HITACHI
Inspire the Next

Cosminexus　リファレンス　API 編

文法書

3020-3-M42-60

マニュアルの購入方法

このマニュアル，および関連するマニュアルをご購入の際は，巻末の「ソフトウェアマニュアルのサービス ご案内」をご参照ください。

■対象製品

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

P-2443-7D74 uCosminexus Application Server Standard 07-60

P-2443-7K74 uCosminexus Application Server Enterprise 07-60

P-2443-7M74 uCosminexus Web Redirector 07-60

P-2443-7S74 uCosminexus Service Platform 07-60 ※

●**適用** OS：Windows Server 2003，Windows Server 2003 R2，Windows Vista，Windows XP

P-2443-7E74 uCosminexus Developer Standard 07-60

P-2443-7F74 uCosminexus Developer Professional 07-60

P-2443-7T74 uCosminexus Service Architect 07-60 ※

P-2443-7U74 uCosminexus Operator 07-60 ※

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

P-2443-7H74 uCosminexus Client 07-60

●**適用** OS：AIX 5L V5.2，AIX 5L V5.3

P-1M43-7D71 uCosminexus Application Server Standard 07-60 ※

P-1M43-7K71 uCosminexus Application Server Enterprise 07-60 ※

P-1M43-7M71 uCosminexus Web Redirector 07-60 ※

P-1M43-7S71 uCosminexus Service Platform 07-60 ※

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

P-1J43-7D71 uCosminexus Application Server Standard 07-60 ※

P-1J43-7K71 uCosminexus Application Server Enterprise 07-60 ※

P-1J43-7M71 uCosminexus Web Redirector 07-60 ※

●**適用** OS：Red Hat Enterprise Linux AS 3（x86），Red Hat Enterprise Linux AS 4（x86），Red Hat Enterprise Linux ES 3（x86），Red Hat Enterprise Linux ES 4（x86），Red Hat Enterprise Linux AS 3（AMD64 & Intel EM64T），Red Hat Enterprise Linux AS 4（AMD64 & Intel EM64T），Red Hat Enterprise Linux ES 3（AMD64 & Intel EM64T），Red Hat Enterprise Linux ES 4（AMD64 & Intel EM64T）

P-9S43-7D71 uCosminexus Application Server Standard 07-60 ※

P-9S43-7K71 uCosminexus Application Server Enterprise 07-60 ※

P-9S43-7M71 uCosminexus Web Redirector 07-60 ※

●**適用** OS：Red Hat Enterprise Linux AS 3（x86），Red Hat Enterprise Linux AS 4（x86），Red Hat Enterprise Linux 5 Advanced Platform（x86），Red Hat Enterprise Linux ES 3（x86），Red Hat Enterprise Linux ES 4（x86），Red Hat Enterprise Linux 5（x86），Red Hat Enterprise Linux AS 3（AMD64 & Intel EM64T），Red Hat Enterprise Linux AS 4（AMD64 & Intel EM64T），Red Hat Enterprise Linux 5 Advanced Platform（AMD/Intel 64），Red Hat Enterprise Linux ES 3（AMD64 & Intel EM64T），Red Hat Enterprise Linux ES 4（AMD64 & Intel EM64T），Red Hat Enterprise Linux 5（AMD/Intel 64）

P-9S43-7S71 uCosminexus Service Platform 07-60 ※

●**適用** OS：Red Hat Enterprise Linux AS 3（IPF），Red Hat Enterprise Linux AS 4（IPF），Red Hat Enterprise Linux 5 Advanced Platform（Intel Itanium）

P-9V43-7D71 uCosminexus Application Server Standard 07-60 ※

P-9V43-7K71 uCosminexus Application Server Enterprise 07-60 ※

P-9V43-7M71 uCosminexus Web Redirector 07-60 ※

●適用 OS：Solaris 9，Solaris 10

P-9D43-7D71 uCosminexus Application Server Standard 07-60 ※

P-9D43-7K71 uCosminexus Application Server Enterprise 07-60 ※

P-9D43-7M71 uCosminexus Web Redirector 07-60 ※

P-9D43-7S71 uCosminexus Service Platform 07-60 ※

※印の製品については，サポート時期をご確認ください。

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

本製品では日立トレース共通ライブラリをインストールします。

■輸出時の注意

本製品を輸出される場合には，外国為替および外国貿易法ならびに米国の輸出管理関連法規などの規制をご確認の上，必要な手続きをお取りください。

なお，ご不明な場合は，弊社担当営業にお問い合わせください。

■商標類

■発行

2006 年 4 月 (第 1 版) 3020-3-M42
2007 年 12 月 (第 4 版) 3020-3-M42-60

■著作権

変更内容

変更内容（3020-3-M42-60）uCosminexus Application Server Standard 07-60，uCosminexus Developer Standard 07-60，uCosminexus Developer Professional 07-60，uCosminexus Client 07-60，uCosminexus Application Server Enterprise 07-60，uCosminexus Web Redirector 07-60，uCosminexus Service Platform 07-60，uCosminexus Service Architect 07-60，uCosminexus Operator 07-60

追加・変更内容	変更個所
バッチアプリケーションで使用できる API の説明を追加した。	1.1，5.1，7.1，7.2，7.3，9.5
@Resource の mappedName で次の設定条件を使用できるようにした。 • javax.jms.ConnectionFactory • javax.jms.TopicConnectionFactory • javax.jms.Topic	10.1
@Resource で次のタイプのリソースを DI できるようにした。 • javax.jms.ConnectionFactory • javax.jms.TopicConnectionFactory • javax.jms.Topic • 管理対象オブジェクトの独自のインタフェース	10.2
バッチアプリケーションで使用できるプロパティ ejbserver.batch.currentdir を追加した。	11.1
次の製品の適用 OS に，Red Hat Enterprise Linux 5 Advanced Platform（x86），Red Hat Enterprise Linux 5（x86），Red Hat Enterprise Linux 5 Advanced Platform（AMD/Intel 64），および Red Hat Enterprise Linux 5（AMD/Intel 64）を追加した。 • uCosminexus Service Platform	－
次の製品の適用 OS に Linux（IPF）を追加した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise • uCosminexus Web Redirector	－
次の製品の適用 OS に Solaris を追加した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise • uCosminexus Web Redirector • uCosminexus Service Platform	－
HP-UX（PA-RISC）のサポート中止に伴い，この OS に適応していた次の製品の記述を削除した。 • uCosminexus Application Server Standard • uCosminexus Application Server Enterprise • uCosminexus Web Redirector	－
次の製品の適用 OS から HP-UX（IPF）を削除した。 • 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-M42-40）uCosminexus Application Server Standard 07-50，uCosminexus Developer Standard 07-50，uCosminexus Developer Professional 07-50，uCosminexus Client 07-50，uCosminexus Application Server Enterprise 07-50，uCosminexus Web Redirector 07-50，uCosminexus Service Platform 07-50，uCosminexus Service Architect 07-50，uCosminexus Operator 07-50

追加・変更内容
監査ログで使用する API を追加した。
セキュリティフィルタで使用する API についての記述を削除した。
アノテーション参照抑止機能を追加した。
AIX 5L V5.1 のサポート中止に伴い，この OS に適応していた次の製品の記述を削除した。 • uCosminexus Application Server Enterprise • uCosminexus Application Server Standard • uCosminexus Service Platform • uCosminexus Web Redirector
Microsoft(R) Windows(R) 2000 Advanced Server Operating System，Microsoft(R) Windows(R) 2000 Datacenter Server Operating System，および Microsoft(R) Windows(R) 2000 Server Operating System のサポート中止に伴い，これらの OS に適応していた次の製品の記述を削除した。 • uCosminexus Application Server Enterprise • uCosminexus Application Server Standard • uCosminexus Client • uCosminexus Developer Professional • uCosminexus Developer Standard • uCosminexus Operator • uCosminexus Service Architect • uCosminexus Service Platform • uCosminexus Web Redirector
Microsoft(R) Windows(R) 2000 Professional Operating System のサポート中止に伴い，この OS に適応していた次の製品の記述を削除した。 • uCosminexus Client • uCosminexus Developer Professional • uCosminexus Developer Standard • uCosminexus Operator • uCosminexus Service Architect
次の製品の適用 OS に，HP-UX（IPF）を追加した。 • uCosminexus Service Platform
次の製品の適用 OS に，HP-UX 11i V3（IPF）を追加した。 • uCosminexus Application Server Enterprise • uCosminexus Application Server Standard • uCosminexus Web Redirector
次の製品の適用 OS に，Red Hat Enterprise Linux ES 3（AMD64 & Intel EM64T），および Red Hat Enterprise Linux ES 4（AMD64 & Intel EM64T）を追加した。 • uCosminexus Application Server Enterprise • uCosminexus Application Server Standard • uCosminexus Service Platform • uCosminexus Web Redirector

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

変更内容（3020-3-M42-20）uCosminexus Application Server Standard 07-10，uCosminexus Application Server Enterprise 07-10，uCosminexus Developer Standard 07-10，uCosminexus Developer Professional 07-10，uCosminexus Client 07-10，uCosminexus Web Redirector 07-10，uCosminexus Service Platform 07-10，uCosminexus Service Architect 07-10，uCosminexus Operator 07-10

追加・変更機能
DataSource クラスに次のメソッドを追加した。 • getBufferPoolSize メソッド • setBufferPoolSize メソッド
@AroundInvoke，@ExcludeDefaultInterceptors，@ExcludeClassInterceptors および @Interceptors のパッケージを，javax.interceptor に変更した。
Java Type によって異なる DD の対応の説明に，type 属性として javax.xml.ws.Service と javax.jws.WebService を追加した。
uCosminexus Application Server Standard，uCosminexus Application Server Enterprise および uCosminexus Web Redirector の適用 OS に，HP-UX（PA-RISC）を追加した。
uCosminexus Application Server Standard，uCosminexus Application Server Enterprise および uCosminexus Web Redirector の適用 OS に，Linux（IPF）を追加した。
uCosminexus Service Platform の適用 OS に，AIX を追加した。

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。
Cosminexus では，次に示すプログラムプロダクトを使用してアプリケーションサーバを構築，運用します。なお，これらのプログラムプロダクトを使用して構築したシステムを，Cosminexus システムといいます。

- P-1J43-7D71　uCosminexus Application Server Standard
- P-1J43-7K71　uCosminexus Application Server Enterprise
- P-1J43-7M71　uCosminexus Web Redirector
- P-1M43-7D71　uCosminexus Application Server Standard
- P-1M43-7K71　uCosminexus Application Server Enterprise
- P-1M43-7M71　uCosminexus Web Redirector
- P-1M43-7S71　uCosminexus Service Platform
- P-2443-7D74　uCosminexus Application Server Standard
- P-2443-7E74　uCosminexus Developer Standard
- P-2443-7F74　uCosminexus Developer Professional
- P-2443-7H74　uCosminexus Client
- P-2443-7K74　uCosminexus Application Server Enterprise
- P-2443-7M74　uCosminexus Web Redirector
- P-2443-7S74　uCosminexus Service Platform
- P-2443-7T74　uCosminexus Service Architect
- P-2443-7U74　uCosminexus Operator
- P-9D43-7D71　uCosminexus Application Server Standard
- P-9D43-7K71　uCosminexus Application Server Enterprise
- P-9D43-7M71　uCosminexus Web Redirector
- P-9D43-7S71　uCosminexus Service Platform
- P-9S43-7D71　uCosminexus Application Server Standard
- P-9S43-7K71　uCosminexus Application Server Enterprise
- P-9S43-7M71　uCosminexus Web Redirector
- P-9S43-7S71　uCosminexus Service Platform
- P-9V43-7D71　uCosminexus Application Server Standard
- P-9V43-7K71　uCosminexus Application Server Enterprise
- P-9V43-7M71　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の機能で使用するファイルについて説明しています。

- Cosminexus Component Container
- Cosminexus Component Container - Client
- Cosminexus Operator Plug-in

- Cosminexus Component Container - Redirector
- Cosminexus Component Transaction Monitor
- Cosminexus DABroker Library
- 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。OS ごとの違いがある場合の表記方法については，「適用 OS の違いによる機能相違点の表記」を参照してください。

■対象読者

このマニュアルは，Cosminexus 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。
なお，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 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

また，このマニュアルは，マニュアル「Cosminexus　機能解説」を理解していることを前提としていますので，あらかじめお読みいただくことをお勧めします。

■マニュアルの構成

このマニュアルは，次に示す章から構成されています。

第 1 章　API とタグライブラリの概要
Cosminexus 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ています。

第 2 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ています。

第 3 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ています。

第 4 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ています。

第 5 章　ユーザログ機能で使用する API
ユーザログ機能で使用する API について説明しています。

第 6 章　性能解析トレースで使用する API

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 について説明しています。

第 7 章　監査ログ出力で使用する API

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 について説明しています。

第 8 章　JavaVM で使用する API

JavaVM で使用する API について説明しています。

第 9 章　Cosminexus DABroker Library で使用する API

Cosminexus DABroker Library で使用する JDBC インタフェースおよび API について説明しています。

第 10 章　Cosminexus が対応しているアノテーションおよび Dependency Injection

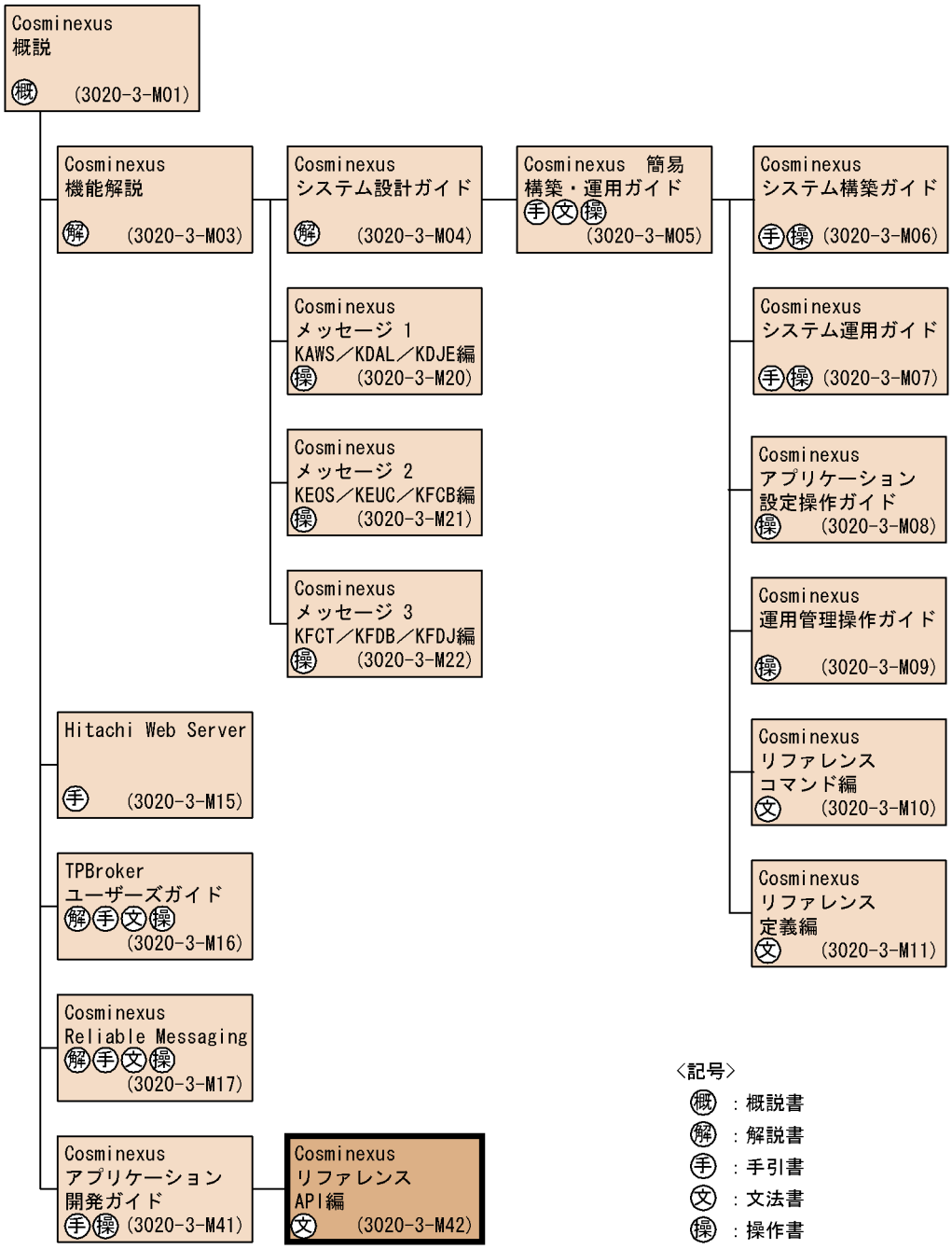
Cosminexus が対応しているアノテーションおよび Dependency Injection について説明しています。

第 11 章　アプリケーション開発時に使用できるプロパティ

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ています。

■関連マニュアル

Cosminexus のマニュアル体系について，次の図に示します。



マニュアル体系図で示した関連マニュアルについて，それぞれの位置づけを次に示します。

Cosminexus　概説

Cosminexus の製品概要について説明しています。

Cosminexus　**機能解説**

Cosminexus Component Container の機能を中心に，uCosminexus Application Server の概要と提供する機能について説明しています。

Cosminexus　**システム設計ガイド**

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　**簡易構築・運用ガイド**

セットアップウィザードおよび Smart Composer 機能を使用して，システムを構築・運用する手順について説明しています。また，セットアップウィザードおよび Smart Composer 機能が提供するコマンドやファイルについても説明しています。

Cosminexus　**システム構築ガイド**

システム構築時に必要な機能の設定方法について説明しています。

Cosminexus　**システム運用ガイド**

Cosminexus を使用したシステムの運用方法のうち，Smart Composer 機能を使用する運用以外の方法について説明しています。

Cosminexus　**アプリケーション設定操作ガイド**

Cosminexus Component Container のサーバ管理コマンド，および Server Plug-in を使用した操作について説明しています。

Cosminexus　**運用管理操作ガイド**

Cosminexus Component Container の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　**リファレンス　コマンド編**

Cosminexus のシステムを構築・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　**リファレンス　定義編**

Cosminexus のシステムを構築・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形式について説明しています。

Cosminexus　**メッセージ** 1　KAWS ／ KDAL ／ KDJE **編，**Cosminexus　**メッセージ** 2　KEOS ／ KEUC ／ KFCB **編，**Cosminexus　**メッセージ** 3　KFCT ／ KFDB ／ KFDJ **編**

Cosminexus で出力されるメッセージについて説明しています。

Hitachi Web Server

Hitachi Web Server（Web サーバ）の構築，管理方法について説明しています。

TPBroker　**ユーザーズガイド**

Cosminexus TPBroker の概要，機能，運用方法について説明しています。

Cosminexus Reliable Messaging

Cosminexus RM を使用したメッセージの非同期通信によるアプリケーションの連携方法に

ついて説明しています。

Cosminexus　アプリケーション開発ガイド

構築した Cosminexus のシステムで動作させる，アプリケーションの開発方法について説明しています。

また，マニュアル体系図に示したマニュアル以外で，このマニュアルと関連する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。

- 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 7 UAP 開発ガイド（UNIX(R)/Windows(R) 用）（3000-6-276）
- 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）
- スケーラブルデータベースサーバ HiRDB Version 7 SQL リファレンス（UNIX(R)/Windows(R) 用）（3000-6-277）

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
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 UAP 開発ガイド
スケーラブルデータベースサーバ HiRDB Version 7 UAP 開発ガイド（UNIX(R)/Windows(R) 用）	
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	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 7 SQL リファレンス（UNIX(R)/Windows(R) 用）	

■マニュアル体系の変更について

07-00 では，06-70 のマニュアル「Cosminexus Version 6　リファレンス（Windows(R) 用）（3020-3-E56）」と「Cosminexus Version 6　リファレンス（UNIX(R) 用）（3000-3-987）」の内容を 3 分冊して，マニュアル体系を変更しました。

目次構成の対応は次のようになっています。

06-70 のマニュアル	07-00 以降のマニュアル
第 1 編　コマンド	マニュアル「Cosminexus　リファレンス　コマンド編」へ移動
第 2 編　ファイル	マニュアル「Cosminexus　リファレンス　定義編」へ移動
第 3 編　オプション	マニュアル「Cosminexus　リファレンス　定義編」へ移動
第 4 編　API ／タグライブラリ	マニュアル「Cosminexus　リファレンス　API 編」へ移動
付録 A　ログ出力のデフォルトの設定	削除
付録 B　TPBroker の運用支援機能用定義ファイル	削除

06-70 のマニュアル	07-00 以降のマニュアル
付録 C　Naming Manager 定義ファイル	削除
付録 D　拡張 MIB オブジェクト定義ファイル	マニュアル「Cosminexus　リファレンス　定義編」へ移動
付録 E　CMP のマッピング一覧	削除
付録 F　Web アプリケーション用 DD（web.xml）	マニュアル「Cosminexus　リファレンス　定義編」へ移動
付録 G　csecanalyz コマンドによるシステムのセキュリティ設定のチェック内容	削除
付録 H　dabsetup（Cosminexus DABroker Library のセットアップ）※	マニュアル「Cosminexus　リファレンス　コマンド編」へ移動

注※
マニュアル「Cosminexus Version 6　リファレンス（UNIX(R) 用）」だけにある目次項目です。

■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 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　アプリケーション開発ガイド」の Developer Standard 使用時の注意事項に関する説明を参照してください。

■このマニュアルでの表記

このマニュアルで使用する表記と，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

表記			製品名
Developer	Developer Professional		uCosminexus Developer Professional
	Developer Standard		uCosminexus Developer Standard
HiRDB または HiRDB サーバ	HiRDB/Parallel Server		HiRDB/Parallel Server Version 7
			HiRDB/Parallel Server Version 8
	HiRDB/Single Server		HiRDB/Single Server Version 7
			HiRDB/Single Server Version 8
HiRDB Run Time または HiRDB クライアント			HiRDB/Run Time Version 7
			HiRDB/Run Time Version 8
IPF			Itanium(R) Processor Family
Oracle	Oracle9i		Oracle9*i*
			Oracle9*i* R2
	Oracle10g		Oracle 10*g*
			Oracle 10*g* R2
SQL Server	SQL Server 2000		Microsoft(R) SQL Server(TM) 2000
	SQL Server 2005		Microsoft(R) SQL Server(TM) 2005
SQL Server の JDBC ドライバ	SQL Server 2000 Driver for JDBC		Microsoft(R) SQL Server(TM) 2000 Driver for JDBC
	SQL Server 2005 JDBC Driver		Microsoft(R) SQL Server(TM) 2005 JDBC Driver
UNIX	AIX		AIX 5L V5.2
			AIX 5L V5.3
	HP-UX	HP-UX（IPF）	HP-UX 11i V2（IPF）
			HP-UX 11i V3（IPF）
	Linux	Linux（IPF）	Red Hat Enterprise Linux AS 3（IPF）
			Red Hat Enterprise Linux AS 4（IPF）
			Red Hat Enterprise Linux 5 Advanced Platform（Intel Itanium）
		Linux（x86 ／ AMD64 & Intel EM64T）	Red Hat Enterprise Linux AS 3（x86）
			Red Hat Enterprise Linux AS 4（x86）
			Red Hat Enterprise Linux 5 Advanced Platform（x86）
			Red Hat Enterprise Linux ES 3（x86）
			Red Hat Enterprise Linux ES 4（x86）

表記			製品名
			Red Hat Enterprise Linux 5（x86）
			Red Hat Enterprise Linux AS 3（AMD64 & Intel EM64T）
			Red Hat Enterprise Linux AS 4（AMD64 & Intel EM64T）
			Red Hat Enterprise Linux 5 Advanced Platform（AMD/Intel 64）
			Red Hat Enterprise Linux ES 3（AMD64 & Intel EM64T）
			Red Hat Enterprise Linux ES 4（AMD64 & Intel EM64T）
			Red Hat Enterprise Linux 5（AMD/Intel 64）
	Solaris		Solaris 9
			Solaris 10
Web Redirector			uCosminexus Web Redirector
Windows Server 2003	Windows Server 2003 Enterprise Edition		Microsoft(R) Windows Server(R) 2003，Enterprise Edition Operating System（x86）
	Windows Server 2003 Standard Edition		Microsoft(R) Windows Server(R) 2003，Standard Edition Operating System（x86）
Windows Server 2003 R2	Windows Server 2003 R2 Enterprise Edition		Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition Operating System（x86）
	Windows Server 2003 R2 Standard Edition		Microsoft(R) Windows Server(R) 2003 R2, Standard Edition Operating System（x86）
Windows Server 2003（x64）	Windows Server 2003 Enterprise x64 Edition		Microsoft(R) Windows Server(R) 2003，Enterprise x64 Edition Operating System
	Windows Server 2003 Standard x64 Edition		Microsoft(R) Windows Server(R) 2003，Standard x64 Edition Operating System
Windows Server 2003 R2（x64）	Windows Server 2003 R2 Enterprise x64 Edition		Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition Operating System
	Windows Server 2003 R2 Standard x64 Edition		Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition Operating System
Windows Vista	Windows Vista Business		Microsoft(R) Windows Vista(R) Business
	Windows Vista Enterprise		Microsoft(R) Windows Vista(R) Enterprise
	Windows Vista Ultimate		Microsoft(R) Windows Vista(R) Ultimate
Windows XP			Microsoft(R) Windows(R) XP Professional Operating System
XDM/RD E2			VOS3 XDM/RD E2

なお，Windows Server 2003，Windows Server 2003 R2，Windows Server 2003（x64），

Windows Server 2003 R2（x64），Windows Vista，および Windows XP を総称して Windows と表記することがあります。
このマニュアルで使用している表記と，対応する Cosminexus の機能名を次に示します。

表記	Cosminexus 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
Server Plug-in	Cosminexus Server Plug-in
Smart Composer	Cosminexus Smart Composer

このマニュアルで使用している表記と，対応する Java 関連用語を次に示します。

表記	Java 関連用語
DI	Dependency Injection
EAR	Enterprise ARchive
EJB または Enterprise JavaBeans	Enterprise JavaBeans™
EJB QL	EJB™ Query Language
J2EE または Java 2 Platform, Enterprise Edition	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
JavaBeans	JavaBeans™
JavaMail	JavaMail™
JAXP	Java™ API for XML Processing
JCA	J2EE™ Connector Architecture
JCE	Java™ Cryptography Extension

表記	Java 関連用語
JDBC	JDBC™
	Java™ Database Connectivity
JDK	Java™ Development Kit
JNDI	Java Naming and Directory Interface™
JNI	Java™ Native Interface
JSF	JavaServer™ Faces Reference Implementation (RI) Version: 1.1_01 FCS
JSP	JavaServer Pages™
JTA	Java™ Transaction API
JTS	Java™ Transaction Service
Servlet またはサーブレット	Java™ Servlet
WAR	Web ARchive

■適用 OS の違いによる機能相違点の表記

このマニュアルは，適用 OS が Windows，AIX，HP-UX，Linux，および Solaris の製品に対応します。OS によって記述を書き分ける場合，次に示す表記を使用して，それぞれの説明に OS 名を明記しています。

表記	意味
Windows の場合	Windows に該当する表記です。
AIX の場合	AIX に該当する表記です。
HP-UX の場合	HP-UX に該当する表記です。
Linux の場合	Linux に該当する表記です。
Solaris の場合	Solaris に該当する表記です。
UNIX の場合	UNIX（AIX，HP-UX，Linux，Solaris）に該当する表記です。

■このマニュアルで使用している略語

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

英略語	英字での表記
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
DIT	Directory Information Tree
DMZ	Demilitarized Zone
DN	Distinguished Name
DNS	Domain Name System
DoS	Denial of Service attack
DTD	Document Type Definition
EIS	Enterprise Information System
EJB QL	EJB Query Language
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

英略語	英字での表記
JAR	Java Archive
JDBC	Java Database Connectivity
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
SOAP	Simple Object Access Protocol
SPI	Service Provider Interface
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

英略語	英字での表記
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
UTF	UCS Transformation Format
VM	Virtual Machine
WSDL	Web Service Description Language
XML	Extensible Markup Language

■このマニュアルで使用している記号

このマニュアルで使用する記号について次に示します。

記号	意　味
|	横に並べられた複数の項目に対する項目間の区切りを示し，「または」を意味します。 （例） A | B A または B を指定することを示します。
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを示します。項目が横に並べられ，記号|で区切られている場合は，そのうちの一つを選択します。 （例） {A | B | C} A，B または C のどれかを指定することを示します。
[ ]	この記号で囲まれている項目は省略してもよいことを示します。複数の項目が横に並べて記述されている場合には，すべてを省略するか，記号 { } と同じくどれか一つを選択します。 （例１） [A] 「何も指定しない」か「A を指定する」ことを示します。 （例２） [B | C] 「何も指定しない」か「B または C を指定する」ことを示します。
…	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。
< >	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。

■このマニュアルで使用している構文要素

このマニュアルで使用する構文要素の種類を次に示します。

種類	定義
英字	A ～ Z　a ～ z
英小文字	a ～ z
英大文字	A ～ Z

種類	定義
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。

■常用漢字以外の漢字の使用について

このマニュアルでは，常用漢字を使用することを基本としていますが，次に示す用語については，常用漢字以外の漢字を使用しています。
鍵（かぎ）　個所（かしょ）　必須（ひっす）

■ 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。

目次

1 API とタグライブラリの概要

この章では，Cosminexus 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

Cosminexus 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
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。	2 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。	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
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。	5 章
性能解析トレースで使用する API	性能解析トレースで Cosminexus システム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。	6 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。	7 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。	8 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。	9 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，「10.　Cosminexus 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。	5 章
性能解析トレースで使用する API	性能解析トレースで Cosminexus システム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。	6 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。	7 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。	8 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。	9 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 API の記述形式

2 章，および 4 章から 9 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。

1.3 タグライブラリの記述形式

3 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 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。

2.1　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 2-1　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

2.2 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

2.3　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

2.4 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。

2.5 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

2.6 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

2.7　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。

2.8 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。

2.9 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

2.10 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。

2.11 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，「2.32　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。

2.12　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。

2.13 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

2.14 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。

2.15　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 2-2　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

2.16 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

2.17 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。

2.18　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。

2.19 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

2.20 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。

2.21　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。

2.22 WebLogoutCallback クラス

説明

Web アプリケーションに渡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 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。

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

2.23 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。

2.24 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

2.25 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。

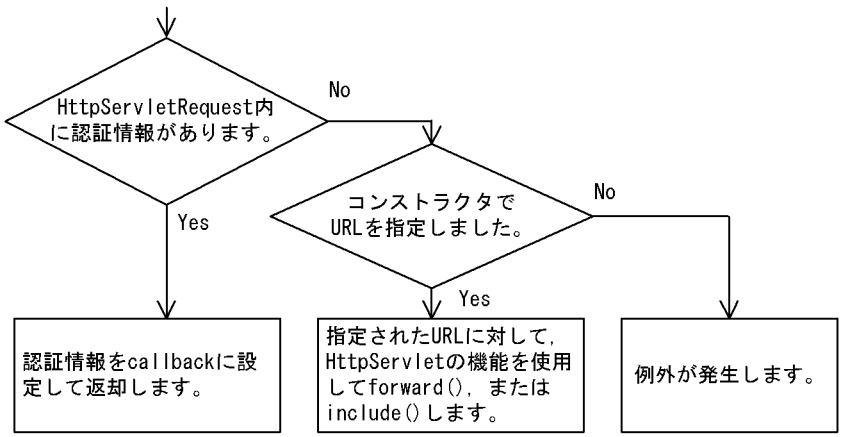
戻り値

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 2-1　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。

2.26　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。

2.27 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。

2.28　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。

2.29 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

2.30 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。

2.31 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。

2.32　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 2-3　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 2-4　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。

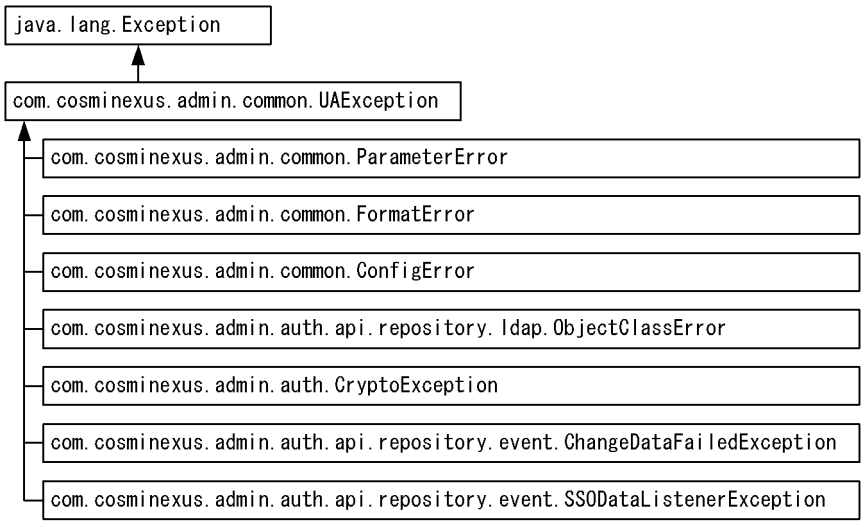
例外名	エラーメッセージ文字列	発生条件
	invalid data	• ユーザ ID ／パスワードに誤りがあり，認証できません。 • 証明書から取り出したユーザ ID に対応したエントリが，リポジトリ内にありません。
	no data	該当セッション内で認証済みである場合に，呼び出そうとしたレルムに対応したシングルサインオン用認証情報が定義されていません。
javax.security.auth.login.LoginException	invalid parameter	Credential を作成しようとしたときの属性名と属性の一覧の指定内容に次のような誤りがあります。 • 属性名が指定されていません。 • 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。

例外名	エラーメッセージ文字列	発生条件
	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。
	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　リファレンス　定義編」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 2-2 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 2-5 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。

項番	例外名	内容
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。

3 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。

3.1 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 3-1 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: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。
<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:chpw/> タグ	指定したユーザのパスワードを変更します。

この章の 3.2 ～ 3.12 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。

3.2 <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 3-2 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。

3.3 <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,「3.2 <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 3-3 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。

3.4 <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 3-4 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　リファレンス　定義編」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
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

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※

指定する場合はどちらかを指定します。

3.5　<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 3-5　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。

3.6 <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 3-6 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。

3.7 <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 3-7 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。

3.8 <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 3-8 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。

3.9 <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 3-9 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。

3.10 <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 3-10 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。

3.11 <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 3-11 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。

3.12 <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 3-12 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。

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　リファレンス　定義編」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　機能解説」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，Cosminexus 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 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。

5.1 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 5-1 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。

5.2 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。
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。

メソッド名	機能
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。

6 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。

6.1 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 6-1 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。

6.2 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
  <Cosminexus 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 48)。

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の取得については，マニュアル「Cosminexus　機能解説」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　システム運用ガイド」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

7 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。

7.1 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 7-1 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
```

7.2 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 7-2 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，Cosminexus 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 7-3　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）
動作情報	0 ～ 32	任意。 なお，Cosminexus 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 7-4　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。

項目	推奨値	
	長さ（バイト）	文字種
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，Cosminexus 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 7-6　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）
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 7-3 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。
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 7-4 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 7-5 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"	
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"	

フィールド名	実際の値	意味
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 7-5 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。
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。

監査事象の種別	動作情報	意味
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 7-6　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，Cosminexus 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 Cosminexus 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，Cosminexus 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

7.3 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，「7.2　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　リファレンス　定義編」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　システム運用ガイド」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 7-7　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

7.4 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。

8 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 に準拠しています。対応する Sun Microsystems 社製の JDK のバージョンは JDK 5.0 です。JDK 5.0 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 のドキュメントを参照してください。

8.1　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 8-1　JavaVM で使用する API の一覧

クラス名	機能
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。

8.2 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。

9 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。

9.1 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 9-1 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

9.2 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　機能解説」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 9-2　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す。	○	○
jdbcCompliant	無条件に true を返却します。	○	○

（凡例）

○：該当します。

9.3 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 9-3 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。

9.4 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。

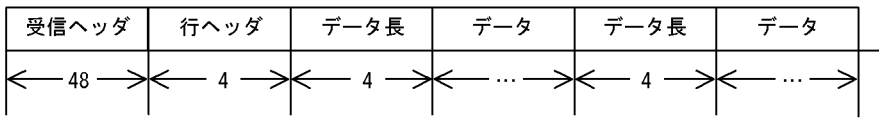
ResultSetMetaData や DatabaseMetaData の情報取得では有効になりません。

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 9-4　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 9-5 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

9.5 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，Java VM の返す値となります。

表 9-6 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。

setXXX メソッド	HiRDB 列属性		
	TIME	DATE	TIMESTAMP
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。
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 型の精度により，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 9-7　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	×	×	○	×	×	－	－	－	－	－	－	－

setXXX メソッド	HiRDB のデータ型											
	SMALLINT	INTEGER	FLOAT	REAL	DECIMAL	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY	BLOB
setBigDecimal	×	×	○	○	×	−	−	−	−	−	−	−
setBoolean	○	○	○	○	×	−	−	−	−	−	−	−
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 9-8 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	−	−	−	−	−	−	−	−	−	○	−	−

setObject メソッド	HiRDB のデータ型											
	SMALLINT	INTEGER	FLOAT	REAL	DECIMAL	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY	BLOB
byte[]	－	－	－	－	－	－	－	－	－	－	－	－
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　機能解説」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　リファレンス　コマンド編」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 9-9　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。

9.6　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 9-10　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。

9.7 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 9-11 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 throw 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。

9.8 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 9-12 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。

9.9 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 9-13 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 9-14 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 9-14 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 9-14　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

9.10 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 9-15 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 9-16 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　機能解説」の受信バッファのプーリングに関する説明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。

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。

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。

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 を行う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 することができます（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 の位置づけ子（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。

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。

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。

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　機能解説」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　メッセージ 3　KFCT ／ KFDB ／ KFDJ 編」を参照してください。また，受信バッファプール数の設定については，マニュアル「Cosminexus　機能解説」の受信バッファのプーリングに関する説明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　メッセージ 3　KFCT ／ KFDB ／ KFDJ 編」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

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

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

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，Java VM 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　リファレンス　定義編」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 することができます（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。

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

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 を行う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

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，Cosminexus の J2EE 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

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

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

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

9.11 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。呼び出しを行うと SQLException をスローします。

- setBinaryStream
- setBytes
- truncate

10 Cosminexus が対応しているアノテーションおよび Dependency Injection

この章では，Cosminexus 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　機能解説」を参照してください。

10.1　Cosminexus が対応しているアノテーション

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。

Cosminexus が対応しているアノテーションの一覧を次の表に示します。

表 10-1　Cosminexus が対応しているアノテーションの一覧

パッケージ	アノテーション	アノテーションの説明	属性	属性のデフォルト値	属性の説明
javax.annotation	@Resource	リソースへの参照を宣言します。 クラス，メソッド，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。	String name()	• 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	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。
			Class type() ※1	• 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	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 authenticationType()	CONTAINER	リソースに使用する認証タイプを設定します。
			boolean shareable()	true	リソースを共用するかどうかを設定します。
			String mappedName() ※2	""	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。
			String description()	""	リソースの説明を設定します。
	@Resources	@Resource を複数設定します。 なお，クラスにだけ設定できます。	Resource[] value	なし	複数のリソース（@Resource）を定義します。

パッケージ	アノテーション	アノテーションの説明	属性	属性のデフォルト値	属性の説明
	@PostConstruct	Enterprise Bean インスタンス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。	－	なし	－
	@PreDestroy	Enterprise Bean インスタンス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。	－	なし	－
javax.annotation.security	@RunAs	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。	String value()	なし	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。
	@DeclareRoles	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。	String[] value()	なし	参照するセキュリティロール名を設定します。
	@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。	String[] value()	なし	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。
	@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。	－	なし	－
	@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。	－	なし	－
javax.ejb	@Stateless	Stateless Session Bean のクラスに設定します。	String name()	Stateless Session Bean のパッケージを除いたクラス名	Stateless Session Bean の名称を設定します。
			String mappedName() ※3	なし	－
			String description()	""	Stateless Session Bean の説明を設定します。

パッケージ	アノテーション	アノテーションの説明	属性	属性のデフォルト値	属性の説明
	@Stateful	Stateful Session Bean のクラスに設定します。	String name()	Stateful Session Bean のパッケージを除いたクラス名	Stateful Session Bean の名称を設定します。
			String mappedName() ※3	なし	－
			String description()	""	Stateful Session Bean の説明を設定します。
	@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。	String value()	""	対応する create<METHOD>() 名を設定します。
	@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。	boolean retainIfException()	false	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。
	@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。 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。	Class[] 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。

パッケージ	アノテーション	アノテーションの説明	属性	属性のデフォルト値	属性の説明
	@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。 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。	Class[] 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。
	@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。	Class value()	なし	リモートホーム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。	Class value()	なし	ローカルホームインタフェースを設定します。
	@TransactionManagement	Enterprise Bean のトランザクション管理種別を設定します。 なお，クラスにだけ設定できます。	TransactionManagementType value()	CONTAINER	トランザクション管理種別を設定します。
	@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。 クラス，メソッドに設定できます。	TransactionAttributeType value()	REQUIRED	トランザクション属性を設定します。
	@PostActivate ※4	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。	—	なし	—

パッケージ	アノテーション	アノテーションの説明	属性	属性のデフォルト値	属性の説明
	@PrePassivate ※4	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。	―	なし	―
	@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。	―	なし	―
	@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。	boolean rollback()	false	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。
	@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。 クラス，メソッド，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。	String name()	• 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	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。
			Class beanInterface()	• 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	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。

パッケージ	アノテーション	アノテーションの説明	属性	属性のデフォルト値	属性の説明
			String 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。
			String mappedName() ※3	なし	－
			String description()	""	参照する EJB の説明を設定します。
	@EJBs	@EJB を複数設定します。 なお，クラスにだけ設定できます。	EJB[] value()	なし	@EJB を設定します。
javax.interceptor	@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。	－	なし	－
	@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。	－	なし	－
	@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。	－	なし	－
	@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。	Class[] value()	なし	適用するインターセプタクラスを設定します。

（凡例）

－：該当しません。

注※ 1

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 の対応を表 10-2 に示します。

注※ 2

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 属性の設定条件を表 10-3 に示します。

注※ 3

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。

注※ 4

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。

表 10-2 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

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
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
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。

表 10-3　@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	×

設定条件（Java Type，リソースなど）	使用可否※1
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。

10.2　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，表 10-4 に示すリソースのタイプを DI できます。

表 10-4　@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。

11 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。

11.1 バッチアプリケーションで使用できるプロパティ

11.1 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");
```

索引

A

B

C

D

E

G

H

I

J

L

M

N

O

P

R

S

U

W

あ

か

と

ひ

れ

ソフトウェアマニュアルのサービス ご案内

１．マニュアル情報ホームページ

ソフトウェアマニュアルの情報をインターネットで公開しています。

URL　http://www.hitachi.co.jp/soft/manual/

ホームページのメニューは次のとおりです。

■マニュアル一覧	日立コンピュータ製品マニュアルを製品カテゴリ，マニュアル名称，資料番号のいずれかから検索できます。
■CD-ROMマニュアル	日立ソフトウェアマニュアルと製品群別CD-ROMマニュアルの仕様について記載しています。
■マニュアルのご購入	マニュアルご購入時のお申し込み方法を記載しています。
■オンラインマニュアル	一部製品のマニュアルをインターネットで公開しています。
■サポートサービス	ソフトウェアサポートサービスお客様向けページでのマニュアル公開サービスを記載しています。
■ご意見・お問い合わせ	マニュアルに関するご意見，ご要望をお寄せください。

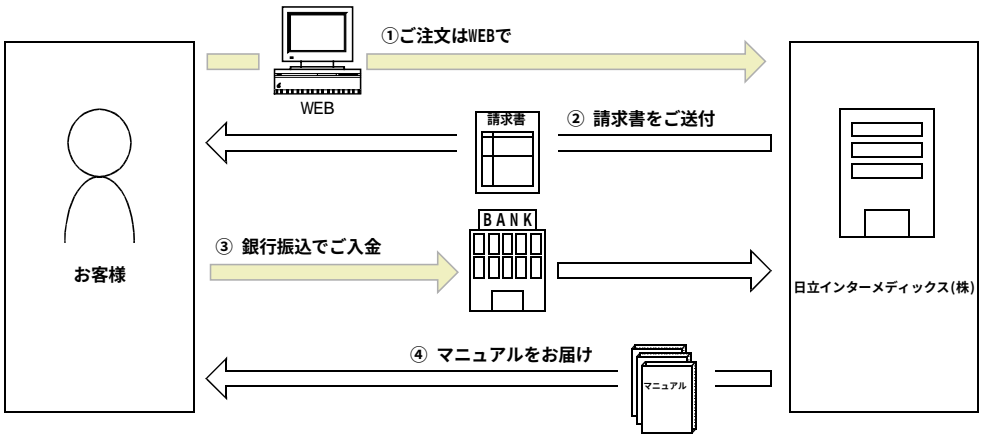
２．インターネットでのマニュアル公開

2 種類のマニュアル公開サービスを実施しています。

(1) マニュアル情報ホームページ「オンラインマニュアル」での公開
　製品をよりご理解いただくためのご参考として，一部製品のマニュアルを公開しています。

(2) ソフトウェアサポートサービスお客様向けページでのマニュアル公開
　ソフトウェアサポートサービスご契約のお客様向けにマニュアルを公開しています。公開しているマニュアルの一覧，本サービスの対象となる契約の種別などはマニュアル情報ホームページの「サポートサービス」をご参照ください。

３．マニュアルのご注文



① マニュアル情報ホームページの「マニュアルのご購入」にアクセスし，お申し込み方法をご確認のうえ WEB からご注文ください。ご注文先は日立インターメディックス(株)となります。

② ご注文いただいたマニュアルについて請求書をお送りします。

③ 請求書の金額を指定銀行へ振り込んでください。

④ 入金確認後 7 日以内にお届けします。在庫切れの場合は，納期を別途ご案内いたします。