                             String objclassesFile)
    throws ConfigError, FormatError, IOException,
           FileNotFoundException,SecurityException;

  public LdapUserEnumeration listUsers()
    throws NamingException;
  public LdapUserEnumeration listUsers(String uid)
    throws NamingException;
  public UserData getUserData(String uid)
    throws NamingException;
  public void addUserData(String uid,
                          UserData userData)
    throws ObjectClassError, NamingException;
  public void addUserData(String uid,
                          UserData userData,
                          String name, String value)
    throws ObjectClassError, NamingException;
```

```
    public void removeUserData(String uid)
      throws NamingException;
    public void modifyUserData(String uid, UserData userData)
      throws ObjectClassError, NamingException;
  }
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
LdapUserDataManager コンストラクタ	LdapUserDataManager クラスのインスタンスを生成します。
addUserData メソッド（形式 1）	ユーザを追加します。ユーザエントリの DN に uid を使用します。
addUserData メソッド（形式 2）	ユーザを追加します。ユーザエントリの DN に任意の属性を使用します。
getUserData メソッド	ユーザ情報を取得します。
listUsers メソッド（形式 1）	すべてのユーザ ID の一覧を取得します。
listUsers メソッド（形式 2）	ユーザ ID の一覧を取得します。
modifyUserData メソッド	ユーザ情報を修正します。
removeUserData メソッド	ユーザを削除します。

LdapUserDataManager コンストラクタ

説明

LdapUserDataManager クラスのインスタンスを生成します。ユーザ属性情報やオブジェクトクラスは，オブジェクトやファイルで指定できます。また，省略もできます。

形式

```
public LdapUserDataManager(String name)
  throws ConfigError;

public LdapUserDataManager(String name,
                           AttributeEntry[] aliases)
  throws ConfigError, FormatError;

public LdapUserDataManager(String name,
                           String aliasesFile)
  throws ConfigError, FormatError, IOException,
FileNotFoundException,
  SecurityException;

public LdapUserDataManager(String name,
                           AttributeEntry[] aliases,
                           ObjectClassEntry[] ocEntries)
  throws ConfigError, FormatError;

public LdapUserDataManager(String name,
                           AttributeEntry[] aliases,
                           String objclassesFile)
  throws ConfigError, FormatError, IOException,
```

```
FileNotFoundException,
  SecurityException;

public LdapUserDataManager(String name,
                           String aliasesFile,
                           ObjectClassEntry[] ocEntries)
  throws ConfigError, FormatError, IOException,
FileNotFoundException,
  SecurityException;

public LdapUserDataManager(String name,
                           String aliasesFile,
                           String objclassesFile)
  throws ConfigError, FormatError, IOException,
FileNotFoundException,
  SecurityException;
```

パラメタ

name：

アクセス対象にする LDAP ディレクトリサーバの設定名を指定します。設定名はユーザ管理のコンフィグレーションファイルで定義します。

aliases：

参照または更新するユーザ属性情報として，AttributeEntry オブジェクトの配列を指定します。指定した内容で必要な情報がない場合は FormatError 例外が発生します。このパラメタを省略した場合や null を指定した場合，属性を参照および更新できません。ただし，パスワードは更新できます。

aliasesFile：

参照または更新するユーザ属性情報として，ファイル名を指定します。指定した内容で必要な情報がない場合は FormatError 例外が発生します。このパラメタを省略した場合や null を指定した場合，属性を参照および更新できません。ただし，パスワードは更新できます。

ocEntries：

LDAP ディレクトリサーバにエントリを作成したり，修正したりするときに使用するオブジェクトクラスの配列を指定します。指定した内容で必要な情報がない場合は FormatError 例外が発生します。このパラメタを省略した場合や null を指定した場合，ユーザ情報の追加や変更時に ObjectClassError 例外が発生します。

objclassesFile：

LDAP ディレクトリサーバのエントリのオブジェクトクラスが定義されたファイル名を指定します。指定した内容で必要な情報がない場合は FormatError 例外が発生します。このパラメタを省略した場合や null を指定した場合，ユーザ情報の追加や変更時に ObjectClassError 例外が発生します。

例外

java.io.FileNotFoundException：

ファイルがない，ファイルではなくディレクトリである，またはそれ以外の何かの理由で開くことができません（FileInputStream クラスのコンストラクタで例外が呼び出された場合）。

java.lang.SecurityException：
SecurityManager が存在し，SecurityManager の checkRead メソッドでファイルへの読み込みアクセスが拒否されました。

java.io.IOException：
ファイルの読み込みに失敗しました。

com.cosminexus.admin.common.ConfigError：
設定名が統合ユーザ管理のコンフィグレーションファイルにありません。

com.cosminexus.admin.common.FormatError：
aliases，aliasesFile，ocEntries および objclassesFile に指定された内容で必要な情報がありません。または余分に指定されています。

addUserData メソッド（形式 1）

説明

ユーザを追加します。すでにユーザがいる場合は例外が発生します。

LDAP ディレクトリサーバに作成するユーザエントリの DN には，ユーザ ID の属性（uid）と値が使用されます。

ユーザエントリはベース DN の直下に作成されます。また，コンストラクタで指定したユーザ属性情報にサブコンテキストの属性が含まれる場合，サブコンテキストのエントリも作成されます。

このメソッドを呼び出した結果，サブコンテキストの更新中に例外が発生した場合，ユーザ情報は不完全に更新された状態になっています。その場合，原因を取り除いてから removeUserData メソッドで該当ユーザを削除し，再びこのメソッドを呼び出してください。

形式

```
public void addUserData(String uid,
                        UserData userData)
  throws ObjectClassError, NamingException;
```

パラメタ

uid：
ユーザ ID を指定します。

userData：

ユーザ情報を格納した UserData オブジェクトを指定します。

例外

com.cosminexus.admin.auth.api.repository.ldap.ObjectClassError：
LDAP ディレクトリサーバにエントリを作成するために必要なオブジェクトクラスが指定されていません。

javax.naming.CommunicationException：
LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameAlreadyBoundException：
指定したユーザ ID がすでにあります。

その他 JNDI の例外：
バインド DN の指定ミスなどです。

戻り値

なし

注意事項

getUserData メソッドで取得した UserData オブジェクトにパスワードは格納されていません。そのため，getUserData メソッドで取得した UserData オブジェクトをそのまま addUserData メソッドのパラメタに指定しても，ユーザの完全なコピーをすることはできません。パスワードを新たに設定してください。

addUserData メソッド（形式 2）

説明

ユーザを追加します。すでにユーザがいる場合は例外が発生します。

LDAP ディレクトリサーバに作成するユーザエントリの DN には，このメソッドで指定した属性名と値が使用されます。

ユーザエントリはベース DN の直下に作成されます。また，コンストラクタで指定したユーザ属性情報にサブコンテキストの属性が含まれる場合，サブコンテキストのエントリも作成されます。

このメソッドを呼び出した結果，サブコンテキストの更新中に例外が発生した場合，ユーザ情報は不完全に更新された状態になっています。その場合，原因を取り除いてから removeUserData メソッドで該当ユーザを削除し，再びこのメソッドを呼び出してください。

形式

```
public void addUserData(String uid,
                        UserData userData,
                        String name,
                        String value)
  throws ObjectClassError, NamingException;
```

パラメタ

uid：
　ユーザ ID を指定します。

userData：
　ユーザ情報を格納した UserData オブジェクトを指定します。

name：
　ユーザエントリの DN に使用する属性名を指定します。

value：
　ユーザエントリの DN に使用する属性値を指定します。

例外

com.cosminexus.admin.auth.api.repository.ldap.ObjectClassError：
　LDAP ディレクトリサーバにエントリを作成するために必要なオブジェクトクラスが指定されていません。

javax.naming.CommunicationException：
　LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameAlreadyBoundException：
　指定したユーザ ID がすでにあります。

その他 JNDI の例外：
　バインド DN の指定ミスなどです。

戻り値

なし

注意事項

getUserData メソッドで取得した UserData オブジェクトにパスワードは格納されていません。そのため，getUserData メソッドで取得した UserData オブジェクトをそのまま addUserData メソッドのパラメタに指定しても，ユーザの完全なコピーをすることはできません。パスワードを新たに設定してください。

getUserData メソッド

説明

ユーザ情報を取得します。取得した UserData オブジェクトにパスワードは格納されていません。

形式

```
public UserData getUserData(String uid)
  throws NamingException;
```

パラメタ

uid：
　ユーザ ID を指定します。

例外

javax.naming.CommunicationException：
　LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameNotFoundException：
　指定したユーザ ID がありません。

その他 JNDI の例外：
　バインド DN の指定ミスなどです。

戻り値

ユーザ情報を格納した UserData オブジェクトを返却します。

listUsers メソッド（形式 1）

説明

すべてのユーザ ID の一覧を取得します。addUserData メソッドまたは removeUserData メソッドを呼び出した場合，以前に返却された LdapUserEnumeration オブジェクトにその結果が反映されるかどうかは不定です。

形式

```
public LdapUserEnumeration listUsers()
  throws NamingException;
```

パラメタ

なし

例外

javax.naming.CommunicationException：
LDAP ディレクトリサーバへの接続に失敗しました。

その他 JNDI の例外：
バインド DN の指定ミスなどです。

戻り値

ユーザ ID の一覧を格納した LdapUserEnumeration オブジェクトを返却します。

listUsers メソッド（形式 2）

説明

ユーザ ID の一覧を取得します。addUserData メソッドまたは removeUserData メソッドを呼び出した場合，以前に返却された LdapUserEnumeration オブジェクトにその結果が反映されるかどうかは不定です。

形式

```
public LdapUserEnumeration listUsers(String uid)
  throws NamingException;
```

パラメタ

uid：
ユーザ ID を指定します。ユーザ ID にはワイルドカード（*）を含めることができます。このパラメタを省略した場合や null を指定した場合は，すべてのユーザ ID の一覧を取得します。

例外

javax.naming.CommunicationException：
LDAP ディレクトリサーバへの接続に失敗しました。

その他 JNDI の例外：
バインド DN の指定ミスなどです。

戻り値

ユーザ ID の一覧を格納した LdapUserEnumeration オブジェクトを返却します。

modifyUserData メソッド

説明

ユーザ情報を修正します。指定したユーザがない場合は例外が発生します。

このメソッドでは，UserData オブジェクトの生成後，変更された属性だけが既存の属性に上書きされます。

例えば，リポジトリの既存のユーザ情報が次に示す属性を持っていたとします。

属性名	full name	tel
値	Hitachi Taro	111-1111
		222-2222

このとき，次のコードで生成した UserData オブジェクトをこのメソッドのパラメタに指定します。

```
UserData data = new UserData();
data.addAttribute("tel", "111-2222");
```

すると，リポジトリのユーザ情報は次のように変更されます。

属性名	full name	tel
値	Hitachi Taro	111-2222
		－

（凡例）－：情報がないことを示します。

このメソッドを呼び出した結果，サブコンテキストの更新中に例外が発生した場合，ユーザ情報は不完全に更新された状態になっています。その場合，原因を取り除いてから再びこのメソッドを呼び出してください。

形式

```
public void modifyUserData(String uid,
                           UserData userData)
  throws ObjectClassError, NamingException;
```

パラメタ

uid：

ユーザ ID を指定します。

userData：

ユーザ情報を格納した UserData オブジェクトを指定します。

例外：

com.cosminexus.admin.auth.api.repository.ldap.ObjectClassError：
LDAP ディレクトリサーバにエントリを作成するために必要なオブジェクトクラスが指定されていません。

javax.naming.CommunicationException：
LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameNotFoundException：
指定したユーザ ID がありません。

その他 JNDI の例外：
バインド DN の指定ミスなどです。

戻り値

なし

removeUserData メソッド

説明

ユーザを削除します。指定したユーザがない場合は例外が発生します。LDAP ディレクトリサーバの，指定したユーザエントリ以下のすべてのエントリが削除されます。

このメソッドを呼び出した結果，サブコンテキストの更新中に例外が発生した場合，ユーザ情報は不完全に更新された状態になっています。その場合，原因を取り除いてから再びこのメソッドを呼び出してください。

形式

```
public void removeUserData(String uid)
  throws NamingException;
```

パラメタ

uid：
ユーザ ID を指定します。

例外

javax.naming.CommunicationException：
LDAP ディレクトリサーバへの接続に失敗しました。

javax.naming.NameNotFoundException：
指定したユーザ ID がありません。

その他 JNDI の例外：
　　バインド DN の指定ミスなどです。

戻り値

なし

8.7　LdapUserEnumeration インタフェース

説明

ユーザ ID の一覧を参照するためのインタフェースです。
LdapUserEnumeration インタフェースのパッケージ名は，
com.cosminexus.admin.auth.api.repository.ldap です。

形式

```
interface LdapUserEnumeration extends java.util.Enumeration
{
  public boolean hasMore()
    throws NamingException;
  public boolean hasMoreElements();
  public String next()
    throws NamingException;
  public Object nextElement();
  public close()
    throws NamingException;
}
```

メソッド一覧

メソッド名	機能
close メソッド	オブジェクトをクローズします。
hasMore メソッド	一覧に次のユーザ ID があるかどうかを判定します。 （NamingException 例外の呼び出し：あり）
hasMoreElements メソッド	一覧に次のユーザ ID があるかどうかを判定します。 （NamingException 例外の呼び出し：なし）
next メソッド	一覧の次のユーザ ID を取得します。 （NamingException 例外の呼び出し：あり 戻り値の型：String 型）
nextElement メソッド	一覧の次のユーザ ID を取得します。 （NamingException 例外の呼び出し：なし 戻り値の型：Object 型）

close メソッド

説明

このオブジェクトをクローズして，使用中のリソースを解放します。false を返すまで hasMore メソッドや hasMoreElements メソッドを呼び出し続けた場合は，このメソッドを呼び出す必要はありません。

形式

```
public void close()
  throws NamingException;
```

パラメタ

なし

例外

javax.naming.NamingException：
　クローズ中に NamingException 例外が発生しました。

戻り値

なし

hasMore メソッド

説明

一覧に次のユーザ ID があるかどうかを判定します。

形式

```
public boolean hasMore()
  throws NamingException;
```

パラメタ

なし

例外

javax.naming.NamingException：
　次のユーザ ID があるかどうかを判定中に NamingException 例外が発生しました。

戻り値

true：
　次のユーザ ID がありました。

false：
　次のユーザ ID がありませんでした。

hasMoreElements メソッド

説明

一覧に次のユーザ ID があるかどうかを判定します。例外が発生した場合は false を返却します。

形式

```
public boolean hasMoreElements();
```

パラメタ

なし

例外

なし

戻り値

true：
　次のユーザ ID がありました。

false：
　次のユーザ ID がありませんでした。

next メソッド

説明

一覧の次のユーザ ID を取得します。

形式

```
public String next()
  throws NamingException;
```

パラメタ

なし

例外

java.util.NoSuchElementException：
　次のユーザ ID がないときにこのメソッドを呼び出しました。

javax.naming.NamingException：
　次のユーザ ID を取得中に NamingException 例外が発生しました。

戻り値

次のユーザ ID を返却します。

nextElement メソッド

説明

一覧の次のユーザ ID を取得します。next メソッドとの違いは，NamingException 例外が発生しないことと，戻り値の型は Object 型であることです。戻り値の Object オブジェクトを String にキャストして参照してください。このメソッドを実行中に NamingException 例外が発生した場合は null を返却します。

形式

```
public Object nextElement();
```

パラメタ

なし

例外

java.util.NoSuchElementException：
　次のユーザ ID がないときにこのメソッドを呼び出しました。

戻り値

次のユーザ ID を返却します。

8.8　LoginUtil クラス

説明

統合ユーザ管理のセッション内でログインしているユーザの有無を調べます。
LoginUtil クラスのパッケージ名は，com.cosminexus.admin.auth.util です。

形式

```
class LoginUtil
{
  public static boolean check(HttpServletRequest request,
                              HttpServletResponse response);
  public static boolean check(HttpServletRequest request,
                               HttpServletResponse response,
                               String realmName);
}
```

メソッド一覧

メソッド名	機能
check メソッド（形式 1）	セッション内でログインしているユーザの有無を調べます。
check メソッド（形式 2）	セッション内でログインしているユーザの有無を調べます。特定のレルム内でログインしているユーザを調べるために使用します。

注意事項

このクラスの check メソッドを使わなくても，HttpSession にログイン時に生成された Subject を関連づけ，Subject の Principal の有無によってログインの有無を判断できます。この方式で確認する場合は，統合ユーザ管理の機能を使ってセッションを停止しないでください。

check メソッド（形式 1）

説明

セッション内でログインしているユーザの有無を調べます。このセッション内でどれかのレルムでログインしているユーザを検索し，一人でもログインしているユーザが見つかれば true を返却します。

形式

```
public static boolean check(HttpServletRequest request,
                            HttpServletResponse response);
```

パラメタ

request：

JSP/Servlet に渡された HttpServletRequest のリファレンスを指定します。null を指定した場合は NullPointerException 例外が発生します。

response：
JSP/Servlet に渡された HttpServletResponse のリファレンスを指定します。null を指定した場合は NullPointerException 例外が発生します。

例外

java.lang.NullPointerException：
このメソッドのパラメタに null を指定しました。

戻り値

true：
ログインしているユーザが見つかりました。

false：
ログインしているユーザが見つかりませんでした。

check メソッド（形式 2）

説明

セッション内でログインしているユーザの有無を調べます。realmName は，特定のレルム内でログインしているユーザを調べるために使用します。

形式

```
public static boolean check(HttpServletRequest request,
                            HttpServletResponse response,
                            String realmName);
```

パラメタ

request：
JSP/Servlet に渡された HttpServletRequest のリファレンスを指定します。null を指定した場合は NullPointerException 例外が発生します。

response：
JSP/Servlet に渡された HttpServletResponse のリファレンスを指定します。null を指定した場合は NullPointerException 例外が発生します。

realmName：
特定のレルム内でログインしているユーザを調べる場合に指定します。null を指定した場合は NullPointerException 例外が発生します。

例外

java.lang.NullPointerException：

このメソッドのパラメタに null を指定しました。

戻り値

true：

ログインしているユーザが見つかりました。

false：

ログインしているユーザが見つかりませんでした。

8.9 ObjectClassEntry クラス

説明

LDAP ディレクトリサーバに作成するユーザエントリやサブコンテキストのオブジェクトクラスを格納するクラスです。
ObjectClassEntry クラスのパッケージ名は,
com.cosminexus.admin.auth.api.repository.ldap です。

形式

```
class ObjectClassEntry
{
public ObjectClassEntry();
public ObjectClassEntry(String[] objectClasses);
public ObjectClassEntry(String subcontext,
                        String[] objectClasses);

public void setObjectClasses(String[] objectClasses);
public String[] getObjectClasses();
public void setSubcontext(String subcontext);
public String getSubcontext();
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
ObjectClassEntry コンストラクタ	ObjectClassEntry クラスのインスタンスを生成します。
getObjectClasses メソッド	setObjectClasses メソッドまたはコンストラクタで指定したオブジェクトクラスを取得します。
getSubcontext メソッド	setSubcontext メソッドまたはコンストラクタで指定したサブコンテキストを取得します。
setObjectClasses メソッド	オブジェクトクラスをオブジェクトに格納します。
setSubcontext メソッド	サブコンテキストをオブジェクトに格納します。

ObjectClassEntry コンストラクタ

説明

インスタンスを生成します。パラメタにオブジェクトクラスを指定すると，ユーザエントリのオブジェクトクラスとしてこのオブジェクトに格納されます。

形式

```
public ObjectClassEntry();

public ObjectClassEntry(String[] objectClasses);

public ObjectClassEntry(String subcontext,
                        String[] objectClasses);
```

パラメタ

subcontext：

サブコンテキストを指定します。このパラメタを省略した場合や，null や空文字 ("") を指定した場合は，ユーザエントリを表します。AttributeEntry オブジェクトに指定するサブコンテキストと同じ文字列を指定してください。

objectClasses：

ユーザエントリのオブジェクトクラスを String の配列で指定します。このパラメタを省略した場合や null を指定した場合は，何も格納されません。

例外

なし

getObjectClasses メソッド

説明

setObjectClasses メソッドまたはコンストラクタで指定した値を取得します。値が格納されていない場合に getObjectClasses メソッドを呼び出したときは null が返却されます。

形式

```
public String[] getObjectClasses();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトに格納されている値を返却します。

getSubcontext メソッド

説明

setSubcontext メソッドまたはコンストラクタで指定した値を取得します。値が格納されていない場合に getSubcontext メソッドを呼び出したときは null が返却されます。

形式

```
public String getSubcontext();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトに格納されている値を返却します。

setObjectClasses メソッド

説明

オブジェクトクラスをこのオブジェクト内に格納します。すでに値が格納されている場合に setObjectClasses メソッドを呼び出したときは上書きされます。

形式

```
public void setObjectClasses(String[] objectClasses);
```

パラメタ

objectClasses：

オブジェクトクラスを String の配列で指定します。

例外

なし

戻り値

なし

setSubcontext メソッド

説明

サブコンテキストをこのオブジェクト内に格納します。すでに値が格納されている場合に setSubcontext メソッドを呼び出したときは値が上書きされます。

形式

```
public void setSubcontext(String subcontext);
```

パラメタ

subcontext：

サブコンテキストを指定します。null や空文字 ("") を指定した場合はユーザエントリを表します。AttributeEntry オブジェクトに設定するサブコンテキストと同じ文字列を指定してください。

例外

なし

戻り値

なし

8.10 PasswordCryptography インタフェース

説明

入力したパスワードを暗号化するためのインタフェースです。
PasswordCryptography インタフェースのパッケージ名は，
com.cosminexus.admin.auth.security です。

形式

```
interface PasswordCryptography
{
  public byte[] encrypt(byte[] plain);
}
```

メソッド一覧

メソッド名	機能
encrypt メソッド	リポジトリに登録されている暗号化形式に合わせてパスワードを暗号化します。

encrypt メソッド

説明

リポジトリに登録されている暗号化形式に合わせてパスワードを暗号化します。

形式

```
public byte[] encrypt(byte[] plain);
```

パラメタ

plain：

ログインモジュールがこのメソッドを呼び出すときに，ユーザが指定したパスワード（平文）が格納されます。

例外

なし

戻り値

暗号化した結果を返却します。

8.11　PasswordUtil クラス

説明

ユーザのパスワードを変更するためのクラスです。
PasswordUtil クラスのパッケージ名は，com.cosminexus.admin.auth.util です。

形式

```
class PasswordUtil
{
  public static void changePassword(String name,
                                    String uid,
                                    String oldPassword,
                                    String newPassword)
    throws LoginException,
           SecurityException;

}
```

メソッド一覧

メソッド名	機能
changePassword メソッド	パスワードを変更します。

changePassword メソッド

説明

パラメタで指定された name，uid および oldPassword で本人の確認をして，その結果に問題がなければ新しいパスワードに変更します。シングルサインオン用認証情報が登録されている場合は，シングルサインオン情報リポジトリの内容も変更されます。

このメソッドは static メソッドです。

形式

```
public static void changePassword(String name,
                                  String uid,
                                  String oldPassword,
                                  String newPassword)
    throws LoginException,
      SecurityException;
```

パラメタ：

name：

認証に使用するログインモジュール（LoginContext）のアプリケーション名（name）を指定します。

uid：

変更するユーザのユーザ ID を指定します。

oldPassword：
　変更前のパスワードを指定します。

newPassword：
　変更後のパスワードを指定します。

例外

javax.security.auth.login.LoginException：
　認証に必要な情報がありません。または，ユーザ ID ／パスワードが誤っています。

java.lang.SecurityException：
　アクセス権がありません。

戻り値

なし

注意事項

- シングルサインオン用認証情報は，レルム名，暗号鍵ファイル，およびシングルサインオン情報リポジトリにアクセスする情報が定義されている場合だけ変更します。それ以外の状態（シングルサインオン用の定義がない）の場合は変更しません。
- シングルサインオン用認証情報が登録されている場合に，リポジトリで例外が発生したり（NamingException 例外が発生），暗号化に失敗したりしたとき，このメソッドは LoginException 例外で失敗します。このとき，パスワードは元の状態に戻します（ロールバックします）。また，パスワードを元に戻すのに失敗した場合も LoginException 例外が発生します。例外クラスの詳細については，「8.32　例外クラス」を参照してください。
- name で指定したアプリケーションで WebPasswordLoginModule または WebPasswordLDAPLoginModule を使用していない場合，LoginException 例外が発生します。
- LDAP ディレクトリサーバが Active Directory の場合は，WebPasswordLDAPLoginModule を使用しているアプリケーションを name に指定してください。

8.12　Principal インタフェース

説明

WebPasswordLoginModule が認証したときのユーザ ID を参照するときに使用します。日立の実装クラスでは，java.security.Principal インタフェースを継承して作成されたものを認証した Subject に関連づけます。そのため，Principal を参照する場合は，Subject から Principal を求めて getName メソッドで参照してください。
Principal インタフェースのパッケージ名は，java.security です。

8.13 SSOData クラス

説明

シングルサインオン用認証情報を格納するクラスです。
SSOData クラスのパッケージ名は,
com.cosminexus.admin.auth.api.repository.ldap です。

形式

```
class SSOData
{
  public SSOData();

  public void setSecretData(String secretData)
    throws CryptoException, UnsatisfiedLinkError,
SecurityException;
  public void setPublicData(String publicData);
  public String getPublicData();
  public Enumeration getMappingRealms();
  public String getMapping(String realm);
  public void setMapping(String realm,
                         String uid);
  public void removeMapping(String realm);
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
SSOData コンストラクタ	SSOData クラスのインスタンスを生成します。
getMapping メソッド	レルム名に対応するユーザ ID を取得します。
getMappingRealms メソッド	レルム名一覧を取得します。
getPublicData メソッド	PublicData を取得します。
removeMapping メソッド	指定したレルムを削除します。
setMapping メソッド	接続先レルム名とユーザ ID を格納します。
setPublicData メソッド	PublicData を格納します。
setSecretData メソッド	SecretData を格納します。

SSOData コンストラクタ

説明

SSOData クラスのインスタンスを生成します。

形式

```
public SSOData();
```

パラメタ

なし

例外

なし

getMapping メソッド

説明

このオブジェクトに格納されているマッピング情報から，レルム名に対応するユーザ ID を取得します。指定したレルム名に対応するユーザ ID がない場合は null が返却されます。

形式

```
public String getMapping(String realm);
```

パラメタ

realm：
　接続先のレルム名を指定します。

例外

なし

戻り値

ユーザ ID を返却します。

getMappingRealms メソッド

説明

このオブジェクトに格納されているマッピング情報から，レルム名一覧を取得します。

レルム名を参照する場合は，まず，このメソッドで取得した Enumeration オブジェクトに対して nextElement メソッドを実行して，Object オブジェクトを取得します。取得した Object オブジェクトを String にキャストして，値を参照してください。

形式

```
public Enumeration getMappingRealms();
```

パラメタ

なし

例外

なし

戻り値

レルム名一覧を格納した Enumeration オブジェクトを返却します。

getPublicData メソッド

説明

このオブジェクトに格納されている PublicData を取得します。値が格納されていない場合に getPublicData メソッドを呼び出したときは null が返却されます。

形式

```
public String getPublicData();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトに格納されている値を返却します。

removeMapping メソッド

説明

このオブジェクトに格納されているマッピング情報から，指定したレルムを削除します。指定したレルム名が格納されていない場合は何もしません。

形式

```
public void removeMapping(String realm);
```

パラメタ

realm：
　接続先のレルム名を指定します。

例外

なし

戻り値

なし

setMapping メソッド

説明

パラメタに指定された接続先レルム名とユーザ ID をこのオブジェクト内に格納します。すでに同じレルム名でユーザ ID が格納されている場合は上書きされます。

形式

```
public void setMapping(String realm,
                       String uid);
```

パラメタ

realm：
　接続先のレルム名を指定します。

uid：
　接続先レルムのユーザ ID を指定します。

例外

なし

戻り値

なし

setPublicData メソッド

説明

パラメタに指定された PublicData をこのオブジェクト内に格納します。すでに値が格納されている場合は上書きされます。

形式

```
public void setPublicData(String publicData);
```

パラメタ

publicData：
　PublicData を指定します。

例外

なし

戻り値

なし

setSecretData メソッド

説明

このオブジェクトに SecretData を格納します。格納するとき，SecretData は暗号化されます。すでに SecretData が格納されている場合は上書きされます。

形式

```
public void setSecretData(String secretData);
```

パラメタ

secretData：
　SecretData を指定します。

例外

com.cosminexus.admin.auth.CryptoException：
　暗号鍵ファイルを読み込めないため，SecretData の暗号化に失敗しました。

java.lang.UnsatisfiedLinkError：
　シングルサインオンライブラリの読み込みに失敗しました。

java.lang.SecurityException：
　SecurityManager が存在し，SecurityManager の checkRead メソッドでファイルへの読み込みアクセスが拒否されました。

戻り値

なし

8.14　SSODataEvent クラス

説明

シングルサインオン用認証情報の更新内容を格納するクラスです。
SSODataEvent クラスのパッケージ名は，
com.cosminexus.admin.auth.api.repository.event です。

形式

```
class SSODataEvent
{
  public SSODataEvent(String uid,
                      String secretData,
                      String publicData,
                      String oldSecretData,
                      String oldPublicData);

  public String getUserId();
  public String getSecretData();
  public String getPublicData();
  public String getOldSecretData();
  public String getOldPublicData();
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
SSODataEvent コンストラクタ	SSODataEvent クラスのインスタンスを生成します。
getOldPublicData メソッド	変更前の PublicData を取得します。
getOldSecretData メソッド	変更前の SecretData を取得します。
getPublicData メソッド	PublicData を取得します。
getSecretData メソッド	SecretData を取得します。
getUserId メソッド	ユーザ ID を取得します。

SSODataEvent コンストラクタ

説明

インスタンスを生成し，パラメタに指定されたユーザ ID，SecretData，PublicData，変更前の SecretData，および変更前の PublicData を格納します。

形式

```
public SSODataEvent(String uid,
                    String secretData,
                    String publicData,
                    String oldSecretData,
                    String oldPublicData);
```

パラメタ

uid：
　ユーザ ID を指定します。

secretData：
　SecretData を指定します。

publicData：
　PublicData を指定します。

oldSecretData：
　変更前の SecretData を指定します。

oldPublicData：
　変更前の PublicData を指定します。

例外

なし

getOldPublicData メソッド

説明

このオブジェクトに格納されている変更前の PublicData を取得します。

形式

```
public String getOldPublicData();
```

パラメタ

なし

例外

なし

戻り値

変更前の PublicData を返却します。設定されていない場合は null を返却します。

getOldSecretData メソッド

説明

このオブジェクトに格納されている変更前の SecretData を取得します。

形式

```
public String getOldSecretData();
```

パラメタ

なし

例外

なし

戻り値

変更前の SecretData を返却します。設定されていない場合は null を返却します。

getPublicData メソッド

説明

このオブジェクトに格納されている PublicData を取得します。

形式

```
public String getPublicData();
```

パラメタ

なし

例外

なし

戻り値

PublicData を返却します。設定されていない場合は null を返却します。

getSecretData メソッド

説明

このオブジェクトに格納されている SecretData を取得します。

形式

```
public String getSecretData();
```

パラメタ

なし

例外

なし

戻り値

SecretData を返却します。設定されていない場合は null を返却します。

getUserId メソッド

説明

このオブジェクトに格納されているユーザ ID を取得します。

形式

```
public String getUserId();
```

パラメタ

なし

例外

なし

戻り値

ユーザ ID を返却します。

8.15　SSODataListener インタフェース

説明

シングルサインオン用認証情報が更新されたときに，更新が通知されるシングルサインオン用認証情報リスナクラスが実装しなければならないインタフェースです。シングルサインオン用認証情報の更新に同期して他システムの認証情報を更新したい場合，このインタフェースを実装したクラスを作成してください。作成したクラスのインスタンス（オブジェクト）を LdapSSODataManager オブジェクトに addSSODataListener メソッドで登録してください。

SSODataListener インタフェースのパッケージ名は，com.cosminexus.admin.auth.api.repository.event です。

SSODataListener インタフェースのメソッドは LdapSSODataManager クラスのメソッドから呼び出されます。このとき，パラメタとして SSODataEvent オブジェクトが渡されます。

SSODataListener インタフェースの各メソッドが呼び出されるタイミング（呼び出す LdapSSODataManager クラスのメソッド）と，パラメタで渡される SSODataEvent オブジェクトに格納される値の内容を，次の表に示します。

表 8-2　SSODataEvent オブジェクトに格納される値

呼び出す LdapSSODataManager **クラスのメソッド**	**呼び出される** SSODataListener **インタフェースのメソッド**	SSODataEvent **オブジェクトに格納される値**				
		ユーザ ID	SecretData	PublicData	**変更前の** SecretData	**変更前の** PublicData
addSSOData メソッド	ssoDataAdded メソッド	○	○	○	－	－
modifySSOData メソッド	ssoDataModified メソッド	○	○	○	○	○
removeSSOData メソッド	ssoDataRemoved メソッド	○	○	○	－	－

（凡例）

○：格納されます。

－：格納されません。

ssoDataAdded メソッド，ssoDataModified メソッド，および ssoDataRemoved メソッドの各メソッドで問題が発生した場合，問題の原因がわかるメッセージが格納された ChangeDataFailedException 例外をスローするように，クラスを作成してください。LdapSSODataManager のメソッド呼び出し元には，それらの例外オブジェクトが格納された SSODataListenerException 例外が発生します。

形式

```
interface SSODataListener extends java.util.EventListener
{
  public void ssoDataAdded(SSODataEvent event)
    throws ChangeDataFailedException;
  public void ssoDataModified(SSODataEvent event)
    throws ChangeDataFailedException;
  public void ssoDataRemoved(SSODataEvent event)
    throws ChangeDataFailedException;
}
```

メソッド一覧

メソッド名	機能
ssoDataAdded メソッド	シングルサインオン用認証情報が追加されたときに呼び出されます。
ssoDataModified メソッド	シングルサインオン用認証情報が変更されたときに呼び出されます。
ssoDataRemoved メソッド	シングルサインオン情報が削除されたときに呼び出されます。

ssoDataAdded メソッド

説明

シングルサインオン用認証情報が追加されたときに呼び出されます。

形式

```
public void ssoDataAdded(SSODataEvent event)
  throws ChangeDataFailedException;
```

パラメタ

event：

シングルサインオン用認証情報が格納されます。

例外

com.cosminexus.admin.auth.api.repository.event.ChangeDataFailedException：

他システムの認証情報の更新に失敗しました。

戻り値

なし

ssoDataModified メソッド

説明

シングルサインオン用認証情報が変更されたときに呼び出されます。

形式

```
public void ssoDataModified(SSODataEvent event)
  throws ChangeDataFailedException;
```

パラメタ

event：

シングルサインオン用認証情報が格納されます。

例外

com.cosminexus.admin.auth.api.repository.event.ChangeDataFailedException：

他システムの認証情報の更新に失敗しました。

戻り値

なし

ssoDataRemoved メソッド

説明

シングルサインオン情報が削除されたときに呼び出されます。

形式

```
public void ssoDataRemoved(SSODataEvent event)
  throws ChangeDataFailedException;
```

パラメタ

event：

シングルサインオン用認証情報が格納されます。

例外

com.cosminexus.admin.auth.api.repository.event.ChangeDataFailedException：

他システムの認証情報の更新に失敗しました。

戻り値

なし

8.16 SSODataListenerException クラス

説明

シングルサインオン用認証情報リスナクラスで例外が発生した場合に呼び出される例外クラスです。

SSODataListenerException クラスのパッケージ名は,

com.cosminexus.admin.auth.api.repository.event です。

形式

```
class SSODataListenerException extends UAException
{
  public SSODataListenerException();
  public SSODataListenerException(String msg);

  public void setException(SSODataListener listener,
                           ChangeDataFailedException exception);
  public SSODataListener[] getListeners();
  public ChangeDataFailedException getException(SSODataListener
listener);
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
SSODataListenerException コンストラクタ	SSODataListenerException クラスのインスタンスを生成します。
getException メソッド	例外オブジェクトを取得します。
getListeners メソッド	例外に格納されているリスナを取得します。
setException メソッド	例外オブジェクトを格納します。

SSODataListenerException コンストラクタ

説明

パラメタに指定されたエラーメッセージを使用して ,SSODataListenerException クラスのインスタンスを生成します。

形式

```
public SSODataListenerException();

public SSODataListenerException(String msg);
```

パラメタ

msg：

エラーメッセージを指定します。

例外

なし

getException メソッド

説明

このオブジェクトに格納されている例外オブジェクトを取得します。指定したリスナの例外オブジェクトが格納されていない場合は null を返却します。

形式

```
public ChangeDataFailedException getException(SSODataListener
listener);
```

パラメタ

listener：

例外が発生したリスナオブジェクトを指定します。

例外

なし

戻り値

ChangeDataFailedException オブジェクトを返却します。

getListeners メソッド

説明

この例外に格納されているリスナをすべて取得します。

形式

```
public SSODataListener[] getListeners();
```

パラメタ

なし

例外

なし

戻り値

リスナオブジェクトの配列を返却します。

setException メソッド

説明

例外の原因になった例外オブジェクトをこのオブジェクトに格納します。すでに同じリスナの例外が格納されていた場合は上書きします。

形式

```
public void setException(SSODataListener listener,
                         ChangeDataFailedException exception);
```

パラメタ

listener：

例外が発生したリスナオブジェクトを指定します。

exception：

リスナで発生した ChangeDataFailedException オブジェクトを指定します。

例外

なし

戻り値

なし

8.17　UserAttributes インタフェース

説明

ユーザ認証後，Subject に関連づけられた属性を取得するためのインタフェースです。

UserAttributes インタフェースのパッケージ名は，com.cosminexus.admin.auth です。

形式

```
interface UserAttributes
{
  public Object getAttribute(String alias)
    throws IllegalStateException;
  public Enumeration getAttributes(String alias)
    throws IllegalStateException;
  public void addAttribute(String alias,
                           Object attr)
    throws IllegalStateException;
  public Enumeration getAttributeNames()
    throws IllegalStateException;
  public void removeAttribute(String alias)
    throws IllegalStateException;
  public int size()
    throws IllegalStateException;
  public Enumeration getAliases()
    throws IllegalStateException;
}
```

メソッド一覧

メソッド名	機能
addAttribute メソッド	Subject に属性を追加します。
getAttribute メソッド	Subject に関連づけられた属性を取得します。
getAttributeNames メソッド	Subject に関連づけられた属性名の一覧を取得します。
getAttributes メソッド	Subject に関連づけられた属性をすべて取得します。
removeAttribute メソッド	Subject に関連づけられた属性を削除します。
size メソッド	Subject に関連づけられた属性の総数を取得します。
getAliases メソッド	推奨されていません。getAttributeNames メソッドを使用してください。

注意事項

このオブジェクトが無効な場合に，各メソッドの呼び出しで java.lang.IllegalStateException 例外が発生します。この例外は，java.lang.RuntimeException 例外を継承しているため，catch および throws に記述しなくてもコンパイルできるので注意してください。

addAttribute メソッド

説明

Subject に属性を追加します。一つの属性に対して，複数の属性値を関連づけることができます。このメソッドを使用して関連づけた属性は，ユーザ管理リポジトリには反映されません。

形式

```
public void addAttribute(String alias,
                         Object attr)
  throws IllegalStateException;
```

パラメタ

alias：

Subject に関連づける属性名を指定します。

attr：

Subject に関連づける属性値を指定します。

例外

java.lang.IllegalStateException：

このオブジェクトが無効な場合に発生します。また，この例外は，java.lang.RuntimeException 例外を継承しているため，catch および throws に記述しなくてもコンパイルできるので注意してください。

この例外は，次の条件で発生します。

- このオブジェクトを持つ Subject が read-only です。
- logout メソッドによってログアウト処理されています。

戻り値

なし

getAttribute メソッド

説明

Subject に関連づけられた属性を取得します。要求元では，返された Object をキャストして値を参照します。同じ属性で複数の値を持つ場合は，最初に見つかった Object を返却します。

形式

```
public Object getAttribute(String alias)
  throws IllegalStateException;
```

パラメタ

alias：

Subject に関連づけられた属性名を指定します。AttributeEntry クラスに属性の別名を指定した場合は，その別名を指定します。

例外

java.lang.IllegalStateException：

このオブジェクトが無効な場合に発生します。また，この例外は，java.lang.RuntimeException 例外を継承しているため，catch および throws に記述しなくてもコンパイルできるので注意してください。

この例外は，次の条件で発生します。

- このオブジェクトを持つ Subject が read-only です。
- logout メソッドによってログアウト処理されています。

戻り値

指定した Subject に関連づけられた属性値を返却します。見つからない場合は null を返却します。

getAttributeNames メソッド

説明

Subject に関連づけられた属性名の一覧を取得します。AttributeEntry クラスに属性の別名を指定した場合は，その別名を返却します。

形式

```
public Enumeration getAttributeNames()
  throws IllegalStateException;
```

パラメタ

なし

例外

java.lang.IllegalStateException：

このオブジェクトが無効である場合に発生します。また，この例外は，java.lang.RuntimeException 例外を継承しているため，catch および throws に記述しなくてもコンパイルできるので注意してください。

この例外は，次の条件で発生します。

- このオブジェクトを持つ Subject が read-only です。
- logout メソッドによってログアウト処理されています。

戻り値

Subject に関連づけられた属性の名称の一覧を返却します。AttributeEntry クラスに属性の別名を指定した場合は，その別名を返却します。

getAttributes メソッド

説明

Subject に関連づけられた属性をすべて取得します。要求元では，Enumeration の nextElement メソッドで Object を取得し，キャストして値を参照します。

形式

```
public Enumeration getAttributes(String alias)
  throws IllegalStateException;
```

パラメタ

alias：

Subject に関連づけられた属性名を指定します。AttributeEntry クラスに属性の別名を指定した場合は，その別名を指定します。

例外

java.lang.IllegalStateException：

このオブジェクトが無効である場合に発生します。また，この例外は，java.lang.RuntimeException 例外を継承しているため，catch および throws に記述しなくてもコンパイルできるので注意してください。

この例外は，次の条件で発生します。

- このオブジェクトを持つ Subject が read-only です。
- logout メソッドによってログアウト処理されています。

戻り値

指定した Subject に関連づけられた属性値を返却します。見つからない場合は null を返却します。

removeAttribute メソッド

説明

Subject に関連づけられた属性を削除します。このメソッドを使用して削除した属性は，ユーザ管理リポジトリには反映されません。複数の属性値が関連づけられていてもすべて削除されます。

形式

```
public void removeAttribute(String alias)
  throws IllegalStateException;
```

パラメタ

alias：

Subject に関連づけられた属性の名称を指定します。AttributeEntry クラスに属性の別名を指定した場合は，その別名を指定します。

例外

java.lang.IllegalStateException：

このオブジェクトが無効な場合に発生します。また，この例外は，java.lang.RuntimeException 例外を継承しているため，catch および throws に記述しなくてもコンパイルできるので注意してください。

この例外は，次の条件で発生します。

- このオブジェクトを持つ Subject が read-only です。
- logout メソッドによってログアウト処理されています。

戻り値

なし

size メソッド

説明

Subject に関連づけられた属性の総数を取得します。

形式

```
public int size()
  throws IllegalStateException;
```

パラメタ

なし

例外

java.lang.IllegalStateException：

このオブジェクトが無効な場合に発生します。また，この例外は，java.lang.RuntimeException 例外を継承しているため，catch および throws に記述しなくてもコンパイルできるので注意してください。

この例外は，次の条件で発生します。

- このオブジェクトを持つ Subject が read-only です。
- logout メソッドによってログアウト処理されています。

戻り値

Subject に関連づけられた属性の総数を返却します。

8.18　UserData クラス

説明

ユーザ情報を格納するクラスです。
UserData クラスのパッケージ名は,
com.cosminexus.admin.auth.api.repository.ldap です。

形式

```
class UserData
{
  public UserData();

  public void setPassword(String password);
  public Enumeration getAttributeNames();
  public Object getAttribute(String name);
  public Enumeration getAttributes(String name);
  public void addAttribute(String name,
                           Object attr);
  public void removeAttribute(String name);
  public int size();
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
UserData コンストラクタ	UserData クラスのインスタンスを生成します。
addAttribute メソッド	このオブジェクトに属性値を一つ追加します。
getAttribute メソッド	このオブジェクトに格納されている内容から，指定した属性名に対応する属性値の一つを取得します。
getAttributeNames メソッド	このオブジェクトに格納されている属性名の一覧を取得します。
getAttributes メソッド	このオブジェクトに格納されている内容から，指定した属性名に対応する属性値をすべて取得します。
removeAttribute メソッド	このオブジェクトから属性を削除します。
setPassword メソッド	パスワードをこのオブジェクトに格納します。
size メソッド	このオブジェクトに格納されている属性の総数を取得します。

UserData コンストラクタ

説明

インスタンスを生成します。

形式

```
public UserData();
```

パラメタ

なし

例外

なし

addAttribute メソッド

説明

このオブジェクトに属性値を一つ追加します。一つの属性に対して複数の属性値を関連づけることができます。

形式

```
public void addAttribute(String name,
                         Object attr);
```

パラメタ

name：
　属性の名称を指定します。属性に別名がある場合は，その別名を指定します。

attr：
　属性の値を指定します。

例外

なし

戻り値

なし

getAttribute メソッド

説明

このオブジェクトに格納されている内容から，指定した属性名に対応する属性値の一つを取得します。属性値が複数の場合は，それらの値のどれか一つを取得します。

形式

```
public Object getAttribute(String name);
```

パラメタ

name：

属性の名称を指定します。属性に別名がある場合は，その別名を指定します。

例外

なし

戻り値

属性値を返却します。見つからない場合は null を返却します。

getAttributeNames メソッド

説明

このオブジェクトに格納されている属性名の一覧を取得します。

形式

```
public Enumeration getAttributeNames();
```

パラメタ

なし

例外

なし

戻り値

属性名の一覧を格納した Enumeration オブジェクトを返却します。

getAttributes メソッド

説明

このオブジェクトに格納されている内容から，指定した属性名に対応する属性値をすべて取得します。要求元では，Enumeration の nextElement メソッドで Object を取得し，キャストすることで値を参照します。

形式

```
public Enumeration getAttributes(String name);
```

パラメタ

name：

属性の名称を指定します。属性に別名がある場合は，その別名を指定します。

例外

なし

戻り値

属性値を格納した Enumeration オブジェクトを返却します。見つからない場合は null を返却します。

removeAttribute メソッド

説明

このオブジェクトから属性を削除します。複数の属性値が関連づけられている場合，すべて削除されます。

形式

```
public void removeAttribute(String name);
```

パラメタ

name：
　属性の名称を指定します。属性に別名がある場合は，その別名を指定します。

例外

なし

戻り値

なし

setPassword メソッド

説明

パスワードをこのオブジェクトに格納します。すでに値が格納されている場合は上書きされます。

形式

```
public void setPassword(String password);
```

パラメタ

password：
　パスワードを指定します。

例外

なし

戻り値

なし

size メソッド

説明

このオブジェクトに格納されている属性の総数を取得します。

形式

```
public int size();
```

パラメタ

なし

例外

なし

戻り値

属性の総数を返却します。

8.19 WebCertificateCallback クラス

説明

Web サーバで認証した結果を CallbackHandler からログインモジュールに渡すための実装クラスです。

WebCertificateCallback クラスのパッケージ名は,
com.cosminexus.admin.auth.callback です。

形式

```
class WebCertificateCallback implements
javax.security.auth.callback.Callback
{
  public WebCertificateCallback(String attrName);

  public void setSubjectID(String name);
  public String getSubjectID();
  public void setRequest(HttpServletRequest req);
  public HttpServletRequest getRequest();
  public void setResponse(HttpServletResponse res);
  public HttpServletResponse getResponse();
  public void setAttributeEntries(AttributeEntry[] aliases);
  public AttributeEntry[] getAttributeEntries();
  public void setTagID(String tid);
  public String getTagID();
  public void setTagEntry(String entry);
  public String getTagEntry();
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebCertificateCallback コンストラクタ	WebCertificateCallback クラスのインスタンスを生成します。
getAttributeEntries メソッド	setAttributeEntries メソッドで指定した属性の一覧を格納しているオブジェクトのリファレンスを取得します。
getRequest メソッド	setRequest メソッドで指定した HttpServletRequest のリファレンスを取得します。
getResponse メソッド	setResponse メソッドで指定した HttpServletResponse のリファレンスを取得します。
getSubjectID メソッド	setSubjectID メソッドで指定した DN 名を取得します。
getTagEntry メソッド	setTagEntry で指定した entry を取得します。
getTagID メソッド	setTagID で指定した TagID を取得します。
setAttributeEntries メソッド	パラメタに指定された属性の一覧を格納しているオブジェクトのリファレンスをオブジェクトに保管します。
setRequest メソッド	パラメタに指定された HttpServletRequest のリファレンスをオブジェクトに保管します。
setResponse メソッド	パラメタに指定された HttpServletResponse のリファレンスをオブジェクトに保管します。

コンストラクタ・メソッド名	機能
setSubjectID メソッド	パラメタに指定された DN 名をオブジェクトに保管します。
setTagEntry メソッド	パラメタに指定された login タグの entry 要素を保管します。
setTagID メソッド	パラメタに指定された login タグの id 要素を保管します。

WebCertificateCallback コンストラクタ

説明

WebCertificateCallback クラスのインスタンスを生成します。インスタンスは WebCertificateLoginModule の login メソッドの延長で生成されます。

形式

```
public WebCertificateCallback(String attrName);
```

パラメタ

attrName：

DN 名から分解するときの属性名を指定します。

例外

なし

getAttributeEntries メソッド

説明

setAttributeEntries メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public AttributeEntry[] getAttributeEntries();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getRequest メソッド

説明

setRequest メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public HttpServletRequest getRequest();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getResponse メソッド

説明

setResponse メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public HttpServletResponse getResponse();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getSubjectID メソッド

説明

setSubjectID メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public String getSubjectID();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getTagEntry メソッド

説明

setTagEntry メソッドで指定した内容を取得します。API を使用してログインした場合は null が返却されます。

形式

```
public String getTagEntry();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getTagID メソッド

説明

setTagID メソッドで指定した内容を取得します。API を使用してログインした場合は null が返却されます。

形式

```
public String getTagID();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

setAttributeEntries メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setAttributeEntries(AttributeEntry[] aliases);
```

パラメタ

aliases：

属性の一覧を格納しているオブジェクト（AttributeEntry の配列）のリファレンスを指定します。

例外

なし

戻り値

なし

setRequest メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setRequest(HttpServletRequest req);
```

パラメタ

req：

HttpServletRequest のリファレンスを指定します。

例外

なし

戻り値

なし

setResponse メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setResponse(HttpServletResponse res);
```

パラメタ

res：

HttpServletResponse のリファレンスを指定します。

例外

なし

戻り値

なし

setSubjectID メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setSubjectID(String uid);
```

パラメタ

uid：
　DN 名を指定します。

例外

なし

戻り値

なし

setTagEntry メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。

形式

```
public void setTagEntry(String tid);
```

パラメタ

tid：
　login タグの entry 要素を指定します。

例外

なし

戻り値

なし

setTagID メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。

形式

```
public void setTagID(String tid);
```

パラメタ

tid：

login タグの id 要素を指定します。

例外

なし

戻り値

なし

8.20 WebCertificateHandler クラス

説明

Web サーバで SSL 認証した結果の情報を取得するための実装クラスです。ユーザ認証ライブラリの CallbackHandler です。

WebCertificateHandler クラスのパッケージ名は，com.cosminexus.admin.auth.callback です。

形式

```
class WebCertificateHandler
{
  public WebCertificateHandler(HttpServletRequest request,
                               HttpServletResponse response,
                               AttributeEntry[] aliases)
    throws ParameterError;
  public WebCertificateHandler(HttpServletRequest request,
                               HttpServletResponse response,
                               String aliasesFile)
    throws ParameterError, FormatError, FileNotFoundException,
           IOException, SecurityException;

  public void handle(Callback[] callbacks)
    throws IOException, UnsupportedCallbackException;
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebCertificateHandler コンストラクタ	WebCertificateHandler クラスのインスタンスを生成します。
handle メソッド	SSL 認証の結果情報を取得します。

WebCertificateHandler コンストラクタ

説明

WebCertificateHandler クラスのインスタンスを生成します。request および response は，必ず指定します。null が指定されていた場合は，ParameterError 例外が呼び出されます。

形式

```
public WebCertificateHandler(HttpServletRequest request,
                             HttpServletResponse response,
                             AttributeEntry[] aliases)
    throws ParameterError;

public WebCertificateHandler(HttpServletRequest request,
                             HttpServletResponse response,
                             String aliasesFile)
    throws ParameterError, FormatError, FileNotFoundException,
           IOException, SecurityException;
```

パラメタ

request：
JSP/Servlet 起動時のパラメタをそのまま指定します。

response：
JSP/Servlet 起動時のパラメタをそのまま指定します。

aliases：
認証が成功したときに作成する Credential（UserAttributes）に格納する情報を指定します。取得する情報がない場合は null を指定します。この場合，Credential は作成されません（空の UserAttributes オブジェクトが作成されます）。aliases には AttributeEntry オブジェクトの配列を指定します。指定した内容で必要な情報がない場合は，FormatError 例外が発生します（必須の値が格納されていない，または Format にない値が指定されている場合）。

aliasesFile：
認証が成功したときに作成する Credential（UserAttributes）に格納する情報を指定します。取得する情報がない場合は null を指定します。この場合，Credential は作成されません（空の UserAttributes オブジェクトが作成されます）。aliasesFile にはファイル名を指定します。指定した内容で必要な情報がない場合は，FormatError 例外が発生します（必須の値が格納されていない，または Format にない値が指定されている場合）。

例外

java.io.FileNotFoundException：
ファイルがない，ファイルではなくディレクトリである，または何かの理由で開くことができません（FileInputStream クラスのコンストラクタで例外が発生した場合）。

java.lang.SecurityException：
SecurityManager が存在し，SecurityManager の checkRead メソッドでファイルへの読み込みアクセスが拒否されました。

java.io.IOException：
ファイルの読み込みに失敗しました。

com.cosminexus.admin.common.ParameterError：
HttpServletRequest または HttpServletResponse のリファレンスが指定されていません。

com.cosminexus.admin.common.FormatError：
aliases および aliasesFile に指定された内容で必要な情報がありません。または余分に指定されています。

handle メソッド

説明

Web サーバで SSL 認証した結果の情報を取得して，WebCertificateCallback オブジェクト（Callback の実装クラス）のリファレンスに設定します。

形式

```
public void handle(Callback[] callbacks)
   throws IOException, UnsupportedCallbackException;
```

パラメタ

callbacks：

WebSSOCallback オブジェクトのリファレンスを指定されていた場合，セッション情報を設定して返却します。このクラス以外のクラスが指定されていた場合は，コンストラクタに指定された CallbackHandler の handle メソッドを呼び出します。

例外

java.io.IOException：

HttpServletRequest 内に Web サーバで認証した結果がありません。

javax.security.auth.callback.UnsupportedCallbackException：

サポートしていない callbacks リファレンスが指定されています。

戻り値

callbacks に値を設定して返却します。このメソッドでは，戻り値はありません。

8.21　WebCertificateLoginModule クラス

説明

JAAS のログインモジュールの実装クラスです。Cosminexus 標準ログインモジュールです。Web サーバで認証された証明書からユーザ属性を求めます。

WebCertificateLoginModule クラスのパッケージ名は，com.cosminexus.admin.auth.login です。

8.22 WebLogoutCallback クラス

説明

Web アプリケーションにわたってきたユーザ情報を CallbackHandler からログインモジュールに渡すための実装クラスです。
WebLogoutCallback クラスのパッケージ名は,
com.cosminexus.admin.auth.callback です。

形式

```
class WebLogoutCallback implements
javax.security.auth.callback.Callback
{
  private HttpSession session = null;
  private String userID = null;

  public WebLogoutCallback();
  public void setSession(HttpSession session);
  public String getSession();
  public void setUserID(String userID);
  public String getUserID();
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebLogoutCallback コンストラクタ	WebLogoutCallback クラスのインスタンスを生成します。
getSession メソッド	setSession メソッドで指定した HttpSession オブジェクトのリファレンスを取得します。
getUserID メソッド	setUserID メソッドで指定したユーザ ID を取得します。
setSession メソッド	パラメタに指定された HttpSession オブジェクトのリファレンスを保管します。
setUserID メソッド	パラメタに指定されたユーザ ID を保管します。

WebLogoutCallback コンストラクタ

説明

WebLogoutCallback クラスのインスタンスを生成します。Cosminexus 標準ログインモジュールに WebLogoutHandler が指定され，かつ logout メソッドが呼ばれた場合に生成されます。

形式

```
public WebLogoutCallback();
```

パラメタ

なし

例外

なし

getSession メソッド

説明

setSession メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public HttpSession getSession();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getUserID メソッド

説明

setUserID メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public String getUserID();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

setSession メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setSession(HttpSession session);
```

パラメタ

session：

HttpSession オブジェクトのリファレンスを指定します。

例外

なし

戻り値

なし

setUserID メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setUserID(String userID);
```

パラメタ

userID：

ユーザ ID を指定します。

例外

なし

戻り値

なし

8.23 WebLogoutHandler クラス

説明

ログアウトするユーザからユーザ ID を取得する JAAS Callback Handler クラスの実装です。

WebLogoutHandler クラスのパッケージ名は，

com.cosminexus.admin.auth.callback です。

形式

```
class WebLogoutHandler
{
  public WebLogoutHandler(HttpSession session , String userID)
throws ParameterError;
  public void handle(Callback[] callbacks)
  throws java.io.IOException, UnsupportedCallbackException;
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebLogoutHandler コンストラクタ	WebLogoutHandler クラスのインスタンスを生成します。
handle メソッド	ユーザ ID を取得します。

WebLogoutHandler コンストラクタ

説明

WebLogoutHandler クラスのインスタンスを生成します。

session パラメタおよび userID パラメタは，必ず指定してください。null が指定されていた場合は，ParameterError 例外が発生します。

形式

```
public WebLogoutHandler(HttpSession session, String userID);
```

パラメタ

session：

JSP/Servlet 起動時のパラメタをそのまま指定します。

userID：

ログアウトするユーザ ID を指定します。

例外

com.cosminexus.admin.common.ParameterError：

HttpSession または String のリファレンスが指定されていません。

handle メソッド

説明

ユーザ ID を取得し，WebLogoutCallback オブジェクト（Callback の実装クラス）のリファレンスに設定して統合ユーザ管理フレームワークが提供するログインモジュールに渡します。

形式

```
public void handle(Callback[] callbacks)
  throws UnsupportedCallbackException;
```

パラメタ

callbacks：

WebLogoutCallback オブジェクトのリファレンスを指定されていた場合，ユーザ情報を設定して返却します。上記以外のオブジェクトのリファレンスが指定されていた場合は，UnsupportedCallbackException 例外が発生します。

例外

java.io.IOException：

HttpServletRequest 内に Web サーバで認証した結果がありません。

javax.security.auth.callback.UnsupportedCallbackException：

サポートしていない callbacks リファレンスが指定されました。

戻り値

callbacks に値を設定して返却します。このメソッドでは，戻り値はありません。

8.24 WebPasswordCallback クラス

説明

Web アプリケーションに渡された認証情報を CallbackHandler からログインモジュールに渡すための実装クラスです。
WebPasswordCallback クラスのパッケージ名は，
com.cosminexus.admin.auth.callback です。

形式

```
class WebPasswordCallback implements
javax.security.auth.callback.Callback
{
  public static final int GETPW;
  public static final int NOPW;

  public WebPasswordCallback();

  public void setName(String name);
  public String getName();
  public void setPassword(String password);
  public String getPassword();
  public void setRequest(HttpServletRequest req);
  public HttpServletRequest getRequest();
  public void setResponse(HttpServletResponse res);
  public HttpServletResponse getResponse();
  public void setAttributeEntries(AttributeEntry[] aliases);
  public AttributeEntry[] getAttributeEntries();

  public void setOption(int option);
  public int getOption();
  public void setTagID(String tid);
  public String getTagID();
  public void setTagEntry(String entry);
  public String getTagEntry();
}
```

メンバ属性

GETPW：

この Callback オブジェクトにすべての情報を設定することを要求します。

NOPW：

この Callback オブジェクトにユーザ ID ／パスワードを除いた情報を設定することを要求します（このとき，url に対して forward ／ include はしません）。

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebPasswordCallback コンストラクタ	WebPasswordCallback クラスのインスタンスを生成します。
getAttributeEntries メソッド	setAttributeEntries メソッドで指定した，属性の一覧が格納されたオブジェクトのリファレンスを取得します。
getName メソッド	setName メソッドで指定したユーザ ID を取得します。

コンストラクタ・メソッド名	機能
getOption メソッド	設定されているオプションを取得します。
getPassword メソッド	setPassword メソッドで指定したパスワードを取得します。
getRequest メソッド	setRequest メソッドで指定した HttpServletRequest のリファレンスを取得します。
getResponse メソッド	setResponse メソッドで指定した HttpServletResponse のリファレンスを取得します。
getTagEntry メソッド	setTagEntry メソッドで指定した entry を取得します。
getTagID メソッド	setTagID メソッドで指定した TagID を取得します。
setAttributeEntries メソッド	パラメタに指定された，属性の一覧が格納されたオブジェクトのリファレンスをオブジェクトに保管します。
setName メソッド	パラメタに指定されたユーザ ID をこのオブジェクトに保管します。
setOption メソッド	CallbackHandler で設定する内容を要求します。
setPassword メソッド	パラメタに指定されたパスワードをこのオブジェクトに保管します。
setRequest メソッド	パラメタに指定された HttpServletRequest のリファレンスをこのオブジェクトに保管します。
setResponse メソッド	パラメタに指定された HttpServletResponse のリファレンスをこのオブジェクトに保管します。
setTagEntry メソッド	CallbackHandler メソッドで設定する内容を要求します。パラメタに指定された login タグの entry 要素を保管します。
setTagID メソッド	CallbackHandler メソッドで設定する内容を要求します。パラメタに指定された login タグの id 要素を保管します。

WebPasswordCallback コンストラクタ

説明

WebPasswordCallback クラスのインスタンスを生成します。インスタンスは WebPasswordLoginModule の login メソッドの延長で生成されます。

形式

```
public WebPasswordCallback();
```

パラメタ

なし

例外

なし

getAttributeEntries メソッド

説明

setAttributeEntries メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public AttributeEntry[] getAttributeEntries();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getName メソッド

説明

setName メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public String getName();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getOption メソッド

説明

設定されているオプションを取り出します。値を保管していない場合は GETPW が返却されます（デフォルトは，すべての情報を設定することを要求します）。

形式

```
public int getOption();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getPassword メソッド

説明

setPassword メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public String getPassword();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getRequest メソッド

説明

setRequest メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public HttpServletRequest getRequest();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getResponse メソッド

説明

setResponse メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public HttpServletResponse getResponse();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getTagEntry メソッド

説明

setTagEntry メソッドで指定した内容を取得します。API を使用してログインした場合は null が返却されます。

形式

```
public String getTagEntry();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getTagID メソッド

説明

setTagID メソッドで指定した内容を取得します。API を使用してログインした場合は null が返却されます。

形式

```
public String getTagID();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

setAttributeEntries メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setAttributeEntries(AttributeEntry[] aliases);
```

パラメタ

aliases：

属性の一覧を格納しているオブジェクト（AttributeEntry の配列）のリファレンスを指定します。

例外

なし

戻り値

なし

setName メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setName(String uid);
```

パラメタ

uid：

ユーザ ID を指定します。

例外

なし

戻り値

なし

setOption メソッド

説明

CallbackHandler で設定する内容を要求します。NOPW が指定された場合，ユーザ ID ／パスワードを除いた情報を設定することを CallbackHandler に対して要求します（このとき，url に対して forward ／ include はしません）。GETPW が指定された場合，この Callback オブジェクトに格納できるすべての情報を設定することを CallbackHandler に対して要求します。

すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setOption(int option);
```

パラメタ

option：

GETPW または NOPW を設定します。

- GETPW
  この Callback オブジェクトにすべての情報を設定することを要求します。
- NOPW
  この Callback オブジェクトにユーザ ID ／パスワードを除いた情報を設定することを要求します（このとき，url に対して forward ／ include はしません）。

例外

なし

戻り値

なし

setPassword メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setPassword(String password);
```

パラメタ

password：

パスワードを指定します。

例外

なし

戻り値

なし

setRequest メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setRequest(HttpServletRequest req);
```

パラメタ

req：

HttpServletRequest のリファレンスを指定します。

例外

なし

戻り値

なし

setResponse メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setResponse(HttpServletResponse res);
```

パラメタ

res：

HttpServletResponse のリファレンスを指定します。

例外

なし

戻り値

なし

setTagEntry メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。

形式

```
public void setTagEntry(String tid);
```

パラメタ

tid：

login タグの entry 要素を指定します。

例外

なし

戻り値

なし

setTagID メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。

形式

```
public void setTagID(String tid);
```

パラメタ

tid：

login タグの id 要素を指定します。

例外

なし

戻り値

なし

8.25 WebPasswordHandler クラス

説明

Web ブラウザを介して，ユーザからユーザ ID およびパスワードを取得する JAAS Callback Handler クラスの実装です。
ユーザ ID およびパスワードは，それぞれ HTTP リクエストの com.cosminexus.admin.auth.name パラメタおよび com.cosminexus.admin.auth.password パラメタに設定してください。
WebPasswordHandler クラスのパッケージ名は，com.cosminexus.admin.auth.callback です。

形式

```
class WebPasswordHandler
{
  public WebPasswordHandler(HttpServletRequest request,
                            HttpServletResponse response,
                            AttributeEntry [] aliases,
                            String url,
                            boolean urlforward)
    throws FormatError, ParameterError;
  public WebPasswordHandler(HttpServletRequest request,
                            HttpServletResponse response,
                            String aliasesFile,
                            String url,
                            boolean urlforward)
    throws IOException, SecurityException, FormatError,
    ParameterError;

  public void handle(Callback[] callbacks)
    throws IOException, UnsupportedCallbackException;
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebPasswordHandler コンストラクタ	WebPasswordHandler クラスのインスタンスを生成します。
handle メソッド	認証情報を取得します。

WebPasswordHandler コンストラクタ

説明

WebPasswordHandler クラスのインスタンスを生成します。

WebPasswordHandler コンストラクタには，Credential（UserAttributes）に格納するユーザ情報（属性）をメモリで指定する場合と，ファイルで指定する場合の二つの形式があります。なお，request パラメタおよび response パラメタは，必ず指定してください。null が指定されていた場合は，ParameterError 例外が発生します。

形式

```
public WebPasswordHandler(HttpServletRequest request,
                          HttpServletResponse response,
                          AttributeEntry[] aliases,
                          String url,
                          boolean urlforward)
  throws FormatError, ParameterError;

public WebPasswordHandler(HttpServletRequest request,
                          HttpServletResponse response,
                          String aliasesFile,
                          String url,
                          boolean urlforward)
  throws IOException, SecurityException, FormatError,
  ParameterError;
```

パラメタ

request：

JSP/Servlet 起動時のパラメタをそのまま指定します。

response：

JSP/Servlet 起動時のパラメタをそのまま指定します。

aliases：

認証が成功したときに作成する Credential（UserAttributes）に格納する情報を指定します。取得する情報がない場合は null を指定します。このとき，Credential は作成されません（空の UserAttributes オブジェクトが作成されます）。aliases には，AttributeEntry オブジェクトの配列を指定します。指定した内容で必要な情報がない場合は，FormatError 例外が発生します（必須の値が格納されていない，または Format にない値が指定されている場合）。

aliasesFile：

認証が成功したときに作成する Credential（UserAttributes）に格納する情報を指定します。取得する情報がない場合は null を指定します。このとき，Credential は作成されません（空の UserAttributes オブジェクトが作成されます）。aliasesFile にはファイル名を指定します。指定した内容で必要な情報がない場合は，FormatError 例外が発生します（必須の値が格納されていない，または Format にない値が指定されている場合）。

url：

ユーザから認証情報（ユーザ ID ／パスワード）を取得するための URL を指定します。URL の指定がある場合，urlforward の指定に従って Login Form を RequestDispatcher オブジェクトに渡します（ユーザに対して入力情報を取得する場合）。この指定が不要であれば null を指定します。また，null の場合は，urlforward の値を参照しません。null の場合で，handle メソッドの延長で認証情報の取得ができない（HttpServletRequest 内に格納されていない）ときは，LoginContext クラスの login メソッドの延長で LoginException 例外が発生します。

urlforward：

URL の表示方法を選択します。指定された URL に対して，true の場合，RequestDispatcher オブジェクトの forward メソッドを呼び出します。false の場合，RequestDispatcher オブジェクトに対して include メソッドを呼び出します。

例外

java.io.FileNotFoundException：

ファイルがない，ファイルではなくディレクトリである，または何かの理由で開くことができません（FileInputStream クラスのコンストラクタで例外が発生した場合）。

java.lang.SecurityException：

SecurityManager が存在し，SecurityManager の checkRead メソッドでファイルへの読み込みアクセスが拒否されました。

java.io.IOException：

ファイルの読み込みに失敗しました。

com.cosminexus.admin.common.ParameterError：

HttpServletRequest または HttpServletResponse のリファレンスが指定されていません。

com.cosminexus.admin.common.FormatError：

aliases および aliasesFile に指定された内容で必要な情報がありません。または余分に指定されています。

handle メソッド

説明

認証情報を取得し，WebPasswordCallback オブジェクト（Callback の実装クラス）のリファレンスに設定してユーザ認証ライブラリのログインモジュールに渡します。

形式

```
public void handle(Callback[] callbacks)
  throws IOException, UnsupportedCallbackException;
```

パラメタ

callbacks：

WebPasswordCallback オブジェクトのリファレンスを指定されていた場合，認証情報を設定して返却します。WebSSOCallback オブジェクトのリファレンスが指定されていた場合は，セッション情報を設定して返却します。上記以外のオブジェクトのリファレンスが指定されていた場合は，UnsupportedCallbackException 例外が

発生します。

例外

java.io.IOException：

HttpServletRequest 内にユーザ ID ／パスワードの情報がありません。取り出すパラメタ名は，「注意事項」を参照してください。

javax.security.auth.callback.UnsupportedCallbackException：

サポートしていない callbacks リファレンスが指定されていました。

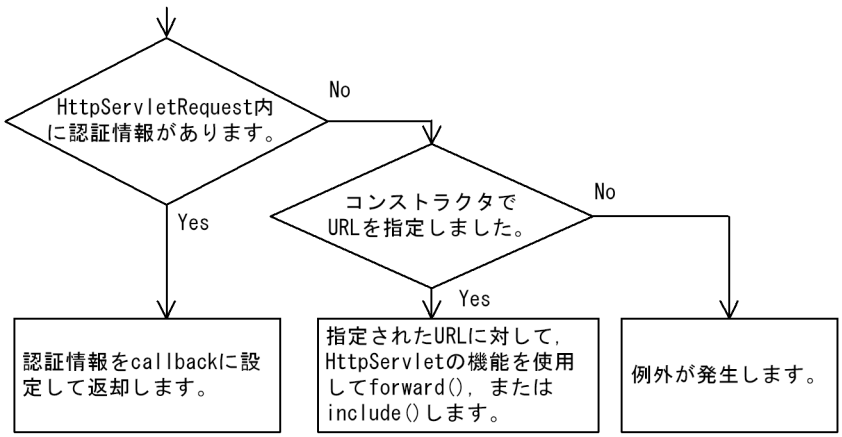
戻り値

callbacks に値を設定して返却します。このメソッドでは，戻り値はありません。

注意事項

認証情報は，次の図に示す順序で読み込まれます。

図 8-1　認証情報の読み込み順序



HttpServletRequest 内の認証情報は，次に示すパラメタ名から取得します。

- com.cosminexus.admin.auth.name
  ユーザが指定したユーザ ID を指定します。
- com.cosminexus.admin.auth.password
  ユーザが指定したパスワードを指定します。

8.26　WebPasswordJDBCLoginModule クラス

説明

JAAS のログインモジュールの実装クラスです。JDBC を使ってデータベースにアクセスし，パスワード認証をします。
WebPasswordJDBCLoginModule クラスのパッケージ名は，
com.cosminexus.admin.auth.login です。

8.27 WebPasswordLDAPLoginModule クラス

説明

JAAS のログインモジュールの実装クラスです。LDAP ディレクトリサーバとバインドした結果で認証をします。
WebPasswordLDAPLoginModule クラスのパッケージ名は,
com.cosminexus.admin.auth.login です。

8.28　WebPasswordLoginModule クラス

説明

JAAS のログインモジュールの実装クラスです。Web アプリケーションのためのパスワード認証をします。

WebPasswordLoginModule クラスのパッケージ名は,
com.cosminexus.admin.auth.login です。

8.29 WebSSOCallback クラス

説明

Web アプリケーションに渡されたセッション情報を CallbackHandler からログインモジュールに渡すための実装クラスです。
WebSSOCallback クラスのパッケージ名は,
com.cosminexus.admin.auth.sso.callback です。

形式

```
class WebSSOCallback implements
javax.security.auth.callback.Callback
{
  public WebSSOCallback();

  public void setRequest(HttpServletRequest req);
  public HttpServletRequest getRequest();
  public void setResponse(HttpServletResponse res);
  public HttpServletResponse getResponse();
  public void setTagID(String tid);
  public String getTagID();
  public void setTagEntry(String entry);
  public String getTagEntry();

}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebSSOCallback コンストラクタ	WebSSOCallback クラスのインスタンスを生成します。
getRequest メソッド	setRequest メソッドで指定した HttpServletRequest のリファレンスを取得します。
getResponse メソッド	setResponse で指定した HttpServletResponse のリファレンスを取得します。
getTagEntry メソッド	setTagEntry メソッドで指定した entry を取得します。
getTagID メソッド	setTagID メソッドで指定した TagID を取得します。
setRequest メソッド	パラメタに指定された HttpServletRequest のリファレンスをオブジェクトに保管します。
setResponse メソッド	パラメタに指定された HttpServletResponse のリファレンスをオブジェクトに保管します。
setTagEntry メソッド	パラメタに指定された login タグの entry 要素を保管します。
setTagID メソッド	パラメタに指定された login タグの id 要素を保管します。

WebSSOCallback コンストラクタ

説明

WebSSOCallback クラスのインスタンスを生成します。インスタンスは WebSSOLoginModule の login メソッドの延長で生成されます。

形式

```
public WebSSOCallback();
```

パラメタ

なし

例外

なし

getRequest メソッド

説明

setRequest メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public HttpServletRequest getRequest();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getResponse メソッド

説明

setResponse メソッドで指定した内容を取得します。値を保管していない場合は null が返却されます。

形式

```
public HttpServletResponse getResponse();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getTagEntry メソッド

説明

setTagEntry メソッドで指定した内容を取得します。API を使用してログインした場合は null が返却されます。

形式

```
public String getTagEntry();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

getTagID メソッド

説明

setTagID メソッドで指定した内容を取得します。API を使用してログインした場合は null が返却されます。

形式

```
public String getTagID();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが保管している値を返却します。

setRequest メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setRequest(HttpServletRequest req);
```

パラメタ

req：
　HttpServletRequest のリファレンスを指定します。

例外

なし

戻り値

なし

setResponse メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。すでに値を保管している情報がある場合は上書きされます。

形式

```
public void setResponse(HttpServletResponse res);
```

パラメタ

res：

HttpServletResponse のリファレンスを指定します。

例外

なし

戻り値

なし

setTagEntry メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。

形式

```
public void setTagEntry(String tid);
```

パラメタ

tid：

login タグの entry 要素を指定します。

例外

なし

戻り値

なし

setTagID メソッド

説明

パラメタに指定された値をこのオブジェクト内に保管します。

形式

```
public void setTagID(String tid);
```

パラメタ

tid：

login タグの id 要素を指定します。

例外

なし

戻り値

なし

8.30 WebSSOHandler クラス

説明

Web ブラウザを介して，セッションに関する情報を取得する JAAS CallbackHandler クラスの実装です。シングルサインオンライブラリの CallbackHandler です。

このクラスに各システムのログインモジュールに対応した CallbackHandler のリファレンスを指定することで，各システムの CallbackHandler の実装を変更しないでシングルサインオンの機能を実現できます。

WebSSOHandler クラスのパッケージ名は，com.cosminexus.admin.auth.sso.callback です。

形式

```
class WebSSOHandler
{
  public WebSSOHandler(HttpServletRequest request,
                       HttpServletResponse response,
                       CallbackHandler ch)
    throws ParameterError;

  public void handle(Callback[] callbacks)
    throws IOException, UnsupportedCallbackException;
}
```

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
WebSSOHandler コンストラクタ	WebSSOHandler クラスのインスタンスを生成します。
handle メソッド	セッション情報を取得します。

WebSSOHandler コンストラクタ

説明

WebSSOHandler クラスのインスタンスを生成します。request および response は，必ず指定します。null が指定されていた場合は，ParameterError 例外が発生します。

形式

```
public WebSSOHandler(HttpServletRequest request,
                     HttpServletResponse response,
                     CallbackHandler ch)
  throws ParameterError;
```

パラメタ

request：

JSP/Servlet 起動時のパラメタをそのまま指定します。

response：
JSP/Servlet 起動時のパラメタをそのまま指定します。

ch：
各システムの CallbackHandler のリファレンスを指定します。不要な場合は null を指定します。

例外

com.cosminexus.admin.common.ParameterError：
HttpServletRequest または HttpServletResponse のリファレンスが指定されていません。

handle メソッド

説明

セッション情報を取得して，WebSSOCallback オブジェクト（Callback の実装クラス）のリファレンスに設定してシングルサインオンライブラリのログインモジュールに渡します。

形式

```
public void handle(Callback[] callbacks)
  throws IOException, UnsupportedCallbackException;
```

パラメタ

callbacks：
WebSSOCallback オブジェクトのリファレンスが指定されていた場合，セッション情報を設定して返却します。このクラス以外が指定されていた場合は，コンストラクタに指定された CallbackHandler の handle メソッドを呼び出します。

例外

java.io.IOException：
各システムの CallbackHandler の handle メソッドでこの例外が発生しました。

javax.security.auth.callback.UnsupportedCallbackException：
この例外は，次の条件で発生します。
- コンストラクタに各システムの CallbackHandler のリファレンスを指定していません（null の場合）。
- 各システムの CallbackHandler の handle メソッドでこの例外が発生しました。

戻り値

callbacks に値を設定して返却します。このメソッドでは，戻り値はありません。

8.31 WebSSOLoginModule クラス

説明

JAAS のログインモジュールの実装クラスです。シングルサインオンをするためにほかのログインモジュールを呼び出します。
WebSSOLoginModule クラスのパッケージ名は,
com.cosminexus.admin.auth.sso.login です。

8.32　例外クラス

統合ユーザ管理の API で使用する例外クラスについて説明します。使用する例外クラスには，JAAS で規定されているログインモジュールの例外クラスと日立で提供する API（JAAS 以外）の例外クラスがあります。

（1） JAAS のログインモジュールの例外クラス

JAAS で規定されているログインモジュールの例外クラスを次の表に示します。

表 8-3　JAAS のログインモジュールの例外クラス一覧

項番	例外名	内容
1	javax.security.auth.login.LoginException	項番 2 ～ 4 の親クラスです。コンストラクタのパラメタに msg（java.lang.String）を持ちます。
2	javax.security.auth.login.AccountExpiredException	ユーザアカウントが期限切れであることを通知します。
3	javax.security.auth.login.CredentialExpiredException	Credential が期限切れであることを通知します。
4	javax.security.auth.login.FailedLoginException	認証が失敗したことを通知します。

また，ユーザ認証ライブラリ／シングルサインオンライブラリのログインモジュールでは，これらの例外にエラーメッセージ文字列を設定して送出します。エラーメッセージ文字列の一覧を次の表に示します。

なお，ここで示した例外以外でも，JAAS のコンフィグレーションファイルの記述に誤りがある場合に LoginContext クラスをインスタンス化すると，java.lang.SecurityException 例外が発生します。その場合は，次の表のエラーメッセージ文字列を参照して，JAAS のコンフィグレーションファイルの内容を修正してください。

表 8-4　ユーザ認証ライブラリ／シングルサインオンライブラリのログインモジュールの例外

例外名	エラーメッセージ文字列	発生条件
javax.security.auth.login.FailedLoginException	data not found	認証情報が，渡されたパラメタ内にありません。WebPasswordHandler クラスに渡した HttpServletRequest 内にユーザ ID ／パスワードが格納されていません。

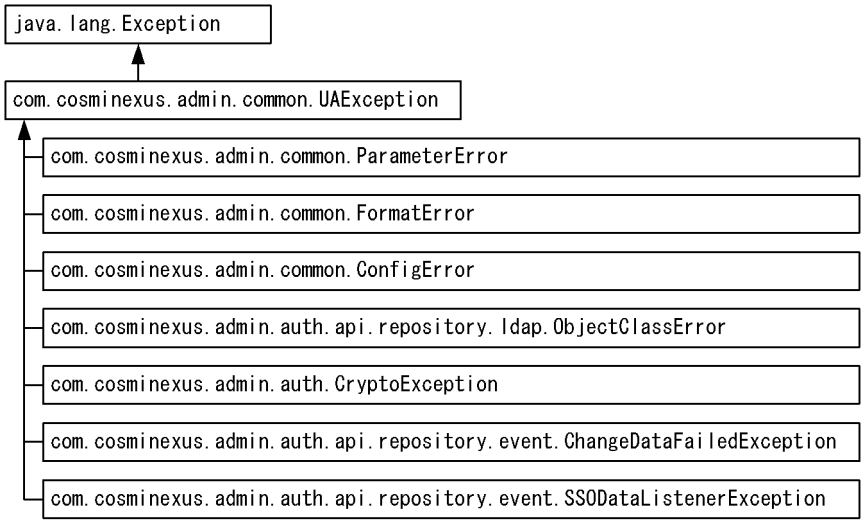
例外名	エラーメッセージ文字列	発生条件
	invalid data	• ユーザ ID ／パスワードに誤りがあり，認証できません。 • 証明書から取り出したユーザ ID に対応したエントリが，リポジトリ内にありません。
	no data	該当セッション内で認証済みである場合に，呼び出そうとしたレルムに対応したシングルサインオン用認証情報が定義されていません。
javax.security.auth.login.LoginException	invalid parameter	Credential を作成しようとしたときの属性名と属性の一覧の指定内容に次のような誤りがあります。 • 属性名が指定されていません。 • 同じ Alias が重複して指定されました。
	SQL の例外名称	JDBC を使用したアクセスに失敗しました。この例外が発生した場合は，エラーメッセージ文字列を参照して対処してください。
	JNDI の例外名称	LDAP のアクセスに失敗しました。 • LDAP サーバが見つかりません（CommunicationException）。 • バインド DN の指定ミスです（AuthenticationException）。
	not supported	未サポートの CallbackHandler を使用しました。 • WebSSOLoginModule または WebPasswordLoginModule で必要とする情報が，CallbackHandler で求められません。 • handle メソッドで例外が発生しました。この例外は，上記の条件でユーザ管理が提供する CallbackHandler だけで発生します。
	no class for xxx	WebSSOLoginModule から呼び出すクラスが使用できません（xxx は com.cosminexus.admin.auth.sso.loginmodule で指定された値）。 • インスタンス化ができません。JAAS のログインモジュールを継承していません。また，アクセス権限がない，クラスパスが設定されていない場合があります。
	config error	• JAAS のコンフィグレーションファイルに必要な情報が記述されていないため，処理を続行できませんでした。 • Cosminexus 標準ログインモジュールによるユーザ管理のコンフィグレーションファイルに必要な情報が記述されていないため，処理を続行できませんでした。
	invalid session	HttpSession オブジェクトに関連づけようとしましたが，HttpSession オブジェクトが無効になりました。

例外名	エラーメッセージ文字列	発生条件
	crypto error	暗号化／復号化で失敗しました。 • JNI 機能で呼び出すシングルサインオンライブラリの共用ライブラリが見つかりません（java.library.path の設定に問題があります）。 • 復号化に失敗しました（暗号化と復号化で異なる鍵を使用しています）。
	no sso data	シングルサインオン用の情報がありません。 • シングルサインオンをするために必要な情報が定義されていませんでした。
	no principal	Principal がないため，最初に認証したユーザを特定することができませんでした。
	class cast error	リポジトリから取り出した型と統合ユーザ管理のコンフィグレーションファイルに指定された型が不一致です。一致するようにしてください。ua.conf（統合ユーザ管理のコンフィグレーションファイル）の com.cosminexus.admin.auth.ldap.password.encrypt を参照してください。ua.conf 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（サーバ定義）」の「14.3　ua.conf（統合ユーザ管理のコンフィグレーションファイル）」を参照してください。
	not found driver	JDBC を使用しました。 • WebPasswordJDBCLoginModule でドライバが見つかりません。ドライバを正しい場所に格納してください。
	その他	各システムのログインモジュールで発生しました。 • WebSSOLoginModule で，ユーザ認証ライブラリ以外のログインモジュールから発生しました。

(2) 日立で提供する API の例外クラス

日立で提供する API（JAAS 以外）の例外クラスは，次の図に示す階層を持ちます。

図 8-2 例外クラスの階層



例外クラス一覧を次の表に示します。

表 8-5 日立で提供する API の例外クラス一覧

項番	例外名	内容
1	com.cosminexus.admin.common.UAException	項番 2 ～ 8 の親クラスです。
2	com.cosminexus.admin.common.ParameterError	各 API のパラメタの指定に誤りがありました。
3	com.cosminexus.admin.common.FormatError	Format を持つものに対する指定に誤りがありました。
4	com.cosminexus.admin.common.ConfigError	コンフィグレーションファイルの内容に誤りがありました。
5	com.cosminexus.admin.auth.api.repository.ldap.ObjectClassError	オブジェクトクラスの指定に誤りがありました。
6	com.cosminexus.admin.auth.CryptoException	暗号化／復号化に失敗しました。
7	com.cosminexus.admin.auth.api.repository.event.ChangeDataFailedException	他システムの認証情報の更新に失敗した場合に，リスナクラスが呼び出されます。
8	com.cosminexus.admin.auth.api.repository.event.SSODataListenerException	他システムの認証情報の更新に失敗した場合に，LdapSSODataManager クラスが呼び出されます。

9 統合ユーザ管理フレームワークで使用するタグライブラリ

この章では，統合ユーザ管理フレームワークで使用する JSP タグライブラリについて説明します。

9.1 タグライブラリのタグの一覧

統合ユーザ管理フレームワークでは，JSP で統合ユーザ管理フレームワークの機能を利用してユーザ認証を実装するための JSP タグライブラリを提供します。

統合ユーザ管理フレームワークの JSP タグライブラリを JSP にインポートするためには，次のコードを JSP に記述します。

```
<%@ taglib uri="http://cosminexus.com/admin/auth/uatags"
prefix="ua" %>
```

統合ユーザ管理フレームワークが提供する JSP タグライブラリに含まれるタグの一覧を次の表に示します。

表 9-1 JSP タグライブラリのタグの一覧

タグ名	概要
<ua:attributeEntries>Entries</ua:attributeEntries> タグ	<ua:attributeEntry/> タグと協調して，ログイン時に取得するユーザ属性の一覧を指定します。
<ua:attributeEntry/> タグ	ログイン時に取得する個々のユーザ属性を指定します。
<ua:chpw/> タグ	指定したユーザのパスワードを変更します。
<ua:exception>Body</ua:exception> タグ	例外が発生した場合の処理を Body に記述します。
<ua:getPrincipalName/> タグ	ログインしているユーザの Principal 名（ユーザ ID）を取得または表示します。
<ua:getAttribute/> タグ	ログインしているユーザのユーザ属性値を取得または表示します。
<ua:getAttributes/> タグ	ログインしているユーザのユーザ属性値（Multi-Value）を取得または表示します。
<ua:getAttributeNames/> タグ	ログインしているユーザのユーザ属性名の一覧を取得または表示します。
<ua:login/> タグ	ログインします。
<ua:logout/> タグ	ログアウトします。
<ua:notLogin>Body</ua:notLogin> タグ	ログインしていない場合の処理を Body に記述します。各 JSP ページの先頭に記述することで，その JSP ページを処理する前にログインしているかどうか確認できます。

この章の 9.2 ～ 9.12 では，各タグのタグ属性について表形式で説明しています。表中の「型」はタグ属性の結果として定義されるスクリプト変数の型を，「C/R」はタグ属性の値を JSP コンパイル時に評価するか（C），実行時に評価するか（R）を示します。

9.2 <ua:attributeEntries>Entries</ua:attributeEntries> タグ

(1) 説明

<ua:attributeEntry/> タグと協調して，ログイン時に取得するユーザ属性の一覧を指定します。取得するユーザ属性の一覧は，Entries に <ua:attributeEntry/> タグを複数記述して指定します。

```
<ua:attributeEntries id="ae">
  <ua:attributeEntry attrName="cn" />
  <ua:attributeEntry attrName="sn" />
  ...
</ua:attributeEntries>
```

(2) タグ属性

タグ属性を次の表に示します。

表 9-2 タグ属性（<ua:attributeEntries>Entries</ua:attributeEntries> タグ）

タグ属性	説明	型	要否	C/R
id="id"	id に AttributeEntry クラスの配列のインスタンスを識別する識別子を指定します。さらに，id はこのインスタンスを参照するスクリプト変数の変数名としても利用されます。このインスタンスは scope タグ属性で指定されたスコープを持ちます。ここで使用する識別子は，すべてのスコープで一意の識別子にする必要があります。	AttributeEntry[]	○	C
scope="scope"	scope に id タグ属性で指定したスクリプト変数のスコープを指定します。指定できる値は，"page"，"request"，"session"，"application" のどれかです。それぞれの値の意味は <jsp:useBean/> タグを参照してください。scope タグ属性を省略した場合は，"page" を仮定します。	－	△	C

（凡例）
○：必要です。
△：任意です。
－：該当しません。
C：タグ属性の値を JSP コンパイル時に評価します。

9.3 <ua:attributeEntry/> タグ

(1) 説明

ログイン時に取得する個々のユーザ属性を指定します。詳細については,「9.2 <ua:attributeEntries>Entries</ua:attributeEntries> タグ」を参照してください。

(2) タグ属性

タグ属性を次の表に示します。

表 9-3 タグ属性（<ua:attributeEntry/> タグ）

タグ属性	説明	型	要否	C/R
attrName="attrName"	attrName にログイン時に取得するユーザ属性の名称を指定します。	－	○	R
alias="alias"	alias に取得するユーザ属性の別名を指定します。	－	△	R
subCxt="subCxt"	subCxt にユーザ管理コンテキストからのサブコンテキストを指定します。	－	△	R

（凡例）
○：必要です。
△：任意です。
－：該当しません。
R：タグ属性の値を実行時に評価します。

9.4　<ua:chpw/> タグ

(1) 説明

指定したユーザのパスワードを変更します。

(2) タグ属性

タグ属性を次の表に示します。

表 9-4　タグ属性（<ua:chpw/> タグ）

タグ属性	説明	型	要否	C/R
entry="JAAS entry"	JAAS entry に JAAS コンフィグレーションファイルのエントリ名を指定します。	－	○	R
userid="userid"	userid にパスワードを変更するユーザのユーザ ID を指定します。	－	○※	R
useridParam="useridParam"	useridParam にパスワードを変更するユーザのユーザ ID が格納された HTTP リクエストパラメタの名前を指定します。	－		R
oldpw="oldpw"	oldpw に現在のパスワードを平文で指定します。	－	○※	R
oldpwParam="oldpwParam"	oldpwParam に現在のパスワードが格納された HTTP リクエストパラメタの名前を指定します。	－		R
newpw="newpw"	newpw に新しいパスワードを平文で指定します。	－	○※	R
newpwParam="newpwParam"	newpwParam に新しいパスワードが格納された HTTP リクエストパラメタの名前を指定します。	－		R
excepId="excepId"	excepId にパスワード変更処理中に発生した例外のインスタンスを識別する識別子を指定します。このインスタンスは excepScope タグ属性で指定されたスコープを持ちます。ここで使用する識別子は，すべてのスコープで一意の識別子にする必要があります。excepId タグ属性を省略した場合は，パスワード変更処理中に発生した例外は，JspException 例外として <ua:chpw/> タグの外側に伝わります。	－	△	C

タグ属性	説明	型	要否	C/R
excepScope="scope"	scope に excepId タグ属性で指定したスクリプト変数のスコープを指定します。指定できる値は，"page", "request"，"session"，"application" のどれかです。それぞれの値の意味は，<jsp:useBean/> タグを参照してください。excepScope タグ属性を省略した場合は，"page" を仮定します。	－	△	C

（凡例）

○：必要です。

△：任意です。

－：該当しません。

C：タグ属性の値を JSP コンパイル時に評価します。

R：タグ属性の値を実行時に評価します。

注※

どちらかを指定します。

9.5 <ua:exception>Body</ua:exception> タグ

(1) 説明

指定した例外が発生した場合の処理を Body に記述します。

(2) タグ属性

タグ属性を次の表に示します。

表 9-5 タグ属性（<ua:exception>Body</ua:exception> タグ）

タグ属性	説明	型	要否	C/R
name="name"	name に <ua:login/> タグや <ua:chpw/> タグなどで指定した例外オブジェクトの識別子（excepId タグ属性で指定）を指定します。誤った識別子を指定した場合は何もしません。	－	○	C
type="type"	type に catch する例外オブジェクトのクラス名を，パッケージ名を含めた完全名で指定します。catch しようとする例外オブジェクトが指定したクラス名のクラスにキャストできる場合，Body が実行されます。type タグ属性を省略した場合は，"java.lang.Throwable" を仮定します。	－	△	C
proceed="proceed"	proceed に Body 処理後，残りの JSP ページを処理するかどうかを指定します。"true" を指定した場合（Ignore Case）は残りの JSP ページを処理し，そうでない場合は残りの JSP ページを処理しません。proceed タグ属性を省略した場合は，"false" を仮定します。	－	△	R

（凡例）

○：必要です。

△：任意です。

－：該当しません。

C：タグ属性の値を JSP コンパイル時に評価します。

R：タグ属性の値を実行時に評価します。

9.6 <ua:getPrincipalName/> タグ

(1) 説明

ログインしているユーザの Principal 名（ユーザ ID）を取得または表示します。

(2) タグ属性

タグ属性を次の表に示します。

表 9-6 タグ属性（<ua:getPrincipalName/> タグ）

タグ属性	説明	型	要否	C/R
id="id"	id にログインしているユーザの Principal 名（ユーザ ID）を参照するインスタンスを識別する識別子を指定します。さらに，id はこのインスタンスを参照するスクリプト変数の変数名としても利用されます。このインスタンスはページスコープを持ちます。そのため，同一 JSP ページ内で参照できます。したがって，id には，ページスコープで一意の識別子を指定する必要があります。id タグ属性を省略した場合は，取得した Principal 名を JSP に埋め込みます。	String	△	C
name="name"	name に <ua:login/> タグで指定した JAAS LoginContext オブジェクトの識別子（id タグ属性で指定）を指定します。誤った識別子を指定した場合は，id スクリプト変数に null を設定します。	－	○	R

（凡例）

○：必要です。

△：任意です。

－：該当しません。

C：タグ属性の値を JSP コンパイル時に評価します。

R：タグ属性の値を実行時に評価します。

9.7 <ua:getAttribute/> タグ

(1) 説明

ログインしているユーザのユーザ属性値を取得または表示します。

(2) タグ属性

タグ属性を次の表に示します。

表 9-7 タグ属性（<ua:getAttribute/> タグ）

タグ属性	説明	型	要否	C/R
id="id"	id にログインしているユーザのユーザ属性値を参照するインスタンスを識別する識別子を指定します。さらに，id はこのインスタンスを参照するスクリプト変数の変数名としても利用されます。このインスタンスはページスコープを持ちます。そのため，同一 JSP ページ内で参照できます。したがって，id には，ページスコープで一意の識別子を指定する必要があります。id タグ属性を省略した場合は，取得したユーザ属性値を toString メソッドを使用して JSP に埋め込みます。	type タグ属性の値	△	C
name="name"	name に <ua:login/> タグで指定した JAAS LoginContext オブジェクトの識別子（id タグ属性で指定）を指定します。誤った識別子を指定した場合は，id スクリプト変数に null を設定します。	－	○	R
attrName="attrName"	attrName に取得するユーザ属性の属性名を指定します。指定したユーザ属性がない場合，id スクリプト変数に null を設定します。	－	○	R
type="type"	type に取得する属性オブジェクトのクラス名を，パッケージ名を含めた完全名で指定します。type タグ属性を省略した場合は，"java.lang.String" を仮定します。	－	△	C

（凡例）

○：必要です。

△：任意です。

－：該当しません。

C：タグ属性の値を JSP コンパイル時に評価します。

R：タグ属性の値を実行時に評価します。

9.8 <ua:getAttributes/> タグ

(1) 説明

ログインしているユーザのユーザ属性値（Multi-Value）を取得または表示します。

(2) タグ属性

タグ属性を次の表に示します。

表 9-8 タグ属性（<ua:getAttributes/> タグ）

タグ属性	説明	型	要否	C/R
id="id"	id にログインしているユーザのユーザ属性値（Multi-Value）を参照するインスタンスを識別する識別子を指定します。さらに，id はこのインスタンスを参照するスクリプト変数の変数名としても利用されます。このスクリプト変数の型は，java.util.Enumeration です。このインスタンスはページスコープを持ちます。そのため，同一 JSP ページ内で参照できます。したがって，id には，ページスコープで一意の識別子を指定する必要があります。id タグ属性を省略した場合は，取得したユーザ属性値を toString メソッドを使用して JSP に 1 行ずつ埋め込みます。	java.util.Enumeration	△	C
name="name"	name に <ua:login/> タグで指定した JAAS LoginContext オブジェクトの識別子（id タグ属性で指定）を指定します。誤った識別子を指定した場合は，id スクリプト変数に null を設定します。	－	○	R
attrName="attrName"	attrName に取得するユーザ属性の属性名を指定します。指定したユーザ属性がない場合，id スクリプト変数に null を設定します。	－	○	R

（凡例）

○：必要です。

△：任意です。

－：該当しません。

C：タグ属性の値を JSP コンパイル時に評価します。

R：タグ属性の値を実行時に評価します。

9.9 <ua:getAttributeNames/> タグ

(1) 説明

ログインしているユーザのユーザ属性名の一覧を取得または表示します。

(2) タグ属性

タグ属性を次の表に示します。

表 9-9 タグ属性（<ua:getAttributeNames/> タグ）

タグ属性	説明	型	要否	C/R
id="id"	id にログインしているユーザのユーザ属性名の一覧を参照するインスタンスを識別する識別子を指定します。さらに，id はこのインスタンスを参照するスクリプト変数の変数名としても利用されます。このスクリプト変数の型は，java.util.Enumeration です。このインスタンスはページスコープを持ちます。そのため，同一 JSP ページ内で参照できます。したがって，id には，ページスコープで一意の識別子を指定する必要があります。id タグ属性を省略した場合は，取得したユーザ属性名を toString メソッドを使用して JSP に 1 行ずつ埋め込みます。	java.util.Enumeration	△	C
name="name"	name に <ua:login/> タグで指定した JAAS LoginContext オブジェクトの識別子（id タグ属性で指定）を指定します。誤った識別子を指定した場合は，id スクリプト変数に null を設定します。	－	○	R

（凡例）

○：必要です。

△：任意です。

－：該当しません。

C：タグ属性の値を JSP コンパイル時に評価します。

R：タグ属性の値を実行時に評価します。

9.10 <ua:login/> タグ

(1) 説明

指定した JAAS コンフィグレーション・エントリを使用してログインします。このとき，HTTP リクエストオブジェクトに，com.cosminexus.admin.auth.name パラメタおよび com.cosminexus.admin.auth.password パラメタが設定されている必要があります。また，<ua:login/> タグに対応した <ua:logout/> タグが記述されていない場合は，セッション切断時に暗黙的にログアウトします。

(2) タグ属性

タグ属性を次の表に示します。

表 9-10 タグ属性（<ua:login/> タグ）

タグ属性	説明	型	要否	C/R
id="id"	id に JAAS LoginContext クラスのインスタンスを識別する識別子を指定します。さらに，id はこのインスタンスを参照するスクリプト変数の変数名としても利用されます。このインスタンスはセッションスコープを持ちます。つまり，同一セッションに参加する異なる JSP ページで参照できます。ここで使用する識別子は，すべてのスコープで一意の識別子にする必要があります。	javax.security.auth.login.LoginContext	○	C
entry="JAAS entry"	JAAS entry に JAAS コンフィグレーションファイルのエントリ名を指定します。	－	○	R
attrFile="attrFile"	attrFile にユーザ管理リポジトリから取得する属性を記述したファイルのファイル名を指定します。ファイルの形式については，シングルサインオン用認証情報の CSV 形式ファイルを参照してください。シングルサインオン用認証情報の CSV 形式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（サーバ定義）」の「14.4 シングルサインオン用認証情報の CSV 形式ファイル」を参照してください。	－	△※	R
attrEntName="attrEntName"	attrEntName に <ua:attributeEntries> タグの id タグ属性で指定した識別子を指定します。	－		R

タグ属性	説明	型	要否	C/R
excepId="excepId"	excepId にログイン処理中に発生した例外のインスタンスを識別する識別子を指定します。このインスタンスは excepScope タグ属性で指定されたスコープを持ちます。ここで使用する識別子は，すべてのスコープで一意の識別子にする必要があります。excepId タグ属性を省略した場合は，ログイン処理中に発生した例外は JspException 例外として <ua:login/> タグの外側に伝わります。	－	△	C
excepScope="scope"	scope に excepId タグ属性で指定したスクリプト変数のスコープを指定します。指定できる値は，"page"，"request"，"session"，"application" のどれかです。それぞれの値の意味は，<jsp:useBean/> タグを参照してください。excepScope タグ属性を省略した場合は，"page" を仮定します。	－	△	C

（凡例）

○：必要です。

△：任意です。

－：該当しません。

C：タグ属性の値を JSP コンパイル時に評価します。

R：タグ属性の値を実行時に評価します。

注※

指定する場合はどちらかを指定します。

9.11 <ua:logout/> タグ

(1) 説明

ログアウトします。事前にログインしていない場合は，何もしません。

(2) タグ属性

タグ属性を次の表に示します。

表 9-11 タグ属性（<ua:logout/> タグ）

タグ属性	説明	型	要否	C/R
name="name"	name に <ua:login/> タグで指定した JAAS LoginContext オブジェクトの識別子（id タグ属性で指定）を指定します。誤った識別子を指定した場合は，何もしません。	－	○	R

（凡例）
○：必要です。
－：該当しません。
R：タグ属性の値を実行時に評価します。

9.12 <ua:notLogin>Body</ua:notLogin> タグ

(1) 説明

ログインしていない場合の処理を Body に記述します。各 JSP ページの先頭に記述することで，その JSP ページを処理する前にログインしているかどうかを確認できます。

(2) タグ属性

タグ属性を次の表に示します。

表 9-12 タグ属性（<ua:notLogin>Body</ua:notLogin> タグ）

タグ属性	説明	型	要否	C/R
realm="realm"	realm にレルム名を指定します。指定したレルムにログインしていない場合，Body を実行します。realm タグ属性を省略した場合は，どのレルムにもログインしていない場合に Body を実行します。	－	△	R
proceed="proceed"	proceed に Body 処理後，残りの JSP ページを処理するかどうかを指定します。"true" を指定した場合（Ignore Case）は残りの JSP ページを処理し，そうでない場合は残りの JSP ページを処理しません。proceed タグ属性を省略した場合は，"false" を仮定します。	－	△	R

（凡例）
△：任意です。
－：該当しません。
R：タグ属性の値を実行時に評価します。

10 ユーザログ機能で使用する API

この章では，ユーザログ機能で使用する API について説明します。

10.1　ユーザログ機能で使用する API の一覧

J2EE アプリケーション，バッチアプリケーション，または EJB クライアントアプリケーションが出力するログ（ユーザログ）を日立トレース共通ライブラリ形式で出力する場合に使用する API の一覧を次に示します。

表 10-1　ユーザログ機能で使用する API の一覧

クラス名	機能
CJLogRecord クラス	LogRecord クラスに，MsgID や AppName パラメタを追加したクラスです。このクラスのメソッドを利用して作成した LogRecord オブジェクト（CJLogRecord オブジェクト）を Logger.log メソッドに渡すことで，MsgID や AppName のフィールド値も実行時に指定した値で出力できます。

10.2 CJLogRecord クラス

説明

java.util.logging.LogRecord クラスに MsgID や AppName パラメタを追加したクラスです。MsgID や AppName が指定された場合の LogRecord オブジェクト（以降，CJLogRecord オブジェクトと呼びます）を作成するためのスタティックメソッドを提供しています。

CJLogRecord クラスのパッケージ名は，
com.hitachi.software.ejb.application.userlog です。

メソッド一覧

メソッド名	機能
create メソッド（形式 1）	Level，Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 2）	Level，Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 3）	Level，Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 4）	Level，Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 5）	Level，Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 6）	Level，Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 7）	Level，Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 8）	Level，Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 9）	Level，Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 10）	Level，Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 1）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 2）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 3）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

メソッド名	機能
createp メソッド（形式 4）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 5）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 6）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 7）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 8）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 9）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 10）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 1）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 2）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 3）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 4）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 5）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。

メソッド名	機能
createrb メソッド（形式 6）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 7）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 8）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 9）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 10）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

なお，CJLogRecord クラスの継承元である LogRecord クラスおよび各メソッドのパラメタに指定する Level は，java.util.logging パッケージに属するクラスです。

create メソッド（形式 1）

説明

Level，Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 2）

説明

Level，Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 String appName,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 3）

説明

Level，Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object param1,
                                 String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：
　LogRecord にセットするオブジェクトを指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 4）

説明

Level，Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object param1,
                                 String appName,
                                 String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：
　LogRecord にセットするオブジェクトを指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 5）

説明

Level，Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 6）

説明

Level，Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 String appName,
                                 String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
LogRecord にセットする例外オブジェクトを指定します。

appName：
AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 7）

説明

Level，Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object[] params,
                                 String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

params：
ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 8）

説明

Level，Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object[] params,
                                 String appName,
                                 String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

params：
ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：
AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 9）

説明

Level，Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブ

ジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 Object[] params,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 10）

説明

Level，Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 Object[] params,
                                 String appName,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
LogRecord にセットする例外オブジェクトを指定します。

params：
ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：
AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 1）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
ロギングの要求を発行したクラス名を指定します。

sourceMethod：
ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 2）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  String appName,
                                  String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
ロギングの要求を発行したクラス名を指定します。

sourceMethod：
ロギングの要求を発行したメソッド名を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

appName：
AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 3）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object param1,
                                  String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：
　LogRecord にセットするオブジェクトを指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 4）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object param1,
                                  String appName,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：

LogRecord にセットするオブジェクトを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 5）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
　LogRecord にセットする例外オブジェクトを指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 6）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  String appName,
                                  String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
ロギングの要求を発行したメソッド名を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
LogRecord にセットする例外オブジェクトを指定します。

appName：
AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 7）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object[] params,
                                  String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
ロギングの要求を発行したクラス名を指定します。

sourceMethod：
ロギングの要求を発行したメソッド名を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

params：
ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 8）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object[] params,
                                  String appName,
                                  String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
ロギングの要求を発行したクラス名を指定します。

sourceMethod：
ロギングの要求を発行したメソッド名を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

params：
ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：
AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 9）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  Object[] params,
                                  String msgID);
```

パラメタ

level：
メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
ロギングの要求を発行したクラス名を指定します。

sourceMethod：
ロギングの要求を発行したメソッド名を指定します。

msg：
文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
LogRecord にセットする例外オブジェクトを指定します。

params：
ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：
MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 10）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  Object[] params,
                                  String appName,
                                  String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
　LogRecord にセットする例外オブジェクトを指定します。

params：
　ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 1）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

bundleName：
　msg を地域化するためのリソースバンドル名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 2）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   String appName,
                                   String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

bundleName：
　msg を地域化するためのリソースバンドル名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 3）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
                                    Object param1,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：

LogRecord にセットするオブジェクトを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 4）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Object param1,
                                   String appName,
                                   String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

bundleName：
　msg を地域化するためのリソースバンドル名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：
　LogRecord にセットするオブジェクトを指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 5）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Throwable thrown,
                                   String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

bundleName：
　msg を地域化するためのリソースバンドル名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
　LogRecord にセットする例外オブジェクトを指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 6）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
                                    Throwable thrown,
                                    String appName,
                                    String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

bundleName：
　msg を地域化するためのリソースバンドル名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
　LogRecord にセットする例外オブジェクトを指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 7）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Object[] params,
                                   String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 8）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Object[] params,
                                   String appName,
                                   String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

bundleName：
　msg を地域化するためのリソースバンドル名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

params：
　ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 9）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
                                    Throwable thrown,
                                    Object[] params,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 10）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Throwable thrown,
                                   Object[] params,
                                   String appName,
                                   String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass ：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

bundleName：
　msg を地域化するためのリソースバンドル名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：
　LogRecord にセットする例外オブジェクトを指定します。

params：
　ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

11 監査ログ出力で使用する API

この章では，J2EE アプリケーションまたはバッチアプリケーションで監査ログを出力する場合に使用する API について説明します。

11.1 監査ログ出力で使用する API の一覧

監査ログ出力で使用する API の一覧を次の表に示します。

表 11-1 監査ログ出力で使用する API の一覧

クラス名	機能
AuditLogRecord クラス	監査ログのレコードを表すクラスです。
UserAuditLogger クラス	J2EE アプリケーションまたはバッチアプリケーションから監査ログを出力するためのロガークラスです。
例外クラス	監査ログの API で発生する例外を表す例外クラスです。

監査ログ出力で使用する API を使用する場合，次の JAR ファイルをクラスパスに指定してコンパイルする必要があります。

Windows の場合

```
%COSMINEXUS_HOME%¥common¥lib¥auditlog.jar
```

UNIX の場合

```
/opt/Cosminexus/common/lib/auditlog.jar
```

11.2 AuditLogRecord クラス

説明

監査ログのレコードを表すクラスです。
J2EE アプリケーションまたはバッチアプリケーションから監査ログとして出力する情報は，このクラスの set で始まるメソッドを使用して設定します。
監査ログに出力する情報のうち，監査事象の種別，監査事象の結果，および動作情報については，指定できる値が決まっています。このクラスでは，これらの情報を指定するための定数をフィールドとして定義しています。
なお，設定する値の前後に空白を指定した場合，空白は削除されます。また，空文字（""）や空白だけを設定した場合，値は設定されていないものとみなされます。
AuditLogRecord クラスのパッケージ名は，com.hitachi.software.auditlog です。

形式

```
public class AuditLogRecord
  extends Object
  implements Serializable
```

メソッド一覧

メソッド名	機能
getAfterInfo メソッド	変更後情報を取得します。
getAuthority メソッド	権限情報を取得します。
getBeforeInfo メソッド	変更前情報を取得します。
getCategory メソッド	監査事象の種別を取得します。
getDetectionPoint メソッド	検出場所を取得します。
getHaid メソッド	冗長化識別情報を取得します。
getLocation メソッド	ロケーション情報を取得します。
getMessage メソッド	自由記述を取得します。
getMessageId メソッド	メッセージ ID を取得します。
getObjectInfo メソッド	オブジェクト情報を取得します。
getObjectLocation メソッド	オブジェクトロケーション情報を取得します。
getOperation メソッド	動作情報を取得します。
getOutputPoint メソッド	出力元の場所を取得します。
getReceiverHost メソッド	リクエスト送信先ホストを取得します。
getReceiverPort メソッド	リクエスト送信先ポート番号を取得します。
getResult メソッド	監査事象の結果を取得します。
getSenderHost メソッド	リクエスト送信元ホストを取得します。
getSenderPort メソッド	リクエスト送信元ポート番号を取得します。

メソッド名	機能
getServiceInstance メソッド	サービスインスタンス名を取得します。
getSubjectId メソッド	サブジェクト識別情報を取得します。
getSubjectPoint メソッド	指示元の場所を取得します。
setAfterInfo メソッド	変更後情報を設定します。
setAuthority メソッド	権限情報を設定します。
setBeforeInfo メソッド	変更前情報を設定します。
setCategory メソッド	監査事象の種別を設定します。
setDetectionPoint メソッド	検出場所を設定します。
setHaid メソッド	冗長化識別情報を設定します。
setLocation メソッド	ロケーション情報を設定します。
setMessage メソッド	自由記述を設定します。
setMessageId メソッド	メッセージ ID を設定します。
setObjectInfo メソッド	オブジェクト情報を設定します。
setObjectLocation メソッド	オブジェクトロケーション情報を設定します。
setOperation メソッド	動作情報を設定します。
setOutputPoint メソッド	出力元の場所を設定します。
setReceiverHost メソッド	リクエスト送信先ホストを設定します。
setReceiverPort メソッド	リクエスト送信先ポート番号を設定します。
setResult メソッド	監査事象の結果を設定します。
setSenderHost メソッド	リクエスト送信元ホストを設定します。
setSenderPort メソッド	リクエスト送信元ポート番号を設定します。
setServiceInstance メソッド	サービスインスタンス名を設定します。
setSubjectId メソッド	サブジェクト識別情報を設定します。
setSubjectPoint メソッド	指示元の場所を設定します。

監査ログのレコードに設定する値の推奨値

監査ログのレコードに設定する値には，項目ごとに推奨されている長さおよび文字種があります。監査ログの各項目の推奨値を次の表に示します。

表 11-2 監査ログのレコードに設定する値の推奨値

項目	推奨値	
	長さ（バイト）	文字種
変更後情報	規定はありません。	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）

項目	推奨値	
	長さ（バイト）	文字種
権限情報	0 ～ 128	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
変更前情報	規定はありません。	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
監査事象の種別	種別によって異なります。	「A ～ Z」（%x41-5A）および「a ～ z」（/%x61-7A）のアルファベット。なお，アプリケーションサーバでは，監査事象の種別の推奨値をフィールドとして定義しています。「表 11-3　監査事象の種別の推奨値を表すフィールドの一覧」を参照してください。
検出場所	0 ～ 255	次の文字。 •「0 ～ 9」の数値（%x30-39） •「A ～ Z」（%x41-5A）および「a ～ z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
冗長化識別情報	1 ～ 2	「0 ～ 9」の数値（%x30-39）。
ロケーション情報	0 ～ 32	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
自由記述	規定はありません。	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
メッセージ ID	9 ～ 32	次の文字。 •「0 ～ 9」の数値（%x30-39） •「A ～ Z」のアルファベット（%x41-5A） •「-」（%x2D）
オブジェクト情報	0 ～ 256	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
オブジェクトロケーション情報	規定はありません。	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）

項目	推奨値	
	長さ（バイト）	文字種
動作情報	0 ～ 32	任意。 なお，アプリケーションサーバでは，動作情報の推奨値をフィールドとして定義しています。「表 11-4 動作情報の推奨値を表すフィールドの一覧」を参照してください。
出力元の場所	0 ～ 255	次の文字。 •「0 ～ 9」の数値（%x30-39） •「A ～ Z」（%x41-5A）および「a ～ z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
リクエスト送信先ホスト	0 ～ 255	次の文字。 •「0 ～ 9」の数値（%x30-39） •「A ～ Z」（%x41-5A）および「a ～ z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
リクエスト送信先ポート番号	1 ～ 5	「0 ～ 9」の数値（%x30-39）。
監査事象の結果	結果によって異なります。	「A ～ Z」（%x41-5A）および「a ～ z」（%x61-7A）のアルファベット。 なお，アプリケーションサーバでは，監査事象の結果の推奨値をフィールドとして定義しています。「表 11-6　監査事象の結果の推奨値を表すフィールドの一覧」を参照してください。
リクエスト送信元ホスト	0 ～ 255	次の文字。 •「0 ～ 9」の数値（%x30-39） •「A ～ Z」（%x41-5A）および「a ～ z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
リクエスト送信元ポート番号	1 ～ 5	「0 ～ 9」の数値（%x30-39）。
サービスインスタンス名	0 ～ 128	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
サブジェクト識別情報	0 ～ 256	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）

項目	推奨値	
	長さ（バイト）	文字種
指示元の場所	0 ～ 255	次の文字。 • 「0 ～ 9」の数値（%x30-39） • 「A ～ Z」（%x41-5A）および「a ～ z」（%x61-7A）のアルファベット • 「-」（%x2D） • 「.」(%x2E)

監査事象の種別の推奨値を表すフィールド

監査事象の種別の推奨値を表すフィールドの一覧を，次の表に示します。監査事象の種別は，setCategory メソッドで設定します。

表 11-3 監査事象の種別の推奨値を表すフィールドの一覧

フィールド名	実際の値	意味
public static final String CATEGORY_ACCESS_CONTROL	"AccessControl"	管理者または一般利用者が，管理リソースまたはセキュリティリソースへのアクセスを試みて，成功または失敗したことを示す事象です。
public static final String CATEGORY_ANOMALY_EVENT	"AnomalyEvent"	しきい値オーバーなどの異常が発生したことを示す事象です。
public static final String CATEGORY_AUTHENTICATION	"Authentication"	管理者または一般利用者が，認証を試みて，成功または失敗したことを示す事象です。
public static final String CATEGORY_CONFIGURATION_ACCESS	"ConfigurationAccess"	管理者が許可された運用操作を実行して，操作が正常終了または失敗したことを示す事象です。
public static final String CATEGORY_CONTENT_ACCESS	"ContentAccess"	重要なデータへのアクセスを試みて，成功または失敗したことを示す事象です。
public static final String CATEGORY_EXTERNAL_SERVICE	"ExternalService"	外部サービスとの通信結果を示す事象です。
public static final String CATEGORY_FAILURE	"Failure"	ソフトウェアの異常を示す事象です。
public static final String CATEGORY_LINK_STATUS	"LinkStatus"	機器間のリンク状態を示す事象です。
public static final String CATEGORY_MAINTENANCE	"Maintenance"	保守作業を実行して，操作が正常終了または失敗したことを示す事象です。

フィールド名	実際の値	意味
public static final String CATEGORY_MANAGEMENT_ACTION	"ManagementAction"	プログラムの重要なアクションが実行されたことを示す事象です。 ほかの監査事象を契機として実行するアクションを示します。
public static final String CATEGORY_START_STOP	"StartStop"	ソフトウェアの起動と終了を示す事象です。

動作情報の推奨値を表すフィールド

動作情報の推奨値を表すフィールドの一覧を，次の表に示します。動作情報は，setOperation メソッドで設定します。

表 11-4　動作情報の推奨値を表すフィールドの一覧

フィールド名	実際の値	意味
public static final String OPERATION_ADD	"Add"	動作情報の意味は監査事象の種別との組み合わせで決まります。監査事象の種別と組み合わせた動作情報の意味については，「表 11-5 監査事象の種別と動作情報の組み合わせ」を参照してください。
public static final String OPERATION_BACKUP	"Backup"	
public static final String OPERATION_DELETE	"Delete"	
public static final String OPERATION_DOWN	"Down"	
public static final String OPERATION_ENFORCE	"Enforce"	
public static final String OPERATION_INSTALL	"Install"	
public static final String OPERATION_INVOKE	"Invoke"	
public static final String OPERATION_LOGIN	"Login"	
public static final String OPERATION_LOGOFF	"Logoff"	
public static final String OPERATION_LOGON	"Logon"	
public static final String OPERATION_LOGOUT	"Logout"	
public static final String OPERATION_MAINTAIN	"Maintain"	
public static final String OPERATION_NOTIFY	"Notify"	
public static final String OPERATION_OCCUR	"Occur"	

フィールド名	実際の値	意味
public static final String OPERATION_RECEIVE	"Receive"	
public static final String OPERATION_REFER	"Refer"	
public static final String OPERATION_REQUEST	"Request"	
public static final String OPERATION_RESPONSE	"Response"	
public static final String OPERATION_SEND	"Send"	
public static final String OPERATION_START	"Start"	
public static final String OPERATION_STOP	"Stop"	
public static final String OPERATION_UNINSTALL	"Uninstall"	
public static final String OPERATION_UP	"Up"	
public static final String OPERATION_UPDATE	"Update"	

動作情報の意味は監査事象の種別との組み合わせで決まります。監査事象の種別と，動作情報の組み合わせについて，次の表に示します。

表 11-5　監査事象の種別と動作情報の組み合わせ

監査事象の種別	動作情報	意味
StartStop	Start	開始または起動を表します。
	Stop	終了または停止を表します。
Authentication	Login	ログインを表します。
	Logout	ログアウトを表します。
	Logon	ログオンを表します。
	Logoff	ログオフを表します。
AccessControl	Enforce	実施を表します。
ConfigurationAccess	Refer	設定情報の参照を表します。
	Add	設定情報の追加を表します。
	Update	設定情報の更新を表します。
	Delete	設定情報の削除を表します。
Failure	Occur	発生を表します。

監査事象の種別	動作情報	意味
LinkStatus	Up	リンク活性を表します。
	Down	リンク非活性を表します。
ExternalService	Request	要求を表します。
	Response	応答を表します。
	Send	発信を表します。
	Receive	受信を表します。
ContentAccess	Refer	参照を表します。
	Add	追加を表します。
	Update	更新またはアップデートを表します。
	Delete	削除を表します。
Maintenance	Install	インストールを表します。
	Uninstall	アンインストールを表します。
	Update	更新またはアップデートを表します。
	Backup	バックアップを表します。
	Maintain	保守作業を表します。
AnomalyEvent	Occur	発生を表します。
ManagementAction	Invoke	管理者などの呼び出しを表します。
	Notify	管理者などへの通知を表します。

監査事象の結果の推奨値を表すフィールド

監査事象の結果の推奨値を表すフィールドの一覧を，次の表に示します。監査事象の結果は，setResult メソッドで設定します。

表 11-6 監査事象の結果の推奨値を表すフィールドの一覧

フィールド名	実際の値	意味
public static final String RESULT_FAILURE	"Failure"	事象の失敗を表します。
public static final String RESULT_OCCURRENCE	"Occurrence"	成功，失敗の分類がない事象の発生を表します。
public static final String RESULT_SUCCESS	"Success"	事象の成功を表します。

getAfterInfo メソッド

説明

変更後情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getAfterInfo();
```

パラメタ

なし

例外

なし

戻り値

変更後情報が返されます。

getAuthority メソッド

説明

権限情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getAuthority();
```

パラメタ

なし

例外

なし

戻り値

権限情報が返されます。

getBeforeInfo メソッド

説明

変更前情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は

出力されません。この場合，タグも出力されません。

形式

```
public String getBeforeInfo();
```

パラメタ

なし

例外

なし

戻り値

変更前情報が返されます。

getCategory メソッド

説明

監査事象の種別を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public String getCategory();
```

パラメタ

なし

例外

なし

戻り値

監査事象の種別が返されます。

getDetectionPoint メソッド

説明

検出場所を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getDetectionPoint();
```

パラメタ

なし

例外

なし

戻り値

検出場所が返されます。

getHaid メソッド

説明

冗長化識別情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getHaid();
```

パラメタ

なし

例外

なし

戻り値

冗長化識別情報が返されます。

getLocation メソッド

説明

ロケーション情報を取得します。

この項目に値を設定していない場合，アプリケーションサーバによって自動的に付加された値（性能解析トレースのルートアプリケーション情報）が出力されます。

形式

```
public String getLocation();
```

パラメタ

なし

例外

なし

戻り値

ロケーション情報が返されます。

getMessage メソッド

説明

自由記述を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getMessage();
```

パラメタ

なし

例外

なし

戻り値

自由記述が返されます。

getMessageId メソッド

説明

メッセージ ID を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public String getMessageId();
```

パラメタ

なし

例外

なし

戻り値

メッセージ ID が返されます。

getObjectInfo メソッド

説明

オブジェクト情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getObjectInfo();
```

パラメタ

なし

例外

なし

戻り値

オブジェクト情報が返されます。

getObjectLocation メソッド

説明

オブジェクトロケーション情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getObjectLocation();
```

パラメタ

なし

例外

なし

戻り値

オブジェクトロケーション情報が返されます。

getOperation メソッド

説明

動作情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getOperation();
```

パラメタ

なし

例外

なし

戻り値

動作情報が返されます。

getOutputPoint メソッド

説明

出力元の場所を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getOutputPoint();
```

パラメタ

なし

例外

なし

戻り値

出力元の場所が返されます。

getReceiverHost メソッド

説明

リクエスト送信先ホストを取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getReceiverHost();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信先ホストが返されます。

getReceiverPort メソッド

説明

リクエスト送信先ポート番号を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public int getReceiverPort();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信先ポート番号が返されます。

getResult メソッド

説明

監査事象の結果を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public String getResult();
```

パラメタ

なし

例外

なし

戻り値

監査事象の結果が返されます。

getSenderHost メソッド

説明

リクエスト送信元ホストを取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getSenderHost();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信元ホストが返されます。

getSenderPort メソッド

説明

リクエスト送信元ポート番号を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public int getSenderPort();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信元ポート番号が返されます。

getServiceInstance メソッド

説明

サービスインスタンス名を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getServiceInstance();
```

パラメタ

なし

例外

なし

戻り値

サービスインスタンス名が返されます。

getSubjectId メソッド

説明

サブジェクト識別情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にアプリケーションサーバによって自動的に付加された値（OS のアカウント）が出力されます。なお，OS のアカウントとして出力される情報は，OS ごとに異なります。

Windows の場合
　ユーザ名が出力されます。

UNIX の場合
　実効ユーザ ID が出力されます。

形式

```
public String getSubjectId();
```

パラメタ

なし

例外

なし

戻り値

サブジェクト識別情報が返されます。

getSubjectPoint メソッド

説明

指示元の場所を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getSubjectPoint();
```

パラメタ

なし

例外

なし

戻り値

指示元の場所が返されます。

setAfterInfo メソッド

説明

変更後情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setAfterInfo(String info);
```

パラメタ

info：

変更後情報を指定します。

例外

なし

戻り値

なし

setAuthority メソッド

説明

権限情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setAuthority(String authority);
```

パラメタ

authority：
　権限情報を指定します。

例外

なし

戻り値

なし

setBeforeInfo メソッド

説明

変更前情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setBeforeInfo(String info);
```

パラメタ

info：
　変更前情報を指定します。

例外

なし

戻り値

なし

setCategory メソッド

説明

監査事象の種別を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public void setCategory(String category);
```

パラメタ

category：
　監査事象の種別を指定します。

例外

なし

戻り値

なし

setDetectionPoint メソッド

説明

検出場所を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setDetectionPoint(String detectionPoint);
```

パラメタ

detectionPoint：
　検出場所を指定します。

例外

なし

戻り値

なし

setHaid メソッド

説明

冗長化識別情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setHaid(String haid);
```

パラメタ

haid：
　冗長化識別情報を指定します。

例外

なし

戻り値

なし

setLocation メソッド

説明

ロケーション情報を設定します。

この項目に値を設定していない場合，アプリケーションサーバによって，性能解析トレースのルートアプリケーション情報が設定されます。

形式

```
public void setLocation(String location);
```

パラメタ

location：
　ロケーション情報を指定します。

例外

なし

戻り値

なし

setMessage メソッド

説明

自由記述を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setMessage(String message);
```

パラメタ

message：
　自由記述を指定します。

例外

なし

戻り値

なし

setMessageId メソッド

説明

メッセージ ID を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public void setMessageId(String messageId);
```

パラメタ

messageId：
　メッセージ ID をで指定します。

例外

なし

戻り値

なし

setObjectInfo メソッド

説明

オブジェクト情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setObjectInfo(String objectInfo);
```

パラメタ

objectInfo：
　オブジェクト情報を指定します。

例外

なし

戻り値

なし

setObjectLocation メソッド

説明

オブジェクトロケーション情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setObjectLocation(String objectLocation);
```

パラメタ

objectLocation：
　オブジェクトロケーション情報を指定します。

例外

なし

戻り値

なし

setOperation メソッド

説明

動作情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setOperation(String operation);
```

パラメタ

operation：
　動作情報を指定します。

例外

なし

戻り値

なし

setOutputPoint メソッド

説明

出力元の場所を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setOutputPoint(String outputPoint);
```

パラメタ

outputPoint：
　出力元の場所を指定します。

例外

なし

戻り値

なし

setReceiverHost メソッド

説明

リクエスト送信先ホストを設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setReceiverHost(String receiverHost);
```

パラメタ

receiverHost：
　リクエスト送信先ホストを指定します。

例外

なし

戻り値

なし

setReceiverPort メソッド

説明

リクエスト送信先ポート番号を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setReceiverPort(int receiverPort);
```

パラメタ

receiverPort：
　リクエスト送信先ポート番号を指定します。

例外

なし

戻り値

なし

setResult メソッド

説明

監査事象の結果を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public void setResult(String result);
```

パラメタ

result：
　監査事象の結果を指定します。

例外

なし

戻り値

なし

setSenderHost メソッド

説明

リクエスト送信元ホストを設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setSenderHost(String senderHost);
```

パラメタ

senderHost：
　リクエスト送信元ホストを指定します。

例外

なし

戻り値

なし

setSenderPort メソッド

説明

リクエスト送信元ポート番号を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setSenderPort(int senderPort);
```

パラメタ

senderPort：
　リクエスト送信元ポート番号を指定します。

例外

なし

戻り値

なし

setServiceInstance メソッド

説明

サービスインスタンス名を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setServiceInstance(String serviceInstance);
```

パラメタ

serviceInstance：
　サービスインスタンス名を指定します。

例外

なし

戻り値

なし

setSubjectId メソッド

説明

サブジェクト識別情報を設定します。指定できるのは，アカウント識別子（ユーザ ID）だけです。

この項目に値を設定していない場合，または null を設定した場合は，OS のアカウントが設定されます。

OS のアカウントとして設定される情報は，OS ごとに異なります。

Windows の場合
　ユーザ名が設定されます。

UNIX の場合
　実効ユーザ ID が設定されます。

なお，パラメタに，OS のアカウントを指定してはいけません。

形式

```
public void setSubjectId(String subjectId);
```

パラメタ

subjectId：
　サブジェクト識別情報（ユーザ ID）を指定します。

例外

なし

戻り値

なし

setSubjectPoint メソッド

説明

指示元の場所を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setSubjectPoint(String subjectPoint);
```

パラメタ

subjectPoint：
　　指示元の場所を指定します。

例外

なし

戻り値

なし

11.3 UserAuditLogger クラス

説明

J2EE アプリケーションまたはバッチアプリケーションから監査ログを出力するためのロガークラスです。
UserAuditLogger クラスのパッケージ名は，com.hitachi.software.auditlog です。

形式

```
public class UserAuditLogger
```

メソッド一覧

メソッド名	機能
getLogger メソッド	パラメタに指定した発生コンポーネント名ごとに，UserAuditLogger オブジェクトを取得します。
isEnabled メソッド	監査ログが有効か無効かを判定します。
isLoggable メソッド	パラメタに指定したメッセージ ID から，監査ログの出力要否を判定します。
log メソッド	パラメタに指定した AuditLogRecord オブジェクトを基に，監査ログを出力します。

getLogger メソッド

説明

パラメタに指定した発生コンポーネント名ごとに，UserAuditLogger オブジェクトを取得します。例えば，getLogger(new String("Component")) メソッドを 2 回実行した場合は，2 回とも同じ UserAuditLogger オブジェクトを取得できます。

発生コンポーネント名と UserAuditLogger オブジェクトの対応は，UserAuditLogger クラスの内部で管理されています。内部で管理されている発生コンポーネント名と，パラメタに指定した発生コンポーネント名の判定は，String.equals(component) メソッドで実行されます。内部で管理されている発生コンポーネント名とパラメタに指定した発生コンポーネント名が一致した場合に，対応する UserAuditLogger オブジェクトが返されます。

形式

```
public static UserAuditLogger getLogger(String component);
```

パラメタ

component：
　発生コンポーネント名を指定します。

例外

com.hitachi.software.auditlog.AuditLogException：
初期化に失敗しました。

戻り値

UserAuditLogger：
監査ログが有効な場合に，UserAuditLogger オブジェクトが返されます。
パラメタに指定した発生コンポーネント名ごとに，異なるオブジェクトが返されます。

null：
監査ログが無効な場合に返されます。

isEnabled メソッド

説明

監査ログが有効か無効かを判定します。

なお，監査ログ定義ファイルの auditlog.enabled キーに false が指定されている場合，監査ログは無効になります。

形式

```
public static boolean isEnabled();
```

パラメタ

なし

例外

なし

戻り値

true：
監査ログが有効な場合に返されます。

false：
監査ログが無効な場合に返されます。

isLoggable メソッド

説明

パラメタに指定したメッセージ ID から，監査ログの出力要否を判定します。

出力要否は，監査ログ定義ファイルの auditlog.filtered.message.list キーに，指定したメッセージ ID が含まれているかどうかで判定されます。

auditlog.filtered.message.list キーに，指定したメッセージ ID が含まれている場合は，戻り値として false が返され，監査ログを出力できません。
auditlog.filtered.message.list キーに，指定したメッセージ ID が含まれていない場合は，戻り値として true が返され，監査ログを出力できます。

なお，このメソッドでは，監査ログが有効か無効かについては判定しません。

形式

```
public boolean isLoggable(String messageId);
```

パラメタ

messageId：

メッセージ ID を指定します。必ず一つのメッセージ ID を指定してください。
例えば，「String messageId = "message-1,message-2";」のように複数のメッセージ ID を指定した場合，一つながりの文字列として認識されます。「String messageId = "message-1";」のように，メッセージ ID を一つだけ指定してください。

例外

なし

戻り値

true：

監査ログを出力できる場合に返されます。パラメタに「null」または空文字を指定した場合も true が返されます。

false：

監査ログを出力できない場合に返されます。

log メソッド

説明

パラメタに指定した AuditLogRecord オブジェクトを基に，監査ログを出力します。出力される監査ログは，AuditLogRecord クラスの set で始まるメソッドで設定した値で

す。

監査ログのレコードは，出力項目ごとに推奨値が決められています。出力項目ごとの推奨値については，「11.2　AuditLogRecord クラス」を参照してください。ただし，推奨されている文字種以外の文字を設定したり，推奨されている文字数を超過して設定したりした場合も，設定した値がそのまま出力されます。

監査ログの出力先は，監査ログ定義ファイルで指定します。監査ログ定義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(サーバ定義)」の「13.2　監査ログ定義ファイル」を参照してください。また，監査ログの出力形式については，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「6.4.3　監査ログの出力形式」を参照してください。

すでに監査ログを出力したファイルが存在する場合，出力した監査ログには，そのファイルのアクセス権限が引き継がれます。監査ログを出力したファイルが存在しない場合は，監査ログを出力したファイルに対して，次のアクセス権限が設定されます。

表 11-7　監査ログを出力したファイルのアクセス権限

OS	所有者	設定されるアクセス権限
Windows の場合	出力先のフォルダの設定が引き継がれます。	
UNIX の場合	監査ログを出力したプロセスのユーザおよびプライマリグループ	666※

注※

umask によるマスクが行われます。umask=0022 が設定されている場合，実際には 644 となります。

形式

```
public void log(AuditLogRecord)
  throws AuditLogException;
```

パラメタ

AuditLogRecord：

AuditLogRecord オブジェクトを指定します。

例外

AuditLogException：

監査ログの出力が失敗しました。次の要因が考えられます。

- 監査ログを出力するプロセスの実行ユーザに，監査ログの書き込み権限がない。
- 監査ログを出力する際にディスクフルになった。
- 指定が必要な出力項目が指定されていない。

戻り値

なし

11.4 例外クラス

監査ログ出力で使用する API で発生する例外を表す，例外クラスについて説明します。

例外名

com.hitachi.software.auditlog.AuditLogException

内容

監査ログに関する例外を表すクラスです。
コンストラクタの型は Exception クラスと同じです。また，メソッドについても Exception クラスのすべてのメソッドを継承しています。

コンストラクタ

- AuditLogException()
- AuditLogException(String message)
- AuditLogException(String message, Throwable cause)
- AuditLogException(Throwable cause)

これらのコンストラクタでは，Exception クラスのコンストラクタが呼び出されます。

メソッド

AuditLogException クラスおよび Exception クラスで定義されたメソッドはありません。
Exception クラスが Throwable クラスから継承したメソッドだけが使用できます。

12 性能解析トレースで使用する API

この章では，性能解析トレースのルートアプリケーション情報取得機能で使用する API について説明します。

12.1　性能解析トレースで使用する API の一覧

性能解析トレースで使用する API の一覧を次の表に示します。

表 12-1　性能解析トレースで使用する API の一覧

クラス名	機能
CprfTrace クラス	性能解析トレース関連の機能を提供します。

12.2　CprfTrace クラス

説明

性能解析トレース関連の機能を提供します。
CprfTrace クラスのパッケージ名は，com.hitachi.software.ejb.application.prf です。

メソッド一覧

メソッド名	機能
getRootApInfo メソッド	現在のスレッドが保持している性能解析トレースのルートアプリケーション情報を文字列表現で返します。

注意事項

このクラスを使用する場合は，次の JAR ファイルをクラスパスに指定してコンパイルします。

- Windows の場合
  <アプリケーションサーバのインストールディレクトリ >¥CC¥lib¥ejbserver.jar
- UNIX の場合
  /opt/Cosminexus/CC/lib/ejbserver.jar

getRootApInfo メソッド

説明

現在のスレッドが保持している性能解析トレースのルートアプリケーション情報を文字列表現で返します。

ルートアプリケーション情報の文字列表現とは，ルートアプリケーション情報を構成する IP アドレス，プロセス ID，および通信番号をスラッシュ (/) で区切ったものです（最大長 45）。

例：`"10.209.15.130/1234/0x0000000000000001"`

ルートアプリケーション情報の文字列表現をログファイルなどに記録することによって，任意のタイミングで性能解析トレースファイルとの突き合わせが可能となるため，トラブルシュートに役立てることができます。

性能解析トレースのルートアプリケーション情報取得については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編」の「6.4　性能解析トレースのルートアプリケーション情報取得のための実装」を参照してください。
ルートアプリケーション情報を利用したログ調査については，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「6.6.7　ルートアプリケーション情報を利用したログ調査」を参照してください。

形式

```
public static final String getRootApInfo();
```

パラメタ

なし

例外

なし

戻り値

ルートアプリケーション情報の文字列表現。

次の場合には，null を返します。

- Cosminexus Performance Tracer がインストールされていないなど，現在のスレッドが保持しているルートアプリケーション情報が存在しない場合
- EJB コンテナ外および Web コンテナ外から呼び出された場合
- EJB クライアントから呼び出された場合

13 JavaVM で使用する API

この章では，日立の JavaVM で使用する API について説明します。
なお，日立の JavaVM は，J2SE 5.0 および Java SE 6 に準拠しています。詳細については，マニュアル「Cosminexus アプリケーションサーバ 概説」を参照してください。また，JDK 5.0 および JDK 6 で使用できる API については，Sun Microsystems 社が提供している JDK 5.0 および JDK 6 のドキュメントを参照してください。

13.1　JavaVM で使用する API の一覧

JavaVM で使用する API の一覧を，次の表に示します。

表 13-1　JavaVM で使用する API の一覧

クラス名	機能
BasicExplicitMemory クラス	Explicit メモリブロックを表すクラスです。なお，このクラスは，ほかの JDK 製品では利用できません。
ExplicitMemory クラス	Explicit メモリブロックを表す抽象クラスです。 派生クラスである BasicExplicitMemory クラスで処理する内容を定義します。なお，このクラスは，ほかの JDK 製品では利用できません。
MemoryArea クラス	Explicit メモリブロックまたは Java ヒープを表す抽象クラスです。なお，このクラスは，ほかの JDK 製品では利用できません。
MemoryInfo クラス	ガーベージコレクションのメモリ情報を取得します。なお，このクラスは，ほかの JDK 製品では利用できません。
例外クラス	JavaVM で使用する API で発生する例外を表す例外クラスです。なお，このクラスは，ほかの JDK 製品では利用できません。

パッケージ名の省略

なお，この章では java.lang および MemoryArea パッケージに所属するクラスのクラス名は，完全名ではなくクラス名だけを記載します。

例

java.lang.Object の場合：Object

13.2　BasicExplicitMemory クラス

説明

アプリケーションサーバが生成した Explicit メモリブロックを表すクラスです。
ExplicitMemory クラスを継承しています。
BasicExplicitMemory クラスのパッケージは，JP.co.Hitachi.soft.jvm.MemoryArea です。

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
BasicExplicitMemory コンストラクタ（形式 1）	Explicit メモリブロックを初期化します。
BasicExplicitMemory コンストラクタ（形式 2）	Explicit メモリブロックを初期化して名称を設定します。
getName メソッド	Explicit メモリブロック名称を返却します。

BasicExplicitMemory コンストラクタ（形式 1）

説明

Explicit メモリブロックを初期化します。初期化することで，オブジェクトを Explicit ヒープに配置できます。

Explicit メモリブロックを初期化して，インスタンスの名称を「BasicExplicitMemory-<Explicit メモリブロックの ID>」にします。ただし，Explicit メモリブロックのメモリ領域は確保しません。

次の条件に当てはまる場合は，無効な Explicit メモリブロックにします。

- オプション HitachiUseExplicitMemory が OFF の場合
  （-XX:+HitachiUseExplicitMemory を指定していない場合）
- Explicit メモリブロックの最大数を超えた場合

形式

```
public BasicExplicitMemory();
```

パラメタ

なし

例外

なし

戻り値

なし

BasicExplicitMemory コンストラクタ（形式 2）

説明

Explicit メモリブロックを初期化します。また，初期化と同時にこの Explicit メモリブロックの名称を設定します。

インスタンスの名称はパラメタ name になります。ただし，パラメタ name が null の場合は，インスタンスの名称は「BasicExplicitMemory-<Explicit メモリブロックの ID>」になります。

次の条件に当てはまる場合は無効な Explicit メモリブロックにします。

- オプション HitachiUseExplicitMemory が OFF の場合
  （-XX:+HitachiUseExplicitMemory を指定していない場合）
- Explicit メモリブロックの最大数を超えた場合

形式

```
public BasicExplicitMemory(String name)
```

パラメタ

name：
　名称を表す文字列（String）を指定します。

例外

なし

戻り値

なし

getName メソッド

説明

このオブジェクトが表す Explicit メモリブロックの名称を返却します。

形式

```
public String getName();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが表す Explicit メモリブロックの名称をこのオブジェクトの名称を表すインスタンスフィールドから取得し，その参照を返却します。

注意事項

デフォルトで設定されている名前に付いている Explicit メモリブロックの ID の一意性は保証されています。ただし，その ID を持つインスタンスが破棄されたとき，同じ Explicit メモリブロックの ID が再利用される場合があります。この場合，同時に存在することがない異なるインスタンスが同じデフォルト名を持つことがあります。

13.3　ExplicitMemory クラス

説明

Explicit メモリブロックを表す抽象クラスです。BasicExplicitMemory での処理を定義します。

ExplicitMemory クラスのパッケージは，JP.co.Hitachi.soft.jvm.MemoryArea です。

メソッド一覧

メソッド名	機能
countExplicitMemories メソッド	Explicit メモリブロック数を返却します。
freeMemory メソッド	Explicit メモリブロックの利用できるサイズを返却します。
getMemoryUsage メソッド	Explicit ヒープの利用状況を返却します。
isActive メソッド	Explicit メモリブロックが処理できるかどうかを返却します。
isReclaimed メソッド	Explicit メモリブロックが解放予約状態または解放済み状態かどうかを返却します。
newArray メソッド（形式 1）	Explicit メモリブロックに配列オブジェクトを生成します。
newArray メソッド（形式 2）	
newInstance メソッド（形式 1）	Explicit メモリブロックにオブジェクトを生成します。
newInstance メソッド（形式 2）	
newInstance メソッド（形式 3）	
reclaim メソッド（形式 1）	Explicit メモリブロックを解放予約します。
reclaim メソッド（形式 2）	
reclaim メソッド（形式 3）	
reclaim メソッド（形式 4）	
setName メソッド	Explicit メモリブロック名称を name に設定します。
toString メソッド	Explicit メモリブロック名称を返却します。
totalMemory メソッド	Explicit メモリブロックの確保済みサイズを返却します。
usedMemory メソッド	Explicit メモリブロックの使用されているメモリのサイズを返却します。

countExplicitMemories メソッド

説明

Explicit ヒープにある Explicit メモリブロックの個数を返却します。Explicit メモリブロックの状態が解放済み，または無効の場合はカウントしません。

形式

```
public static int countExplicitMemories();
```

パラメタ

なし

例外

なし

戻り値

Explicit ヒープにある Explicit メモリブロックの個数をカウントして，int 型で返却します。

注意事項

- 解放予約済み状態の Explicit メモリブロックをカウントするため，Explicit メモリブロックを明示的に操作しなくても，時間経過によって個数が変化する場合があります。
- ExplicitMemory インスタンスの個数を数えるのではなく，Explicit メモリブロックの実体数を数えるメソッドです。

freeMemory メソッド

説明

Explicit メモリブロックが利用できるメモリサイズを返却します。

形式

```
public long freeMemory();
```

パラメタ

なし

例外

InaccessibleMemoryAreaException：

　サポートされていない機能です。

戻り値

共通エラーチェックをして，次のどちらかの値を返却します。共通エラーチェックについては，「13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

0：

　API の処理ができない場合に返却します。

このオブジェクトが表す Explicit メモリブロックが利用できるメモリサイズ（バイト数）：

API の処理ができる場合は，このオブジェクトが表す Explicit メモリブロックが利用できるメモリサイズを long 型で返却します。

getMemoryUsage メソッド

説明

Explicit ヒープの利用状況を返却します。

形式

```
public static java.lang.management.MemoryUsage getMemoryUsage();
```

パラメタ

なし

例外

なし

戻り値

Explicit ヒープの利用状況をフィールドとして保持している java.lang.management.MemoryUsage インスタンスへの参照を，次に示す値で返却します。

init：

Explicit ヒープの初期値です。常に 0 となります。

used：

Explicit ヒープの使用されているメモリサイズ（バイト数）です。

committed：

Explicit ヒープの確保済みサイズ（バイト数）です。

max：

-XX:HitachiExplicitHeapMaxSize で指定した最大 Explicit ヒープサイズの値（バイト数）です。ただし，オプション HitachiUseExplicitMemory が OFF の場合（-XX:+HitachiUseExplicitMemory を指定した場合）は 0 を返却します。

注意事項

MemoryUsage インスタンスが持つ値は getMemoryUsage() を呼び出した時点での値です。MemoryUsage インスタンスから各フィールドを読み出した時点では，実際の値と異なる場合があります。

isActive メソッド

説明

このオブジェクトが表す Explicit メモリブロックが処理できる状態であるかどうかを返却します。

形式

```
public boolean isActive();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが表す Explicit メモリブロックの状態を次に示す boolean 型で返却します。

true：

処理できる状態です。有効な Explicit メモリブロックで，サブ状態が Enable の場合に返却します。

false：

処理できない状態です。次のどちらかの状態の場合にこの値を返却します。

- 無効になっている Explicit メモリブロックの場合
- 有効な Explicit メモリブロックで，サブ状態が Disable の場合

注意事項

ExplicitMemory は，一度無効状態になった場合，再度有効状態になることはありません。

isReclaimed メソッド

説明

このオブジェクトが表す Explicit メモリブロックが解放予約状態または解放済み状態であるかどうかを返却します。

形式

```
public boolean isReclaimed();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが表す Explicit メモリブロックの状態を次に示す boolean 型で返却します。

true：
　このオブジェクトが表す Explicit メモリブロックが解放予約状態または解放済み状態の場合に返却します。

false：
　このオブジェクトが表す Explicit メモリブロックが有効な状態の場合に返却します。

newArray メソッド（形式 1）

説明

パラメタ type の表すクラスのパラメタ length 長の配列インスタンスをこのオブジェクトが表す Explicit メモリブロックに直接生成します。

形式

```
public Object newArray(Class type, int length);
```

パラメタ

type：
　直接生成する配列インスタンスのクラスを指定します。

length：
　直接生成する配列インスタンスの長さを指定します。

例外

NullPointerException：
　パラメタ type が null です。

NegativeArraySizeException：
　パラメタ length が 0 未満です。

IllegalArgumentException：

パラメタ length が 0 以上で，パラメタ type が 255 次元以上の配列クラスまたは Void.TYPE です。

InaccessibleMemoryAreaException：
サポートされていない機能です。

戻り値

type 型の配列であるこのオブジェクトが示す Explicit メモリブロックに直接生成し，その参照を返却します。配列の長さは length 長です。

共通エラーチェックによって，処理できないと判定された場合は，java.lang.reflect.Array.newInstance(Class<?> componentType,int length) をパラメタ type，パラメタ length で呼び出し，その結果を返却します。共通エラーチェックについては，「13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

newArray メソッド（形式 2）

説明

dimensions.length 次元の配列インスタンスをこのオブジェクトが表す Explicit メモリブロックに直接生成します。なお，パラメタ type の表すクラスの n 次元目の要素数は dimensions[n-1] です。

形式

```
public Object newArray(Class type, int[] dimensions);
```

パラメタ

type：
直接生成する配列インスタンスのクラスを指定します。

dimensions：
直接生成する配列インスタンスの次元数および要素数を指定します。

例外

NullPointerException：
パラメタ dimensions またはパラメタ type のうち，どちらかのパラメタの値または両方のパラメタの値が null です。

NegativeArraySizeException：
パラメタ dimensions が負の値を要素に持っています。

IllegalArgumentException：

次のどれかに当てはまる場合にスローされます。

- パラメタ dimensions の dimensions.length が 0 以下または 255 より大きい値の場合
- パラメタ type の次元数とパラメタ dimensions の dimensions.length との合計が 255 より大きい値の場合
- パラメタ type が Void.TYPE である場合

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

n 次元目の要素数が dimensions[n-1] である，dimensions.length 次元の type 型の配列オブジェクトをこのオブジェクトが表す Explicit メモリブロックに直接生成し，参照を返却します。

共通エラーチェックによって，処理できないと判定された場合は，java.lang.reflect.Array.newInstance (Class<?> componentType,int[] dimensions) をパラメタ type，パラメタ dimensions で呼び出し，その結果を返却します。共通エラーチェックについては，「13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

newInstance メソッド（形式 1）

説明

パラメタ type の表すクラスのインスタンスをこのオブジェクトが表す Explicit メモリブロックに直接生成します。パラメタで指定したクラスのインスタンスだけを Explicit メモリブロックに生成します。パラメタで指定したクラスのインスタンスのコンストラクタなどによる初期化で生成されるオブジェクトについては，Java ヒープに生成します。type をこのオブジェクトとした場合の Class.newInstance() と類似していますが，一部異なります。

形式

```
public Object newInstance(Class type);
```

パラメタ

type：

直接生成する配列インスタンスのクラスです。

例外

NullPointerException：

パラメタ type，またはパラメタ type が表すクラスが null です。

SecurityException：

SecurityManager があり，かつ次のうちどれかに当てはまる場合にスローされます。

- s.checkMemberAccess(type, Member.PUBLIC) の呼び出しがこのコンストラクタへのアクセスを許可しない。
- 呼び出し側のクラスローダが異なる。
- 現在のクラスローダの上位クラスローダと s.checkPackageAccess() の呼び出しがこのクラスのパッケージへのアクセスを許可しない。

NoSuchMethodException：

パラメタ type またはパラメタ type が表すクラスに，public のパラメタなしコンストラクタがありません。

ExceptionInInitializerError：

パラメタ type またはパラメタ type が表すクラスの初期化に失敗しました。

InstantiationException：

パラメタ type またはパラメタ type が表すクラスが抽象クラスまたはインタフェースです。

InvocationTargetException：

パラメタ type またはパラメタ type が表すクラスのコンストラクタの実行で例外が発生しました。

IllegalAccessException：

クラスまたはその nullary コンストラクタにアクセスできません。

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

このオブジェクトが表す Explicit メモリブロックに生成されたインスタンスへの参照を返却します。

共通エラーチェックで処理できないと判定された場合は，パラメタ type をレシーバとして Class.newInstance() のメソッドを呼び出し，その結果を返却します。共通エラーチェックについては，「13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

注意事項

パラメタ type には，public クラスを与えることを推奨します。

newInstance メソッド（形式 2）

説明

パラメタ type の表すクラスのインスタンスを Explicit メモリブロックに直接生成します。パラメタ args に指定した値がインスタンスを生成するコンストラクタの引数として渡されます。パラメタで指定したクラスのインスタンスのコンストラクタなどによる初期化で生成されるオブジェクトについては，Java ヒープに生成します。

形式

```
public Object newInstance(Class type, Object... args);
```

パラメタ

type：
　直接生成する配列インスタンスのクラスです。

args：
　コンストラクタに渡すパラメタです。

例外

NullPointerException：
　パラメタ type またはパラメタ args の値のどちらかまたは両方が null です。

SecurityException：
　SecurityManager があり，かつ次のうちどれかに当てはまる場合にスローされます。
- s.checkMemberAccess(type,Member.PUBLIC) の呼び出しがこのコンストラクタへのアクセスを許可しない場合
- 呼び出し側のクラスローダが異なる場合
- 現在のクラスローダの上位クラスローダと s.checkPackageAccess() の呼び出しがこのクラスのパッケージへのアクセスを許可しない場合

NoSuchMethodException：
　パラメタ args の要素と同じ型のパラメタを持つ public コンストラクタがパラメタ type の表すクラスにありません。

ExceptionInInitializerError：
　パラメタ type またはパラメタ type が表すクラスの初期化に失敗しました。

InstantiationException：
　パラメタ type またはパラメタ type が表すクラスが抽象クラスまたはインタフェースです。

InvocationTargetException：
　パラメタ type またはパラメタ type が表すクラスのコンストラクタの実行で例外が

発生しました。

InaccessibleMemoryAreaException：
サポートされていない機能です。

IllegalAccessException：
Constructor オブジェクトが Java 言語アクセス制御を実施するため，基本となるコンストラクタにアクセスできません。

IllegalArgumentException：
次のどれかに当てはまる場合にスローされます。
- 実パラメタ数と仮パラメタ数が異なる場合
- プリミティブ引数のラップ解除変換が失敗した場合
- プリミティブ引数のラップ解除後，パラメタ値を対応する仮パラメタ型に変換できない場合
- コンストラクタが列挙型に関連している場合

戻り値

このオブジェクトが表す Explicit メモリブロックに生成されたインスタンスへの参照を返却します。

共通エラーチェックをして，処理できないと判定した場合，type.getConstructor(arg_types※) として，java.lang.reflect.Constructor インスタンスを取得します。この場合，java.lang.reflect.Constructor をレシーバ，パラメタ args をパラメタとして，java.lang.reflect.Constructor.newInstance(Object... initargs) のメソッドを呼び出し，その結果を返却します。共通エラーチェックについては，「13.6 Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

注※
arg_types は，パラメタ args の各要素をこのオブジェクトとして Object.getClass() を呼び出した結果を要素とする Class 配列です。

注意事項

パラメタ type には，public クラスを与えることを推奨します。

プリミティブ型を引数とするコンストラクタを呼び出すことはでききません。プリミティブ型を引数とするコンストラクタを呼び出す場合は，newInstance メソッド（形式 3）を使用します。次に newInstance メソッド（形式 3）を使用したコード例を示します。

```
import JP.co.Hitachi.soft.jvm.MemoryArea.*;
import java.lang.reflect.*;
public class test1 {
  public static void main(String[] args) throws Exception {
    ExplicitMemory em = new BasicExplicitMemory();
```

```
    TheClass obj = null;

    Constructor cons = TheClass.class.getConstructor(new
Class[]{int.class});
    obj = (TheClass)em.newInstance(cons,1);            // 実行成功

    obj = (TheClass)em.newInstance(TheClass.class,1);  //
NoSuchMethodExceptionをスロー
  }
}

public class TheClass {
  public TheClass(int i){}
}
```

newInstance メソッド（形式 3）

説明

パラメタ cons が表すコンストラクタをパラメタ args で実行し，このオブジェクトが表す Explicit メモリブロックに直接生成します。パラメタで指定したクラスのインスタンスだけを Explicit メモリブロックに生成します。パラメタで指定したクラスのインスタンスのコンストラクタなどによる初期化で生成されるオブジェクトについては，Java ヒープに生成します。

形式

```
public Object newInstance(java.lang.reflect.Constructor cons,
Object... args);
```

パラメタ

cons：

直接生成する配列インスタンスのコンストラクタを指定します。

args：

コンストラクタに渡すパラメタを指定します。

例外

NullPointerException：

パラメタ cons またはパラメタ args の値のどちらかまたは両方が null です。

ExceptionInInitializerError：

パラメタ cons が表すコンストラクタでクラスの初期化に失敗しました。

InstantiationException：

パラメタ cons が表すコンストラクタが抽象クラスです。

IllegalArgumentException：

パラメタ cons が示すコンストラクタのパラメタとパラメタ args が一致しません。

InvocationTargetException：
　パラメタ cons またはパラメタ args が表すコンストラクタの実行で例外が発生しました。

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

このオブジェクトが表す Explicit メモリブロックに生成されたインスタンスへの参照を返却します。

共通エラーチェックで処理できないと判定された場合，java.lang.reflect.Constructor.newInstance(Object... initargs) のパラメタ cons をこのオブジェクト，パラメタ args をパラメタとして呼び出し，その結果を返却します。共通エラーチェックについては，「13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

注意事項

パラメタ cons には，public クラスのコンストラクタを与えることを推奨します。

reclaim メソッド（形式 1）

説明

パラメタ areas のすべての要素に対して解放予約をします。

パラメタ areas が null 以外の場合に，パラメタ areas のすべての要素に対して，要素をパラメタとして ExplicitMemory.reclaim(ExplicitMemory area) を呼び出した場合と同じ処理をします。処理する要素の順序は未定義とします。ある要素に対する処理で例外が発生した場合は，その例外をスローします。例外のスローまでに未処理だった要素に対する処理は実行しません。

形式

```
public static void reclaim(ExplicitMemory... areas);
```

パラメタ

areas：
　要素に解放予約をする Explicit メモリブロックを持つ配列を指定します。

例外

NullPointerException：
　パラメタ areas が null です。

InaccessibleMemoryAreaException：
サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

reclaim メソッド（形式 2）

説明

パラメタ area が null 以外の値の場合に共通エラーチェックで処理できると判定されたとき，パラメタ area に排他処理を実施してから，パラメタ area の表す Explicit メモリブロックを解放予約します。

次の条件に当てはまる場合は，処理しません。

- パラメタ area が null の場合
- 共通エラーチェックで処理できないと判定された場合

形式

```
public static void reclaim(ExplicitMemory area);
```

パラメタ

area：
解放予約をする Explicit メモリブロックを指定します。

例外

InaccessibleMemoryAreaException：
サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

reclaim メソッド（形式 3）

説明

パラメタ area0，area1 の表す Explicit メモリブロックを解放予約します。

パラメタ area0，パラメタ area1 それぞれをパラメタとして ExplicitMemory.reclaim(ExplicitMemory area) を呼び出した場合と同じ処理をします。パラメタ area0，パラメタ area1 の処理順序は未定義とします。一方のパラメタに対する処理で，例外が発生した場合は，その例外をスローします。もう一方のパラメタが未処理である場合，処理はしません。

形式

```
public static void reclaim(ExplicitMemory area0, ExplicitMemory
area1);
```

パラメタ

area0：

解放予約をする Explicit メモリブロックその 1 を指定します。

area1：

解放予約をする Explicit メモリブロックその 2 を指定します。

例外

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

reclaim メソッド（形式 4）

説明

パラメタ areas のすべての要素に対して解放予約をします。

パラメタ areas が null 以外の場合は，パラメタ areas のすべての要素に対して，要素をパラメタとして ExplicitMemory.reclaim(ExplicitMemory area) を呼び出した場合と同じ処理をします。処理する要素の順序は未定義とします。ある要素に対する処理で例外が発生した場合は，その例外をスローします。例外のスローまでに未処理だった要素に対する処理はしません。

形式

```
public static void reclaim(Iterable<ExplicitMemory> areas);
```

パラメタ

areas：

解放予約をする Explicit メモリブロックのイテレータを指定します。

例外

NullPointerException：

パラメタ areas の値が null です。

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された

Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

setName メソッド

説明

Explicit メモリブロックに名称を設定します。このオブジェクトが表す Explicit メモリブロックの名称として，このオブジェクトの名称を表すインスタンスフィールドにパラメタ name を設定します。

この名称は主にデバッグ用に設定します。設定した値はイベントログまたはスレッドダンプで表示します。

形式

```
public void setName(String name);
```

パラメタ

name：
　設定する名称を表す String を指定します。

例外

NullPointerException：
　パラメタ name の値が null です。

戻り値

なし

注意事項

複数の ExplicitMemory に同じ名前を設定できるため，名称に一意性はありません。

toString メソッド

説明

このオブジェクトの文字列表現を返却します。this.getName() を呼び出し，その結果を返却します。

形式

```
public String toString();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトの文字列表現を表す String 型オブジェクトへの参照を返却します。

totalMemory メソッド

説明

Explicit メモリブロックの確保済み総サイズを返却します。

形式

```
public long totalMemory();
```

パラメタ

なし

例外

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

共通エラーチェックをして，次のどちらかの値を返却します。

0：
　API の処理ができない場合に返却します。

このオブジェクトが表す Explicit メモリブロックの確保済みの総メモリサイズ（バイト数）：
　API の処理ができる場合は，このオブジェクトが表す Explicit メモリブロックが利用できるメモリサイズを long 型で返却します。

共通エラーチェックについては，「13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

usedMemory メソッド

説明

Explicit メモリブロックの使用されているメモリのサイズを返却します。

形式

```
public long usedMemory();
```

パラメタ

なし

例外

InaccessibleMemoryAreaException：

　　サポートされていない機能です。

戻り値

このオブジェクトが表す Explicit メモリブロックの使用されているメモリサイズ（バイト数）を long 型で返却します。

共通エラーチェックをして処理できないと判定された場合は，0 を返却します。共通エラーチェックについては，「13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

13.4　MemoryArea クラス

説明

Explicit メモリブロックまたは Java ヒープを表す抽象クラスです。MemoryArea クラスのパッケージは，JP.co.Hitachi.soft.jvm.MemoryArea です。

MemoryArea クラスに含まれるメソッドは，抽象メソッドのため処理がありません。各メソッドの処理については，派生クラスの同一シグネチャメソッドを参照してください。各メソッドの形式と参照先を次の表に示します。

メソッドの形式・同一シグネチャメソッドの一覧

メソッド名	形式	同一シグネチャメソッド
freeMemory メソッド	`public abstract long freeMemory();`	freeMemory メソッド
getName メソッド	`public abstract String getName();`	getName メソッド
newArray メソッド（形式 1）	`public abstract Object newArray(Class type, int number);`	newArray メソッド（形式 1）
newArray メソッド（形式 2）	`public abstract Object newArray(Class type, int[] dimensions);`	newArray メソッド（形式 2）
newInstance メソッド（形式 1）	`public abstract Object newInstance(Class type);`	newInstance メソッド（形式 1）
newInstance メソッド（形式 2）	`public abstract Object newInstance(Class type, Object... args);`	newInstance メソッド（形式 2）
newInstance メソッド（形式 3）	`public abstract Object newInstance(java.lang.reflect.Constructor cons, Object... args);`	newInstance メソッド（形式 3）
setName メソッド	`public abstract void setName(String name);`	setName メソッド
toString メソッド	`public abstract String toString();`	toString メソッド
totalMemory メソッド	`public abstract long totalMemory();`	totalMemory メソッド
usedMemory メソッド	`public abstract long usedMemory();`	usedMemory メソッド

13.5　MemoryInfo クラス

説明

Java プログラムから直接ガーベージコレクションのメモリ情報を取得できます。
例えば，現在使用中のサイズは次の式で求められます。

```
getXXXTotalMemory()－getXXXFreeMemory()
```

MemoryInfo クラスのパッケージは，JP.co.Hitachi.soft.jvm です。

メソッド一覧

メソッド名	機能
getEdenFreeMemory メソッド	Eden 領域の空きサイズを取得します。
getEdenMaxMemory メソッド	Eden 領域の最大使用サイズを取得します。
getEdenTotalMemory メソッド	Eden 領域の使用可能サイズを取得します。
getPermFreeMemory メソッド	Permanent 領域の空きサイズを取得します。
getPermMaxMemory メソッド	Permanent 領域の最大使用サイズを取得します。
getPermTotalMemory メソッド	Permanent 領域の使用可能サイズを取得します。
getSurvivorFreeMemory メソッド	Survivor 領域の空きサイズを取得します。
getSurvivorMaxMemory メソッド	Survivor 領域の最大使用サイズを取得します。
getSurvivorTotalMemory メソッド	Survivor 領域の使用可能サイズを取得します。
getTenuredFreeMemory メソッド	Tenured 領域の空きサイズを取得します。
getTenuredMaxMemory メソッド	Tenured 領域の最大使用サイズを取得します。
getTenuredTotalMemory メソッド	Tenured 領域の使用可能サイズを取得します。

使用例

メモリ情報を取得する際のメソッドの使用例を次に示します。

Perm 領域の空きサイズを求める場合

```
free_memory =
JP.co.Hitachi.soft.jvm.MemoryInfo.getPermFreeMemory()
```

現在使用中の Eden 領域を求める場合

```
use_memory =
JP.co.Hitachi.soft.jvm.MemoryInfo.getEdenTotalMemory()-JP.c
o.Hitachi.soft.jvm.MemoryInfo.getEdenFreeMemory()
```

getEdenFreeMemory メソッド

説明

Eden 領域の空きサイズを取得します。

形式

```
getEdenFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Eden 領域の空きサイズ（バイト数）を long 型で返却します。

getEdenMaxMemory メソッド

説明

Eden 領域の最大使用サイズを取得します。

形式

```
getEdenMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Eden 領域の最大使用サイズ（バイト数）を long 型で返却します。

getEdenTotalMemory メソッド

説明

Eden 領域の使用可能サイズを取得します。

形式

```
getEdenTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Eden 領域の使用可能サイズ（バイト数）を long 型で返却します。

getPermFreeMemory メソッド

説明

Permanent 領域の空きサイズを取得します。

形式

```
getPermFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Permanent 領域の空きサイズ（バイト数）を long 型で返却します。

getPermMaxMemory メソッド

説明

Permanent 領域の最大使用サイズを取得します。

形式

```
getPermMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Permanent 領域の最大使用サイズ（バイト数）を long 型で返却します。

getPermTotalMemory メソッド

説明

Permanent 領域の使用可能サイズを取得します。

形式

```
getPermTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Permanent 領域の使用可能サイズ（バイト数）を long 型で返却します。

getSurvivorFreeMemory メソッド

説明

Survivor 領域の空きサイズを取得します。

形式

```
getSurvivorFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Survivor 領域の空きサイズ（バイト数）を long 型で返却します。

getSurvivorMaxMemory メソッド

説明

Survivor 領域の最大使用サイズを取得します。

形式

```
getSurvivorMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Survivor 領域の最大使用サイズ（バイト数）を long 型で返却します。

getSurvivorTotalMemory メソッド

説明

Survivor 領域の使用可能サイズを取得します。

形式

```
getSurvivorTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Survivor 領域の使用可能サイズ（バイト数）を long 型で返却します。

getTenuredFreeMemory メソッド

説明

Tenured 領域の空きサイズを取得します。

形式

```
getTenuredFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Tenured 領域の空きサイズ（バイト数）を long 型で返却します。

getTenuredMaxMemory メソッド

説明

Tenured 領域の最大使用サイズを取得します。

形式

```
getTenuredMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Tenured 領域の最大使用サイズ（バイト数）を long 型で返却します。

getTenuredTotalMemory メソッド

説明

Tenured 領域の使用可能サイズを取得します。

形式

```
getTenuredTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Tenured 領域の使用可能サイズ（バイト数）を long 型で返却します。

13.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）

有効な Explicit メモリブロックを示していない場合，Explicit ヒープを操作する多くの API は処理できなくなります。そこで，各 API 共通のエラーチェックルーチンを定義して，API の処理ができるかどうか判断します。共通エラーチェックは，各 API の処理対象である Explicit メモリブロックの状態によって，API を処理できるかどうかを判断します。共通エラーチェックで返却される値は次のとおりです。

true：

API の処理を続けることができると判断します。処理対象である Explicit メモリブロックの状態が有効な場合に返却されます。

false：

API を処理できないと判断します。処理対象である Explicit メモリブロックの状態が無効な場合に返却されます。

InaccessibleMemoryAreaException（例外クラス）：

サポートしていない機能を実行しようとした場合にスローされます。

InaccessibleMemoryAreaException クラスの詳細は，「13.7　例外クラス」を参照してください。

なお，処理対象の Explicit メモリブロックが次の状態である場合にスローされます。

- 解放済みの場合
- 解放予約済み，または自動解放明示予約済みの場合
- 有効，無効，解放済み，および解放予約済み以外の状態の場合

13.7　例外クラス

JavaVM で使用する API で発生する例外を表す，例外クラスについて説明します。

例外クラスの一覧を次に示します。

表 13-2　JavaVM で使用する API で発生する例外クラス

例外クラス名	説明
JP.co.Hitachi.soft.jvm.MemoryArea.MemoryManagementException	JP.co.Hitachi.soft.jvm.MemoryArea パッケージで定義する例外クラスの基底クラスです。Java プログラムで，生成またはスローすることは想定しません。 派生クラスは，InaccessibleMemoryAreaException クラスです。
JP.co.Hitachi.soft.jvm.MemoryArea.InaccessibleMemoryAreaException	MemoryArea クラスのインスタンスに対して，サポートしていない機能を実行しようとした場合にスローします。 例えば，削除予約済みまたは削除済みの ExplicitMemory クラスのインスタンスに対する削除操作が該当します。 Java プログラムで，生成またはスローすることは想定しません。 基底クラスは，MemoryManagementException クラスです。

14 Cosminexus DABroker Library で使用する API

この章では，Cosminexus DABroker Library で使用する JDBC インタフェースおよび API について説明します。

14.1　Cosminexus DABroker Library で使用する API の一覧

Cosminexus DABroker Library で使用する API の一覧を，次の表に示します。

表 14-1　Cosminexus DABroker Library で使用する API の一覧

クラス名	主な機能
Driver クラス	• データベースの接続 • 指定された URL の妥当性チェック • DriverManager.getConnection メソッドで指定する接続プロパティの情報取得 • ドライバのバージョン情報の返却
Connection クラス	• Statement クラス，PreparedStatement クラス，および CallableStatement クラスのオブジェクト生成 • トランザクションのコミットまたはロールバック • オートコミットモードの設定
Statement クラス	• SQL の実行 • 検索結果として，ResultSet（ResultSet オブジェクト）の生成 • 更新結果として，更新行数の返却 • 最大検索行数の設定 • 検索制限時間の設定
PreparedStatement クラス	• ? パラメタ付き SQL の実行 • ? パラメタの設定 • 検索結果として，ResultSet オブジェクトの生成および返却 • 更新結果として，更新行数の返却
CallableStatement クラス	• ストアドプロシジャの実行 • INPUT および INOUT パラメタの設定（PreparedStatement クラスの setXXX メソッドを使用） • INOUT および OUTPUT パラメタの登録 • INOUT および OUTPUT パラメタ値の取得
ResultSet クラス	• 行単位の ResultSet 内の移動 • 結果データの返却 • 検索結果データが NULL 値かどうかの通知
ResultSetMetaData クラス	• ResultSet の各列に対する，データ型およびデータ長のメタ情報の返却
DatabaseMetaData クラス	• 接続データベースに関する各種情報の返却 • 表一覧，列一覧などの一覧系情報を，ResultSet に格納して返却
DataSource クラス	• データベースの接続情報の設定および取得
Blob インタフェース	• Blob データの取得

14.2　Driver クラス

説明

Driver クラスでは，主に次の機能を提供します。

- データベースの接続
- 指定された URL の妥当性チェック
- DriverManager.getConnection メソッドで指定する接続プロパティの情報取得
- ドライバのバージョン情報の返却

Driver クラスの提供する各メソッドの詳細と使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

Driver クラスを使用したデータベース接続の設定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 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。

制限事項

Cosminexus DABroker Library で使用する Driver クラスの制限事項を，次に示します。

表 14-2　Driver クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
acceptURL	URL 構文の必須項目が設定されている場合は true を返却します。必須項目が不足している場合は false を返却します。	○	○
getPropertyInfo	引数 info に与えられた以外に設定可能なプロパティ情報の配列を返却します。	○	○
jdbcCompliant	無条件に true を返却します。	○	○

（凡例）

○：該当します。

14.3　Connection クラス

説明

Connection クラスでは，主に次の機能を提供します。

- Statement クラス，PreparedStatement クラス，および CallableStatement クラスのオブジェクト生成
- トランザクションのコミットまたはロールバック
- オートコミットモードの設定

Connection クラスの提供する各メソッドの詳細，および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

注意事項

● Connection のクローズ

Connection の close メソッドを発行しないでアプリケーションを終了すると，Cosminexus DABroker Library 側にリソースが残る場合があります。このため，SQL 実行などでエラーの発生によって終了する場合は，catch ブロックまたは finally ブロック内で close メソッドを発行してください。

Connection の close メソッドの発行でエラーが発生した場合，SQLException をスローしません。また，プーリング使用時（ConnectionPoolDataSource を使用した接続），および XA 使用時（XADataSource を使用した接続）は Connection の close メソッド実行でデータベースとの物理的な切断はしません。

● カタログ

Cosminexus DABroker Library では，接続データベース種別にかかわらず，カタログをサポートしていません。このため，getCatalog メソッドは無条件に null を返却し，setCatalog メソッドは何もしません。

● アクセスモード

Cosminexus DABroker Library では，アクセスモードの変更をサポートしていないため，isReadOnly メソッドは無条件に false を返却し，setReadOnly メソッドは何もしません。

● トランザクション分離モード

Cosminexus DABroker Library では，トランザクション分離モードの変更をサポートしていません。このため，getTransactionIsolation メソッドでは，常に TRANSACTION_READ_UNCOMMITTED を返却し（データベースの仕様を表しているものではありません），setTransactionIsolation メソッドは TRANSACTION_READ_UNCOMMITTED 以外が指定された場合，SQLException をスローします。

● データベースアクセスリソース数の制限

Cosminexus DABroker Library を利用するアプリケーションでは，一つのコネクショ

ンで利用できるリソース数の最大は 1024 個です。ここでいうリソースとは，Statement クラス，PreparedStatement クラス，CallableStatement クラスのオブジェクト，および DatabaseMetaData クラスの中で一覧系情報を ResultSet に格納するメソッドの同時実行数を指します。

● オートコミット

次のすべての条件に一致する更新 SQL（INSERT，UPDATE および DELETE）を実行する場合は，事前に検索中の ResultSet オブジェクトまたは検索中の ResultSet オブジェクトを生成した Statement（継承したクラスも含みます）オブジェクトに対して close メソッドを実行してください。

- コネクションプーリングを使用してデータベースに接続している，または XA を用いてデータベースに接続している場合
- 接続データベースが HiRDB の場合
- ホールダブルカーソルを使用していない場合
- Statement クラスの setExecuteDirectMode メソッドを，引数 Mode に false を指定して実行している場合
- オートコミットを有効にしている場合

close メソッドを実行しないで更新 SQL を実行した場合，更新 SQL 実行後 ResultSet オブジェクトをクローズした時に HiRDB でエラーが発生することがあります。この場合，SQLException によるエラーの通知はされませんが，HiRDB のエラーログ，SQL トレースおよび Cosminexus DABroker Library の拡張データベースアクセストレースに HiRDB のエラーメッセージ（KFPA11501-E）を出力します。

制限事項

Cosminexus DABroker Library で使用する Connection クラスの制限事項を，次に示します。

表 14-3　Connection クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
close	通常接続時はデータベースとの接続を解除します。プーリング使用時（ConnectionPoolDataSource を使用した接続），および XA 使用時（XADataSource を使用した接続）は物理的な切断はしません。Connection.close メソッドの実行でエラーが発生した場合は SQLException をスローしません。	○	○

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
createStatement	更新結果を反映する ResultSet をサポートしていないため，ResultSet タイプに TYPE_SCROLL_SENSITIVE を指定した場合，TYPE_SCROLL_INSENSITIVE に切り替え，SQLWarning を設定します。また，並行処理タイプに CONCURE_UPDATABLE を指定した場合，CONCUR_READ_ONLY に切り替え，SQLWarning を設定します。	－	○
getCatalog	無条件に null を返却します。	○	○
getTransactionIsolation	無条件に TRANSACTION_READ_UNCOMMITTED を返却します。	○	○
getTypeMap	ユーザ定義型をサポートしていないため，無条件に SQLException をスローします。	－	○
isReadOnly	無条件に false を返却します。	○	○
prepareCall	更新結果を反映する ResultSet をサポートしていないため，ResultSet タイプに TYPE_SCROLL_SENSITIVE を指定した場合，TYPE_SCROLL_INSENSITIVE に切り替え，SQLWarning を設定します。また，並行処理タイプに CONCURE_UPDATABLE を指定した場合，CONCUR_READ_ONLY に切り替え，SQLWarning を設定します。	－	○
prepareStatement	更新結果を反映する ResultSet をサポートしていないため，ResultSet タイプに TYPE_SCROLL_SENSITIVE を指定した場合，TYPE_SCROLL_INSENSITIVE に切り替え，SQLWarning を設定します。また，並行処理タイプに CONCURE_UPDATABLE を指定した場合，CONCUR_READ_ONLY に切り替え，SQLWarning を設定します。	－	○
setCatalog	使用できません。	○	○
setReadOnly	使用できません。	○	○
setTransactionIsolation	使用できません。	○	○
setTypeMap	ユーザ定義型をサポートしていないため，無条件に SQLException をスローします。	－	○

（凡例）

○：該当します。

－：該当しません。

14.4　Statement クラス

説明

Statement クラスでは，主に次の機能を提供します。

- SQL の実行
- 検索結果として，ResultSet（ResultSet オブジェクト）の生成
- 更新結果として，更新行数の返却
- 最大検索行数の設定
- 検索制限時間の設定

Statement クラスの提供する各メソッドの詳細，および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

注意事項

● マルチスレッド

一つの Statement オブジェクトを複数のスレッドで使用する場合，「SQL の実行～ ResultSet の取得～ ResultSet のクローズ」という一連の処理を，スレッドごとにシリアライズする必要があります。シリアライズしないで並行して処理した場合の動作は保証しません。

各スレッドには，それぞれ別の Statement オブジェクトを割り当ててください。

● 複数の ResultSet

Cosminexus DABroker Library では，複数の ResultSet を返却する機能をサポートしていません。このため，getMoreResults メソッドは無条件に false を返却し，現在オープンしている ResultSet が存在する場合，その ResultSet をクローズします。

● カーソル名称

Cosminexus DABroker Library では，位置決めされた更新および削除をサポートしていないため，setCursorName メソッドは何もしません。

● 検索制限時間

Cosminexus DABroker Library では，検索の時間監視はできません。このため，setQueryTimeout メソッドで指定した値は無効となります。

● setFetchSize メソッドの使用について

setFetchSize メソッドが有効になるのは，通常の検索の場合だけです。

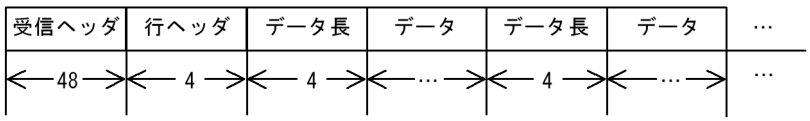
ResultSetMetaData や DatabaseMetaData の情報取得では有効になりません。

CLOB や BLOB などの LONGVARCHAR・LONGVARBINARY 型データを含むテーブルの検索では，setFetchSize メソッドの設定値は有効になりません。

setFetchSize メソッドを使用する場合，データ取得用のバッファサイズは setFetchSize メソッドの指定値，および検索するカラムの定義長などによって変化します。そのため，setFetchSize メソッドでの指定値が大き過ぎる場合や検索するカラムの定義長の合計が極端に大きくなる場合，メモリ不足が発生する可能性があるので

注意が必要です。
setFetchSize メソッドでの指定値が 1 以上の場合，必要となるバッファ領域は自動的に確保されます。この場合，どの程度の領域を必要とするかの計算方法を，次の表に示します。
バッファ領域に指定できる最大値は 2147483647 バイトです。最大値を超えないようにしてください。
ただし，Cosminexus DABroker Library の最小バッファサイズが 64 キロバイトのため，領域サイズが 64 キロバイトより小さい場合は，64 キロバイトに設定されます。
また，領域サイズが接続時にプロパティの BUF_SIZE，または DataSource の setBufSize メソッドで指定した値を超えた場合は，setFetchSize の指定値に必要な領域サイズが有効となります。

表 14-4　setFetchSize メソッド使用で必要となるバッファ領域

DBMS	データ型	必要となるバッファ領域（単位：バイト）
共通	 受信ヘッダ ＋（(行ヘッダ ＋ データ長 ＋ データ ＋ …）＊ 検索件数）	
HiRDB	int，integer，smallflt，real	4
	date	10
	smallint	2
	dec，decimal(m.n)	(m + 1) / 2
	float，double precision	8
	char(n)，mchar(n)，rowid(n)，varchar(n)，mvarchar(n)	n
	nchar(n)，nvarchar(n)	n * 2
	time	8
	timestamp(n)	7 + n / 2
ORACLE	varchar2(n)，char(n)，raw(n)，mlslabel(n)	n
	number	22
	float，rowid	8
	date	20

（例 1）
HiRDB 接続で FetchSize=3，検索カラムが varchar(10)，integer の場合
受信ヘッダ (48)+((行ヘッダ (4)+ データ長 (4)+varchar データ (10)+ データ長 (4)+integer デー

タ (4))*3)
となり，必要となるバッファ領域は 126 バイトとなります。この場合，合計が 64 キロバイト以下のため，確保される領域は 64 キロバイトになります。

（例 2）
HiRDB 接続で FetchSize=100，検索カラムが varchar(1000)，integer の場合
受信ヘッダ (48)+((行ヘッダ (4)+ データ長 (4)+varchar データ (1000)+integer データ (4))*100)
となり，101248 バイトのバッファ領域が確保されます。確保される領域が，プロパティの BUF_SIZE，または DataSource の setBufSize の指定値を超えた場合は，setFetchSize の指定値に必要な領域サイズが有効となります。

● Statement のクローズ
プーリング使用時（ConnectionPoolDataSource を使用した接続），および XA 使用時（XADataSource を使用した接続）で Statement の close メソッドの発行でエラーが発生した場合，SQLException をスローしません。また，プーリング使用時，および XA 使用時で Statement の close メソッド実行中にデータベースとの物理的な切断でエラーが発生してコネクションプーリングが使用できなくなった場合，ConnectionEventListener.connectionErrorOccurred は発生しません。

● 非同期キャンセル
接続データベースが HiRDB または ORACLE の場合，cancel メソッドを使用して非同期キャンセルを発行できます。ただし，XADataSource を使用した接続の場合，非同期キャンセルの実行は接続データベースの仕様に従います（HiRDB は XADataSource を使用した接続時の非同期キャンセルをサポートしていません。そのため非同期キャンセル要求は有効になりません）。
また，HiRDB クライアント経由 XDM/RD E2 接続時の非同期キャンセルはサポートしていません。

制限事項

Cosminexus DABroker Library で使用する Statement クラスの制限事項を，次に示します。

表 14-5 Statement クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
cancel	接続データベース種別が HiRDB または ORACLE の場合だけ有効です。	○	○

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
close	プーリング使用時（ConnectionPoolDataSource 使用した接続），および XA 使用時（XADataSource を使用した接続）で Statement.close メソッド実行中にエラーが発生した場合，SQLException をスローしません。また，プーリング使用時，および XA 使用時で Statement.close メソッド実行中にデータベースとの物理的な切断でエラーが発生してコネクションプーリングが使用できなくなった場合，ConnectionEventListener.connectionErrorOccurred は発生しません。	○	○
getFetchSize	0，または setFetchSize メソッドで指定された値が setMaxRows メソッドで指定された値より小さい場合は setFetchSize メソッドで指定された値を，setFetchSize メソッドで指定された値が setMaxRows メソッドで指定された値より大きい場合は setMaxRows メソッドで指定された値を返却します。	－	○
getMaxFieldSize	setMaxFieldSize が指定されていない場合 0 を返却します。	○	○
getMaxRows	setMaxRows が指定されていない場合は 0 を返却します。	○	○
getMoreResults	無条件に結果なし =false を返却します。	○	○
setCursorName	使用できません。	○	○
setFetchDirection	FETCH_FORWARD 以外を指定した場合，SQLException をスローします。	－	○
setFetchSize	接続データベース種別が HiRDB，または ORACLE の場合で，CLOB や BLOB などの LONG VARCHAR・LONG VARBINARY 型データ列の検索を含まないときだけ有効となります。	－	○
setMaxFieldSize	0 より小さい値を指定した場合，SQLException をスローします。	○	○
setMaxRows	0 より小さい値を指定した場合，SQLException をスローします。	○	○

（凡例）

○：該当します。

－：該当しません。

Cosminexus DABroker Library 提供メソッド

Statement クラスで提供している，Cosminexus DABroker Library だけの機能を次に示します。

メソッド一覧

メソッド名	機能
getExecuteDirectMode メソッド	接続データベースが HiRDB の場合，Statement クラスを使用したデータベースの更新で，HiRDB の Execute Direct 機能を使用するかどうかの設定情報を取得します。
setExecuteDirectMode メソッド	接続データベースが HiRDB の場合，Statement クラスを使用したデータベースの更新で，HiRDB の Execute Direct 機能を使用するかどうかを設定します。

getExecuteDirectMode メソッド

説明

接続データベースが HiRDB の場合，INSERT，UPDATE，DELETE など Statement クラスを使用したデータベースの更新で，HiRDB の Execute Direct 機能を使用するかどうかの設定情報を取得します。

形式

```
public boolean getExecuteDirectMode();
```

パラメタ

なし

例外

なし

戻り値

boolean：

Execute Direct 機能を使用するかどうかを次の値で取得します。

- true
  Execute Direct 機能を使用します。
- false
  Execute Direct 機能を使用しません。

setExecuteDirectMode メソッド

説明

接続データベースが HiRDB の場合，INSERT，UPDATE，DELETE など Statement クラスを使用したデータベースの更新で，HiRDB の Execute Direct 機能を使用するかどうかを設定します。

このメソッドが呼び出されない場合は，false を設定します。

なお，この指定は接続データベースが HiRDB の場合だけ有効です。HiRDB 以外のデータベースの場合は，指定の有無にかかわらず Execute Direct 機能を使用しません。

形式

```
public void setExecuteDirectMode(boolean Mode);
```

パラメタ

Mode：

HiRDB の Execute Direct 機能を使用するかどうかを次の値で設定します。

- true
  Execute Direct 機能を使用します。
- false
  Execute Direct 機能を使用しません。

例外

なし

戻り値

なし

14.5　PreparedStatement クラス

説明

PreparedStatement クラスでは，主に次の機能を提供します。

- ? パラメタ付き SQL の実行
- ? パラメタの設定
- 検索結果として，ResultSet オブジェクトの生成および返却
- 更新結果として，更新行数の返却

PreparedStatement クラスの提供する各メソッドの詳細および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。
なお，PreparedStatement クラスは Statement クラスのサブクラスです。Statement クラスの機能をすべて継承します。

注意事項

- PreparedStatement クラスは Statement クラスのサブクラスであるため，Statement クラスの注意事項はすべて該当します。Statement クラスの注意事項を参照してください。
- HiRDB を使用する場合での TIME 型，DATE 型，TIMESTAMP 型の列に対するデータ変換処理
  TIME 型，DATE 型，または TIMESTAMP 型の列に対して，setTime，setDate，setTimestamp および setString メソッドを使用してデータが設定された場合，HiRDB のデータ型に応じてデータが変換されます。列のデータ型と，メソッドとの組み合わせによる，変換処理について次の表に示します。なお，実際に存在しない日時を指定した場合は，JavaVM の返す値となります。

表 14-6　TIME 型，DATE 型，TIMESTAMP 型と setXXX メソッドとの変換

setXXX メソッド	HiRDB 列属性		
	TIME	DATE	TIMESTAMP
setTime(Time Obj) ※1	設定値をデータベースに格納します。	HiRDB のエラーを返します。	設定値 hh:mm:ss[.000000] の前に 1970-01-01 を付加したデータをデータベースに格納します。
setDate(Date Obj) ※2	HiRDB のエラーを返します。	設定値をデータベースに格納します。	設定値 yyyy-MM-DD の後ろに 00:00:00[.000000] を付加したデータをデータベースに格納します。
setTimestamp(Timestamp Obj) ※3	HiRDB のエラーを返します。	設定値から yyyy-MM-DD を抜き出したデータをデータベースに格納します。	設定値をデータベースに格納します。

setXXX メソッド	HiRDB 列属性		
	TIME	DATE	TIMESTAMP
setString(hh:mm:ss 形式の文字列)	設定値をデータベースに格納します。	HiRDB のエラーを返します。	HiRDB のエラーを返します。
setString(yyyy-MM-DD 形式の文字列)	HiRDB のエラーを返します。	設定値をデータベースに格納します。	HiRDB のエラーを返します。
setString(yyyy-MM-DD hh:mm:ss[.ffffff] 形式の文字列) ※4	HiRDB のエラーを返します。	HiRDB のエラーを返します。	設定値をデータベースに格納します。

注※ 1　Time Obj：java.sql.Time オブジェクト「時：分：秒」の値を持つオブジェクト。

注※ 2　Date Obj：java.sql.Date オブジェクト「年－月－日」の値を持つオブジェクト。

注※ 3　Timestamp Obj：java.sql.Timestamp オブジェクト「年－月－日　時：分：秒．ナノ秒」の値を持つオブジェクト。

注※ 4　[.ffffff]：HiRDB の Timestamp 型の精度によって，小数点以下のけた数が変わります。

● HiRDB を使用する場合での setXXX メソッド使用時のオーバーフローチェック

XXX は，?パラメタの適切な型です。HiRDB には，値によってはオーバーフローが発生するデータ型が存在します。HiRDB のデータ型と，メソッドとの組み合わせによるオーバーフローの発生の有無について，次の表に示します。オーバーフロー発生時は例外を返します。

表 14-7　HiRDB のデータ型と setXXX メソッドの組み合わせによるオーバーフローの発生の有無

setXXX メソッド	HiRDB のデータ型											
	SMALLINT	INTEGER	FLOAT	REAL	DECIMAL	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY	BLOB
setByte	○	○	○	○	×	－	－	－	－	－	－	－
setShort	○	○	○	○	×	－	－	－	－	－	－	－
setInt	×	○	○	○	×	－	－	－	－	－	－	－
setLong	×	×	○	○	×	－	－	－	－	－	－	－
setFloat	×	×	○	○	×	－	－	－	－	－	－	－
setDouble	×	×	○	×	×	－	－	－	－	－	－	－
setBigDecimal	×	×	○	○	×	－	－	－	－	－	－	－
setBoolean	○	○	○	○	×	－	－	－	－	－	－	－

setXXX メソッド	HiRDB のデータ型											
	SMALLINT	INTEGER	FLOAT	REAL	DECIMAL	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY	BLOB
setString	×	×	×	×	×	−	−	×	×	×	−	−
setBytes	−	−	−	−	−	−	−	−	−	−	−	−
setDate	−	−	−	−	−	−	−	○	−	×	−	−
setTime	−	−	−	−	−	−	−	−	○	×	−	−
setTimestamp	−	−	−	−	−	−	−	−	−	○	−	−

（凡例）

○：値にかかわらずオーバーフローしません。

×：値によってはオーバーフローする場合があります。

−：この組み合わせでは使用できません。

表 14-8　HiRDB のデータ型と setObject メソッドの組み合わせによるオーバーフローの発生の有無

setObject メソッド	HiRDB のデータ型											
	SMALLINT	INTEGER	FLOAT	REAL	DECIMAL	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY	BLOB
Byte	○	○	○	○	×	−	−	−	−	−	−	−
Short	○	○	○	○	×	−	−	−	−	−	−	−
Integer	×	○	○	○	×	−	−	−	−	−	−	−
Long	×	×	○	○	×	−	−	−	−	−	−	−
Decimal	×	×	○	○	×	−	−	−	−	−	−	−
Float	×	×	○	○	×	−	−	−	−	−	−	−
Double	×	×	○	×	×	−	−	−	−	−	−	−
Boolean	○	○	○	○	×	−	−	−	−	−	−	−
String	×	×	×	×	×	−	−	×	×	×	−	−
Date	−	−	−	−	−	−	−	○	−	−	−	−
Time	−	−	−	−	−	−	−	−	○	−	−	−
Timestamp	−	−	−	−	−	−	−	−	−	○	−	−
byte[]	−	−	−	−	−	−	−	−	−	−	−	−
BLOB	−	−	−	−	−	−	−	−	−	−	−	−

setObject メソッド	HiRDB のデータ型											
	SMALLINT	INTEGER	FLOAT	REAL	DECIMAL	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY	BLOB
CLOB	―	―	―	―	―	―	―	―	―	―	―	―
Array	―	―	―	―	―	―	―	―	―	―	―	―
Ref	―	―	―	―	―	―	―	―	―	―	―	―
Struct	―	―	―	―	―	―	―	―	―	―	―	―

（凡例）
○：値にかかわらずオーバーフローしません。
×：値によってはオーバーフローする場合があります。
―：この組み合わせでは使用できません。

● Commit または Rollback にわたっての SQL の前処理の保持機能

Cosminexus DABroker Library は DBMS が HiRDB または ORACLE の場合，SQL の前処理を Commit または Rollback にわたって保持します。このため，HiRDB の場合，SELECT，INSERT，DELETE，UPDATE，PURGE および CALL のどれかでアクセスする次のスキーマ資源に対して，ほかのユーザが定義系 SQL 文を発行すると，スキーマ資源にアクセスしていたコネクションを DISCONNECT するまでの間，定義系 SQL は排他待ちの状態になります。

- 表
- ビュー
- ストアドプロシジャ
- ストアドファンクション
- 抽象データ型

XADataSource を使用して，バージョンが 07-01 より前の HiRDB に接続した場合，この機能は有効になりません。バージョンが 07-01 以降の HiRDB に接続する場合に，この機能が有効になります。Cosminexus DABroker Library を使用したデータベース接続の実装については，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「11.6　Cosminexus DABroker Library を使用したデータベース接続の実装」を参照してください。

Cosminexus Component Container でコネクションをプーリングしている場合，定義系 SQL を発行する前に次のコマンドを実行してください。Cosminexus Component Container の cjclearpool コマンドを実行することで，コネクションプールからコネクションを破棄できます。

```
cjclearpool < サーバ名称 > -mode normal -resall
```

この場合，< サーバ名称 > で指定した J2EE サーバまたはバッチサーバに接続してい

るすべてのデータソースについてコネクションが削除されます。J2EE アプリケーションまたはバッチアプリケーションでコネクションをクローズしたときに，物理コネクションもクローズされます。
cjclearpool 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「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。

制限事項

Cosminexus DABroker Library で使用する PreparedStatement クラスの制限事項を，次に示します。

表 14-9　PreparedStatement クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
setArray	SQL 配列型をサポートしていないため，無条件に SQLException をスローします。	－	○
setBlob	接続データベースが Oracle9i，または Oracle10g 以外の場合，SQL BLOB 型をサポートしていないため，無条件に SQLException をスローします。	－	○
setClob	接続データベースが Oracle9i，または Oracle10g 以外の場合，SQL CLOB 型をサポートしていないため，無条件に SQLException をスローします。	－	○
setNull	ユーザ定義型をサポートしていないため，無条件に SQLException をスローします。	－	○
setRef	SQL 構造化型をサポートしていないため，無条件に SQLException をスローします。	－	○

（凡例）
○：該当します。
－：該当しません。

Cosminexus DABroker Library 提供メソッド

PreparedStatement クラスで提供している，Cosminexus DABroker Library だけの機能を次に示します。

メソッド一覧

メソッド名	機能
getBlockUpdate メソッド	接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかの設定情報を取得します。
setBlockUpdate メソッド	接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかを設定します。

getBlockUpdate メソッド

説明

接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかの設定情報を取得します。

形式

```
public boolean getBlockUpdate();
```

パラメタ

なし

例外

なし

戻り値

boolean：

複数のパラメタセットを一度に処理するかどうかを次の値で取得します。

- true
  複数のパラメタセットを一度に処理します。
- false
  複数のパラメタセットを一つずつ分割して処理します。

注意事項

- この機能は，HiRDB の配列を使用した更新機能を使用します。このため，HiRDB の配列を使用した更新機能の使用条件に満たない場合はエラーとなります。なお，配列を使用した更新機能の詳細については，マニュアル「HiRDB UAP 開発ガイド」の配列を使用した機能，およびマニュアル「HiRDB SQL リファレンス」の EXECUTE 文（SQL の実行形式 2），INSERT 文（行挿入）形式 3，DELETE 文（行削除）形式 2，UPDATE 文（行更新）形式 3，形式 4 を参照してください。
- この機能は，HiRDB に対する INSERT，UPDATE，DELETE 処理以外には使用できません。
- 複数のパラメタセット各列のデータ型はすべて同じにしてください。複数のパラメタセット各列のデータ型が異なる場合，更新処理実行時にエラーになります。
- DECIMAL 型データを挿入する場合，配列に指定する DECIMAL データの精度および位置取りはすべて同じにしてください。精度および位置取りが異なる場合はエラーになります。
- 複数のパラメタセットを一度に処理する場合に，バッチ更新の途中でエラーが発生すると，エラーが発生する直前までの更新処理はすべて無効になります。

setBlockUpdate メソッド

説明

接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかを設定します。

このメソッドが呼び出されない場合は，false を設定します。ただし，データベース接続時に DriverManager クラスの getConnection メソッドの引数に BLOCK_UPDATE=true を指定した場合，このメソッドが呼び出されないときに設定される値は，true になります。

形式

```
public void setBlockUpdate(boolean Mode);
```

パラメタ

Mode：

複数のパラメタセットを一度に処理するかどうかを次の値で設定します。

- true
  複数のパラメタセットを一度に処理します。
- false
  複数のパラメタセットを一つずつ分割して処理します。

例外

なし

戻り値

なし

注意事項

- この指定は，接続データベースが HiRDB の場合だけ有効です。HiRDB 以外のデータベースの場合は，指定の有無にかかわらず，複数のパラメタセットを一つずつ分割して処理します。
- 実際にパラメタセットが一度に処理されるかどうかは，Cosminexus DABroker Library および HiRDB の仕様に従います。
- この機能は，HiRDB の配列を使用した更新機能を使用します。このため，HiRDB の配列を使用した更新機能の使用条件に満たない場合はエラーとなります。なお，配列を使用した更新機能の詳細については，マニュアル「HiRDB UAP 開発ガイド」の配列を使用した機能，およびマニュアル「HiRDB SQL リファレンス」の EXECUTE 文（SQL の実行形式 2），INSERT 文（行挿入）形式 3，DELETE 文（行削除）形式 2，UPDATE 文（行更新）形式 3，形式 4 を参照してください。

- この機能は，HiRDB に対する INSERT，UPDATE，DELETE 処理以外には使用できません。
- 複数のパラメタセット各列のデータ型はすべて同じにしてください。複数のパラメタセット各列のデータ型が異なる場合，更新処理実行時にエラーになります。
- DECIMAL 型データを挿入する場合，配列に指定する DECIMAL データの精度および位置取りはすべて同じにしてください。精度および位置取りが異なる場合はエラーになります。
- 複数のパラメタセットを一度に処理する場合，バッチ更新の途中でエラーが発生すると，エラーが発生する直前までの更新処理はすべて無効になります。

14.6　CallableStatement クラス

説明

CallableStatement クラスでは，主に次の機能を提供します。なお，CallableStatement クラスは PreparedStatement クラスのサブクラスです。PreparedStatement クラスおよび Statement クラスの機能をすべて継承します。

- ストアドプロシジャの実行
- INPUT および INOUT パラメタの設定（PreparedStatement クラスの setXXX メソッドを使用）
- INOUT および OUTPUT パラメタの登録
- INOUT および OUTPUT パラメタ値の取得

CallableStatement クラスの提供する各メソッドの詳細および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

注意事項

● CallableStatement クラスは PreparedStatement クラスのサブクラスであるため，PreparedStatement クラス，および Statement クラスの注意事項はすべて該当します。PreparedStatement クラスおよび Statement クラスの注意事項を参照してください。

● ResultSet を伴うストアドプロシジャ

Cosminexus DABroker Library では，ResultSet を伴うストアドプロシジャをサポートしていません。このため，getMoreResults メソッドは無条件に false を返却し，getResultSet メソッドは無条件に null を返却します。

● ストアドプロシジャの OUT パラメタ取得

setXXX メソッド，および registerOutParameter メソッドで設定した情報は，execute メソッド，executeUpdate メソッド，または executeQuery メソッド実行後，getXXX メソッドを使用して必要な OUT パラメタをすべて取得し，それ以降その実行結果が不要になるまでは変更しないでください。OUT パラメタを取り出すときにパラメタの設定情報を参照するため，途中で clearParameter メソッドでパラメタの設定情報をクリアしたり，setXXX メソッド，および registerOutParameter メソッドでパラメタ情報を再設定したりすると，getXXX メソッドでエラーが発生し OUT パラメタの取得ができなくなります。

（例）OUT パラメタを取得する前に clearParameter メソッドを実行した場合

（実行する TEST_PROC(?) の ? パラメタは REAL 型の OUT パラメタとする）

```
cstmt = con.prepareCall("{CALL TEST_PROC(?)}");
cstmt.registerOutParameter(1, Types.REAL);  // OUTパラメタ情報の設定
cstmt.execute();   // プロシジャ実行
cstmt.clearParameter();   // パラメタ情報のクリア
float float_value = cstmt.getFloat(1);   // OUTパラメタの取得，エラーの発生
```

発生するエラー：KFDJ05006-E

```
Output attribute is not able to acuire information because it does
not exist in a parameter.
```

● DECIMAL 型使用時の注意事項

DECIMAL 型の OUT パラメタ，または INOUT パラメタを設定するとき，小数点以下のけた数（以下スケール値）を受け入れない形式の registerOutParameter(int parameterIndex，int sqlType) メソッドを使用するとスケール値は 0 とみなされます。

また，INOUT パラメタへの設定は最後に実行したメソッドのスケール値が有効となります。そのため DECIMAL 型の INOUT パラメタを設定する場合は，スケール値を受け入れない形式の registerOutParameter(int parameterIndex，int sqlType) メソッドを先に設定したあとに setBigDecimal(int parameterIndex，BigDecimal x) メソッドを設定，または setBigDecimal(int parameterIndex，BigDecimal x) メソッドを設定したあとにスケール値を受け入れる形式の registerOutParameter(int parameterIndex，int sqlType，int scale) メソッドを設定してください。このとき，setBigDecimal メソッドに指定する java.math.BigDecimal オブジェクトのスケール値，および registerOutParameter メソッドに指定するスケール値はデータベースに定義しているスケール値と同じにする必要があります。

（例 1）registerOutParameter メソッドを先に設定する場合 (定義長 (15,5) とする)

```
cstmt.registerOutParameter(1, Types.DECIMAL);
cstmt.setBigDecimal(1, new
java.math.BigDecimal("123.45000"));
```

（例 2）scale 付き registerOutParameter メソッドを使用する場合 (定義長 (15,5) とする)

```
cstmt.setBigDecimal(1, new
java.math.BigDecimal("123.45000"));
cstmt.registerOutParameter(1,Types.DECIMAL,5);
```

● データベースから 0 バイトデータを取得する場合の注意事項

データベースから 0 バイトデータを取得すると，null が返されることがあります。発生条件を次に示します。

HiRDB の場合

次の条件が重なった場合に発生します。

- データベースに格納されている 0 バイトデータを，ストアドプロシジャの OUT パラメタ，または INOUT パラメタで取得する場合。
- 列のデータ型が次のどれかの場合。
  VARCHAR
  NVARCHAR
  MVARCHAR
  BINARY

BLOB

Oracle の場合

次の条件が重なった場合に発生します。

- データベースに格納されている 0 バイトデータを，ResultSet クラス，ストアドプロシジャの OUT パラメタ，または INOUT パラメタで取得する場合。
- 列のデータ型が 0 バイトデータを格納できるデータ型の場合。

制限事項

Cosminexus DABroker Library で使用する CallableStatement クラスの制限事項を，次に示します。

表 14-10　CallableStatement クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
getArray	SQL 配列型をサポートしていないため，無条件に SQLException をスローします。	－	○
getBlob	接続データベースが Oracle9i，または Oracle10g 以外の場合，SQL BLOB 型をサポートしていないため，無条件に SQLException をスローします。	－	○
getClob	接続データベースが Oracle9i，または Oracle10g 以外の場合，SQL CLOB 型をサポートしていないため，無条件に SQLException をスローします。	－	○
getRef	SQL 構造化型をサポートしていないため，無条件に SQLException をスローします。	－	○

（凡例）

○：該当します。

－：該当しません。

14.7　ResultSet クラス

説明

ResultSet クラスでは，主に次の機能を提供します。

- 行単位の ResultSet 内の移動
- 結果データの返却
- 検索結果データが NULL 値かどうかの通知

ResultSet クラスの提供する各メソッドの詳細および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

注意事項

● マルチスレッド

一つの ResultSet オブジェクトを複数のスレッドで並行して使用する場合の動作は保証しません。

一つの ResultSet オブジェクトは一つのスレッドで処理してください。

● setFetchSize メソッドの使用について

Statement クラスの注意事項を参照してください。

● HiRDB のホールダブルカーソル機能の使用について

Cosminexus DABroker Library では，HiRDB のホールダブルカーソル機能を使用することによって，複数のコミットにわたってカーソルを保持できます。使用するためには，接続時にプロパティまたは URL に HIRDB_CURSOR=true を設定するか，DataSource クラスの setHiRDBCursorMode(true) を設定する必要があります。

● ResultSet のクローズ

プーリング使用時（ConnectionPoolDataSource を使用した接続），および XA 使用時（XADataSource を使用した接続）で ResultSet の close メソッド実行中にデータベースとの物理的な切断でエラーが発生しコネクションプーリングが使用できなくなった場合，ConnectionEventListener.connectionErrorOccurred は発生しません。

制限事項

Cosminexus DABroker Library で使用する ResultSet クラスの制限事項を，次に示します。

表 14-11　ResultSet クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
close	ステートメントのクローズ時，または次の SQL 実行時に暗黙的にクローズします。プーリング使用時（ConnectionPoolDataSource を使用した接続），および XA 使用時（XADataSource を使用した接続）で ResultSet.close メソッド実行中にデータベースとの物理的な切断でエラーが発生しコネクションプーリングが使用できなくなった場合 ConnectionEventListener.connectionErrorOccurred は発生しません。	○	○
cancelRowUpdates	更新可能型 ResultSet をサポートしていないため，無条件に SQLException をスローします。	－	○
deleteRow			
findColumn	指定された列名が ResultSet オブジェクトに含まれていないため列インデックスが取得できない場合，SQLException をスローします。	○	○
absolute	absolute(-2) と指定した場合，カーソルは最終行の前の行へ移動します。	－	○
afterLast	結果セットに行がない場合，何も処理しません。	－	○
beforeFirst			
cancelRowUpdates	更新可能型 ResultSet をサポートしていないため，無条件に SQLException をスローします。	－	○
deleteRow			
getArray	SQL 配列型をサポートしていないため，無条件に SQLException をスローします。	－	○
getBlob	接続データベースが HiRDB，Oracle9i，または Oracle10g 以外の場合，SQL LONGVARBINARY 型および SQL BLOB 型をサポートしていないため，無条件に SQLException をスローします。	－	○
getClob	接続データベースが Oracle9i，または Oracle10g 以外の場合，SQL CLOB 型をサポートしていないため，無条件に SQLException をスローします。	－	○
getFetchSize	0，または setFetchSize メソッドで指定された値が setMaxRows メソッドで指定された値より小さい場合は setFetchSize メソッドで指定された値を，setFetchSize メソッドで指定された値が setMaxRows メソッドで指定された値より大きい場合は setMaxRows メソッドで指定された値を返却します。	－	○
getRef	SQL 構造化型をサポートしていないため，無条件に SQLException をスローします。	－	○
isFirst	結果セットに行がない場合 false を返却します。	－	○
isLast	結果セットに行がない場合 false を返却します。	－	○

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
insertRow	更新可能型 ResultSet をサポートしていないため，無条件に SQLException をスローします。	－	○
moveToCurrentRow			
moveToInsertRow			
previous	カーソルの移動方向が next メソッドと逆方向になります。	－	○
refreshRow	更新可能型 ResultSet をサポートしていないため，無条件に SQLException をスローします。	－	○
relative	結果セットに行がない場合 SQLException をスローします。カーソルが先頭行の前，または最終行の後ろにある場合は有効として処理します。処理結果が結果セットの先頭行，最終行を超える場合はそれぞれ先頭行の前，最終行の後ろへ移動します。	－	○
rowDeleted	更新可能型 ResultSet をサポートしていないため，無条件に SQLException をスローします。	－	○
rowInserted			
rowUpdated			
setFetchDirection	FETCH_FORWARD 以外を指定した場合，SQLException をスローします。	－	○
setFetchSize	接続データベース種別が HiRDB，または ORACLE の場合で，CLOB や BLOB などの LONG VARCHAR・LONG VARBINARY 型データ列の検索を含まないときだけ有効となります。	－	○

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
updateAsciiStream	更新可能型 ResultSet をサポートしていないため，無条件に SQLException をスローします。	－	○
updateBigDecimal			
updateBinaryStream			
updateBoolean			
updateByte			
updateBytes			
updateCharacterStream			
updateDate			
updateDouble			
updateFloat			
updateInt			
updateLong			
updateNull			
updateObject			
updateRow			
updateShort			
updateString			
updateTime			
updateTimestamp			

（凡例）

○：該当します。

－：該当しません。

14.8　ResultSetMetaData クラス

説明

ResultSetMetaData クラスでは，主に次の機能を提供します。

- ResultSet の各列に対する，データ型およびデータ長のメタ情報の返却

ResultSetMetaData クラスの提供する各メソッドの詳細および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

制限事項

Cosminexus DABroker Library で使用する ResultSetMetaData クラスの制限事項を，次に示します。

表 14-12　ResultSetMetaData クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
getCatalogName	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。それ以外の場合，無条件に null を返却します。	○	○
getColumnClassName	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。それ以外の場合，無条件に false を返却します。	－	○
getColumnDisplaySize	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。	○	○
getColumnLabel	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。それ以外の場合，カラムのラベル（カラムヘッダ）をサポートしていないため，カラム名を返却します。	○	○
getColumnName	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。データベースから返される列名称を返却します。	○	○
getColumnType	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。	○	○
getPrecision	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。列属性が DECIMAL，NUMERIC の場合は精度を返却し，それ以外は列長を返却します。	○	○
getScale	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。列属性が DECIMAL，NUMERIC の場合は小数点以下のけた数を返却し，それ以外は 0 を返却します。	○	○
getSchemaName getTableName	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。それ以外の場合，無条件に null を返却します。	○	○

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
isAutoIncrement	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。それ以外の場合，無条件に false を返却します。	○	○
isCaseSensitive			
isCurrency			
isDefinitelyWritable			
isNullable	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。	○	○
isReadOnly	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。それ以外の場合，無条件に false を返却します。	○	○
isSearchable	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。	○	○
isSigned	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。列属性が数値属性の場合 true を返却し，それ以外の場合は false を返却します。	○	○
isWritable	column が 0 以下，またはカラム数を超えている場合は SQLException をスローします。それ以外の場合，無条件に false を返却します。	○	○

（凡例）

○：該当します。

－：該当しません。

14.9　DatabaseMetaData クラス

説明

DatabaseMetaData クラスでは，主に次の機能を提供します。

- 接続データベースに関する各種情報の返却
- 表一覧，列一覧などの一覧系情報を，ResultSet に格納して返却

DatabaseMetaData クラスの提供する各メソッドの詳細および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

制限事項

Cosminexus DABroker Library で使用する DatabaseMetaData クラスの制限事項を，次に示します。

表 14-13　DatabaseMetaData クラスの制限事項

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
dataDefinitionIgnoredInTransactions	無条件に false を返却します。	○	○
deletesAreDetected	更新結果を反映する ResultSet をサポートしていないため，無条件に false を返却します。	－	○
getBestRowIdentifier	行を一意に識別するテーブルの最適なカラムに関する記述を返却します（返す結果は常に 0 件）。	○	○
getCatalogs	カタログ名称に関する記述を返却します（返す結果は常に 0 件）。	○	○
getCrossReference	主キーテーブルの主キーカラムを参照する外部キーテーブル中の，外部キーカラムに関する記述を返却します（返す結果は常に 0 件）。	○	○
getDatabaseProductVersion	無条件に null を返却します。	○	○
getDefaultTransactionIsolation	無条件に TRANSACTION_READ_UNCOMMITTED を返却します。	○	○
getExportedKeys	主キーのカラムを参照する，外部キーのカラムに関する記述を返却します（返す結果は常に 0 件）。	○	○
getIdentifierQuoteString	無条件に引用符 (") を返却します。	○	○
getImportedKeys	外部キーのカラムを参照する，主キーのカラムに関する記述を返却します（返す結果は常に 0 件）。	○	○
getMaxConnections	無条件に 0 を返却します。	○	○

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
getMaxStatements	接続データベースが HiRDB または Oracle の場合，無条件に 1024 を，それ以外の場合は，64 を返却します。	○	○
getUDTs	ユーザ定義型に関する記述を返却します（返す結果は常に 0 件）。	－	○
getVersionColumns	自動的に更新されるカラムに関する記述を返却します（返す結果は常に 0 件）。	○	○
isReadOnly	アクセスモードを変更できないため，無条件に false を返却します。	○	○
insertsAreDetected	更新結果を反映する ResultSet をサポートしていないため，無条件に false を返却します。	－	○
othersDeletesAreVisible			
othersInsertsAreVisible			
othersUpdatesAreVisible			
ownDeletesAreVisible			
ownInsertsAreVisible			
ownUpdatesAreVisible			
storesLowerCaseQuotedIdentifiers	無条件に false を返却します。	○	○
supportsANSI92EntryLevelSQL	無条件に true を返却します。	○	○
supportsANSI92FullSQL	無条件に false を返却します。	○	○
supportsANSI92IntermediateSQL			
supportsCatalogsInIndexDefinitions			
supportsCatalogsInPrivilegeDefinitions			
supportsMixedCaseIdentifiers			
supportsMultipleResultSets			
supportsMultipleTransactions	無条件に true を返却します。	○	○
supportsPositionedDelete	無条件に false を返却します。	○	○
supportsPositionedUpdate			
supportsSelectForUpdate			
supportsSchemasInDataManipulation	無条件に true を返却します。	○	○
supportsTransactionIsolationLevel	与えられたトランザクションアイソレーションレベルが TRANSACTION_READ_UNCOMMITTED の場合，true を返却します。	○	○
supportsTransactions	無条件に true を返却します。	○	○

メソッド名	制限事項	JDBC1.0 での制限	JDBC2.0 での制限
usesLocalFilePerTable	無条件に false を返却します。	○	○
usesLocalFiles			
supportsOpenCursorsAcrossCommit	HiRDB 接続時 表 14-14 を参照してください。 Oracle 接続時 常に true を返却します。 XDM/RD E2 接続時 XDM/RD E2 11-01 以前の場合，常に false を返却します。 XDM/RD E2 11-02 以降の場合，HiRDB 接続時と同じです。詳細については，表 14-14 を参照してください。	－	○
supportsOpenCursorsAcrossRollback			
supportsOpenStatementsAcrossCommit			
supportsOpenStatementsAcrossRollback			
supportsBatchUpdates	無条件に true を返却します。	－	○
supportsResultSetConcurrency	ResultSet タイプが TYPE_FORWARD_ONLY または TYPE_SCROLL_INSENSITIVE で，並行処理タイプが CONCUR_READ_ONLY の場合，true を返却します。	－	○
updatesAreDetected	更新結果を反映する ResultSet をサポートしていないため，無条件に false を返却します。	－	○

（凡例）

○：該当します。

－：該当しません。

表 14-14　HiRDB 接続時の supportsOpenXXXX メソッドの戻り値

接続時の HiRDB_CURSOR の指定値	
FALSE	TRUE
supportsOpenCursorsAcrossCommit=false supportsOpenCursorsAcrossRollback=false supportsOpenStatementsAcrossCommit=true supportsOpenStatementsAcrossRollback=true	supportsOpenCursorsAcrossCommit=true supportsOpenCursorsAcrossRollback=false supportsOpenStatementsAcrossCommit=true supportsOpenStatementsAcrossRollback=true

14.10　DataSource クラス

説明

DataSource クラスでは，次の機能を提供します。

- データベースの接続情報の設定および取得

このクラスのパッケージ名は，JP.co.Hitachi.soft.DBPSV_Driver です。

JdbcDbpsvDataSource クラス

JDBC2.0 拡張機能で提供する，JNDI 連携機能を使用したデータベースへアクセスするためのインタフェースです。

表 14-15　JdbcDbpsvDataSource クラスのメソッド一覧

メソッド名	機能
getBlockUpdate メソッド	接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかの設定を取得します。
getBufferPoolSize メソッド	受信バッファプール数を取得します。
getBufSize メソッド	Cosminexus DABroker Library からの受信データのバッファ長を取得します。
getDatabaseName メソッド	接続するデータベースの種別を取得します。
getDBEnv メソッド	Cosminexus DABroker Library の接続先データベース定義情報を取得します。
getDBHostName メソッド	接続する HiRDB のホスト名を取得します。
getDescription メソッド	接続するデータベースに必要な接続付加情報を取得します。
getEncodLang メソッド	エンコード文字形態を取得します。
getExecuteDirectMode メソッド	接続データベースが HiRDB の場合，Statement クラスを使用したデータベースの更新で HiRDB の Execute Direct 機能を使用するかどうかの設定を取得します。
getHiRDBCursorMode メソッド	接続データベースが HiRDB の場合，検索時にカーソルが複数の Commit または Rollback にわたって有効かどうかの設定を取得します。
getJDBC_IF_TRC メソッド	JDBC インタフェースメソッドトレースの取得の有無を取得します。
getLONGVARBINARY_Access メソッド	HiRDB で LONGVARBINARY（列属性 BLOB，列属性 BINARY）へのアクセス方法を取得します。
getNetworkProtocol メソッド	Cosminexus DABroker Library との接続種別を取得します。
getNotErrorOccurred メソッド	connectionErrorOccurred が呼ばれるかどうかの設定を取得します。
getOSAuthorize メソッド	OS 認証機能を使用してデータベースに接続するかどうかの設定を取得します。

メソッド名	機能
getPassword メソッド	データベースアクセスのパスワードを取得します。
getPortNumber メソッド	接続する Cosminexus DABroker Library のポート番号を取得します。
getRowSize メソッド	JDBC で取り扱うバッファ長を取得します。
getServerName メソッド	接続する Cosminexus DABroker Library のホスト名を取得します。
getSQLWarningIgnore メソッド	データベースから返される警告を Connection クラスで保持するかどうかの情報を取得します。
getSV_EVENT_TRC メソッド	Cosminexus DABroker Library とのイベントトレースの取得の有無を取得します。
getTRC_NO メソッド	トレースのエントリ数を取得します。
getUapName メソッド	アプリケーション名称を取得します。
getUser メソッド	データベースアクセスのユーザ ID を取得します。
setBlockUpdate メソッド	接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかを設定します。
setBufferPoolSize メソッド	受信バッファプール数を設定します。
setBufSize メソッド	Cosminexus DABroker Library からの受信データのバッファ長を設定します。
setDatabaseName メソッド	接続するデータベースの種別を設定します。
setDBEnv メソッド	Cosminexus DABroker Library の接続先データベース定義情報を設定します。
setDBHostName メソッド	接続する HiRDB のホスト名を設定します。
setDescription メソッド	接続するデータベースに必要な接続付加情報を設定します。
setEncodLang メソッド	エンコード文字形態を設定します。
setExecuteDirectMode メソッド	接続データベースが HiRDB の場合，Statement クラスを使用したデータベースの更新で HiRDB の Execute Direct 機能を使用するかどうかを設定します。
setHiRDBCursorMode メソッド	接続データベースが HiRDB の場合，検索時にカーソルが複数の Commit または Rollback にわたって有効かどうかを設定します。
setJDBC_IF_TRC メソッド	JDBC インタフェースメソッドトレースの取得の有無を設定します。
setLONGVARBINARY_Access メソッド	HiRDB で LONGVARBINARY（列属性 BLOB，列属性 BINARY）へのアクセス方法を設定します。
setNetworkProtocol メソッド	Cosminexus DABroker Library との接続種別を設定します。
setNotErrorOccurred メソッド	connectionErrorOccurred が呼ばれるかどうかを設定します。
setOSAuthorize メソッド	OS 認証機能を使用してデータベースに接続するかどうかを設定します。

メソッド名	機能
setPassword メソッド	データベースアクセスのパスワードを設定します。
setPortNumber メソッド	接続する Cosminexus DABroker Library のポート番号を設定します。
setRowSize メソッド	JDBC で取り扱うバッファ長を指定します。
setServerName メソッド	接続する Cosminexus DABroker Library のホスト名を設定します。
setSQLWarningIgnore メソッド	データベースから返される警告を Connection クラスで保持するかどうかの情報を設定します。
setSV_EVENT_TRC メソッド	Cosminexus DABroker Library とのイベントトレースの取得の有無を設定します。
setTRC_NO メソッド	トレースのエントリ数を設定します。
setUapName メソッド	アプリケーション名称を設定します。
setUser メソッド	データベースアクセスのユーザ ID を設定します。

JdbcDbpsvXADataSource クラス

JDBC2.0 拡張機能で提供する，トランザクション連携のためのインタフェースです。

表 14-16　JdbcDbpsvXADataSource クラスのメソッド一覧

メソッド名	機能
getRMID メソッド	リソースマネジャの識別子を取得します。
getXACloseString メソッド	XA_CLOSE 文字列を取得します。
getXALocalCommitMode メソッド	XA 使用時，データベースのオートコミットを有効にしているかどうかの設定を取得します。
getXAOpenString メソッド	XA_OPEN 文字列を取得します。
getXAThreadMode メソッド	XA 使用時のスレッドモードを取得します。
setRMID メソッド	リソースマネジャの識別子を設定します。
setXACloseString メソッド	XA_CLOSE 文字列を設定します。
setXALocalCommitMode メソッド	XA 使用時，データベースのオートコミットを有効にするかどうかを設定します。
setXAOpenString メソッド	XA_OPEN 文字列を設定します。
setXAThreadMode メソッド	XA 使用時のスレッドモードを設定します。

getBlockUpdate メソッド

説明

接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかの設定情報を取得します。

形式

```
public boolean getBlockUpdate();
```

パラメタ

なし

例外

なし

戻り値

boolean：

複数のパラメタセットを一度に処理するかどうかを次の値で取得します。

- true
  複数のパラメタセットを一度に処理します。
- false
  複数のパラメタセットを一つずつ分割して処理します。

getBufferPoolSize メソッド

説明

setBufferPoolSize メソッドで設定した，受信バッファプール数を取得します。
setBufferPoolSize メソッドで受信バッファプール数が設定されていない場合は，0 を取得します。

このメソッドは，受信バッファプール数が設定されている場合に使用します。受信バッファプール数の設定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編」の「11.4　受信バッファのプーリングの設定」を参照してください。

形式

```
public int getBufferPoolSize();
```

パラメタ

なし

例外

なし

戻り値

int：
受信バッファプール数を取得します。

getBufSize メソッド

説明

setBufSize メソッドで設定した，Cosminexus DABroker Library からの受信データを格納するためのバッファ長を取得します。

setBufSize メソッドで受信バッファ長が設定されていない場合は，64 を取得します。

形式

```
public int getBufSize();
```

パラメタ

なし

例外

なし

戻り値

int：
バッファ長を取得します。

getDatabaseName メソッド

説明

setDatabaseName メソッドで設定された，接続するデータベースの種別を取得します。

接続データベース種別が設定されていない場合は，null を取得します。

形式

```
public String getDatabaseName();
```

パラメタ

なし

例外

なし

戻り値

String：
接続するデータベースの種別を取得します。

getDBEnv メソッド

説明

setDBEnv メソッドで設定された，接続するデータベースに対する接続先データベース定義情報を取得します。

接続先データベース定義情報が設定されていない場合は，null を取得します。

形式

```
public String getDBEnv();
```

パラメタ

なし

例外

なし

戻り値

String：
Cosminexus DABroker Library に設定してある，接続先データベース定義情報を取得します。

getDBHostName メソッド

説明

setDBHostName メソッドで設定された，接続する HiRDB のホスト名を取得します。

HiRDB のホスト名が設定されていない場合は，null を取得します。

形式

```
public String getDBHostName();
```

パラメタ

なし

例外

なし

戻り値

String：
HiRDB のホスト名を取得します。

getDescription メソッド

説明

setDescription メソッドで設定された，接続するデータベースに必要な接続付加情報を取得します。

接続付加情報が設定されていない場合は，null を取得します。

形式

```
public String getDescription();
```

パラメタ

なし

例外

なし

戻り値

String：
接続付加情報を取得します。

getEncodLang メソッド

説明

エンコード文字形態を取得します。

形式

```
public String getEncodLang();
```

パラメタ

なし

例外

なし

戻り値

String：

setEncodLang メソッドで設定した，文字エンコーディング情報を取得します。

getExecuteDirectMode メソッド

説明

接続データベースが HiRDB の場合，INSERT，UPDATE，DELETE など，Statement クラスを使用したデータベースの更新で，HiRDB の Execute Direct 機能を使用するかどうかの設定情報を取得します。

形式

```
public boolean getExecuteDirectMode();
```

パラメタ

なし

例外

なし

戻り値

boolean：

Execute Direct 機能を使用するかどうかを次の値で取得します。

- true
  Execute Direct 機能を使用します。
- false
  Execute Direct 機能を使用しません。

getHiRDBCursorMode メソッド

説明

HiRDB で検索時にカーソルが複数の Commit にわたって有効かどうかの設定情報を取得

します。

MetaData を取得する場合，true を指定して LOCK TABLE UNTIL DISCONNECT をしないで，複数の Commit にわたって Fetch をするとエラーが発生します。MetaData を取得する場合は，true を設定しないでください。

MetaData を取得する場合，false またはデフォルトの状態で ResultSet の Fetch 実行中に Commit を実行するとエラーが発生します。

この機能を使用する場合，マニュアル「HiRDB SQL リファレンス」の DECLARE CURSOR（カーソル宣言）形式 1 を参照してください。

形式

```
public boolean getHiRDBCursorMode();
```

パラメタ

なし

例外

なし

戻り値

boolean：

HiRDB で検索時にカーソルが複数の Commit にわたって有効かどうかの設定情報を次の値で取得します。

- true
  カーソルは保持されます。アプリケーションは続けて Fetch できます（LOCK TABLE UNTIL DISCONNECT が前提です）。
- false
  カーソルはクローズされますが，ステートメントは有効です。アプリケーションは，Prepare しないで，再度 Execute できます。

getJDBC_IF_TRC メソッド

説明

setJDBC_IF_TRC メソッドで設定した，JDBC インタフェースメソッドトレースの取得の有無を取得します。

形式

```
public boolean getJDBC_IF_TRC();
```

パラメタ

なし

例外

なし

戻り値

boolean：

トレースの取得の有無を次の値で取得します。

- true
  取得します。
- false
  取得しません。

getLONGVARBINARY_Access メソッド

説明

setLONGVARBINARY_Access メソッドで指定された，LONGVARBINARY（列属性 BLOB，列属性 BINARY）のデータベースアクセス方法の設定情報を返却します。

形式

```
public String getLONGVARBINARY_Access();
```

パラメタ

なし

例外

なし

戻り値

String：

LONGVARBINARY（列属性 BLOB，列属性 BINARY）のデータベースアクセス方法の設定情報を取得します。何も設定されていない場合，"REAL" を返却します。

- "REAL"
  実データでアクセスします。
- "LOCATOR"
  HiRDB の位置付け子（locator）機能を使用してアクセスします。

getNetworkProtocol メソッド

説明

setNetworkProtocol メソッドで設定された，Cosminexus DABroker Library との接続種別を取得します。

接続種別が設定されていない場合は，null を取得します。

形式

```
public String getNetworkProtocol();
```

パラメタ

なし

例外

なし

戻り値

String：

接続種別を取得します。

getNotErrorOccurred メソッド

説明

ConnectionEventListener.connectionErrorOccurred の発生を抑止するかどうかの設定情報を取得します。

形式

```
public boolean getNotErrorOccurred();
```

パラメタ

なし

例外

なし

戻り値

boolean：

ConnectionEventListener.connectionErrorOccurred を発生させるかどうかの設定情報を次の値で取得します。

- true
  connectionErrorOccurred を発生させません。
- false
  connectionErrorOccurred を発生させます。

getOSAuthorize メソッド

説明

OS 認証機能を使用してデータベースに接続するかどうかの設定情報を取得します。

このメソッドは，setOSAuthorize(boolean Mode) メソッドで設定した内容を取得します。

setOSAuthorize(boolean Mode) メソッドが呼び出されなかった場合は，false が設定されます。

形式

```
public boolean getOSAuthorize();
```

パラメタ

なし

例外

なし

戻り値

boolean：

OS 認証機能を使用するかどうかの設定情報を次の値で取得します。

- true
  OS 認証機能を使用します。
- false
  OS 認証機能を使用しません。

getPassword メソッド

説明

setPassword メソッドで設定した，データベースアクセスのパスワードを取得します。

setPassword メソッドでパスワードが設定されていない場合は，null を取得します。
setPassword メソッドでパスワードを設定し，ユーザ ID とパスワードを引数に持つ

getConnection メソッドを呼び出した場合は，getConnection メソッドで指定したパスワードを取得します。

形式

```
public String getPassword();
```

パラメタ

なし

例外

なし

戻り値

String：
パスワードを取得します。

getPortNumber メソッド

説明

setPortNumber メソッドで設定された，Cosminexus DABroker Library のポート番号を取得します。

ポート番号が設定されていない場合は，「40179」を取得します。

形式

```
public int getPortNumber();
```

パラメタ

なし

例外

なし

戻り値

int：
Cosminexus DABroker Library のポート番号を取得します。

getRowSize メソッド

説明

JDBC で取り扱うバッファ長を取得します。

形式

```
public int getRowSize();
```

パラメタ

なし

例外

なし

戻り値

int：

バッファ長を取得します。指定を省略した場合は，16 を取得します。

getServerName メソッド

説明

setServerName メソッドで設定された，Cosminexus DABroker Library のホスト名を取得します。

Cosminexus DABroker Library のホスト名が設定されていない場合は，null を取得します。

形式

```
public String getServerName();
```

パラメタ

なし

例外

なし

戻り値

String：

Cosminexus DABroker Library のホスト名または IP アドレスを取得します。

getSQLWarningIgnore メソッド

説明

データベースから返される警告を Connection クラスで保持するかどうかの設定情報を取得します。

形式

```
public boolean getSQLWarningIgnore();
```

パラメタ

なし

例外

なし

戻り値

boolean：

警告を保持しないかどうかの設定情報を次の値で取得します。

- true
  警告を保持しません。
- false
  警告を保持します。

getSV_EVENT_TRC メソッド

説明

setSV_EVENT_TRC メソッドで設定した，Cosminexus DABroker Library とのイベントトレースの取得の有無を取得します。

形式

```
public boolean getSV_EVENT_TRC();
```

パラメタ

なし

例外

なし

戻り値

boolean：

トレースの取得の有無を次の値で取得します。

- true
  取得します。
- false
  取得しません。

getTRC_NO メソッド

説明

setTRC_NO メソッドで設定した，トレースのエントリ数を取得します。

setTRC_NO メソッドでエントリ数が指定されていない場合は，100 を取得します。

形式

```
public int getTRC_NO();
```

パラメタ

なし

例外

なし

戻り値

int：
トレースのエントリ数を取得します。

getUapName メソッド

説明

setUapName メソッドで設定した，アプリケーション名称を取得します。

setUapName メソッドでアプリケーション名称が設定されていない場合は，JDBC ドライバの製品名称を取得します。

形式

```
public String getUapName();
```

パラメタ

なし

例外

なし

戻り値

String：
アプリケーション名称を取得します。

getUser メソッド

説明

setUser メソッドで設定した，データベースアクセスのユーザ ID を取得します。

setUser メソッドでユーザ ID が設定されていない場合は，null を取得します。

setUser メソッドでユーザ ID を設定し，ユーザ ID とパスワードを引数に持つ getConnection メソッドを呼び出した場合は，getConnection メソッドで指定したユーザ ID を取得します。

形式

```
public String getUser();
```

パラメタ

なし

例外

なし

戻り値

String：
ユーザ ID を取得します。

setBlockUpdate メソッド

説明

接続データベースが HiRDB の場合，? パラメタを使用したデータベースの更新で，複数のパラメタセットを一度に処理するかどうかを設定します。

このメソッドが呼び出されない場合は，false を設定します。

この指定は，接続データベースが HiRDB の場合だけ有効です。HiRDB 以外のデータ

ベースの場合は，指定の有無にかかわらず，複数のパラメタセットを一つずつ分割して処理します。Cosminexus DABroker Library を使用してデータベースに接続するための各種情報の設定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 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。

形式

```
public void setBlockUpdate(boolean Mode);
```

パラメタ

Mode：

複数のパラメタセットを一度に処理するかどうかを次の値で設定します。

- true
  複数のパラメタセットを一度に処理します。
- false
  複数のパラメタセットを一つずつ分割して処理します。

例外

なし

戻り値

なし

setBufferPoolSize メソッド

説明

受信バッファプール数を設定します。

0 を設定した場合，受信バッファプール数は設定されません。この場合，すべての受信バッファをプールします。0 未満の値を設定した場合，またはこのメソッドで受信バッファプール数を明示的に設定していない場合は，0 を仮定して，すべての受信バッファをプールします。

1025 以上の値を指定した場合は，1024 を仮定します。

値を仮定した場合は，Exception トレースログにメッセージ（KFDJ30010-W）を出力します。ただし，SQLWarning は作成されません。メッセージの詳細については，マニュアル「Cosminexus アプリケーションサーバ メッセージ 3」の「KFDJ30010-W」を参照してください。受信バッファプール数の設定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 保守／移行／互換編」の「11.4　受信バッファのプーリングの設定」を参照してください。

形式

```
public void setBufferPoolSize(int size);
```

パラメタ

size：

受信バッファプール数を設定するかどうかを指定します。また，受信バッファプール数を設定する場合，受信バッファプールの数を指定します。
次の値で指定します。

- 0
  受信バッファプール数を指定しません。この場合，すべての受信バッファがプールされます。これは，明示的に受信バッファプール数を指定しない場合と同じ動作です。
- 1 ～ 1024
  受信バッファプール数を指定する場合に，プールする受信バッファの数を 1 ～ 1024 の範囲で指定します。

例外

なし

戻り値

なし

setBufSize メソッド

説明

Cosminexus DABroker Library からの受信データを格納するためのバッファ長を設定します。

このメソッドが呼び出されない場合は，受信バッファ長として 64 を仮定します。

1 より小さい値を設定した場合は 64 を仮定します。

16000 より大きい値を指定した場合は 16000 を仮定します。

値を仮定した場合は，Exception トレースログにメッセージ（KFDJ30009-W）を出力します。メッセージは，DataSource.getConnection() メソッド，PooledConnection.getConnection() メソッド，または XAConnection.getConnection() メソッドによって，Connection オブジェクトを取得するときに出力します。メッセージの詳細については，マニュアル「Cosminexus アプリケーションサーバ メッセージ 3」の「KFDJ30009-W」を参照してください。

BLOB データなどの長大データを使用する場合，想定されるデータ以上の値を指定して

ください。

このメソッドの指定値の上限が 16000 キロバイトのため，BLOB データなど，16000 キロバイトを超える長大データにはアクセスできません。データベース種別が HiRDB の場合は，指定したバイト数を取り出す STRING のインタフェース（BLOB データの部分抽出機能）があるため，ここで指定したサイズ以下のサイズに分割してから取り出してください。BLOB データの部分抽出機能の詳細については，マニュアル「HiRDB UAP 開発ガイド」，およびマニュアル「HiRDB SQL リファレンス」を参照してください。

形式

```
public void setBufSize(int buf_size);
```

パラメタ

buf_size：

バッファ長を 1 ～ 16000 の範囲（単位：キロバイト）で指定します。

例外

なし

戻り値

なし

setDatabaseName メソッド

説明

接続するデータベースの種別を設定します。

このメソッドは，データベースとの接続時に必ず呼び出します。

形式

```
public void setDatabaseName(String database_name);
```

パラメタ

database_name：

接続するデータベースの種別を指定します。

接続データベース種別	設定する内容※1
HiRDB	"HIRDB"
ORACLE	"ORACLE"
Oracle8i※2	"ORACLE8I"

注※ 1
設定する内容には，大文字，小文字および大文字・小文字の混在する内容を指定できます。

注※ 2
接続データベース種別に Oracle8i を指定して，Cosminexus DABroker Library 環境設定の「使用する ORACLE のバージョン」に ORACLE9i を指定した場合，Oracle9i に，ORACLE10g を指定した場合，Oracle10g に接続できます。

例外

引数の内容が null の場合，または接続データベース種別に上記の内容以外が指定された場合は，SQLException をスローします。

戻り値

なし

setDBEnv メソッド

説明

Cosminexus DABroker Library の接続先データベース定義情報を設定します。

接続するデータベースに対応する，Cosminexus DABroker Library の接続先データベース定義情報を次の形式で指定します。

データベース種別名：データベース名

このメソッドは，接続するデータベースが Cosminexus DABroker Library-Database Connection Server 経由の場合，必ず呼び出します。接続するデータベースが Cosminexus DABroker Library-Database Connection Server 経由以外の場合は，設定は無視されます。

形式

```
public void setDBEnv(String db_env);
```

パラメタ

db_env：
接続先データベース定義情報を指定します。

例外

なし

戻り値

なし

setDBHostName メソッド

説明

接続する HiRDB のホスト名を設定します。

このメソッドで指定した値は，接続データベース種別が HiRDB のときだけ有効です。ただし，接続付加情報に HiRDB クライアントの環境変数グループ名を指定している場合は，値をこのメソッドで指定しても有効となりません。

形式

```
public void setDBHostName(String db_host_name);
```

パラメタ

db_host_name：
HiRDB のホスト名を指定します。

例外

なし

戻り値

なし

setDescription メソッド

説明

接続するデータベースに必要な接続付加情報を設定します。

接続データベース種別	設定する内容	設定の要否
HiRDB	HiRDB システムのポート番号，または HiRDB クライアントの環境変数グループ名※1，※3	任意
ORACLE	SQL*Net の接続文字列※1，※2	任意
Oracle8i		

注※1
省略時には，Cosminexus DABroker Library の動作環境定義の指定値が有効となります。

注※2
XA 使用時には，XA_OPEN 文字列で指定したデータベース名を指定します。

注※ 3

HiRDB クライアントの環境変数グループ名を指定する場合は，グループ名の先頭に @ を付加します。

（例）

- Windows の場合
  HiRDB クライアントの環境変数グループ名が HiRDB_ENV_GROUP のときは，次のように指定します。

```
@DABENVGRP=HiRDB_ENV_GROUP
```

- UNIX の場合
  HiRDB クライアントの環境変数グループ名のパスが /HiRDB_P/Client/HiRDB.ini のときは，次のように指定します。

```
@DABENVGRP=/HiRDB_P/Client/HiRDB.ini
```

@DABENVGRP= ＜環境変数グループ名＞を指定する場合は，指定内容に半角の空白を含めないでください。指定内容が次に示す例のどれかに該当する場合，DBID は正しく設定されません。

（例）

@DABENVGRP= ＜環境変数グループ名＞の指定内容に半角スペースを含みます。環境変数グループ名は HiRDB_ENV_GROUP（Windows の場合），または /HiRDB_P/Client/HiRDB.ini（UNIX の場合）とします。
半角の空白を△で示します。

- Windows の場合

```
@△DABENVGRP=HiRDB_ENV_GROUP
@DABENVGRP△=HiRDB_ENV_GROUP
@DABENVGRP=△HiRDB_ENV_GROUP
@DABENVGRP=HiRDB_ENV_GROUP△
```

- UNIX の場合

```
@△DABENVGRP=/HiRDB_P/Client/HiRDB.ini
@DABENVGRP△=/HiRDB_P/Client/HiRDB.ini
@DABENVGRP=△/HiRDB_P/Client/HiRDB.ini
@DABENVGRP=/HiRDB_P/Client/HiRDB.ini△
```

形式

```
public void setDescription(String description);
```

パラメタ

description：

接続付加情報を指定します。

例外

なし

戻り値

なし

setEncodLang メソッド

説明

エンコード文字形態を設定します。

String オブジェクトの取得時に unicode から変換する場合，JavaVM の標準エンコードを利用しないで，指定したエンコードを利用します。

なお，UNIX の場合，setEncodLang に指定する言語モードは，Cosminexus DABroker Library の言語モードと合わせる必要があります。言語モードは，Cosminexus DABroker Library 動作環境定義ファイルの共通設定項目の DAB_LANG（LANG 環境変数）で指定します。Cosminexus DABroker Library 動作環境定義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 (サーバ定義)」の「17.2　Cosminexus DABroker Library 動作環境定義ファイル」を参照してください。

形式

```
public void setEncodLang(String encode_lang);
```

パラメタ

encode_lang：
　データ変換で使用する文字エンコーディング情報を指定します。

例外

なし

戻り値

なし

setExecuteDirectMode メソッド

説明

接続データベースが HiRDB の場合，INSERT，UPDATE，DELETE など，Statement クラスを使用したデータベースの更新で，HiRDB の Execute Direct 機能を使用するか

どうかを設定します。

このメソッドが呼び出されない場合は，false を設定します。

この指定は接続データベースが HiRDB の場合だけ有効です。HiRDB 以外のデータベースの場合は，指定の有無にかかわらず，Execute Direct 機能を使用しません。

形式

```
public void setExecuteDirectMode(boolean Mode);
```

パラメタ

Mode：

HiRDB の Execute Direct 機能を使用するかどうかを次の値で設定します。

- true
  Execute Direct 機能を使用します。
- false
  Execute Direct 機能を使用しません。

例外

なし

戻り値

なし

setHiRDBCursorMode メソッド

説明

HiRDB で検索時にカーソルが複数の Commit にわたって有効かどうかを設定します。このメソッドが呼び出されない場合は，false が設定されます。

なお，この機能を使用する場合，マニュアル「HiRDB SQL リファレンス」の DECLARE CURSOR（カーソル宣言）形式 1 を参照してください。

形式

```
public void setHiRDBCursorMode(boolean Mode);
```

パラメタ

Mode：

HiRDB で検索時にカーソルが複数の Commit にわたって有効かどうかを次の値で設定します。

- true
  カーソルは保持されます。アプリケーションは続けて Fetch できます（LOCK

TABLE UNTIL DISCONNECT が前提です)。

- false
  カーソルはクローズされますが，ステートメントは有効です。アプリケーションは，Prepare しないで，再度 Execute できます。

例外

なし

戻り値

なし

注意事項

● 接続データベースが HiRDB 以外の場合，指定値は無視されます。

● ORACLE では，常にカーソルは複数の Commit，または Rollback にわたって有効です。

● SELECT，INSERT，DELETE，UPDATE，PURGE および CALL のどれかでアクセスする次のスキーマ資源に対して，ほかのユーザが定義系 SQL 文を発行すると，スキーマ資源にアクセスしていたコネクションを DISCONNECT するまでの間，定義系 SQL は排他待ちの状態になります。
- 表
- ビュー
- ストアドプロシジャ
- ストアドファンクション
- 抽象データ型

● XADataSource を使用して，バージョンが 07-01 より前の HiRDB に接続した場合，true にしても有効になりません。

true を設定する場合の注意事項

● MetaData を取得する場合，true を指定して LOCK TABLE UNTIL DISCONNECT をしないで，複数の Commit にわたって Fetch をするとエラーが発生します。MetaData を取得する場合は，true を設定しないでください。

● 接続データベースが HiRDB の場合でかつ，検索 SQL（SELECT 文）に「UNTIL DISCONNECT」を記述した場合は，必ず接続時のプロパティまたは URL に HIRDB_CURSOR=true を設定するか，このメソッドで true を設定してください。これらの設定をしなかった場合，Commit をわたった検索中の ResultSet クラスオブジェクトに対して close メソッドを実行してもカーソルをクローズしません。

● 次の表にはアクセスできません。アクセスした場合，HiRDB のエラーになります。
- 分散表
- ディクショナリ表

- 抽象データ型を含む表
- 外部表，または外部表を基表とするビュー表
- 関数呼び出しを指定して導出した名前付きの導出表（ビュー表，WITH 句の問い合わせ）

● カーソルをオープンする前に，そのカーソルを使用する表に対して UNTIL DISCONNECT 指定の LOCK TABLE 文を実行しておく必要があります。

● ROLLBACK を発行すると，使用しているカーソルはすべて閉じられます。

● 排他オプションに WITHOUT LOCK WAIT，または WITHOUT LOCK NOWAIT を指定できません。また，クライアント環境変数でデータ保証レベル（PDISLLVL）に 0，または 1 を指定した場合でも 2（WITH SHARE LOCK 相当）を仮定します。

● カーソルがオープンしている場合，同じコネクションで定義系 SQL は実行できません。また，カーソルがクローズしている場合，同じコネクションで定義系 SQL を実行すると SQL の前処理は無効になります。

● SELECT 文を実行し，その SELECT 文中で使用している表に対して PURGE TABLE 文を実行すると，カーソルはクローズされます。

● オープンしているカーソルで指定した表を，同じコネクションの別のステートメントでオープンすることはできません。

● 次のオペランドを指定する場合は，LOCK 文に IN EXCLUSIVE MODE を指定してください。
- WITH EXCLUSIVE LOCK
- FOR UPDATE 句

false を設定する場合の注意事項

● 通常，Fetch 時，Cosminexus DABroker Library に Fetch データをバッファリングするため，Commit 直後の Fetch でエラーが発生するとは限りません。

● MetaData を取得する場合，false またはデフォルトの状態で ResultSet の Fetch 実行中に，Commit を実行するとエラーが発生します。

● SELECT，INSERT，DELETE，UPDATE，PURGE および CALL のどれかでアクセスする次のスキーマ資源に対して，ほかのユーザが定義系 SQL 文を発行すると，スキーマ資源にアクセスしていたコネクションを DISCONNECT するまでの間，定義系 SQL は排他待ちの状態になります。
- 表
- ビュー
- ストアドプロシジャ
- ストアドファンクション

setJDBC_IF_TRC メソッド

説明

JDBC インタフェースメソッドトレースの取得の有無を設定します。

このメソッドが呼び出されない場合は，false を設定します。

なお，別途 setLogWriter メソッドで有効なログストリームを指定する必要があります。

形式

```
public void setJDBC_IF_TRC(boolean flag);
```

パラメタ

flag：

トレースの取得の有無を次の値で指定します。

- true
  取得します。
- false
  取得しません。

例外

なし

戻り値

なし

setLONGVARBINARY_Access メソッド

説明

LONGVARBINARY（列属性 BLOB，列属性 BINARY）のデータベースアクセス方法を設定します。

なお，次の場合，"LOCATOR" を指定しても "REAL" が指定されたものとみなされます。

- 接続先データベースが HiRDB 以外の場合。
- 接続先データベースである HiRDB の HiRDB サーバまたは HiRDB クライアントライブラリのバージョンが 07-00 以前の場合。

形式

```
public void setLONGVARBINARY_Access(String Mode);
```

パラメタ

String Mode：

LONGVARBINARY（列属性 BLOB，列属性 BINARY）のデータベースアクセス方法を，次の値で設定します。デフォルトは，"REAL" です。

- "REAL"
  実データでアクセスします。
- "LOCATOR"
  HiRDB の位置づけ子（locator）機能を使用してアクセスします。

例外

なし

戻り値

なし

setNetworkProtocol メソッド

説明

Cosminexus DABroker Library との接続種別を設定します。なお，このメソッドは，データベースとの接続時に必ず呼び出します。

形式

```
public void setNetworkProtocol(String network_protocol);
```

パラメタ

network_protocol：

接続種別を指定します。
指定できる接続種別を，次に示します。

接続種別	設定する内容※
ネイティブライブラリ接続（ローカルアクセス）	"lib"

注※

設定する内容には，大文字，小文字および大文字・小文字の混在する内容を指定できます。設定できる種別はネイティブライブラリ接続だけです。

例外

引数の内容が null の場合，または接続種別に指定できる内容以外が指定された場合は，SQLException をスローします。

戻り値

なし

setNotErrorOccurred メソッド

説明

ConnectionEventListener.connectionErrorOccurred の発生を抑止するかどうかを設定します。

ConnectionPooledDataSource，XADataSource 使用時に致命的な接続エラーが発生した場合，ConnectionEventListener.connectionErrorOccurred が呼ばれますが，このメソッドで，その呼び出しを抑止できます。

このメソッドが呼び出されない場合は，false が設定されます。

形式

```
public void setNotErrorOccurred(boolean Mode);
```

パラメタ

Mode：

connectionErrorOccurred の発生を抑止するかどうかを次の値で設定します。

- true
  connectionErrorOccurred を発生させません。
- false
  connectionErrorOccurred を発生させます。

例外

なし

戻り値

なし

注意事項

● 通常は，このメソッドを呼び出さないか，false を設定してください。

● ORACLE の FailOver 機能，または HiRDB の再接続機能を使用する場合は，true を設定してください。

setOSAuthorize メソッド

説明

OS 認証機能を使用してデータベースに接続するかどうかを設定します。

このメソッドが呼び出されない場合は，false が設定されます。

なお，Oracle に対する OS 認証機能を使用する場合は，アプリケーションサーバ，または Web コンテナサーバを OS 認証で使用するユーザで起動する必要があります。

このメソッドは，Oracle9i 9.2.0 または Oracle10g 10.1.0 に接続する場合だけ有効です。それ以外のバージョンの Oracle に接続した場合は動作を保証しません。接続するデータベースが ORACLE 以外の場合は，指定値に関係なく，OS 認証機能を使用しません。

形式

```
public void setOSAuthorize(boolean Mode);
```

パラメタ

Mode：

OS 認証機能を使用するかどうかを次の値で指定します。

- true
  OS 認証機能を使用します。
- false
  OS 認証機能を使用しません。

例外

なし

戻り値

なし

setPassword メソッド

説明

データベースアクセスのパスワードを設定します。

パスワードは，getConnection メソッドの引数でも指定できます。

このメソッドでパスワードを設定し，ユーザ ID とパスワードを引数に持つ getConnection メソッドを呼び出した場合は，getConnection メソッドで指定したパスワードが優先されます。

形式

```
public void setPassword(String password);
```

パラメタ

password：
　パスワードを指定します。

例外

なし

戻り値

なし

setPortNumber メソッド

説明

接続する Cosminexus DABroker Library のポート番号を設定します。

このメソッドが呼び出されない場合は，ポート番号として「40179」を使用します。

形式

```
public void setPortNumber(int port_no);
```

パラメタ

port_no：
　Cosminexus DABroker Library のポート番号を指定します。

例外

引数の内容が 0 より小さい場合，SQLException をスローします。

戻り値

なし

setRowSize メソッド

説明

JDBC で取り扱うバッファ長を，メガバイト単位の数字文字列で指定します。

指定を省略した場合，数字以外を指定した場合，または 16 より小さい値を指定した場合は 16 を仮定します。

512 より大きい値を指定した場合は 512 を仮定します。

形式

```
public void setRowSize(int row_size);
```

パラメタ

row_size：

バッファ長を 16 ～ 512 の範囲のメガバイト単位で指定します。

例外

なし

戻り値

なし

setServerName メソッド

説明

接続する Cosminexus DABroker Library のホスト名，または IP アドレスを設定します。

形式

```
public void setServerName(String server_name);
```

パラメタ

server_name：

Cosminexus DABroker Library のホスト名または IP アドレスを指定します。

例外

引数の内容が null の場合，SQLException をスローします。

戻り値

なし

setSQLWarningIgnore メソッド

説明

データベースから返される警告を Connection クラスで保持するかどうかの情報を設定します。このメソッドが呼び出されない場合は，false が設定されます。

形式

```
public void setSQLWarningIgnore(boolean Mode);
```

パラメタ

Mode：

警告を保持するかどうかを次の値で設定します。

- true
  警告を保持しません。
- false
  警告を保持します。

例外

なし

戻り値

なし

setSV_EVENT_TRC メソッド

説明

Cosminexus DABroker Library とのイベントトレースの取得の有無を設定します。

このメソッドが呼び出されない場合は，false を設定します。

なお，別途 setLogWriter メソッドで有効なログストリームを指定する必要があります。

形式

```
public void setSV_EVENT_TRC(boolean flag);
```

パラメタ

flag：

トレースの取得の有無を次の値で指定します。

- true
  取得します。
- false
  取得しません。

例外

なし

戻り値

なし

setTRC_NO メソッド

説明

トレースのエントリ数を設定します。

このメソッドが呼び出されない場合は，トレースのエントリ数は 500 となります。

10 ～ 1000 以外の値が設定された場合は，トレースを取得しません。

形式

```
public void setTRC_NO(int trc_no);
```

パラメタ

trc_no：

トレースのエントリ数を 10 ～ 1000 の範囲で指定します。

例外

なし

戻り値

なし

setUapName メソッド

説明

アプリケーション名称を設定します。

このメソッドが呼び出されない場合は，アプリケーション名称として，JDBC ドライバの製品名称を Cosminexus DABroker Library に通知します。

設定されたアプリケーション名称は，Cosminexus DABroker Library のデータベースアクセストレースの PAPNAME，および拡張データベースアクセストレースの Client Name に出力されます。

形式

```
public void setUapName(String uap_name);
```

パラメタ

uap_name：
　アプリケーション名称を指定します。

例外

なし

戻り値

なし

setUser メソッド

説明

データベースアクセスのユーザ ID を設定します。

ユーザ ID は，getConnection メソッドの引数でも指定できます。

このメソッドでユーザ ID を設定し，ユーザ ID とパスワードを引数に持つ getConnection メソッドを呼び出した場合は，getConnection メソッドで指定したユーザ ID が優先されます。

形式

```
public void setUser(String user);
```

パラメタ

user：
　ユーザ ID を指定します。

例外

なし

戻り値

なし

getRMID メソッド

説明

setRMID メソッドで設定した，リソースマネジャの識別子を取得します。

setRMID メソッドで識別子が設定されていない場合は，1 を取得します。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public int getRMID();
```

パラメタ

なし

例外

なし

戻り値

int：
　リソースマネジャの識別子を取得します。

getXACloseString メソッド

説明

setXACloseString メソッドで設定した，XA_CLOSE 文字列を取得します。

setXACloseString メソッドで XA_CLOSE 文字列が設定されていない場合は，null を取得します。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public String getXACloseString();
```

パラメタ

なし

例外

なし

戻り値

String：
　XA_CLOSE 文字列を取得します。

getXALocalCommitMode メソッド

説明

XA 使用時，トランザクションが分散トランザクションでないとき，データベースのオートコミットモードを有効にしているかどうかの設定情報を取得します。

形式

```
public boolean getXALocalCommitMode();
```

パラメタ

なし

例外

なし

戻り値

boolean：

XA 使用時のオートコミットの設定情報を次の値で取得します。

- true
  データベースのオートコミットが有効です。
- false
  データベースのオートコミットは無効です。

getXAOpenString メソッド

説明

setXAOpenString メソッドで設定した，XA_OPEN 文字列を取得します。

setXAOpenString メソッドで XA_OPEN 文字列が設定されていない場合は，null を取得します。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public String getXAOpenString();
```

パラメタ

なし

例外

なし

戻り値

String：

XA_OPEN 文字列を取得します。

getXAThreadMode メソッド

説明

setXAThreadMode メソッドで設定した，XA 使用時のスレッドモードを取得します。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public boolean getXAThreadMode();
```

パラメタ

なし

例外

なし

戻り値

boolean：

XA 使用時のスレッドモードを次の値で取得します。

- true
  マルチスレッドモード
- false
  シングルスレッドモード

setRMID メソッド

説明

リソースマネジャの識別子を設定します。

このメソッドが呼び出されない場合は，識別子は 1 となります。

複数のリソースマネジャを同時に使用する場合，リソースマネジャごとに一意な識別子を設定してください。

リソースマネジャの識別子については，各 DBMS のマニュアルを参照してください。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public void setRMID(int rmid);
```

パラメタ

rmid：

リソースマネジャの識別子を，1 以上の正の数値で指定します。

例外

引数の内容が 1 より小さい場合，SQLException をスローします。

戻り値

なし

setXACloseString メソッド

説明

XA_CLOSE 文字列を設定します。なお，XA_CLOSE 文字列については，各 DBMS のマニュアルを参照してください。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public void setXACloseString(String xa_string);
```

パラメタ

xa_string：

XA_CLOSE 文字列を指定します。

例外

なし

戻り値

なし

setXALocalCommitMode メソッド

説明

XA 使用時，トランザクションが分散トランザクションでないとき，データベースのオートコミットモードを有効にするかどうかを設定します。

このメソッドが呼び出されない場合は，false を設定します。

使用するアプリケーションサーバで FullJTA（J2ee1.3）の機能を使用する場合は，必ず true を設定してください。

形式

```
public void setXALocalCommitMode(boolean AutoCommitMode);
```

パラメタ

AutoCommitMode：

オートコミットを次の値で設定します。

- true
  データベースのオートコミットを有効にします。
- false
  データベースのオートコミットを無効にします。

例外

なし

戻り値

なし

setXAOpenString メソッド

説明

XA_OPEN 文字列を設定します。なお，XA_OPEN 文字列については，各 DBMS のマニュアルを参照してください。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public void setXAOpenString(String xa_string);
```

パラメタ

xa_string：

XA_OPEN 文字列を指定します。

例外

なし

戻り値

なし

setXAThreadMode メソッド

説明

XA 使用時のスレッドモードを設定します。

このメソッドが呼び出されない場合は，false を設定します。

リソースマネジャが提供する XA ライブラリがマルチスレッド対応で，アプリケーションがマルチスレッドで動作する場合は，true を設定してください。

このメソッドは，JdbcDbpsvXADataSource クラスでだけ提供しています。

形式

```
public void setXAThreadMode(boolean mode);
```

パラメタ

mode：

XA 使用時のスレッドモードを次の値で指定します。

- true
  マルチスレッドモード
- false
  シングルスレッドモード

例外

なし

戻り値

なし

14.11　Blob インタフェース

説明

Blob インタフェースでは，主に次の機能を提供します。

- Blob データの取得

Blob インタフェースの提供する各メソッドの詳細，および使用方法については，JavaSoft 提供の JDBC 関連ドキュメントを参照してください。

注意事項

● ResultSet の getBlob メソッド，または CallableStatement の getBlob メソッドで取得した Blob オブジェクト

ResultSet の getBlob メソッド，または CallableStatement の getBlob メソッドで取得した Blob オブジェクトを使用する場合，ロケータアクセスであるかどうかで動作が異なります。

- ロケータアクセスでない場合
  ResultSet の getBlob メソッド，または CallableStatement の getBlob メソッドで取得した時点でのデータを？パラメタの値とします。
- ロケータアクセスの場合
  setBlob() メソッドを呼び出したとき，内部で Blob.getBytes(1, (int)(Blob.length())) を実行します。？パラメタの値は Blob.getBytes(1, (int)(Blob.length())) で得られたデータとなります。

制限事項

次のメソッドは未サポートです。呼び出しをすると SQLException をスローします。

- setBinaryStream
- setBytes
- truncate

15 アプリケーション開発時に使用できるプロパティ

この章では，アプリケーションを開発するときに使用できるプロパティについて説明します。

15.1 バッチアプリケーションで使用できるプロパティ

15.1　バッチアプリケーションで使用できるプロパティ

ここでは，バッチアプリケーションを開発するときに使用できるプロパティについて説明します。

ejbserver.batch.currentdir プロパティ

説明

バッチ実行コマンド（cjexecjob）を実行したカレントディレクトリの絶対パスを取得します。

バッチアプリケーション内で使用してください。

使用例

使用例を示します。

```
File f = new File(System.getProperty("ejbserver.batch.currentdir") +
System.getProperty("file.separator") + "DataFile.txt");
```

付録

付録 A　Java ヒープメモリのリークを起こしやすい JavaAPI クラス

JavaAPI クラスを使用した場合，JavaAPI クラスのインスタンスが Java ヒープメモリ上に作成されます。しかし，表 A-1 および表 A-2 に示す JavaAPI クラスのメソッドを使用して，表の条件に該当する場合，JavaAPI クラス以外のインスタンスも Java ヒープメモリ上に作成されます。

作成されたユーザ作成クラスのインスタンスは，JavaAPI クラスが削除されるまで Java ヒープメモリ上に残ります。このため，ユーザが意識しない間に Java ヒープメモリの使用量が増え，Java ヒープメモリがリークするおそれがあるので注意してください。

ただし，表 A-2 の JavaAPI クラスについては，ユーザ作成クラスのインスタンスを削除することもできます。

Java ヒープメモリがリークしやすいクラス一覧について，インスタンスの削除方法がある場合とない場合に分けて表に示します。表 A-2 については，ユーザ作成のインスタンスの削除方法も示します。

参考

Java ヒープメモリのリークを起こしやすい JavaAPI クラスや条件，およびユーザ作成クラスのインスタンス削除方法については，08-00 以降変更される場合があります。

表 A-1　Java ヒープメモリがリークしやすいクラス一覧（削除方法がない場合）

JavaAPI クラス名	Java ヒープメモリにユーザ作成クラスのインスタンスを作成する条件
java.lang.ClassLoader	defineClass() メソッドの引数に指定した ProtectionDomain がユーザ作成クラスと関連づけられている場合。
java.net.URL	URL クラスインスタンスを生成する際，URLStreamHandlerFactory が作成する URLStreamHander クラスがユーザ作成クラスの場合。
java.text.DateFormat	DateFormat クラスを継承したユーザ作成クラスのコンストラクタで super() を呼び出した場合。
java.util.logging.Level	Level クラスを継承したユーザ作成クラスのコンストラクタで super() を呼び出した場合。
java.util.logging.LogManager	addLogger() メソッドの引数に Logger クラスを継承したユーザ作成クラスを指定した場合。
java.util.logging.Logger	Logger クラスを継承したユーザ作成クラスで getAnonymousLogger() メソッドや setParent() メソッドを呼び出した場合。

JavaAPI クラス名	Java ヒープメモリにユーザ作成クラスのインスタンスを作成する条件
javax.accessibility.AccessibleBundle	toDisplayString() メソッドの引数に指定したリソースバンドル名に対応するユーザ実装のクラスで，そのユーザ実装のクラス内で保持しておくキーと値のペアの値に，ユーザクラスのオブジェクトを設定している場合。
javax.print.attribute.standard.MediaSize	MediaSize クラスを継承したユーザ作成クラスのコンストラクタで super() を呼び出した場合。
java.rmi.server.UnicastRemoteObject	exportObject() メソッドの引数 (RMIClientSocketFactory,RMIServerSocketFactory) にユーザ作成クラスを指定した場合。

表 A-2　Java ヒープメモリがリークしやすいクラス一覧（削除方法がある場合）

JavaAPI クラス名	Java ヒープメモリにユーザ作成クラスのインスタンスを作成する条件	ユーザ作成クラスのインスタンス削除方法
java.beans.Introspector	getBeanInfo() メソッドの引数 (beanClass) にユーザ作成クラスを指定した場合。	flushCaches() メソッドや flushFromCaches（）メソッドを実行する。
java.beans.PropertyEditorManager	registerEditor() メソッドの引数 (editorClass) にユーザ作成クラスを指定した場合。	registerEditor() メソッドの editorClass に null を指定する。
java.util.logging.Logger	addHandler() メソッドの引数に Handler クラスを継承したユーザ作成クラスを指定した場合。	removeHandler() メソッドを実行する。
javax.imageio.ImageReader	addIIOReadWarningListener() メソッドの引数に IIOReadWarningListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOReadWarningListener() メソッドを実行する。
	addIIOReadProgressListener() メソッドの引数に IIOReadProgressListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOReadProgressListener() メソッドを実行する。
	addIIOReadUpdateListener() メソッドの引数に IIOReadUpdateListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOReadUpdateListener() メソッドを実行する。
javax.imageio.ImageWriter	addIIOWriteProgressListener() メソッドの引数に IIOWriteProgressListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOWriteProgressListener() メソッドを実行する。
	addIIOWriteWarningListener() メソッドの引数に IIOWriteWarningListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOWriteWarningListener() メソッドを実行する。
javax.naming.InitialContext	addToEnvironment() メソッドの引数 (propVal) にユーザ作成クラスを指定した場合。	removeFromEnvironment() メソッドを実行する。

JavaAPI クラス名	Java ヒープメモリにユーザ作成クラスのインスタンスを作成する条件	ユーザ作成クラスのインスタンス削除方法
javax.naming.spi.NamingManager	getContinuationContext() メソッドの引数にユーザ作成クラスを指定した場合。	得られた Context の close() を実装して super() を実行する。
javax.print.attribute.HashAttributeSet	add() メソッドの引数にユーザ作成クラスを指定した場合。	remove(Attribute) メソッドや remove(Class) メソッドを実行する。

付録 B　JavaVM 内部で暗黙にスレッドを生成する JavaAPI クラス

J2SE および Java SE の API を使用してスレッドの生成が起きるのは，通常は java.lang.Thread クラスや java.util.concurrent パッケージのスレッド関係のクラスを使用した場合です。

しかし，Java SE API の一部には暗黙にスレッドを生成するものがあります。これらの API を使用した場合，意図しないでスレッド数が増加し C ヒープ領域を消費することがあるため，注意が必要です。ここでは，J2SE 5.0 の JavaVM 内部でスレッドを生成する API や機能を示します。

付録 B.1　スレッド生成処理一覧

ここでは，次の API や機能が JavaVM 内部で生成するスレッドについて説明します。

- GUI 関連の API
- JMX 関連の API
- JNDI 関連の API
- RMI 関連の API
- その他の API や機能

(1) GUI 関連の API

GUI 関連の API で，次のスレッドを生成します。

特定の条件なし
: AWT または Swing の GUI 機能を使用すると，スレッドを生成することがあります。作成するスレッドは Java プロセスで最大六つです。

java.awt.EventQueue クラス
: EventQueue インスタンスごとに，一つのスレッドを生成することがあります。

java.awt.FileDialog クラス
: show() メソッドを呼び出すと，スレッドを一つ作成します。この呼び出しによるスレッドは FileDialog インスタンスごとに最大一つです。

java.awt.image.renderable.RenderableImageProducer クラス
: startProduction() メソッドを呼び出すとスレッドを一つ生成します。

java.awt.print.PrinterJob クラス
: 次のメソッドを呼び出すと，スレッドを一つ作成します。
  - print()（2 種両方）

- printDialog()（2 種両方）
- pageDialog()（2 種両方）

javax.swing.JEditorPane クラス
　JEditorPane インスタンスごとに，一つのスレッドを生成することがあります。

javax.swing.JFileChooser クラス
　javax.swing.JFileChooser インスタンスごとに，一つのスレッドを生成することがあります。

javax.swing.JTable クラス
　print()(4 種類すべて）を呼び出すと，スレッドを一つ生成します。

javax.swing.Timer
　start() または restart() メソッドを呼び出すと，スレッドを一つ生成します。この呼び出しによるスレッドは Timer インスタンスごとに最大一つです。

javax.swing.text.LayoutQueue クラス
　addTask() メソッドを呼び出すと，スレッドを一つ生成します。この呼び出しによるスレッドは LayoutQueue インスタンスごとに最大一つです。

javax.swing.text.AsyncBoxView クラス
　次のメソッドを呼び出すと，API 内部で LayoutQueue インスタンスを作成し，その延長でスレッドを生成することがあります。
- preferenceChanged()
- replace()
- setSize()

javax.swing.text.html.FormView クラス
　submitData() メソッドを呼び出すと，スレッドを一つ生成します。

インプットメソッドの使用
　java.awt.im や java.awt.im.spi で提供されるインプットメソッドを使用するとスレッドを生成することがあります。このスレッドは Java プロセスで最大一つです。

（2）JMX 関連の API

JMX 関連の API で，次のスレッドを生成します。

SNMP(Simple Network Management Protocol) によるリソース監視
　SNMP を使用してリソースを監視・管理している場合，最大で 9 個のスレッドを生成します。

javax.management.remote.rmi. RMIConnection インターフェース
　RMIConnection インターフェースを実装したクラスのインスタンスごとに，一つのスレッドを生成することがあります。

javax.management.remote.rmi.RMIConnector クラス

connect()（2 種両方）メソッドを呼び出すと，確立した接続一つごとにスレッドを二つ生成します。

（3） JNDI 関連の API

JNDI 関連の API で，次のスレッドを生成します。

特定の条件なし

ネーミングやディレクトリの操作で，JNDI コンテキスト一つごとにスレッドを二つ生成します。

javax.naming.event.EventContext インターフェース

EventContext インターフェースを実装したクラスの addNamingListene()（2 種両方）メソッドを呼び出すと，一つのスレッドを作成することがあります。この呼び出しによるスレッドはディレクトリコンテキストごとに最大一つです。

（4） RMI 関連の API

RMI 関連の API で，次のスレッドを生成します。

RMI サーバ側

JavaVM 内部で次のスレッドを生成します。

- 特定の条件なしに最大六つのスレッドを作成します。このスレッドは JavaVM プロセスの終了まで維持されます。
- RMI クライアントから接続待機向けに，リモートオブジェクトをエクスポートした TCP ポート数分のスレッドを生成します。
- RMI クライアントからのメソッド呼び出しがあると，その制御のため一つのスレッドを生成します。
- java.rmi.server.Unreferenced インターフェースを実装したリモートオブジェクトの unreferenced() メソッド呼び出し向けに，一つのスレッドを生成します。

RMI クライアント側

接続している RMI サーバと同数のスレッドを生成します。

（5） そのほかの API や機能

そのほかの API や機能で，次のスレッドを生成します。

HTTP/HTTPS 通信

HTTP/HTTPS 通信をすると Keep Alive の制御のためスレッドを一つ生成します。このスレッドは Java プロセスで最大一つです。

DNS 通信

Windows 環境で DNS 通信をすると，スレッドを一つ生成することがあります。このスレッドは Java プロセスで最大一つです。

ファイナライザの明示実行

java.lang.System クラスや java.lang.Runtime クラスの runFinalization() を呼び出すとスレッドを一つ生成します。

ファイナライズ滞留解消機能

ファイナライズ処理が滞留した場合，滞留解消のためにスレッドを一つ生成します。なお，ファイナライズ滞留解消機能はデフォルトで有効な JavaVM の機能で，-DJP.co.Hitachi.soft.jvm.autofinalizer プロパティで有効と無効を制御できます。

外部プロセスの作成

UNIX 環境で java.lang.ProcessBuilder クラス等によって外部プロセスを作成すると，スレッドを一つ生成します。このスレッドは java.lang.Process インスタンスごとに最大一つです。

java.nio.channels.Selector クラス

Windows 環境で Selector クラスを使用すると，Selector インスタンスへのチャネルの登録数が 1,024 増えるごとに，スレッドを一つ生成します。

java.util.prefs.Preferences クラス

Preferences クラスを使用すると，スレッドを一つ生成することがあります。このスレッドは Java プロセスで最大一つです。

付録 C　このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

付録 C.1　関連マニュアル

アプリケーションサーバのマニュアルについて次に示します。

- Cosminexus **アプリケーションサーバ** V8 **概説（**3020-3-U01**）**
  アプリケーションサーバの概要について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **ファーストステップガイド（**3020-3-U02**）**
  Application Server または Developer を使用して，サンプルプログラムを動かすためのシステムを構築する手順について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **システム設計ガイド（**3020-3-U03**）**
  システム設計時に，システムの目的に応じたシステム構成や運用方法を検討するための指針について説明しています。また，チューニングの方法についても説明しています。
- Cosminexus **アプリケーションサーバ** V8 **システム構築・運用ガイド（**3020-3-U04**）**
  セットアップウィザードおよび Smart Composer 機能を使用したシステムの構築・運用の手順について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **機能解説 基本・開発編** (Web **コンテナ**) **（**3020-3-U05**）**
  アプリケーションサーバで提供する Web コンテナの機能，および Web コンテナに関連する機能（Web サーバ，サーブレット／ JSP など）について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **機能解説 基本・開発編** (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 **アプリケーション設定操作ガイド**

（3020-3-U12）
アプリケーションサーバで動作するアプリケーションの操作方法について説明しています。

- Cosminexus **アプリケーションサーバ** V8 **運用管理ポータル操作ガイド**（3020-3-U13）
運用管理ポータルの使用方法について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **リファレンス コマンド編**（3020-3-U14）
アプリケーションサーバを構築・運用するときに使用するコマンドについて説明しています。
- Cosminexus **アプリケーションサーバ** V8 **リファレンス 定義編 (サーバ定義)**（3020-3-U15）
アプリケーションサーバを構築・運用するとき，またはアプリケーションを開発するときに，使用するファイルのうち，J2EE サーバや Management Server などのサーバの定義に使用するファイルの形式について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **リファレンス 定義編 (アプリケーション／リソース定義)**（3020-3-U16）
アプリケーションサーバを構築・運用するとき，またはアプリケーションを開発するときに使用するファイルのうち，アプリケーションやリソースの属性設定に使用するファイルの形式について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **仮想化システム構築・運用ガイド**（3020-3-U18）
アプリケーションサーバを仮想化したサーバ上に構築する場合の設計，構築，運用の手順について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **アプリケーション開発ガイド**（3020-3-U25）
アプリケーションサーバで動作させるアプリケーションの開発方法について説明しています。
- Cosminexus **アプリケーションサーバ** V8 **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。

- Hitachi Web Server（3020-3-U17）
- TPBroker Version 5 トランザクショナル分散オブジェクト基盤 TPBroker　ユーザーズガイド（3020-3-U19）
- Cosminexus アプリケーションサーバ V8 Cosminexus Reliable Messaging（3020-3-U21）
- Cosminexus アプリケーションサーバ V8 Web サービス開発の手引（3020-3-U31）
- スケーラブルデータベースサーバ HiRDB Version 8 UAP 開発ガイド（3020-6-356）
- スケーラブルデータベースサーバ HiRDB Version 8 SQL リファレンス（3020-6-357）
- HiRDB Version 9 UAP 開発ガイド（3020-6-456）
- HiRDB Version 9 SQL リファレンス（3020-6-457）

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 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
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 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 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 Web サービス開発の手引	Cosminexus アプリケーションサーバ Web サービス開発の手引

正式名称	このマニュアルでの表記
HiRDB Version 9 UAP 開発ガイド	HiRDB UAP 開発ガイド
スケーラブルデータベースサーバ HiRDB Version 8 UAP 開発ガイド	
HiRDB Version 9 SQL リファレンス	HiRDB SQL リファレンス
スケーラブルデータベースサーバ HiRDB Version 8 SQL リファレンス	

付録 C.2　このマニュアルでの表記

このマニュアルで使用している表記と，対応する製品名を次に示します。

表記		製品名
Application Server	Application Server Enterprise	uCosminexus Application Server Enterprise
	Application Server Standard	uCosminexus Application Server Standard
Developer	Developer Professional	uCosminexus Developer Professional
	Developer Standard	uCosminexus Developer Standard
HiRDB または HiRDB サーバ	HiRDB Server	HiRDB Server Version 9
	HiRDB/Parallel Server	HiRDB/Parallel Server Version 8
	HiRDB/Single Server	HiRDB/Single Server Version 8
HiRDB Run Time または HiRDB クライアント		HiRDB/Run Time Version 8
		HiRDB/Run Time Version 9
IPF		Itanium(R) Processor Family
Oracle	Oracle10g	Oracle 10*g*
		Oracle 10*g* R2
		Oracle Database 10*g*
	Oracle11g	Oracle Database 11*g*
		Oracle Database 11*g* R2
	Oracle9i	Oracle9*i*
		Oracle9*i* R2
UNIX	AIX	AIX 5L V5.3
		AIX V6.1
		AIX V7.1
	HP-UX または HP-UX（IPF）	HP-UX 11i V2（IPF）
		HP-UX 11i V3（IPF）

表記			製品名
	Linux	Linux（IPF）	Red Hat Enterprise Linux(R) AS 4（IPF）
			Red Hat Enterprise Linux(R) 5 Advanced Platform（Intel Itanium）
			Red Hat Enterprise Linux(R) 5（Intel Itanium）
		Linux（x86/AMD64 & Intel EM64T）	Red Hat Enterprise Linux(R) AS 4（x86）
			Red Hat Enterprise Linux(R) ES 4（x86）
			Red Hat Enterprise Linux(R) AS 4（AMD64 & Intel EM64T）
			Red Hat Enterprise Linux(R) ES 4（AMD64 & Intel EM64T）
			Red Hat Enterprise Linux(R) 5 Advanced Platform（x86）
			Red Hat Enterprise Linux(R) 5（x86）
			Red Hat Enterprise Linux(R) 5 Advanced Platform（AMD/Intel 64）
			Red Hat Enterprise Linux(R) 5（AMD/Intel 64）
			Red Hat Enterprise Linux(R) Server 6（32-bit x86）
			Red Hat Enterprise Linux(R) Server 6（64-bit x86_64）
			Red Hat Enterprise Linux(R) AS 4（x86）
			Red Hat Enterprise Linux(R) ES 4（x86）
	Solaris		Solaris 10（SPARC）
			Solaris 10（x64）
			Solaris 9（SPARC）
Web Redirector			uCosminexus Web Redirector
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）
	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
DB Connector for Cosminexus RM	DB Connector for Cosminexus Reliable Messaging
Management Server	Cosminexus Management Server
PRF	Cosminexus Performance Tracer
Smart Composer	Cosminexus Smart Composer

このマニュアルで使用している表記と，対応する Java 関連用語を次に示します。

表記	Java 関連用語
DI	Dependency Injection
EAR	Enterprise ARchive
EJB または Enterprise JavaBeans	Enterprise JavaBeans™
EJB QL	EJB™ Query Language

表記	Java 関連用語
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
Java	Java™
Java 2 Runtime Environment, Standard Edition	Java™ 2 Runtime Environment, Standard Edition
Java 2 SDK, Standard Edition	Java™ 2 Software Development Kit, Standard Edition
JavaAPI	Java™ Application Programming Interface
JavaBeans	JavaBeans™
Java EE または Java Platform, Enterprise Edition	Java™ Platform, Enterprise Edition
JavaMail	JavaMail™
Java SE	Java SE™ Platform, Standard Edition
JavaVM	Java™ Virtual Machine
JAXP	Java™ API for XML Processing
JCA	J2EE™ Connector Architecture
JCE	Java™ Cryptography Extension
JDBC	JDBC™
	Java™ Database Connectivity
JDK	JDK™
	Java™ Development Kit
JMX	Java™ Management Extensions
JNDI	Java Naming and Directory Interface™
JNI	Java™ Native Interface
JPA	Java Persistence API
JPQL	Java Persistence Query Language
JSF	JavaServer™ Faces Reference Implementation (RI) Version: 1.1_01 FCS
JSP	JSP™
	JavaServer Pages™

表記	Java 関連用語
JTA	Java™ Transaction API
JTS	Java™ Transaction Service
Servlet またはサーブレット	Java™ Servlet
WAR	Web ARchive

付録 C.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
ASCII	American Standard Code for Information Interchange
BMP	Bean-Managed Persistence
BMT	Bean-Managed Transaction
CA	Certification Authority
CMP	Container-Managed Persistence
CMR	Container-Managed Relationship
CMT	Container-Managed Transaction
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CR	Carriage Return
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CSV	Comma Separated Value
CUI	Character User Interface
DB	Database
DBMS	Database Management System
DD	Deployment Descriptor
DDL	Data Definition Language
DIT	Directory Information Tree
DMZ	Demilitarized Zone
DN	Distinguished Name
DNS	Domain Name System
DoS	Denial of Service attack
DTD	Document Type Definition

英略語	英字での表記
EIS	Enterprise Information System
EUC	Extended UNIX Code
FF	Form Feed
GC	Garbage Collection
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Security
IDE	Integrated Development Environment
IIOP	Internet Inter-Orb Protocol
ISAPI	Internet Server Application Programming Interface
ISO	International Organization for Standardization
JIS	Japanese Industrial Standards
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LF	Line Feed
MDA	Model Driven Architecture
MIB	Management Information Base
OID	Object Identifier
OMG	Object Management Group
ORB	Object Request Broker
OS	Operating System
OTS	Object Transaction Service
PIM	Platform Independent Model
POA	Portable Object Adapter
PSM	Platform Specific Model
RAC	Real Application Clusters
RDB	Relational Database
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SFO	Session Fail Over
SHA	Secure Hash Algorithm
SOA	Service Oriented Architecture
SPI	Service Provider Interface

英略語	英字での表記
SPP	Service Providing Program
SSL	Secure Sockets Layer
TCS	Transaction Context Server
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
UNC	Universal Naming Convention
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
UTF	UCS Transformation Format
VM	Virtual Machine
WSDL	Web Service Description Language
XML	Extensible Markup Language

付録 C.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

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

あ

か

く

せ

た

と

は

ひ

ゆ

れ