HITACHI
Inspire the Next

uCosminexus DocumentBroker Version 5 リファレンス API

解説・文法書

3021-3-403-10

■対象製品

R-1595F-43 uCosminexus DocumentBroker Developer 05-10（適用 OS：Windows Server 2008 R2，Windows Server 2012，Windows 7 (x86)，Windows 7 (x64)，Windows 8 (x86)，Windows 8 (x64)）

■輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問い合わせください。

■商標類

■マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

製品名称	略称	
Microsoft(R) Active Directory(R)	Active Directory	
Microsoft(R) Internet Explorer(R)	Internet Explorer	
Microsoft(R) Office Word	Word	
Microsoft(R) Word		
Windows(R) 8 Pro (32 ビット版)	Windows 8	Windows
Windows(R) 8 Enterprise (32 ビット版)		
Windows(R) 8 Pro (64 ビット版)		
Windows(R) 8 Enterprise (64 ビット版)		
Microsoft(R) Windows(R) 7 Professional 日本語版 (32 ビット版)	Windows 7	
Microsoft(R) Windows(R) 7 Enterprise 日本語版 (32 ビット版)		
Microsoft(R) Windows(R) 7 Ultimate 日本語版 (32 ビット版)		
Microsoft(R) Windows(R) 7 Professional 日本語版 (64 ビット版)		
Microsoft(R) Windows(R) 7 Enterprise 日本語版 (64 ビット版)		
Microsoft(R) Windows(R) 7 Ultimate 日本語版 (64 ビット版)		
Microsoft(R) Windows Server(R) 2012 Datacenter 日本語版	Windows Server 2012	
Microsoft(R) Windows Server(R) 2012 Standard 日本語版		

製品名称	略称	
Microsoft(R) Windows Server(R) 2008 R2 Enterprise 日本語版	Windows Server 2008 R2	
Microsoft(R) Windows Server(R) 2008 R2 Standard 日本語版		

■発行

2013 年 4 月 3021-3-403-10

■著作権

変更内容

変更内容（3021-3-403-10）

追加・変更内容	変更個所
入力ストリームを使用した文書の操作が可能になりました。	1.10.14，4.2，4.2.12，4.2.20，5.4，5.4.3，5.4.4，5.4.5，5.14，5.15，5.25，5.25.6，5.25.7，5.25.13，5.25.14，6.4.10
JDBC コネクションの取得	6.6.10
文書空間に接続中のユーザの特権を変更するメソッドが追加になりました。	6.6.11，9.1.12
前提 OS に Windows Server 2012 および Windows 8 を追加しました。	―

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，次に示すプログラムプロダクトで提供する DocumentBroker クラスライブラリの詳細，インターフェースの詳細，メソッドの文法，およびメッセージの詳細についてリファレンス形式で説明したものです。

- R-1595F-43　uCosminexus DocumentBroker Developer

■対象読者

このマニュアルは，uCosminexus DocumentBroker Developer が提供する DocumentBroker クラスライブラリを使用して，ユーザアプリケーションプログラムを開発し，システムを構築，運用および管理する方を対象にしています。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 に関する知識
- Java™ 言語に関する知識
- SQL 言語に関する知識

■マニュアルの構成

このマニュアルは，次に示す章と付録から構成されています。

第 1 章　DocumentBroker の機能

DocumentBroker クラスライブラリで実現できる機能の概要，システム構成，処理の流れなどについて説明しています。

第 2 章　edmSQL の文法

edmSQL の文法について説明しています。

第 3 章　DocumentBroker のクラス，インターフェース，およびメソッド

DocumentBroker のクラス，インターフェース，およびメソッドを一覧表形式で説明しています。また，DocumentBroker クラスライブラリで提供するクラスライブラリのリファレンスマニュアルのクラス，インターフェース，およびメソッドの記述形式（ページレイアウト）について説明しています。

第 4 章　ファクトリクラス詳細

ファクトリクラスのクラス，インターフェース，およびメソッドについて説明しています。

第 5 章　パラメタクラス詳細

パラメタクラスのインターフェース，およびメソッドについて説明しています。

第 6 章　文書管理クラス詳細

文書管理クラスのインターフェース，およびメソッドについて説明しています。

第 7 章　メタクラス詳細

メタクラスのインターフェース，およびメソッドについて説明しています。

第 8 章　例外クラス詳細

例外クラスの詳細について説明しています。

第 9 章　定数定義クラス詳細

定数定義クラスの定義内容についてカテゴリごとに説明しています。

第 10 章　ライブラリ情報取得クラス詳細

ライブラリ情報取得クラスのクラスとそのメソッドについて説明しています。

第 11 章　トレースクラス詳細

トレースクラスのクラスとそのメソッドについて説明しています。

付録 A　このマニュアルの参考情報

関連マニュアル，このマニュアルで使用している略語の意味などを説明しています。

付録 B　用語解説

DocumentBroker クラスライブラリで使用する用語について説明しています。

■読書手順

このマニュアルは，利用目的に合わせて章を選択してお読みいただけます。次に示す表を参考にして，お読みになる章を選択してください。

このマニュアルを読む目的	記述個所
DocumentBroker クラスライブラリで実現できる機能の概要，システム構成，処理の流れなどについて知りたい	1 章
edmSQL の文法について知りたい	2 章
どのような DocumentBroker クラスライブラリのクラス，インターフェースおよびメソッドがあるかを知りたい	3 章
クラス，インターフェースおよびメソッドの説明形式について知りたい	3.9 節
ファクトリクラスのインターフェースおよびメソッドについて知りたい	4 章
パラメタクラスのインターフェースおよびメソッドについて知りたい	5 章
文書管理クラスのインターフェースおよびメソッドについて知りたい	6 章
メタクラスのインターフェースおよびメソッドについて知りたい	7 章
例外クラスについて知りたい	8 章
定数定義クラスの定義内容について知りたい	9 章
ライブラリ情報取得クラスのクラスおよびメソッドについて知りたい	10 章
トレースクラスのクラスおよびメソッドについて知りたい	11 章
関連マニュアルや略語の意味など，このマニュアルの参考情報について知りたい	付録 A
uCosminexus DocumentBroker の用語を知りたい	付録 B

■このマニュアルで使用する記号

このマニュアルで使用する記号を次に示します。

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」の意味を表します。 （例） A ｜ B A または B を指定することを示します。
＿	括弧で囲まれた複数項目のうち 1 項目に対し使用され，括弧内のすべてを省略した場合にシステムが取る標準値を示します。 （例）〔A ｜ B〕 何も指定しない場合は A が仮定されます。
{ }	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。

記号	意味
〔 〕	この記号で囲まれている項目は省略してもよいことを意味します。複数の項目が横に並べて記述されている場合には，すべてを省略するか，記号｛｝と同じくどれか一つを選択します。 （例１）〔A〕 「何も指定しない」か「A を指定する」ことを示します。 （例２）〔B ｜ C〕 「何も指定しない」か「B または C を指定する」ことを示します。
＜ ＞	この記号で囲まれている項目は，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。
：：＝	この記号の左にあるものを右にあるもので定義することを示します。 （例） A：：＝B 「A とは B である」と定義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。
...	この記号の直前の項目を繰り返し，複数個指定できることを示します。 （例） A... A を複数個指定できることを示します。

■このマニュアルで使用する構文要素記号

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。

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

注　すべて半角文字を使用してください。

目次

1 DocumentBroker の機能

この章では，DocumentBroker クラスライブラリの機能について説明します。また，DocumentBroker クラスライブラリを使用してユーザアプリケーションプログラムを作成する方法について説明します。

1.1 パッケージとクラス

この節では，DocumentBroker が DocumentBroker クラスライブラリとして提供するパッケージとクラスについて説明します。

1.1.1 パッケージ名

DocumentBroker クラスライブラリは，次のパッケージ名で提供されています。

パッケージ名

```
jp.co.Hitachi.soft.docbroker.client
```

1.1.2 クラスの分類

DocumentBroker クラスライブラリのクラスおよびインターフェースは，機能や用途から，次の八つのクラスに分類できます。

- ファクトリクラス
- パラメタクラス
- 文書管理クラス
- メタクラス
- 定数定義クラス
- 例外クラス
- ライブラリ情報取得クラス
- トレースクラス

1.1.3 ファクトリクラス

ファクトリクラスは，次の機能を持つクラスおよびインターフェースの総称です。

- パラメタクラスのオブジェクトを生成して，パラメタクラスのインターフェースを取得する
- 文書管理クラスのセッションオブジェクトを作成して，セッションオブジェクトのインターフェース（DbjSession インターフェース）を取得する
- メタクラスのインターフェース（DbjMetaManager インターフェース）を取得する

ファクトリクラスには，DbjFactory0200 クラスと DbjFactory インターフェースが含まれます。

1.1.4 パラメタクラス

パラメタクラスは，DocumentBroker クラスライブラリ固有のデータをメソッドで受け渡す場合に使用するインターフェースの総称です。

(1) パラメタクラスの機能

パラメタクラスの機能について説明します。

パラメタクラスのインターフェースは，次のような情報を受け渡すために使用します。

- 文書管理オブジェクトのプロパティ情報

- 文書のコンテンツ情報
- 文書のアップロード情報
- リファレンスファイル文書のパス情報
- チェックアウト情報
- レンディション情報
- リンク設定情報
- アクセス制御情報
- 検索結果集合
- 検索結果取得情報
- ? パラメタの情報

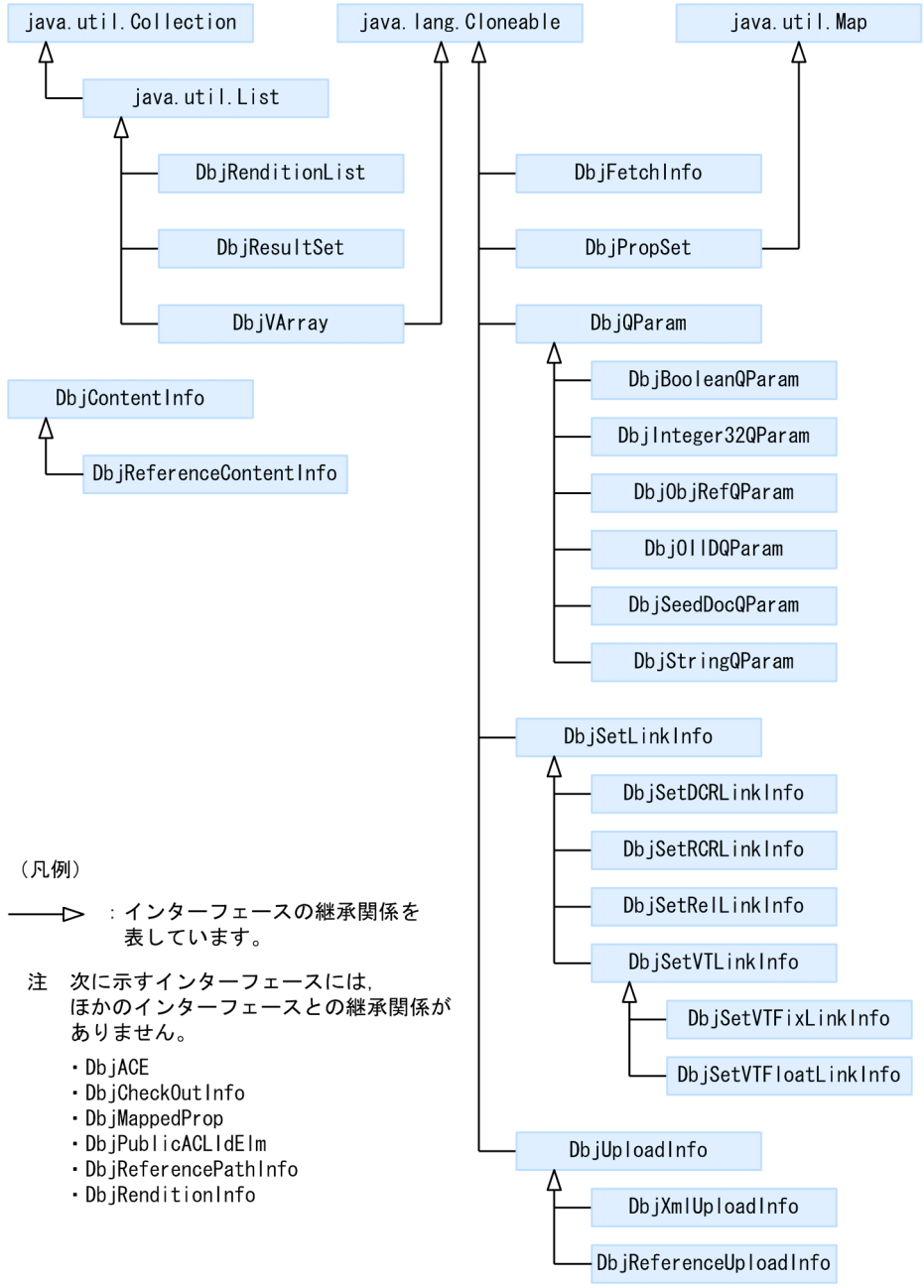
パラメタクラスのインターフェースをデータ型とするこれらの情報は，文書管理クラスのメソッドの引数に指定したり，メソッドの戻り値として返却されたりします。また，これらの情報を参照したり設定したりするメソッドは，パラメタクラスのインターフェースごとに定義されています。

なお，パラメタクラスのインターフェースは，java.util.Collection インターフェース，java.util.List インターフェース，java.util.Map インターフェースおよび java.lang.Cloneable インターフェースなどを継承しています。

(2) インターフェースの継承関係

パラメタクラスのインターフェースの継承関係を，次の図に示します。

図 1-1 パラメタクラスのインターフェースの継承関係



1.1.5 文書管理クラス

文書管理クラスは，DocumentBroker の文書管理に必要な機能を提供するインターフェースの総称です。

(1) 文書管理クラスの機能

文書を管理するための主要な機能は，文書管理クラスのクラスおよびインターフェースとして提供されています。

文書管理クラスのクラスおよびインターフェースでは，例えば，次のような機能を提供しています。

● 文書空間とのセッションを管理する機能

● 文書管理オブジェクトを作成する機能

● 文書管理オブジェクトを検索する機能

● 文書管理オブジェクトを操作する機能

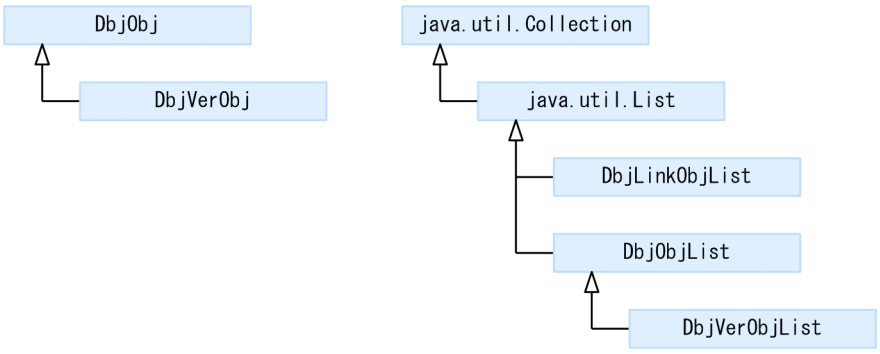
● 複数の文書管理オブジェクトをまとめて操作する機能

このほか，文書管理クラスでは，DocumentBroker の文書管理モデルに基づいた管理を実現するための機能を提供しています。DocumentBroker の文書管理モデルについては，マニュアル「DocumentBroker Version 5　概説」の「文書管理モデル」の説明を参照してください。

(2) インターフェースの継承関係

文書管理クラスのインターフェースの継承関係を，次の図に示します。

図 1-2　文書管理クラスのインターフェースの継承関係



（凡例）

：インターフェースの継承関係を表しています。

注　次に示すインターフェースには，ほかのインターフェースとの継承関係がありません。

- DbjDocSpace
- DbjFactory
- DbjLinkObj
- DbjSession

1.1.6　メタクラス

メタクラスは，文書空間のメタ情報を扱うためのインターフェースの総称です。

(1) メタクラスの機能

メタクラスのインターフェースを使用すると，文書空間のメタ情報を取得できます。

メタ情報とは，DocumentBroker で管理されている，DocumentBroker クラスや文書管理オブジェクトのプロパティに関する情報です。

メタクラスのインターフェースで取得できるのは，次のような情報です。

● 文書空間識別子の情報

● DocumentBroker クラスの名前やスーパークラス，サブクラスに関する情報

● 文書管理オブジェクトのプロパティの名前やデータ型に関する情報

● 拡張子とレンディションタイプの対応に関する情報

(2) インターフェースの継承関係

メタクラスのインターフェースである，DbjMetaManager インターフェース，DbjMeta インターフェース，DbjClassDesc インターフェースおよび DbjPropDesc インターフェースには，継承関係がありません。

1.1.7 定数定義クラス

定数定義クラスとは，DocumentBroker クラスライブラリのメソッドで指定する**定数**が定義されている DbjDef **クラス**と，DbjTraceDef **クラス**のことです。

DocumentBroker クラスライブラリで提供する定数はすべて，DbjDef クラスと DbjTraceDef クラスに，static final で定義されています。

DbjDef クラスでは，次の表に示すカテゴリが定義されています。

表 1-1 定数定義クラスのカテゴリー覧

カテゴリ	説明
DbjDef **クラス**	
ACL	アクセス制御機能付き検索を実行するかどうかを表す定数
CONTENTTYPE	コンテンツ種別を表す定数
DATATYPE	文書管理オブジェクトのプロパティのデータ型を表す定数
DMA BOOLEAN	真偽値を表す定数
INDEXTYPE	全文検索インデクスの作成方法を表す定数
LINK	リンク種別を表す定数
LOCK	ロック種別を表す定数
OBJTYPE	文書管理オブジェクトのオブジェクト種別を表す定数
OPERATEMODE	コンテンツのパス操作モードを表す定数
ORDER	パーミッションの取得順序を表す定数
PERM	パーミッションを表す定数
PRIV	ユーザの特権を表す定数
REFERENCETYPE	リファレンス種別を表す定数
RELATIONEND	文書間リンク設定情報一覧の取得条件を表す定数
RENDSTATUS	レンディションステータスを表す定数
SUBJECTTYPE	サブジェクト種別を表す定数
SYSSUBJECT	システムサブジェクトを表す定数
Others	そのほかの定数
DbjTraceDef **クラス**	
TRACELEVEL	ユーザアプリケーションプログラムのトレース情報のトレースレベルを表す定数
TRACEOUTPUT	ユーザアプリケーションプログラムのトレース情報の出力先を表す定数

定数定義クラスで定義されている定数の詳細については，「9. 定数定義クラスの詳細」を参照してください。

1.1.8 例外クラス

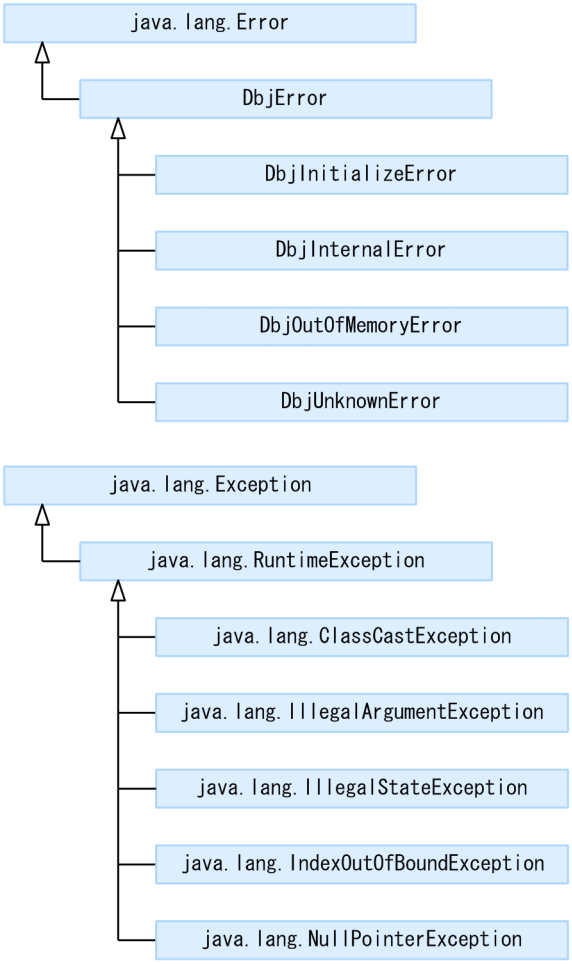
例外クラスは，DocumentBroker クラスライブラリで発生する例外のうち，DocumentBroker クラスライブラリ固有の例外を扱うクラスの総称です。

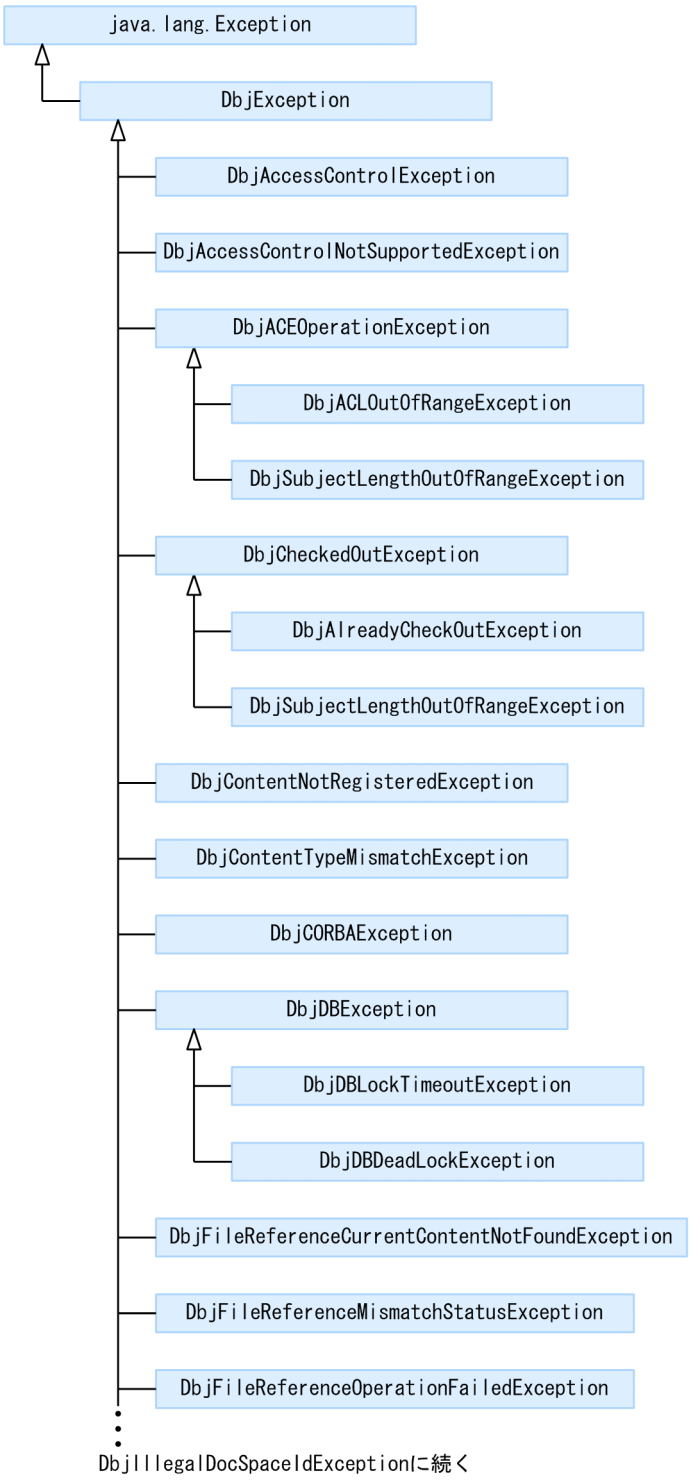
ユーザアプリケーションプログラムの処理中に，DocumentBroker クラスライブラリ固有のエラーが発生すると，例外クラスのオブジェクトがスローされます。

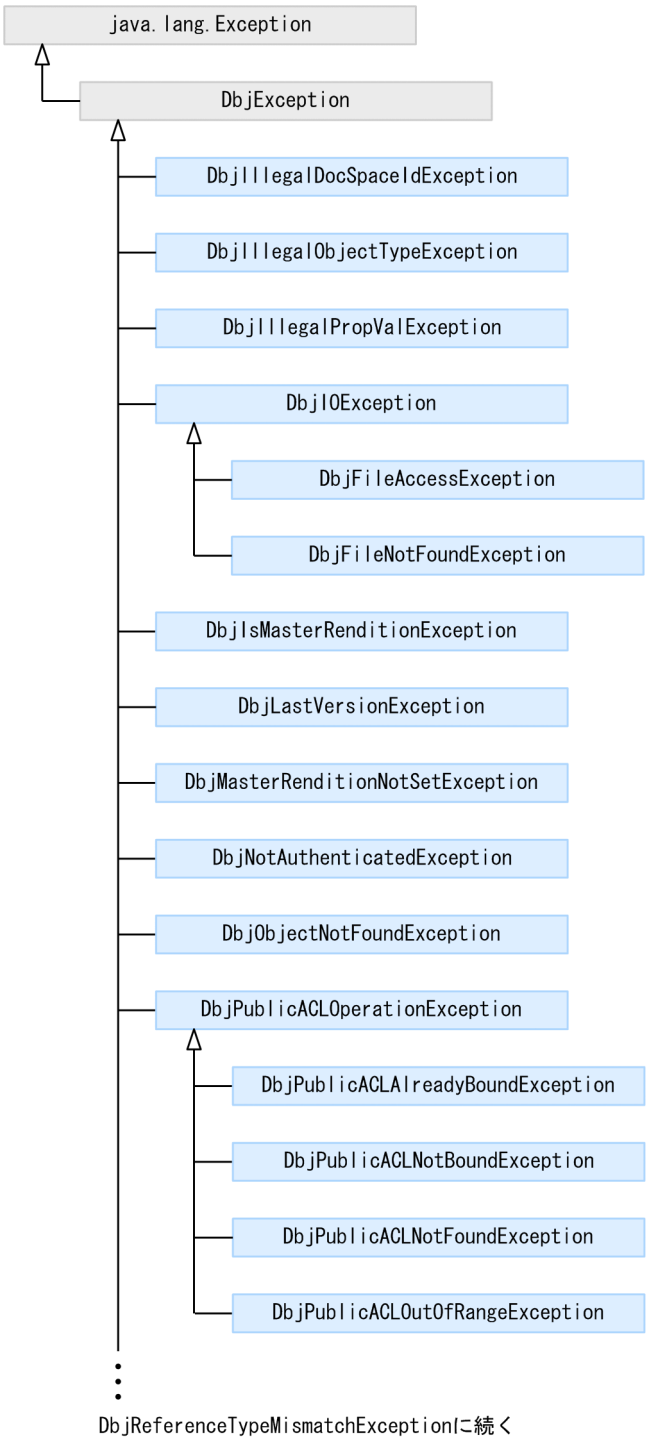
例外クラスの各クラスは，java.lang.Exception クラスまたは java.lang.Error クラスを継承しています。

例外クラスのクラスの継承関係を，次の図に示します。

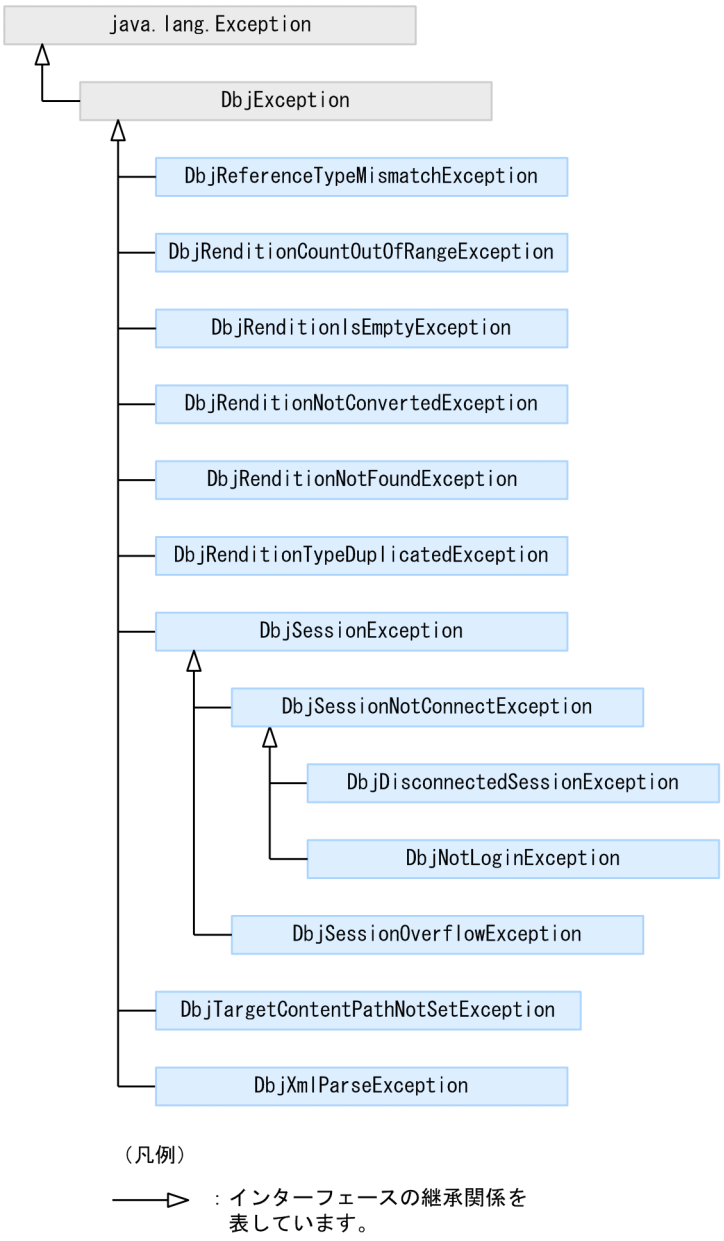
図 1-3 例外クラスのクラスの継承関係




java.lang.Exception
DbjException
DbjAccessControlException
DbjAccessControlNotSupportedException
DbjACEOperationException
DbjACLOutOfRangeException
DbjSubjectLengthOutOfRangeException
DbjCheckedOutException
DbjAlreadyCheckOutException
DbjSubjectLengthOutOfRangeException
DbjContentNotRegisteredException
DbjContentTypeMismatchException
DbjCORBAException
DbjDBException
DbjDBLockTimeoutException
DbjDBDeadLockException
DbjFileReferenceCurrentContentNotFoundException
DbjFileReferenceMismatchStatusException
DbjFileReferenceOperationFailedException
DbjIllegalDocSpaceIdExceptionに続く



DbjReferenceTypeMismatchExceptionに続く



1.1.9 ライブラリ情報取得クラス

ライブラリ情報取得クラスとは，DbjLibInfo **クラス**のことです。

DocumentBroker のバージョン情報を返却するメソッドを提供しています。

1.1.10 トレースクラス

トレースクラスとは，DbjTrace **クラス**のことです。

トレースクラスでは，DocumentBroker クラスライブラリを使用してユーザアプリケーションプログラムを開発・運用する上で必要なトレース情報を出力するメソッドを提供しています。

1.2 DocumentBroker で扱うデータ

この節では，DocumentBroker クラスライブラリで扱うデータについて説明します。

1.2.1 Java の基本データ型および Java が提供するインターフェース

Java の基本データ型および Java が提供するインターフェースで表すデータ型のうち，DocumentBroker クラスライブラリで使用するのは次の表に示すデータ型です。

表 1-2 DocumentBroker クラスライブラリで使用する Java の基本データ型およびインターフェース

データ型	意味	データの種類
boolean	真偽値	Java の基本データ型で表すデータ
int	整数値	
Integer（java.lang.Integer）	整数値	Java が提供するクラスで表すデータ
String（java.lang.String）	文字列	
Object（java.lang.Object）	オブジェクト	
Collection（java.util.Collection）	コレクション※1	Java が提供するインターフェースで表すデータ
Comparator（java.util.Comparator）	コンパレータ※2	
List（java.util.List）	リスト※3	
Map（java.util.Map）	マップ※4	

注※1 要素オブジェクトの集合を表すデータです。
注※2 オブジェクトのコレクション全体の順序付けをする比較関数です。
注※3 順序付けられた要素オブジェクトの集合を表すデータです。
注※4 キー値と対応付けられた要素オブジェクトの集合を表すデータです。

1.2.2 プロパティのデータ型と DocumentBroker で扱うデータ型の対応

DocumentBroker で管理するプロパティには，次のデータ型があります。

- BOOL 型
- INT 型
- VARRAY 型
- STR 型
- STRLIST 型

これらのデータ型を持つプロパティの値を DocumentBroker クラスライブラリのメソッドを使用して設定したり参照したりする場合の，プロパティのデータ型と Java で扱うデータ型との対応を，次の表に示します。

表 1-3 プロパティのデータ型と Java で扱うデータ型の対応

プロパティのデータ型（定数）	Java で扱うデータ型
BOOL 型 （DbjDef.DATATYPE_BOOL）	int 型 Integer 型※ 1
INT 型 （DbjDef.DATATYPE_INT）	int 型 Integer 型※ 1
VARRAY 型 （DbjDef.DATATYPE_VARRAY）	DbjVArray 型
STR 型 （DbjDef.DATATYPE_STR）	String 型
STRLIST 型※ 2 （DbjDef.DATATYPE_STRLIST）	List 型（要素は String 型）

注※ 1 戻り値などで，Integer 型の null 値を表す場合に使用します。
注※ 2 dbrProp_GroupList プロパティを取得する場合に使用します。

DocumentBroker クラスライブラリで扱うプロパティのデータ型と，DocumentBroker で扱うデータ型の対応については，「3.9.3. 文書管理オブジェクトのプロパティのデータ型」を参照してください。なお，表 1-2 で示した以外のデータ型を持つプロパティのデータ型は，UNKNOWN 型（定数：DbjDef.DATATYPE_UNKNOWN）になります。

1.2.3 定数

DocumentBroker クラスライブラリのメソッドで使用する定数は，定数定義クラスに定義されています。

定数の詳細については，「9. 定数定義クラス詳細」を参照してください。

1.3 ユーザアプリケーションプログラムの作成

Windwos の場合，DocumentBroker クラスライブラリが提供する API を使用してユーザアプリケーションプログラムを開発できます。

1.3.1 開発環境

DocumentBroker クラスライブラリが提供する API を使用してユーザアプリケーションプログラムを開発するためには，JDK 6.0 が必要です。

1.3.2 プログラムのコンパイル

(1) Java コンパイラのオプション

DocumentBroker クラスライブラリでは，JDK 6.0 の機能を使用しています。このため，ユーザアプリケーションプログラムは，JDK 6.0 機能を利用したソースとしてコンパイルする必要があります。javac コマンドの -source オプションには 1.6 または 6 を指定してください。

(2) クラスパス

ユーザアプリケーションプログラムをコンパイルする場合は，次に示す JAR ファイルをクラスパスに設定します。

< インストールディレクトリ >¥DocBroker¥Developer¥lib¥djlib5.jar

注意事項

各アプリケーションの war に，次に示す JAR ファイルを組み込んでください。
< インストールディレクトリ >¥DocBroker¥Developer¥lib¥djlib5_j2ee.jar

1.4 プログラムの流れ

この節では，DocumentBroker クラスライブラリを使用して作成するプログラムの流れについて説明します。

1.4.1 インターフェースの取得

DocumentBroker クラスライブラリのメソッドは，クラスまたはインターフェースごとに定義されています。インターフェースで定義されているメソッドを実行するためには，まず，そのインターフェースを取得する必要があります。インターフェースは，そのインターフェースを戻り値として返却するメソッドを実行して取得します。

DocumentBroker クラスライブラリで提供するインターフェースは，DbjFactory0200 **クラス**を起点として取得していきます。このクラスは，DbjFactory インターフェース，または DbjMetaManager インターフェースを戻り値として返却するメソッドを提供しています。パラメタクラス，または文書管理クラスのインターフェースを取得する場合は，まず，DbjFactory インターフェースを取得して，このインターフェースから必要なインターフェースを順番に取得してください。メタクラスのインターフェースを取得する場合は，まず，DbjMetaManager インターフェースを取得して，このインターフェースから必要なインターフェースを順番に取得してください。

1.4.2 文書管理オブジェクトを操作する場合のインターフェースの取得の流れ

DocumentBroker クラスライブラリでは，文書管理オブジェクトを操作することで，DocumentBroker の文書管理機能を利用します。

文書管理オブジェクトを操作する場合は，まず，DbjFactory0200#getFactory メソッドを実行して，DbjFactory インターフェースを取得します。

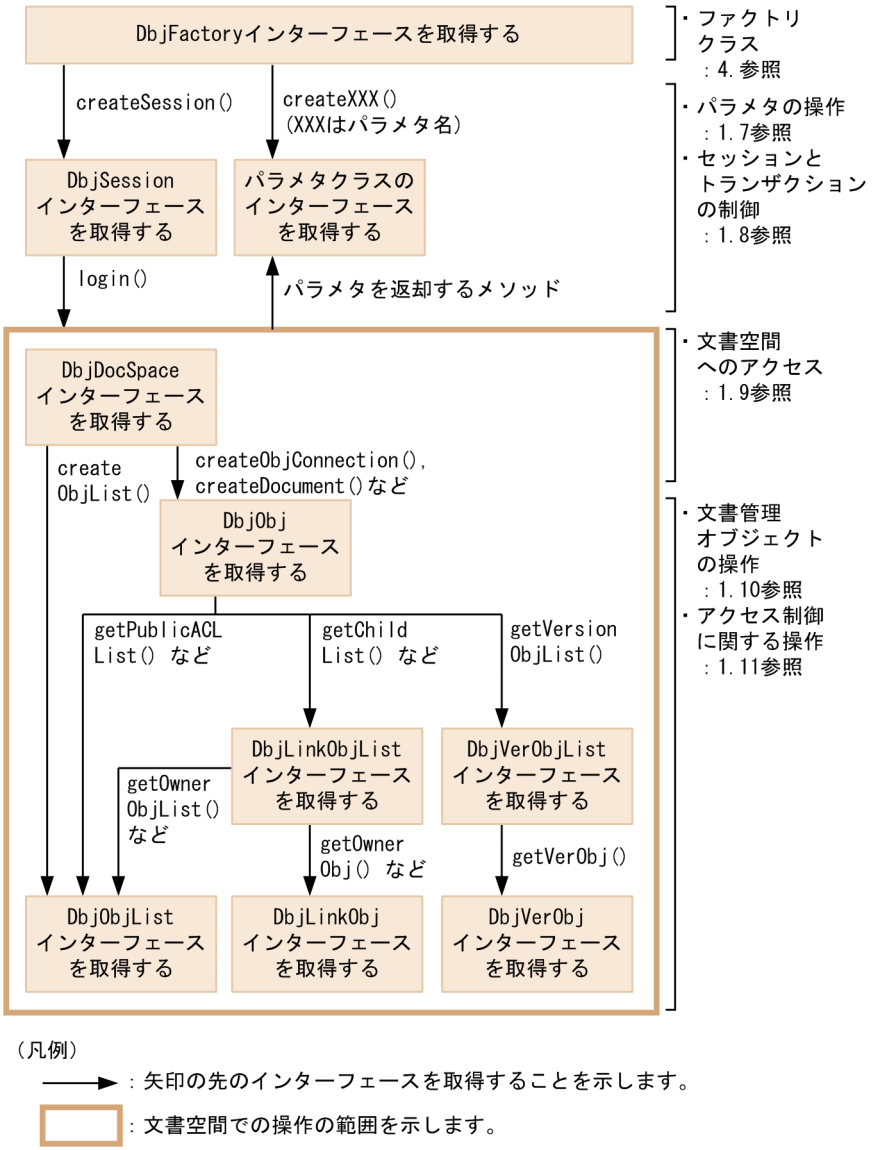
DbjFactory インターフェースを取得したあとの，文書管理機能を使用するユーザアプリケーションプログラムの処理手順は，次のようになります。

1. DbjFactory インターフェースを使用して，DbjSession インターフェースを取得します。
2. DbjSession インターフェースを使用して，セッションを確立します。同時に，ユーザ認証も実行します。ユーザ認証が成功してセッションが確立できると，DbjDocSpace インターフェースを取得できます。
3. DbjDocSpace インターフェースを使用して，文書空間にアクセスします。必要なインターフェースを取得しながら，文書管理機能を実行します。
4. 文書空間での操作が終了したら，DbjSession インターフェースを使用してセッションを切断します。

このほか，必要に応じてパラメタクラスのインターフェースを取得して使用したり，個々の文書管理オブジェクトを操作するインターフェースを取得したりします。

文書管理クラスとパラメタクラスのインターフェースを使用して，文書管理オブジェクトを操作する場合の，インターフェースの取得手順を，次の図に示します。

図 1-4　文書管理オブジェクトを操作する場合に使用するインターフェースの取得手順



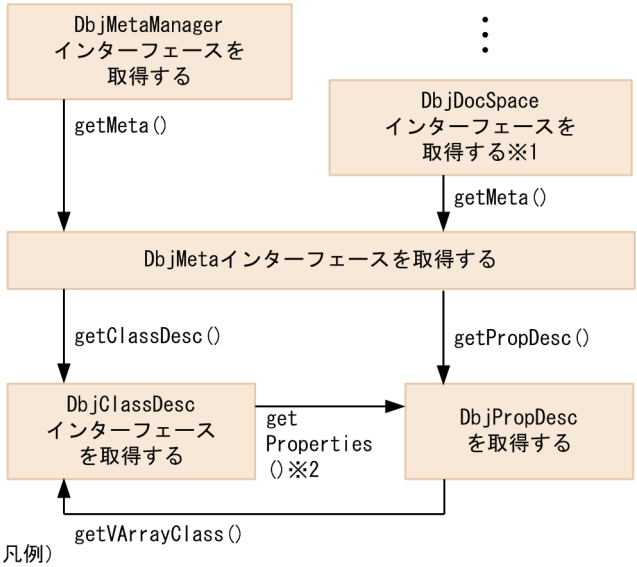
1.4.3　メタ情報を取得する場合のインターフェースの取得の流れ

メタ情報を取得する場合は，まず，次のどちらかのメソッドを実行して，メタクラスのインターフェースを取得します。

- DbjFactory0200#getMetaManager メソッドを実行して，DbjMetaManager インターフェースを取得する
- DbjDocSpace#getMeta メソッドを実行して，DbjMeta インターフェースを取得する

メタ情報を取得する場合に使用するインターフェースの取得手順を，次の図に示します。

図 1-5　メタ情報を取得する場合のインターフェースの取得手順



（凡例）

→ ：矢印の先のインターフェースを取得することを示します。

注※1　DbjDocSpaceインターフェースの取得方法は，図1-6を参照してください。

注※2　DbjPropDescインターフェースを要素としたリストが取得できます。

1.5 プログラミング時の注意と設定

この節では，DocumentBroker クラスライブラリを使用したアプリケーションを開発・実行するときの注意事項と設定について説明します。

1.5.1 使用できる文字コード種別

DocumentBroker クラスライブラリを使用してアプリケーションを開発したり，開発したクライアントアプリケーションを使用して文書管理オブジェクトを操作したりする場合，DocumentBroker クラスライブラリを使用したプログラミング時には，次の文字コード種別を使用できます。

- Shift-JIS
- UTF-8（使用できる文字コードの範囲は UCS-2 または UCS-4 です）

DocumentBroker クラスライブラリを使用したアプリケーションで使用する文字コード種別は，メタ情報初期化ファイルの DocSpaceCharacterSet プロパティの内容に合わせる必要があります。ただし，文字コード種別が Shift-JIS である文書空間に対して UTF-8 の文字コードを使用するときには，Shift-JIS の文字コードにエンコードできる範囲で UTF-8 の文字コードを指定してください。

メタ情報初期化ファイルについては，マニュアル「DocumentBroker Version 5　システム導入・運用ガイド」を参照してください。

1.6 ファクトリクラス

この節では，ファクトリクラスについて説明します。

ファクトリクラスは，パラメタクラスのオブジェクトの生成，セッションオブジェクトの生成，および文書空間メタ情報アクセスインターフェースの取得を実行する，クラスならびにインターフェース群です。

ファクトリクラスには，次のクラスおよびインターフェースが含まれます。

● DbjFactory0200 クラス

● DbjFactory インターフェース

ファクトリクラスの詳細については「4. ファクトリクラス詳細」を参照してください。

1.6.1 DbjFactory0200 クラスの機能

DocumentBroker クラスライブラリの機能を使用するための起点となるクラスです。次のインターフェースを取得するためのメソッドを提供しています。

● DbjFactory インターフェースを取得するメソッド
セッションオブジェクトおよびパラメタクラスのオブジェクトを作成する場合に，このインターフェースを取得します。

● DbjMetaManager インターフェースを取得するメソッド
文書空間のメタ情報にアクセスする場合に，このインターフェースを取得します。

1.6.2 DbjFactory インターフェースの機能

DbjFactory インターフェースでは，パラメタクラスのオブジェクト，セッションオブジェクトおよび XML トランスレーターオブジェクトを作成して，そのオブジェクトを扱うためのインターフェースを返却します。

DbjFactory インターフェースは，DbjFactory0200#getFactory メソッドを実行して取得します。
DbjFactory インターフェースのメソッドで取得できるのは，次のインターフェースです。

● パラメタクラスのインターフェース群

● 文書管理クラスの DbjSession インターフェース

(1) パラメタクラスのインターフェース群

パラメタクラスのインターフェース群は，DocumentBroker クラスライブラリ固有のデータを，メソッドの引数として設定したり，戻り値として取得したりする場合に使用するインターフェース群です。

パラメタクラスのインターフェースは，作成する情報ごとに提供されているメソッドを実行して取得します。例えば，文書管理オブジェクトのプロパティ情報を作成したい場合には DbjFactory#createPropSet メソッドを実行して DbjPropSet インターフェースを取得します。文書のアップロード情報を作成したい場合には DbjFactory#createUploadInfo メソッドを実行して DbjUploadInfo インターフェースを取得します。

パラメタクラスのインターフェースと作成できる情報の種類については，「1.7 パラメタの操作」を参照してください。

ここでは，パラメタクラスのインターフェースのうち，DbjPropSet インターフェースを取得する例を示

します。

// パラメタクラスのインターフェースを取得する例

```
// DbjFactoryインターフェースを取得する
DbjFactory factory = DbjFactory0200.getFactory();

// プロパティ値集合オブジェクトを作成して,
// DbjPropSetインターフェースを取得する
DbjPropSet props = factory.createPropSet();
```

(2) 文書管理クラスの DbjSession インターフェース

DbjSession インターフェースは，文書空間とのセッションを管理する機能を持つインターフェースです。文書管理オブジェクトを操作するためには，DbjSession インターフェースを取得して，文書空間とのセッションを確立する必要があります。

セッションの詳細については,「1.8　セッションとトランザクションの制御」を参照してください。

DbjSession インターフェースを取得する例を示します。

// DbjSessionインターフェースを取得する例

```
// DbjFactoryインターフェースを取得する
DbjFactory factory = DbjFactory0200.getFactory();

//DbjSessionインターフェースを取得する（引数は文書空間識別子（GUID））
DbjSession sess = factory.createSession( docspaceid );
```

文書空間識別子は，デフォルトの値を使用することもできます。デフォルトの値の設定方法については,マニュアル「DocumentBroker Version 5　システム導入・運用ガイド」の「動作環境定義ファイル」の説明を参照してください。

1.7 パラメタの操作

この節では，パラメタクラスのインターフェースについて説明します。

1.7.1 パラメタクラスのインターフェースの機能

DocumentBroker クラスライブラリで扱うデータのうち，DocumentBroker クラスライブラリ固有のデータの受け渡しには，パラメタクラスのインターフェースを使用します。

パラメタクラスのインターフェースは，メソッドの引数のデータ型として使用したり，メソッドの戻り値のデータ型として使用したりします。また，引数に設定する情報を操作するためのメソッドや，戻り値に設定された情報を操作するためのメソッドを提供しています。

例えば，文書のコンテンツを更新する場合，更新するコンテンツやレンディションタイプなどは，文書のアップロード情報としてまとめて一つの引数に指定します。文書のアップロード情報は，パラメタクラスの DbjUploadInfo インターフェースで扱うオブジェクトです。また，検索を実行すると，戻り値として検索結果集合が返却されます。検索結果集合は，パラメタクラスの DbjResultSet インターフェースで扱うオブジェクトです。

なお，パラメタクラスのインターフェースで扱うオブジェクトには，プロパティを持っているものがあります。プロパティには，それぞれのオブジェクトが表す情報の内容が設定されます。このプロパティに値を設定したり，設定されている値を参照したりするためには，パラメタクラスのインターフェースのメソッドを使用します。プロパティに値を設定するメソッドを，setter **メソッド**といいます。プロパティの値を取得するメソッドを，getter **メソッド**といいます。

例えば，文書のアップロード情報に，更新するコンテンツやレンディションタイプなどを設定する場合は，DbjUploadInfo インターフェースのメソッドを使用します。また，検索を実行して取得した検索結果集合から，個々の検索結果を参照する場合は，DbjResultSet インターフェースのメソッドを使用します。

このように，パラメタクラスのインターフェースは，文書管理クラスのメソッドの引数に指定する情報を設定したり，戻り値として返却される情報を参照したりする場合に使用します。パラメタクラスの詳細については「5. パラメタクラス詳細」を参照してください。

また，パラメタクラスのインターフェースには，java.util.Collection インターフェース，java.util.List インターフェース，java.util.Map インターフェース，java.lang.Cloneable インターフェースなどを継承しているものがあります。継承関係については，「1.1.4(2) インターフェースの継承関係」を参照してください。

1.7.2 パラメタクラスのインターフェースの種類

ここでは，パラメタクラスのインターフェースの種類について説明します。

パラメタクラスのインターフェースと扱う情報との対応を，次の表に示します。

表 1-4 パラメタクラスのインターフェースと扱う情報

インターフェース	扱う情報
DbjACE	ACE
DbjCheckOutInfo	チェックアウト情報
DbjContentInfo	コンテンツ情報
DbjConvertContentInfo	コンテンツ種別変換

インターフェース	扱う情報
DbjFetchInfo	検索結果取得情報
DbjPropSet	プロパティ値集合
DbjPublicACLIdElm	パブリック ACL の OIID 値
DbjQParam	―
DbjBooleanQParam	BOOL 型の値を表す？パラメタ
DbjInteger32QParam	INT 型の値を表す？パラメタ
DbjObjQParam	オブジェクトリファレンスの値を表す？パラメタ
DbjOIIDQParam	OIID 文字列の値を表す？パラメタ
DbjSeedDocQParam	種文章を表す？パラメタ
DbjStringQParam	STR 型の値を表す？パラメタ
DbjReferenceContentInfo	リファレンスファイル文書のコンテンツ情報
DbjReferencePathInfo	リファレンスファイル文書のパス情報
DbjReferenceUploadInfo	リファレンスファイル文書のアップロード情報
DbjRenditionInfo	レンディション情報
DbjRenditionList	レンディション情報リスト
DbjResultSet	検索結果集合
DbjSetLinkInfo	―
DbjSetDCRLinkInfo	直接型リンク設定情報
DbjSetRelLinkInfo	文書間リンク設定情報
DbjSetRCRLinkInfo	参照型リンク設定情報
DbjUploadInfo	文書のアップロード情報
DbjVArray	可変長配列（VARRAY 型のプロパティの値）

（凡例）

―：ほかのインターフェースのスーパーインターフェースです。

(1) DbjACE インターフェース

ACE の値を扱うためのインターフェースです。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- サブジェクト（subject）
- サブジェクト種別（subjectType）
- パーミッション（permission）
- プロパティ値集合（propSet）

なお，ACE は，文書管理オブジェクトの VARRAY 型のプロパティの要素です。このため，ほかの文書管理オブジェクトのプロパティと同様，DbjPropSet インターフェースを使用して操作することもできます。propSet プロパティは，ACE に対応する DbjPropSet インターフェースが設定されているプロパティです。ACE をプロパティ値集合として DbjVArray インターフェースで扱うオブジェクト（可変長配列）に設定する場合に使用します。

このインターフェースの使用方法については，「1.11.2　ローカル ACL と ACE の操作」を参照してくださ

い。

(2) DbjCheckOutInfo インターフェース

チェックアウト情報を扱うためのインターフェースです。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- チェックアウト状態（checkOut）
- チェックアウトユーザ識別子（checkOutUserId）
- チェックアウトバージョン識別子（checkOutVersionId）

(3) DbjContentInfo インターフェース

文書のコンテンツ情報を扱うためのインターフェースです。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- 文書に登録されているファイル名（retrievalName）
- レンディションタイプ（renditionType）

このインターフェースの使用方法については，「1.10.9　文書のコンテンツの操作」を参照してください。

(4) DbjConvertContentInfo インターフェース

コンテンツ変換機能を使用した文書のコンテンツ種別の変換時に使用する情報を扱うためのインターフェースです。

このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- 変換先のコンテンツ種別（convertType）
- 実行モード（executeMode）
- 対象レンディション範囲（sourceScope）
- マスタレンディション変換フラグ（changeMaster）
- 対象レンディションタイプ（renditionType）
- レンディションステータス（checkRenditionStatus）
- コメントの付与フラグ（investSourceComment）
- コメントの付与方式（investMode）
- レンディションコメント（renditionComment）
- リファレンスファイル情報（referenceTargetPath）

(5) DbjFetchInfo インターフェース

検索結果取得情報を扱うためのインターフェースです。検索結果取得情報は，キャッシュ付き検索実行時に，検索結果の取得方法について指定する情報です。キャッシュ付き検索は，一覧を取得するメソッドでも実行できます。

このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- 検索結果の取得開始位置（startIndex）
- 取得件数（fetchCount）

● 最大取得件数（maxFetchCount）

● キャッシュ名（cacheName）

● キャッシュキー（cacheKey）

● キャッシュの全件数（cacheTotal）

● 検索結果ソート用コンパレータ（comparator）

なお，comparator プロパティは，検索結果をソートするために使用する java.util.Comparator インターフェースです。

このインターフェースの使用方法については，「1.9.4　文書管理オブジェクトの検索」を参照してください。

（6） DbjPropSet インターフェース

プロパティ値集合を扱うインターフェースです。

プロパティ値集合とは，文書管理オブジェクトのプロパティ名とそのプロパティの値のペアを要素とする集合オブジェクトです。次のような特徴があります。

● DbjPropSet インターフェースは，java.util.Map インターフェースを継承しています。プロパティ値集合は，要素のキーを文書管理オブジェクトのプロパティ名として，要素の値をプロパティの値とするマップとして扱うことができます。java.util.Map インターフェースの機能でプロパティ値集合を扱うこともできます。

● マップであるため，プロパティ値集合のキー値であるプロパティ名は重複しません。また，要素間に順序性はありません。

● プロパティ名に null 値は設定できませんが，値に null 値は設定できます。

● プロパティ値集合をコピーする場合は，java.lang.Cloneable インターフェースの機能でコピーします。

このインターフェースの使用方法については，「1.10.7　文書管理オブジェクトのプロパティの操作」を参照してください。

（7） DbjPublicACLIdElm インターフェース

文書，フォルダまたは独立データがバインドしているパブリック ACL を扱うインターフェースです。VARRAY 型のプロパティの要素として扱います。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

● パブリック ACL の OIID（Id）

● プロパティ値集合（propSet）

なお，バインドしているパブリック ACL の OIID は，VARRAY 型のプロパティ（文書管理オブジェクトの dbrProp_ACLIdElem プロパティ）の要素です。このため，ほかの文書管理オブジェクトのプロパティと同様，DbjPropSet インターフェースを使用して操作することもできます。propSet プロパティは，バインドしているパブリック ACL の OIID に対応する DbjPropSet インターフェースが設定されているプロパティです。パブリック ACL の OIID をプロパティ値集合として DbjVArray インターフェースで扱うオブジェクト（可変長配列）に設定する場合に使用します。

このインターフェースの使用方法については，「1.11.3　パブリック ACL の操作」を参照してください。

(8) DbjQParam インターフェース

? パラメタに設定する値を扱うインターフェースのスーパーインターフェースです。このインターフェースを継承して，DocumentBroker で扱うデータ型ごとにサブインターフェースが定義されています。

このインターフェースのサブインターフェースをデータ型として値を設定する場合，edmSQL の定数表現ではなく DocumentBroker クラスライブラリで扱うデータ型に従った値を指定してください。

サブインターフェースの使用方法については，「1.9.4　文書管理オブジェクトの検索」を参照してください。

(a) DbjBooleanQParam インターフェース

BOOL 型の値を ? パラメタに設定するためのインターフェースです。

(b) DbjInteger32QParam インターフェース

INT 型の値を ? パラメタに設定するためのインターフェースです。

(c) DbjObjQParam インターフェース

オブジェクトリファレンスの値を ? パラメタに設定するためのインターフェースです。指定した OIID 文字列は，オブジェクトリファレンスの形式に変換されて ? パラメタに設定されます。

(d) DbjOIIDQParam インターフェース

OIID 文字列を ? パラメタに設定するためのインターフェースです。指定した OIID 文字列は，dmaProp_OIID プロパティの格納形式（16 バイト）に変換されて ? パラメタに設定されます。

(e) DbjSeedDocQParam インターフェース

概念検索で ? パラメタに種文章を設定するためのインターフェースです。

(f) DbjStringQParam インターフェース

STR 型の値を ? パラメタに設定するためのインターフェースです。

(9) DbjReferenceContentInfo インターフェース

リファレンスファイル文書のコンテンツ情報を扱うためのインターフェースです。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- コンテンツロケーション（contentLocation）
- コンテンツのリファレンス種別（referenceType）

このインターフェースの使用方法については，「1.10.12　リファレンスファイル文書の操作」を参照してください。

(10) DbjReferencePathInfo インターフェース

リファレンスファイル文書のパス情報を扱うためのインターフェースです。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- コンテンツのパス操作モード（contentOperateMode）
- 登録するコンテンツのパスまたはダウンロード先のパス（entry）
- コンテンツ格納先パス（targetPath）

● 削除するディレクトリのルートパス（deleteRootPath）

このインターフェースの使用方法については，「1.10.12　リファレンスファイル文書の操作」を参照してください。

（11）DbjReferenceUploadInfo インターフェース

リファレンスファイル文書のアップロード情報を扱うためのインターフェースです。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

● リファレンスファイル文書のパス情報（referencePathInfo）

このインターフェースの使用方法については，「1.10.12　リファレンスファイル文書の操作」を参照してください。

（12）DbjRenditionInfo インターフェース

レンディション情報を扱うためのインターフェースです。このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

● レンディションタイプ（renditionType）

● レンディションに設定されているプロパティ値集合（propSet）

このインターフェースの使用方法については，「1.10.11　マルチレンディション文書のレンディションの操作」を参照してください。

（13）DbjRenditionList インターフェース

レンディション情報のリストを扱うためのインターフェースです。リストの要素は，DbjRenditionInfo インターフェースです。

このインターフェースの使用方法については，「1.10.11　マルチレンディション文書のレンディションの操作」を参照してください。

（14）DbjResultSet インターフェース

検索結果集合を扱うためのインターフェースです。検索結果集合は，文書管理オブジェクトのプロパティ値を要素に持つ，行と列の二次元データにメタデータを付けたものとして表されます。

検索結果集合は，行を要素とするリストとして扱うことができます。また，各行は，文書管理オブジェクトのプロパティ値を要素とするリストとして扱うことができます。

このインターフェースの使用方法については，「1.9.4　文書管理オブジェクトの検索」を参照してください。

（15）DbjSetLinkInfo インターフェース

リンク設定情報を扱うためのスーパーインターフェースです。このインターフェースを継承して，リンク種別ごとにサブインターフェースが定義されています。

このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

● リンク種別（linkType）

● リンクオブジェクトのプロパティ（propSet）

● リンク対象となるオブジェクト（targetObj）

サブインターフェースの使用方法については，「1.10.13　リンクの操作」を参照してください。

(a) DbjSetDCRLinkInfo インターフェース

直接型リンクを表すリンク設定情報を扱うインターフェースです。

(b) DbjSetRelLinkInfo インターフェース

文書間リンクを表すリンク設定情報を扱うインターフェースです。

(c) DbjSetRCRLinkInfo インターフェース

参照型リンクを表すリンク設定情報を扱うインターフェースです。

(16) DbjUploadInfo インターフェース

文書のアップロード情報を扱うインターフェースです。

このインターフェースで扱うオブジェクトは，次に示すプロパティを持っています。

- コンテンツとして登録するファイルのファイル名を含んだローカルパス（filePath）
- コンテンツとして登録するファイルのファイル名（retrievalName）
- コンテンツとして登録するファイルのレンディションタイプ（renditionType）
- レンディションのプロパティ値集合（propSet）
- 全文検索インデクスの作成に使用するファイルパス名（indexPath）

このインターフェースの使用方法については，「1.10.9　文書のコンテンツの操作」を参照してください。

(17) DbjVArray インターフェース

VARRAY 型のプロパティの値である可変長配列を扱うためのインターフェースです。

可変長配列は，DbjVArray インターフェースを使用して，追加，削除などの更新処理を実行します。

このインターフェースは，java.util.List インターフェースを継承しているので，可変長配列の要素はリストの要素としても扱うこともできます。ただし，DbjVArray インターフェースで可変長配列の要素を扱うことによって，各要素がメタプロパティとして定義されているプロパティを持つことが保証されます。

このインターフェースの使用方法については，「1.10.7　文書管理オブジェクトのプロパティの操作」を参照してください。

1.8 セッションとトランザクションの制御

この節では，セッションとトランザクションの制御について説明します。セッションとトランザクションは，DbjSession インターフェースの機能で制御します。

1.8.1 セッションとトランザクション

ここでは，セッションとトランザクションについて説明します。

(1) セッション

セッションとは，文書空間に接続している間のことです。文書空間に接続することを，**セッションの確立**といいます。文書空間への接続とは，DocumentBroker と接続して，文書空間にログインすることです。文書管理オブジェクトを操作するためには，まず，セッションを確立する必要があります。

文書管理オブジェクトの操作が完了したら，セッションを切断します。**セッションの切断**とは，DocumentBroker との接続を切断して，文書空間からログアウトすることです。

セッションは，セッションオブジェクトによって管理します。**セッションオブジェクト**は，セッションを確立および切断する機能とセッション内のトランザクションを制御する機能を持つ DocumentBroker クラスライブラリのオブジェクトです。DbjFactory#createSession メソッド実行時に，メモリ空間上に作成されます。DbjSession インターフェースの機能を実装したオブジェクトです。

(2) トランザクション

セッション内での文書空間へのアクセスはトランザクション単位に分割されます。**トランザクション**とは，文書管理オブジェクト操作の処理単位です。トランザクション単位で，文書管理オブジェクトに対する操作を確定または取り消すことができます。

トランザクションの範囲は，DocumentBroker クラスライブラリのメソッド単位か，またはユーザアプリケーションプログラムが明示的に指定した開始と終了の間の範囲になります。ユーザアプリケーションプログラムで明示的なトランザクションの範囲を指定しない場合は，メソッド単位でトランザクションが分割されます。トランザクションの範囲を明示的に指定するかメソッド単位で分割するかは，ユーザアプリケーションプログラムで選択できます。

ただし，トランザクションは，一つのセッション内だけで有効です。複数のセッションにわたったトランザクションは指定できません。また，一つのセッション内で，明示的に範囲を指定できるトランザクションは一つだけです。一つのトランザクションが終了しないうちに，別のトランザクションを明示的に開始することはできません。

1.8.2 DbjSession インターフェースの機能

ここでは，DbjSession インターフェースの機能について説明します。DbjSession インターフェースは，DbjFactory#createSession メソッドを実行して取得します。

DbjSession インターフェースには，次の機能があります。

- セッション管理
- トランザクション制御
- ログイン情報の取得
- 文書空間にアクセスするインターフェースの取得

● コンテンツ格納先ベースパスの取得と設定

1.8.3 セッション管理

ここでは，DbjSession インターフェースの，セッション管理機能について説明します。

(1) セッションの確立（ログイン）

セッションは，DbjSession#login メソッドによって確立します。一つの DbjSession インターフェースで確立できるセッションは一つです。

セッションの確立では，ユーザ識別子とパスワードによるユーザ認証も実行されます。認証に失敗した場合，セッションは確立されないで，例外がスローされます。

DbjSession#login メソッドの実行に成功すると，DbjDocSpace インターフェースが返却されます。

DbjSession#login メソッドを実行して，セッションを確立する処理の例を示します。

// セッションを確立する処理の例

```
DbjSession sess = null;
DbjDocSpace docspc = null;
try {
  // DbjSessionインターフェースを取得する
  sess = DbjFactory0200.getFactory().createSession(docspaceid);
                  // 接続先の文書空間識別子(GUID文字列)を指定する

  // セッションを確立する(ログイン)
  docspc = sess.login( "suzuki", "passwd" );

} catch(DbjNotAuthenticatedException e) {
  // 認証エラーの場合
  System.out.println("Not authenticated");

} catch(DbjException e) {
  // そのほかのエラーの場合

}
```

(2) セッションの切断（ログアウト）

セッションは，DbjSession#logout メソッドによって切断します。セッションの切断は，確立時と同じセッションオブジェクトのインターフェースで実行します。

また，明示的に開始したトランザクションが終了していない場合など，セッションを切断時にセッション内に未確定のトランザクションがある場合は，そのトランザクションは自動的にロールバック処理されます。

(3) セッションチェック

ここでは，セッションチェックについて説明します。

セッションがすでに切断されているセッションオブジェクトを使用して文書空間にアクセスしようとすると，例外がスローされます。

DbjSession インターフェースでは，文書空間にアクセスする前に，セッションが有効であるかどうかチェックする機能を提供しています。これを**セッションチェック**機能といいます。

セッションオブジェクトを使用する前に，セッションチェックを実行することによって，切断されたセッションに対してアクセスすることを防げます。また，セッションが切断されていた場合は，必要に応じて

再度セッションを確立する処理を実行できます。

セッションチェックの例を示します。

// セッションチェックの例

```
// セッションが切断されていたら，再度セッション確立処理をする
public void authIfNoSession(DbjSession sess)
{
  // セッションチェック処理
  if ( !sess.checkSession() ) {
        // セッションを再度確立する
        ......
  }
}
```

1.8.4 トランザクション制御

ここでは，トランザクション制御について説明します。

(1) トランザクションの開始と終了

トランザクションは，ユーザアプリケーションプログラムで明示的に開始と終了を指定して制御できます。明示的に制御しない場合は，トランザクションはメソッドごとに開始・終了されます。

トランザクションは，DbjSession#begin メソッドによって開始します。明示的に開始したトランザクションは，DbjSession#commit メソッドまたは DbjSession#rollback メソッドによって終了させます。DbjSession#commit メソッドを実行した場合は，トランザクション内の処理が確定されます（**コミット**）。DbjSession#rollback メソッドを実行した場合は，トランザクション内の処理が取り消されます（**ロールバック**）。

明示的に終了しない場合でも，DbjSession#logout メソッドによって文書空間との接続が切断されたときには，ロールバック処理が実行されて，トランザクションは自動的に終了します。また，一部の文書空間にアクセスするメソッドが失敗した場合も，強制的にロールバック処理が実行されて，トランザクションが終了することがあります。

(2) トランザクションの範囲

トランザクション範囲を指定する処理の例を次に示します。

なお，この例は，トランザクションの範囲について理解するための例ですので，処理自体に意味はありません。

// トランザクション範囲を指定する例

```
  // sess：DbjSessionインターフェース

// セッションを確立する（ログイン）
DbjDocSpace docspc = sess.login( username, passwd );
try {
  // DbjDocSpaceインターフェースからDbjObjインターフェースを取得する
  DbjObj obj = docspc.createObjConnection( oiid );

  // Nameプロパティを取得する準備をする
  Set<String> propdef = new HashSet<String>();
  propdef.add( "Name" );

  // メソッド単位のトランザクションの例
  // 文書管理オブジェクトからNameプロパティを取得
  // メソッドの実行によって，暗黙的にトランザクションが開始・終了される
```

```
  obj.readProperties( propdef );

  // 明示的にトランザクションを開始・終了する例
  // 明示的にトランザクションを開始する--------------------------
  sess.begin();

  // 文書管理オブジェクトのWRITEロックを取得して
  // Nameプロパティを取得する
  // トランザクションはメソッド終了後も継続される
  obj.lock(DbjDef.LOCK_WRITE).readProperties();

  // 文書管理オブジェクトにNameプロパティを更新する
  // トランザクションはメソッド終了後も継続される
  obj.writeProperties();

  // 明示的にトランザクションを終了する--------------------------
  sess.commit();

} catch(Exception e) {

  // 処理中にエラーが発生した場合は,
  // トランザクション中の処理を取り消して
  // トランザクションを終了する-----------------------------------
  sess.rollback();
}

// セッションを切断する（ログアウト）
sess.logout();
```

なお，DbjSession#begin メソッドによって明示的にトランザクションを開始した場合は，DbjSession#commit メソッドまたは DbjSession#rollback メソッドによって明示的にトランザクションを終了するまで，再度 DbjSession#begin メソッドを実行することはできません。一つのセッション内で明示的に開始できるトランザクションは一つだけです。

1.8.5 ログインユーザ情報の取得

ここでは，ログイン情報の取得について説明します。

アクセス制御機能に対応した文書空間に対してセッションを確立している場合，ログインしたユーザのユーザ情報を取得することができます。ユーザ情報の取得は，DbjSession#getLoginUserInfo メソッドによって実行します。ユーザ情報として取得したい情報を表すプロパティ名の集合を指定して，必要なユーザ情報が取得できます。

ユーザ情報として取得できるプロパティは，次のとおりです。

- ユーザ識別子（dbrProp_UserId）
- 所属するグループの数（dbrProp_GroupCount）
- 所属するグループのグループ識別子の一覧（dbrProp_GroupList）
- 特権（dbrProp_UserPrivilege）
- ユーザ権限（dbrProp_UserPermission）

なお，DbjSession#getLoginUserInfo メソッドは，文書空間にアクセスするメソッドではありません。このため，このメソッドがトランザクションに影響することはありません。

また，セッションを確立している文書空間がアクセス制御機能に対応しているかどうかは，DbjDocSpace#isAccessControlMode メソッドによって確認できます。

ログインユーザ情報を取得する例を示します。

// ログインユーザ情報を取得する例

```
  // sess：DbjSessionインターフェース

// セッションを確立する
DbjDocSpace docspc = sess.login( username, passwd );

if ( !docspc.isAccessControlMode() ) {
  // アクセス制御機能に対応していない文書空間の場合
  System.out.println
  ("Sorry, user information cannot be obtained.");
  return;
}

// 取得するプロパティ名の一覧を作成する
Set<String> propdef = new HashSet<String>();
propdef.add("dbrProp_UserId");          // ユーザ識別子
propdef.add("dbrProp_UserPrivilege");   // 特権
propdef.add("dbrProp_GroupList");
                        // 所属するグループのグループ識別子の一覧

// ユーザ情報を取得する
DbjPropSet propSet = sess.getLoginUserInfo(propdef);

// 取得したユーザ情報を表示する
System.out.println("userId = "
+propSet.getStringVal("dbrProp_UserId"));
System.out.println("userPriv = "
+propSet.getIntVal("dbrProp_UserPrivilege"));
System.out.println("groupList = "
+propSet.getListRef("dbrProp_GroupList"));

// セッションを切断する
sess.logout();
```

1.9 文書空間へのアクセス

この節では，文書空間へのアクセスについて説明します。文書空間には，DbjDocSpace インターフェースの機能でアクセスします。

1.9.1 文書空間アクセスオブジェクト

ここでは，文書空間アクセスオブジェクトについて説明します。

DbjSession#login メソッドの実行が成功すると，戻り値として DbjDocSpace インターフェースが返却されます。DbjDocSpace インターフェースは，接続した文書空間を概念的なオブジェクト（**文書空間アクセスオブジェクト**）として扱うためのインターフェースです。

このオブジェクトは，セッションを切断するまで有効です。

1.9.2 DbjDocSpace インターフェースの機能

ここでは，DbjDocSpace インターフェースの機能について説明します。DbjDocSpace インターフェースは，DbjSession#login メソッドを実行して取得します。

DbjDocSpace インターフェースには，次の機能があります。

- 文書管理オブジェクトの作成
- 文書管理オブジェクトの検索
- 文書管理オブジェクトにアクセスするためのインターフェースの取得

なお，文書管理オブジェクトを検索して，検索条件に合致する文書管理オブジェクトをすべて削除する機能もあります。

DbjDocSpace インターフェースは，特定の文書管理オブジェクトに依存しない，文書空間全体を対象にする機能を提供しています。また，特定の文書管理オブジェクトにアクセスするためのインターフェースを取得するための機能も提供しています。

1.9.3 文書管理オブジェクトの作成

ここでは，文書管理オブジェクトの作成について説明します。

(1) 文書管理オブジェクトの作成に使用するメソッド

DbjDocSpace インターフェースでは，作成する文書管理オブジェクトの種別ごとにメソッドが定義されています。

作成できる文書管理オブジェクトの種別と，作成に使用するメソッドについて，次の表に示します。

表 1-5 文書管理オブジェクトの種別と作成に使用するメソッド

文書管理オブジェクト	メソッド
バージョンなし文書	createDocument
バージョン付き文書	createVrDocument
バージョンなしフォルダ	createFolder
独立データ	createIndependentData

文書管理オブジェクト	メソッド
パブリック ACL	createPublicACL

なお，文書管理オブジェクトの作成が成功した場合，作成した文書管理オブジェクトを操作するための DbjObj インターフェースがメソッドの戻り値として返却されます。DbjObj インターフェースの詳細については，「1.10　文書管理オブジェクトの操作」を参照してください。

また，文書管理オブジェクト作成時には，次の情報を指定できます。

● 文書管理オブジェクトのトップオブジェクトクラス

● 文書管理オブジェクトのプロパティの初期値

● 文書のアップロード情報（文書の場合）

● 関連付ける文書またはフォルダ（文書またはフォルダの場合）

● バインドする文書，フォルダまたは独立データ（パブリック ACL の場合）

それぞれの情報について説明します。

文書管理オブジェクトのトップオブジェクトクラス

文書管理オブジェクトのトップオブジェクトを作成する基になる DocumentBroker クラス（トップオブジェクトクラス）を指定できます。各文書管理オブジェクトで指定できるトップオブジェクトクラスについては，マニュアル「DocumentBroker Version 5　概説」の「文書管理オブジェクトクラスと DocumentBroker クラスの対応」の説明を参照してください。

DocumentBroker では，DocumentBroker クラスに必要なユーザ定義プロパティを追加定義したサブクラスが作成できます。例えば，ユーザ定義プロパティを追加した usrClass_DocVersion クラス（edmClass_VersionTracedDocVersion クラスのサブクラス）を作成していた場合，この DocumentBroker クラスを文書管理オブジェクトのトップオブジェクトクラスとして指定できます。ユーザ定義プロパティの追加方法については，マニュアル「DocumentBroker Version 5　システム導入・運用ガイド」を参照してください。

文書管理オブジェクトのプロパティの初期値

プロパティの初期値を指定できます。例えば，「文書のタイトル」，「作成者」，「作成日時」などをユーザ定義プロパティとして定義している場合に，それらの値を文書管理オブジェクトの作成と同時に設定できます。

文書のアップロード情報（文書の場合）

バージョンなし文書またはバージョン付き文書を作成する場合は，文書のコンテンツとして登録するファイルやレンディションタイプなどを設定した，文書のアップロード情報を指定できます。複数の文書のアップロード情報を指定すれば，マルチレンディション文書になります。

関連付ける文書またはフォルダ（文書またはフォルダの場合）

作成する文書またはフォルダを，既存の文書管理オブジェクトに関連付けることができます。例えば，既存のフォルダで管理する文書として登録したり，既存のフォルダの下位フォルダとして登録したりできます。

また，文書を作成する場合，文書間リンクで既存の文書と関連付けることもできます。ただし，この場合，作成した文書がリンク元文書，既存の文書がリンク先文書になりますので，注意してください。

作成する文書管理オブジェクトのオブジェクト種別ごとの，設定できるリンクの種別は，次のとおりです。

バージョンなし文書を作成する場合

- フォルダとの直接型リンクおよび参照型リンク

- 文書との文書間リンク

バージョン付き文書を作成する場合

- フォルダとの直接型リンク，参照型リンク
- 文書との文書間リンク

バージョンなしフォルダを作成する場合

- フォルダとの直接型リンクおよび参照型リンク

バインドする文書，フォルダまたは独立データ（パブリック ACL の場合）

作成するパブリック ACL をバインドする文書管理オブジェクトのリストを指定できます。

ここでは，バージョン付き文書を作成する例を示します。なお，バージョン付き文書にプロパティを指定する場合の指定方法については，「1.10.7(3) バージョン付きオブジェクトのプロパティの操作」を参照してください。

// バージョン付き文書を作成する例

```
  // factory ： DbjFactoryインターフェース
  // docspc  ： DbjDocSpaceインターフェース

//初期値として設定するプロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();

// バージョニングオブジェクトのプロパティ(Author)を設定する
props.setPropVal( "Author", "suzuki" );

// カレントバージョンのプロパティ(Ver)を設定する
// （バージョンオブジェクトのプロパティは@を付けて指定する）
props.setPropVal( "@Ver", "01-00" );

//文書のアップロード情報のリストを作成する
List<DbjUploadInfo> uploadlist = new ArrayList<DbjUploadInfo>();
uploadlist.add( factory.createUploadInfo(
                file,
                retrievalName,
                null,      // レンディションタイプは自動で設定する
                null,      // レンディションプロパティは設定しない
                null ) ); // 全文検索インデクスは作成しない

// リンク設定情報のリストを作成する（直接型リンクで関連付ける）
List<DbjSetLinkInfo> linklist = new ArrayList<DbjSetLinkInfo>();
linklist.add( factory.createSetDCRLinkInfo(
                docspc.createObjConnection(parentoiid) ,
                          // 上位フォルダのOIID
                null ) ); // リンクプロパティは指定しない

// バージョン付き文書を作成する
DbjObj obj = docspc.createVrDocument(
             "mdmClass_CfgH",
             // バージョニングオブジェクトのトップオブジェクトクラス
             "mdmClass_Document",
             // バージョンオブジェクトのトップオブジェクトクラス
             props,            // プロパティ値集合
             uploadlist,       // 文書のアップロード情報のリスト
             linklist );       // リンク設定情報のリスト
```

(2) 文書の全文検索インデクスの作成

ここでは，文書の全文検索インデクスの作成方法について説明します。

全文検索インデクスは，バージョンなし文書またはバージョン付き文書の作成時に作成できます。また，文書のコンテンツ更新時にも作成できます。

全文検索インデクスの作成方法は，文書のアップロード情報（DbjUploadInfo インターフェース）で指定します。ただし，文書のトップオブジェクトクラスが全文検索機能付き文書クラス以外の場合，全文検索インデクスは作成されません。全文検索機能付き文書クラスの作成方法については，マニュアル「DocumentBroker Version 5　概説」の全文検索インデクスの作成を参照してください。また，文書の作成時に全文検索インデクスを作成する場合は，全文検索インデクスの作成方法を指定した文書のアップロード情報は，リストの先頭に指定する必要があります。

全文検索インデクスの作成方法には，次の 2 通りの方法があります。

● コンテンツとして登録するファイルから全文検索インデクスを作成する方法

「text/」から始まる特定のレンディションタイプを持つ文書の場合，コンテンツとして登録するファイルから全文検索インデクスを作成できます。マルチレンディション文書の場合は，マスタレンディションのコンテンツに登録するファイルのレンディションタイプが特定のレンディションタイプであるときに作成できます。

文書のアップロード情報に，全文検索インデクスの情報として，定数「DbjDef.INDEXPATH_SAME」を指定します。

● 文書のコンテンツに対応するテキストデータを用意して全文検索インデクスを作成する方法

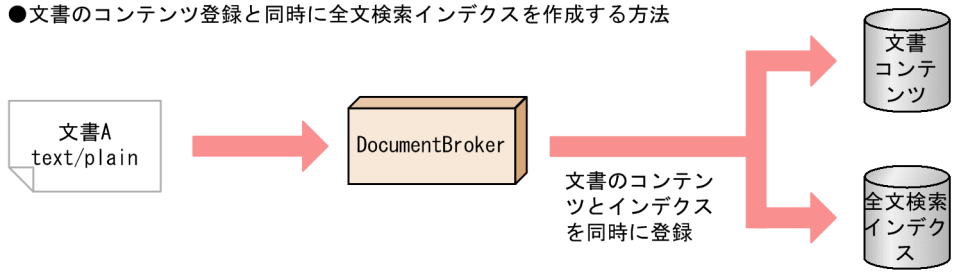
特定のレンディションタイプを持たない文書に対しては，文書のコンテンツに対応するテキストデータを全文検索インデクスとして登録します。

例えば，Word で作成した文書など，「text/」以外で始まるレンディションタイプのファイルからは，全文検索インデクスが作成できません。この場合，全文検索インデクスを作成するためのテキストデータを，ユーザアプリケーションプログラムなどによって，別に用意する必要があります。Word などのアプリケーションプログラムでテキストデータを抽出してください。このテキストデータのパス名を，文書のアップロード情報の全文検索インデクスの情報として指定します。

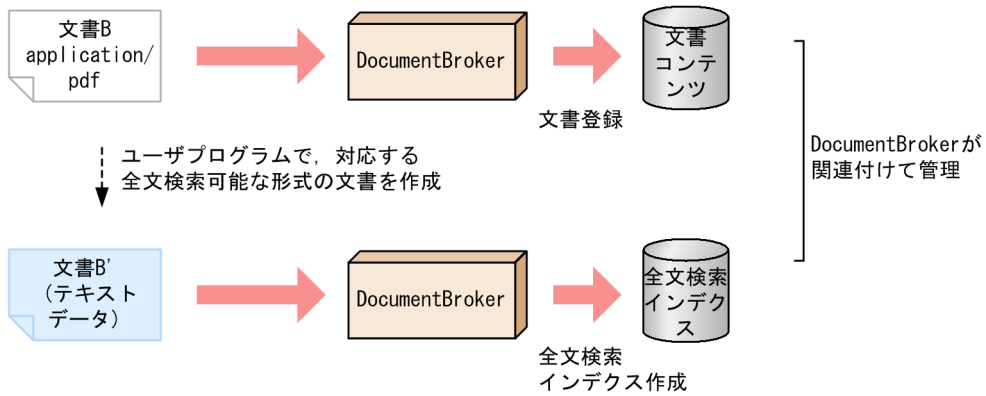
それぞれの方法の概要を次の図に示します。

図 1-6 全文検索インデクスの作成方法

●文書のコンテンツ登録と同時に全文検索インデクスを作成する方法



●文書のコンテンツに対応するテキストデータを用意して全文検索インデクスを登録する方法



文書Aは特定のレンディションタイプの文書です。
文書Bはアプリケーションプログラムで作成した，特定のレンディションタイプでない文書です。
文書B'は，文書Bのコンテンツに対応する内容のテキストデータです。

注意事項

DocumentBroker では，文書のコンテンツと全文検索インデクスの内容との対応は管理しません。作成または更新する文書の内容と，全文検索インデクス生成用ファイルの内容との対応は，ユーザアプリケーションプログラムで管理してください。

ここでは，バージョンなし文書の作成と同時に，コンテンツとして登録するファイルから全文検索インデクスを作成する例を示します。

// コンテンツとして登録するファイルから
// 全文検索インデクスを作成する例

```
  // factory ： DbjFactoryインターフェース
  // docspc  ： DbjDocSpaceインターフェース

// 初期値として設定するプロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();

// プロパティ(Author)を設定する
props.setPropVal( "Author", "suzuki" );

// 文書のアップロード情報のリストを作成する
List<DbjUploadInfo> uploadlist = new ArrayList<DbjUploadInfo>();
uploadlist.add( factory.createUploadInfo(
                  file_txt,
                  retrievalName,
                  null,     // レンディションタイプは自動で設定する
                  null,     // レンディションプロパティは設定しない
```

```
                    DbjDef.INDEXPATH_SAME ) );
                            // 全文検索インデクスを作成する

// バージョンなし文書を作成する
DbjObj obj = docspc.createDocument(
           "mdmClass_Document", // 全文検索機能付き文書クラス
            props,              // プロパティ値集合
            uploadlist,         // 文書のアップロード情報のリスト
            null );             // リンクは設定しない
```

1.9.4 文書管理オブジェクトの検索

ここでは，文書管理オブジェクトの検索について説明します。

文書管理オブジェクトの検索では，検索条件を edmSQL 文で指定します。検索結果は検索結果集合として取得します。検索条件の指定方法と検索結果の取得方法については，マニュアル「DocumentBroker Version 5　概説」の検索機能および「2.　edmSQL の文法」を参照してください。

文書管理オブジェクトの検索は，DbjDocSpace#executeSearch メソッドで実行します。検索実行が成功した場合，戻り値として，検索結果集合を扱う DbjResultSet インターフェースが返却されます。

なお，DbjDocSpace#executeSearch メソッドでは，次のような種類の検索が実行できます。

- edmSQL 文だけを指定する単純な検索
- ? パラメタを指定する検索
- ロック指定検索
- アクセス制御機能付き検索
- 名前付き検索結果を取得する検索
- キャッシュ検索

それぞれの検索方法について説明します。

(1) edmSQL 文だけを指定する単純な検索

ここでは，edmSQL 文だけを指定する単純な検索の例を示します。

// edmSQL文だけを指定する検索の例

```
  // docspc ： DbjDocSpaceインターフェース

// 検索実行
DbjResultSet result = docspc.executeSearch(
            "SELECT Name FROM DV  WHERE Number < 10", // edmSQL文
            null,  // ?パラメタは指定しない
            null );// 検索結果取得情報は指定しない
System.out.println( "result = " + result );
```

検索を実行すると，検索結果集合を表すオブジェクトが作成されます。この例の場合，検索結果集合は変数 result に返却されます。検索結果を参照する場合は，result のインターフェースである，DbjResultSet インターフェースを使用します。

(2) ? パラメタを指定する検索

ここでは，? パラメタを指定した検索の例を示します。? パラメタは，リストにして，

DbjDocSpace#executeSearch メソッドの引数に指定します。? パラメタの値は，パラメタクラスのインターフェースを使用して受け渡します。? パラメタを扱うインターフェースは，DocumentBroker のデータ型ごとに定義されていますので，データ型に合ったインターフェースを使用してください。

edmSQL 文中に複数の ? パラメタがある場合，その個数と同じ数の要素を持つ ? パラメタのリストを指定します。

// ?パラメタを指定する検索の例

```
  // factory：DbjFactoryインターフェース
  // docspc ：DbjDocSpaceインターフェース

// ?パラメタに，INT型の値10を指定する
List<DbjQParam> qparams = new ArrayList<DbjQParam>();
qparams.add( factory.createInteger32QParam( 10 ));

// 検索実行
DbjResultSet result = docspc.executeSearch(
            "SELECT Name FROM DV WHERE Number < ?", // edmSQL文
            qparams, // ?パラメタの指定
            null );  // 検索結果取得情報は指定しない
System.out.println( "result = " + result );
```

(3) ロック指定検索

ここでは，ロック指定検索の例を示します。この検索は，引数にロック種別が指定できる形式の DbjDocSpace#executeSearch メソッドで実行します。ロック種別を指定できない形式を使用した場合や，ロック種別に DbjDef.LOCK_NONE を指定した場合，検索結果として取得した文書管理オブジェクトにロックは設定されません。

// ロック指定検索の例

```
  // docspc ： DbjDocSpaceインターフェース

DbjResultSet result = docspc.executeSearch(
            "SELECT Name  FROM DV  WHERE Number < 10", // edmSQL文
            null,               // ?パラメタは指定しない
            null,               // 検索結果取得情報は指定しない
            DbjDef.LOCK_READ ); // readロックを設定する
```

(4) 検索実行時のアクセス制御モードの変更

ここでは，検索実行時のアクセス制御モードを変更する例を示します。アクセス制御機能に対応した文書空間の場合，デフォルトの設定では，アクセス制御機能付き検索が実行されます。検索結果をアクセス制御しない場合は，アクセス制御モードを変更してください。

アクセス制御モードの変更は，DbjDocSpace#changeSearchACLMode メソッドで実行します。なお，このメソッドは，アクセス制御機能に対応した文書空間以外では実行できません。変更したアクセス制御モードは，次に DbjDocSpace#changeSearchACLMode メソッドを実行するまで有効です。

この例では，アクセス制御モードを変更して，アクセス制御機能なし検索を実行します。

// アクセス制御モードを変更する検索の例

```
  // docspc：DbjDocSpaceインターフェース

// アクセス制御機能を無視する検索モード（アクセス制御機能なし検索）
// に変更する
docspc.changeSearchACLMode( DbjDef.WITHOUT_ACL );
```

```
// 検索実行（アクセス制御機能なし検索）
DbjResultSet result = docspc.executeSearch(
            "SELECT Name FROM DV WHERE Number < 10", // edmSQL文
            null,          // ?パラメタは指定しない
            null );        // 検索結果取得情報は指定しない
```

なお，アクセス制御機能付き検索についての詳細は，マニュアル「DocumentBroker Version 5　概説」の「アクセス制御機能付き検索」の説明を参照してください。

(5) 名前付き検索結果を取得する検索

ここでは，名前付き検索結果を取得する検索の例を示します。この検索は，引数に選択項目のリストが指定できる形式の DbjDocSpace#executeSearch メソッドで実行します。名前付き検索結果および名前なし検索結果については，マニュアル「DocumentBroker Version 5　概説」の「名前付き検索結果を取得する検索」の説明を参照してください。

選択項目のリストを指定する引数には，検索結果集合の列に対応する名前をリストで設定します。リストの要素には，列に付ける名前として，SELECT 句に指定する選択項目（プロパティ名）を設定します。

なお，名前付き検索結果を取得する場合，edmSQL 文の SELECT 句は，「$_」と記述してください。「$_」は，検索実行時に，選択項目のリストに指定した項目に置換されます。

// 名前付き検索結果を取得する例1

```
  // docspc ： DbjDocSpaceインターフェース

// 列名を指定したリストを作成する
List<String> selectItems = new ArrayList<String>();
selectItems.add("dmaProp_OIID");     // プロパティ名
selectItems.add("S0.Name");          // 相関名.プロパティ名

  // 下記のedmSQL文に展開される
  // SELECT dmaProp_OIID, S0.Name  FROM DV S0  WHERE Number < 10
  // 検索結果集合resultの列名は，dmaProp_OIID,S0.Nameとなる

// 検索実行
DbjResultSet result = docspc.executeSearch(
            selectItems,         // 列名を設定したリスト
            "SELECT $_ FROM DV S0 WHERE Number < 10",
                                 // edmSQL文
             null,               // ?パラメタは指定しない
             null );             // 検索結果取得情報は指定しない
System.out.println(
"Column Name[0] is "+ result.getColumnName(0));
System.out.println(
"Column Name[1] is "+ result.getColumnName(1));
```

名前なし検索結果として取得した検索結果集合に，あとから列名を付けることもできます。名前なし検索結果に対する列名の設定は，DbjResultSet#setColumnMetaName メソッドで実行します。

また，名前付き検索結果の任意の列から，列名を削除する（列名に null を設定する）こともできます。ただし，すべての列名を削除しても，名前付き検索結果は名前なし検索結果にはなりません。

次に，名前なし検索結果にあとから列名を設定する例を示します。

// 名前なし検索結果に列名を設定する例

```
  // docspc ： DbjDocSpaceインターフェース

// 検索実行
```

```
DbjResultSet result = docspc.executeSearch(
   "SELECT dmaProp_OIID, S0.Name FROM DV S0  WHERE Number < 10",
                    // edmSQL文
   null,            // ?パラメタは指定しない
   null );          // 検索結果取得情報は指定しない
  // この時点では名前なし検索結果である

// 列名を設定する
result.setColumnMetaName(0, "dmaProp_OIID");//プロパティ名を列名に設定する
  // この時点で名前付き検索結果になる
result.setColumnMetaName(1, "S0.Name");// 相関名.プロパティ名を列名に設定する
```

なお，名前付き検索結果に対して，DbjResultSet#getPropSet メソッドを実行すると，取得するプロパティ値集合のプロパティ名に，検索結果集合の列名がマッピングされます。ただし，列名に相関名またはクラス名が含まれている場合は，列名のうち，プロパティ名の部分だけがプロパティ名としてマッピングされます。

ここでは，検索を実行して取得した検索結果の 1 行目を，プロパティ値集合として取得する例を示します。

// 名前付き検索結果を取得する例2

```
  // docspc ： DbjDocSpaceインターフェース

// 列名を指定したリストを作成する
List<String> selectItems = new ArrayList<String>();
selectItems.add("dmaProp_OIID");  // プロパティ名
selectItems.add("S0.Name");       // 相関名.プロパティ名

  // 下記のedmSQL文に展開される
  // SELECT dmaProp_OIID, S0.Name  FROM DV S0  WHERE Number < 10
  // 検索結果集合resultの列名は，dmaProp_OIID, S0.Nameとなる

// 検索実行
DbjResultSet result = docspc.executeSearch(
            selectItems,     // 列名を指定したリストを指定
            "SELECT $_  FROM DV S0  WHERE Number < 10",
                             // edmSQL文
            null,            // ?パラメタは指定しない
            null );          // 検索結果取得情報は指定しない

// 検索結果集合の列名を出力する
System.out.println(
"Column Name[0] is "+ result.getColumnName(0));
System.out.println(
"Column Name[1] is "+ result.getColumnName(1));

// カーソルを先頭行に移動する
result.next();

// 検索結果集合の１行目をプロパティ値集合として取得する
DbjPropSet props = result.getPropSet();
// propNamesの内容は{"dmaProp_OIID", "Name"}になる
Set<String> propNames = props.keySet();
```

検索結果集合に複数の DocumentBroker クラスの列（文書管理オブジェクトのプロパティ）が含まれる場合は，DbjResultSet#getPropSet メソッドで行のプロパティ値集合を指定する時に，引数に表名（DocumentBroker クラス名）が指定できる形式を使用して，特定の DocumentBroker クラスのプロパティだけを取得することもできます。例えば，検索結果集合に S0.dmaProp_OIID と S1.Name という列が含まれる場合，表名として引数に "S0" を指定して，

```
props = result.getPropSet("S0");
```

とすれば，相関名 S0 で表される DocumentBroker クラスのプロパティである dmaProp_OIID だけが取得できます。DocumentBroker クラスを指定しない場合は，検索結果集合に含まれるすべての DocumentBroker クラスのプロパティが取得できます。

(6) キャッシュ検索

ここでは，キャッシュ検索の例を示します。キャッシュ検索を実行する場合，キャッシュ名を設定した検索結果取得情報を指定して検索を実行します。検索結果取得情報は，DbjFetchInfo インターフェースで設定します。検索結果取得情報では，キャッシュ名のほか，検索結果の取得開始位置，取得件数，キャッシュの最大件数およびキャッシュキーを設定できます。キャッシュ名を設定していない場合，キャッシュ検索は実行されません。キャッシュ検索の概要については，マニュアル「DocumentBroker Version 5　概説」の「検索結果キャッシュと検索結果取得情報を指定した検索」を参照してください。

検索結果キャッシュを使用した検索の例を示します。

// 検索結果キャッシュを使用した検索の例１

```
  // factory ： DbjFactoryインターフェース
  // docspc ： DbjDocSpaceインターフェース

// 検索結果キャッシュを作成して，先頭から100件ずつ検索結果を取得する

// 検索結果取得情報を作成する
// キャッシュ名は"cacheA"とする
DbjFetchInfo fetchinfo = factory.createFetchInfo(
             0,         // 取得開始位置
             100,       // 取得件数=100件
             -1,        // 検索結果キャッシュ取得最大件数(最大)
             "cacheA", // キャッシュ名
             DbjDef.INITIAL_KEY );     // キャッシュキー

DbjResultSet result = null ;

while ( true ) {

  // ループの2回目以降では検索結果キャッシュから検索結果を100件取得する
  result = docspc.executeSearch(
           "SELECT dmaProp_OIID  FROM DV", // edmSQL文
           null,                  // ?パラメタは指定しない
           fetchinfo );           // 検索結果取得情報

  // 検索結果キャッシュの最終行を取得するまでループする
  if ( fetchinfo.getStartIndex()
          + fetchinfo.getFetchCount() >= fetchinfo.getCacheTotal() ) {
          break;
  }

  fetchinfo.setStartIndex(
    fetchinfo.getStartIndex() + fetchinfo.getFetchCount() );

  // 同じfetchinfoを使用するため，キャッシュキーを再設定する
  // 必要はない

}
```

キャッシュ検索とキャッシュなし検索を組み合わせて使うこともできます。

検索結果キャッシュから取得した検索結果を？パラメタとして指定して，キャッシュなし検索を実行する例を示します。この例では，まず，検索結果として OIID を取得するキャッシュ検索を実行します。キャッシュ名は，「search_1」です。次に，「search_1」から検索結果（OIID）を 20 件ずつ取得しながら，その OIID を持つオブジェクトの Name プロパティをキャッシュなし検索によって取得します。

```
// 検索結果キャッシュを使用した検索の例2

  // factory ： DbjFactoryインターフェース
  // docspc ： DbjDocSpaceインターフェース

// 検索取得開始位置を指定する
int startix = 0;

// 取得件数=20件
int fetchcount = 20;

// 検索結果取得情報fetchinfoを作成する
// 取得件数とキャッシュ名は固定で使用するためここで設定する
// 取得開始位置はループ内で設定するため設定しない
DbjFetchInfo fetchinfo = factory.createFetchInfo();
fetchinfo.setFetchCount( fetchcount );
fetchinfo.setCacheName( "search_1" ); // キャッシュ名を指定する

// 検索結果を分割して取得する
while( true ) {

  fetchinfo.setStartIndex( startix ); // 取得開始位置を設定する

  // キャッシュ検索を実行する
  DbjResultSet result = docspc.executeSearch(
            "SELECT dmaProp_OIID FROM DV WHERE ...",//edmSQL文
            null,          // ?パラメタは指定しない
            fetchinfo);    // 検索結果取得情報

  // 検索結果集合resultを表示する
  while( result.next() )
  {
    // 検索結果として取得したOIIDを?パラメタの値に設定する
    DbjObj obj = docspc.createObjConnection(
            result.getStringVal(0) );
    List<DbjQParam> qparamlst = new ArrayList<DbjQParam>();
    qparamlst.add( factory.createOIIDQParam(obj) );

    // キャッシュなし検索を実行する
    DbjResultSet result2 = docspc.executeSearch(
            "SELECT Name  FROM DV  WHERE dmaProp_OIID=?",
             qparamlst,       // ?パラメタ（検索結果として取得したOIID）
             null );

    result2.next();

    Sysmtem.out.println("name="+result2.getStringVal(0));
  }

  // 開始位置+取得件数がキャッシュ件数以上になったらbreakする
  if ( startix + fetchcount >= fetchinfo.getCacheTotal() ) {
    break;
  }

  startix = startix + fetchcount; // 開始位置をインクリメントする

}
```

なお，検索結果取得情報は，DbjDocSpace#executeSearch メソッド以外に，文書管理オブジェクトの一覧を取得する場合にも使用できます。

1.9.5 検索条件に合致した文書管理オブジェクトの削除

ここでは，検索条件に合致する文書管理オブジェクトをすべて削除する操作について説明します。この操作には，DbjDocSpace#removeObjects メソッドを使用します。

DbjDocSpace#removeObjects メソッドの引数には，edmSQL 文と？パラメタのリストを指定します。edmSQL 文の SELECT 句の一つ目の項目には，dmaProp_OIID プロパティを指定してください。なお，項目名は「dmaProp_OIID」でなく，dmaProp_OIID プロパティを示す相関名付きの名前などでもかまいません。

検索条件を指定して文書管理オブジェクトを削除する例を示します。

// 検索条件を指定して文書管理オブジェクトを削除する例

```
  // docspc：DbjDocSpaceインターフェース

docspc.removeObjects(
  "SELECT dmaProp_OIID FROM DV WHERE mdmProp_Num>100",
  null );             // ?パラメタは指定しない
```

1.9.6 既存の文書管理オブジェクトにアクセスするインターフェースの取得

ここでは，既存の文書管理オブジェクトにアクセスするインターフェースの取得について説明します。

DbjDocSpace インターフェースから取得できる，文書管理オブジェクトを操作するためのインターフェースは，次のとおりです。

● DbjObj インターフェース

● DbjObjList インターフェース

● DbjLinkObjList インターフェース

これらのインターフェースについては，「1.10　文書管理オブジェクトの操作」を参照してください。

ここでは，検索で取得した OIID を基に，DbjObj インターフェースを取得する例を示します。

// 文書管理オブジェクトにアクセスするインターフェースを取得する例

```
  // docspc：DbjDocSpaceインターフェース

//OIIDをresultに取得する
DbjResultSet result = docspc.executeSearch(
    "SELECT dmaProp_OIID  FROM DV  WHERE Number = 100", // edmSQL文
    null,
    null );

result.next();

//resultで取得したOIIDを基にDbjObjインターフェースを取得する
String oiid = result.getStringVal(0);
DbjObj obj = docspc.createObjConnection(oiid);
```

1.9.7 メタ情報を取得するインターフェースの取得

DbjDocSpace インターフェースでは，次のインターフェースも取得できます。

● DbjMeta インターフェース

DbjMeta インターフェースの詳細については，「1.12　メタ情報の取得」を参照してください。

ここでは，セッションを確立している文書空間のメタ情報を取得する例を示します。

```
// セッションを確立している文書空間の，
// 文書管理オブジェクトのプロパティのデータ型を取得する例

  // sess：DbjSessionインターフェース

DbjDocSpace docspc = sess.login( user, passwd );

// DbjMetaインターフェースを取得
DbjMeta meta = docspc.getMeta();

// Nameプロパティのデータ型を取得する
int datatype = meta.getPropDataType("Name");
```

1.10　文書管理オブジェクトの操作

この節では，文書管理オブジェクトの操作について説明します。

1.10.1　文書管理オブジェクトと Proxy オブジェクト

ここでは，文書管理オブジェクトと Proxy オブジェクトの関係について説明します。文書管理オブジェクトは，データベース上に存在する DocumentBroker オブジェクトと対応する，永続的なオブジェクトです。文書空間上の文書やフォルダを表します。文書管理オブジェクトについての詳細は，マニュアル「DocumentBroker Version 5　概説」を参照してください。

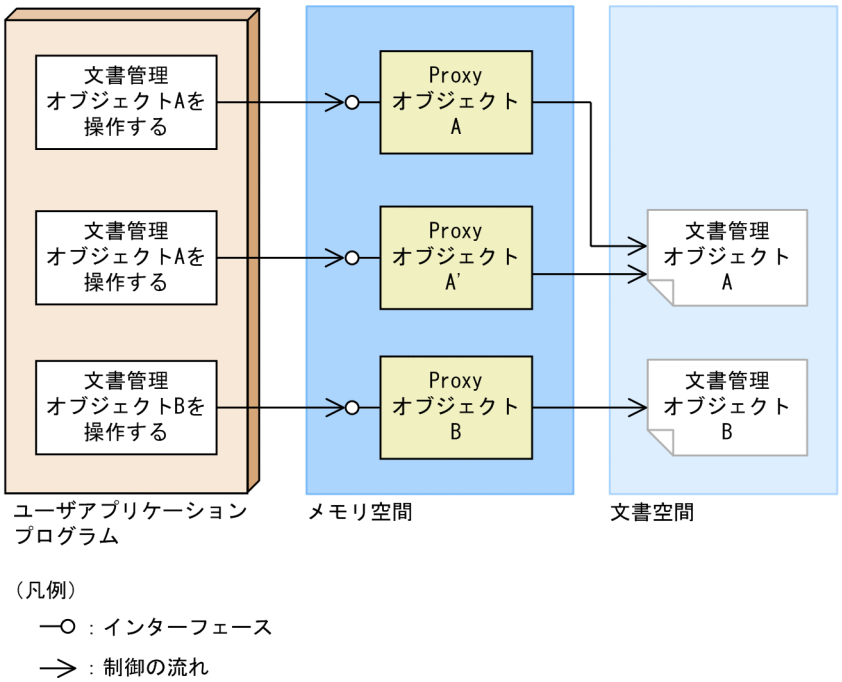
Proxy オブジェクトは，文書管理オブジェクトの概念的な**代理オブジェクト**です。Proxy オブジェクトは，メモリ空間上に存在します。

DocumentBroker クラスライブラリで作成したユーザアプリケーションプログラムで文書管理オブジェクトを操作する場合，文書管理オブジェクトを直接操作するのではなく，Proxy オブジェクトを通して間接的に操作します。このため，文書管理オブジェクトを操作するためには，文書管理オブジェクトそのもののインターフェースではなく，Proxy オブジェクトのインターフェースを取得して操作します。

Proxy オブジェクトと文書管理オブジェクトの対応は，n:1（n は 1 以上の整数）です。一つの文書管理オブジェクトに対して，複数の Proxy オブジェクトからアクセスすることがあります。Proxy オブジェクトは，文書管理オブジェクトを操作するためのインターフェース（DbjObj インターフェース，DbjLinkObj インターフェースなど）を取得するごとにメモリ上に作成されます。Proxy オブジェクトが対象にする文書管理オブジェクトを，Proxy オブジェクトの**ターゲットオブジェクト**といいます。

Proxy オブジェクトを通して文書管理オブジェクトを操作する場合の例を，次の図に示します。

図 1-7　Proxy オブジェクトを通して文書管理オブジェクトを操作する例

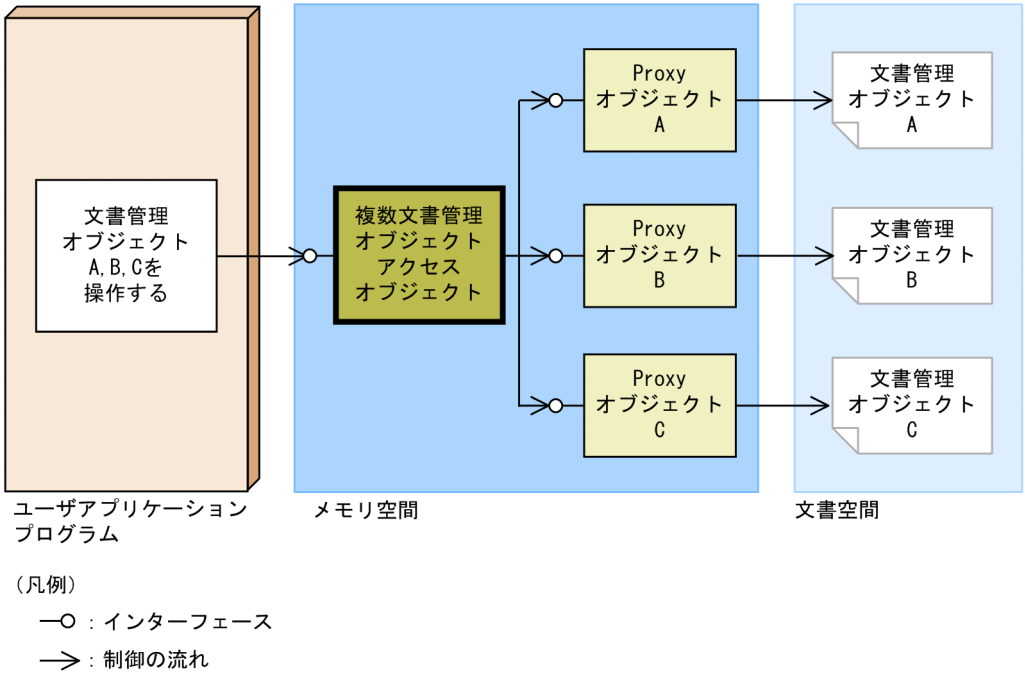


また，DocumentBroker クラスライブラリでは，複数の文書管理オブジェクトをまとめて操作することも

できます。複数の文書管理オブジェクトを操作する場合は，複数の Proxy オブジェクトのインターフェースを要素に持つ，**複数文書管理オブジェクトアクセスオブジェクト**のインターフェースを使用します。

複数の文書管理オブジェクトを操作する例を，次の図に示します。

図 1-8　複数の文書管理オブジェクトを操作する例



複数文書管理オブジェクトアクセスオブジェクトのインターフェースは，Proxy オブジェクトのインターフェースである DbjObj インターフェース，DbjVerObj インターフェースまたは DbjLinkObj インターフェースを要素とするリストのインターフェースです。

なお，Proxy オブジェクトには，文書管理オブジェクトのうち，リンクオブジェクトに対応する Proxy オブジェクトと，それ以外の文書管理オブジェクトに対応する Proxy オブジェクトがあります。リンクオブジェクトに対応する Proxy オブジェクトを特に，**リンク Proxy オブジェクト**といいます。以降，単に Proxy オブジェクトと説明する場合は，リンク Proxy オブジェクト以外の Proxy オブジェクトのことを指します。

1.10.2　Proxy オブジェクトのプロパティ

ここでは，Proxy オブジェクトのプロパティについて説明します。

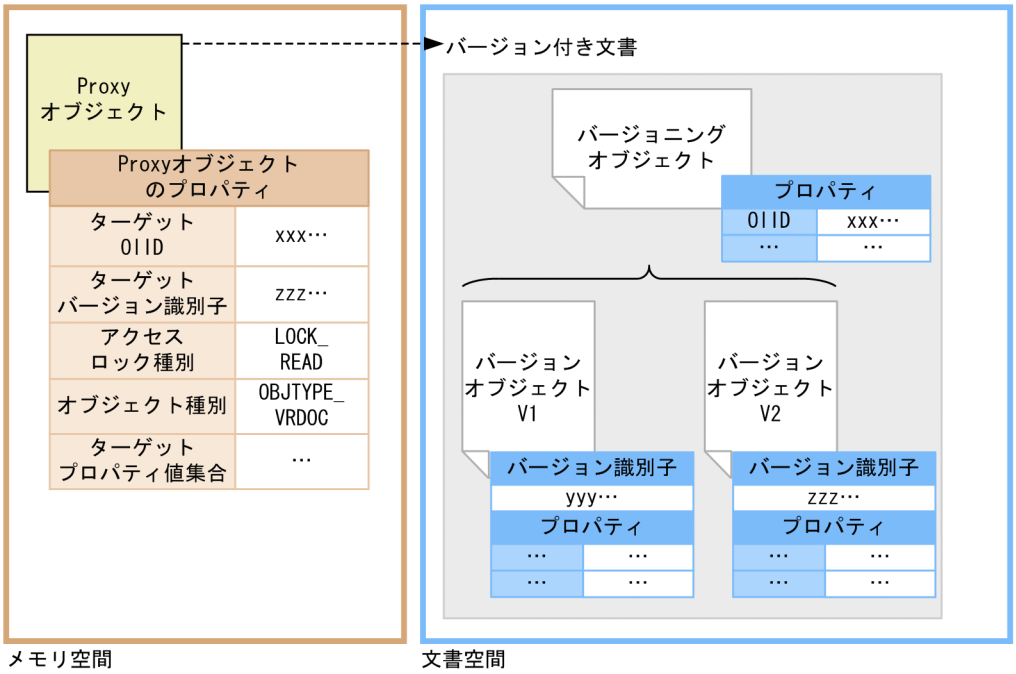
Proxy オブジェクトはプロパティを持っています。このプロパティは，文書管理オブジェクトのプロパティとは異なりますので，注意してください。

Proxy オブジェクトのプロパティを次に示します。

- ターゲット OIID
- ターゲットバージョン識別子
- アクセスロック種別
- オブジェクト種別
- ターゲットプロパティ値集合

Proxy オブジェクトのプロパティと文書管理オブジェクトの関係を，次の図に示します。

図 1-9　Proxy オブジェクトのプロパティと文書管理オブジェクトの関係



（凡例）

-----►：矢印の先がターゲットオブジェクトであることを示します。

注　図中のOIIDやバージョン識別子などの値は，実際の形式とは異なります。

それぞれのプロパティについて説明します。

(1) ターゲット OIID

ターゲットオブジェクトの OIID が設定されるプロパティです。

(2) ターゲットバージョン識別子

ターゲットオブジェクトがバージョン付きオブジェクトの場合に，操作対象になるバージョンオブジェクトのバージョン識別子が設定されるプロパティです。ターゲットバージョン識別子に対応するバージョンを，**ターゲットバージョン**といいます。

このプロパティには，ターゲットオブジェクトがバージョニングオブジェクトの場合に有効な値が格納されます。デフォルトのターゲットバージョン識別子はカレントバージョンのバージョン識別子になります。図 1-9 の場合は，バージョン識別子「zzz…」を持つ「V2」がターゲットバージョンです。ターゲットバージョンは，必要に応じて，任意のバージョンに変更できます。

なお，ターゲットオブジェクトがバージョンオブジェクト（バージョン付きオブジェクトの 1 バージョン）の場合またはバージョン付きオブジェクトでない場合，ターゲットバージョン識別子プロパティの値は無効です。

(3) アクセスロック種別

ターゲットオブジェクトに設定するロック種別が設定されるプロパティです。

(4) オブジェクト種別

ターゲットオブジェクトのオブジェクト種別が設定されるプロパティです。

オブジェクト種別とは，次のどれかです。

- バージョンなし文書
- バージョン付き文書
- バージョンなしフォルダ
- 独立データ
- パブリック ACL

(5) ターゲットプロパティ値集合

ターゲットオブジェクトである文書管理オブジェクトのプロパティを参照または更新するためのプロパティ値集合が設定されるプロパティです。

文書管理オブジェクトのプロパティを参照する場合，まず，文書管理オブジェクトのプロパティ値集合が，このプロパティに読み込まれます。この処理を，「ターゲットオブジェクトから Proxy オブジェクトにプロパティを**ロード**する」といいます。

また，文書管理オブジェクトのプロパティを設定する場合も，Proxy オブジェクトのターゲットプロパティ値集合の内容が，文書管理オブジェクトに反映（設定）されます。この処理を，「Proxy オブジェクトのプロパティをターゲットオブジェクトに**フラッシュ**する」といいます。

文書管理オブジェクトのプロパティの操作についての詳細は，「1.10.7　文書管理オブジェクトのプロパティの操作」を参照してください。

1.10.3　リンク Proxy オブジェクトのプロパティ

ここでは，リンク Proxy オブジェクトのプロパティについて説明します。

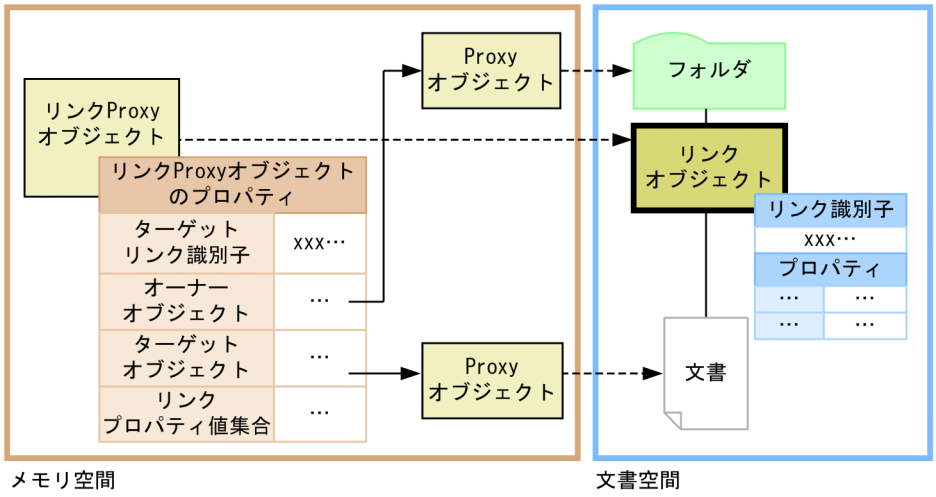
リンク Proxy オブジェクトはプロパティを持っています。このプロパティは，文書空間に存在するリンクオブジェクトのプロパティとは異なりますので，注意してください。

リンク Proxy オブジェクトのプロパティを次に示します。

- ターゲットリンク識別子
- オーナーオブジェクト
- ターゲットオブジェクト
- リンクプロパティ値集合

リンク Proxy オブジェクトのプロパティと文書空間のリンクオブジェクトの関係を，次の図に示します。

図 1-10　リンク Proxy オブジェクトのプロパティと文書管理オブジェクトの関係



（凡例）

-----►：矢印の先がターゲットオブジェクトであることを示します。

——►：矢印の先がプロパティの値が示すオブジェクトであることを示します。

注　図中のリンク識別子などの値は，実際の形式とは異なります。

それぞれのプロパティについて説明します。

(1) ターゲットリンク識別子

ターゲットオブジェクトであるリンクオブジェクト（**ターゲットリンクオブジェクト**）のリンク識別子が設定されるプロパティです。

(2) オーナーオブジェクト

ターゲットリンクオブジェクトによってリンクが設定されている文書管理オブジェクトのうち，リンク元である文書管理オブジェクトの Proxy オブジェクトが設定されるプロパティです。

例えば，ターゲットオブジェクトがフォルダと文書のリンクを表すリンクオブジェクトである場合は，フォルダの Proxy オブジェクトが設定されます。また，ターゲットオブジェクトが文書間リンクを表すリンクオブジェクトである場合は，リンク元の文書の Proxy オブジェクトが設定されます。

(3) ターゲットオブジェクト

ターゲットリンクオブジェクトによってリンクが設定されている文書管理オブジェクトのうち，リンク先である文書管理オブジェクトの Proxy オブジェクトが設定されるプロパティです。

例えば，リンク Proxy オブジェクトのターゲットオブジェクトがフォルダと文書のリンクを表すリンクオブジェクトである場合は，文書の Proxy オブジェクトが設定されます。また，ターゲットオブジェクトが文書間リンクを表すリンクオブジェクトである場合は，リンク先の文書の Proxy オブジェクトが設定されます。

(4) リンクプロパティ値集合

ターゲットオブジェクトであるリンクオブジェクトのプロパティを参照または更新するためのプロパティ値集合が設定されるプロパティです。

1.10.4 文書管理オブジェクトを操作するインターフェースの機能

ここでは，文書管理オブジェクトを操作するインターフェースの機能について説明します。

(1) 文書管理オブジェクトを操作するインターフェースの種類

特定の文書管理オブジェクトを操作する場合に使用するインターフェースについて説明します。
DocumentBroker クラスライブラリでは，文書管理オブジェクトを，種類に関係なく同じインターフェースを使用して操作します。例えば，文書管理オブジェクトのプロパティを操作する場合は，操作する対象が文書やフォルダなど，どの文書管理オブジェクトであっても，同じ DbjObj インターフェースなどのメソッドを使用します。ただし，操作する文書管理オブジェクトの種類によって，実行できるメソッドは異なります。例えば，フォルダを操作するための DbjObj インターフェースを取得している場合に，文書のコンテンツを更新する DbjObj#uploadContents メソッドを実行することはできません。詳細は，「(3) メソッドとオブジェクト種別の対応」を参照してください。

文書管理オブジェクトを操作するために使用するインターフェースの種類を次に示します。また，インターフェースを取得するメソッドの例を示します。

● DbjObj インターフェース
個々の文書管理オブジェクトの操作を実行します。
次のメソッドで取得します。
- DbjDocSpace#createObjConnection メソッド
- DbjDocSpace#createDocument メソッドなど，文書管理オブジェクトを作成するメソッド
- DbjLinkObj#getOwnerObj メソッドなど，リンク元オブジェクトまたはリンク先オブジェクトを取得するメソッド

● DbjVerObj インターフェース
次のメソッドで取得します。
- DbjVerObjList#getVerObj メソッド

このインターフェースの機能を持つ Proxy オブジェクトのターゲットオブジェクトは，必ずバージョンオブジェクトです。このインターフェースでは，DbjObj インターフェースと同様に，個々の文書管理オブジェクトの操作を実行します。また，ターゲットオブジェクトであるバージョンオブジェクトのバージョン識別子をプロパティとして保持しています。

● DbjObjList インターフェース
複数の文書管理オブジェクトの一括操作を実行します。
次のメソッドで取得します。
- DbjDocSpace#createObjList メソッド
- DbjLinkObjList#getOwnerObjList メソッドなど，リンク元オブジェクトまたはリンク先オブジェクトの一覧を取得するメソッド

● DbjVerObjList インターフェース
複数の文書管理オブジェクトの一括操作を実行します。リストの要素は DbjVerObj インターフェースです。
次のメソッドで取得します。
- DbjObj#getVersionObjList メソッド

● DbjLinkObj インターフェース
リンクオブジェクトの操作を実行します。
次のメソッドで取得します。
- DbjLinkObjList#getLinkObj メソッド

● DbjLinkObjList インターフェース
複数のリンクオブジェクトの一括操作を実行します。
次のメソッドで取得します。
- DbjDocSpace#createLinkObjList メソッド
- DbjObj#getChildList メソッドなど，リンク元オブジェクトまたはリンク先オブジェクトの一覧を取得するメソッド

なお，DbjObjList インターフェース，DbjVerObjList インターフェースおよび DbjLinkObjList インターフェースは，java.util.List インターフェースを継承しています。

(2) 文書管理オブジェクトを操作するインターフェースで実行できる操作

ここでは，文書管理オブジェクトを操作するインターフェースで実行できる操作について説明します。

文書管理オブジェクトを操作するインターフェースで実行できる操作を，次の表に示します。それぞれの操作の詳細については，参照先の説明を参照してください。

表 1-6 文書管理オブジェクトを操作するインターフェースで実行できる操作

実行できる操作の種類	説　明	参照先
文書管理オブジェクトの情報を取得する	OIID やオブジェクト種別，文書管理オブジェクトを構成する DocumentBroker クラスの情報を取得します。	1.10.5
文書管理オブジェクトへのアクセス方法を変更する	バージョン付きオブジェクトの操作対象のバージョンを変更したり，文書管理オブジェクトに設定するロック種別を変更したりします。	1.10.6
文書管理オブジェクトのプロパティを操作する	文書管理オブジェクトに設定したプロパティを参照したり，更新したりします。	1.10.7
文書管理オブジェクトを削除する	文書管理オブジェクトを削除します。	1.10.8
文書のコンテンツを操作する	文書のコンテンツをダウンロードしたり，アップロードしたりします。	1.10.9
バージョン付きオブジェクトのバージョンを操作する	バージョン付きオブジェクトをバージョンアップしたり，バージョン一覧を取得したり，バージョンを削除したりします。	1.10.10
マルチレンディション文書を操作する	マルチレンディション文書を作成したり，レンディションを追加したり，レンディションにプロパティを設定したりします。	1.10.11
リファレンスファイル文書を操作する	リファレンスファイル文書を作成したり，削除したりします。また，リファレンスファイル文書のコンテンツをアップロードしたり，ダウンロードしたりします。	1.10.12
文書管理オブジェクトを関連付けて管理する	文書管理オブジェクトを関連付けるために，リンクオブジェクトを使用して管理します。また，リンクをたどって文書管理オブジェクトを取得します。	1.10.13
複数の文書管理オブジェクトを一括して操作する	複数の文書管理オブジェクトのプロパティやコンテンツを一括して参照したり，複数の文書管理オブジェクトを一括して移動したり，削除したりします。	1.10.14
アクセス制御機能を使用する	ローカル ACL の値を設定したり，パブリック ACL を作成して文書管理オブジェクトからバインドしたりします。	1.11

(3) メソッドとオブジェクト種別の対応

DbjObj インターフェースおよび DbjVerObj インターフェースのメソッドと，実行できる文書管理オブジェクトのオブジェクト種別の対応を次の表に示します。それぞれのメソッドについては，各メソッドの説明を参照してください。

表 1-7 メソッドを実行できる文書管理オブジェクトのオブジェクト種別

メソッド	D	VD	F	IP	PA
DbjObj インターフェース					
addRendition	○	○	×	×	×
bindPublicACL	○	○	○	○	×
cancelCheckOut	×	○	×	×	×
changeMasterRendition	○	○	×	×	×
checkIn	×	○	×	×	×
checkOut	×	○	×	×	×
convertContentIType	○	○	×	×	×
deleteRendition	○	○	×	×	×
deleteVersion	×	○	×	×	×
downloadContents	○	○	×	×	×
getBindObjectList	×	×	×	×	○
getCheckOutStatus	×	○	×	×	×
getChildList	×	×	○	×	×
getClassName	○	○	○	○	○
getDCRParent	○	○	○	×	×
getLockType	○	○	○	○	○
getObjType	○	○	○	○	○
getOiid	○	○	○	○	○
getParentList	○	○	○	×	×
getPublicACLList	○	○	○	○	×
getRelList	○	○	×	×	×
getRenditionList	○	○	×	×	×
getTargetVersion	×	○	×	×	×
getVersionId	○※	○※	○※	○※	○※
getVersioningInfo	○※	○※	○※	○※	○※
getVersionObjList	×	○	×	×	×
link	○	○	○	×	×
lock	○	○	○	○	○
move	○	○	○	×	×
propSet	○	○	○	○	○
readProperties	○	○	○	○	○
removeObject	○	○	○	○	○
setPropSet	○	○	○	○	○
setTargetVersion	×	○	×	×	×
unbindPublicACL	○	○	○	○	×
unlink	○	○	○	×	×
unlinkByLinkId	○	○	○	×	×
uploadContents	○	○	×	×	×

メソッド	D	VD	F	IP	PA
writeProperties	○	○	○	○	○
writeRenditionProperties	○	○	×	×	×
DbjVerObj インターフェース					
getVersionId	○※	○※	○※	○※	○※

（凡例）

D：バージョンなし文書またはバージョン付き文書のバージョンオブジェクト
VD：バージョン付き文書
F：バージョンなしフォルダのバージョンオブジェクト
IP：独立データ
PA：パブリック ACL
○：実行できます。
×：実行できません。または実行しても無効です。

注※　バージョン付きオブジェクトのバージョンオブジェクト以外で実行した場合は，null が返却されます。

なお，DbjObjList インターフェースおよび DbjVerObjList インターフェースで実行できるメソッドは，それぞれの要素である DbjObj インターフェースおよび DbjVerObj インターフェースが対応する文書管理オブジェクトで実行できるメソッドです。

1.10.5　文書管理オブジェクトの情報の取得

ここでは，文書管理オブジェクトの情報を取得する方法について説明します。

文書管理オブジェクトの情報とは，次の情報です。

● OIID

● オブジェクト種別

● DocumentBroker クラス名

(1) OIID の取得

OIID は，DbjObj#getOiid メソッドによって取得します。文書管理オブジェクトの OIID 文字列が取得できます。

OIID を取得する例を示します。

```
// OIIDの取得例
  // obj：DbjObjインターフェース
System.out.println( "OIID=" + obj.getOiid() );
```

(2) オブジェクト種別の取得

オブジェクト種別は，DbjObj#getObjType メソッドによって取得します。

オブジェクト種別は，次に示す DbjDef クラスの定数で表されます。

● バージョンなし文書（DbjDef.OBJTYPE_DOC）

● バージョン付き文書（DbjDef.OBJTYPE_VRDOC）

● バージョンなしフォルダ（DbjDef.OBJTYPE_FOLDER）

● 構成管理できないバージョンなしフォルダ（DbjDef.OBJTYPE_NVTFOLDER）

● 独立データ（DbjDef.OBJTYPE_IP）

● パブリック ACL（DbjDef.OBJTYPE_PUBLICACL）

● すべての種別のオブジェクト（DbjDef.ANY）

● 不明な種別のオブジェクト（DbjDef.UNKNOWN）

オブジェクト種別を取得する例を示します。

// オブジェクト種別の取得例

```
  // obj：DbjObjインターフェース
System.out.println( "ObjectType=" + obj.getObjType() );
```

（3） DocumentBroker クラス名の取得

文書管理オブジェクトを構成している DocumentBroker オブジェクトの DocumentBroker クラス名は，DbjObj#getClassName メソッドによって取得します。

バージョン付きオブジェクトの場合は，バージョニングオブジェクトのトップオブジェクトクラス（dmaClass_ConfigurationHistory クラスまたはそのサブクラス）と，バージョンオブジェクトのトップオブジェクトクラス（dmaClass_DocVersion クラスもしくはそのサブクラス）の二つの DocumentBroker クラス名が取得できます。それ以外の文書管理オブジェクトの場合は，一つの DocumentBroker クラス名が取得できます。

ターゲットオブジェクトの DocumentBroker クラス名を取得する例を示します。

// DocumentBrokerクラス名の取得例

```
  // obj：DbjObjインターフェース
System.out.println( "ClassName=" + obj.getClassName() );
```

1.10.6 アクセス方法に関する情報の取得と変更

ここでは，次に示す情報を取得および変更する操作について説明します。

● ロック種別

● 操作対象になるバージョン

（1） ロック種別の取得と変更

文書管理オブジェクトの操作は，次の 2 種類に分類できます。

参照系の操作

操作対象になる DocumentBroker クラスライブラリのオブジェクトまたは文書管理オブジェクトの状態を変化させない操作です。例えば，文書のコンテンツの参照や文書やフォルダのプロパティの参照などは参照系の操作です。

なお，参照系の操作を実行するメソッドを，**参照系メソッド**といいます。

更新系の操作

操作対象になる DocumentBroker クラスライブラリのオブジェクトまたは文書管理オブジェクトの状態を変化させる操作です。例えば，文書のコンテンツの更新や文書やフォルダのプロパティの更新などは更新系の操作です。

なお，更新系の操作を実行するメソッドを，**更新系メソッド**といいます。

参照系メソッドを実行すると，デフォルトの設定では，文書管理オブジェクトに対して read ロックが設定されます。これを，明示的に write ロックを設定するように変更できます。

文書管理オブジェクトに設定するロック種別の変更は，DbjObj#lock メソッドで実行します。このメソッドを実行すると，同じ文書管理オブジェクトに対して，指定したロック種別を設定する DbjObj インターフェースが取得できます。例えば，write ロックを設定するインターフェースを取得した場合は，そのインターフェースで参照系メソッドを実行したときにも，必ず write ロックが設定されます。

なお，取得するインターフェースは，DbjObj#lock メソッドを実行したインターフェースとは異なるインターフェースです。元のインターフェースを使用した場合のアクセス方法が変更されるのではありません。

異なるロック種別で文書管理オブジェクトにアクセスする例を示します。なお，readProperties メソッドは，文書管理オブジェクトのプロパティを参照する場合に使用する，参照系メソッドです。

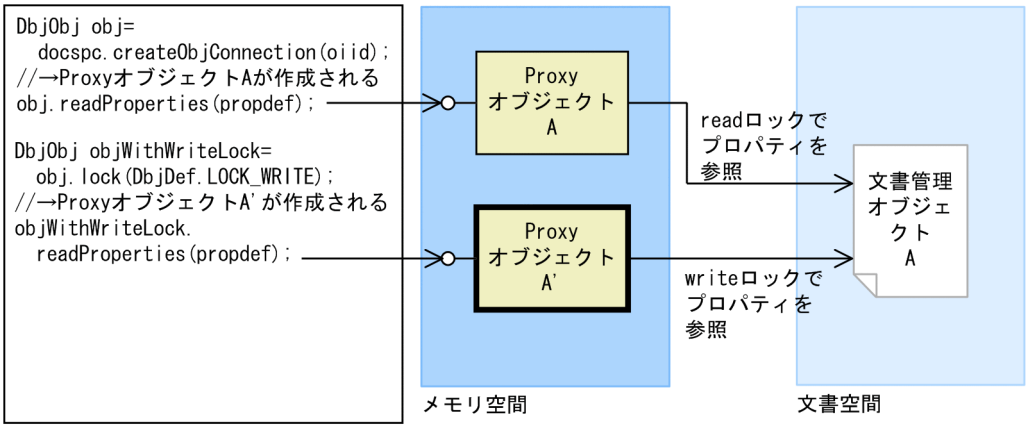
// 異なるロック種別で文書管理オブジェクトにアクセスする例

```
  // docspc：DbjDocSpaceインターフェース

// DbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( oiid );

// デフォルト(=readロックを設定)のままプロパティを参照する…(a)
obj.readProperties( propdef );

// writeロックを設定するためのDbjObjインターフェースを取得する
DbjObj objWithWriteLock = obj.lock( DbjDef.LOCK_WRITE );

// writeロックを設定してプロパティを参照する…(b)
objWithWriteLock.readProperties( propdef );
```

(a) では，文書管理オブジェクトに read ロックが設定されます。(b) では，文書管理オブジェクトに write ロックが設定されます。

この操作で作成される Proxy オブジェクトと，そのインターフェースを使用した操作について，次の図に示します。

図 1-11 異なるロック種別でアクセスする例



（凡例）

—○ ：インターフェース

→ ：制御の流れ

このように，図 1-11 の obj と objWithWriteLock は，同じ文書管理オブジェクトを対象とする，異なる Proxy オブジェクトのインターフェースです。したがって，「obj.lock」を実行したあとでも，obj を使用した場合に設定されるロックの種別は変わりません（この例の場合は read ロックのままです）。

DbjObj インターフェースで参照系メソッドを実行する場合に，そのインターフェースで設定するロック種別を確認するときは，DbjObj#getLockType メソッドを実行します。取得するロック種別の値は，Proxy オブジェクトのアクセスロック種別プロパティに設定されている値です。

(2) 操作対象になるバージョンの取得と変更

バージョン付きオブジェクトを操作する場合，デフォルトの状態では，カレントバージョンのバージョンオブジェクトが操作の対象になります。これを，指定したバージョンのバージョンオブジェクトを操作するように変更できます。例えば，バージョン 3 まであるバージョン付き文書の，バージョン 2 のオブジェクトのコンテンツを参照したい場合などに，対象バージョンを変更します。

対象バージョンの変更は，DbjObj#setTargetVersion メソッドで実行します。このメソッドは，そのインターフェースでのすべての操作の対象を指定したバージョンに変更します。

操作対象のバージョンを変更する例を示します。

// 操作対象のバージョンを変更する例

```
  // docspc：DbjDocSpaceインターフェース

// DbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( oiid );

// デフォルトのバージョン(カレントバージョン)のプロパティを参照する
obj.readProperties( propdef );

// 操作対象のバージョンを，指定バージョン(versionId)に変更する
obj.setTargetVersion( versionId );

// versionIdで指定されたバージョンのプロパティを参照する
obj.readProperties( propdef );

// 操作対象をカレントバージョンに戻す
obj.setTargetVersion( null );
```

DbjObj インターフェースでバージョン付きオブジェクトを操作する場合に，そのインターフェースの操作対象になるバージョンを確認するときには，DbjObj#getTargetVersion メソッドを実行します。このメソッドで取得できるバージョン識別子の値は，Proxy オブジェクトのターゲットバージョン識別子プロパティの値です。

1.10.7 文書管理オブジェクトのプロパティの操作

ここでは，文書管理オブジェクトのプロパティの操作について説明します。

文書管理オブジェクトのプロパティとは，文書管理オブジェクトが持っているプロパティです。DocumentBroker オブジェクト上に値が存在するものと，DocumentBroker オブジェクトの状態を基に演算などによって求められるものがあります。文書管理オブジェクトのプロパティの詳細については，マニュアル「DocumentBroker Version 5　概説」の「文書管理オブジェクトクラスのプロパティ一覧」を参照してください。

文書管理オブジェクトのプロパティの操作は，次の 2 種類の操作に分けられます。

- プロパティの値の参照
- プロパティの値の更新

なお，プロパティ管理のモデルついては，マニュアル「DocumentBroker Version 5　概説」の「属性管理モデル」を参照してください。

(1) プロパティの値の参照

文書管理オブジェクトのプロパティの値は，次の手順で参照します。

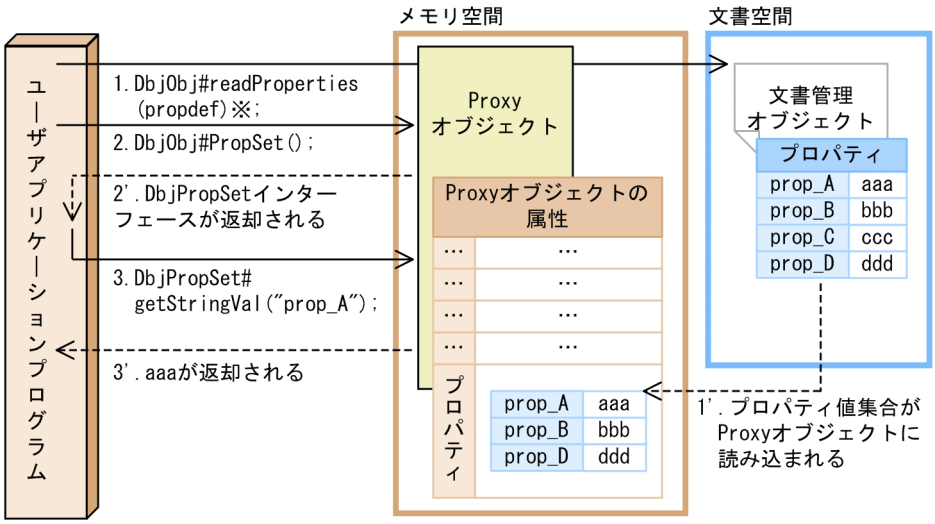
1. **文書管理オブジェクトのプロパティの値を，Proxy オブジェクトのターゲットプロパティ値集合プロパティにロードします。**
   ロードには，DbjObj#readProperties メソッドを使用します。
2. **プロパティ値集合を扱うインターフェース（DbjPropSet インターフェース）を取得します。**
   DbjPropSet インターフェースは，ターゲットプロパティ値集合プロパティにロードした Proxy オブジェクトのプロパティ値集合を扱うインターフェースです。DbjPropSet インターフェースは，DbjObj#propSet メソッドを実行して取得します。
   このインターフェースには，プロパティの値をデータ型ごとに取得するためのメソッド（getIntVal メソッド，getStringVal メソッドなど）が定義されています。取得したい値のデータ型に応じたメソッドを実行します。

なお，文書管理オブジェクトのプロパティの値を，Proxy オブジェクトのターゲットプロパティ値集合プロパティにロードする方法として，次の 2 種類の方法があります。

- ロードするプロパティの名称を指定する方法
  DbjObj#readProperties メソッドの引数に，参照したいプロパティ名のコレクションを指定します。ここで指定したプロパティのプロパティ値集合が，Proxy オブジェクトにロードされます。
- すでに Proxy オブジェクトに読み込んであるプロパティを再度ロードする方法
  DbjObj#readProperties メソッドの引数がない形式を使用します。すでに Proxy オブジェクトのターゲットプロパティ値集合プロパティにロードされているプロパティが，再度ロードされます。ロードされるプロパティの種類は変更されないで，値だけが更新されます。一度トランザクションを終了したあとで，再度プロパティ値を参照する場合に，最新の状態に更新するときなどに使用します。

プロパティの値の参照の流れを，次の図に示します。

図 1-12 プロパティの値の参照の流れ



注※ propdefは，"prop_A"，"prop_B"および"prop_D"を要素としたプロパティのコレクションです。

ここでは，まず，集合に指定したプロパティのプロパティ値集合を，Proxy オブジェクトにロードする例を示します。

// プロパティを参照する例1（取得するプロパティ名の集合を指定する方法）

```
  // docspc：DbjDocSpaceインターフェース

// 参照するプロパティ名を要素とする集合
// (Titleプロパティ とAuthorプロパティ)を作成する
Set<String> propdef = new HashSet<String>();
propdef.add( "Title" );
propdef.add( "Author" );

// DbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( oiid );

// プロパティ値集合をProxyオブジェクトにロードする
obj.readProperties( propdef );

// Proxyオブジェクトのターゲットプロパティ値集合プロパティから
// プロパティ値を取得する
String title = obj.propSet().getStringVal("Title");
String author = obj.propSet().getStringVal("Author");
```

次に，すでに Proxy オブジェクトにロードされているプロパティ値集合を，再度ロードする例を示します。

// プロパティを参照する例2
// （すでにロードされているプロパティ値集合を再度ロードする方法）

```
// 「プロパティを参照する例1」の続き

// プロパティ値集合をProxyオブジェクトに再度ロードする
//(Titleプロパティ とAuthorプロパティの値が更新される)
obj.readProperties();

// Proxyオブジェクトのターゲットプロパティ値集合プロパティから
```

```
// プロパティ値を取得する
String title = obj.propSet().getStringVal("Title");
String author = obj.propSet().getStringVal("Author");
```

(2) プロパティの値の更新

文書管理オブジェクトのプロパティの値は，次の手順で更新します。

設定されているプロパティ値に関係なく，新しいプロパティ値を設定する場合

1. **プロパティ値集合（DbjPropSet）を作成します。**
   DbjFactory#createPropSet メソッドなどで取得した，DbjPropSet インターフェースを使用します。DbjPropSet インターフェースでは，プロパティ値集合に値を設定するためのメソッド（setPropVal メソッドおよび setPropRef メソッド）が定義されています。
2. **プロパティ値集合の値を文書管理オブジェクトにフラッシュします。**
   DbjObj#writeProperties メソッドの，引数にプロパティ値集合（DbjPropSet）を指定できる形式を使用します。

設定されているプロパティ値を基に，新しいプロパティ値に更新する場合

1. **更新したいプロパティの値を，Proxy オブジェクトにロードします。**
   DbjObj#readProperties メソッドを使用します。
2. **Proxy オブジェクトのターゲットプロパティ値集合プロパティの値を更新します。**
   Proxy オブジェクトのターゲットプロパティ値集合プロパティの値の操作には，DbjObj#PropSet メソッドで取得できる DbjPropSet インターフェースを使用します。このインターフェースでは，プロパティ値集合の値を参照するメソッド（getIntVal メソッド，getStringVal メソッドなど）が定義されています。また，プロパティ値集合の値を更新するメソッド（setPropVal メソッドおよび setPropRef メソッド）も定義されています。
3. **Proxy オブジェクトのターゲットプロパティ値集合プロパティの値のうち，特定のプロパティの値だけを文書管理オブジェクトに反映したい場合，反映するプロパティ名のコレクションを作成します。**
   例えば，Proxy オブジェクトにロードしたプロパティのうち，一部だけを文書管理オブジェクトにフラッシュしたい場合などに，そのプロパティを指定します。
4. **プロパティ値集合の値を文書管理オブジェクトにフラッシュします。**
   DbjObj#writeProperties メソッドの，引数のない形式またはプロパティ名のコレクションを指定できる形式を使用します。

ここでは，まず，プロパティ値集合（DbjPropSet）の値を文書管理オブジェクトに設定する例を示します。

// プロパティを更新する例1（プロパティ値集合の値で更新する例）

```
  // docspc：DbjDocSpaceインターフェース

// 更新するプロパティ値集合(Titleプロパティ とAuthorプロパティ)を作成する
DbjPropSet props = factory.createPropSet();
props.setPropVal( "Title", "report" );
props.setPropVal( "Author", "suzuki" );

// DbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( oiid );

// 文書管理オブジェクトのプロパティ値を更新する
obj.writeProperties( props );
```

次に，Proxy オブジェクトのターゲットプロパティ値集合プロパティにロードしたプロパティの値を更新して，文書管理オブジェクトにフラッシュする例を示します。

```
// プロパティを更新する例2（Proxyオブジェクトの値をフラッシュする方法）

  // docspc：DbjDocSpaceインターフェース

// 参照するプロパティ名を要素とする一覧
// (TitleプロパティとAuthorプロパティ)を作成する
Set<String> propdef = new HashSet<String>();
propdef.add( "Title" );
propdef.add( "Author" );

// DbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( oiid );

// プロパティ値集合をProxyオブジェクトにロードする
obj.readProperties( propdef );

// Titleプロパティの値に'#'を追加する
String title = obj.propSet().getStringVal( "Title" ) + "#";
obj.propSet().setPropVal( "Title", title );

// 文書管理オブジェクトに
// TitleプロパティとAuthorプロパティの値をフラッシュする
obj.writeProperties();
```

Proxy オブジェクトのターゲットプロパティ値集合プロパティにロードしたプロパティの値を更新して，一部のプロパティの値だけを文書管理オブジェクトにフラッシュする例を示します。

```
// プロパティを更新する例3
// （Proxyオブジェクトの一部のプロパティの値をフラッシュする方法）

  // docspc：DbjDocSpaceインターフェース

// 参照するプロパティ名を要素とするコレクション
// (TitleプロパティとAuthorプロパティ)を作成する
Set<String> propdef = new HashSet<String>();
propdef.add( "Title" );
propdef.add( "Author" );

// DbjObjインターフェース取得
DbjObj obj = docspc.createObjConnection( oiid );

// プロパティ値集合をProxyオブジェクトにロードする
obj.readProperties( propdef );

// Titleプロパティの値に'#'を追加する
String title = obj.propSet().getStringVal( "Title" ) + "#";
obj.propSet().setPropVal( "Title", title );

// 文書管理オブジェクトに
// Titleプロパティの値だけをフラッシュする
Set<String> updateprop = new HashSet<String>();
updateprop.add( "Title" );
obj.writeProperties( updateprop );
```

(3) バージョン付きオブジェクトのプロパティの操作

バージョン付きオブジェクトのプロパティを操作する場合，バージョニングオブジェクトのプロパティとバージョンオブジェクトのプロパティを同時に操作できます。

バージョニングオブジェクトのプロパティとバージョンオブジェクトのプロパティは，データベース上では，次に示すように異なる DocumentBroker オブジェクト上に定義されています。

バージョニングオブジェクトのプロパティ

dmaClass_ConfigurationHistory クラスまたはそのサブクラスのオブジェクトに定義されています。

バージョンオブジェクトのプロパティ

- バージョン付き文書の場合は，dmaClass_DocVersion クラスまたはそのサブクラスのオブジェクトに定義されています。

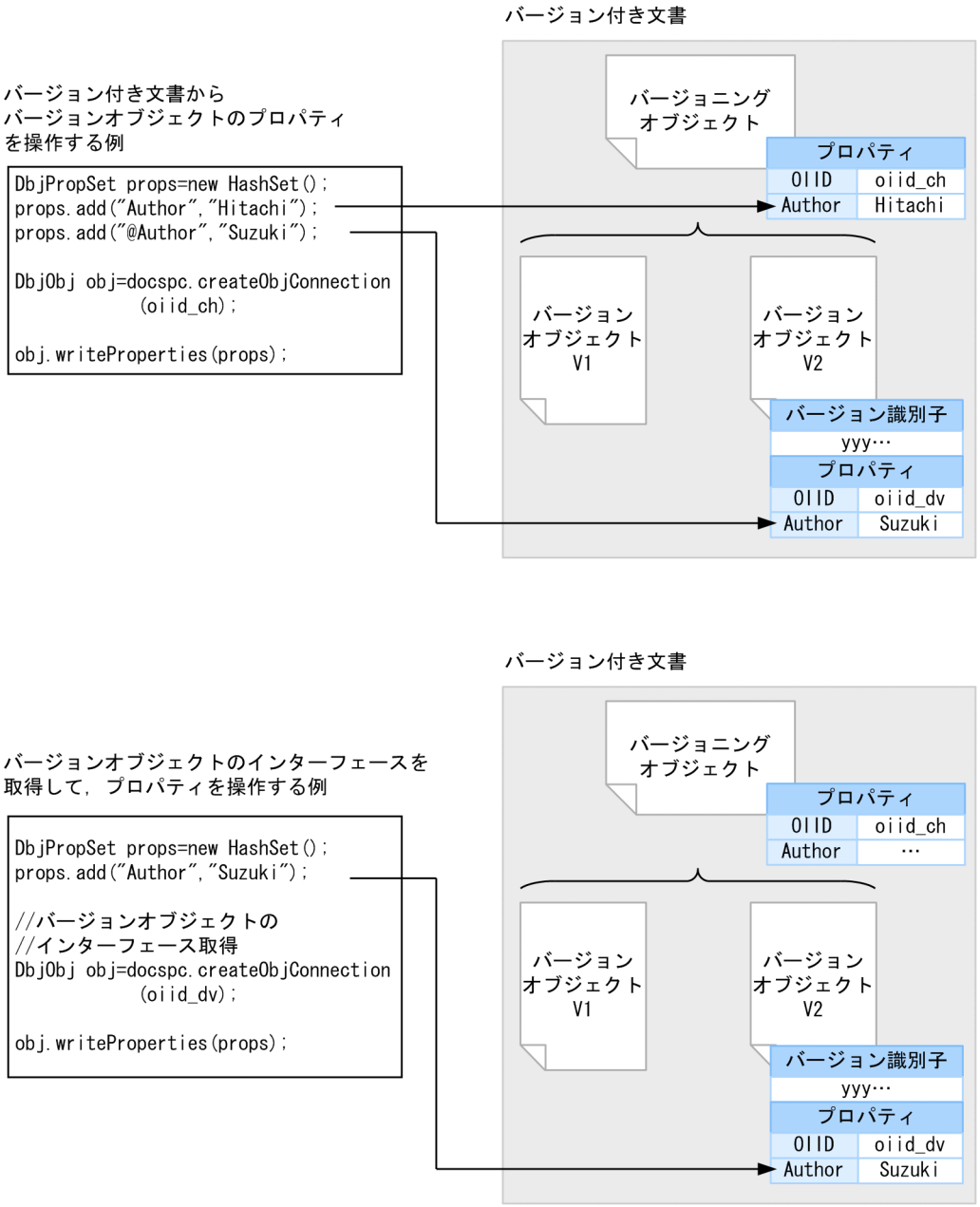
バージョニングオブジェクトのプロパティとバージョンオブジェクトのプロパティは，バージョンオブジェクトのプロパティ名に **@ プレフィックス**を付けて指定することで，明示的に区別して指定します。

例えば，バージョン付き文書のバージョニングオブジェクトに文書全体の著者名を表す Author プロパティが定義されていて，バージョンオブジェクトにもバージョンごとの著者名を表す Author プロパティが定義されている場合，バージョンオブジェクトの Author プロパティは，"@Author" として指定します。これは，Proxy オブジェクトのターゲットプロパティ値集合プロパティにロードしたプロパティ値集合を扱う場合でも同じです。バージョニングオブジェクトとバージョンオブジェクトのプロパティを同時に操作する場合は，バージョンオブジェクトのプロパティ名に「@」を付けて区別します。

ただし，バージョン付きオブジェクトのプロパティを扱う場合でも，バージョンオブジェクトの DbjObj インターフェースを取得して操作する場合には，「@」は付けません。例えば，バージョン付きオブジェクトから直接カレントバージョンのプロパティを取得する場合は「@」を付けますが，バージョン付きオブジェクトからカレントバージョンの DbjObj インターフェースを取得して，そのインターフェースを使用してバージョンオブジェクトのプロパティを取得する場合は「@」を付けません。

バージョン付きオブジェクトのプロパティの操作を次の図に示します。

図 1-13　バージョン付きオブジェクトのプロパティの操作



ここでは，バージョニングオブジェクトの Author プロパティとバージョンオブジェクトの Author プロパティを同時に取得して参照する例を示します。

// バージョン付きオブジェクトのプロパティを参照する例

```
  //docspc：DbjDocSpaceインターフェース

// 参照するプロパティ名を要素とするコレクションを作成する
Set<String> propdef = new HashSet<String>();
propdef.add( "Author" );
propdef.add( "@Author" );

// バージョニングオブジェクトのDbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( oiid );
```

```
// プロパティ値集合をProxyオブジェクトにロードする
obj.readProperties( propdef );

// バージョニングオブジェクトのAuthorプロパティを取得する
String authorOfVersioning = obj.propSet().getStringVal("Author");

// バージョンオブジェクトのAuthorプロパティを取得する
String authorOfVersion = obj.propSet().getStringVal("@Author");
```

(4) VARRAY 型のプロパティの参照

VARRAY 型のプロパティは，可変長配列を値とするプロパティです。可変長配列は，パラメタクラスの DbjVArray インターフェースを使用して操作します。

VARRAY 型のプロパティの値は，次のどちらかの方法で取得できます。

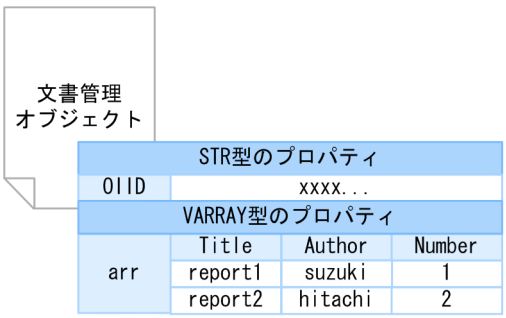
- Proxy オブジェクトに設定されているプロパティ値集合から DbjPropSet インターフェースを使用して取得する
- 検索結果集合から DbjResultSet インターフェースを使用して取得する

これらのインターフェースを使用して，VARRAY 型のプロパティを参照する手順は，次のとおりです

1. **Proxy オブジェクトに文書管理オブジェクトのプロパティをロードして，DbjPropSet インターフェースを取得します。または，検索を実行して，検索結果集合の DbjResultSet インターフェースを取得します。**
2. **DbjPropSet インターフェースまたは DbjResultSet インターフェースから DbjVArray インターフェースを取得します。**
   - DbjPropSet#getVArrayRef メソッドまたは DbjResultSet#getVArrayRef メソッドを使用した場合は，VARRAY 型のプロパティに設定された可変長配列の参照を取得できます。
   - DbjPropSet#getVArrayVal メソッドまたは DbjResultSet#getVArrayVal メソッドを使用した場合は，VARRAY 型のプロパティに設定された可変長配列のコピーを取得できます。
3. **DbjVArray インターフェースのメソッドで値を取得します。**

ここでは，DbjPropSet インターフェースを使用して VARRAY 型のプロパティを参照する例を示します。この例では，次の図に示す文書管理オブジェクトの VARRAY 型のプロパティを対象にします。

図 1-14　例で使用する VARRAY 型のプロパティ




```
// VARRAY型のプロパティを参照する例

  //docspc：DbjDocSpaceインターフェース

Set propdef = new HashSet();
propdef.add( "arr" );
```

```
// VARRAY型のプロパティの値をロードする
DbjObj obj = docspc.createObjConnection( oiid );
obj.readProperties( propdef );

// Proxyオブジェクトのターゲットプロパティ値集合プロパティから,
// DbjVArrayインターフェースの参照を取得する
DbjVArray varray = obj.propSet().getVArrayRef( "arr" );

// VARRAY型のプロパティの各要素の値を出力する
for (int i=0; i<varray.size();i++){
  System.out.println("Title["+i+"]="
                  varray.propSet(i).getStringVal( "Title" ) );
  System.out.println("Author["+i+"]="
                  varray.propSet(i).getStringVal( "Author" ) );
}
```

(5) VARRAY 型のプロパティの更新

VARRAY 型のプロパティは，次のような手順で更新します。

1. **可変長配列の要素に指定するプロパティ名のコレクションを作成します。**
2. **可変長配列を作成します。**
3. **可変長配列の要素を作成します。**
4. **要素に値を設定します。**
5. 4. **を可変長配列の要素として追加します。**
   4. と 5. は要素の数だけ繰り返します。
6. **可変長配列を，**VARRAY **型のプロパティの値としてプロパティ値集合に設定します。**
7. **プロパティ値集合を，文書管理オブジェクトにフラッシュします。**

VARRAY 型のプロパティを更新する例を示します。なお，対象とする VARRAY 型のプロパティは，図 1-14 と同じです。

// VARRAY型のプロパティを更新する例

```
  // factory：DbjFactoryインターフェース
  // docspc：DbjDocSpaceインターフェース
  // obj：DbjObjインターフェース

// 可変長配列の要素のプロパティ名のコレクションを作成する
Set<String> propdef = new HashSet<String>();
propdef.add( "Title" );
propdef.add( "Author" );
propdef.add( "Number" );

// 可変長配列を作成する
DbjVArray varray = factory.createVArray( propdef );

// 可変長配列の要素を作成する
DbjPropSet elm = factory.createPropSet();

// 一つ目の要素に値を設定する
elm.setPropVal( "Title" , "report1" );
elm.setPropVal( "Author" , "suzuki" );

// 可変長配列に一つ目の要素を追加する
varray.addPropSet( elm );

// 二つ目の要素に値を設定する
elm.setPropVal( "Title" , "report2" );
elm.setPropVal( "Author" , "hitachi" );
```

```
// 可変長配列に二つ目の要素を追加する
varray.addPropSet( elm );

// Numberプロパティの値を要素ごとに設定する
for(int i = 0; i < varray.size(); i ++ ) {
    // 可変長配列の要素を参照してNumberプロパティの値を設定する
    varray.propSet(i).setPropVal( "Number", i );
}

// プロパティ値集合を作成する
// （arrプロパティに可変長配列の値を設定する）
DbjPropSet props = factory.createPropSet();
props.setPropVal( "arr" , varray );

// 文書管理オブジェクトにプロパティ値集合をフラッシュする
obj.writeProperties( props );
```

(6) そのほかのオブジェクトのプロパティの操作

レンディションのプロパティの操作，およびリンクオブジェクトのプロパティの操作は，この項で説明したプロパティの操作とは異なるインターフェースおよびメソッドを使用して実行します。

レンディションのプロパティの操作については，「1.10.11　マルチレンディション文書のレンディションの操作」を参照してください。

リンクオブジェクトのプロパティの操作については，「1.10.13　リンクの操作」を参照してください。

1.10.8　文書管理オブジェクトの削除

ここでは，文書管理オブジェクトの削除について説明します。

文書管理オブジェクトは，DbjObj#removeObject メソッドで削除します。正常に削除された場合は，true が返却されます。すでに削除されている文書管理オブジェクトを削除しようとした場合は，例外ではなく false が返却されます。それ以外の原因で文書管理オブジェクトが削除されなかった場合は，例外がスローされます。

文書管理オブジェクトを削除する例を示します。

// 文書管理オブジェクトを削除する例

```
  // docspc：DbjDocSpaceインターフェース

// DbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( oiid );

// 文書管理オブジェクトを削除する
if ( !obj.removeObject() ) {
  // falseが返却された場合，すでに削除されている旨を出力する
  System.out.println("Object has been already removed.");
}
```

1.10.9　文書のコンテンツの操作

ここでは，文書のコンテンツの操作について説明します。

なお，バージョン付き文書のコンテンツを操作する場合，デフォルトの設定では，カレントバージョンのコンテンツが対象になります。操作対象のバージョンを変更する場合は，DbjObj#setTargetVersion メソッドを実行して，ターゲットバージョンを変更してください。

文書のコンテンツ管理の考え方については，マニュアル「DocumentBroker Version 5　概説」の「レン

ディション管理モデル」の説明を参照してください。

(1) コンテンツのダウンロード

ここでは，文書のコンテンツをローカルパスのファイルにダウンロードして，参照する操作について説明します。

文書のコンテンツは，DbjObj#downloadContents メソッドでダウンロードします。
DbjObj#downloadContents メソッドの引数に指定するファイルパスには，ローカルパスを指定します。

手順を示します。

1. **DbjObj インターフェースを取得します。**
   コンテンツをダウンロードする文書管理オブジェクトの oiid を指定して，
   DbjDocSpace#createObjConnection メソッドを実行します。
2. **コンテンツをダウンロードします。**
   DbjObj#downloadContents メソッドを実行します。メソッド実行時には，次の情報を指定します。
   - コンテンツをダウンロードするレンディションのレンディションタイプ
   - ダウンロード先のファイル

マルチレンディション文書の場合，レンディションタイプを指定してダウンロードするコンテンツを特定します。レンディションタイプに null を指定した場合は，マスタレンディションのコンテンツがダウンロードできます。ダウンロードしたコンテンツは，ダウンロード先ファイルとして指定したパスに，指定したファイル名でコピーされます。指定するファイル名は任意です。文書管理オブジェクトの dbrProp_RetrievalName プロパティの値と同じにする必要はありません。

コンテンツをダウンロードする例を示します。この例では，マスタレンディションのコンテンツを「test.doc」にダウンロードします。

// コンテンツをダウンロードする例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// ダウンロードするパスを指定する（ファイル名を含む）
File dwlFile1 = new File( parentdir, "test.doc" );

// マスタレンディションのコンテンツを
// dwlFilePathで指定したパスにダウンロードする
DbjContentInfo  continfo1 = obj.downloadContents(
                            null,
                            dwlFile1.getCanonicalPath() );
```

(2) コンテンツのアップロード

ローカルパス上のファイルの内容をアップロードして，文書のコンテンツを更新する操作について説明します。

文書のコンテンツは，DbjObj#uploadContents メソッドでアップロードします。DbjObj#uploadContents メソッドの引数に指定するファイルパスには，ローカルパスを指定します。

手順を示します。

1. **文書のアップロード情報を作成します。**
   文書のアップロード情報は，DbjUploadInfo インターフェースで表します。
   DbjUploadInfo インターフェースを使用する場合，次の情報を指定します。

- アップロードするファイルのファイルパス
- アップロードするファイルのファイル名
- 文書に設定するレンディションタイプ
- レンディションのプロパティ
- 全文検索インデクス作成用のファイルパス
  全文検索インデクス作成用の指定は，マスタレンディションのコンテンツを更新する場合にだけ有効です。

2. **コンテンツをアップロードします。**
   DbjObj#uploadContents メソッドを実行します。メソッド実行時には，次の情報を指定します。
   - アップロードの対象にするレンディションタイプ
   - 文書のアップロード情報

コンテンツをアップロードする時に，同時に全文検索インデクスを登録することもできます。全文検索インデクスは，次のどちらかの方法で作成できます。

● アップロードするコンテンツから自動作成して登録する（特定のレンディションタイプのファイルをコンテンツに登録する場合）

● アップロードするコンテンツとは別に，全文検索インデクス作成用のファイルを指定して登録する

全文検索インデクスの作成方法については，マニュアル「DocumentBroker Version 5　概説」を参照してください。

なお，マルチレンディション文書の場合，全文検索インデクスは，レンディションタイプに null を指定し，マスタレンディションに対してコンテンツをアップロードする場合にだけ作成されます。サブレンディションのコンテンツをアップロードする場合には，全文検索インデクス作成の指定は無視されます。マルチレンディション文書のコンテンツのアップロードについては，「1.10.11　マルチレンディション文書のレンディションの操作」を参照してください。

文書のアップロード情報に指定するレンディションタイプと，DbjObj#uploadContents メソッドの引数に指定するレンディションタイプが異なる場合は，コンテンツをアップロードしたレンディションのレンディションタイプが，文書のアップロード情報に指定したレンディションタイプに変更されます。なお，マルチレンディション文書の場合に，マスタレンディションにコンテンツが登録されていないとき，または DbjObj#uploadContents メソッドの引数のレンディションタイプに null を指定したときは，マスタレンディションのコンテンツが更新されます。

コンテンツをアップロードする例を示します。この例では，「tmp.html」というファイルを登録します。文書管理オブジェクトの dbrProp_RetrievalName プロパティは，「test.html」と設定します。また，登録するコンテンツから全文検索インデクスを作成します。

// コンテンツをアップロードする例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// アップロードするコンテンツのファイルパスを指定する
File uplFile = new File( parentdir, "tmp.html" );

// 文書のアップロード情報を作成する
// （レンディションタイプは拡張子から自動設定するようにする）
DbjUploadInfo upinfo = factory.createUploadInfo(
                uplFile.getCanonicalPath(),
                "test.html",    // retrievalNameプロパティを指定
                null,           // レンディションタイプを指定
                                // （ここでは拡張子（.html）から決定する）
                null,
```

```
                DbjDef.INDEXPATH_SAME );
                                // 全文検索インデクス作成

// マスタレンディションにコンテンツをアップロードする
obj.uploadContents(
                null,           // マスタレンディションが対象
                upinfo );       // 文書のアップロード情報を指定
```

1.10.10 バージョン付きオブジェクトのバージョン操作

ここでは，バージョン付きオブジェクトのバージョン操作について説明します。

次に示す操作は，バージョニングオブジェクトのインターフェースを使用して実行します。

- バージョンのチェックアウト
- バージョンのチェックイン
- バージョンのチェックアウトの取り消し
- バージョンオブジェクトの一覧取得
- バージョンの削除

なお，チェックアウト，チェックインまたはチェックアウトの取り消しと同時に，仮のバージョンオブジェクトまたはカレントバージョンのバージョンオブジェクトにプロパティを設定できます。

次の操作は，バージョンオブジェクトのインターフェースを使用して実行します。

- バージョン識別子の取得
- バージョンオブジェクトからバージョニングオブジェクトの取得

バージョン管理の考え方については，マニュアル「DocumentBroker Version 5　概説」の「バージョン管理モデル」を参照してください。

(1) チェックアウト

チェックアウトは，バージョニングオブジェクトのインターフェースを使用して，DbjObj#checkOut メソッドで実行します。

また，チェックアウトをする時に，チェックアウト中のバージョニングオブジェクトおよび仮のバージョンオブジェクトに対して，プロパティを設定できます。なお，チェックアウト後のターゲットバージョンは，仮のバージョンになります。

チェックアウトの例を示します。この例では，チェックアウトしたバージョニングオブジェクトおよび仮のバージョンオブジェクトに対して，次のプロパティを設定します。

- バージョニングオブジェクトの CheckOutFlag プロパティに，チェックアウト中であることを示す値「1」を設定する。
- 仮のバージョンオブジェクトの VerCount プロパティに，バージョン番号を示す値「チェックアウト前のバージョン番号 +1」を設定する。

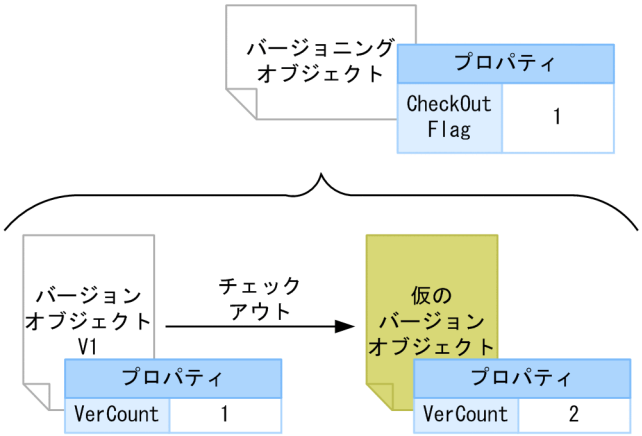
// チェックアウトの例

```
  // obj：DbjObjインターフェース

// Proxyオブジェクトに取得するプロパティ名を設定する
Set<String> propdef = new HashSet<String>();
propdef.add( "CheckOutFlag" );
propdef.add( "@VerCount" );
```

```
// カレントバージョンのプロパティをProxyオブジェクトにロードする
obj.readProperties( propdef );

// プロパティの値を更新する
obj.propSet().setPropVal("@VerCount",
        obj.propSet().getIntVal("@VerCount") + 1 );
obj.propSet().setPropVal("CheckOutFlag", 1 );

// チェックアウトと同時にProxyオブジェクトのプロパティを
// バージョニングオブジェクトおよび
// 仮のバージョンオブジェクトのプロパティにフラッシュする
obj.checkOut();
```

チェックアウト後のプロパティの値は，次の図に示すようになります。

図 1-15　チェックアウト後のプロパティの値



(2) チェックイン

チェックアウトしたバージョン付きオブジェクトは，バージョニングオブジェクトのインターフェースを使用して，DbjObj#checkIn メソッドでチェックインします。

また，チェックインする時に，チェックイン後のカレントバージョンオブジェクトに対して，プロパティを設定できます。なお，チェックイン後のターゲットバージョンは，カレントバージョンになります。

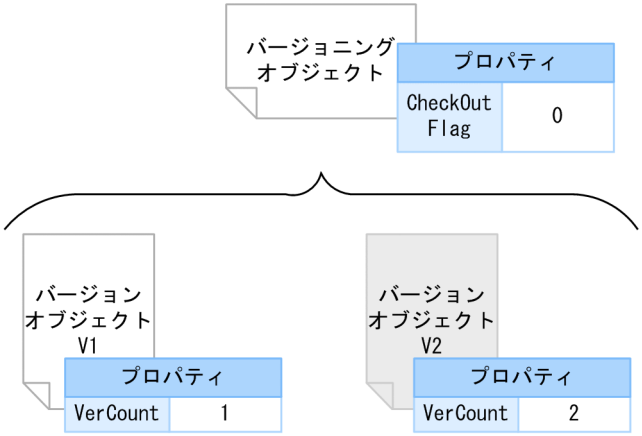
チェックインの例を示します。この例では，(1) でチェックアウトしたバージョン付きオブジェクトをチェックインします。同時に，バージョニングオブジェクトの CheckOutFlag プロパティの値を，チェックアウト中ではないことを示す値「0」に更新します。

// チェックインの例

```
  // obj : DbjObjインターフェース

// ProxyオブジェクトのCheckOutFlagプロパティの値を設定する
obj.propSet().setPropVal( "CheckOutFlag", 0 );

// チェックインと同時にプロパティの値をフラッシュする
obj.checkIn();
```

チェックイン後のプロパティの値は，次のようになります。

図 1-16 チェックイン後のプロパティの値



(3) チェックアウトの取り消し

チェックアウトを取り消す場合は，バージョニングオブジェクトのインターフェースを使用して，DbjObj#cancelCheckOut メソッドで取り消します。

チェックアウトを取り消す場合，取り消したあとのカレントバージョンオブジェクトに対してプロパティを設定できます。なお，チェックアウト時にバージョニングオブジェクトのプロパティを設定した場合，チェックアウトを取り消した時に元のプロパティの値に戻したい場合は，明示的にチェックアウト前の値でプロパティを更新し直す必要があります。また，チェックイン後のターゲットバージョンは，チェックアウト前のカレントバージョンになります。

チェックアウトの取り消しの例を示します。この例では，(1) でチェックアウトしたバージョン付きオブジェクトのチェックアウトを取り消します。同時に，チェックアウトした時に設定したプロパティの値も元に戻すように更新します。

//チェックアウトの取り消しの例

```
  // obj：DbjObjインターフェース

// ProxyオブジェクトのCheckOutFlagプロパティに
// チェックアウト前の値を設定する
obj.propSet().setPropVal( "CheckOutFlag", 0 );

// チェックアウトの取り消しと同時にプロパティ値も元に戻す
obj.cancelCheckOut();
```

チェックアウト取り消し後には，バージョニングオブジェクトの CheckOutFlag プロパティの値は「0」に，カレントバージョンオブジェクトの VerCount プロパティの値は「1」になります。

(4) バージョンオブジェクトの一覧の取得

バージョン付きオブジェクトに含まれるバージョンオブジェクトの一覧は，DbjObj#getVersionObjList メソッドで取得します。また，バージョンオブジェクトの一覧を取得する時に，任意のプロパティの値を同時に取得することもできます。

バージョンオブジェクトの一覧を取得する例を示します。この例では，バージョンオブジェクトのうち，最も古いバージョンの Counter プロパティを取得して，値を更新します。

// バージョンオブジェクトの一覧を取得する例

```
  // obj : DbjObjインターフェース

// 取得するプロパティ名を設定する
Set<String> propdef = new HashSet<String>();
propdef.add( "Counter" );

// バージョン一覧を昇順(古いものから順番)に取得する
DbjVerObjList verlst = obj.getVersionObjList(
                          propdef,
                          DbjDef.ORDER_ASC,
                          null );

// 最も古いバージョンオブジェクトのインターフェースを取得する
DbjVerObj oldestVersion = verlst.getVerObj(0);

// 最も古いバージョンオブジェクトのCounterプロパティの値を1増やす
int counter = oldestVersion.propSet().getIntVal( "Counter" );
counter ++;
oldestVersion.propSet().setPropVal( "Counter", counter );

// Proxyオブジェクトのプロパティの値を文書管理オブジェクトに
// フラッシュする
oldestVersion.writeProperties();
```

(5) バージョンの削除

バージョン付きオブジェクトの特定のバージョンを削除する場合は，DbjObj#deleteVersion メソッドを実行します。削除するバージョンは，メソッドの引数にバージョン識別子を指定することによって特定します。複数のバージョン識別子をリストとして指定すると，複数のバージョンを一括して削除できます。ただし，すべてのバージョンを削除することはできません。少なくとも一つのバージョンは残しておく必要があります。

なお，DbjObj#deleteVersion メソッドは，バージョニングオブジェクトのインターフェースで実行します。この操作の結果は，削除するバージョンオブジェクトのインターフェースを使用して，DbjObj#removeObject メソッドを実行した場合と同じです。

バージョンを削除する例を示します。この例では，最新バージョン以外のバージョンを削除します。

// バージョンを削除する例

```
  // obj：DbjObjインターフェース

// バージョン一覧を降順(新しいものから順番)に取得する
DbjVerObjList verlst = obj.getVersionObjList(
               null,
               DbjDef.ORDER_DESC,
               null );

// 削除するバージョン識別子のリストを作成する
// リストから，最新バージョンのバージョン識別子だけを削除する
List<String> delVersionList = verlst.getVersionIdList();
delVersionList.removeObjects(0);

// 一覧に指定したバージョン識別子のバージョンを一括削除する
obj.deleteVersion( delVersionList );
```

(6) バージョニングオブジェクトのインターフェースの取得

バージョンオブジェクトのインターフェースからバージョニングオブジェクトのインターフェースを取得するには，DbjObj#getVersioningInfo メソッドを実行します。

バージョニングオブジェクトのインターフェースを取得する例を示します。この例では，バージョンオブジェクトのインターフェースからバージョニングオブジェクトのインターフェースを取得します。その後，バージョニングオブジェクトでまとめているバージョンオブジェクトの一覧を取得します。

// バージョニングオブジェクトのインターフェースを取得する例

```
  // obj：DbjObjインターフェース
  // （バージョンオブジェクトのインターフェース）

// バージョンオブジェクトのインターフェースからバージョニングオブジェクトの
// インターフェースを取得する
DbjObj versioningObj = obj.getVersioningInfo();

// バージョニングオブジェクトのバージョン一覧を
// 降順(新しいものから順番)で取得する
DbjVerObjList verlst = versioningObj.getVersionObjList(
              null,
              DbjDef.ORDER_DESC,
              null );
```

(7) バージョン識別子の取得

ここでは，バージョン識別子の取得について説明します。バージョン識別子は，次のどちらかのメソッドで取得します。

● DbjObj#getVersionId メソッド

● DbjVerObj#getVersionId メソッド

DbjVerObj インターフェースは，ターゲットオブジェクトのバージョン識別子をプロパティに持っています。DbjVerObj#getVersionId メソッドでは，このプロパティの値を取得します。

DbjObj インターフェースを使用して，バージョン識別子を取得する例を示します。この例では，まず，バージョンオブジェクトのインターフェースでバージョン識別子を取得し，そのあとでバージョニングオブジェクトのインターフェースで取得したバージョン識別子を持つバージョンオブジェクトを削除します。

// バージョン識別子を取得する例

```
  // obj：DbjObjインターフェース
  // （バージョンオブジェクトのインターフェース）

// バージョン識別子を取得する
String versionId = obj.getVersionId();

// バージョニングオブジェクトを取得する
DbjObj versioningObj = obj.getVersioningInfo();

// バージョン識別子を指定してバージョンを削除する
List<String> dels = new ArrayList<String>();
dels.add( versionId );
versioningObj.deleteVersion( dels );
```

1.10.11 マルチレンディション文書のレンディションの操作

ここでは，マルチレンディション文書に対する次の操作について説明します。

● マルチレンディション文書の作成

● レンディションの追加

● レンディションの削除

● レンディションの変更

● レンディションのプロパティの操作

なお，バージョン付き文書のレンディションを操作する場合，デフォルトの設定では，カレントバージョ

ンのレンディションが対象になります。操作対象のバージョンを変更する場合は，DbjObj#setTargetVersion メソッドを実行して，ターゲットバージョンを変更してください。

マルチレンディション文書の管理の考え方については，マニュアル「DocumentBroker Version 5　概説」の「レンディション管理モデル」を参照してください。

(1) マルチレンディション文書の作成

マルチレンディション文書は，次のどちらかの方法で，文書に複数のレンディションを登録して作成します。

● 文書作成時に，複数のレンディションを同時に登録する

● 既存の文書に対して，レンディションを追加する

ここでは，文書作成時に，複数のレンディションを登録することで，マルチレンディション文書を作成する方法について説明します。

文書の作成時に複数のレンディションを登録する場合，まず，複数の文書のアップロード情報（DbjUploadInfo インターフェース）を要素にしたリストを作成します。リストには，マスタレンディションにする文書のアップロード情報を，先頭の要素として指定してください。このリストを，DbjDocSpace#createDocument メソッドまたは DbjDocSpace#createVrDocument メソッドの引数に指定して，メソッドを実行します。

リストは，次の点に注意して作成してください。

● 全文検索インデクス作成の指定は，リストの先頭に指定した文書のアップロード情報だけで有効です。

マルチレンディション文書を作成する例を示します。

// マルチレンディション文書を作成する例

```
  //factory：DbjFactoryインターフェース
  //docspc：DbjDocSpaceインターフェース

File uplFilePath_doc = new File( parentdir, "test.doc" );

//文書のアップロード情報のリストを作成する（要素は二つ）
DbjUploadInfo upinfo_doc = factory.createUploadInfo(
                      uplFilePath_doc.getCanonicalPath(),
                      originalFileName_doc,
                      null,
                      null,
                      null );
DbjUploadInfo upinfo_pdf = factory.createUploadInfo(
                      null,
                      originalFileName_pdf,
                      null,
                      null,
                      null ); // 全文検索インデクス指定は無効

// リストを作成する
List<DbjUploadInfo> uplist = new ArrayList<DbjUploadInfo>();

// 文書のアップロード情報をリストに追加する
uplist.add( upinfo_doc ); // マスタレンディションになる
uplist.add( upinfo_pdf );

// 文書を作成する
// （test.docとtest.pdfが初期登録される）
DbjObj obj = docspc.createDocument(
             "mdmClass_Document",
             null,
             uplist,// 文書のアップロード情報のリスト
```

```
                null );
```

(2) レンディションの追加

既存の文書に対してレンディションを追加する場合，DbjObj#addRendition メソッドを実行します。複数の文書のアップロード情報を要素にしたリストをメソッドの引数に指定すれば，複数のレンディションを同時に追加できます。

なお，レンディション追加時に，次の機能は使用できません。

- 全文検索インデクスは作成できません。文書のアップロード情報に全文検索インデクス作成の指定をしても，無効です。

サブレンディションを追加する例を示します。

// サブレンディションを追加する例

```
  // docspc：DbjDocSpaceインターフェース
  // obj：DbjObjインターフェース

// 文書のアップロード情報を作成する
DbjUploadInfo upinfo_pdf = factory.createUploadInfo(
                      null,
                      originalFileName_pdf,
                      null,
                      null,
                      null ); // 全文検索インデクス指定は無効

// リストを作成する
List<DbjUploadInfo> uplist = new ArrayList<DbjUploadInfo>();

// 文書のアップロード情報をリストに追加する
uplist.add( upinfo_pdf );

// 既存の文書にレンディションを追加する
// (pdf形式のレンディションが追加される)
obj.addRendition( uplist );
```

(3) レンディションの削除

レンディションは，DbjObj#deleteRendition メソッドで削除できます。

メソッドの引数には，レンディションタイプを表す文字列を要素とするリストを指定します。要素のレンディションタイプを複数にすると，複数のレンディションを一括して削除できます。

複数のレンディションを一括して削除する例を示します。

// 複数のレンディションを一括して削除する例

```
  // obj：DbjObjインターフェース

// 削除対象のレンディションタイプを指定したリストを作成する
List<String> renlist = new ArrayList<String>();
renlist.add( "text/plain" );
renlist.add( "text/html" );

// レンディションを一括して削除する
obj.deleteRendition( renlist );
```

(4) マスタレンディションの変更

マスタレンディションは，DbjObj#changeMasterRendition メソッドで変更できます。

マスタレンディションを変更する例を示します。この例では，レンディションタイプが「text/html」のレ

ンディションを，マスタレンディションに変更します。

// マスタレンディションの変更例

```
  // obj：DbjObjインターフェース
obj.changeMasterRendition( "text/html" );
```

(5) レンディションを指定したコンテンツの更新

DbjObj#uploadContents メソッド実行時に，レンディションを指定してコンテンツを更新できます。

なお，レンディションタイプに null を指定してマスタレンディションのコンテンツを更新する時以外は，全文検索インデクスは更新できません。

レンディションを指定してコンテンツを更新する例を示します。

// レンディションを指定してコンテンツを更新する例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

//アップロードするコンテンツのファイルパスを指定する
File uplFile_txt = new File( parentdir, "test.txt" );
File uplFile_doc = new File( parentdir, "test.doc" );

// マスタレンディションの文書のアップロード情報を作成する
DbjUploadInfo upinfo_txt = factory.createUploadInfo(
                   uplFile_txt.getCanonicalPath(),
                   "test.txt",    // retrievalNameプロパティを指定
                   "text/plain",  // レンディションタイプを指定
                   null,
                   DbjDef.INDEXPATH_SAME );
                                  // マスタレンディションを更新する
                                  // 場合だけ有効になる

// サブレンディションの文書のアップロード情報を作成する
DbjUploadInfo upinfo_doc = factory.createUploadInfo(
                   uplFile_doc.getCanonicalPath(),
                   "test.doc",    // retrievalNameプロパティを指定
                   "application/ms-word",
                                  // レンディションタイプを指定
                   null,
                   null );

// マスタレンディション（text/plain）にコンテンツをアップロードする
//  （マスタレンディションはtext/plainであるとする）
obj.uploadContents(
            null,                  // マスタレンディションを指定
            upinfo_txt );          // 文書のアップロード情報を指定

// サブレンディション（application/ms-word）にコンテンツを
// アップロードする
obj.uploadContents(
            "application/ms-word", // レンディションタイプを指定
            upinfo_doc );          //文書のアップロード情報を指定
```

(6) レンディションタイプの変更

登録済みのレンディションタイプは，DbjObj#uploadContents メソッドを実行してコンテンツをアップロードする時に変更できます。変更する場合は，コンテンツをアップロードする時に，メソッドの引数に指定するレンディションタイプと，文書のアップロード情報に指定するレンディションタイプに，異なるレンディションタイプを指定します。

それぞれに指定するレンディションタイプは，次のとおりです。

メソッドの引数に指定するレンディションタイプ

コンテンツをアップロードする対象になるレンディションのレンディションタイプを指定します。例えば，「application/ms-word」というレンディションタイプを指定した場合は，このレンディションタイプを持つマスタレンディションまたはサブレンディションのコンテンツが，更新する対象になります。ただし，マスタレンディションにコンテンツが登録されていない場合，またはレンディションタイプとして null を指定した場合は，マスタレンディションのコンテンツが対象になります。

文書のアップロード情報に指定するレンディションタイプ

登録するコンテンツに設定するレンディションタイプを指定します。null を指定した場合は，登録するファイルの拡張子とレンディション定義ファイルの内容に従って，レンディションタイプが設定されます。

メソッドの引数に指定したレンディションタイプと文書のアップロード情報に指定したレンディションタイプが異なる場合，更新するレンディションのレンディションタイプを，文書のアップロード情報に指定したレンディションタイプで変更することになります。例えば，

```
DbjUploadInfo upinfo = factory.createUploadInfo(
            uplfilePath,
            "test.html"
            "text/html",
            null,
            null );
obj.uploadContents(
            "text/plain",
            upinfo );
```

という指定をした場合は，「text/plain」というレンディションタイプのレンディションのコンテンツにファイル「test.html」を登録して，同時にレンディションタイプを「text/html」に変更することになります。ただし，文書のアップロード情報のレンディションタイプに，すでにほかのレンディションのレンディションタイプとして登録されているレンディションタイプを指定した場合は，エラーになり，例外がスローされます。

なお，マスタレンディションのレンディションタイプ変更時以外は，全文検索インデクスを作成できません。

レンディションタイプを変更する例を示します。ここでは，「text/plain」と登録していたレンディションタイプを，「text/html」に変更します。

// レンディションタイプを変更する例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// アップロードするコンテンツのパスを設定する
File uplFilePath = new File( parentdir, "test.html" );

// 文書のアップロード情報を作成する
DbjUploadInfo upinfo = factory.createUploadInfo(
               uplFilePath,
               "test.html",  // retrievalNameプロパティを指定
               "text/html",  // 変更後のレンディションタイプ
               null,
               null );

// マスタレンディションのレンディションタイプを変更する
// (text/plain -> text/htmlに変更する)
```

```
obj.uploadContents(
                null,          // マスタレンディションが対象
                upinfo );      // 文書のアップロード情報
```

(7) レンディションのプロパティの参照

レンディションのプロパティは，DbjObj#getRenditionList メソッドを使用して参照します。レンディションのプロパティとして参照できるのは，dbrProp_RetrievalName プロパティおよび dbrProp_RenditionStatus プロパティです。

なお，DbjObj#getRenditionList メソッドを実行すると，文書中のすべてのレンディションのプロパティが，リストで取得できます。特定のレンディションのプロパティだけを取得することはできません。

レンディションのプロパティを参照する例を示します。

// レンディションのプロパティを参照する例

```
  // obj：DbjObjインターフェース

// 取得するプロパティ名のコレクションを作成する
Set<String> propdef = new HashSet<String>();
propdef.add("dbrProp_RetrievalName");

// dbrProp_RetrievalNameプロパティのリストを取得する
DbjRenditionList renlist = obj.getRenditionList( propdef );

// 取得したレンディションのプロパティを一覧表示する
for ( int i = 0; i < renlist.size(); i ++ ) {
  System.out.println( renlist.getRenditionInfo(i)
        .propSet().getStringVal("dbrProp_RetrievalName") );
}
```

なお，マスタレンディションのプロパティは，DbjRenditionList インターフェースおよび DbjRenditionInfo インターフェースを使わないで，ほかのプロパティと同様の方法でも参照できます。

マスタレンディションの dbrProp_RetrievalName プロパティを参照する例を示します。

// マスタレンディションのdbrProp_RetrievalName**プロパティを参照する例**

```
  // docspc：DbjDocSpaceインターフェース

// 取得するプロパティ名のコレクションを作成する
Set<String> propdef = new HashSet<String>();
propdef.add("dbrProp_RetrievalName");

DbjObj obj = docspc.createObjConnection( oiid );
// プロパティ値集合をProxyオブジェクトにロードする
obj.readProperties( propdef );

// プロパティの値を参照する
String retrievalName
  = obj.propSet().getStringVal("dbrProp_RetrievalName");
```

(8) レンディションのプロパティの更新

レンディションのプロパティは，DbjObj#writeRenditionProperties メソッドで更新します。レンディションのプロパティとして参照できるのは，dbrProp_RetrievalName プロパティおよび dbrProp_RenditionStatus プロパティです。

レンディションのプロパティを更新する例を示します。

// レンディションのプロパティを更新する例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// レンディションのプロパティを更新するための
// プロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();
props.setPropVal("dbrProp_RetrievalName", "a.txt");

// レンディションのプロパティを更新する
obj.writeRenditionProperties(
                "text/plain",// 対象にするレンディションタイプ
                props );
```

1.10.12 リファレンスファイル文書の操作

ここでは，リファレンスファイル文書の次の操作について説明します。

- リファレンスファイル文書の作成
- リファレンスファイル文書のコンテンツのアップロード
- リファレンスファイル文書のコンテンツのダウンロード
- リファレンスファイル文書の削除

それぞれの操作に使用する各メソッドの詳細については，「6. 文書管理クラス詳細」を参照してください。

これらの操作を実行する前に，まず，コンテンツロケーションをコンテンツの相対パスで管理するために，DbjSession#setReferencePath メソッドでコンテンツ格納先ベースパスを設定しておきます。

なお，リファレンスファイル文書の管理の考え方については，マニュアル「DocumentBroker Version 5 概説」の「リファレンスファイル文書の管理モデル」を参照してください。また，リファレンスファイルを管理するバージョン付き文書のバージョンの操作については，「1.10.10 バージョン付きオブジェクトのバージョン操作」を参照してください。

(1) リファレンスファイル文書の作成

リファレンスファイル文書を作成するには，文書オブジェクト生成メソッド（DbjDocSpace#createDocument メソッドおよび DbjDocSpace#createVrDocument メソッド）に，リファレンスファイル文書のアップロード情報を指定します。

リファレンスファイル文書を作成する例を次に示します。

// リファレンスファイル文書を作成する例

```
// session ： DbjSessionインターフェース
// factory ： DbjFactoryインターフェース
// docsp : DbjDocSpaceインターフェース

// コンテンツ格納先ベースパスの設定
session.setReferencePath(
                basePath );      // コンテンツ格納先ベースパス

// 初期値として設定するプロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();
// バージョニングオブジェクトのプロパティ(Author)を設定する
props.setPropVal("Author", "suzuki");
// カレントバージョンのプロパティ(Ver)を設定する
//  （バージョンオブジェクトのプロパティは@を付けて指定する）
props.setPropVal("@Ver", "01-00");
```

```
// リファレンスファイル文書のパス情報を作成する
DbjReferencePathInfo pathInfo = factory.createReferencePathInfo(
                DbjDef.OPERATEMODE_USER_RELATIVE_CONTENT,
                            // コンテンツのパス操作モード
                file,       // 登録するファイルのパス
                targetPath, // コンテンツ格納先パス
                null  );    // 削除するディレクトリのルートパス

// 文書のアップロード情報のリストを作成する
List<DbjUploadInfo> uploadlist = new ArrayList<DbjUploadInfo>();
uploadlist.add( factory.createReferenceUploadInfo(
                null,
                retrievalName, // retrievalNameプロパティを指定
                null,          // レンディションタイプは自動で設定する
                null,          // レンディションプロパティは設定しない
                null,          // 全文検索インデクスは作成しない
                pathInfo));    // パス情報

// リンク設定情報のリストを作成する（直接型リンクで関連付ける）
List<DbjSetLinkInfo> linklist = new ArrayList<DbjSetLinkInfo>();
linklist.add( factory.createSetDCRLinkInfo(
              docspc.createObjConnection(parentoiid),
                // 上位フォルダのOIID
null ) );       // リンクプロパティは指定しない

// バージョン付き文書オブジェクトを作成する
DbjObj obj = docspc.createVrDocument(
    "mdmClass_CfgH", // バージョニングオブジェクトのトップオブジェクトクラス
    "mdmClass_Document", // バージョンオブジェクトのトップオブジェクトクラス
    props,               // プロパティ値集合
    uploadlist,          // 文書のアップロード情報のリスト
    linklist );          // リンク設定情報のリスト
```

(2) リファレンスファイル文書のコンテンツのアップロード

リファレンスファイル文書のコンテンツをアップロードするには，DbjObj#updateContents メソッドに，リファレンスファイル文書のアップロード情報を指定します。

リファレンスファイル文書のコンテンツをアップロードする例を次に示します。

// リファレンスファイル文書のコンテンツをアップロードする例

```
// factory ： DbjFactoryインターフェース
// obj ： DbjObjインターフェース

// コンテンツ格納先ベースパスは設定済み
// リファレンスファイル文書のパス情報を作成する
DbjReferencePathInfo pathInfo = factory.createReferencePathInfo(
        DbjDef.OPERATEMODE_USER_RELATIVE_CONTENT,
                // コンテンツのパス操作モード
        file,   // 登録するファイルのパス
        null,   // コンテンツ格納先パス
        null ); // 削除するディレクトリのルートパス

// 文書のアップロード情報を作成する
DbjUploadInfo upinfo = factory.createReferenceUploadInfo(
         null,
         retrievalName, // retrievalNameプロパティを指定
         null,          // レンディションタイプは自動で設定する
         null,          // レンディションプロパティは設定しない
         null,          // 全文検索インデクスは作成しない
         pathInfo);     // パス情報

// コンテンツをアップロードする
obj.uploadConents(
         null,     // マスタレンディションが対象
```

```
        upinfo ); // 文書のアップロード情報を指定
```

(3) リファレンスファイル文書のコンテンツのダウンロード

リファレンスファイル文書のコンテンツをダウンロードするには，DbjObj#downloadContents メソッドに，ダウンロード先のパス情報を指定します。

リファレンスファイル文書のコンテンツをダウンロードする例を次に示します。

// リファレンスファイル文書のコンテンツをダウンロードする例

```
// factory ： DbjFactoryインターフェース
// obj ： DbjObjインターフェース

// コンテンツ格納先ベースパスは設定済み
// ダウンロード先のパス情報を作成する
DbjReferencePathInfo pathInfo = factory.createReferencePathInfo(
DbjDef.OPERATEMODE_USER_RELATIVE_CONTENT, // コンテンツのパス操作モード
        file,   // ダウンロード先のファイルのパス
        null,   // コンテンツ格納先パス
        null ); // 削除するディレクトリのルートパス

// コンテンツをダウンロードする
DbjReferenceContentInfo  continfo = obj.downloadConents(
        null,       // レンディションタイプ
        pathInfo);  // ダウンロード用パス情報
```

(4) リファレンスファイル文書の削除

リファレンスファイル文書を削除するには，DbjObj#removeObject メソッドに，削除時に使用するパス情報を指定します。

リファレンスファイル文書を削除する例を次に示します。

// リファレンスファイル文書を削除する例

```
// factory ： DbjFactoryインターフェース
// obj ： DbjObjインターフェース

// コンテンツ格納先ベースパスは設定済み
// リファレンスファイル文書のパス情報を作成する
DbjReferencePathInfo pathInfo = factory.createReferencePathInfo(
DbjDef.OPERATEMODE_USER_RELATIVE_CONTENT,
                    // コンテンツのパス操作モード
        null,       // コンテンツロケーション
        null,       // コンテンツ格納先パス
        dirPath ); // 削除するディレクトリのルートパス

// 文書オブジェクトを削除する
obj.removeObject( pathInfo );
```

1.10.13 リンクの操作

ここでは，リンクの操作について説明します。

リンクを設定して管理できるのは，文書管理オブジェクトのうち，文書およびフォルダです。したがって，この項の説明中の文書管理オブジェクトとは，文書またはフォルダを指します。

また，フォルダを使用したリンクでは特に，リンク元オブジェクトであるフォルダを**上位オブジェクト**，リンク先オブジェクトである文書またはフォルダを**下位オブジェクト**といいます。

(1) リンクの操作の概要

ここでは，リンクの操作の概要について説明します。

DocumentBroker クラスライブラリで設定できるリンクは，次の 2 種類です。

● 直接型リンク

● 参照型リンク

● 文書間リンク

リンク種別と，リンク元またはリンク先に設定できるオブジェクト種別の対応は，次に示す表のとおりです。

表 1-8　リンク種別とリンク元またはリンク先に設定できるオブジェクト種別

リンク種別	リンク元オブジェクトのオブジェクト種別	リンク先オブジェクトのオブジェクト種別
直接型リンク	バージョンなしフォルダ	バージョンなし文書 バージョン付き文書 バージョンなしフォルダ
参照型リンク	バージョンなしフォルダ	バージョンなし文書 バージョン付き文書 バージョンなしフォルダ
文書間リンク	バージョンなし文書 バージョン付き文書	バージョンなし文書 バージョン付き文書

リンクの操作には，次の操作があります。

● リンクの設定

● リンクをたどる文書管理オブジェクトの参照

● リンクのプロパティの操作

● リンクの解除

リンクの操作には，文書またはフォルダのインターフェースである DbjObj インターフェースを使用する方法と，リンクオブジェクトのインターフェースである DbjLinkObj インターフェースを使用する方法があります。また，文書管理オブジェクトの作成と同時にリンクを設定する場合は，DbjDocSpace インターフェースを使用してリンクを設定することもできます。なお，リンクオブジェクトは，リンク元の文書管理オブジェクトに属するオブジェクトです。DbjObj インターフェースを使用して操作する場合，リンクの設定，解除またはプロパティの変更などは，リンク元オブジェクトから実行します。リンクオブジェクトには，それぞれリンク識別子が設定されています。このリンク識別子を使用して，リンクを特定できます。

(2) 文書管理オブジェクト作成時のリンクの設定

リンクの設定には，2 種類の方法があります。

● 文書管理オブジェクトを作成すると同時に既存の文書管理オブジェクトへのリンクを設定する方法

● 既存の文書管理オブジェクト同士をリンクで関連付ける方法

なお，文書管理オブジェクトごとに，設定できるリンクの種別は異なります。

ここでは，DbjDocSpace インターフェースの createXXX（XXX は作成する文書管理オブジェクトの種別によって異なります）メソッドを実行して，文書管理オブジェクトの作成と同時に既存の文書管理オブジェクトへのリンクを設定する方法について説明します。文書管理オブジェクトの作成については，

「1.9.3　文書管理オブジェクトの作成」を参照してください。

文書管理オブジェクト作成時に，既存のフォルダからリンクを設定する例を示します。この例では，バージョン付き文書を，作成すると同時に二つのフォルダからリンクさせます。一つのフォルダからは直接型リンク，もう一つのフォルダからは参照型リンクでリンクさせます。リンクさせるオブジェクトについての情報は，DbjSetLinkInfo インターフェースのサブインターフェースを要素とするリストで指定します。

// 文書管理オブジェクト作成時に，既存のフォルダからのリンクを設定する例

```
  // factory：DbjFactoryインターフェース
  // docspc：DbjDocSpaceインターフェース

// リンク設定情報のリストを作成する
// parentoiid1, parentoiid2はそれぞれ上位オブジェクトのOIID

List<DbjSetLinkInfo> linklist = new ArrayList<DbjSetLinkInfo> ();
linklist.add( factory.createSetDCRLinkInfo(
              docspc.createObjConnection( parentoiid1 ),null ) );
linklist.add( factory.createSetRCRLinkInfo(
              docspc.createObjConnection( parentoiid2 ),null ) );

// バージョン付き文書を作成する
DbjObj obj = docspc.createVrDocument(
         "mdmClass_CfgH",
           // バージョニングオブジェクトのトップオブジェクトクラス
         "mdmClass_Document",
           // バージョンオブジェクトのトップオブジェクトクラス
         null,                  // 初期プロパティは設定しない
         null,                  // 初期コンテンツは登録しない
         linklist );            // リンク設定情報のリスト
```

(3) 既存の文書管理オブジェクト間のリンクの設定

ここでは，既存の文書管理オブジェクトからほかの文書管理オブジェクトにリンクを設定する方法について説明します。この操作には，リンク元オブジェクトの，DbjObj#link メソッドを使用します。メソッドの引数には，リンク先オブジェクトを，次のどちらかのリストで指定します。

● DbjSetLinkInfo インターフェースのサブインターフェースのリスト

● DbjObj インターフェースのリスト

リストの要素を DbjSetLinkInfo インターフェースのサブインターフェースにすると，リンク先のオブジェクトごとにリンク種別を設定できます。例えば，「文書 1 は直接型リンクで関連付けて，文書 2 は参照型リンクで関連付けたい」という場合は，この形式を使用します。

リストの要素を DbjObj インターフェースにすると，リストに指定したすべてのリンク先オブジェクトを，引数に指定したリンク種別で一括して関連付けられます。例えば，「複数の文書をまとめて直接型リンクで関連付けたい」という場合は，この形式を使用できます。ただし，この形式で指定できるリンク種別は，直接型リンクまたは参照型リンクだけです。

ここでは，まず，リストの要素を DbjSetLinkInfo インターフェースのサブインターフェースにして，個々にリンク設定情報を指定する例を示します。

// DbjSetLinkInfoインターフェースを使用して
// 個々にリンク設定情報を指定する例

```
  // factory：DbjFactoryインターフェース
  // docspc：DbjDocSpaceインターフェース

// リンク設定情報のリストを作成する
```

```
// childoiid1, childoiid2はそれぞれリンク先オブジェクトのOIID
List<DbjSetLinkInfo> linklist = new ArrayList<DbjSetLinkInfo> ();
linklist.add( factory.createSetDCRLinkInfo(
              docspc.createObjConnection( childoiid1 ),null ) );
linklist.add( factory.createSetRCRLinkInfo(
              docspc.createObjConnection( childoiid2 ),null ) );

// フォルダのDbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( parentoiid );

// リンク先オブジェクトにリンクを設定する
// 二つのオブジェクトがフォルダから直接型リンクと参照型リンクで
// 関連付けられる
obj.link( linklist );
```

次に DbjObj インターフェースを使用して，複数の文書管理オブジェクトをまとめて関連付ける例を示します。

// DbjObjインターフェースを使用して
// 同じリンク種別で関連付ける例

```
  // docspc：DbjDocSpaceインターフェース

// リンク対象となるリンク先オブジェクトのリストを作成する
// childoiid1, childoiid2はそれぞれリンク先オブジェクトのOIID
List<DbjObj> childlist = new ArrayList<DbjObj>();
childlist.add( docspc.createObjConnection( childoiid1 ) );
childlist.add( docspc.createObjConnection( childoiid2 ) );

// フォルダのDbjObjインターフェースを取得する
DbjObj obj = docspc.createObjConnection( parentoiid );

// リンク先オブジェクトにリンクを設定する
// 二つのオブジェクトがフォルダから直接型リンクで関連付けられる
obj.link( DbjDef.LINK_DCR, childlist );
```

また，DbjObj#link メソッドを使用して，検索結果として取得したオブジェクトをまとめて一つのフォルダに関連付けることもできます。

検索結果として取得したオブジェクトをまとめてフォルダに関連付ける例を示します。

// 検索結果として取得したオブジェクトをまとめてフォルダに関連付ける例

```
  // docspc：DbjDocSpaceインターフェース
  // parentObj：上位オブジェクトのインターフェース

// 関連付けるリンク先オブジェクトを検索する
DbjResultSet result = docspc.executeSearch(
              "SELECT dmaProp_OIID  FROM DV  WHERE ...",
              null,
              null );

// 検索結果として取得したOIIDを基に
// DbjObjListインターフェースを取得する
//（検索結果の0列目がdmaProp_OIIDである）
DbjObjList childlist = docspc.createObjList(result, 0 , null);

// 上位オブジェクトからリンクを設定する
// 検索結果が上位オブジェクトに直接型リンクで関連付けられる
parentObj.link( DbjDef.LINK_DCR, childlist );
```

(4) リンクをたどる文書管理オブジェクトの参照

リンクによって関連付けられた文書管理オブジェクトは，リンクをたどって参照できます。ここでは，リ

ンクをたどって文書管理オブジェクトを参照する方法について説明します。

文書またはフォルダのインターフェースを使用して，リンクをたどって文書管理オブジェクトを参照する場合に使用するメソッドとリンク種別の関係を，次の表に示します。

表 1-9　リンクをたどる参照に使用するメソッドとリンク種別の関係

メソッド	たどる元になる文書管理オブジェクト	リンク種別
getDCRParent	文書 フォルダ	直接型リンク
getParentList	文書 フォルダ	直接型リンク 参照型リンク
getChildList	フォルダ	直接型リンク 参照型リンク
getRelList	文書	文書間リンク

ここでは，直接型リンク，参照型リンクをたどる参照をたどる参照と，文書間リンクをたどる参照について，分けて説明します。

(a) 直接型リンクまたは参照型リンクをたどる参照

ここでは，フォルダを使用した関連付けの場合の，直接型リンク，参照型リンクをたどる参照について説明します。

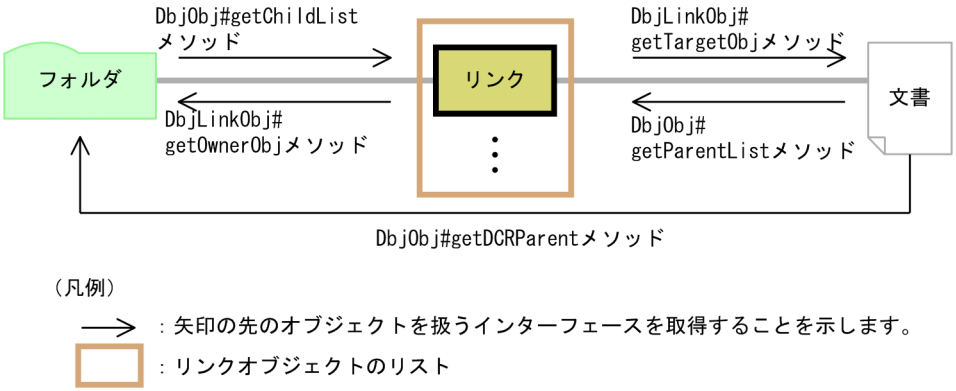
上位オブジェクトから下位オブジェクトを取得する場合，DbjObj#getChildList メソッドを使用します。このメソッドでは，上位オブジェクトから下位オブジェクトを関連付けているリンクオブジェクトのリストを扱うインターフェース（DbjLinkObjList インターフェース）を取得できます。メソッド実行時には，「直接型リンクだけを取得したい」，「直接型リンクと参照型リンクを取得したい」など，リンクの種別も指定できます。次に，DbjLinkObjList インターフェースの要素である DbjLinkObj インターフェースを取得して，DbjLinkObj#getTargetObj メソッドによって，下位オブジェクトのインターフェースを取得します。

下位オブジェクトから上位オブジェクトを取得する場合，DbjObj#getParentList メソッドを使用します。このメソッドでは，下位オブジェクトを上位オブジェクトに関連付けているリンクオブジェクトのリストを扱うインターフェース（DbjLinkObjList インターフェース）を取得できます。メソッド実行時には，リンクの種別を指定できます。次に，DbjLinkObjList インターフェースの要素である DbjLinkObj インターフェースを取得して，DbjLinkObj#getOwnerObj メソッドによって，それぞれの上位オブジェクトのインターフェースを取得します。

また，上位オブジェクトと下位オブジェクトが直接型リンクで関連付けられている場合，下位オブジェクトの DbjObj#getDCRParent メソッドで，リンクオブジェクトを取得しないで，直接上位オブジェクトのインターフェースを取得することもできます。

フォルダを使用したリンクを設定している場合の，リンクをたどる参照に使用するメソッドの関係について，次の図に示します。

図 1-17　リンクをたどる参照に使用するメソッドの関係（フォルダを使用する場合）



取得した上位オブジェクトまたは下位オブジェクトを一括操作したい場合は，DbjLinkObjList インターフェースを取得してから，DbjLinkObjList#getOwnerObjList または DbjLinkObjList#getTargetObjList メソッドを実行して，DbjObjList インターフェースを取得することもできます。複数オブジェクトを一括操作する場合の詳細については，「1.10.14　複数の文書管理オブジェクトの一括操作」を参照してください。

ここでは，まず，上位オブジェクトから直接型リンクで関連付けられている下位オブジェクトを取得して，下位オブジェクトの OIID の一覧を出力する例を示します。

// 下位オブジェクトのOIIDの一覧を出力する例

```
  // parentObj：DbjObjインターフェース
  // （上位オブジェクトのインターフェース）

// 下位オブジェクトを取得する
DbjLinkObjList objlist = parentObj.getChildList(
            null,                // 取得するプロパティは指定しない
            null,                // リンクのプロパティは指定しない
            DbjDef.LINK_DCR,     // 直接型リンクだけを取得する
            DbjDef.OBJTYPE_DOC, // 文書だけを取得する
            null );              // 検索結果取得情報は指定しない

// 下位オブジェクトのOIID一覧を出力する
for(int i = 0 ; i < objlist.size(); i++ ) {
  System.out.println("Child OIID["+i+"]="
  + objlist.getLinkObj(i).getTargetObj().getOiid() );
}
```

次に，下位オブジェクトから参照型リンクで関連付けられている上位オブジェクトの一覧を取得して，上位オブジェクトの OIID の一覧を出力する例を示します。

// 上位オブジェクトのOIIDの一覧を出力する例

```
  // childObj：DbjObjインターフェース
  // （下位オブジェクトのインターフェース）

// 上位オブジェクトを取得する
DbjLinkObjList linklist
        = childObj.getParentList(
            null,                // 取得するプロパティは指定しない
            null,                // リンクのプロパティは指定しない
            DbjDef.LINK_RCR,     // 参照型リンクだけを取得する
            DbjDef.OBJTYPE_FOLDER, // フォルダを取得する
            null );              // 検索結果取得情報は指定しない
```

```
// DbjObjListインターフェースを取得する
DbjObjList objlist = linklist.getOwnerObjList();

// 上位オブジェクトのOIID一覧を出力する
for(int i = 0 ; i < objlist.size(); i++ ) {
  System.out.println("Parent OIID["+i+"]="
           + objlist.getObj(i).getOiid() );
}
```

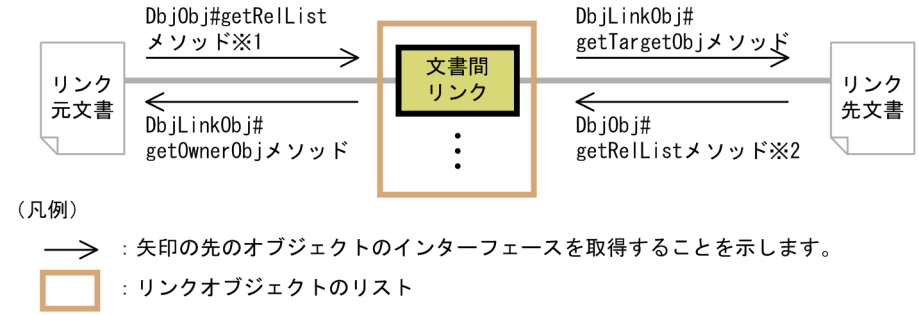
(b) 文書間リンクをたどる参照

文書間リンクをたどってオブジェクトを参照する場合，DbjObj#getRelList メソッドを使用します。リンク元オブジェクトからリンク先オブジェクトを取得する場合も，リンク先オブジェクトからリンク元オブジェクトを取得する場合も，同じメソッドを使用します。リンク先オブジェクト，リンク元オブジェクトのどちらを取得するかは，メソッドの引数に**リレーション種別**として指定します。

DbjObj#getRelList メソッドを実行すると，DbjLinkObjList インターフェースが取得できます。このインターフェースから，要素である DbjLinkObj インターフェースを取得します。DbjLinkObj インターフェースを使用してリンク元オブジェクトを取得する場合は，DbjLinkObj#getOwnerObj メソッドを，リンク先オブジェクトを取得する場合は，DbjLinkObj#getTargetObj メソッドを実行します。

文書間リンクを設定している場合に，リンクをたどる参照に使用するメソッドの関係について，次の図に示します。

図 1-18　リンクをたどる参照に使用するメソッドの関係（文書間リンクを使用する場合）



注※1　リレーション種別を指定する引数にDbjDef.RELATION_HEADを指定します。
注※2　リレーション種別を指定する引数にDbjDef.RELATION_TAILを指定します。

また，取得したリンク元オブジェクトまたはリンク先オブジェクトを一括操作したい場合は，DbjLinkObjList インターフェースを取得してから，DbjLinkObjList#getOwnerObjList または DbjLinkObjList#getTargetObjList メソッドを実行して，DbjObjList インターフェースを取得することもできます。

ここでは，リンク元文書からリンク先文書の一覧を取得して，リンク先文書の OIID の一覧を出力する例を示します。

// リンク先オブジェクトのOIIDの一覧を出力する例

```
  // obj：DbjObjインターフェース

// リンク先オブジェクトを取得する
DbjLinkObjList objlist = obj.getRelList(
           null,                    // 取得するプロパティは指定しない
           null,                    // リンクのプロパティは指定しない
           DbjDef.RELATIONEND_HEAD, // リンク先オブジェクトを取得する
           DbjDef.OBJTYPE_DOC,      // 文書だけを取得する
```

```
            null );                          // 検索結果取得情報は指定しない

// リンク先オブジェクトのOIID一覧を出力する
for(int i = 0 ; i < objlist.size(); i++ ) {
  System.out.println("OIID["+i+"]="
  + objlist.getLinkObj(i).getTargetObj().getOiid() );
}
```

(5) リンクのプロパティの操作

リンクのプロパティとは，リンクオブジェクトに設定されているプロパティのことです。

リンクのプロパティも，ほかの文書管理オブジェクトのプロパティと同様に，Proxy オブジェクト（リンク Proxy オブジェクト）のリンクプロパティ値集合プロパティにロードして参照します。また，更新する場合は，リンク Proxy オブジェクトのリンクプロパティ値集合プロパティの値をフラッシュします。

リンクのプロパティのロードとフラッシュに使用するメソッドを次に示します。

リンクのプロパティのロードに使用するメソッド

- DbjObj#getChildList メソッド
- DbjObj#getParentList メソッド
- DbjObj#getRelList メソッド
- DbjLinkObj#readProperties メソッド
- DbjLinkObjList#readProperties メソッド

DbjObj インターフェースのメソッドは，一覧の取得と同時にリンクのプロパティをロードします。なお，文書管理オブジェクトのプロパティも同時にロードできます。

リンクのプロパティのフラッシュに使用するメソッド

- DbjObj#link メソッド
- DbjDocSpace#createDocument メソッド
- DbjDocSpace#createVrDocument メソッド
- DbjDocSpace#createFolder メソッド
- DbjLinkObj#writeProperties メソッド
- DbjLinkObjList#writeProperties メソッド

DbjObj インターフェースおよび DbjDocSpace インターフェースのメソッドでは，リンクの設定と同時にリンクのプロパティをフラッシュします。

リンクのプロパティを参照および更新する例を示します。

// リンクのプロパティの参照と更新の例

```
  // factory：DbjFactoryインターフェース
  // parentObj：DbjObjインターフェース
  // （フォルダのインターフェース）

// 特定のフォルダから下位オブジェクトを取得する
DbjLinkObjList linklist = parentObj.getChildList(
                           null,
                           null,
                           DbjDef.LINK_DCR,
                           DbjDef.OBJTYPE_ANY,
                           null );

// 更新するプロパティのプロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();
props.setPropVal( "status", 0 );

// linklistの先頭の要素の，リンクのプロパティを更新する
```

```
DbjLinkObj linkObj = linklist.getLinkObj(0);
linkObj.writeProperties( props );
```

次に，DbjLinkObjList インターフェースを使用して，複数のリンクのプロパティを，一括して更新する例を示します。

// 複数のリンクのプロパティの参照と更新の例

```
  // factory：DbjFactoryインターフェース
  // parentObj：DbjObjインターフェース
  // （フォルダのインターフェース）

// 特定のフォルダから下位オブジェクトを取得する
DbjLinkObjList linklist = parentObj.getChildList(
                            null,
                            null,
                            DbjDef.LINK_DCR,
                            DbjDef.OBJTYPE_ANY,
                            null );

// 更新するプロパティのプロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();
props.setPropVal( "status", 0 );

// 複数のリンクのプロパティを更新する
linklist.writeProperties( props );
```

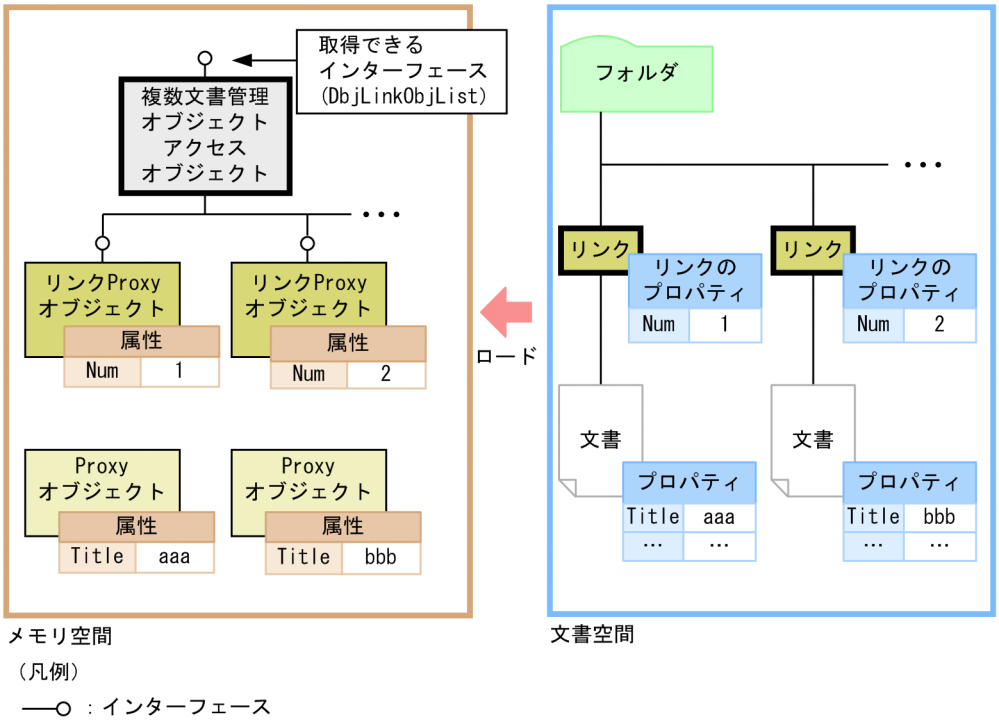
リンクのプロパティと同時に，下位オブジェクトのプロパティを参照する例を示します。

// リンクのプロパティと同時に下位オブジェクトのプロパティも参照する例

```
  // parentObj：DbjObjインターフェース

// 取得する下位オブジェクトのプロパティ名のコレクションを作成する
Set<String> propdef = new HashSet<String>();
propdef.add( "Title" );

// 取得するリンクのプロパティ名のコレクションを作成する
Set<String> linkpropdef = new HashSet<String>();
linkpropdef.add( "num" );

// フォルダから下位オブジェクトを取得する
DbjLinkObjList linklist
     = parentObj.getChildList(
          propdef,             // 取得する下位オブジェクトのプロパティ
          linkpropdef,         // 取得するリンクのプロパティ
          DbjDef.LINK_DCR,
          DbjDef.OBJTYPE_ANY,
          null );

// 先頭のリンクのプロパティを出力する
System.out.println(
  linklist.getLinkObj(0).propSet().getIntVal( "num" ) );

// 下位オブジェクトのインターフェースのリストを取得して,
// 先頭のリンクで関連付けているオブジェクトのプロパティを出力する
System.out.println(
  linklist.getLinkObj(0).getTargetObj()
  .propSet().getStringVal( "Title" ) );
```

なお，このコーディング例の DbjObj#getChildList メソッドでロードできるプロパティおよび取得できるインターフェースは，次の図のようになります。

図 1-19　DbjObj#getChildList メソッドでロードできるプロパティと取得できるインターフェース



(6) 文書管理オブジェクトの移動

フォルダから直接型リンクで関連付けられている文書管理オブジェクトを，別のフォルダから直接型リンクで関連付けるように変更できます。これによって，あるフォルダで管理していた文書やフォルダを，別のフォルダに移動するような管理ができます。文書管理オブジェクトの移動には，DbjObj#move メソッドを使用します。

フォルダに直接型リンクでリンク付けられている文書を移動する例を示します。

// フォルダに直接型リンクでリンク付けられている文書を移動する例

```
  // docspc：DbjDocSpaceインターフェース
  // parentObj：DbjObjインターフェース

// フォルダに直接型リンクで関連付けている文書一覧を取得する
DbjLinkObjList linklist = parentObj.getChildList(
                              null,
                              null,
                              DbjDef.LINK_DCR,
                              DbjDef.OBJTYPE_DOC,
                              null );

// 先頭の要素の文書を別なフォルダに移動する
DbjObj parentObj2 = docspc.createObjConnection( parentoiid2 );
linklist.getLinkObj(0).getTargetObj().move( parentObj2 );
```

(7) リンクの解除

リンクは，次のどれかのメソッドで解除します。

リンク元オブジェクトのインターフェース（DbjObj インターフェース）を使用する場合

- DbjObj#unlink メソッド
- DbjObj#unlinkByLinkId メソッド

リンクオブジェクトのインターフェース（DbjLinkObj インターフェース）を使用する場合

- DbjLinkObj#removeObject メソッド

リンクオブジェクトのリストのインターフェース（DbjLinkObjList インターフェース）を使用する場合

- DbjLinkObjList#removeObjects メソッド

DbjObj インターフェースおよび DbjLinkObjList インターフェースのメソッドで解除する場合，複数のリンクをまとめて解除できます。さらに，DbjObj インターフェースのメソッドでは，解除するリンクを特定することもできます。この場合は，下位オブジェクトの DbjObj インターフェースのリスト，リンクの DbjLinkObj インターフェースのリストまたはリンク識別子のリストを指定します。

DbjLinkObjList インターフェースおよび DbjLinkObj インターフェースのメソッドを使用して解除する場合は，リンクオブジェクトそのものを削除することで，リンクを解除します。

まず，DbjObj インターフェースを使用して，下位オブジェクトを指定して解除する例を示します。

// 下位オブジェクトを指定してリンクを解除する例

```
  // parentObj：DbjObjインターフェース
  // （フォルダのインターフェース）

// リンクの一覧を取得する
DbjLinkObjList linklist = parentObj.getChildList(
                              null,
                              null,
                              DbjDef.LINK_DCR|DbjDef.LINK_RCR,
                              DbjDef.OBJTYPE_ANY,
                              null );

// 下位オブジェクトの一覧を取得する
DbjObjList childlist = linklist.getTargetObjList();

// 指定した下位オブジェクトとのリンクをすべて解除する
parentObj.unlink( childlist );
```

次に，DbjObj インターフェースを使用して，リンク識別子を指定してリンクを解除する例を示します。

// リンク識別子を指定してリンクを解除する例

```
  // parentObj：DbjObjインターフェース
  // （フォルダのインターフェース）

// リンクの一覧を取得する
DbjLinkObjList linklist = parentObj.getChildList(
                              null,
                              null,
                              DbjDef.LINK_DCR|DbjDef.LINK_RCR,
                              DbjDef.OBJTYPE_ANY,
                              null );

// リンク識別子の一覧を取得する
List<String> linkIdList = linklist.getLinkIdList();

// 指定した下位オブジェクトとのリンクをすべて解除する
parentObj.unlinkByLinkId( linkIdList );
```

DbjLinkObj インターフェースおよび DbjLinkObjList インターフェースを使用して，リンクオブジェクトを削除してリンクを解除する例を示します。

// リンクオブジェクトを削除してリンクを解除する例

```
  // parentObj：DbjObjインターフェース
  // （フォルダのインターフェース）

// 特定のフォルダの下位オブジェクトを取得する
DbjLinkObjList linklist = parentObj.getChildList(
                              null,
                              null,
                              DbjDef.LINK_DCR,
                              DbjDef.OBJTYPE_ANY,
                              null );

// 先頭のリンクだけを解除する
linklist.getLinkObj(0).removeObject();

// 複数のリンクを一括して解除する
linklist.removeObjects();
```

1.10.14 入力ストリームを使用した文書の操作

入力ストリームを使用した次の文書の操作について説明します。

- 文書の作成
- 文書のコンテンツのアップロード
- 文書のコンテンツのダウンロード

(1) 文書の作成

文書を作成するには，文書オブジェクト生成メソッド（DbjDocSpace#createDocument メソッドおよび DbjDocSpace#createVrDocument メソッド）に，文書のアップロード情報を指定します。

文書を作成する例を次に示します。

// バージョン付き文書を作成する例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// 初期値として設定するプロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();

// バージョニングオブジェクトのプロパティ(Author)を設定する
props.setPropVal( "Author", "suzuki" );

// カレントバージョンのプロパティ (Ver)を設定する
// （バージョンオブジェクトのプロパティは@を付けて指定する）
props.setPropVal( "@Ver", "01-00" );

// 入力ストリーム
InputStream stream;

// 文書のアップロード情報のリストを作成する
List<DbjUploadInfo> uploadlist = new ArrayList<DbjUploadInfo>();
uploadlist.add( factory.createUploadInfoByStream(
             stream,          // 入力ストリーム
             retrievalName,   // retrievalNameプロパティを指定
             null,            // レンディションタイプは自動で設定する
             null,            // レンディションプロパティは設定しない
             null ) );        // 全文検索インデクスは作成しない

// リンク設定情報のリストを作成する(直接型リンクで関連付ける)
List<DbjSetLinkInfo>  linklist = new ArrayList<DbjSetLinkInfo>();
linklist.add( factory.createSetDCRLinkInfo(
             docspc.createObjConnection(parentoiid),
                          //上位フォルダのOIID
```

```
            null ) );       // リンクプロパティは指定しない

// バージョン付き文書を作成する
DbjObj obj = docspc.createVrDocument(
            "mdmClass_CfgH",
            // バージョニングオブジェクトのトップオブジェクトクラス
            "mdmClass_Document",
            // バージョンオブジェクトのトップオブジェクトクラス
            props,          // プロパティ値集合
            uploadlist,     // 文書のアップロード情報のリスト
            linklist );     // リンク設定情報のリスト
```

// リファレンスファイル文書を作成する例

```
  // session：DbjSessionのインターフェース
  // factory：DbjFactoryインターフェース
  // docsp：DbjDocSpaceインターフェース

// コンテンツ格納先ベースパスの設定
session.setReferencePath( basePath );       // コンテンツ格納先ベースパス

// 初期値として設定するプロパティ値集合を作成する
DbjPropSet props = factory.createPropSet();
// バージョニングオブジェクトのプロパティ(Author)を設定する
props.setPropVal( "Author", "suzuki" );
// カレントバージョンのプロパティ (Ver)を設定する
// （バージョンオブジェクトのプロパティは@を付けて指定する）
props.setPropVal( "@Ver", "01-00" );

// リファレンスファイル文書のパス情報を作成する
DbjReferencePathInfo pathInfo = factory.createReferencePathInfo(
            DbjDef.OPERATEMODE_RELATIVE_CONTENT,
                        // コンテンツのパス操作モード
            null,       // 登録するファイルのパス
            targetPath, // コンテンツ格納先パス
            null  );    // 削除するディレクトリのルートパス

// 入力ストリーム
InputStream stream;

// 文書のアップロード情報のリストを作成する
List<DbjUploadInfo> uploadlist = new ArrayList<DbjUploadInfo>();
uploadlist.add( factory.createReferenceUploadInfoByStream(
            stream,         // 入力ストリーム
            retrievalName,  // retrievalNameプロパティを指定
            null,           // レンディションタイプは自動で設定する
            null,           // レンディションプロパティは設定しない
            null,           // 全文検索インデクスは作成しない
            pathInfo  ) );  // パス情報

// リンク設定情報のリストを作成する(直接型リンクで関連付ける)
List<DbjSetLinkInfo>  linklist = new ArrayList<DbjSetLinkInfo>();
linklist.add( factory.createSetDCRLinkInfo(
            docspc.createObjConnection(parentoiid),
                    //上位フォルダのOIID
            null ) );  // リンクプロパティは指定しない

// バージョン付き文書を作成する
DbjObj obj = docspc.createVrDocument(
            "mdmClass_CfgH",
               // バージョニングオブジェクトのトップオブジェクトクラス
            "mdmClass_Document",
               // バージョンオブジェクトのトップオブジェクトクラス
            props,          // プロパティ値集合
            uploadlist,     // 文書のアップロード情報のリスト
            linklist );     // リンク設定情報のリスト
```

(2) コンテンツのアップロード

文書のコンテンツをアップロードする場合，DbjObj#updateContents メソッドに，文書のアップロード情報を指定します。

文書をアップロードする例を次に示します。

// コンテンツをアップロードする例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// 文書のアップロード情報を作成する
// （レンディションタイプは拡張子から自動設定するようにする）
DbjUploadInfo upinfo = factory.createUploadInfoByStream(
              stream,          // 入力ストリーム
              "test.html",     // retrivalNameプロパティを指定
              null,            // レンディションタイプは自動で設定する
              null,            // レンディションプロパティは設定しない
              null);           // 全文検索インデクスは作成しない

// マスタレンディションにコンテンツをアップロードする
obj.uploadConents(
              null,        // マスタレンディションが対象
              upinfo );    // 文書のアップロード情報を指定
```

// リファレンスファイル文書のコンテンツをアップロードする例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// コンテンツ格納先ベースパスは設定済み
// リファレンスファイル文書のパス情報を作成する
DbjReferencePathInfo pathInfo = factory.createReferencePathInfo(
              DbjDef.OPERATEMODE_RELATIVE_CONTENT,
                           // コンテンツのパス操作モード
              null,        // 登録するファイルのパス
              null,        // コンテンツ格納先パス
              null );      // 削除するディレクトリのルートパス

// 文書のアップロード情報を作成する
DbjUploadInfo upinfo = factory.createReferenceUploadInfoByStream(
              stream,          // 入力ストリーム
              "test.html",     // retrivalNameプロパティを指定
              null,    // レンディションタイプは自動で設定する
              null,    // レンディションプロパティは設定しない
              null,    // 全文検索インデクスは作成しない
              pathInfo));      // パス情報

// コンテンツをアップロード
obj.uploadConents(
              null,    // マスタレンディションが対象
              upinfo );    // 文書のアップロード情報を指定
```

(3) コンテンツのダウンロード

文書のコンテンツをダウンロードする場合は，DbjObj#downloadContents メソッドの形式 3 でダウンロードします。

コンテンツ（マスタレンディション）をダウンロードする例を次に示します。

// 文書のコンテンツをダウンロードする例

```
  // factory：DbjFactoryインターフェース
```

```
  // obj：DbjObjインターフェース

// マスタレンディションのコンテンツをダウンロード
DbjContentInfo  continfo = obj.downloadConents(null);
InputStream stream = continfo.getInputStream();
```

// リファレンスファイル文書のコンテンツをダウンロードする例

```
  // factory：DbjFactoryインターフェース
  // obj：DbjObjインターフェース

// マスタレンディションのコンテンツをダウンロード
DbjContentInfo  continfo = obj.downloadConents(null);
InputStream stream = continfo.getInputStream();
DbjReferenceContentInfo refcontinfo = continfo.getReferenceContentInfo();
continfo.close();
```

1.10.15 複数の文書管理オブジェクトの一括操作

複数の文書管理オブジェクトの一括操作について説明します。これは，複数の文書管理オブジェクトに，1トランザクションでアクセスする機能です。

複数の文書管理オブジェクトを操作する場合には，DbjObjList インターフェースまたは DbjVerObjList インターフェースを使用します。複数のリンクオブジェクトを操作する場合には，DbjLinkObjList インターフェースを使用します。

これらのインターフェースは，java.util.List インターフェースを継承しています。リストの要素は，それぞれ，DbjObj インターフェース，DbjVerObj インターフェース，DbjLinkObj インターフェースです。

ここでは，次の操作について説明します。

- 複数の文書管理オブジェクトのプロパティ一括操作
- 複数の文書管理オブジェクトに同じ種別のロックを設定した操作
- 複数の文書管理オブジェクトの一括削除
- 複数の文書管理オブジェクトの一括移動

なお，操作する文書管理オブジェクトにバージョン付きオブジェクトが含まれる場合，デフォルトの設定では，カレントバージョンが操作の対象になります。操作対象のバージョンを変更する場合には，要素である Proxy オブジェクトのインターフェースを取得して DbjObj#setTargetVersion メソッドを実行し，それぞれの要素のターゲットバージョンを変更してください。

(1) 複数の文書管理オブジェクトのプロパティ一括操作

DbjObjList インターフェースを使用して，複数の文書管理オブジェクトのプロパティを一括して操作できます。

プロパティの参照には，DbjObjList#readProperties メソッドを使用します。このメソッドの使用方法は，DbjObj#readProperties メソッドと同じです。ただし，DbjObjList インターフェースの要素である DbjObj インターフェースに対して，一括してメソッドが実行できます。引数を指定しない形式の DbjObjList#readProperties メソッドを実行した場合は，要素の DbjObj インターフェースで扱う Proxy オブジェクトにすでにロードされているプロパティ値集合を更新します。文書空間ごとに異なる種類のプロパティがロードされている場合は，文書管理オブジェクトごとに異なる種類のプロパティをロードします。

プロパティの更新には，DbjObjList#writeProperties メソッドを使用します。このメソッドの使用方法も DbjObj#writeProperties メソッドと同じです。Proxy オブジェクトごとに設定したプロパティの値を，複

数の文書管理オブジェクトに一括してフラッシュします。

プロパティの操作の詳細については,「1.10.7　文書管理オブジェクトのプロパティの操作」を参照してください。

複数の文書管理オブジェクトのプロパティを一括操作する例を示します。この例では, 検索でヒットした複数の文書管理オブジェクトのプロパティを一括して参照および更新します。

// 複数の文書管理オブジェクトのプロパティを一括操作する例

```
  // docspc：DbjDocSpaceインターフェース

// 検索を実行する
DbjResultSet result = docspc.executeSearch(
              "SELECT dmaProp_OIID  FROM DV  WHERE ...",
              null,
              null );

// 検索結果からDbjObjListインターフェースを取得する
DbjObjList objlist = docspc.createObjList( result, 0 , null );

// Countプロパティを一括してProxyオブジェクトにロードする
Set<String> propdef = new HashSet<String>();
propdef.add( "Count" );

objlist.readProperties( propdef );

// Proxyオブジェクトのプロパティの値を更新する
for ( int i = 0; i < objlist.size(); i ++ ) {
  int count = objlist.getObj(i).propSet().getIntVal( "Count" );
  count ++;
  objlist.getObj(i).propSet().setPropVal( "Count", count );
}

// 文書管理オブジェクトのプロパティを一括してフラッシュする
objlist.writeProperties();
```

(2) 複数の文書管理オブジェクトに同じ種別のロックを設定するインターフェースの取得

検索や一覧取得メソッドで取得した複数の文書管理オブジェクトに対して，同じ種別のロックを設定して操作するためのインターフェースを取得できます。この操作には，DbjObjList#lock メソッドを使用します。

複数の文書管理オブジェクトに write ロックを設定して操作する例を示します。この例では，検索結果として取得した文書管理オブジェクトに，write ロックを設定して操作します。

// 複数の文書管理オブジェクトにwriteロックを設定して操作する例

```
  // docspc：DbjDocSpaceインターフェース

// 検索を実行する
DbjResultSet result = docspc.executeSearch(
              "SELECT dmaProp_OIID  FROM DV  WHERE ...”,
              null,
              null );

// DbjObjListインターフェースを取得する
DbjObjList objlist = docspc.createObjList( result, 0 , null );

// writeロックを設定するDbjObjListインターフェースを取得する
DbjObjList objlistWithWriteLock = objlist.lock(DbjDef.LOCK_WRITE);

// writeロックでCountプロパティを参照する
objlistWithWriteLock.readProperties( Collections.singleton("Count") );
```

(3) 複数の文書管理オブジェクトの一括削除

DbjObjList インターフェースを使用して，複数文書管理オブジェクトを一括して削除できます。削除は，DbjObjList#removeObjects メソッドで実行します。

複数の文書管理オブジェクトを一括して削除する例を示します。この例では，特定のフォルダの下位オブジェクトをすべて削除します。

// 複数文書管理オブジェクトを一括して削除する例

```
  // parentObj：DbjObjインターフェース

// 特定フォルダから下位オブジェクトの一覧を取得する
DbjLinkObjList linklist = parentObj.getChildList(
                                null,
                                null,
                                DbjDef.LINK_DCR|DbjDef.LINK_RCR,
                                DbjDef.OBJTYPE_ANY,
                                null );

// リンクオブジェクトのリストからDbjObjListインターフェースを取得する
DbjObjList objlist = linklist.getTargetObjList();

// 一括削除する
objlist.removeObjects();
```

(4) 複数の文書管理オブジェクトの一括移動

フォルダから直接型リンクで関連付けられている文書管理オブジェクト（文書またはフォルダ）を，一括して別のフォルダに直接型リンクで関連付けるように変更できます。これによって，あるフォルダで管理していた文書を，別のフォルダに一括して移動するような管理ができます。複数の文書管理オブジェクトの移動には，DbjObjList#move メソッドを使用します。

フォルダに直接型リンクでリンク付けられている文書を一括して移動する例を示します。

// フォルダに直接型リンクでリンク付けられている文書を一括して移動する例

```
  // docspc：DbjDocSpaceインターフェース
  // parentObj：DbjObjインターフェース

// フォルダに直接型リンクで関連付けている文書一覧を取得する
DbjLinkObjList linklist = parentObj.getChildList(
                              null,
                              null,
                              DbjDef.LINK_DCR,
                              DbjDef.OBJTYPE_DOC,
                              null );

// 下位オブジェクトの一覧を取得する
DbjObjList childlist = linklist.getTargetObjList();

// 下位オブジェクトを別のフォルダに移動する
DbjObj parentObj2 = docspc.createObjConnection( parentoiid2 );
childlist.move( parentObj2 );
```

1.11 アクセス制御に関する操作

この節では，アクセス制御に関する操作について説明します。

アクセス制御機能を使用した管理の考え方については，マニュアル「DocumentBroker Version 5 概説」の「アクセス制御モデル」の説明を参照してください。

1.11.1 アクセス制御に使用するインターフェース

アクセス制御は，文書管理オブジェクトに対してアクセス制御情報を設定することで，実行します。

アクセス制御情報ごとに，使用するインターフェースおよびメソッドについて説明します。

ACFlag

ACFlag は，文書管理オブジェクトの dbrProp_OwnerPermission プロパティ，dbrProp_PrimaryGroupPermission プロパティおよび dbrProp_EveryonePermission プロパティとして設定します。このため，そのほかのプロパティの操作方法と同様に，DbjObj インターフェースなどを使用して操作します。
プロパティの操作については，「1.10.7 文書管理オブジェクトのプロパティの操作」を参照してください。

ローカル ACL

ローカル ACL は，文書管理オブジェクトの dbrProp_ACL プロパティに，VARRAY 型のプロパティとして設定します。可変長配列の要素は ACE です。
ACE の値の設定や取得には，パラメタクラスの DbjACE インターフェースが使用できます。また，ほかの VARRAY 型のプロパティの要素と同様に，DbjPropSet インターフェースも使用できます。DbjACE インターフェースを使用すると，ACE の値の取得や設定が，DbjPropSet インターフェースを使用する場合よりも単純なコーディングで実行できます。
可変長配列の操作には，パラメタクラスの DbjVArray インターフェースを使用します。また，可変長配列を文書管理オブジェクトのプロパティに設定する操作には，DbjObj インターフェースなどを使用します。
DbjACE インターフェースは，DbjFactory#createACE メソッドで取得します。DbjVArray インターフェースは，DbjFactory#createVArray メソッドなどで取得します。DbjObj インターフェースは，DbjDocSpace#createObjConnection メソッドなどで取得します。

パブリック ACL

パブリック ACL は，独立した文書管理オブジェクトです。DbjDocSpace#createPublicACL メソッドで作成します。また，パブリック ACL に設定するアクセス情報は，パブリック ACL のローカル ACL として操作します。
文書管理オブジェクトをパブリック ACL に指定したアクセス情報でアクセス制御するためには，文書管理オブジェクトからパブリック ACL をバインドします。バインドしているパブリック ACL の OIID は，文書管理オブジェクトの dbrProp_PublicACLIds プロパティに，VARRAY 型のプロパティとして設定されています。可変長配列の要素は，パブリック ACL の OIID を設定した文書管理オブジェクトのプロパティ（dbrProp_ACLIdElem プロパティ）です。
パブリック ACL の OIID の設定や取得には，DbjPublicACLIdElm インターフェースが使用できます。また，ほかの VARRAY 型のプロパティの要素と同様に，DbjPropSet インターフェースも使用できます。DbjPublicACLIdElm インターフェースを使用すると，パブリック ACL の OIID の取得や設定が，DbjPropSet インターフェースを使用する場合よりも単純なコーディングで実行できます。

以降の項では，次の操作について説明します。

● ローカル ACL と ACE の操作

● パブリック ACL の操作

1.11.2 ローカル ACL と ACE の操作

ローカル ACL は，文書管理オブジェクトのプロパティ（dbrProp_ACL プロパティ）として設定します。このプロパティは，VARRAY 型のプロパティです。VARRAY 型のプロパティの値である可変長配列は，パラメタクラスの DbjVArray インターフェースで操作します。VARRAY 型のプロパティの操作については，「1.10.7　文書管理オブジェクトのプロパティの操作」を参照してください。

可変長配列の要素である ACE は，パラメタクラスの DbjACE インターフェースで操作できます。

ローカル ACL を文書管理オブジェクトのプロパティとして設定する流れは，次のようになります。

1. **DbjVArray インターフェースを取得して，可変長配列を作成します。**
2. **DbjACE インターフェースを取得して，ACE を作成し，値を設定します。**
3. **ACE を，可変長配列の要素として追加します。**
   設定する ACE の数だけ，2. と 3. を繰り返します。
4. **可変長配列をプロパティ値集合（DbjPropSet）に設定します。**
5. **プロパティ値集合を，文書管理オブジェクトにフラッシュします。**

文書管理オブジェクトにローカル ACL を設定する例を示します。

// 文書管理オブジェクトにローカルACLを設定する例

```
  // factory：DbjFactoryインターフェース

// 可変長配列を作成する
DbjVArray acl = factory.createVArray( null );

// 一つ目のACEを作成する
//  （システムサブジェクトですべてのユーザに
//   参照更新権を設定する）
DbjACE elm = factory.createACE(factory.createPropSet());
elm.setSystemSubject( DbjDef.SYSSUBJECT_EVERYONE );
elm.setPermission( DbjDef.PERM_READ_WRITE );

// ACEをACLに追加する
acl.addPropSet( elm.propSet() );

// 二つ目のACEを作成する
//  （ユーザhitachiにフルコントロールを設定する）
elm.setUserSubject( "hitachi" );
elm.setPermission( DbjDef.PERM_FULL_CONTROL );

// ACEをACLに追加する
acl.addPropSet( elm.propSet() );

// 可変長配列をプロパティ値集合に設定する
DbjPropSet props = factory.createPropSet();
props.setPropVal( "dbrProp_ACL", acl );

// プロパティ値集合を文書管理オブジェクトにフラッシュする
obj.writeProperties( props );
```

1.11.3 パブリック ACL の操作

パブリック ACL は，独立した文書管理オブジェクトです。ここでは，パブリック ACL の作成と文書管理オブジェクトへのバインド，また，文書管理オブジェクトからのパブリック ACL のバインドについて説明します。また，文書管理オブジェクトからバインドしているパブリック ACL の一覧を取得したり，パブリック ACL からバインドされている文書管理オブジェクトの一覧を取得したりする方法も説明します。

(1) パブリック ACL の作成

パブリック ACL は，ほかの文書管理オブジェクトと同様に，DbjDocSpace インターフェースのメソッドで作成します。

パブリック ACL を作成する例を示します。この例では，パブリック ACL を作成して，同時に文書管理オブジェクトからバインドさせます。

// パブリックACLを作成する例

```
  // factory：DbjFactoryインターフェース
  // docspc：DbjDocSpaceインターフェース

// パブリックACLに設定するローカルACLを作成する
DbjVArray acl = factory.createVArray( null );

DbjACE elm = factory.createACE(factory.createPropSet());
elm.setSystemSubject( DbjDef.SYSSUBJECT_EVERYONE );
elm.setPermission( DbjDef.PERM_READ_WRITE );

acl.addPropSet( elm.propSet() );

elm.setUserSubject( "hitachi" );
elm.setPermission( DbjDef.PERM_FULL_CONTROL );

acl.addPropSet( elm.propSet() );

DbjPropSet props = factory.createPropSet();
props.setPropVal( "dbrProp_ACL", acl );

// バインドするオブジェクトのリストを作成する
List<DbjObj> bindobjlist = new ArrayList<DbjObj>();
bindobjlist.add( docspc.createObjConnection( oiid1 ));
bindobjlist.add( docspc.createObjConnection( oiid2 ));

// パブリックACLを作成する
DbjObj pacl = docspc.createPublicACL(
                           "edmClass_PublicACL",
                           props,
                           bindobjlist );
```

(2) パブリック ACL のバインド

ここでは，文書管理オブジェクトから，パブリック ACL にバインドする方法について説明します。パブリック ACL のバインドは，DbjObj#bindPublicACL メソッドで実行します。

文書管理オブジェクトからパブリック ACL をバインドする例を示します。この例では，バインドするパブリック ACL を検索で取得して，検索結果として得られたパブリック ACL をすべて一括してバインドします。

// 文書管理オブジェクトからパブリックACLをバインドする例

```
  // docspc：DbjDocSpaceインターフェース
  // obj：DbjObjインターフェース
```

```
  //   （文書管理オブジェクトのインターフェース）

// パブリックACLを検索する
DbjResultSet result = docspc.executeSearch(
      "SELECT dmaProp_OIID  FROM edmClass_PublicACL WHERE...",
      null,
      null );

// 検索結果からDbjObjListインターフェースを取得する
DbjObjList objlist = docspc.createObjList( result , 0 , null );

// 検索結果として得られた複数のパブリックACLを一括してバインドする
obj.bindPublicACL( objlist );
```

(3) バインドしているパブリック ACL の一覧取得

ここでは，文書管理オブジェクトにバインドしているパブリック ACL の一覧を取得する方法について説明します。一覧の取得は，DbjObj#getPublicACLList メソッドで実行します。

バインドしているパブリック ACL の一覧を取得する例を示します。

// バインドしているパブリックACLの一覧を取得する例

```
  // obj：DbjObjインターフェース
  //   （文書管理オブジェクトのインターフェース）

// バインドしているパブリックACLの一覧を取得する
DbjObjList objlist = obj.getPublicACLList ( null );

// パブリックACLのOIIDを出力する
for ( int i = 0 ; i<objlist.size() ; i++ ){
   System.out.println( "oiid = " + objlist.getObj(i).getOiid() );
}
```

(4) バインドするパブリック ACL の変更

ここでは，文書管理オブジェクトにバインドしているパブリック ACL を変更する方法について説明します。バインドしているパブリック ACL は，dbrProp_PublicACLIds プロパティに設定されています。このプロパティは VARRAY 型のプロパティです。要素は，DbjPublicACLIdElm インターフェースまたは DbjPropSet インターフェースで操作します。

文書管理オブジェクトからバインドしているパブリック ACL を変更する例を示します。ここでは，バインドするパブリック ACL を，一つ削除して一つ追加します。

// 文書管理オブジェクトからバインドしているパブリックACLを変更する例

```
  // obj：DbjObjインターフェース
  //   （文書管理オブジェクトのインターフェース）

// バインドしているパブリックACLを要素とした可変長配列を取得する
Set<String> propdef = new HashSet<String>();
propdef.add( "dbrProp_PublicACLIds" );
obj.readProperties( propdef );

DbjVArray varray = obj.propSet()
  .getVArrayRef( "dbrProp_PublicACLIds" );

// 一つ目の要素を可変長配列から削除する
varray.remove(0);

// 追加するパブリックACLのOIIDを要素に設定する
DbjPublicACLIdElm elm = factory.createPublicACLIdElm();
elm.setId( pacloiid );
```

```
// 要素を可変長配列の末尾に追加する
varray.addPropSet( elm.propSet() );

// dbrProp_PublicACLIdsプロパティを更新する
obj.writeProperties();
```

(5) パブリック ACL をバインドしている文書管理オブジェクトの一覧取得

ここでは，パブリック ACL がバインドされている文書管理オブジェクトの一覧を取得する方法について説明します。一覧の取得は，DbjObj#getBindObjectList メソッドで実行します。

パブリック ACL をバインドしている文書管理オブジェクトの一覧取得の例を示します。

// パブリックACLをバインドしている文書管理オブジェクトの一覧取得

```
  // obj：DbjObjインターフェース
  //   （パブリックACLのインターフェース）

// バインドされている文書管理オブジェクトとの一覧を取得する
DbjObjList objlist = obj.getBindObjectList(
             null,
             DbjDef.OBJTYPE_DOC, // 文書を取得
             null );

// バインドされている文書管理オブジェクトのOIIDを出力する
for ( int i = 0; i<objlist.size() ; i++ ){
   System.out.println( "oiid = " + objlist.getObj(i).getOiid() );
}
```

(6) パブリック ACL のアンバインド

ここでは，パブリック ACL をアンバインドする方法について説明します。

パブリック ACL のアンバインドは，DbjObj#unbindPublicACL メソッドで実行します。

文書管理オブジェクトからパブリック ACL をアンバインドする例を示します。

// 文書管理オブジェクトからパブリックACLをアンバインドする例

```
  // docspc：DbjDocSpaceインターフェース
  // obj：DbjObjインターフェース
  //    （文書管理オブジェクトのインターフェース）

// アンバインドするパブリックACLのリストを作成する
List<DbjObj> unbindlist = new ArrayList<DbjObj>();
unbindlist.add( docspc.createObjConnection( pacloiid ) );

// アンバインドを実行する
obj.unbindPublicACL( unbindlist );
```

1.12 メタ情報の取得

この節では，メタ情報の取得について説明します。

1.12.1 メタ情報を取得するインターフェースの機能

DocumentBroker で管理されているメタ情報は，メタクラスのメソッドで取得できます。

メタクラスとは，次のインターフェース群の総称です。

● DbjMetaManager インターフェース
DbjFactory0200#getMetaManager メソッドで取得できます。

● DbjMeta インターフェース
次のどちらかのメソッドで取得できます。
- DbjMetaManager#getMeta メソッド
- DbjDocSpace#getMeta メソッド

● DbjClassDesc インターフェース
次のどれかのメソッドで取得できます。
- DbjClassDesc#getSuperClass メソッド
- DbjMeta#getClassDesc メソッド
- DbjPropDesc#getVArrayClass メソッド

● DbjPropDesc インターフェース
次のどちらかのメソッドで取得できます。
- DbjMeta#getPropDesc メソッド
- DbjClassDesc#getProperties メソッド
取得するリストの要素として取得できます。

メタ情報は，文書空間に定義されている DocumentBroker クラスおよび文書管理オブジェクトのプロパティの定義情報です。

DocumentBroker クラスについてのメタ情報を，**クラスディスクリプション**といいます。クラスディスクリプションは，クラスごとに定義されています。文書管理オブジェクトのプロパティについてのメタ情報を，**プロパティディスクリプション**といいます。プロパティディスクリプションは，プロパティごとに定義されています。

メタクラスのインターフェースでは，クラスディスクリプションおよびプロパティディスクリプションの定義を参照できます。また，このほかのメタ情報として，文書空間識別子や，レンディションタイプと拡張子の対応などの情報も参照できます。

1.12.2 メタ情報の取得

ここでは，メタ情報の取得の例を示します。

(1) 文書空間の情報の取得

文書空間の情報は，DbjMeta インターフェースのメソッドで取得できます。取得できるのは，次の情報です。

● 文書空間識別子

● 文書管理オブジェクトのプロパティのデータ型

● 拡張子に対応するレンディションタイプおよびレンディションタイプに対応する拡張子

文書空間の情報を取得する例を示します。

// 文書空間の情報を取得する例

```
// DbjMetaManagerインターフェースを取得する
DbjMetaManager metamgr = DbjFactory0200.getMetaManager();

DbjMeta meta = metamgr.getMeta( docspaceid );

// Authorプロパティのデータ型を取得する
int datatype = meta.getPropDataType( "Author" );

// 拡張子に対応するレンディションタイプを取得する
String rendtype = meta.getRenditionType( "sgm" );

// レンディションタイプに対応する拡張子を取得する
String ext = meta.getExtFromRenditionType( "text/html" );
```

(2) クラスディスクリプションおよびプロパティディスクリプションの取得

クラスディスクリプションは，DbjClassDesc インターフェースを使用して取得します。

DbjClassDesc インターフェースを使用して取得できるのは，次の情報です。

● DocumentBroker クラス名

● DocumentBroker クラスに定義されているプロパティのプロパティディスクリプション

● スーパークラスのクラスディスクリプション

● サブクラスのクラスディスクリプション

プロパティディスクリプションは，DbjPropDesc インターフェースのメソッドで取得します。

DbjPropDesc インターフェースを使用して取得できるのは，次の情報です。

● プロパティ名

● プロパティのデータ型

● VARRAY 型のプロパティのクラスディスクリプション※

注※

VARRAY 型のプロパティの要素は，データベース上では，DocumentBroker クラスの edmClass_Struct クラスのオブジェクトの情報として格納されています。このため，VARRAY 型のプロパティの各要素のメタ情報は，クラスディスクリプションとして定義されています。

まず，DbjClassDesc インターフェースの使用例を示します。

// DbjClassDescインターフェースの使用例

```
DbjMetaManager metamgr = DbjFactory0200.getMetaManager();
DbjMeta meta = metamgr.getMeta( docspaceid );

// Documentクラスのクラスディスクリプションを取得する
//  （DbjClassDescインターフェースを取得する）
DbjClassDesc cd = meta.getClassDesc( "Document" );
```

```
// サブクラスのクラスディスクリプションを取得する
List<DbjClassDesc> sublist = cd.getSubClasses();

// サブクラスのクラス名を出力する
for(int i=0;i<sublist.size();i++){
  DbjClassDesc tmp = (DbjClassDesc)sublist.get(i);
  System.out.println("sub class name["+i+"]=" + tmp.getName());
}
```

次に，DbjPropDesc インターフェースの使用例を示します。

// DbjPropDescインターフェースの使用例

```
  // meta：DbjMetaインターフェース

// Documentクラスのクラスディスクリプションを取得する
DbjClassDesc cd = meta.getClassDesc( "Document" );

// プロパティディスクリプションを取得する
List<DbjPropDesc> proplist = cd.getProperties();

// プロパティ名およびプロパティのデータ型を出力する
for(int i=0;i<proplist.size();i++){
  DbjPropDesc tmp = (DbjPropDesc)proplist.get(i);
  System.out.println("prop name["+i+"]=" + tmp.getName()
   + ",type " + tmp.getDataType() );
}
```

1.13 例外処理

ここでは，例外処理について説明します。

1.13.1 例外の種類

DocumentBroker クラスライブラリの処理でエラーが発生した場合，例外がスローされます。ユーザアプリケーションプログラムでは，try ～ catch 節によって，エラー処理を実行してください。

DocumentBroker クラスライブラリで発生する例外には，次のような種類があります。

● ユーザアプリケーションプログラムを操作したエンドユーザの操作ミスによって発生する例外

● ユーザアプリケーションプログラム開発時に発生する例外

● DocumentBroker クラスライブラリが動作する環境が不正な場合に発生する例外

● そのほかの例外

DocumentBroker クラスライブラリの処理で発生する例外の詳細な内容については，「8. 例外クラス詳細」を参照してください。

1.13.2 例外の種類ごとの処理方法

ここでは，例外の種類ごとの処理方法について説明します。

(1) エンドユーザの操作ミスによって発生する例外の処理

例外クラスでは，主にエンドユーザの操作ミスによって発生する例外に対応するクラスを提供しています。例えば，ログイン時に入力したユーザ識別子またはパスワードが不正だった場合の処理や，文書管理オブジェクトのプロパティに不正な値を設定しようとした場合の処理などを，業務処理部分から切り分けてコーディングすることができます。

なお，サブクラスとして定義されているクラスで表されるエラーをすべて区別して処理する必要はありません。ユーザアプリケーションプログラムで実現したい例外処理に応じて，クラスを使い分けてください。

例えば，エンドユーザがコンテンツをダウンロードする時，次のような例外が発生する可能性があります。

- ダウンロードしようとしたファイルにアクセス権がない
- 指定したファイルが存在しない

これらの例外に対して，要因ごとに異なるエラー処理を実行するユーザアプリケーションプログラムを作成したい場合は，それぞれの要因に対応する例外クラスを使用して例外処理を実行します。この場合は，DbjFileAccessException クラスのオブジェクトをキャッチする処理と，DbjFileNotFoundException クラスのオブジェクトをキャッチする処理を分けてコーディングします。

また，この二つのどちらの例外が発生した場合でも，ファイルにアクセスできないことだけを表示するだけでよい場合などには，二つの例外のスーパークラスである DbjIOException クラスのオブジェクトをキャッチする処理をコーディングします。

このように，実行したい例外処理ごとに，必要に応じて例外クラスのスーパークラスまたはサブクラスを使用してください。

ここでは，ログイン処理で指定したユーザ識別子またはパスワードが不正な場合に発生する例外をキャッチする例を示します。

// 例外処理の例

```
DbjSession sess = null;
DbjDocSpace docspc = null;
try {
  sess = DbjFactory0200.getFactory().createSession( docspaceid );

  // ログイン処理
  docspc = sess.login( "hitachi", "passwd" );

} catch(DbjNotAuthenticatedException e) {
  // 認証エラーが発生した場合
  System.out.println("Not authenticated");

} catch(DbjException e) {
  // そのほかのエラーが発生した場合
  .....
}
```

(2) ユーザアプリケーションプログラム開発時に発生する例外

ユーザアプリケーションプログラム開発時には，コーディングのミスによって例外が発生する可能性があります。この場合も，例外クラスのオブジェクトがスローされます。

ただし，コーディングのミスによって発生する例外は，デバッグによって解消される例外です。このため，この例外に対して，ユーザアプリケーションプログラムに細かい例外処理をコーディングする必要はありません。

コーディングのミスによって発生する例外は，すべて DbjException クラスのオブジェクトとしてスローされます。したがって，ユーザアプリケーションプログラム開発時の例外処理では DbjException クラスのオブジェクトをキャッチするコーディングをしておいて，必要に応じてメッセージ情報を参照してデバッグを実行してください。

メッセージ情報については，マニュアル「DocumentBroker Version 5　メッセージ」を参照してください。

(3) DocumentBroker クラスライブラリが動作する環境が不正な場合に発生する例外

データベースでエラーが発生した場合や，環境が不正な場合に発生したエラーは，DbjException クラスのサブクラスである DbjDBException クラスのオブジェクトとしてスローされます。ユーザアプリケーションプログラムでこれらの例外をキャッチした場合には，DocumentBroker クラスライブラリの動作環境を見直してください。

(4) そのほかの例外

これまでに説明した例外以外に，メモリ不足や内部エラーなど，ユーザアプリケーションプログラムでは対処できない例外があります。これらの例外は，java.lang.Error クラスのサブクラスである DbjError クラスのオブジェクトで表されます。このエラーは，直接スローされることはありません。したがって，キャッチする必要もありません。

1.14 ライブラリ情報の取得

この節では，ライブラリ情報取得クラスの機能について説明します。ライブラリ情報の取得クラスとは，DbjLibInfo クラスのことです。

DocumentBroker のバージョン情報は，DbjLibInfo#getVersion メソッドを実行して取得できます。

バージョンは，10 進数で表されます。下位 2 けたは，マイナーバージョン，その上位のけたはメジャーバージョンを表します。例えば，バージョンが「03-00」の場合は，300 が取得できます。

製品のバージョン情報を取得する例を示します。

// 製品のバージョン情報を取得する例

```
System.out.println( "version=" + DbjLibInfo.getVersion() );
```

1.15 ユーザアプリケーションプログラムのトレース情報の出力

この節では，DocumentBroker クラスライブラリを使用してユーザアプリケーションプログラムを開発・運用する上で必要なトレース情報の出力方法について説明します。

1.15.1 トレース情報を出力するメソッドの機能

トレース情報は，次の表に示すトレースクラスのメソッドで出力できます。トレースクラスとは，DbjTrace クラスのことです。トレースクラスの詳細については，

「11. トレースクラス詳細」を参照してください。

表 1-10 トレースクラスのメソッドの機能

メソッド名	機能
トレース情報を出力するメソッド	
arg	パラメタ情報の出力
call	外部 API の呼び出し情報の出力
enter	メソッドの入口情報の出力
error	エラー情報の出力
exit	メソッドの出口情報の出力
hint	下位のメソッドのエラー情報の出力
init	トレースクラスの初期化
msg	メッセージの出力
returned	外部 API からのリターン情報の出力

1.15.2 トレース情報の出力先

トレース情報は，次に示す出力先に出力できます。

表 1-11 トレース情報の出力先

出力先	出力可否
コマンドプロンプト（標準出力・標準エラー出力）	○
アプリケーショントレースファイル	○
アプリケーションエラーログファイル	○

（凡例）
○：トレース情報を出力します。
－：該当しません。

出力する情報の種類や，使用目的に応じてトレース情報の出力先を決定してください。レース情報の出力先と出力する情報の目安を次の表に示します。

表 1-12 トレース情報の出力先と出力する情報の目安

トレース情報の出力先	出力する情報の目安	使用するメソッド	定数の値※1
コマンドプロンプト	• DocumentBroker クラスライブラリを使用したユーザアプリケーションプログラムの開始および終了時の情報 • 重要なメソッドの入り口および出口の情報 • 外部から与えられた情報 • DocumentBroker クラスライブラリを使用したユーザアプリケーションプログラムを利用するユーザが対処できないエラーの情報	• arg • enter • error • exit • msg	PROMPT
アプリケーショントレースファイル	• DocumentBroker クラスライブラリを使用したユーザアプリケーションプログラムの開始および終了時の情報 • 重要なメソッドの入り口および出口の情報 • メソッドの入り口および出口の情報 • 外部から与えられた情報 • 外部 API の呼び出し時およびリターン時の情報 • メソッドの引数の情報 • 下位メソッドのエラーの情報 • DocumentBroker クラスライブラリを使用したユーザアプリケーションプログラムを利用するユーザが対処できるエラーの情報 • DocumentBroker クラスライブラリを使用したユーザアプリケーションプログラムを利用するユーザが対処できないエラーの情報	• arg • call • enter • error • exit • hint • msg • returned	TRACE
アプリケーションエラーログファイル	• DocumentBroker クラスライブラリを使用したユーザアプリケーションプログラムを利用するユーザが対処できるエラーの情報 • DocumentBroker クラスライブラリを使用したユーザアプリケーションプログラムを利用するユーザが対処できないエラーの情報	• arg • error • msg	ERRORLOG

注※1

各メソッドの引数 output にこれらの定数を指定して出力先を決定します。これらの定数は DbjTraceDef クラスのメンバとして定義されています。

1.15.3 トレース情報の出力形式

トレース情報は，次に示す形式で出力されます。

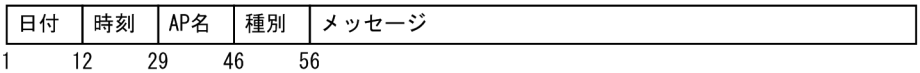
各項目間には1文字の空白が入ります。

(1) コマンドプロンプトの出力形式

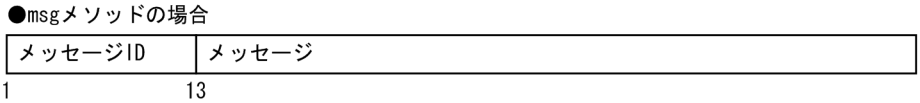
コマンドプロンプトの出力形式を次の図に示します。

図 1-20 コマンドプロンプトの出力形式

●msgメソッド以外のメソッドの場合



●msgメソッドの場合



説明

出力形式の各項目では，次の表に示す情報が出力されます。

表 1-13 コマンドプロンプトの出力項目

項目名	出力される情報
日付	トレース情報の取得日付が yyyy/mm/dd の形式で出力されます。 yyyy/mm/dd 形式の yyyy には西暦年号（4 けた），mm には月（2 けた），dd には日（2 けた）が出力されます。
時刻	トレース情報の取得時刻が hh:mm:ss.sss の形式で出力されます。 hh:mm:ss.sss 形式の hh には時（2 けた），mm には分（2 けた），ss には秒（2 けた），sss にはミリ秒（3 けた）が出力されます。
AP 名	ユーザアプリケーションプログラムの名称が出力されます。名称が 16 バイトを超えている場合は，16 バイト以内の文字列が出力されます。
種別	トレース情報を出力したメソッドの種別が出力されます。 実行されたメソッドと種別に出力される文字列の対応を表 1-13 に示します。
メッセージ	メッセージが出力されます。種別ごとに異なるメッセージが出力されます。
メッセージ ID	メッセージ ID が出力されます。メッセージ ID が 11 バイトを超えている場合は，11 バイト以内の文字列が出力されます。

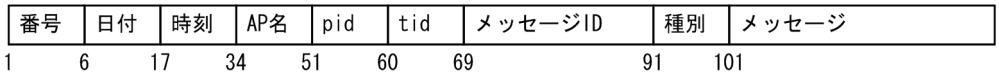
表 1-14 実行されたメソッドと種別に出力される文字列の対応

実行されたメソッド	種別に出力される文字列
arg	“ △△△△ “
enter	“ FB △△ “
exit	“ FE △△ “
error	“ ER △△ ““ または EC △△ “

(2) アプリケーショントレースファイルまたはアプリケーションエラーログファイルの出力形式

アプリケーショントレースファイル，またはアプリケーションエラーログファイルの出力形式を次の図に示します。

図 1-21 アプリケーショントレースファイルまたはアプリケーションエラーログファイルの出力形式



説明

出力形式の各項目では，次の表に示す情報が出力されます。

表 1-15 アプリケーショントレースファイルまたはアプリケーションエラーログファイルの出力項目

項目名	出力される情報
番号	4 けたの通番が出力されます。
日付	トレース情報の取得日付が yyyy/mm/dd の形式で出力されます。 yyyy/mm/dd 形式の yyyy には西暦年号（4 けた），mm には月（2 けた），dd には日（2 けた）が出力されます。
時刻	トレース情報の取得時刻が hh:mm:ss.sss の形式で出力されます。 hh:mm:ss.sss 形式の hh には時（2 けた），mm には分（2 けた），ss には秒（2 けた），sss にはミリ秒（3 けた）が出力されます。
rsvd	予備領域 (4 バイト) の空白が出力されます。

項目名	出力される情報
AP 名	ユーザアプリケーションプログラムの名称が出力されます。名称が 16 バイトを超えている場合は，16 バイト以内の文字列が出力されます。
pid	プロセス ID が出力されます。
tid	スレッド ID が出力されます。
メッセージ ID	メッセージ ID が出力されます。メッセージ ID が 11 バイトを超えている場合は，11 バイト以内の文字列が出力されます。
種別	トレース情報を出力したメソッドの種別が出力されます。 実行されたメソッドと種別に出力される文字列の対応を表 6-13 に示します。
メッセージ	メッセージが出力されます。種別ごとに異なるメッセージが出力されます。
CRLF	レコード終端符号 (0x0D,0x0A) です。

1.15.4 トレース情報の出力範囲

次に示す出力先にトレース情報を出力するときは，トレースレベルを指定できます。トレースレベルを指定すると，トレース情報をどの範囲まで出力するかを選択できます。

- コマンドプロンプト
- アプリケーショントレースファイル

(1) トレースレベルを指定する目的

システムが開発段階にあるか，または運用段階にあるかによって，必要なトレース情報が異なります。例えば，システムが運用段階にある場合，メソッドの入り口および出口の情報は重要な情報ではありません。しかし，システムが開発段階にある場合や，障害の発生個所を特定する場合などは，メソッドの入り口および出口の情報は重要な情報になります。

このように，状況に応じて必要な情報が異なるため，トレースレベルを指定し，取得する情報の範囲を選択してください。

トレースレベルを指定するメリットを次に示します。

- むだな出力情報を少なくすることで，重要な情報を整理しやすくする
- 出力先のファイル容量を削減できる

用途や，目的を考えないですべての情報を出力すると，重要な情報が探せない，または重要な情報が上書きされたなどの不都合が生じることがあります。

(2) 各トレースレベルで出力する情報の目安

トレースレベルには，次の表に示すレベルがあります。各トレースレベルの用途に応じて出力する情報を決定してください。

表 1-16 トレースレベルで出力する情報の目安

用途	APTraceLevel の値※1	level 引数の値※2	出力する情報の目安
障害監視	0	ERROR	**●出力先がコマンドプロンプトの場合** ERROR レベルでは，次に示す情報を出力するようにします。 • ユーザアプリケーションプログラムの開始および終了時の情報 • ユーザアプリケーションプログラムを利用するユーザが対処できないエラーの情報 **●出力先がアプリケーショントレースファイルの場合** ERROR レベルでは，すべてのエラー情報を出力するようにします。

用途	APTraceLevel の値※1	level 引数の値※2	出力する情報の目安
通常運用	10	MANAGE	**●出力先がコマンドプロンプトの場合** MANAGE レベルでは，ERROR レベルで出力される情報をすべて出力するようにします。 **●出力先がアプリケーショントレースファイルの場合** MANAGE レベルでは，ERROR レベルで出力される情報をすべて出力するようにします。それに加えて次に示す情報を出力するようにします。 • 外部 API の呼び出しおよびリターン時の情報 • 重要なメソッドの入り口および出口の情報
障害調査	20	HINT	**●出力先がコマンドプロンプトの場合** HINT レベルでは，MANAGE レベルで出力される情報をすべて出力するようにします。それに加えて次に示す情報を出力するようにします。 • 重要なメソッドの入り口および出口の情報 • 外部から与えられた情報 **●出力先がアプリケーショントレースファイルの場合** HINT レベルでは，MANAGE レベルで出力される情報をすべて出力するようにします。それに加えて次に示す情報を出力するようにします。 • 下位メソッドのエラーの情報 • 外部から与えられた情報 • 重要なメソッドの引数の情報
デバッグ	30	DEBUG	**●出力先がコマンドプロンプトの場合** DEBUG レベルでは，HINT レベルで出力される情報をすべて出力するようにします。 **●出力先がアプリケーショントレースファイルの場合** DEBUG レベルでは，HINT レベルで出力される情報をすべて出力するようにします。それに加えて次に示す情報を出力するようにします。 • メソッドの入り口および出口の情報 • メソッドの引数の情報

注※1

動作環境定義ファイルの値です。動作環境定義ファイルについては，マニュアル「DocumentBroker Version 5　システム導入・運用ガイド」を参照してください。

注※2

各メソッドの引数 level にこれらの定数を指定して出力範囲を選択します。これらの定数は DbjTraceDef クラスのメンバとして定義されています。

1.16 マルチスレッド環境での注意事項

1 セッションを複数スレッドで使用しないでください。使用した場合，メソッドはエラーになります。

2 edmSQL の文法

この章では，edmSQL の文法について説明します。
なお，edmSQL で使用する文字列の長さやデータ型などは，データベースの制限に従います。DocumentBroker で使用できるデータベースは，HiRDB です。HiRDB の SQL で指定できる文字列の長さやデータ型の制限については，マニュアル「HiRDB　SQL リファレンス」を参照してください。

2.1 edmSQL の文法の概要

この節では，edmSQL の文法の概要として，次の項目について説明します。

● 字句規則

● 構文規則

2.1.1 字句規則

edmSQL では，使用できる字句が定義されています。

定義されているのは，< 区切り文字 >，< トークン >，< キーワード >，< リテラル >，< 識別子 >，< 名前 >，< 特殊なプロパティ >，<? パラメタ > および <OIID> です。

字句規則の詳細については，「2.4　字句規則」を参照してください。

2.1.2 構文規則

edmSQL 文で指定できる構文は，SELECT 文です。edmSQL では，SELECT 文の構文規則が定義されています。

(1) 構文の概要

ここでは，edmSQL 文の構文の概要について説明します。

(a) 検索の実行単位

検索は，edmSQL プログラム単位で実行されます。edmSQL プログラムは，一つの edmSQL 文によって構成されている，1 回の検索実行メソッドによって実行される検索条件です。

edmSQL 文として指定できるのは，問い合わせ文（SELECT 文）だけです。

検索の実行単位については，「2.5　検索の実行単位の構文規則」を参照してください。

(b) 問い合わせ文（SELECT 文）

問い合わせは，問い合わせ文によって表現します。

問い合わせ文には，SELECT 句，FROM 句，WHERE 句および ORDER BY 句が指定できます。最も単純な問い合わせ文は，SELECT 句と FROM 句から構成されます。

それぞれの句は，次のような要素で構成されています。

● 値（スカラー式）

● 述語

● 関数

● データ操作

問い合わせ表現については，「2.6　問い合わせ式の構文規則」を参照してください。

(2) SELECT 文の構成要素

SELECT 文は，次のような要素で構成されます。

(a) 値（スカラー式）

値は，スカラー式として指定します。スカラー式によって，プロパティの指定やルーチンの起動，数値関数および集合関数によって得られる値を表現します。

値を表現するスカラー式の詳細については，「2.7　スカラー式表現の構文規則」を参照してください。

(b) 述語

FROM 句の ON 条件や，WHERE 句に指定する検索条件は，述語によって表現します。述語では，比較述語，論理述語，Between 述語，In 述語，Like 述語，Null 述語および Exists 述語で構成される検索条件が指定できます。

述語の結果は，「真」，「偽」または「不定」のどれかで得られます。

述語の詳細については，「2.8　述語の構文規則」を参照してください。

(c) 関数

DocumentBroker では，基本的な SQL の文法では表現できない edmSQL での拡張機能を，関数で表現します。

関数には，HiRDB Text Search Plug-in，または HiRDB XML Extension の機能を使用した検索を実現する関数と，DocumentBroker で扱うオブジェクトや文字列の変換機能を実現する関数があります。

関数の詳細については，「2.9　関数指定の構文規則」を参照してください。

(d) データ操作

edmSQL で実行できるデータ操作とは，ORDER BY 句による検索結果の並べ替えです。

データ操作の詳細については，「2.10　データ操作の構文規則」を参照してください。

2.2 表記規則

この節では，edmSQL の文法を説明する際に使用する文法規則について説明します。

edmSQL の構文規則は，BNF 表記を使用して説明します。

BNF 表記について，次に示します。

- <a> ::= <b><c> は，構文要素 <a> は，構文要素 <b> および <c> から構成されることを意味します。
- 特殊記号およびキーワードは，終端記号であるため，このマニュアルに記載してあるとおりに記述する必要があります。「<」と「>」で囲まれた構文要素は，終端記号ではない構文要素です。
- 「|」（ストローク）は，選択肢の分割を意味します。A | B | C の場合は，選択肢として，A，B，または C を選択することを意味します。
- 「〔」と「〕」で囲まれた要素は，オプションであり，省略できます。
- 「{」と「}」で囲まれている要素は，その内部の「|」で区切られた要素の中からどれか一つを選択することを意味します。
- 「...」は，直前の構文要素の 1 回以上の繰り返しを意味します。
- 「!!」は，注釈の開始を意味します。注釈は改行で終わります。

2.3 使用できるデータ型と演算子・関数・述語の関係

この節では，edmSQL 文で使用できるデータ型と演算子・関数・述語との関係について説明します。

2.3.1 edmSQL で使用できるデータ型

edmSQL 文で使用できるデータ型は，次のとおりです。

- 論理型
- 整数型
- オブジェクト型
- VariableArray 型
- 文字列型
- バイナリ型

このうち，バイナリ型のデータについては，DocumentBroker クラスライブラリでは String 型のデータとして扱います。

使用できるデータ型ごとに，次の項目について説明します。

- edmSQL 文の表現形式
- 演算子
- 関数
- 述語

2.3.2 論理型

論理型は，値と評価の二つの概念を持ちます。それぞれに対して，TRUE，FALSE，UNKNOWN の三つの表現があります。

ここでは，論理型の概念，論理型のデータ型を使用する表現形式，演算子，関数および述語について説明します。また，DocumentBroker クラスライブラリで扱うデータ型との対応についても説明します。

(1) 論理型の概念

ここでは，論理型の値および評価について説明します。

(a) 値としての論理型

値として扱う論理型のデータ型は，物理的実体として存在します。これは，Boolean 型のプロパティの値または定数値として表現します。

値には，TRUE，FALSE，UNKNOWN の三つの表現があります。edmSQL では，これらの表現が，それぞれ整数値で実装されています。これは，SQL の概念にはない，edmSQL 独自の対応付けです。

値としての論理型の表現と定数値の対応を，次の表に示します。

表 2-1 値としての論理型の表現と定数値の対応

表現	定数値
TRUE	1
FALSE	0
UNKNOWN	2

(b) 評価としての論理型

評価として扱う論理型のデータ型は，物理的な実体を持ちません。検索条件や検索関数の評価結果として返ってくるデータ型です。

評価には，真，偽，不定の三つの値があります。表現と評価の値は，SQL と同じ概念で対応付けられます。

評価としての論理型の表現と値との対応を，次の表に示します。

表 2-2 評価としての論理型の表現と定数値の対応

表現	評価の値
TRUE	真
FALSE	偽
UNKNOWN	不定

(2) 表現形式

ここでは，論理型の値および評価の表現形式について説明します。

(a) 論理型の値の表現形式

値として扱う論理型の表現形式を次の表に示します。

表 2-3 論理型の値の表現形式

値	定義
リテラル	TRUE FALSE UNKNOWN
プロパティ	Boolean 型のプロパティ 格納値：0，1，2
変数として指定できる値	? パラメタ

(b) 論理型の評価の表現形式

評価として扱う論理型を，直接表現する形式はありません。

(3) 演算子・関数

ここでは，論理型の値および評価を被演算子として扱う演算子および関数について説明します。

(a) 論理型の値を被演算子として扱う演算子・関数

論理型の値を被演算子として扱う演算子・関数はありません。

論理型の評価を被演算子として扱う演算子を，次の表に示します。

表 2-4 論理型の評価を被演算子として扱う演算子

演算子の種類	演算子
論理演算子	AND OR NOT

(b) 論理型の評価を被演算子として扱う演算子・関数

論理型の値を被演算子として扱う演算子・関数はありません。

(4) 述語

ここでは，論理型の値および評価を評価対象とする述語について説明します。

(a) 論理型の値を評価対象とする述語

評価対象が，論理型の値である述語を次の表に示します。

表 2-5 評価対象が論理型の値である述語

述語の種類	述語
比較述語	= <>
In 述語	IN

(b) 論理型の評価を評価対象とする述語

評価対象が，論理型の評価である述語を次の表に示します。

表 2-6 評価対象が論理型の評価である述語

述語の種類	述語
論理述語	IS TRUE

(5) DocumentBroker クラスライブラリで扱うデータ型との対応

ここでは，edmSQL で扱う論理型の値および評価と，DocumentBroker クラスライブラリで扱うデータ型との対応について説明します。

(a) DocumentBroker クラスライブラリで扱うデータ型との対応

edmSQL で扱う論理型の値と DocumentBroker クラスライブラリで扱うデータ型との対応を，次の表に示します。

表 2-7 edmSQL の論理型の値と DocumentBroker クラスライブラリで扱うデータ型との対応

edmSQL のデータ型	DocumentBroker クラスライブラリで扱うデータ型	値
論理型（値）	int（定数定義クラスの値）	DbjDef.DMA_TRUE（=1） DbjDef.DMA_FALSE（=0） DbjDef.DMA_UNKNOWN（=2）

(b) 論理型の評価と DocumentBroker クラスライブラリで扱うデータ型との対応

論理型の評価に，DocumentBroker クラスライブラリで扱うデータ型と対応するものはありません。

2.3.3 整数型

整数型で表現するのは，4 バイト（32 ビット）で表現できる符号付き整数値，または符号と数字文字列で表現する，整数リテラルです。

次に，整数型のデータ型を使用する表現形式，演算子，関数および述語について説明します。また，DocumentBroker クラスライブラリの値との対応についても説明します。

(1) 表現形式

整数型の値の表現形式を次の表に示します。

表 2-8 整数型の値の表現形式

値の種類	定義
リテラル	符号および数字列
プロパティ	Integer32 型のプロパティ 格納値：-2,147,483,648 ～ 2,147,483,647
変数として指定できる値	? パラメタ

(2) 演算子・関数

整数型の値を被演算子または引数として扱う演算子および関数を次の表に示します。

表 2-9 整数型の値を被演算子または引数とする演算子・関数

演算子・関数の種類	演算子・関数
整数の四則演算子	+ - * /
単項演算子	+ -
数値関数	ABS()

(3) 述語

評価対象が整数型の値である述語を次の表に示します。

表 2-10 整数型の値を評価対象とする述語

述語の種類	述語
比較述語	= < > <= >= <>
In 述語	IN
Between 述語	BETWEEN

(4) DocumentBroker クラスライブラリで扱うデータ型との対応

DocumentBroker クラスライブラリで扱うデータ型との対応を次の表に示します。

表 2-11 edmSQL の整数型と DocumentBroker クラスライブラリで扱うデータ型との対応

edmSQL のデータ型	DocumentBroker クラスライブラリで扱うデータ型	値
整数型	int，Integer	-2,147,483,648 ～ 2,147,483,647

2.3.4 オブジェクト型

オブジェクトリファレンスとは，DocumentBroker オブジェクトへのリファレンスを示すプロパティの値です。例えば，dmaProp_ParentContainer プロパティの値がこれに当たります。

オブジェクト型の値のうち，オブジェクトリファレンスは検索条件に直接指定できません。検索で指定する場合には，オブジェクトリファレンスが示す実体である DocumentBroker オブジェクトの OIID 文字列を指定します。例えば，dmaProp_ParentContainer プロパティの値を指定したい場合，検索条件には dmaProp_ParentContainer プロパティが参照している実体の DocumentBroker オブジェクト（Container オブジェクトなど）の OIID を表す文字列を指定します。

この OIID 文字列の値は，検索実行時には，次のどちらかの方法でオブジェクトリファレンスの値に変換する必要があります。

● objref 関数で変換する
edmSQL 文にオブジェクトリファレンスの実体である OIID を直接指定する場合の変換方法です。
objref 関数の詳細については，「2.9.3(4) <objref 関数 > の詳細」を参照してください。

● <? パラメタ > を使用して変換する
edmSQL 文に <? パラメタ > を指定しておく場合の変換方法です。<? パラメタ > に OIID 文字列を設定する方法については，「1.7 パラメタの操作」を参照してください。

また，検索結果としてオブジェクト型の値を取得した場合は，oiidstr 関数を使用して，そのオブジェクト型の値が表すオブジェクトリファレンスの実体であるオブジェクトの OIID が取得できます。oiidstr 関数の詳細については，「2.9.3(3) <oiidstr 関数 > の詳細」を参照してください。

次に，オブジェクト型のデータ型を使用する表現形式および述語について説明します。また，DocumentBroker クラスライブラリで扱うデータ型との対応についても説明します。

なお，オブジェクト型の値を使用する演算子および関数はありません。

(1) 表現形式

オブジェクト型の値の表現形式を，次の表に示します。

表 2-12 オブジェクト型データの表現形式

値の種類	定義
プロパティ	Object 型のプロパティ

(2) 述語

評価対象がオブジェクト型の値である述語を次の表に示します。

表 2-13 オブジェクト型の値を評価対象とする述語

述語の種類	述語
比較述語	=

述語の種類	述語
In 述語	IN

なお，比較述語，In 述語では，オブジェクトリファレンス同士を比較できます。

(3) DocumentBroker クラスライブラリで扱うデータ型との対応

オブジェクト型を扱う場合は，< 変換関数 > を使用するか，<? パラメタ > を使用してください。

2.3.5 VariableArray 型

(1) 表現形式

VariableArray 型の値の表現形式を，次の表に示します。

表 2-14 文字列型の値の表現形式

値の種類	定義
プロパティ	VariableArray 型のプロパティ

(2) 述語

評価対象が VariableArray 型の値である述語を次の表に示します。

表 2-15 VariableArray 型の値を評価対象とする述語

述語の種類	述語
比較述語	=
In 述語	IN

(3) DocumentBroker クラスライブラリで扱うデータ型との対応

DocumentBroker クラスライブラリで扱うデータ型との対応を次の表に示します。

表 2-16 edmSQL の VariableArray 型と DocumentBroker クラスライブラリで扱うデータ型との対応

edmSQL のデータ型	DocumentBroker クラスライブラリで扱うデータ型	値
VariableArray 型	DbjVArray	−

2.3.6 文字列型

文字列型で表現するのは，DocumentBroker で使用できる文字コードセットから構成される任意の長さの文字列です。DocumentBroker で使用できる文字コードセットについては，「2.4.1 文字コードセットとの対応」を参照してください。

次に，文字列型のデータ型を使用する表現形式，演算子，関数および述語について説明します。また，DocumentBroker クラスライブラリで扱うデータ型との対応についても説明します。

(1) 表現形式

文字列型の値の表現形式を，次の表に示します。

表 2-17 文字列型の値の表現形式

値の種類	定義
リテラル	' 任意の文字列 ' 詳細は，「2.4.6 ＜リテラル＞」を参照してください。
プロパティ	String 型のプロパティ 格納値：定義長以下の文字列
変数として指定できる値	? パラメタ

(2) 演算子

文字列型の値を被演算子または引数とする，演算子を次の表に示します。

表 2-18 文字列型の値を被演算子または引数とする演算子

演算子の種類	演算子
文字列結合演算子	||

(3) 述語

評価対象が文字列型の値である述語を次の表に示します。

表 2-19 文字列型の値を評価対象とする述語

述語の種類	述語
比較述語	= < > <= >= <>
In 述語	IN
Between 述語	BETWEEN
Like 述語	LIKE XLIKE

(4) DocumentBroker クラスライブラリで扱うデータ型との対応

DocumentBroker クラスライブラリで扱うデータ型との対応を次の表に示します。

表 2-20 edmSQL の文字列型と DocumentBroker クラスライブラリで扱うデータ型との対応

edmSQL のデータ型	DocumentBroker クラスライブラリで扱うデータ型	値
文字列型	String	4,294,967,295 バイト以下の文字列

2.3.7 バイナリ型

バイナリ型で表現するのは，任意のバイト列を表現するデータです。主に，文書のコンテンツを表現します。関数の引数や，戻り値に指定する型としてだけ使用できます。プロパティの型としては定義できません。

バイナリ型が使用できる関数についての詳細は，「2.9 関数指定の構文規則」を参照してください。

(1) 表現形式

バイナリ型のデータの表現形式を，次の表に示します。

表 2-21 バイナリ型のデータの表現形式

値の種類	定義
変数	?パラメタ

バイナリ型のデータは，?パラメタ以外の形式では表現できません。

(2) DocumentBroker クラスライブラリで扱うデータ型との対応

DocumentBroker クラスライブラリで扱うデータ型との対応を，次の表に示します。

表 2-22 edmSQL のバイナリ型と DocumentBroker クラスライブラリで扱うデータ型との対応

edmSQL のデータ型	DocumentBroker クラスライブラリで扱うデータ型	値
バイナリ型	String	4,294,967,295 バイト以下の文字列※

注※ バイナリ型のデータは DocumentBroker クラスライブラリでは String 型の値として取得できます。

2.4 字句規則

ここでは，edmSQL で使用する字句に関する規則について説明します。

2.4.1 文字コードセットとの対応

ここでは，edmSQL で使用できる字句と，文字コードセットとの対応について説明します。

- 各国語を表記するための文字などは，文字コードセットに依存します。
  edmSQL で使用できる文字コードセットは，前提プログラムである OS と HiRDB に依存します。したがって，これらのプログラムで使用できない文字を使用した場合は，前提プログラムのエラーになります。
- edmSQL で使用する字句の規則は，検索の対象になる文書のコンテンツには適用されません。全文検索機能を使用する場合に検索対象になる文書の字句については，全文検索機能を提供する HiRDB Text Search Plug-in，または HiRDB XML Extension に依存します。
- 次の文字コードセットが使用できます。

  文書空間で使用する文字コード種別が Shift-JIS の場合
  　半角文字 JIS X 0201
  　全角文字 JIS X 0208

  文書空間で使用する文字コード種別が UTF-8 の場合
  　MS-Unicode または JIS X 0221 ※

  注※　UCS-4 用のインデクス定義の追加をしていないプロパティを，全文検索するとエラーとなります。UCS-4 用のインデクス定義の詳細は，マニュアル「DocumentBroker Version 5　システム導入・運用ガイド」を参照してください。

2.4.2 edmSQL で使用できる文字

ここでは，edmSQL で使用できる文字について説明します。

(1) 使用できる文字

使用できる文字を，次の表に示します。

表 2-23　edmSQL で使用できる文字

文字	値
英大文字	A ～ Z
英小文字	a ～ z
数字	0 ～ 9

文字	値
特殊文字（半角文字コード）	< 空白 > < ダブルクォート >（"） < パーセント >（%） < アンパサンド >（&） < シングルクォート >（'） < 左括弧 >（(） < 右括弧 >（)） < アスタリスク >（*） <+ 符号 >（+） < コンマ >（,） <- 符号 >（-） < ピリオド >（.） < 斜線 >（/） < コロン >（:） < セミコロン >（;） < 小なり演算子 >（<） < 等号演算子 >（=） < 大なり演算子 >（>） < 疑問符 >（?） < 左角括弧または trigraph >（ [または (??） < 右角括弧または trigraph >（] または ??)） < サーカムフレックス >（^） < 下線文字 >（_） < 垂直棒 >（|） < 左波括弧 >（{） < 右波括弧 >（}）

(2) 規約詳細

edmSQL で使用できる文字の詳細について説明します。

- < 空白 > は，JIS X 0201 では，0x20 に対応します。
- < 拡張文字 > とは，使用できる文字コードセットに対応して，使用可能な文字として拡張した文字です。使用できる文字コードセットについては，「2.4.1　文字コードセットとの対応」を参照してください。

2.4.3 < 区切り文字 >

< トークン > は，< 区切り文字 > によって分割されます。< 区切り文字 > は，< トークン > を抽出するための字句解析で削除される文字です。このため，構文解析を実行する時点では，< 区切り文字 > は出現しません。

(1) < 区切り文字 > として指定できる字句

< 区切り文字 > として指定できる字句は，<white space> です。

(2) 規約詳細

< 区切り文字 > の詳細について説明します。

edmSQL では，次の定義を <white　space> として定義します。

```
Horizontal Tab      [0x09]
Line Feed          [0x0A]
Vertical Tabulation  [0x0B]
```

```
Form Feed          [0x0C]
Carriage Return     [0x0D]
Space              [0x20]
```

[ ] 内は，対応する ASCII コードです。

2.4.4 < トークン >

< トークン > の定義について説明します。< トークン > は，構文の基本要素です。

< トークン > の定義には，次の要素の定義が含まれます。

- < キーワード >
- < リテラル > の一部
- < 識別子 >

それぞれの要素については，「2.4.5 < キーワード >」，「2.4.6 < リテラル >」および「2.4.7 < 識別子 >」を参照してください。

なお，< トークン > として指定できる文字列の長さなどについては，HiRDB の制限に従います。

(1) < トークン > として指定できる字句

< トークン > として指定できる字句の種類と値を次の表に示します。それぞれの字句の詳細については，参照先の説明を参照してください。

表 2-24 < トークン > として指定できる字句の種類と値

種　類	値	参照先
正規識別子	先頭が < 英字 > で，先頭以外が < 英字 >，< 数字 > および < 下線文字 > の値	2.4.7 < 識別子 >
キーワード	< 予約されたキーワード > および < 予約されていないキーワード > の値	2.4.5 < キーワード >
符号なし数値リテラル	符号なしの数字の値	2.4.6 < リテラル >
バイナリ長トークン	< 符号なし数値 > に <multiplier>（k，m，g）が付いた値	2.9 関数指定の構文規則の AS< データ型指定 >
区切られた識別子	< ダブルクォート > で囲まれた値 (< 区切り文字 >，< ダブルクォート以外の文字 > および < 二重ダブルクォート > を含むことができる)	2.4.7 < 識別子 >
文字列リテラル	< シングルクォート > で囲まれた値 (< シングルクォート以外の文字 > または < 二重シングルクォート >（''）が指定できる)	2.4.6 < リテラル >
特殊文字	edmSQL で使用できる特殊文字	2.4.2 edmSQL で使用できる文字
そのほかの文字	• < 不等号演算子 >（<>） • < 大なり等号演算子 >（>=） • < 小なり等号演算子 >（<=） • < 文字列連結演算子 >（||） • < 右矢印演算子 >（->） • < 二重コロン >（::） • < 左角括弧 >（[） • < 右角括弧 >（]） • < アンパサンド >（&）	−

（凡例）
－：該当しません。

(2) 規約詳細

< トークン > の詳細について説明します。

(a) < トークン > が区切られる単位についての詳細

● < トークン > で指定できる要素のうち，< 正規識別子 >，< キーワード >，< 符号なし数値リテラル > および < バイナリ長トークン > は，< トークン > を区切る要素として扱われます。これ以外の < トークン > を指定する場合は，前後に < 区切り文字 > または区切る要素になる < トークン > を指定して，前後との区切りを明確にしてください。区切る要素を指定しないで指定した場合，複数の < トークン > が一つの < トークン > として扱われたり，字句解析エラーになったりします。
< トークン > の表記例と edmSQL 検索実行時の値の対応を，次の表に示します。

表 2-25 トークンの表記例と edmSQL 検索実行時の値の対応

トークンの表記例	edmSQL 検索実行時の値
ABC □ 123	正規識別子 ABC と数値リテラル 123
ABC123	正規識別子 ABC123
ABC'123'	正規識別子 ABC と文字列リテラル '123'
123ABC	正規識別子の先頭に数字は指定できないため，字句解析エラーになります。

（凡例）
□：区切り文字

● < トークン > と < トークン > の間には任意の個数の区切り文字を指定できます。
例えば，□を区切り文字とした場合，「prop □□□ = □□□□□□□□□ 10」は，「prop=10」と同じ意味になります。また，「COUNT □□□□□ (□□□ * □□)」は，「COUNT(*)」と同じ意味になります。

● < トークン > の中に区切り文字を指定することはできません。ただし，< シングルクォート > に囲まれた文字列リテラルや < ダブルクォート > に囲まれて区切られた識別子の中には，区切り文字を指定できます。
また，「'< □ =」は，文字列定数「'< □ =」と識別されるため，エラーにはなりません。

(b) < ダブルクォート >< ダブルクォート以外の文字 >< 二重ダブルクォート > に関する詳細

● < ダブルクォート以外の文字 > は，使用できる文字コードセットの文字から，< ダブルクォート > を除いたものです。edmSQL で使用できる文字で構成されている必要はありません。
ただし，DocumentBroker のメタ定義で使用できる文字および HiRDB で使用できる文字以外は指定できません。

● < 二重ダブルクォート > は，< 区切られた識別子 > 内で < ダブルクォート > を指定する場合に使用します。

(c) < バイナリ長トークン > についての詳細

● < バイナリ長トークン > は，バイナリ型の定義長を指定するためのトークンです。関数の引数に指定する ? パラメタに対して，データ型を明示するための AS< データ型指定 > で使用します。

(d) 使用できる < 特殊文字 > についての詳細

● 次に示す < 特殊文字 > および演算子は，文字列リテラルの字句としてだけ使用できます。それ以外で使

用した場合，字句解析時にエラーになります。
% & : (?? ??) ^ | { } -> ::

2.4.5 < キーワード >

< キーワード > の定義について説明します。< キーワード > には < 予約されたキーワード >（予約語）と < 予約されていないキーワード > があります。< 予約されたキーワード > は正規識別子として使用できませんが，< 予約されていないキーワード > は正規識別子として使用できます。

< キーワード > では，大文字と小文字は区別されません。ただし，このマニュアルの例では，< 予約されたキーワード > を大文字で，< 予約されていないキーワード > を小文字で表記します。

edmSQL のキーワードは SQL のキーワードを基盤として，次のように定義されています。

- edmSQL では，SQL のキーワードおよび HiRDB のキーワードのうち，最小限のものをキーワードとして定義しています。すべての SQL のキーワードおよび HiRDB のキーワードが定義されているのではありません。ただし，ユーザが識別子を指定する場合には，edmSQL のキーワードのほか，HiRDB のキーワードも指定できません。識別子として HiRDB のキーワードを指定した場合，HiRDB のエラーになります。
- edmSQL では，全文検索関数などの関数名は < 予約されていないキーワード > として定義されています。そのほかのキーワードは < 予約されたキーワード > として定義されています。
- DocumentBroker の文書管理モデルで使用する用語として，幾つかのキーワードが < 予約されたキーワード > として定義されています。

(1) < キーワード > として指定できる字句

< キーワード > として指定できる字句を，次の表に示します。

表 2-26 < キーワード > として指定できる字句

種 類	値	特 徴
関数名	concept_with_score contains contains_with_score extracts objref oiid oiidstr score score_concept	予約されていないキーワードです。
multiplier	k m g	
句に使用するキーワード	SELECT FROM WHERE ORDER BY GROUP HAVING	予約されたキーワードです。

種　類	値	特　徴
述語に使用するキーワード	IS ALL SOME ANY IN LIKE XLIKE EXISTS BETWEEN AS ESCAPE	
論理演算に使用するキーワード	NOT AND OR	
リテラルに使用するキーワード	NULL TRUE FALSE UNKNOWN	
結合条件に使用するキーワード	JOIN INNER OUTER LEFT ON	
データ操作および重複排除に使用するキーワード	DISTINCT ASC DESC	
集合関数で使用するキーワード	COUNT	
数値関数で使用するキーワード	ABS	
プロパティのデータ型で使用するキーワード	BINARY BOOL BOOLEAN INT INTEGER STRING	

(2) 規約詳細

< キーワード > の詳細について説明します。

(a) < キーワード > の表記についての詳細

キーワードでは，大文字と小文字が区別されません。正規識別子の規約とは異なりますので，ご注意ください。edmSQL では，キーワードはすべて大文字として管理されます。

例えば，次の表記は，すべて < 予約されたキーワード >「SELECT」として識別されます。

```
Select  SeLect  select  selecT
```

また，次の表記はすべて < 予約されていないキーワード >「contains_with_score」として識別されます。

```
Contains_With_SCORE  contains_with_scorE
CONTAINS_WITH_SCORE
```

(b) < キーワード > と < 正規識別子 > の関係についての詳細

● < 予約されたキーワード > は正規識別子として使用できません。

● < 予約されていないキーワード > は正規識別子として使用できますが，HiRDB で予約語として定義さ

れている場合は，使用できません。使用すると，データベースエラーになります。

2.4.6 < リテラル >

< リテラル > の定義について説明します。< リテラル > とは，null 値以外の値です。

< リテラル > には，次の種類があります。

- 文字列リテラル
- 数値リテラル
- 論理リテラル

(1) < リテラル > として指定できる字句

< リテラル > として指定できる字句を，次の表に示します。

表 2-27 < リテラル > として指定できる字句

種　類	値
文字列リテラル	< シングルクォート > で囲まれた文字列 （文字列には，< シングルクォート以外の文字 >※または < 二重シングルクォート >（''）が指定できる）
数値リテラル	符号（+ または -）付きの数字 （符号は省略できる）
論理リテラル	TRUE FALSE UNKNOWN

注※　< シングルクォート以外の文字 > は，使用できる文字コードセットの文字から，< シングルクォート > を除いたものです。edmSQL で使用できる文字で構成される必要はありません。ただし，文字列リテラルは文字列型の値であるため，文字列型を格納する HiRDB のデータ型（MVARCHAR 型）で制限される文字は使用できません。

(2) 規約詳細

< リテラル > の詳細について説明します。

(a) 文字列リテラルの詳細

文字列リテラルは，任意の文字列定数を表現する場合に使用します。文字列リテラルは，文字列型の値に対応します。したがって，文字列リテラルで使用できる文字列の長さは，HiRDB での制限に従います。HiRDB の制限以上の長さの文字列を文字列リテラルとして指定したい場合は，? パラメタを使用してください。

文字列リテラルの表記例と edmSQL 検索実行時の値の対応を，次の表に示します。なお，文字列リテラルを表す文字列型の値は，HiRDB では MVARCHAR 型のデータとして格納されます。

表 2-28 文字列リテラルの表記例と edmSQL 検索実行時の値の対応

文字列リテラルの表記例	値の扱われ方	edmSQL 検索実行時の値
'aaa'	文字列データ	aaa
'124'	文字列データ	124
'abc あいうえお '	混在文字列データ	abc あいうえお

文字列リテラルの表記例	値の扱われ方	edmSQL 検索実行時の値
'ab"c あ " いうえお '	シングルクォートのエスケープ	ab'c あ ' いうえお

(b) 数値リテラル（符号付き数値リテラル）の詳細

数値リテラルは，任意の数値定数を表現する場合に使用します。なお，DocumentBroker で使用できる数値リテラルは，4 バイトの整数値だけです。

なお，4 バイトの整数値の有効値は，-2,147,483,648 ～ 2,147,483,647 までの値です。この範囲外の値を指定した場合，HiRDB の制限に従って，目的の検索が実行できなかったり，データベースのエラーになったりします。

数値リテラルの表記例と edmSQL 検索実行時の値の対応を，次の表に示します。

表 2-29 数値リテラルの表記例と edmSQL 検索実行時の値の対応

数値リテラルの表記例	値の扱われ方	edmSQL 検索実行時の値
123	数値データ	123
0000000123	数値データ	123
+1234567890	数値データ（正の整数値）	1234567890
-1234567890	数値データ（負の整数値）	-1234567890
987654321000000000	リテラルとしては正しい表記ですが，4 バイト整数値の有効値ではないため，正しく扱われません。	－

（凡例）
－：正しい値になりません。

(c) 論理リテラルの詳細

論理リテラルは，論理値である「真」,「偽」および「不定」の三つの値を，それぞれ「TRUE」,「FALSE」および「UNKNOWN」で表現します。

論理型の値を表す Boolean 型のプロパティの値は，TRUE は「1」，FALSE は「0」，UNKNOWN は「2」として，数値で格納されます。ただし，この値を edmSQL で数値として扱うことはできません。edmSQL で指定する場合は，必ず TRUE，FALSE または UNKNOWN として表現してください。

また，論理リテラルを整数の演算に使用する値として使用することもできません。ただし，DocumentBroker クラスライブラリを使用して，Java の基本データ型の値に変換した場合は，論理リテラルの値を数値として扱うこともできます。この場合の数値のデータベースでの扱われ方は，HiRDB でのデータモデルの対応付けに依存します。

論理リテラルの表記ごとの，論理値を表す TRUE の扱われ方について，例に示します。

- 次の TRUE はデータベースによって，論理述語として処理されます。

```
contains(edmProp_StIndex,'文章[概要{"コンピュータ"}]') IS TRUE
```

- 次の TRUE は DocumentBroker クラスライブラリを通して数値として処理されます。ただし，myProp_OK は Boolean 型のプロパティです。

```
myProp_OK = TRUE
```

2.4.7 < 識別子 >

< 識別子 > の定義について説明します。< 識別子 > には，< 正規識別子 > と < 区切られた識別子 > があります。

なお，識別子には，日本語の文字コードセットは使用できません。

(1) < 識別子 > として指定できる字句

< 識別子 > として指定できる字句を，次の表に示します。

表 2-30 < 識別子 > として指定できる字句

種　類	値
正規識別子	先頭は < 英字 > で，先頭以外が < 英字 >，< 数字 > および < 下線文字 > の値 例 usrClass_ABC
区切られた識別子	< ダブルクォート > で囲まれた値 (< 区切り文字 >，< ダブルクォート以外の文字 >※または < 二重ダブルクォート >（""）を含むことができる） 例 "user class" "user""class"

注※ < ダブルクォート以外の文字 > として指定できるのは，英字，数字および「"」を除いた特殊文字です。

(2) 規約詳細

< 識別子 > の詳細について説明します。

(a) < 正規識別子 > に関する詳細

- edmSQL では，< 正規識別子 > に使用される英字の < 英大文字 > と < 英小文字 > が区別して扱われます。このため，SQL を使用する場合のように，< 英大文字 > と < 英小文字 > を区別するために，< 区切られた識別子 > を使用する必要はありません。
- < 正規識別子 > に < キーワード > は使用できません。< キーワード > と同じ識別子を使用する場合は，< 区切られた識別子 > として指定してください。

(b) < 区切られた識別子 > に関する詳細

- 値が空の区切られた識別子「""」は，無効な識別子として扱われます。これを指定した場合は，字句解析エラーになります。

2.4.8 < 名前 >

< 名前 > の定義について説明します。< 名前 > では，クラス名，プロパティ名，相関名および関数名を表現します。

< 名前 > の基本要素は，< 識別子 > です。

(1) < 名前 > として指定できる字句

< 名前 > として指定できる字句を，次の表に示します。

表 2-31 ＜名前＞として指定できる字句

種　類	値	参照先
クラス名	• ＜識別子＞	• 2.4.7 ＜識別子＞
プロパティ名	• ＜識別子＞ • ＜特殊なプロパティ＞	• 2.4.7 ＜識別子＞ • 2.4.9 ＜特殊なプロパティ＞
関数名	（変換関数） • oiidstr • objref • oiid （全文検索関数） • contains • contains_with_score • score • extracts （概念検索関数） • concept_with_score • score_concept	• 2.9 関数指定の構文規則
相関名	＜識別子＞	• 2.4.7 ＜識別子＞

(2) 規約詳細

＜名前＞の詳細について説明します。

(a) クラス名の詳細

＜クラス名＞は，文書空間内で DocumentBroker クラスを識別するための名称です。使用できる文字列の長さは，HiRDB のテーブル名に使用できる文字列の長さに従います。

＜識別子＞の詳細は，「2.4.7 ＜識別子＞」を参照してください。

(b) プロパティ名の詳細

＜プロパティ名＞は，文書空間内でプロパティを識別するための名称です。使用できる文字列の長さは，HiRDB のカラム名に使用できる文字列の長さに従います。

なお，そのプロパティがどの DocumentBroker クラスのプロパティであるかが明確でない場合は，＜プロパティ名＞を＜クラス名＞または＜相関名＞で修飾する必要があります。

＜識別子＞および＜特殊なプロパティ＞の詳細は，「2.4.7 ＜識別子＞」および「2.4.9 ＜特殊なプロパティ＞」を参照してください。

注意事項

DocumentBroker では，プロパティの定義は文書空間ごとに定義されています。したがって，異なるクラスに登録されたプロパティであっても，同じ文書空間内のクラスであれば，プロパティの性質は同じことが保証されています。ほかのオブジェクト指向言語で，クラスのスコープ単位にプロパティが定義されているものとは異なりますので，ご注意ください。

(c) 関数名の詳細

＜関数名＞は，edmSQL で予約されていないキーワードとして登録されている関数を識別するための名称です。

関数についての詳細は，「2.9 関数指定の構文規則」を参照してください。

(d) 相関名の詳細

< 相関名 > は，同じクラスを結合する場合に，結合するクラスを区別するための名称です。一つの <FROM 句 > の中に，同じクラスを複数回指定する場合には，必ず < 相関名 > を指定して，それぞれのクラスを一意に識別できるようにしてください。

そのほか，< 相関名 > は，< クラス名 > の別名としても使用できます。

< 相関名 > として使用できる文字列の長さなどの制限については，HiRDB の相関名に関する制限に従います。

2.4.9 < 特殊なプロパティ >

< 特殊なプロパティ > の定義について説明します。< 特殊なプロパティ > とは，全文検索で使用する全文検索インデクス用プロパティです。

(1) < 特殊なプロパティ > として指定できる字句

< 特殊なプロパティ > として指定できる字句を，次の表に示します。

表 2-32 < 特殊なプロパティ > として指定できる字句

値	説 明
edmProp_TextIndex	edmProp_TextIndex プロパティを表す < 正規識別子 >
edmProp_ConceptTextIndex	edmProp_ConceptTextIndex プロパティを表す < 正規識別子 >

これらのプロパティの特長については，マニュアル「DocumentBroker Version 5　概説」の全文検索インデクスの作成の説明を参照してください。

(2) 規約詳細

< 特殊なプロパティ > の詳細について説明します。

(a) < 特殊なプロパティ > と < プロパティ名 > についての詳細

< 特殊なプロパティ > は，edmSQL で，DocumentBroker の検索モデルを拡張するために導入されたプロパティです。このプロパティは，edmSQL によって予約されたプロパティ名であるため，< プロパティ名 > としては使用できません。なお，DocumentBroker としても，ユーザ定義プロパティに「dmaProp_」，「edmProp_」および「dbrProp_」で始まるプロパティ名称は使用できません。

DocumentBroker で拡張したプロパティのうち，< 特殊なプロパティ > として定義されていないプロパティについては，ほかのプロパティと同様の処理ができます。

(b) < 特殊なプロパティ > を指定できる関数についての詳細

< 特殊なプロパティ > を指定できる関数を示します。

edmProp_TextIndex を第 1 引数に指定できる関数

- contains 関数
- contains_with_score 関数
- score 関数
- extracts 関数

edmProp_ConceptTextIndex を第 1 引数に指定できる関数

- contains 関数
- contains_with_score 関数
- score 関数
- extracts 関数
- concept_with_score 関数
- score_concept 関数

< 特殊なプロパティ > を引数に指定できる関数の詳細については，「2.9　関数指定の構文規則」を参照してください。

2.4.10　<? パラメタ >

<? パラメタ > は，検索条件の定数を構文解析時には固定しないで，検索実行時に定数を指定するために使用するパラメタです。edmSQL 文には定数を指定する代わりに「?」を指定しておきます。これによって，問い合わせの実行直前に DbjQParam インターフェースのサブインターフェースをデータ型とした値を「?」に設定できます。<? パラメタ > を使用することで，構文解析済みの edmSQL 文を定数値だけを変えて繰り返し利用できるので，処理性能が上がります。

また，概念検索を実行する場合には，検索条件は <? パラメタ > を使用しないと指定できません。

(1) <? パラメタ > として指定できる字句

<? パラメタ > として指定できるのは，「?」だけです。

(2) 規約詳細

<? パラメタ > の詳細について説明します。

- <? パラメタ > で指定される値のデータ型は不定です。このため，edmSQL 文の構文解析時にはデータ型はチェックされません。ただし，AS< データ型指定 > で明示的にデータ型を指定した <? パラメタ > については，データ型チェックの対象になります。
- < ルーチンの起動 > の引数として <? パラメタ > を使用する場合は，AS< データ型指定 > で明示的にデータ型を指定してください。

2.4.11　<OIID>

<OIID> の定義について説明します。

検索対象になるオブジェクトは，OIID によって識別されます。OIID の値は，データベース上では，各オブジェクトの dmaProp_OIID プロパティに，String 型の 16 バイトの値として格納されています。

一方，DocumentBroker クラスライブラリでは，OIID を「dma://」で始まる OIID 文字列（DMA URL）で表記します。これは，実際に dmaProp_OIID プロパティに格納されている値とは異なります。このため，DocumentBroker クラスライブラリで使用する OIID 文字列を直接 edmSQL 文に指定しても，OIID を対象にした検索はできません。このため，OIID 文字列を指定する場合には，OIID 変換インターフェースを使用して，dmaProp_OIID プロパティの値に変換する必要があります。

なお，検索結果として dmaProp_OIID プロパティの値を取得する場合は，DocumentBroker クラスライブラリで使用する OIID 文字列に変換されたものが取得できるため，明示的に変換する必要はありません。

(1) OIID 文字列を dmaProp_OIID プロパティの値に変換する指定（OIID 変換インターフェース）

検索条件に OIID 文字列を指定する場合に，指定した OIID 文字列を dmaProp_OIID プロパティの値に変換するためには，次のどちらかの方法を使用します。

● oiid 関数で変換する

edmSQL 文に OIID 文字列を直接指定する場合の変換方法です。
oiid 関数の詳細については，「2.9.3(4) <objref 関数 > の詳細」を参照してください。

● <? パラメタ > を使用して変換する

edmSQL 文に <? パラメタ > を指定しておく場合の変換方法です。<? パラメタ > に OIID 文字列を設定する方法については，「1.7　パラメタの操作」を参照してください。

(2) OIID 変換インターフェースの使用例

ここでは，OIID 変換インターフェースの使用例について説明します。

OIID 変換インターフェースを使用しないで OIID を検索しようとしている例

dmaProp_OIID プロパティの値と OIID 文字列の形式は異なるため，検索結果が「真」になる（検索結果が取得できる）ことはありません。

```
SELECT  myProp_Foo
FROM    myClass
WHERE  dmaProp_OIID = 'dma:///xxx/xxx/xxx…x'
```

oiid 関数を使用して OIID を検索する例

検索結果が「真」になる（検索結果が取得できる）可能性があります。

```
SELECT  myProp_Foo
FROM    myClass
WHERE  dmaProp_OIID = oiid('dma:///xxx/xxx/xxx…x')
```

<? パラメタ > を使用して OIID を検索する例

検索結果が「真」になる（検索結果が取得できる）可能性があります。

```
SELECT  myProp_Foo
FROM    myClass
WHERE  dmaProp_OIID = ?
```

2.5 検索の実行単位の構文規則

edmSQL 検索は，一つの <edmSQL 文 > から構成される，<edmSQL プログラム > ごとに実行されます。

形式

!! <edmSQL**プログラム>の形式**

```
<edmSQLプログラム> ::= <edmSQL文>
                     |<edmSQL文> <セミコロン>
```

!! <edmSQL**文>の形式**

```
<edmSQL文>::= <問い合わせ文>
```

< 問い合わせ文 > については，「2.6　問い合わせ式の構文規則」を参照してください。

2.5.1 <edmSQL プログラム >

ここでは，<edmSQL プログラム > の形式と規則について説明します。

(1) <edmSQL プログラム > の形式

```
<edmSQLプログラム> ::= <edmSQL文>
                     |<edmSQL文> <セミコロン>
```

(2) <edmSQL プログラム > についての規則

- <edmSQL プログラム > は，一つの <edmSQL 文 > から構成されます。
- <edmSQL プログラム > が <edmSQL 文 > で構成されていない場合は，構文エラーになります。また，< セミコロン > だけを指定した場合も，構文エラーになります。

2.5.2 <edmSQL 文 >

ここでは，<edmSQL 文 > の形式と規則について説明します。

(1) <edmSQL 文 > の形式

```
<edmSQL文>::= <問い合わせ文>
```

(2) <edmSQL 文 > についての規則

- <edmSQL 文 > に指定できるのは，< 問い合わせ文 >（SELECT 文）だけです。
- <edmSQL 文 > として指定できる文字列の長さの最大値は，HiRDB の制限に従います。

2.6 問い合わせ式の構文規則

問い合わせは，< 問い合わせ文 >（SELECT 文）によって表現します。< 問い合わせ文 > には，SELECT 句，FROM 句，WHERE 句および ORDER BY 句を指定します。最も基本的な < 問い合わせ文 > は，SELECT 句と FROM 句から構成されます。

SELECT 句には，検索結果として取得する項目を指定します。

FROM 句および WHERE 句は，検索対象式として指定します。FROM 句には，検索対象になる DocumentBroker クラスを指定します。WHERE 句には，検索条件を指定します。検索条件は，述語で表現します。述語については，「2.8　述語の構文規則」を参照してください。なお，WHERE 句には，副問い合わせも指定できます。

ORDER BY 句には，検索結果として取得した集合をソートするかどうかを指定します。ORDER BY 句については，「2.10　データ操作の構文規則」を参照してください。

形式

!! <問い合わせ文>の形式

```
<問い合わせ文> ::= <問い合わせ式> 〔 <ORDER BY句> 〕
<問い合わせ式> ::= <問い合わせ指定>
                 | <左括弧> <問い合わせ式> <右括弧>
```

!! <問い合わせ指定>の形式

```
<問い合わせ指定> ::= SELECT 〔 <集合指定子> 〕
                     <選択項目の並び> <検索対象式>
<選択項目の並び> ::= <アスタリスク>
                   | <選択項目> 〔 { <コンマ> <選択項目> }... 〕
<選択項目> ::= <一次子>
<集合指定子> ::= DISTINCT | ALL
```

!! <検索対象式>の形式

```
<検索対象式> ::= <FROM句>
                 〔 <WHERE句> 〕
```

!! <FROM句>の形式

```
<FROM句> ::= FROM <検索対象>
<検索対象> ::=   <検索対象参照リスト>
               <結合された検索対象>
<検索対象参照リスト> ::=<検索対象参照>
                        〔 { <コンマ> <検索対象参照> }... 〕
<検索対象参照> ::= <クラス名> 〔<相関名>〕

<結合された検索対象> ::= <条件指定結合>
<条件指定結合> ::= <検索対象一次子> 〔 <結合種別> 〕
                 JOIN <検索対象参照> <結合指定>

<検索対象一次子> ::= <検索対象参照>
                     | <結合された検索対象>
                     | <左括弧><結合された検索対象><右括弧>

<結合指定> ::= <結合条件>
<結合条件> ::= ON <検索条件>

<結合種別> ::= INNER
             | <外部結合種別> 〔 OUTER 〕
<外部結合種別> ::= LEFT
```

!! <WHERE句>の形式

```
<WHERE句> ::= WHERE <検索条件>
```

!! <GROUP BY句>の形式

```
<GROUP BY句> ::= GROUP BY <グループ化項目の並び>
```

```
<グループ化項目の並び> ::= <グループ化項目>
                          [ { <コンマ> <グループ化項目> }... ]
<グループ化項目> ::= <一次子>
```

!! **<HAVING句>の形式**

```
<HAVING句> ::= HAVING <検索条件>
```

!! **<副問い合わせ>の形式**

```
<副問い合わせ> ::=<左括弧> <問い合わせ式> <右括弧>
```

検索条件については,「2.8　述語の構文規則」を参照してください。

2.6.1　< 問い合わせ文 >

ここでは,< 問い合わせ文 > の形式と規則について説明します。

(1) < 問い合わせ文 > の形式

```
<問い合わせ文> ::= <問い合わせ式> 〔 <ORDER BY句> 〕
<問い合わせ式> ::= <問い合わせ指定>
                 | <左括弧> <問い合わせ式> <右括弧>
```

(2) < 問い合わせ式 > についての規則

< 問い合わせ式 > は,少なくとも SELECT 句と FROM 句から構成される問い合わせの表現です。< 問い合わせ式 > は,検索結果集合として評価されます。

(3) <ORDER BY 句 > についての規則

<ORDER BY 句 > には,< 問い合わせ式 > の評価である検索結果集合に対して,検索結果要素を並べ替えるためのソート方法を指定します。

2.6.2　< 問い合わせ指定 >

ここでは,< 問い合わせ指定 > の形式と規則について説明します。

(1) < 問い合わせ指定 > の形式

```
<問い合わせ指定> ::= SELECT 〔 <集合指定子> 〕
                     <選択項目の並び> <検索対象式>
<選択項目の並び> ::= <アスタリスク>
                   | <選択項目> 〔 { <コンマ> <選択項目> }... 〕
<選択項目> ::= <一次子>
```

(2) < 選択項目 > および < 選択項目の並び > についての規則

- < 選択項目の並び > には,検索を実行した場合の結果として出力する項目を指定します。一つまたは複数の < 選択項目 >(< 選択項目の並び >)と < 検索対象式 > を指定することで,検索結果集合が求められます。
- < 選択項目 > に指定できるプロパティは,< 検索対象式 > に指定した DocumentBroker クラスに定義されているプロパティだけです。
- < 選択項目の並び > に < アスタリスク > を指定できるのは,< 検索対象式 > の中の <Exists 述語 > で指定された < 副問い合わせ > の中だけです。
- < 検索対象式 > 中の < 比較述語 > や <In 述語 > で副問い合わせを指定する場合は,それぞれの述語によって求める検索結果と整合性のあるデータ型の項目を < 選択項目 > として指定してください。

● < 選択項目 > の < 一次子 > に指定できる要素は，< プロパティ指定 >，< 集合関数 > または < ルーチンの起動 > だけです。
< ルーチンの起動 > に指定できる関数については，「2.9　関数指定の構文規則」を参照してください。

● < 選択項目 > に，オブジェクトリファレンスを直接指定することはできません。オブジェクトリファレンスを指定する場合は，oiidstr 関数によって OIID 文字列に変換して指定してください。

(3) < 集合指定子 > についての規則

● < 集合指定子 > は，検索結果集合内の結果として同じ要素が返却された場合に，その重複した要素を排除（**重複排除**）する場合に指定します。

● DISTINCT を指定すると，検索結果集合に対して重複排除が実行され，重複を排除した検索結果集合が返却されます。

● ALL を指定すると，重複している要素も含めて，すべての検索結果集合が返却されます。

● 省略した場合は，ALL が仮定されます。

● < 選択項目 > に VariableArray 型のプロパティを指定した場合は，< 集合指定子 > に DISTINCT は指定できません。DISTINCT を指定できる選択項目のデータ型については，HiRDB の制限に従います。

● アクセス制御機能を使用している場合，< 集合指定子 > に DISTINCT は指定できません。

2.6.3　< 検索対象式 >

ここでは，< 検索対象式 > の形式と規則について説明します。

(1) < 検索対象式 > の形式

```
<検索対象式> ::= <FROM句>
                 〔 <WHERE句> 〕
                 〔 <GROUP BY句> 〕
                 〔 <HAVING句> 〕
```

<WHERE 句 >，<GROUP BY 句 > および <HAVING 句 > は，省略できます。

2.6.4　<FROM 句 >

ここでは，<FROM 句 > の形式と規則について説明します。

<FROM 句 > には，検索対象になる一つまたは複数のクラスを指定します。複数のクラスを検索対象に指定する場合には，結合条件も指定できます。

edmSQL では，結合方法として，次の方法が使用できます。

- 複数のクラスを並べて指定した暗黙的な結合
- 結合種別に内部結合（INNER JOIN）を指定した結合
- 結合種別に左外部結合（LEFT OUTER JOIN）を指定した結合

(1) <FROM 句 > の形式

```
<FROM句> ::= FROM <検索対象>
<検索対象> ::=  <検索対象参照リスト>
              |<結合された検索対象>
<検索対象参照リスト> ::= <検索対象参照>
                         〔{ <コンマ> <検索対象参照> }...〕
<検索対象参照> ::= <クラス名> 〔<相関名>〕
```

```
<結合された検索対象> ::= <条件指定結合>
<条件指定結合> ::= <検索対象一次子> 〔 <結合種別> 〕
                   JOIN <検索対象参照> <結合指定>

<検索対象一次子> ::= <検索対象参照>
                       |<結合された検索対象>
                       |<左括弧><結合された検索対象><右括弧>

<結合指定> ::= <結合条件>
<結合条件> ::= ON <検索条件>

<結合種別> ::= INNER
             |<外部結合種別> 〔 OUTER 〕
<外部結合種別> ::= LEFT
```

(2) < 検索対象 > についての規則

< 検索対象 > には < 検索対象参照リスト > を指定します。< 検索対象参照リスト > として < コンマ > で区切った < 検索対象参照 > を指定すると，それぞれの < 検索対象参照 > は暗黙的に結合されて，検索対象として扱われます。

検索対象を暗黙的に結合した場合は，結合条件を <WHERE 句 > に指定することで，内部結合を指定した場合と同じ結合が表現できます。<WHERE 句 > に結合条件を指定しない場合は，指定した DocumentBroker クラスの直積空間が検索の対象になります。

(3) < 結合された検索対象 > についての規則

< 結合された検索対象 > は，複数の検索対象（DocumentBroker クラス）を明示的に結合種別と結合条件を指定して結合する場合に指定します。

(4) < 条件指定結合 > についての規則

- < 条件指定結合 > には，内部結合（INNER JOIN）または左外部結合（LEFT OUTER JOIN）が指定できます。
- < 条件指定結合 > では，< 結合種別 > の左側に指定する < 検索対象一次子 > に，さらに < 結合された検索対象 > を指定できます。つまり，結合のネストができます。
- < 条件指定結合 > を指定する場合は，必ず < 結合指定 >（ON 条件）を指定してください。

(5) < 結合条件 > についての規則

< 結合条件 > に指定するプロパティには，その < 結合条件 > を含む < 検索対象一次子 > および < 検索対象 > で指定している DocumentBroker クラスのプロパティ，またはその < 問い合わせ指定 > の外側の検索対象の DocumentBroker クラスのプロパティを指定してください。

(6) < 結合種別 > についての規則

- 内部結合を指定する場合は，< 結合種別 > に「INNER」を指定します。内部結合を指定すると，指定した DocumentBroker クラスの直積空間のうち，< 結合条件 > を満たす結果の集合が検索対象になります。< 結合条件 > に結合キーとして指定したプロパティが null 値の場合，< 結合条件 > を満たす集合が存在しないので，指定した DocumentBroker クラスの直積空間は検索対象にはなりません。このため，結合を指定したどちらの DocumentBroker クラスのオブジェクトも検索対象になりません。
- 左外部結合を指定する場合は，< 結合種別 > に「LEFT」または「LEFT OUTER」を指定します。外部結合を指定した場合，< 結合条件 > に結合キーとして指定したプロパティが null 値であっても，< 結合種別 > の左側に指定した < 検索対象一次子 > の DocumentBroker クラスのオブジェクトは，すべて検

索対象になります。この場合，< 結合種別 > の右側に指定した検索対象の DocumentBroker クラスのオブジェクトは存在しないことになるため，該当する項目には null 値が設定されます。

● < 結合種別 > を省略すると，「INNER」が仮定されます。

(7) <FROM 句 > 全体についての規則

● 次の項目については，HiRDB の制限に従います。
- 結合できるクラスの数や，結合で指定できるネストの深さ
- < 結合条件 > に指定できるプロパティ
- 実行できる結合および結合の組み合わせ

● 一つの FROM 句内で，相関名は重複できません。また，検索対象クラスのクラス名と相関名も，重複できません。

● FROM 句で指定する < クラス名 > および < 相関名 > の有効範囲は，次のとおりです。副問い合わせでこれらの < クラス名 > および < 相関名 > を指定する場合には，ご注意ください。

< クラス名 > と < 相関名 > の有効範囲

FROM 句で指定した < 相関名 > および < 相関名 > なしで指定した < クラス名 > の有効範囲は，その <FROM 句 > を含む < 問い合わせ指定 > 内全体です。したがって，その < 問い合わせ指定 > 内にある < 副問い合わせ > でも有効になります。

また，< 相関名 > を指定した < クラス名 > の有効範囲は，その FROM 句を含む < 問い合わせ指定 > だけです。したがって，< 問い合わせ指定 > 内にある < 副問い合わせ > では，その名前は有効になりません。

2.6.5 <WHERE 句 >

ここでは，<WHERE 句 > の形式と規則について説明します。

<WHERE 句 > には，検索条件を指定します。

検索対象のオブジェクトに対して，<WHERE 句 > で指定した < 検索条件 > が真である場合に，そのオブジェクトが検索結果として取得できます。取得した検索結果は，< 選択項目 > に指定した項目の値として取得できます。

(1) <WHERE 句 > の形式

```
<WHERE句> ::= WHERE <検索条件>
```

< 検索条件 > については，「2.8　述語の構文規則」を参照してください。

2.6.6 < 副問い合わせ >

ここでは，< 副問い合わせ > の形式と規則について説明します。

(1) < 副問い合わせ > の形式

```
<副問い合わせ> ::=<左括弧> <問い合わせ式> <右括弧>
```

(2) < 副問い合わせ > についての規則

● < 副問い合わせ > は，<In 述語 >，< 比較述語 > または <Exists 述語 > の対象として指定します。それぞれの述語については，「2.8　述語の構文規則」を参照してください。

● < 副問い合わせ > の < 選択項目 > に VariableArray 型のプロパティは指定できません。

● アクセス制御機能を使用している場合も，< 副問い合わせ > ではアクセス制御は実行されません。< 副問い合わせ > の検索結果として取得できる検索結果には，アクセス権を持たないものが含まれる可能性があります。

2.6.7 GROUP BY 句

ここでは，<GROUP BY 句 > の形式と規則について説明します。

<GROUP BY 句 > には，複数のオブジェクトを一つのグループとする条件を指定します。

(1) <GROUP BY 句 > の形式

```
<GROUP BY句> ::= GROUP BY <グループ化項目の並び>
<グループ化項目の並び> ::= <グループ化項目>
                    [ { <コンマ> <グループ化項目> }... ]
<グループ化項目> ::= <一次子>
```

(2) <GROUP BY 句 > 全体についての規則

● GROUP BY 句で指定したプロパティの検索結果がすべて同じオブジェクトを一つのグループとして，検索結果をグループごとに取得できます。

● GROUP BY 句にはプロパティだけが指定できます。

● < グループ化項目の並び > に指定できるプロパティは，GROUP BY 句を指定した問い合わせの FROM 句で指定したクラスのプロパティです。

● < グループ化項目の並び > に指定できる < 一次子 > の最大数は，前提となる DBMS の仕様に制限されます。

● SELECT 句に指定できる選択項目は GROUP BY 句で指定したプロパティおよび集合関数です。

● オブジェクト型，VariableArray 型，バイナリ型のプロパティ，特殊なプロパティおよび関数は GROUP BY 句に指定できません。

● 同一のプロパティを重複して GROUP BY 句に指定することはできません。

2.6.8 HAVING 句

ここでは，<HAVING 句 > の形式と規則について説明します。

<HAVING 句 > には，GROUP BY 句，WHERE 句，FROM 句の結果，得られる各グループを選択する条件を指定します。

(1) <HAVING 句 > の形式

```
<HAVING句> ::= HAVING <検索条件>
```

(2) <HAVING 句 > 全体についての規則

● GROUP BY 句，WHERE 句，または FROM 句の結果，得られる各グループを選択する条件を指定します。

● NULL 述語および LIKE 述語は HAVING 句に直接指定できません。

● HAVING 句に指定できるプロパティは，GROUP BY 句で指定したプロパティ，集合関数の引数として指定したプロパティ，または外側の問い合わせの FROM 句に指定したクラスのプロパティです。

2.7 スカラー式表現の構文規則

値は，スカラー式によって表現されます。スカラー式で表現できる値は，プロパティの指定，ルーチン起動，数値関数および集合関数によって得られる値です。

形式

!! <プロパティ指定>の形式

```
<プロパティ指定> ::= 〔<プロパティ修飾子> <ピリオド>〕 <プロパティ名>
<プロパティ修飾子> ::= <相関名>      !! <相関名>による修飾
                     |<クラス指定> !! <クラス指定>による修飾
```

!! <要素参照>の形式

```
<要素参照> ::= <一次子>
                 <左角括弧> ANY <右角括弧>
```

!! <フィールド参照>の形式

```
<フィールド参照> ::= <要素参照> <ピリオド> <フィールド名>
<フィールド名> ::= <プロパティ名>
```

!! <ルーチンの起動>の形式

```
<ルーチンの起動> ::= <ルーチン名> <引数リスト>
<引数リスト> ::= <左括弧> 〔 <引数>
                   〔 { <コンマ> <引数> }... 〕 〕 <右括弧>
<引数> ::=  <値式>
          |<値式> AS <データ型指定>
<ルーチン名> ::= <関数指定>

<データ型指定> ::= INT
                 |INTEGER
                 |BOOL
                 |BOOLEAN
                 |STRING <左括弧> <符号なし数値> <右括弧>
                 |BINARY <左括弧> <バイナリ長> <右括弧>

<バイナリ長> ::=  <符号なし数値>
                |<バイナリ長トークン>
```

!! <数値関数>の形式

```
<数値関数> ::= <絶対値関数>
<絶対値関数> ::= ABS <左括弧> <値式> <右括弧>
```

!! <集合関数>の形式

```
<集合関数> ::= COUNT <左括弧> <アスタリスク> <右括弧>
             |<一般集合関数>
<一般集合関数> ::= <集合関数種別>
                   <左括弧> 〔 <集合指定子> 〕 <値式> <右括弧>
<集合関数種別> ::= COUNT
<集合指定子> ::= DISTINCT|ALL
```

!! <値式>の形式

```
<値式> ::= <一次子>
          |<符号> <一次子>
          |<値式> <+符号> <値式>
          |<値式> <-符号> <値式>
          |<値式> <アスタリスク> <値式>
          |<値式> <斜線> <値式>
          |<値式> <文字列連結演算子> <値式>
<一次子> ::= <左括弧> <値式> <右括弧>
           |<プロパティ指定>
           |<符号なし値指定>
           |<集合関数>
           |<数値関数>
           |<ルーチンの起動>
           |<フィールド参照>
```

```
<符号なし値指定> ::= <符号なし数値リテラル>
                   | <文字列リテラル>
                   | <疑問符>
                   | <論理リテラル>
<値指定> ::= 〔<符号>〕<符号なし数値リテラル>
           | <文字列リテラル>
           | <疑問符>
           | <論理リテラル>
```

2.7.1 < プロパティ指定 >

ここでは，< プロパティ指定 > の形式と規則について説明します。

(1) < プロパティ指定 > の形式

```
<プロパティ指定> ::= 〔<プロパティ修飾子> <ピリオド>〕 <プロパティ名>
<プロパティ修飾子> ::= <相関名>      !! <相関名>による修飾
                     | <クラス指定> !! <クラス指定>による修飾
```

(2) < プロパティ指定 > についての規則

同じ名称のプロパティを持つクラスを結合する場合など，< プロパティ名 > だけではプロパティを一意に識別できなくなる場合は，< プロパティ修飾子 > を指定して < プロパティ指定 > が一意になるようにしてください。

(3) < プロパティ修飾子 > についての規則

< 相関名 > および < クラス指定 > として指定できるのは，<FROM 句 > に指定したものだけです。

< プロパティ指定 > の指定例を，次の表に示します。

表 2-33 < プロパティ指定 > の指定例

指定例	説　明
myProp_Foo	< 識別子 > によって < プロパティ名 > だけを指定しています。
myClass_XX.myProp_Foo	< プロパティ名 > を < クラス名 > によって修飾して指定しています。
P0.myProp_Foo	< プロパティ名 > を < 相関名 > によって修飾して指定しています。

2.7.2 < 要素参照 >

ここでは，< 要素参照 > の形式と規則について説明します。

< 要素参照 > には，VariableArray 型のプロパティの要素を参照するための表現を指定します。

(1) < 要素参照 > の形式

```
<要素参照> ::= <一次子>
                 <左角括弧> ANY <右角括弧>
```

(2) < 要素参照 > についての規則

- < 要素参照 > は，<WHERE 句 > だけで指定できます。
- < 要素参照 > は，必ず < フィールド参照 > とともに指定してください。
- 要素を指定する値には，「ANY」以外の要素は指定できません。
- < 一次子 > には，< プロパティ指定 > だけが指定できます。

2.7.3 < フィールド参照 >

ここでは，< フィールド参照 > の形式と規則について説明します。

< フィールド参照 > では，VariableArray 型のプロパティの要素を参照する場合の，フィールドの参照を表現します。

(1) < フィールド参照 > の形式

```
<フィールド参照> ::= <要素参照> <ピリオド> <フィールド名>
<フィールド名> ::= <プロパティ名>
```

(2) < フィールド参照 > についての規則

● < フィールド参照 > は，VariableArray 型のプロパティの要素を参照する場合だけ指定します。

● < フィールド名 > には，VariableArray 型のプロパティの < プロパティ名 > を指定します。
例えば，「VariableArray 型のプロパティ Authors の要素 Age が 30 である」という指定は，次のように記述します。

```
Authors[ANY].Age = 30
```

2.7.4 < ルーチンの起動 >

ここでは，< ルーチンの起動 > の形式と規則について説明します。

< ルーチンの起動 > には，edmSQL で提供する関数を起動するための表現を指定します。

edmSQL で使用するルーチンは，関数だけです。

edmSQL によって起動する関数を次に示します。なお，これらの関数名はすべて，< 予約されていないキーワード > として登録されています。

関数の種類

全文検索を実行するための関数

- contains 関数
- contains_with_score 関数
- score 関数
- extracts 関数

概念検索を実行するための関数

- concept_with_score 関数
- score_concept 関数

変換関数

- oiidstr 関数
- objref 関数
- oiid 関数

それぞれの関数の詳細については，「2.9　関数指定の構文規則」を参照してください。

(1) < ルーチンの起動 > の形式

```
<ルーチンの起動> ::= <ルーチン名> <引数リスト>
<引数リスト> ::= <左括弧> 〔 <引数>
                 〔 { <コンマ> <引数> }... 〕 〕 <右括弧>
<引数> ::= <値式>
         | <値式> AS <データ型指定>
```

```
<ルーチン名> ::= <関数指定>

<データ型指定> ::= INT
                 | INTEGER
                 | BOOL
                 | BOOLEAN
                 | STRING <左括弧> <符号なし数値> <右括弧>
                 | BINARY <左括弧> <バイナリ長> <右括弧>

<バイナリ長> ::= <符号なし数値>
               | <バイナリ長トークン>
```

< バイナリ長トークン > については,「2.4.4 < トークン >」を参照してください。

(2) < ルーチンの起動 > についての規則

- 関数の < 引数リスト > に指定できる引数の数は，< ルーチン名 > に指定した，それぞれの関数の仕様に従います。
- < 引数 > に指定できる < 一次子 > は，< ルーチン名 > に指定した関数の仕様に従います。各引数は，関数に規定された順序で指定する必要があります。
- < 引数リスト > に < 一次子 > として <? パラメタ > を指定する場合は，AS< データ型指定 > によって，データ型を明示してください。また，<? パラメタ > 以外の < 一次子 > に対して AS< データ型指定 > を指定することはできません。

(3) AS< データ型指定 > についての規則

AS< データ型指定 > に指定できるデータ型について説明します。関数の仕様に従って，適切なデータ型を指定してください。また，文字列型およびバイナリ型を指定する場合には，定義長も指定してください。

AS< データ型指定 > に指定できる値と指定例を，次の表に示します。

表 2-34 AS< データ型指定 > に指定できる値と指定例

引数のデータ型	指定する値	指定例
論理型の引数	BOOL BOOLEAN	? AS BOOL ? AS BOOLEAN
整数型の引数	INT INTEGER	? AS INT ? AS INTEGER
文字列型の引数	STRING(n ※ 1)	? AS STRING(3200) 定義長 3200 バイトの文字列型データ
バイナリ型の引数	BINARY(n ※ 2)	? AS BINARY(256) 定義長 256 バイトのバイナリ型データ ? AS BINARY(256k) 定義長が 256 キロバイトのバイナリ型データ ? AS BINARY(256m) 定義長が 256 メガバイトのバイナリ型データ ? AS BINARY(2g) 定義長が 2 ギガバイトのバイナリ型データ

注※ 1 文字列型データの定義長を指定します。単位はバイトです。

注※ 2 バイナリ型データの定義長を指定します。次の単位が指定できます。

- (省略)：バイト
- k：キロバイト
- m：メガバイト
- g：ギガバイト

数字と単位の間に < 区切り文字 > を入れることはできません。

2.7.5 < 数値関数 >

ここでは，< 数値関数 > の形式と規則について説明します。

(1) < 数値関数 > の形式

```
<数値関数> ::= <絶対値関数>
<絶対値関数> ::= ABS <左括弧> <値式> <右括弧>
```

(2) < 数値関数 > についての規則

< 数値関数 > としては，絶対値関数（ABS 関数）があります。< 数値関数 > は，< 結合条件 > には指定できません。また，<SELECT 句 > には指定できません。

(3) < 絶対値関数 > の詳細

< 絶対値関数 > の詳細について示します。

関数名

ABS

形式

ABS < 左括弧 > < 値式 > < 右括弧 >

引数

< 値式 >

整数型の値を指定します。

機能

< 値式 > の値の絶対値を算出します。

被演算子のデータ型

整数型

評価値

整数値（整数型）

2.7.6 < 集合関数 >

ここでは，< 集合関数 > の形式と規則について説明します。

< 集合関数 > には，検索結果集合に対する演算を実行するための表現を指定します。

(1) < 集合関数 > の形式

```
<集合関数> ::= COUNT <左括弧> <アスタリスク> <右括弧>
             | <一般集合関数>
<一般集合関数> ::= <集合関数種別> <左括弧> 〔 <集合指定子> 〕
                   <値式> <右括弧>
<集合関数種別> ::= COUNT
<集合指定子> ::= DISTINCT | ALL
```

(2) < 集合関数 > の種類

< 集合関数 > には，次の 2 種類があります。

● COUNT(*)
検索結果の個数を算出します。

● COUNT
検索結果のプロパティのキー数を算出します。

(3) < 集合指定子 > についての規則

● < 集合指定子 > では，演算対象である集合の要素に対して，同じ要素がある場合に，重複排除を実施するかどうかを指定します。DISTINCT か ALL を指定します。

● DISTINCT を指定した場合，演算の対象である要素に対して，重複排除が実施されてから，演算が実行されます。

● ALL を指定した場合，重複排除は実施されません。

● < 集合指定子 > を省略した場合は，「ALL」が仮定されます。

●「COUNT(*)」を指定した場合には，「DISTINCT」の指定は無視されます。

● < 集合指定子 > に「DISTINCT」を指定した場合，集合関数を選択項目とする問い合わせ指定の < 集合指定子 > 指定が，データベースによって制限されることがあります。

● アクセス制御機能を使用している場合，主問い合わせの選択項目に集合関数は指定できません。

(4) COUNT(*) 関数の詳細

COUNT(*) 関数の詳細について説明します。

関数名
COUNT(*)

形式
COUNT < 左括弧 > < アスタリスク > < 右括弧 >

引数
なし (アスタリスク)

機能
検索結果集合の要素の数を算出します。

評価値
整数値（整数型）

属性
この関数は，<SELECT 句 > に指定できます。
また，<ORDER BY 句 > のソートキーとして指定できます。

(5) COUNT 関数の詳細

COUNT 関数の詳細について説明します。

関数名
COUNT

形式
COUNT < 左括弧 > 〔< 集合指定子 >〕< 値式 > < 右括弧 >

引数

< 値式 >

整数型，文字列型，論理型の < プロパティ指定 > だけが指定できます。

機能

検索結果集合から，< 値式 > に指定した要素の数を算出します。

評価値

整数値（整数型）

属性

この関数は，<SELECT 句 > に指定できます。

また，<ORDER BY 句 > のソートキーとして指定できます。

2.7.7 < 値式 >

ここでは，< 値式 > の形式と規則について説明します。

< 値式 > とは，評価として値を得ることができる式の定義です。

(1) < 値式 > の形式

```
<値式> ::= <一次子>
          |<符号> <一次子>
          |<値式> <+符号> <値式>
          |<値式> <-符号> <値式>
          |<値式> <アスタリスク> <値式>
          |<値式> <斜線> <値式>
          |<値式> <文字列連結演算子> <値式>
<一次子> ::= <左括弧> <値式> <右括弧>
            |<プロパティ指定>
            |<符号なし値指定>|<集合関数>|<数値関数>
            |<要素参照>
            |<ルーチンの起動>|<フィールド参照>
<符号なし値指定> ::= <符号なし数値リテラル>
                    |<文字列リテラル>
                    |<疑問符>
                    |<論理リテラル>
<値指定> ::= 〔<符号>〕<符号なし数値リテラル>
            |<文字列リテラル>|<疑問符>|<論理リテラル>
```

(2) < 値式 > についての規則

● < 値式 > 内の演算子は，次の順序で評価されます。

演算子の評価順序

1. 括弧内
2. 単項演算子 < 符号 > の <+ 符号 > または <- 符号 >
3. < アスタリスク > または < 斜線 >
4. 二項演算子の <+ 符号 >，<- 符号 > または < 結合演算子 >

● 演算子の両側に指定する < 値式 > には，同じデータ型のものを指定してください。

● 演算の途中でオーバーフローが発生したり，0 で除算を実行したりした場合，null 値が返却されるかエラーが発生するかについては，HiRDB の仕様に依存します。

● 指定できる演算子のネストの深さは，HiRDB の制限に従います。

● < 値指定 > と < 符号なし値指定 > の違いは，符号が付けられるかどうかです。< 値式 > で指定できるの

は，< 符号なし値指定 > です。符号を付ける場合は，< 値式 > を < 符号 > と < 符号なし値指定 > で構成してください。
< 値指定 > は，<In 述語 > などで使用します。

(3) 演算子の詳細

< 値式 > で指定する演算子の詳細を，次の表に示します。

表 2-35 < 値式 > で指定する演算子の詳細

演算子	意味	被演算子のデータ型	評価値
< 符号 >（単項演算子 +）	値に正符号を付けます。評価値に変化はありません。	整数型	整数値（整数型）
< 符号 >（単項演算子 -）	値の符号を反転します。	整数型	整数値（整数型）
<+ 符号 >（+）	被演算子を加算します。	整数型	整数値（整数型）
<- 符号 >（-）	左辺の被演算子から右辺の被演算子を減算します。	整数型	整数値（整数型）
< アスタリスク >（*）	被演算子を乗算します。	整数型	整数値（整数型）
< 斜線 >（/）	左辺の被演算子を右辺の被演算子で除算します。	整数型	整数値（整数型）
< 文字列連結演算子 >（||）	文字列を連結します。	文字列型	文字列値（文字列型）

注 edmSQL の <+ 符号 > および < アスタリスク > では多項演算は実行できません。このため，多項演算と同じ演算を実行したい場合は，2 項演算を組み合わせてください。

2.8 述語の構文規則

ここでは，述語の構文について説明します。

FROM 句の ON 条件や WHERE 句に指定する検索条件は，述語によって表現します。述語では，次のような演算ができます。

- 比較演算
- 論理演算
- Between 演算
- In 演算
- Like 演算
- Null 演算
- Exists 演算

これらの演算では，評価が「真」,「偽」または「不定」として得られます。

形式

!! **<検索条件>の形式**

```
<検索条件> ::= <左括弧> <検索条件> <右括弧>
             | <述語>
             | NOT <検索条件>
             | <検索条件> OR <検索条件>
             | <検索条件> AND <検索条件>
```

!! **<述語>の形式**

```
<述語> ::= <比較述語>
         | <論理述語>
         | <Between述語>
         | <In述語>
         | <Like述語>
         | <Null述語>
         | <Exists述語>
```

!! **<比較述語>の形式**

```
<比較述語> ::= <値式> <比較演算子> <比較述語値式>
<比較述語値式> ::= <値式> | <副問い合わせ>
<比較演算子> ::= <等号演算子>
               | <不等号演算子>
               | <小なり演算子>
               | <大なり演算子>
               | <小なり等号演算子>
               | <大なり等号演算子>
```

!! **<論理述語>の形式**

```
<論理述語> ::= <値式> IS TRUE
```

!! **<Between述語>の形式**

```
<Between述語> ::= <値式>
                  〔 NOT 〕 BETWEEN <値式> AND <値式>
```

!! **<In述語>の形式**

```
<In述語> ::= <値式> 〔 NOT 〕 IN <In述語値>
<In述語値> ::= <副問い合わせ>
             | <左括弧> <In項目リスト> <右括弧>
<In項目リスト> ::= <値指定> { <コンマ> <値指定> }...
```

!! **<Like述語>の形式**

```
<Like述語> ::= <文字列Like述語>
<文字列Like述語> ::= <値式>
                     〔 NOT 〕 <Like種別> <パターン文字列>
                     〔 ESCAPE <エスケープ文字> 〕
<Like種別> ::=  LIKE
              | XLIKE
<パターン文字列> ::= <値指定>
<エスケープ文字> ::= <値指定>
```

!! **<Null述語>の形式**

```
<Null述語> ::= <値式> IS 〔 NOT 〕 NULL
```

!! **<Exists述語>の形式**

```
<Exists述語> ::= EXISTS <副問い合わせ>
```

2.8.1 < 検索条件 >

ここでは，< 検索条件 > の形式と規則について説明します。

< 検索条件 > は，<FROM 句 > の ON 条件式 < 結合指定 > や，<WHERE 句 > に指定します。< 検索条件 > では，複数の < 検索条件 > の論理演算を指定できます。

(1) < 検索条件 > の形式

```
<検索条件> ::= <左括弧> <検索条件> <右括弧>
             | <述語>
             | NOT <検索条件>
             | <検索条件> OR <検索条件>
             | <検索条件> AND <検索条件>
```

(2) < 検索条件 > についての規則

● < 検索条件 > の論理演算は，次の順序で評価されます。

論理演算の評価順序

1. 括弧内
2. NOT
3. AND
4. OR

● 論理演算で指定できるネスト数については，HiRDB の制限に従います。

● edmSQL で提供している論理演算子 AND および OR は，2 項演算子です。複数の被演算子を指定したい場合は，演算子を複数組み合わせて指定してください。
例えば，「a」，「b」，「c」および「d」の AND 演算は，「a AND b AND c AND d」または「((a AND b) AND c) AND d」のように表記してください。

(3) 論理演算子の詳細

否定演算

演算子：NOT

形式：NOT < 検索条件 >

否定演算の演算結果を次の表に示します。

表 2-36 否定演算の演算結果

NOT	演算結果
真	偽
偽	真

NOT	演算結果
不定	不定

論理和演算

演算子：OR

形式：< 検索条件 > OR < 検索条件 >

論理和演算の演算結果を次の表に示します。

表 2-37 論理和演算の演算結果

OR	真	偽	不定
真	真	真	真
偽	真	偽	不定
不定	真	不定	不定

論理積演算

演算子：AND

形式：< 検索条件 > AND < 検索条件 >

論理積演算の演算結果を次の表に示します。

表 2-38 論理積演算の演算結果

AND	真	偽	不定
真	真	偽	不定
偽	偽	偽	偽
不定	不定	偽	不定

2.8.2 < 述語 >

ここでは，< 述語 > の形式と規則について説明します。

< 述語 > では，edmSQL で提供されている述語の一覧が定義されています。

(1) < 述語 > の形式

```
<述語> ::= <比較述語>
         | <論理述語>
         | <Between述語>
         | <In述語>
         | <Like述語>
         | <Null述語>
         | <Exists述語>
```

2.8.3 < 比較述語 >

ここでは，< 比較述語 > の形式と規則について説明します。

< 比較述語 > は，演算子の両辺の比較によって結果を返却します。

(1) < 比較述語 > の形式

```
<比較述語> ::= <値式> <比較演算子> <比較述語値式>
<比較述語値式> ::= <値式> | <副問い合わせ>
<比較演算子> ::= <等号演算子> | <不等号演算子> | <小なり演算子>
```

| <大なり演算子> | <小なり等号演算子>
| <大なり等号演算子>

(2) < 比較述語 > の評価

- 演算子が示す比較演算子が，両辺の被演算子の関係を満たす場合，「真」になります。
- 演算子の示す比較条件が，両辺の被演算子の関係を満たさない場合，「偽」になります。
- 指定した被演算子が null 値の場合，「不定」になります。
- 指定した被演算子が < 副問い合わせ > であり，その検索結果集合が空の場合，「不定」になります。

(3) < 比較述語 > の規則

- < 比較演算子 > の両辺に指定する被演算子には，同じデータ型の値を指定してください。値が null 値である場合も，同じデータ型にしてください。
- < 比較演算子 > によって比較できるのは，データ型が論理型，整数型，文字列型およびオブジェクト型の値です。ただし，演算子によって使用できるデータ型は異なります。
- 比較できるデータ型については，HiRDB の制限に従います。
- < 副問い合わせ > が指定できるのは，< 比較演算子 > の右辺だけです。
- < 副問い合わせ > を < 比較演算子 > の被演算子に指定する場合，< 副問い合わせ > で指定できる < 選択項目 > は一つだけです。また，< 副問い合わせ > の検索結果として得られる検索件数も，1 件または 0（空集合）になる必要があります。
- <FROM 句 > に指定する < 比較述語 > には，< 副問い合わせ > は指定できません。
- VariableArray 型のプロパティを指定する場合は，必ず < フィールド参照 >（要素指定の添え字は ANY）を指定して，比較できるデータ型のプロパティとして指定してください。また，VariableArray 型のプロパティのフィールド参照同士の比較はできません。

(4) < 比較述語 > で使用する演算子の詳細

演算子の詳細を，次の表に示します。

表 2-39 < 比較述語 > で使用できる演算子の詳細

演算子	意味	比較可能なデータ型
< 等号演算子 >（=）	値が等しい場合に「真」になります。	• 論理型 • 整数型 • 文字列型 • オブジェクト型
< 不等号演算子 >（<>）	値が等しくない場合に「真」になります。	• 論理型 • 整数型 • 文字列型
< 小なり演算子 >（<）	左辺の値が右辺の値よりも小さい場合に「真」になります。	• 整数型 • 文字列型
< 大なり演算子 >（>）	左辺の値が右辺の値よりも大きい場合に「真」になります。	• 整数型 • 文字列型
< 小なり等号演算子 >（<=）	左辺の値が右辺の値よりも小さいか，等しい場合に「真」になります。	• 整数型 • 文字列型
< 大なり等号演算子 >（>=）	左辺の値が右辺の値よりも大きいか，等しい場合に「真」になります。	• 整数型 • 文字列型

2.8.4 < 論理述語 >

ここでは，< 論理述語 > の形式と規則について説明します。

< 論理述語 > では，論理型の値を評価して結果を返却します。

(1) < 論理述語 > の形式

```
<論理述語> ::= <値式> IS TRUE
```

(2) < 論理述語 > の評価

● < 値式 > の論理型の値が，真（TRUE）の場合に「真」になります。

● < 値式 > の論理型の値が，真（TRUE）以外の場合，「偽」になります。

(3) < 論理述語 > の規則

< 値式 > に指定できるのは，戻り値が論理型の < ルーチンの起動 > だけです。なお，< 論理述語 > に指定する < ルーチンの起動 > で呼び出せる関数は，HiRDB Text Search Plug-in，または HiRDB XML Extension と連携するための，次の関数だけです。

- contains 関数
- contains_with_score 関数
- concept_with_score 関数

例えば，WHERE 句に，次のように指定できます。

```
WHERE contains(edmProp_StIndex, '{"コンピュータ"}') IS TRUE
```

2.8.5 <Between 述語 >

ここでは，<Between 述語 > の形式と規則について説明します。

形式

```
<Between述語> ::= <値式>
                  〔 NOT 〕 BETWEEN <値式> AND <値式>
```

(1) <Between 述語 > の評価

● 第 1 の < 値式 > を「値式 1」，第 2 の < 値式 > を「値式 2」，第 3 の < 値式 > を「値式 3」とした場合，「値式 1」の値が，「値式 2」と「値式 3」の範囲内の値の場合，「真」になります。範囲外の場合，「偽」になります。

● 第 1 の < 値式 > を「値式 1」，第 2 の < 値式 > を「値式 2」，第 3 の < 値式 > を「値式 3」とした場合，<Between 述語 > は，次の表記と同じ意味を表します。

```
〔NOT〕((値式2 <= 値式1) AND (値式1 <= 値式3))
```

(2) <Between 述語 > の規則

指定順序が「Between 値式 2 AND 値式 3」の場合に，「値式 2 ≦ 値式 3」である必要はありません。

2.8.6 <In 述語 >

ここでは，<In 述語 > の形式と規則について説明します。

(1) <In 述語 > の形式

```
<In述語> ::= <値式> 〔 NOT 〕 IN <In述語値>
<In述語値> ::= <副問い合わせ>
             | <左括弧> <In項目リスト> <右括弧>
<In項目リスト> ::= <値指定> { <コンマ> <値指定> }...
```

(2) <In 述語 > の評価

- 第 1 の < 値式 > が，<In 述語値 > の中の任意の値と一致する場合，<In 述語 > は「真」になります。
  NOT を指定した場合は，第 1 の < 値式 > が <In 述語値 > の中の任意の値と一致するとき，「偽」になります。
- 第 1 の < 値式 > が，<In 述語値 > の中のすべての値と一致しない場合，<In 述語 > は「偽」になります。
  NOT を指定した場合は，第 1 の < 値式 > が，<In 述語値 > の中のすべての値と一致しないとき，「真」になります。

(3) <In 述語 > の規則

- 第 1 の < 値式 > のデータ型と，<In 述語値 > のデータ型は一致させてください。指定できるデータ型は，論理型，整数型，文字列型およびオブジェクト型です。
- 指定できるデータ型や <In 項目リスト > に指定できる < 値指定 > については，HiRDB の制限に従います。
- <In 項目リスト > を指定する場合は，第 1 の < 値式 > に < 値指定 > は指定できません。
- <In 述語 > で < 副問い合わせ > の検索結果と比較する場合，< 副問い合わせ > で指定できる < 選択項目 > は一つだけです。
- < 副問い合わせ > の検索結果集合が空集合の場合は，<In 述語 > は「偽」になります。
  NOT を指定した場合に，< 副問い合わせ > の検索結果集合が空集合のときは，「真」になります。

2.8.7 <Like 述語 >

ここでは，<Like 述語 > の形式と規則について説明します。

(1) <Like 述語 > の形式

```
<Like述語> ::= <文字列Like述語>
<文字列Like述語> ::= <値式>
                     〔 NOT 〕 <Like種別> <パターン文字列>
                     〔 ESCAPE <エスケープ文字> 〕
<Like種別> ::= LIKE
             | XLIKE
<パターン文字列> ::= <値指定>
<エスケープ文字> ::= <値指定>
```

(2) パターンの詳細

< パターン文字列 >

- _：1 文字の任意の文字を示します。
- %：0 文字以上の任意の文字数の文字列を示します。

< エスケープ文字 >

< エスケープ文字 > は，パターン文字列中に「_」や「%」を記述したい場合に指定する文字です。< エスケープ文字 > には，任意の 1 文字を指定します。

(3) <Like 述語 > の評価

- < 値式 > の値が，< パターン文字列 > で表すパターンと一致した場合，「真」になります。
  NOT を指定した場合は，< 値式 > の値が，< パターン文字列 > の表すパターンと一致したとき，「偽」になります。
- < 値式 > の値が，< パターン文字列 > の表すパターンと一致しない場合，「偽」になります。
  NOT を指定した場合，< 値式 > の値が，< パターン文字列 > の表すパターンと一致しないとき，「真」になります。

(4) <Like 述語 > の規則

- < 値式 > に指定できるのは，文字列型の値だけです。
- < パターン文字列 > に指定できるのは，文字列型の値だけです。
- < エスケープ文字 > に指定できるのは，文字列型の値だけです。
- <Like 述語 > は，<FROM 句 > には指定できません。
- < 値式 >，< パターン文字列 >，< エスケープ文字 > に指定できる値については，HiRDB の制限に従います。また，処理性能を上げるための指定方法については，HiRDB の使用方法に従ってください。
- 指定した文字の認識のされ方は，文字列型との HiRDB のデータ型との対応に依存します。

(5) <Like 種別 > の規則

- <Like 種別 > が LIKE の場合，パターン文字列とのパターン一致で，大文字・小文字を区別して評価します。
- <Like 種別 > が XLIKE の場合，パターン文字列とのパターン一致で，大文字・小文字を区別しないで評価します。

2.8.8 <Null 述語 >

ここでは，<Null 述語 > の形式と規則について説明します。

(1) <Null 述語 > の形式

```
<Null述語> ::= <値式> IS 〔 NOT 〕 NULL
```

(2) <Null 述語 > の評価

- < 値式 > が null 値の場合に「真」になります。
  NOT を指定した場合は，< 値式 > が null 値のとき，「偽」になります。
- < 値式 > が null 値でない場合に「偽」になります。
  NOT を指定した場合は，< 値式 > が null 値でないとき，「真」になります。

(3) <Null 述語 > の規則

- < 値式 > に使用できるのは，データ型が Boolean 型，Integer32 型，String 型のプロパティ，およびオブジェクトリファレンスであるプロパティです。ただし，Boolean 型のプロパティは，null 値を取ることができません。
- 指定できるデータ型の制限は HiRDB に従います。

2.8.9 <Exists 述語 >

ここでは，<Exists 述語 > の形式と規則について説明します。

（1） <Exists 述語 > の形式

```
<Exists述語> ::= EXISTS <副問い合わせ>
```

（2） <Exists 述語 > の評価

- < 副問い合わせ > の検索結果が，空集合でなければ「真」になります。
- < 副問い合わせ > の検索結果が，空集合の場合は「偽」になります。

（3） <Exists 述語 > の規則

- < 副問い合わせ > の検索結果が空集合とは，検索結果の件数が 0 件であることです。つまり，検索結果の件数が 1 以上の場合に，<Exists 述語 > は「真」になります。
- <Exists 述語 > の < 副問い合わせ > では，SELECT 句の < 選択項目 > には，< アスタリスク > が指定できます。これ以外の < 選択項目 > に，< アスタリスク > は指定できません。

2.9 関数指定の構文規則

edmSQL では，基本的な SQL の文法で表現できない DocumentBroker の拡張検索機能を，関数として提供します。

なお，この節で説明する < 関数 > には，SQL の基本機能として提供されている < 集合関数 > および < 数値関数 > は含みません。

2.9.1 edmSQL が提供する関数の概要

edmSQL が提供する関数には，HiRDB の拡張検索機能を使用するための関数と，edmSQL で独自に拡張した関数があります。

それぞれの関数について説明します。

● **HiRDB の拡張検索機能（**HiRDB Text Search Plug-in**，または** HiRDB XML Extension **および** HiRDB Text Search Plug-in Conceptual Extension **を使用した検索の機能）を使用するための関数（**DBMS **関数）**
DBMS 関数には，次の 2 種類があります。
- 全文検索関数（概念検索を含まない全文検索をするための関数）
- 概念検索関数

DBMS 関数は，データベース上で，HiRDB の制限に従って実行されます。

● **edmSQL で独自に拡張した関数（**edmSQL **関数）**
edmSQL 関数として提供されているのは，変換関数です。

edmSQL 関数は，データベースへのアクセスの前後で実行されます。

2.9.2 文書検索関数（DBMS 関数）

ここでは，< 文書検索関数 > の形式と規則について説明します。

（1） < 文書検索関数 > の形式

!! **<文書検索関数>の形式**
```
<文書検索関数> ::= <全文検索関数>
                 | <概念検索関数>
```

!! **<全文検索関数>の形式**
```
<全文検索関数> ::= <contains関数>
                 | <contains_with_score関数>
                 | <score関数>
                 | <extracts関数>
```

!! **<contains関数>の形式**
```
<contains関数> ::= contains <左括弧><プロパティ指定>
                        <コンマ><全文検索条件><右括弧>
```

!! **<contains_with_score関数>の形式**
```
<contains_with_score関数> ::= contains_with_score
                              <左括弧><プロパティ指定>
                              <コンマ><全文検索条件><右括弧>
```

!! **<score関数>の形式**
```
<score関数> ::= score <左括弧><プロパティ指定><右括弧>
```

!! **<extracts関数>の形式**
形式1

```
<extracts関数> ::= extracts <左括弧><プロパティ指定><コンマ>
                        <抽出対象構造文字列> <コンマ>
                        <全文検索条件> <コンマ>
                        <ハイライトタグ文字列>
                        [<コンマ><ドキュメントタイプ>]<右括弧>
```

形式2

```
<extracts関数> ::= extracts <左括弧><全文検索機能付き文字列型プロパティ>
                        <右括弧>
```

!! **<概念検索関数>の形式**

```
<概念検索関数> ::=  <concept_with_score関数>
                  |<score_concept関数>
```

!! **<concept_with_score関数>の形式**

```
<concept_with_score関数> ::= concept_with_score
                             <左括弧><プロパティ指定>
                             <コンマ><概念検索条件><右括弧>
```

!! **<score_concept関数>の形式**

```
<score_concept関数> ::= score_concept
                             <左括弧><プロパティ指定><右括弧>
```

!! **<全文検索条件><概念検索条件><抽出構造文字列>および<ハイライトタグ文字列>の形式**

```
<全文検索条件> ::= <文字列リテラル> | <?パラメタ>
<概念検索条件> :: = <?パラメタ>
<抽出対象構造文字列> ::= <文字列リテラル> | <?パラメタ>
<ハイライトタグ文字列> ::= <文字列リテラル> | <?パラメタ>
```

< 全文検索条件 >，< 概念検索条件 >，< 抽出対象構造文字列 > および < ハイライトタグ文字列 > の指定方法については，マニュアル「HiRDB Text Search Plug-in」，またはマニュアル「HiRDB XML Extension」を参照してください。

(2) < 文書検索関数 > の概要

< 文書検索関数 > には，次の機能があります。

- 全文検索機能（概念検索を含まない全文検索）
- 概念検索機能

< 文書検索関数 > では，次に示す関数が定義されています。なお，以降，関数の説明で使用する "< 全文検索関数 >" という表記は，概念検索の機能を持たない全文検索を実行するための関数を表します。

全文検索関数

- contains 関数
- contains_with_score 関数
- score 関数
- extracts 関数

概念検索関数

- concept_with_score 関数
- score_concept 関数

なお，全文検索関数を実行する場合は，HiRDB Text Search Plug-in，または HiRDB XML Extension が必要です。

また，概念検索関数を実行する場合は，HiRDB Text Search Plug-in，または HiRDB XML Extension のどちらかと，HiRDB Text Search Plug-in Conceptual Extension が必要です。

(3) < 文書検索関数 > の規則

● < 文書検索関数 > は，HiRDB Text Search Plug-in，または HiRDB XML Extension の機能によって提供されている検索機能を edmSQL で使用するための関数です。
この関数で実行する検索の検索条件の指定方法については，マニュアル「HiRDB Text Search Plug-in」，またはマニュアル「HiRDB XML Extension」を参照してください。ただし，関数の第 1 引数には，列ではなく DocumentBroker のプロパティを指定します。

● 第 1 引数の < プロパティ指定 > に指定できるのは，その関数で指定できる < 特殊なプロパティ > または全文検索機能付き文字型プロパティです。

● < 文書検索関数 > は，< 結合条件 > には指定できません。

(4) < 全文検索関数 > の概要

全文検索関数には，次の種類があります。なお，この関数で実行する全文検索に，概念検索は含まれません。

● contains 関数
全文検索を実行します。

● contains_with_score 関数
全文検索を実行すると同時に，スコア値を算出します。

● score 関数
contains_with_score 関数で算出したスコア値を取得します。

● extracts 関数
文書からテキストデータを抽出します。抽出するデータは，構造を持つ文書の特定の構造にハイライトタグを埋め込んだデータです。

(5) <contains 関数 > の詳細

contains 関数の詳細について説明します。

関数名

contains

形式

```
contains <左括弧><プロパティ指定><コンマ><全文検索条件><右括弧>
```

引数

< プロパティ指定 >（< 特殊なプロパティまたは全文検索機能付き文字列型プロパティ >）

●特殊なプロパティの場合

検索対象の全文検索機能付き文書クラスが特定できる形式でプロパティを指定します。
指定できるのは，次の < 特殊なプロパティ > です。

- edmProp_TextIndex プロパティ
- edmProp_ConceptTextIndex プロパティ

●全文検索機能付き文字列型プロパティの場合

全文検索機能付き文字列型プロパティを指定します。

< 全文検索条件 >（文字列型の値：STRING(32000)）

全文検索条件を指定します。
検索タームを含む文書が検索されます。

< 全文検索条件 > には，検索タームのほか，構造指定検索，同義語異表記展開検索，近傍条件検索などを実行するための検索条件を文字列で指定します。指定方法については，マニュアル「HiRDB Text Search Plug-in」，またはマニュアル「HiRDB XML Extension」を参照してください。
< 全文検索条件 > に <? パラメタ > を指定する場合は，次のように AS< データ型指定 > をします。

```
? AS STRING(32000)
```

機能

< プロパティ指定 > で指定したプロパティに対応する文書のコンテンツに対して，< 全文検索条件 > で指定した条件で，全文検索を実行します。

評価値

論理型（評価）

属性

この関数は，<WHERE 句 > の < 検索条件 > に指定できます。

（6） <contains_with_score 関数 > の詳細

contains_with_score 関数の詳細について説明します。

関数名

```
contains_with_score
```

形式

```
contains_with_score <左括弧><プロパティ指定><コンマ><全文検索条件><右括弧>
```

引数

< プロパティ指定 >（< 特殊なプロパティ >）

検索対象の全文検索機能付き文書クラスが特定できる形式でプロパティを指定します。
指定できるのは，次の < 特殊なプロパティ > です。

- edmProp_TextIndex プロパティ
- edmProp_ConceptTextIndex プロパティ

< 全文検索条件 >（文字列型の値：STRING(32000)）

全文検索条件を指定します。全文検索条件に指定した検索タームが含まれる文書が検索されます。定義長は 32,000 バイトです。
< 全文検索条件 > には，検索タームのほか，構造指定検索，同義語異表記展開検索，近傍条件検索などの検索を実行するための検索条件を文字列で指定します。指定方法については，マニュアル「HiRDB Text Search Plug-in」，またはマニュアル「HiRDB XML Extension」を参照してください。
< 全文検索条件 > に <? パラメタ > を指定する場合は，次のように AS< データ型指定 > をします。

```
? AS STRING(32000)
```

機能

< プロパティ指定 > で指定したプロパティに対応する文書に対して，< 全文検索条件 > で指定した条件で，全文検索を実行します。
このとき，検索結果として取得した文書から，全文検索条件が含まれる割合を，スコアとして算出します。
算出したスコアの取得には，<score 関数 > を使用します。

評価値

論理型（評価）

属性

この関数は，<WHERE 句 > の < 検索条件 > に指定できます。

（7） <score 関数 > の詳細

<score 関数 > の詳細について説明します。

関数名

score

形式

score < 左括弧 >< プロパティ指定 >< 右括弧 >

引数

< プロパティ指定 >（< 特殊なプロパティ >）

スコアの値を取得する全文検索機能付き文書クラスが特定できる形式でプロパティを指定します。

指定できるのは，次の < 特殊なプロパティ > です。

- edmProp_TextIndex プロパティ
- edmProp_ConceptTextIndex プロパティ

機能

<contains_with_score 関数 > で検索結果として取得した文書に対して，文書の検索条件に対するスコア値（全文検索条件が含まれる割合に応じた値）を返却します。

評価値

スコア値（整数型）

属性

この関数は，<SELECT 句 > の < 選択項目 > に指定できます。

（8） <extracts 関数 > の詳細

<extracts 関数 > は，検索結果として取得する文書から，テキストデータを抽出する関数です。

<extracts 関数 > は，次の関数と組み合わせて使用することで，全文検索条件を満たした文書のテキストデータを抽出できます。

- contains 関数
- contains_with_score 関数

また，次の関数と組み合わせて使用することで，全文検索条件を満たした全文検索機能付き文字列型プロパティを抽出できます。

- contains 関数

<extracts 関数 > の詳細について説明します。

関数名

extracts

形式 1

extracts <左括弧><プロパティ指定><コンマ><抽出対象構造文字列><コンマ><全文検索条件><コンマ><ハイライトタグ文字列><コンマ><ドキュメントタイプ><右括弧>

形式 2

extracts <左括弧><全文検索機能付き文字列型プロパティ><右括弧>

引数

< プロパティ指定 >（< 特殊なプロパティ >）

抽出対象の全文検索機能付き文書クラスが特定できる形式でプロパティを指定します。
指定できるプロパティは，次の < 特殊なプロパティ > です。

- edmProp_TextIndex プロパティ
- edmProp_ConceptTextIndex プロパティ

< 全文検索機能付き文字列型プロパティ >

全文検索機能付き文字列型プロパティを指定します。

< 抽出対象構造文字列 >（文字列型の値：STRING(1024)）

抽出対象にする構造を表す文字列を指定します。
構造を指定しない場合は，「"」（空文字列）を指定してください。
< 抽出対象構造文字列 > に <? パラメタ > を指定する場合は，次のように AS< データ型指定 > をします。

```
? AS STRING(1024)
```

< 全文検索条件 >（文字列型の値：STRING(32000)）

ハイライト表示する検索タームを指定します。
条件を指定しない場合は，「"」（空文字列）を指定してください。この場合，ハイライトタグは挿入されません。
< 全文検索条件 > に <? パラメタ > を指定する場合は，次のように AS< データ型指定 > をします。

```
? AS STRING(32000)
```

< ハイライトタグ文字列 >（文字列型の値：STRING(255)）

ハイライト表示に使用するタグを表す文字列を指定します。
ハイライトタグを指定しない場合は，「"」（空文字列）を指定してください。この場合，ハイライトタグは挿入されません。
< ハイライトタグ文字列 > に <? パラメタ > を指定する場合は，次のように AS< データ型指定 > をします。

```
? AS STRING(255)
```

< ドキュメントタイプ >（'XML'）

XML 形式で出力する場合に指定します。
省略すると SGML 形式で出力します。

機能

< プロパティ指定 > で指定したプロパティに対応する文書のコンテンツから，< 抽出指定構造文字列 > および < 全文検索条件 > で指定した文字列に < ハイライトタグ文字列 > に指定したタグ（文字列）を埋め込んだ状態で抽出します。
HiRDB Text Search Plug-in，または HiRDB XML Extension の「SGML 出力」に対応します。
また，< 全文検索機能付き文字列型プロパティ > で指定した全文検索機能付き文字列型プロパティの値を抽出します。

評価値

抽出したテキストまたはプロパティの値（バイナリ型データ）

抽出するデータは，HiRDB Text Search Plug-in，または HiRDB XML Extension の BLOB 型で 2 ギガバイトまでのデータです。ただし，検索結果としては，API で扱うことができる String 型の値に変換したものを取得できます。

属性

この関数は，<SELECT 句 > の < 選択項目 > に指定できます。
ただし，この関数を <SELECT 句 > に指定した場合，< 集合指定子 > として DISTINCT は指定できません。

(9) < 概念検索関数 > の概要

概念検索関数には，次の種類があります。

● concept_with_score 関数
概念検索を実行すると同時に，スコア値を算出します。

● score_concept 関数
concept_with_score 関数で算出したスコア値を取得します。

(10) <concept_with_score 関数 > の詳細

<concept_with_score 関数 > の詳細について説明します。

関数名

concept_with_score

形式

```
concept_with_score <左括弧><プロパティ指定><コンマ><概念検索条件><右括弧>
```

引数

< プロパティ指定 >（< 特殊なプロパティ >）
検索対象の全文検索機能付き文書クラスが特定できる形式でプロパティを指定します。
指定できるプロパティは，次の < 特殊なプロパティ > です。

- edmProp_ConceptTextInde

< 概念検索条件 >（バイナリ型のデータ：BINARY(5m)）
概念検索条件を指定します。概念検索条件には，種文章を指定します。種文章を含む < 概念検索条件 > を指定する場合には，必ず <? パラメタ > を使用して指定してください。また，ここで指定する <? パラメタ > には，構造指定検索や同義語異表記検索などの検索を実行するための検索条件も指定します。指定方法については，マニュアル「HiRDB Text Search Plug-in」，またはマニュアル「HiRDB XML Extension」を参照してください。
< 概念検索条件 > を <? パラメタ > で指定する場合は，次のように AS< データ型指定 > をします。

```
? AS BINARY(5m)
```

AS< データ型指定 > を含んだ，concept_with_score 関数を呼び出す論理述語は，次のように指定します。

```
concept_with_score(edmProp_ConceptTextInde,
                   ? AS BINARY(5m)) IS TRUE
```

機能

< プロパティ指定 > で指定したプロパティに対応する文書のコンテンツに対し，< 概念検索条件 > で指定した条件で，概念検索を実行します。

また，検索実行時に，検索結果として取得した文書から検索用特徴タームが含まれる割合をスコアとして算出します。
このスコアを取得する場合は，<score_concept 関数 > を使用します。

評価値
論理型（評価）

属性
この関数は，<WHERE 句 > の < 検索条件 > に指定できます。

（11）<score_concept 関数 > の詳細

<score_concept 関数 > の詳細について説明します。

関数名
score_concept

形式
score_concept < 左括弧 >< プロパティ指定 >< 右括弧 >

引数
< プロパティ指定 >（< 特殊なプロパティ >）
スコアの値を取得する全文検索機能付き文書クラスが特定できる形式でプロパティを指定します。
指定できるのは，次の < 特殊なプロパティ > です。
- edmProp_ConceptTextIndex プロパティ

機能
<concept_with_score 関数 > で検索結果として取得した文書に対して，文書の検索条件に対するスコア値（検索用特徴タームが含まれる割合に応じた値）を返却します。

評価値
スコア値（整数型）

属性
この関数は，<SELECT 句 > の < 選択項目 > に指定できます。

2.9.3 変換関数（edmSQL 関数）

< 変換関数 > は，オブジェクト型の値を，DocumentBroker クラスライブラリで扱えるデータ型に変換するための関数です。DocumentBroker クラスライブラリでは，オブジェクト型のプロパティの値は，すべて OIID 文字列として扱います。

< 変換関数 > には，< 選択可能な変換関数 > と < 検索可能な変換関数 > があります。

（1）< 変換関数 > の形式

!! <変換関数>の形式

```
<変換関数> ::= <選択可能な変換関数>
             | <検索可能な変換関数>

<選択可能な変換関数> ::= <oiidstr関数>

<検索可能な変換関数> ::= <objref関数>
                        | <oiid関数>
```

!! <oiidstr関数>の形式
```
<oiidstr関数> ::= oiidstr <左括弧> <プロパティ指定> <右括弧>
```

!! <objref関数>の形式
```
<objref関数> ::= objref<左括弧> <OIID文字列> <右括弧>
```

!! <oiid関数>の形式
```
<oiid関数> ::= oiid<左括弧> <OIID文字列> <右括弧>

<OIID文字列> ::= <文字列リテラル>
```

(2) < 変換関数 > の概要

< 変換関数 > は，データベースで処理する関数ではなく，DocumentBroker で edmSQL を発行した前後にデータの変換を実行する関数です。

< 変換関数 > には，< 選択可能な変換関数 > と < 検索可能な変換関数 > があります。< 選択可能な変換関数 > は，主問い合わせの < 選択項目 > だけに指定できます。< 検索可能な変換関数 > は，< 検索条件 > だけに指定できます。なお，< 変換関数 > は，< 結合条件 > には指定できません。

これら関数では，次の変換ができます。

● オブジェクトリファレンス（オブジェクト型の値）から OIID 文字列（文字列型の値）への変換

● OIID 文字列（文字列型の値）からオブジェクトリファレンス（オブジェクト型の値）への変換

● OIID 文字列から dmaProp_OIID プロパティの値への変換

(3) <oiidstr 関数 > の詳細

<oiidstr 関数 > の詳細について説明します。

関数名
oiidstr

形式
oiidstr < 左括弧 >< プロパティ指定 >< 右括弧 >

引数
< プロパティ指定 >（オブジェクト型の値）

機能
オブジェクト型の値であるオブジェクトリファレンスを，OIID 文字列表記に変換します。

評価値
文字列型の値。
OIID 文字列表記を，文字列型の値として返却します。

属性
< 選択可能な変換関数 >
この関数は，<SELECT 句 > の < 選択項目 > に指定できます。

(4) <objref 関数 > の詳細

<objref 関数 > の詳細について説明します。

関数名
objref

形式

objref < 左括弧 > <OIID 文字列 >< 右括弧 >

引数

<OIID 文字列 >（文字列型の値）

機能

<OIID 文字列 >（文字列型の値）を，オブジェクトリファレンス（オブジェクト型の値）に変換します。

評価値

オブジェクト型の値。
オブジェクトリファレンスを，オブジェクト型の値として返却します。

属性

< 検索可能な変換関数 >
この関数は，<WHERE 句 > の < 検索条件 > に指定できます。

（5） <oiid 関数 > の詳細

<oiid 関数 > の詳細について説明します。

関数名

oiid

形式

oiid < 左括弧 ><OIID 文字列 >< 右括弧 >

引数

<OIID 文字列 >（文字列型の値）

評価値

文字列型の値。
実際に dmaProp_OIID プロパティに格納されている形式（16 バイトの値）の文字列型の値を返却します。

属性

< 検索可能な変換関数 >
この関数は，<WHERE 句 > の < 検索条件 > に指定できます。

2.10 データ操作の構文規則

ここでは，データ操作を表現する構文について説明します。edmSQL で実行できるデータ操作は，ORDER BY 句による検索結果の並べ替えです。

形式

!! <ORDER BY句>の形式

```
<ORDER BY句> ::= ORDER BY <ソート指定リスト>
<ソート指定リスト> ::= <ソート指定> 〔 { <comma> <ソート指定> }... 〕
<ソート指定> ::= <ソートキー>  〔 <順序指定> 〕
<ソートキー> ::= <プロパティ指定>
               |<符号なし整数>
<順序指定> ::= ASC|DESC
```

2.10.1 <ORDER BY 句 >

ここでは，<ORDER BY 句 > の形式と規則について説明します。

<ORDER BY 句 > では，検索結果を昇順または降順に並べ替える場合の，ソート方法を指定します。<ORDER BY 句 > を省略した場合，検索結果の取得順序は不定になります。これは，検索結果が集合として管理されているためです。<ORDER BY 句 > では，この検索結果集合から要素を取り出す時のソート方法を指定します。

(1) <ORDER BY 句 > の形式

```
<ORDER BY句> ::= ORDER BY <ソート指定リスト>
<ソート指定リスト> ::= <ソート指定> 〔 { <コンマ> <ソート指定> }... 〕
<ソート指定> ::= <ソートキー>  〔 <順序指定> 〕
<ソートキー> ::= <プロパティ指定>
               |<符号なし整数>
<順序指定> ::= ASC|DESC
```

(2) < ソート指定リスト > についての規則

< ソート指定リスト > に指定できる < ソート指定 > の最大数は，HiRDB の制限に従います。

(3) < ソートキー > についての規則

- < ソートキー > に指定できるのは，< プロパティ指定 > またはソート項目を指定する番号である < 符号なし整数 > です。ただし，指定できる < ソートキー > のデータ型については，HiRDB の制限に従います。
- < ソートキー > を < プロパティ指定 > で指定する場合，指定できるのは，最も外側の SELECT 句（主問い合わせの SELECT 句）の < 選択項目 > に指定したプロパティだけです。< 副問い合わせ > の SELECT 句で指定した < 選択項目 > のプロパティは，指定できません。
- 最も外側の SELECT 句（主問い合わせの SELECT 句）の < 選択項目 > が < ルーチンの起動 > や < 集合関数 > の場合，< プロパティ指定 > で < ソートキー > は指定できません。この場合は，ソート項目指定番号を指定してください。ソート項目指定番号とは，検索結果として指定した < 選択項目 > のうち，どの選択項目を < ソートキー > とするかを，< 選択項目 > の出現番号で表現したものです。したがって，SELECT 句で先頭に指定した < 選択項目 > が「1」になります。
- 複数の < ソートキー > を指定した場合は，< ソートキー > を指定した順番（指定の左側からの順番）でソートします。

（4）< 順序指定 > についての規則

- < 順序指定 > では，< ソートキー > を並べる順序の方向を，昇順にするか，降順にするかを指定します。
- ASC は < ソートキー > を昇順に並べる場合に指定します。
- DESC は < ソートキー > を降順に並べる場合に指定します。
- < 順序指定 > を省略した場合は，ASC が仮定されます。

2.11 edmSQL の指定例

この節では，edmSQL の指定例について説明します。

2.11.1 属性検索

ここでは，属性検索（プロパティを指定した検索）での edmSQL の指定例を示します。

(1) 指定例で使用するクラスとプロパティ

この指定例で検索の対象になる文書は，次のようなクラスから作成された文書です。

文書 X

Document X クラスを基に作成された文書です。ユーザ定義プロパティとして，次の表に示すプロパティが設定されています。

表 2-40 Document X クラスのプロパティ

プロパティ名	内容
DocCode	文書を管理するためのコード
Title	タイトル
Author	著者
AuthorId	著者 ID
Abstract	概要
CreateDate	作成日

所有者一覧

OwnersList クラスを基に作成された文書です。ユーザ定義プロパティとして，次の表に示すプロパティが設定されています。

表 2-41 OwnersList クラスのプロパティ

プロパティ名	内容
DocCode	文書を管理するためのコード
Owner	所有者

辞書 Y

Dictionary Y クラスを基に作成された文書です。ユーザ定義プロパティとして，次の表に示すプロパティが設定されています。

表 2-42 Dictionary Y クラスのプロパティ

プロパティ名	内容
Name	名前
Birthday	誕生日
EntryId	エントリー ID

論文

myPaper クラスを基に作成された文書です。次の表に示すプロパティが設定されています。

表 2-43 myPaper クラスのプロパティ

プロパティ名		内容
Title		タイトル
Authors ※	Name	名前
	Org	所属

注※ VariableArray 型のプロパティです。

(2) 一つのクラスに対して実行する属性検索

ここでは，一つのクラスに対して実行する属性検索の指定例を示します。

(a) SELECT 句と FROM 句だけを指定する検索

< 検索対象 > として Document X クラスを，< 選択項目 > として Title プロパティ，Author プロパティおよび Abstract プロパティを取得する例を示します。

指定例

```
SELECT Title,Author,Abstract
FROM   "Document X"
```

(b) SELECT 句，FROM 句および WHERE 句を指定する検索

(a) に加えて，検索条件として WHERE 句に「Author プロパティが『日立太郎』で，CreateDate プロパティが 20000101 以上」という条件を指定する例を示します。

指定例

```
SELECT Title,Author,Abstract
FROM "Document X"
WHERE Author = '日立太郎' AND CreateDate > '20000101'
```

(c) 重複排除を指定する検索

Document X クラスから，Author プロパティを，重複を排除して取得する例を示します。

指定例

```
SELECT DISTINCT Author
FROM "Document X"
```

(d) 検索条件として OIID を指定した検索

Document X クラスから作成された文書から，OIID を指定して，Title プロパティ，Author プロパティおよび Abstract プロパティを取得する例を示します。なお，OIID は，oiid 関数を使用して指定します。

指定例

```
SELECT Title,Author,Abstract
FROM "Document X"
WHERE dmaProp_OIID = oiid('dma:///xxx/xxx/xxxxxxxxxx...xxx')
```

(3) 結合したクラスに対して実行する属性検索

ここでは，複数のクラスを結合した検索対象クラスに対して実行する属性検索の指定例を示します。

なお，edmSQL で実行できる結合の方法には，次の 2 種類があります。

- 内部結合
- 左外部結合

それぞれの場合の指定例を示します。

(a) 二つのクラスを内部結合した検索

Document X クラスと OwnersList クラスを内部結合して，Document X クラスの文書の Title プロパティ，Document X クラスの Author プロパティおよび OwnersList クラスの Owner プロパティを取得する例を示します。ここでは，次の二つの方法の指定例を示します。

● FROM 句に < クラス指定 > を並べて内部結合して，結合の条件を WHERE 句に指定する方法

● INNER JOIN を使用して，二つのクラスの結合を指定する方法

結合では，Document X クラスの DocCode プロパティと OwnersList クラスの DocCode プロパティが一致することを条件とします。

なお，Document X クラスの < 相関名 > として DX，OwnersList クラスの < 相関名 > として OL を使用します。

FROM 句に < クラス指定 > を並べて内部結合して，結合の条件を WHERE 句に指定する方法

次のように指定します。

指定例

```
SELECT DX.Title,DX.Author,OL.Owner
FROM "Document X" DX, OwnersList OL
WHERE DX.DocCode = OL.DocCode
```

INNER JOIN を使用して二つのクラスの結合を指定する検索

次のように指定します。

指定例

```
SELECT DX.Title,DX.Author,OL.Owner
FROM   "Document X" DX INNER JOIN OwnersList OL
           ON DX.DocCode = OL.DocCode
```

(b) 三つのクラスを内部結合した検索

Document X クラス，OwnersList クラスおよび Dictionary Y クラスを結合して，Document X クラスの Title プロパティと Document X クラスの Author プロパティ，Dictionary Y クラスの Birthday プロパティおよび OwnersList クラスの Owner プロパティを取得する例を示します。ここでは，次の二つの方法の指定例を示します。

● FROM 句に < クラス指定 > を並べて内部結合して，結合の条件を WHERE 句に指定する方法

● INNER JOIN を使用して，三つのクラスの結合を指定する方法

結合では，Document X クラスの DocCode プロパティと OwnersList クラスの DocCode プロパティが一致することと，Document X クラスの AuthorId プロパティと Dictionary Y クラスの EntryId プロパティが一致することを条件とします。

FROM 句に < クラス指定 > を並べて内部結合して，結合の条件を WHERE 句に指定する方法

次のように指定します。

指定例

```
SELECT DX.Title,DX.Author,DY.Birthday,OL.Owner
FROM  "Document X" DX,
              OwnersList OL,"Dictionary Y" DY
WHERE  DX.DocCode = OL.DocCode
 AND DX.AuthorId = DY.EntryId
```

INNER JOIN を使用して三つのクラスの結合を指定する検索

次のように指定します。

指定例

```
SELECT DX.Title,DX.Author,DY.Birthday,OL.Owner
FROM ("Document X" HD INNER JOIN OwnersList OL
 ON DX.DocCode = OL.DocCode)
 INNER JOIN "Dictionary Y" DY
  ON DX.AuthorId = DY.EntryId
```

(c) 二つのクラスを外部結合した検索

Document X クラスと OwnersList クラスを外部結合して，Document X クラスの文書の Title プロパティと Document X クラスの Author プロパティおよび OwnersList クラスの Title プロパティを取得する例を示します。

次のように指定します。

指定例

```
SELECT DX.Title,DX.Author,OL.Owner
FROM "Document X" DX LEFT OUTER JOIN OwnersList OL
        ON DX.DocCode = OL.DocCode
```

(d) 三つのクラスを外部結合した検索

Document X クラス，OwnersList クラスおよび Dictionary Y クラスを外部結合して，Document X クラスの Title プロパティ，Document X クラスの Author プロパティ，Dictionary Y クラスの Birthday プロパティおよび OwnersList クラスの Owner プロパティを取得する例を示します。

指定例

```
SELECT DX.Title,DX.Author,DY.Birthday,OL.Owner
FROM ("Document X" DX LEFT OUTER JOIN OwnersList OL
    ON DX.DocCode = OL.DocCode) LEFT OUTER JOIN
  "Dictionary Y" DY ON DX.AuthorId = DY.EntryId
```

(4) 副問い合わせを実行する属性検索

ここでは，副問い合わせを指定する場合の属性検索の指定例を示します。

(a) 比較述語を指定した副問い合わせ

OIID がわかっている Document X クラスのオブジェクトの CreateDate プロパティよりも，Birthday プロパティが新しい（値が大きい）Dictionary Y クラスのオブジェクトの，Name プロパティと Birthday プロパティを取得する例を示します。

指定例

```
SELECT Name, Birthday
FROM "Dictionary Y"
WHERE  Birthday > (SELECT CreateDate
        FROM "Document X"
        WHERE dmaProp_OIID =
               oiid( 'dma:///xxx/xxx/xxxxxxxxxx...xxx'))
```

(b) In 述語を指定した副問い合わせ

CreateDate プロパティが 19990101 よりも値が大きい Document X クラスのオブジェクトの AuthorId プロパティのどれかと，EntryId プロパティの値が等しい，Document X クラスの オブジェクトの Name プロパティと Birthday プロパティを取得する例を示します。

指定例

```
SELECT Name, Birthday
FROM "Dictionary Y"
WHERE EntryId IN (SELECT AuthorId
                      FROM "Document X"
                      WHERE CreateDate < '19990101')
```

(c) Exists 述語を指定した副問い合わせ

Dictionary クラスのオブジェクトの Name プロパティと Document X クラスのオブジェクトの Author プロパティが一致する文書が一つでもあれば，その Dictionary Y クラスの文書の Name プロパティと Birthday プロパティを取得する例を示します。

指定例

```
SELECT Name, Birthday
FROM "Dictionary Y" DY
WHERE EXISTS (SELECT *
                    FROM "Document X" DX
                    WHERE DY.Name = DX.Author)
```

(5) 検索結果を取得する順序を指定した属性検索

ここでは，取得する検索結果の順序を指定する場合の属性検索の指定例を示します。

Document X クラスの文書のうち，CreateDate プロパティの値が 20000101 よりも大きいオブジェクトを検索して，Title プロパティ，Author プロパティ，および CreateDate プロパティを取得します。なお，この例では，Author プロパティによって昇順に並べ替え，その中で CreateDate プロパティで降順に並べ替えて検索結果を取得します。まず，並べ替えのキーとしてソート項目指定番号を指定する例を示します。ソート項目番号は，Author プロパティは SELECT 句で 2 番目に指定されているので「2」，CreateDate プロパティは SELECT 句で 3 番目に指定されているので「3」になります。

指定例

```
SELECT Title,Author,CreateDate
FROM "Document X"
WHERE CreateDate > '20000101'
ORDER BY 2 ASC,3 DESC
```

これを，プロパティ名で指定すると，次のようになります。

指定例

```
SELECT Title,Author,CreateDate
FROM "Document X"
WHERE CreateDate > '20000101'
ORDER BY Author ASC,CreateDate DESC
```

(6) VariableArray 型のプロパティを指定した属性検索

ここでは，VariableArray 型のプロパティの検索方法について説明します。

(a) VariableArray 型のプロパティを取得する検索

ここでは，myPaper クラスの VariableArray 型のプロパティである Authors プロパティを取得する例を示します。

指定例

```
SELECT Title,Authors
FROM myPaper
WHERE CreateDate > '20000101'
```

(b) VariableArray 型のプロパティの要素の値を指定した検索

ここでは，myPaper クラスの VariableArray 型のプロパティである Authors プロパティの，Org プロパティが「日立製作所」であるオブジェクトの Title プロパティと Authors プロパティを取得します。

指定例

```
SELECT Title,Authors
FROM myPaper
WHERE Authors[ANY].Org = '日立製作所'
```

(7) 集合関数を指定した属性検索

ここでは，COUNT(*) 関数および COUNT 関数を使用した検索の指定例を示します。

Document X クラスの文書のうち，作成日が「2000 年 1 月 1 日」よりも新しい（CreateDate プロパティの値が「20000101」よりも大きい）文書の数を取得します。

指定例

```
SELECT COUNT(*) FROM "Document X"
WHERE CreateDate > '20000101'
```

次に，Document X クラスの文書のうち，作成日が「2000 年 1 月 1 日」よりも新しい（CreateDate プロパティの値が「20000101」よりも大きい）文書の Authors プロパティの値の数を，重複排除して取得します。

指定例

```
SELECT COUNT(DISTINCT Authors) FROM "Document X"
WHERE CreateDate > '20000101'
```

2.11.2 全文検索（HiRDB Text Search Plug-in，または HiRDB XML Extension を利用した検索）

ここでは，HiRDB Text Search Plug-in，または HiRDB XML Extension を利用した全文検索実行時の，edmSQL の指定例を示します。

なお，< 全文検索条件 > および < 概念検索条件 > の指定方法については，マニュアル「HiRDB Text Search Plug-in」，またはマニュアル「HiRDB XML Extension」を参照してください。

(1) 指定例で使用するクラスとプロパティ

この指定例で使用する文書は，次のようなクラスから作成された文書です。

文書 X

Document X クラスを基に作成された文書です。Document X クラスは，全文検索機能付き文書クラスとして作成されています。また，ユーザ定義プロパティも設定されています。Document X クラスの主なプロパティを次の表に示します。

表 2-44 Document X クラスのプロパティ

プロパティ識別子	内　容
Code	文書を管理するためのコード
Title	タイトル
Author	著者
Abstract	概要
CreateDate	作成日

2.12 留意事項

ここでは，edmSQL を使用して検索を実行する場合に，留意が必要な内容について説明します。

2.12.1 プロパティに関する制限事項

「dbrProp_」で始まるプロパティは，検索条件には指定できません。

2.12.2 検索条件に指定する値の制限事項

検索条件に設定できる値についての制限事項を示します。

なお，制限事項には，DocumentBroker としての制限のほか，データベースである HiRDB の制限に基づくものもあります。指定する値のバイト数などについては，HiRDB の制限に従います。HiRDB の制限を超えた検索を実行した場合は，データベースエラーになります。HiRDB の制限事項の詳細については，マニュアル「HiRDB　SQL リファレンス」を参照してください。

この項では，FROM 句，SELECT 句，WHERE 句および ORDER BY 句に指定する内容に関する制限事項などについて説明します。

(1) FROM 句に関する制限事項

● ロックを設定するインターフェースを使用した場合に，ロック種別に write ロック（定数：DbjDef.LOCK_WRITE）を指定した場合に，次の条件で検索を実行すると，データベースエラーになります。
- 主問い合わせで FROM 句に指定したクラスを，副問い合わせの FROM 句に指定した条件
- 主問い合わせでクラスの結合を指定した条件

(2) SELECT 句に関する制限事項

● ロックを取得するインターフェースを使用する場合に，ロックの種別に write ロック（定数：DbjDef.LOCK_WRITE）を指定して，次の条件での検索を実行すると，データベースエラーになります。
- 主問い合わせに重複排除（DISTINCT）を指定した条件
- 主問い合わせに COUNT 関数を指定した条件

● VariableArray 型のプロパティのヒット件数を COUNT 関数で取得しようとすると，edmSQL 構文解析エラーになります。

(3) WHERE 句に関する制限事項

● WHERE 句で指定する論理演算のネスト数は，255 個以内で指定してください。255 個を超えた場合は，データベースエラーになります。

(4) 検索条件として指定できる値についての制限

● 検索条件として指定するプロパティの値や，全文検索で指定できる文字については，制限があります。検索条件として指定できるクラスやプロパティについては，マニュアル「DocumentBroker Version 5　概説」の検索対象になる文書管理オブジェクトの説明を参照してください。また，edmSQL 文に指定できる値については，「2.4　字句規則」を参照してください。

(5) オペレータに指定する値についての制限

● 次のような演算を指定した場合は，データベースエラーになります。
- 比較演算の両辺に値を指定した検索は，データベースエラーになります。
  例えば，< 比較演算子 >「=」の両辺に，< リテラル > または <? パラメタ > を指定した検索などはできません。
- 右辺に値を指定する副問い合わせの <In 述語 > の左辺に，値を指定した検索はデータベースエラーになります。
  例えば，副問い合わせに使用する <In 述語 > の両辺に < 文字列リテラル > を指定した検索などはできません。
- / 演算子の右辺（除数）として，0 を指定すると，データベースエラーになります。

(6) ? パラメタに関する制限事項

<? パラメタ > に文字列を指定する場合は，32,000 バイト以内の文字列を指定してください。制限を超えて指定した場合は，データベースエラーになります。

(7) 問い合わせ指定全体に関する制限事項

一つの問い合わせ指定内で，DISTINCT を 2 回以上指定することはできません。

2.12.3 複数のクラスを対象にした検索の制限事項

複数クラスを対象にした検索での制限は，HiRDB の制限に従います。また，複数クラスを対象にした全文検索での制限は，HiRDB Text Search Plug-in または HiRDB XML Extension，および HiRDB の制限に従います。

それぞれの制限については，マニュアル「HiRDB　SQL リファレンス」，およびマニュアル「HiRDB Text Search Plug-in」またはマニュアル「HiRDB XML Extension」の，表の結合に関する規則の説明を参照してください。

2.12.4 アクセス制御機能付き検索を実行する場合の制限事項

● アクセス制御機能付き検索は，必ずアクセス制御機能を使用している文書空間で実行してください。また，アクセス制御機能に対応していないユーザアプリケーションプログラムをアクセス制御機能を使用している文書空間で使用した場合，検索結果が不正になることがあります。文書空間の設定と，ユーザアプリケーションプログラムの機能は一致させてください。

● アクセス制御機能付き検索では，検索結果の重複排除は指定できません。指定した場合は，edmSQL 構文解析エラーになります。

● 検索結果の個数を取得する検索は実行できません。SELECT 句に COUNT 関数を指定することはできません。指定した場合は，edmSQL 構文解析エラーになります。
検索結果の個数は，検索結果を全件取得することで取得してください。

● 副問い合わせは実行できません。副問い合わせを実行した場合，副問い合わせの検索結果に対してアクセス制御がされません。
例えば，次のような検索の場合，検索結果が不正になる可能性があります。
［例］

```
SELECT S0.PropA
FROM ClassA As S0
WHERE S0,PropB In
(SELECT S1.PropC FROM ClassB AS S1 WHERE S1.PropD='mojiretsu')
```

この場合，斜体で記述した副問い合わせの結果に対してはアクセス制御されません。つまり，ユーザが参照権を持たないオブジェクトも検索結果として取得できます。このため，副問い合わせの結果として取得したオブジェクトには，ユーザのアクセス権では参照できないオブジェクトが含まれている可能性があります。
また，全文検索を含む副問い合わせの検索結果には，プロパティを参照する権利を持たないオブジェクトのほか，コンテンツを参照する権利がないオブジェクトが含まれる可能性があります。
アクセス制御機能付き検索で副問い合わせに相当する検索を実行したい場合は，複数の SELECT 句に分けて，複数回 DbjDocSpace#executeSearch メソッドをコールして検索を実行してください。

- アクセス制御機能付き検索では，ユーザが SELECT 句に指定したプロパティ以外に DocumentBroker によってアクセス制御情報を表すプロパティが取得されています。このため，ユーザが SELECT 句に指定できるプロパティ数が，通常の検索で指定できる数（データベースの制限値）よりも少なくなります。

3 DocumentBroker クラスライブラリのクラス，インターフェース，およびメソッド

この章では，DocumentBroker クラスライブラリで使用するクラス，インターフェース，およびメソッドについて説明します。

3.1 ファクトリクラスのインターフェース，およびメソッド一覧

ファクトリクラスのクラス，およびメソッドを一覧形式で説明します。

3.1.1 ファクトリクラスのクラスおよびインターフェース

ファクトリクラスのクラスおよびインターフェースを次の表に示します。なお，それぞれのクラスおよびインターフェースの使用方法については，「1.6　ファクトリクラス」を参照してください。

表 3-1　ファクトリクラスのクラスおよびインターフェース一覧

クラス・インターフェース	説明
ファクトリインターフェースを取得するためのクラス	
DbjFactory0200 クラス	DocumentBroker クラスライブラリのファクトリインターフェースを取得するためのクラス
ファクトリインターフェース	
DbjFactory インターフェース	DocumentBroker クラスライブラリのオブジェクトを生成し，そのインターフェースを返却するファクトリインターフェース

3.1.2 ファクトリクラスのメソッド一覧

ファクトリクラスのクラス，およびインターフェースのメソッドの一覧を示します。

(1) DbjFactory0200 クラス（DocumentBroker クラスライブラリの DbjFactory インターフェースを取得するためのクラス）

DbjFactory0200 クラスのメソッド一覧を次の表に示します。

表 3-2　DbjFactory0200 クラスのメソッド一覧

メソッド	機能
getFactory	ファクトリインターフェースの取得
getMetaManager	メタマネージャーインターフェースの取得

(2) DbjFactory インターフェース（オブジェクトを生成するためのインターフェース）

DbjFactory インターフェースのメソッド一覧を次の表に示します。

表 3-3　DbjFactory インターフェースのメソッド一覧

メソッド	機能
DocumentBroker クラスライブラリ上のオブジェクトを作成するメソッド	
createACE	ACE の作成
createBooleanQParam	BOOL 型の？パラメタの作成
createConvertContentInfo	コンテンツ情報オブジェクトの作成
createFetchInfo	検索結果取得情報オブジェクトの作成
createInteger32QParam	INT 型の？パラメタの作成

メソッド	機能
createOIIDQParam	OIID の？パラメタの作成
createObjQParam	Object 型の？パラメタの作成
createPropSet	プロパティ値集合オブジェクトの作成
createPublicACLIdElm	パブリック ACL の OIID エレメントの作成
createReferencePathInfo	リファレンスファイル文書のパス情報オブジェクトの作成
createReferenceUploadInfo	リファレンスファイル文書のアップロード情報オブジェクトの作成
createSeedDocQParam	種文章の？パラメタの作成
createSession	DocumentBroker クラスライブラリのセッションオブジェクトの作成
createSetDCRLinkInfo	直接型リンク設定情報の作成
createSetRelLinkInfo	文書間リンク設定情報の作成
createSetRCRLinkInfo	参照型リンク設定情報の作成
createStringQParam	STR 型の？パラメタの作成
createUploadInfo	文書のアップロード情報オブジェクトの作成
createVArray	可変長配列オブジェクトの作成

3.2 パラメタクラスのインターフェース，およびメソッド一覧

パラメタクラスのインターフェース，およびメソッドを一覧形式で説明します。

3.2.1 パラメタクラスのインターフェース一覧

パラメタクラスのインターフェースの一覧を次の表に示します。なお，それぞれのインターフェースの機能については，「1.7.1　パラメタクラスのインターフェースの機能」を参照してください。

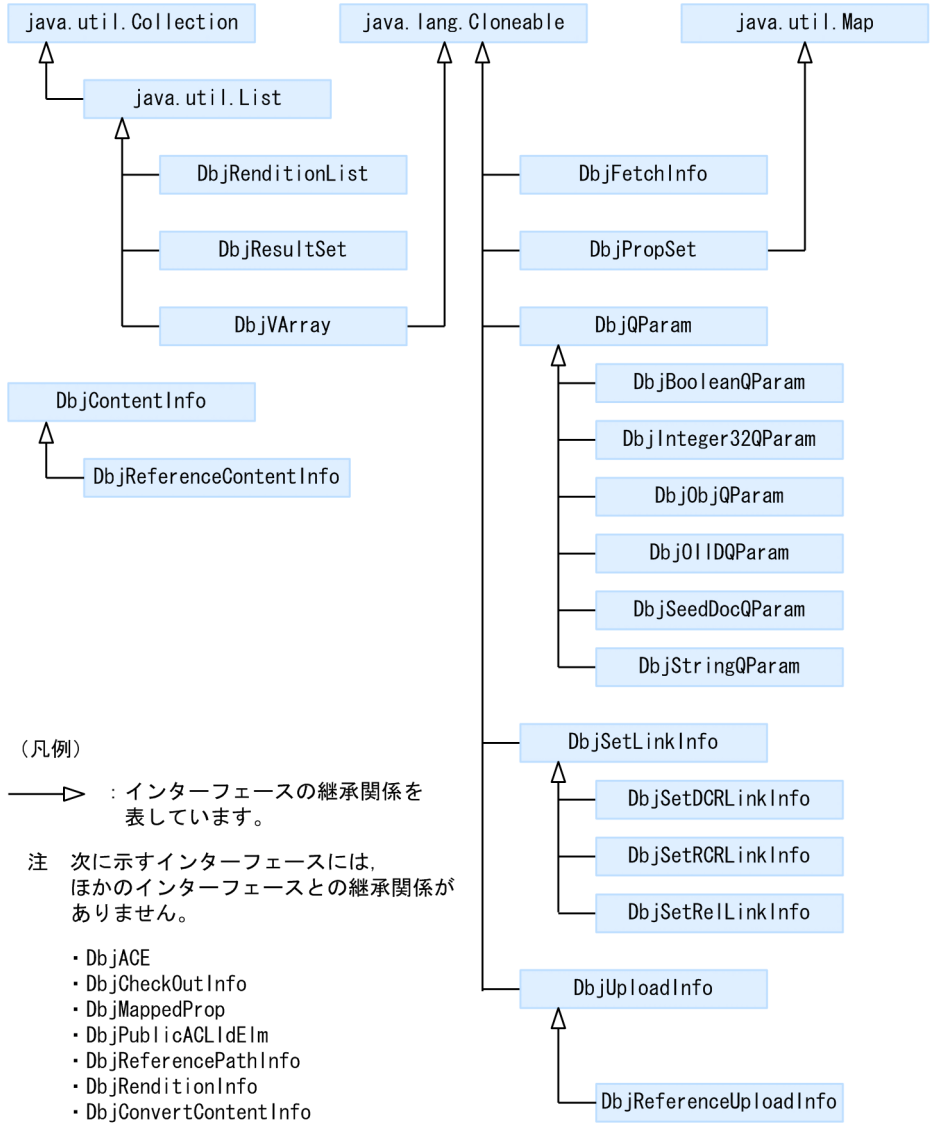
表 3-4　パラメタクラスのインターフェース一覧

インターフェース	説明
? パラメタを扱うインターフェース	
DbjBooleanQParam インターフェース	BOOL 型を表す ? パラメタを扱うインターフェース
DbjInteger32QParam インターフェース	INT 型を表す ? パラメタを扱うインターフェース
DbjObjQParam インターフェース	文書管理オブジェクトを表す ? パラメタを扱うインターフェース
DbjOIIDQParam インターフェース	OIID 文字列を表す ? パラメタを扱うインターフェース
DbjQParam インターフェース	検索メソッドで指定する ? パラメタリストの要素となるインターフェースのスーパーインターフェース
DbjSeedDocQParam インターフェース	種文章を表す ? パラメタを扱うインターフェース
DbjStringQParam インターフェース	STR 型を表す ? パラメタを扱うインターフェース
アクセス制御用プロパティを扱うインターフェース	
DbjACE インターフェース	ACE の値を扱うインターフェース
DbjPublicACLIdElm インターフェース	文書管理オブジェクトにバインドするパブリック ACL の OIID 文字列を扱うインターフェース
アップロード情報を扱うインターフェース	
DbjUploadInfo インターフェース	文書のアップロード情報を扱うインターフェース
DbjReferenceUploadInfo インターフェース	リファレンスファイル文書のアップロード情報を扱うインターフェース
可変長配列オブジェクトを扱うインターフェース	
DbjVArray インターフェース	VARRAY 型プロパティの値である可変長配列オブジェクトを扱うインターフェース
検索結果集合を扱うインターフェース	
DbjResultSet インターフェース	検索結果集合を扱うインターフェース
検索結果取得情報を扱うインターフェース	
DbjFetchInfo インターフェース	検索結果取得情報を扱うインターフェース
チェックアウト情報を扱うインターフェース	
DbjCheckOutInfo インターフェース	バージョン付きオブジェクトのチェックアウト情報を扱うインターフェース
プロパティ値集合を扱うインターフェース	
DbjPropSet インターフェース	プロパティ値集合を扱うインターフェース
文書のコンテンツ情報を扱うインターフェース	

インターフェース	説明
DbjContentInfo インターフェース	文書のコンテンツ情報を扱うインターフェース
DbjConvertContentInfo インターフェース	文書のコンテンツ種別の変換時に使用する情報を扱うインターフェース
DbjReferenceContentInfo インターフェース	リファレンスファイル文書のコンテンツ情報を扱うインターフェース
文書のパス情報を扱うインターフェース	
DbjReferencePathInfo インターフェース	リファレンスファイル文書のパス情報を扱うインターフェース
リンクの設定情報を扱うインターフェース	
DbjSetDCRLinkInfo インターフェース	直接型リンクの設定情報を扱うインターフェース
DbjSetLinkInfo インターフェース	リンクの設定情報を扱うインターフェース
DbjSetRCRLinkInfo インターフェース	参照型リンクの設定情報を扱うインターフェース
DbjSetRelLinkInfo インターフェース	文書間リンクの設定情報を扱うインターフェース
レンディション情報を扱うインターフェース	
DbjRenditionInfo インターフェース	レンディション情報を扱うインターフェース
DbjRenditionList インターフェース	レンディション情報のリストを扱うインターフェース

DocumentBroker クラスライブラリで提供しているこれらのインターフェースには継承関係があり，上位にあるインターフェースで定義したプロパティやメソッドは，下位のインターフェースに継承されます。インターフェースの継承関係を次の図に示します。

図 3-1　パラメタクラスのインターフェースの継承関係



3.2.2　パラメタクラスのメソッド一覧

パラメタクラスの各インターフェースのメソッドの一覧を示します。

（1） DbjACE インターフェース（ACE の値を扱うインターフェース）

DbjACE インターフェースのメソッド一覧を次の表に示します。

表 3-5　DbjACE インターフェースのメソッド一覧

メソッド	機能
ACE の値を取得するメソッド	
getPermission	パーミッションの取得
getSubject	サブジェクトの取得
getSubjectType	サブジェクト種別の取得
propSet	ACE のプロパティ値集合の取得

メソッド	機能
ACE の値を設定するメソッド	
setPermission	パーミッションの設定
setPropSet	ACE のプロパティ値集合の設定
setSubject	サブジェクトの設定
setSubjectType	サブジェクト種別の設定
ACE のサブジェクト種別を設定するメソッド	
setGroupSubject	サブジェクトをグループサブジェクトとして設定
setSystemSubject	サブジェクトをシステムサブジェクトとして設定
setUserSubject	サブジェクトをユーザサブジェクトとして設定

(2) DbjCheckOutInfo インターフェース（バージョン付きオブジェクトのチェックアウト情報を扱うインターフェース）

DbjCheckOutInfo インターフェースのメソッド一覧を次の表に示します。

表 3-6 DbjCheckOutInfo インターフェースのメソッド一覧

メソッド	機能
チェックアウト情報を取得するメソッド	
getCheckOutUserId	チェックアウトしたユーザのユーザ識別子の取得
getCheckOutVersionId	仮のバージョン識別子の取得
isCheckOut	チェックアウト中かどうかの判定

(3) DbjContentInfo インターフェース（文書のコンテンツ情報を扱うインターフェース）

DbjContentInfo インターフェースのメソッド一覧を次の表に示します。

表 3-7 DbjContentInfo インターフェースのメソッド一覧

メソッド	機能
文書のコンテンツ情報を取得するメソッド	
getRenditionType	コンテンツのレンディションタイプの取得
getRetrievalName	コンテンツのファイル名の取得

(4) DbjConvertContentInfo インターフェース（文書のコンテンツ種別の変換時に使用する情報を扱うインターフェース）

DbjConvertContentInfo インターフェースのメソッド一覧を次の表に示します。

表 3-8 DbjConvertContentInfo インターフェースのメソッド一覧

メソッド	機能
文書のコンテンツ種別の変換情報を取得するメソッド	
getConvertType	変換先コンテンツ種別の取得
getExecuteMode	変換元のレンディションを残すかどうかの実行モードの取得
getSourceScope	変換対象レンディション範囲の取得

メソッド	機能
getRenditionType	レンディションタイプの取得
getInvestMode	コメント付与方式の取得
getRenditionComment	レンディションコメントの取得
getReferenceTargetPath	コンテント登録先ディレクトリパスの取得
isCangeMaster	マスタレンディション変換時の変換後レンディション種別の取得
isCheckRenditionStatus	レンディションステータスのチェックの取得
isInvestSourceComment	レンディションコメントの付与時期の取得
文書のコンテンツ種別の変換情報を設定するメソッド	
setConvertType	変換先コンテンツ種別の設定
setExecuteMode	変換元のレンディションを残すかどうかの実行モードの設定
setSourceScope	変換対象レンディション範囲の設定
setRenditionType	レンディションタイプの設定
setInvestMode	コメント付与方式の設定
setRenditionComment	レンディションコメントの設定
setReferenceTargetPath	コンテント登録先ディレクトリパスの設定
setCangeMaster	マスタレンディション変換時の変換後レンディション種別の設定
setCheckRenditionStatus	レンディションステータスのチェックの設定
setInvestSourceComment	レンディションコメントの付与時期の設定

(5) DbjFetchInfo インターフェース（検索結果取得情報を扱うインターフェース）

DbjFetchInfo インターフェースのメソッド一覧を次の表に示します。

表 3-9 DbjFetchInfo インターフェースのメソッド一覧

メソッド	機能
検索結果取得情報を取得するメソッド	
getCacheKey	キャッシュキーの取得
getCacheName	キャッシュ名の取得
getCacheTotal	検索結果キャッシュの全件数の取得
getComparator	Comparator インターフェースの取得
getFetchCount	検索結果の取得件数の取得
getMaxFetchCount	検索結果の最大取得件数の取得
getStartIndex	検索結果の取得開始位置の取得
検索結果取得情報を設定するメソッド	
setCacheKey	キャッシュキーの設定
setCacheName	キャッシュ名の設定
setCacheTotal	検索結果キャッシュの全件数の設定
setComparator	Comparator インターフェースの設定
setFetchCount	検索結果の取得件数の設定
setMaxFetchCount	検索結果の最大取得件数の設定
setStartIndex	検索結果の取得開始位置の設定

(6) DbjPropSet インターフェース（プロパティ値集合を扱うインターフェース）

DbjPropSet インターフェースのメソッド一覧を次の表に示します。

表 3-10 DbjPropSet インターフェースのメソッド一覧

メソッド	機能
プロパティ値を取得するメソッド	
getIntegerVal	Integer 型でのプロパティ値の取得
getIntVal	int 型でのプロパティ値の取得
getListRef	List 型でのプロパティ値の取得
getStringVal	String 型でのプロパティ値の取得
getVArrayRef	VARRAY 型でのプロパティ値の参照の取得
getVArrayVal	VARRAY 型でのプロパティ値の取得
isNull	プロパティ値が NULL 値かどうかの判定
プロパティ値を設定するメソッド	
setNull	NULL 値プロパティの設定
setPropRef	プロパティ値の参照の設定
setPropVal	プロパティ値の設定
プロパティ名を変更するメソッド	
changePropName	プロパティ名の変更
changePropNames	プロパティ名の一括変更

(7) DbjPublicACLIdElm インターフェース（パブリック ACL の OIID 文字列を扱うインターフェース）

DbjPublicACLIdElm インターフェースのメソッド一覧を次の表に示します。

表 3-11 DbjPublicACLIdElm インターフェースのメソッド一覧

メソッド	機能
パブリック ACL の OIID 文字列を取得するメソッド	
getId	パブリック ACL の OIID 文字列の取得
propSet	パブリック ACL の OIID 文字列のプロパティ値集合の取得
パブリック ACL の OIID 文字列を設定するメソッド	
setId	パブリック ACL の OIID 文字列の設定
setPropSet	パブリック ACL の OIID 文字列のプロパティ値集合の設定

(8) DbjQParam インターフェース

DbjQParam インターフェースのメソッド一覧を次の表に示します。

表 3-12 DbjQParam インターフェースのメソッド一覧

メソッド	機能
? パラメタ値を取得するメソッド	

メソッド	機能
getVal	?パラメタ値の取得

(9) DbjReferenceContentInfo インターフェース（リファレンスファイル文書のコンテンツ情報を扱うインターフェース）

DbjReferenceContentInfo インターフェースのメソッド一覧を次の表に示します。

表 3-13 DbjReferenceContentInfo インターフェースのメソッド一覧

メソッド	機能
リファレンスファイル文書のコンテンツ情報を取得するメソッド	
getContentLocation	コンテンツロケーションの取得
getReferenceType	リファレンス種別の取得

(10) DbjReferencePathInfo インターフェース（リファレンスファイル文書のパス情報を扱うインターフェース）

DbjReferencePathInfo インターフェースのメソッド一覧を次の表に示します。

表 3-14 DbjReferencePathInfo インターフェースのメソッド一覧

メソッド	機能
リファレンスファイル文書のパス情報を取得するメソッド	
getContentOperateMode	コンテンツのパス操作モードの取得
getDeleteRootPath	削除するディレクトリのルートパスの取得
getEntry	登録するコンテンツのパスまたはダウンロード先のパスを取得
getTargetPath	コンテンツ格納先パスの取得
リファレンスファイル文書のパス情報を設定するメソッド	
setContentOperateMode	コンテンツのパス操作モードの設定
setDeleteRootPath	削除するディレクトリのルートパスの設定
setEntry	登録するコンテンツのパスまたはダウンロード先のパスの設定
setTargetPath	コンテンツ格納先パスの設定

(11) DbjReferenceUploadInfo インターフェース（リファレンスファイル文書のアップロード情報を扱うインターフェース）

DbjReferenceUploadInfo インターフェースのメソッド一覧を次の表に示します。

表 3-15 DbjReferenceUploadInfo インターフェースのメソッド一覧

メソッド	機能
リファレンスファイル文書のアップロード情報を取得するメソッド	
getReferencePathInfo	リファレンスファイル文書のパス情報の取得
リファレンスファイル文書のアップロード情報を設定するメソッド	
setReferencePathInfo	リファレンスファイル文書のパス情報の設定

(12) DbjRenditionInfo インターフェース（レンディション情報を扱うインターフェース）

DbjRenditionInfo インターフェースのメソッド一覧を次の表に示します。

表 3-16 DbjRenditionInfo インターフェースのメソッド一覧

メソッド	機能
レンディションタイプを取得するメソッド	
getRenditionType	レンディションタイプの取得
propSet	レンディションタイプのプロパティ値集合の取得

(13) DbjRenditionList インターフェース（レンディション情報のリストを扱うインターフェース）

DbjRenditionList インターフェースのメソッド一覧を次の表に示します。

表 3-17 DbjRenditionList インターフェースのメソッド一覧

メソッド	機能
レンディションタイプのリストを取得するメソッド	
getRenditionInfo	レンディション情報の取得
getRenditionTypeList	レンディションタイプのリストの取得

(14) DbjResultSet インターフェース（検索結果集合を扱うインターフェース）

DbjResultSet インターフェースのメソッド一覧を次の表に示します。

表 3-18 DbjResultSet インターフェースのメソッド一覧

メソッド	機能
カーソルを操作するメソッド	
absolute	カーソルの絶対指定行への移動
afterLast	カーソルの最終行の後ろへの移動
beforeFirst	カーソルの先頭行の前への移動
first	カーソルの先頭行への移動
getRow	カーソル行インデクスの取得
isAfterLast	カーソルが最終行の後ろかどうかの判定
isBeforeFirst	カーソルが先頭行の前かどうかの判定
isFirst	カーソルが先頭行かどうかの判定
isLast	カーソルが最終行かどうかの判定
last	カーソルの最終行への移動
next	カーソルの次の行への移動
previous	カーソルの前の行への移動
検索結果集合のメタ情報を扱うメソッド	
findColumnName	列インデクスの取得
getColumnMetaName	列名の取得（表名を含む）
getColumnName	列名の取得

メソッド	機能
getColumnType	列のデータ型の取得
getTableName	表名の取得
isNamed	名前付き検索結果かどうかの判定
setColumnMetaName	列名の設定
検索結果集合のサイズを扱うメソッド	
getColumnCount	列数の取得
getRowCount	行数の取得
指定した行または列のデータを取得するメソッド	
getColumnVals	列データの取得
getIntegerVal	Integer 型での列データの取得
getIntVal	int 型での列データの取得
getObjectRef	データの参照の取得
getObjectVal	データの取得
getPropSet	行データをプロパティ値集合として取得
getRowVals	行データの取得
getStringVal	String 型での列データの取得
getVArrayRef	VARRAY 型でのデータの参照の取得
getVArrayVal	VARRAY 型でのデータの取得
isNull	NULL 値かどうかの判定
列を削除するメソッド	
distinct	列の重複の排除
reduct	名前なし列の削除

（15）DbjSetLinkInfo インターフェース（リンクの設定情報を扱うインターフェース）

DbjSetLinkInfo インターフェースのメソッド一覧を次の表に示します。

表 3-19　DbjSetLinkInfo インターフェースのメソッド一覧

メソッド	機能
リンク設定情報を取得するメソッド	
getLinkType	リンク種別の取得
getTargetObj	リンク先オブジェクトの取得
propSet	リンクオブジェクトのプロパティ値集合の取得
リンク設定情報を設定するメソッド	
setPropSet	リンクオブジェクトのプロパティ値集合の設定
setTargetObj	リンク先オブジェクトの設定

（16）DbjUploadInfo インターフェース（文書のアップロード情報を扱うインターフェース）

DbjUploadInfo インターフェースのメソッド一覧を次の表に示します。

表 3-20 DbjUploadInfo インターフェースのメソッド一覧

メソッド	機能
文書のアップロード情報を取得するメソッド	
getFilePath	登録する文書のフルパスの取得
getIndexPath	全文検索インデクス作成用ファイルのフルパスの取得
getRenditionType	登録する文書のレンディションタイプの取得
getRetrievalName	登録する文書のファイル名の取得
renditionPropSet	レンディションプロパティ値集合の取得
文書のアップロード情報を設定するメソッド	
setFilePath	登録する文書のフルパスの設定
setIndexPath	全文検索インデクス作成用ファイルのフルパスの設定
setRenditionPropSet	レンディションプロパティ値集合の設定
setRenditionType	登録する文書のレンディションタイプの設定
setRetrievalName	登録する文書のファイル名の設定

（17）DbjVArray インターフェース（可変長配列オブジェクトを扱うインターフェース）

DbjVArray インターフェースのメソッド一覧を次の表に示します。

表 3-21 DbjVArray インターフェースのメソッド一覧

メソッド	機能
可変長配列オブジェクトのデータを取得するメソッド	
getPropCount	メタプロパティ数の取得
getPropNameSet	メタプロパティの取得
getPropVals	可変長配列オブジェクトのプロパティ値のリストの取得
propSet	プロパティ値集合の取得
可変長配列オブジェクトのデータを設定するメソッド	
addPropSet	プロパティ値集合（行）の追加
addPropVals	プロパティ値（列）の追加
removeProp	プロパティの削除
可変長配列オブジェクトのプロパティ名を変更するメソッド	
changePropName	可変長配列オブジェクトのプロパティ名の変更
changePropNames	可変長配列オブジェクトのプロパティ名の一括変更

3.3 文書管理クラスのインターフェース，およびメソッド一覧

文書管理クラスのインターフェース，およびメソッドを一覧形式で説明します。

3.3.1 文書管理クラスのインターフェース一覧

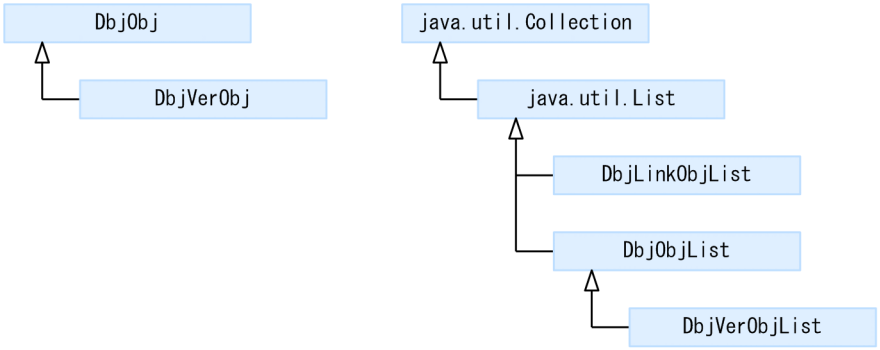
文書管理クラスのインターフェースの一覧を次の表に示します。なお，それぞれのインターフェースの使用方法については，表の参照先欄に示す個所を参照してください。

表 3-22 文書管理クラスのインターフェース一覧

インターフェース	説明	参照先
文書管理セッションを扱うインターフェース		
DbjSession インターフェース	DocumentBroker クラスライブラリのセッションを管理するインターフェース	1.8 セッションとトランザクションの制御
文書空間操作を扱うインターフェース		
DbjDocSpace インターフェース	文書空間へのアクセスを扱うインターフェース	1.9 文書空間へのアクセス
文書管理オブジェクト操作を扱うインターフェース		
DbjObj インターフェース	一つの文書管理オブジェクトへのアクセスを扱うインターフェース	1.10 文書管理オブジェクトの操作
DbjVerObj インターフェース	一つの文書管理オブジェクトにアクセスするインターフェースに加えて，バージョンオブジェクトのバージョン識別子を扱うインターフェース	1.10 文書管理オブジェクトの操作
複数の文書管理オブジェクト操作を扱うインターフェース		
DbjObjList インターフェース	複数の文書管理オブジェクトに一括してアクセスするインターフェース	1.10 文書管理オブジェクトの操作
DbjVerObjList インターフェース	複数の文書管理オブジェクトに一括してアクセスするインターフェースに加えて，リストの要素すべてのバージョン識別子を扱うインターフェース	1.10 文書管理オブジェクトの操作
リンクオブジェクト操作を扱うインターフェース		
DbjLinkObj インターフェース	一つのリンクオブジェクトにアクセスするインターフェース	1.10 文書管理オブジェクトの操作
複数のリンクオブジェクト操作を扱うインターフェース		
DbjLinkObjList インターフェース	複数のリンクオブジェクトに一括してアクセスするインターフェース	1.10 文書管理オブジェクトの操作

DocumentBroker クラスライブラリで提供しているこれらのインターフェースには継承関係があり，上位にあるインターフェースで定義したメソッドは，下位のインターフェースに継承されます。インターフェースの継承関係を次の図に示します。

図 3-2　文書管理クラスのインターフェースの継承関係



（凡例）

：インターフェースの継承関係を表しています。

注　次に示すインターフェースには，ほかのインターフェースとの継承関係がありません。

- DbjDocSpace
- DbjFactory
- DbjLinkObj
- DbjSession

3.3.2　文書管理クラスのメソッド一覧

文書管理クラスの各インターフェースのメソッドの一覧を示します。

(1) DbjDocSpace インターフェース（文書空間へのアクセスを扱うインターフェース）

DbjDocSpace インターフェースのメソッド一覧を次の表に示します。

表 3-23　DbjDocSpace インターフェースのメソッド一覧

メソッド	機能
Proxy オブジェクトを生成するメソッド	
createLinkObjList	複数のリンクオブジェクトアクセスインターフェースの取得
createObjConnection	文書管理オブジェクトアクセスインターフェースの取得
createObjList	複数の文書管理オブジェクトアクセスインターフェースの取得
アクセス制御モードを取得するメソッド	
isAccessControlMode	アクセス制御モードの取得
オブジェクトを削除するメソッド	
removeObjects	検索条件を指定したオブジェクトの削除
検索を扱うメソッド	
changeSearchACLMode	検索実行時のアクセス制御モードの変更
executeSearch	検索の実行
getSearchACLMode	検索実行時のアクセス制御モードの取得
メタ情報を取得するメソッド	
getMeta	文書空間のメタ情報の取得

メソッド	機能
文書管理オブジェクトを生成するメソッド	
createDocument	バージョンなし文書の作成
createFolder	バージョンなしフォルダの作成
createIndependentData	独立データの作成
createPublicACL	パブリック ACL の作成
createVrDocument	バージョン付き文書の作成

(2) DbjLinkObj インターフェース（一つのリンクオブジェクトにアクセスするインターフェース）

DbjLinkObj インターフェースのメソッド一覧を次の表に示します。

表 3-24 DbjLinkObj インターフェースのメソッド一覧

メソッド	機能
リンク Proxy オブジェクトのプロパティを取得するメソッド	
getLinkId	リンク識別子の取得
getLinkType	リンク種別の取得
getOwnerObj	リンク元オブジェクトの取得
getTargetObj	リンク先オブジェクトの取得
リンク Proxy オブジェクトのプロパティ値集合を取得または設定するメソッド	
propSet	リンク Proxy オブジェクトのプロパティ値集合インターフェースの取得
setPropSet	リンク Proxy オブジェクトのプロパティ値集合の設定
リンクオブジェクトのプロパティを取得または設定するメソッド	
readProperties	リンクプロパティの取得
writeProperties	リンクプロパティ値の設定
リンクオブジェクトを削除するメソッド	
removeObject	リンクオブジェクトの削除

(3) DbjLinkObjList インターフェース（DbjLinkObj インターフェースを要素とするリストインターフェースを扱うインターフェース）

DbjLinkObjList インターフェースのメソッド一覧を次の表に示します。

表 3-25 DbjLinkObjList インターフェースのメソッド一覧

メソッド	機能
リンク Proxy オブジェクトのプロパティを一括して取得するメソッド	
getLinkIdList	ターゲットリンク識別子プロパティのリストの取得
getLinkObj	リスト要素の DbjLinkObj インターフェースの取得
getOwnerObjList	リンク元オブジェクトのリストの取得
getTargetObjList	リンク先オブジェクトのリストの取得
リンクオブジェクトのプロパティを一括して取得または設定するメソッド	

メソッド	機能
readProperties	リンクプロパティ値の一括取得
writeProperties	リンクプロパティ値の一括設定
リンクオブジェクトを一括して削除するメソッド	
removeObjects	リンクオブジェクトの一括削除

(4) DbjObj インターフェース（一つの文書管理オブジェクトのアクセスを扱うインターフェース）

DbjObj インターフェースのメソッド一覧を次の表に示します。

表 3-26 DbjObj インターフェースのメソッド一覧

メソッド	機能
アクセス対象のターゲットバージョンを変更するメソッド	
getTargetVersion	アクセス対象ターゲットバージョンのバージョン識別子の取得
setTargetVersion	アクセス対象ターゲットバージョンのバージョン識別子の変更
文書管理オブジェクトのプロパティを取得するメソッド	
getClassName	アクセス対象文書管理オブジェクトを構成する DocumentBroker クラス名の取得
getLockType	Proxy オブジェクトのアクセスロック種別の取得
getObjType	アクセス対象文書管理オブジェクトのオブジェクト種別の取得
getOiid	アクセス対象文書管理オブジェクトの OIID の取得
コンテンツにアクセスするメソッド	
downloadContents	文書のコンテンツのダウンロード
uploadContents	文書のコンテンツのアップロード
バージョンを扱うメソッド	
cancelCheckOut	バージョン付きオブジェクトのチェックアウトの取り消し
checkIn	バージョン付きオブジェクトのチェックイン
checkOut	バージョン付きオブジェクトのチェックアウト
deleteVersion	バージョンの削除
getCheckOutStatus	バージョン付きオブジェクトのチェックアウト状態の取得
getVersionId	バージョンオブジェクトのバージョン識別子の取得
getVersioningInfo	バージョン付きオブジェクトのバージョニングオブジェクトの取得
getVersionObjList	バージョン付きオブジェクトのバージョン一覧の取得
パブリック ACL を扱うメソッド	
bindPublicACL	パブリック ACL のバインド
getBindObjectList	パブリック ACL にバインドしている文書管理オブジェクト一覧の取得
getPublicACLList	バインドしているパブリック ACL 一覧の取得
unbindPublicACL	パブリック ACL のアンバインド

メソッド	機能
プロパティ値集合を取得または設定するメソッド	
propSet	Proxy オブジェクトのプロパティ値集合インターフェースの取得
setPropSet	Proxy オブジェクトのプロパティ値集合の設定
プロパティ値を取得または設定するメソッド	
readProperties	文書管理オブジェクトのプロパティ値の読み込み
writeProperties	文書管理オブジェクトのプロパティ値の設定
文書管理オブジェクトを削除するメソッド	
removeObject	文書管理オブジェクトの削除
リンクをたどって文書管理オブジェクトを取得するメソッド	
getChildList	フォルダのリンク先オブジェクト（下位オブジェクト）の取得
getDCRParent	直接型リンクによるリンク元オブジェクトの取得
getParentList	フォルダのリンク元オブジェクトの取得
getRelList	文書間リンク一覧の取得
リンクおよびリンクを解除するメソッド	
link	リンク先オブジェクトとのリンク
move	直接型リンクが設定されている文書管理オブジェクトの移動
unlink	リンク先オブジェクトとのリンクの解除
unlinkByLinkId	リンク識別子の指定によるリンク先オブジェクトとのリンクの解除
レンディションを扱うメソッド	
addRendition	レンディションの追加
changeMasterRendition	マスタレンディションの変更
deleteRendition	レンディションの削除
getRenditionList	レンディション情報一覧の取得
writeRenditionProperties	レンディションプロパティの設定
コンテンツ種別を変更するメソッド	
convertContentType	コンテンツ種別の変換
アクセスロックが異なる文書管理オブジェクトインターフェースを取得するメソッド	
lock	アクセスロック種別の異なる文書管理オブジェクトインターフェースの取得

（5） DbjObjList インターフェース（複数の文書管理オブジェクトに一括してアクセスするためのインターフェース）

DbjObjList インターフェースのメソッド一覧を次の表に示します。

表 3-27　DbjObjList インターフェースのメソッド一覧

メソッド	機能
オブジェクトを一括して移動するメソッド	
move	直接型リンクが設定されている文書管理オブジェクトの一括移動

メソッド	機能
プロパティ値集合を一括して設定するメソッド	
setPropSet	要素のプロパティ値集合の設定
プロパティ値を一括して取得または設定するメソッド	
readProperties	複数の文書管理オブジェクトプロパティ値の一括取得
writeProperties	複数の文書管理オブジェクトプロパティ値の一括設定
文書管理オブジェクトを一括して削除するメソッド	
removeObjects	文書管理オブジェクトの一括削除
要素を一括して取得するメソッド	
getObj	リスト要素の取得
ロックを一括して指定するメソッド	
lock	アクセスロック種別の異なる複数文書管理オブジェクトインターフェースの取得

(6) DbjSession インターフェース（DocumentBroker のセッションを管理するインターフェース）

DbjSession インターフェースのメソッド一覧を次の表に示します。

表 3-28 DbjSession インターフェースのメソッド一覧

メソッド	機能
セッションを扱うメソッド	
checkSession	セッションが有効かどうかのチェック
getLoginUserInfo	ユーザ情報の取得
login	ログイン
logout	ログアウト
トランザクションを扱うメソッド	
begin	トランザクションの開始
commit	トランザクションの確定
rollback	トランザクションの取り消し
コンテンツ格納先ベースパスを扱うメソッド	
getReferencePath	コンテンツ格納先ベースパスの取得
setReferencePath	コンテンツ格納先ベースパスの設定

(7) DbjVerObj インターフェース（一つの文書管理オブジェクトにアクセスするインターフェースに加え，バージョンオブジェクトのバージョン識別子を扱うインターフェース）

DbjVerObj インターフェースのメソッド一覧を次の表に示します。

表 3-29 DbjVerObj インターフェースのメソッド一覧

メソッド	機能
バージョンオブジェクトのバージョン識別子を取得するメソッド	
getVersionId	バージョンオブジェクトのバージョン識別子の取得

（8） DbjVerObjList インターフェース（DbjVerObj インターフェースを要素とするリストインターフェースを扱うインターフェース）

DbjVerObjList インターフェースのメソッド一覧を次の表に示します。

表 3-30　DbjVerObjList インターフェースのメソッド一覧

メソッド	機能
要素のバージョンオブジェクトのバージョン識別子を一括して取得するメソッド	
getVerObj	要素の DbjVerObj インターフェースの取得
getVersionIdList	オブジェクトのバージョン識別子リストの取得

3.4 メタクラスのインターフェース，およびメソッド一覧

メタクラスのインターフェース，およびメソッドを一覧形式で説明します。

3.4.1 メタクラスのインターフェース一覧

メタクラスのインターフェースの一覧を次の表に示します。なお，それぞれのインターフェースの使用方法については，「1.12　メタ情報の取得」を参照してください。

表 3-31　メタクラスのインターフェース一覧

インターフェース	説明
DbjClassDesc インターフェース	一つのクラスのクラスディスクリプションを扱うインターフェース
DbjMeta インターフェース	一つの文書空間のメタ情報を扱うインターフェース
DbjMetaManager インターフェース	文書空間のメタ情報を管理するインターフェース
DbjPropDesc インターフェース	一つのプロパティのプロパティディスクリプションを扱うインターフェース

なお，DocumentBroker で提供しているメタクラスのインターフェースには，継承関係がありません。

3.4.2 メタクラスのメソッド一覧

メタクラスの各インターフェースでのメソッドの一覧を示します。

(1) DbjClassDesc インターフェース（クラスディスクリプションを扱うインターフェース）

DbjClassDesc インターフェースのメソッド一覧を次の表に示します。

表 3-32　DbjClassDesc インターフェースのメソッド一覧

メソッド	機能
getName	クラス名の取得
getProperties	プロパティディスクリプションの取得
getSubClasses	サブクラスのクラスディスクリプションの取得
getSuperClass	スーパークラスのクラスディスクリプションの取得

(2) DbjMeta インターフェース（文書空間のメタ情報を扱うインターフェース）

DbjMeta インターフェースのメソッド一覧を次の表に示します。

表 3-33　DbjMeta インターフェースのメソッド一覧

メソッド	機能
getClassDesc	指定したクラスのクラスディスクリプションの取得
getDocSpaceId	文書空間識別子の取得
getExtFromRenditionType	レンディションタイプに対応する拡張子の取得
getPropDataType	指定したプロパティのデータ型の取得

メソッド	機能
getPropDesc	指定したプロパティのプロパティディスクリプションの取得
getRenditionType	拡張子に対応するレンディションタイプの取得

(3) DbjMetaManager インターフェース（文書空間のメタ情報を管理するインターフェース）

DbjMetaManager インターフェースのメソッド一覧を次の表に示します。

表 3-34 DbjMetaManager インターフェースのメソッド一覧

メソッド	機能
getMeta	メタ情報の取得

(4) DbjPropDesc インターフェース（プロパティディスクリプションを扱うインターフェース）

DbjPropDesc インターフェースのメソッド一覧を次の表に示します。

表 3-35 DbjPropDesc インターフェースのメソッド一覧

メソッド	機能
getDataType	プロパティのデータ型の取得
getName	プロパティ名の取得
getVArrayClass	VARRAY 型プロパティを扱うクラスのクラスディスクリプションの取得

3.5 例外クラスのクラス一覧，スーパークラス，およびコンストラクタ

例外クラスのクラスの一覧，スーパークラス，およびコンストラクタについて説明します。

3.5.1 例外クラスのクラス一覧

例外クラスのクラス一覧を次の表に示します。なお，例外処理の方法については，「1.13　例外処理」を参照してください。

表 3-36　例外クラスのクラス一覧

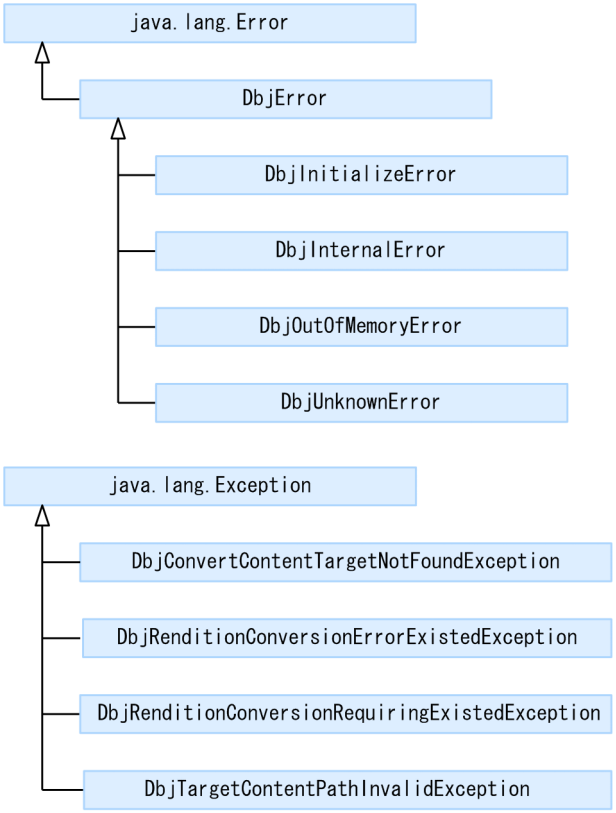
クラス	説明
DbjAccessControlException クラス	ユーザがメソッドを実行する際にその操作に対してアクセス権がなかった場合のエラーを表す例外クラス
DbjAccessControlNotSupportedException クラス	文書空間がアクセス制御機能対応していない場合のエラーを表す例外クラス
DbjACEOperationException クラス	ACE の操作エラーを表すスーパークラス。特定の ACE 操作のエラーはサブクラスがスローされる例外クラス
DbjACLOutOfRangeException クラス	文書管理オブジェクトの ACE の個数が制限値を超えた場合のエラーを表す例外クラス
DbjAlreadyCheckOutException クラス	すでにチェックアウトされている場合のエラーを表す例外クラス
DbjCheckOutException クラス	チェックアウトについてのエラーを表す例外クラスのスーパークラス
DbjContentNotRegisteredException クラス	リファレンスファイル管理機能を使用したオブジェクトをローカルファイルにダウンロードする場合に，オブジェクトにコンテンツが登録されていないときのエラーを表す例外クラス
DbjContentTypeMismatchException クラス	リファレンスファイル管理機能で，オブジェクトのコンテンツ種別とメソッドが要求するコンテンツ種別が一致しない場合のエラーを表す例外クラス
DbjConvertContentTargetNotFoundException クラス	変換対象のコンテンツが存在しない場合のエラーを表す例外クラス
DbjDBDeadLockException クラス	DB のデッドロックエラーを表す例外クラス
DbjDBException クラス	DB エラーを表す例外クラス
DbjDBLockTimeoutException クラス	DB のロックタイムアウトエラーを表す例外クラス
DbjDisconnectedSessionException クラス	文書空間との接続が切断された場合のエラーを表す例外クラス
DbjError クラス	java.lang.Error クラスを継承し，DocumentBroker クラスライブラリ固有の致命的エラーを表す例外クラスのスーパークラス
DbjException クラス	java.lang.Exception クラスを継承し，DocumentBroker クラスライブラリ固有のエラーを表す例外クラスのスーパークラス
DbjFileAccessException クラス	ファイルのアクセス権がない場合のエラーを表す例外クラス
DbjFileNotFoundException クラス	指定ファイルが存在しない場合のエラーを表す例外クラス

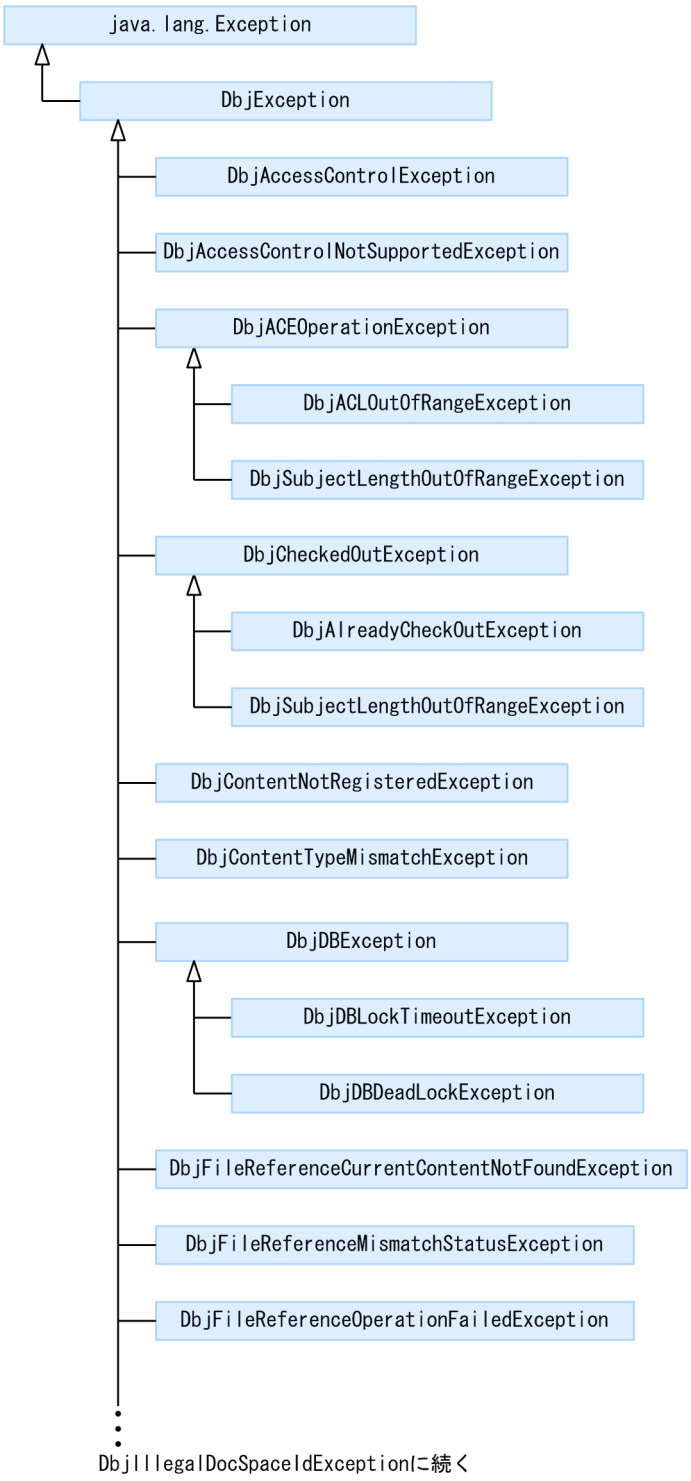
クラス	説明
DbjFileReferenceCurrentContentNotfoundException クラス	リファレンスファイル管理機能を使用したバージョン付き文書オブジェクトで，カレントバージョンにコンテンツが存在しない場合のエラーを表す例外クラス
DbjFileReferenceMismatchStatusException クラス	リファレンスファイル管理機能を使用するサーバでのコンテンツ操作でエラーが発生し，オブジェクトとコンテンツが不整合な状態になった場合のエラーを表す例外クラス
DbjFileReferenceOperationFailedException クラス	リファレンスファイル管理機能を使用するサーバでのコンテンツ操作が失敗した場合のエラーを表す例外クラス
DbjIllegalDocSpaceIdException クラス	不正な文書空間識別子が指定された場合のエラーを表す例外クラス
DbjIllegalObjectTypeException クラス	メソッドによる操作に不適切な文書管理オブジェクトの種別が指定された場合のエラーを表す例外クラス
DbjIllegalPropValException クラス	指定プロパティ値の不正エラーを表す例外クラス
DbjInitializeError クラス	DocumentBroker クラスライブラリ初期化時のエラーを表す例外クラス
DbjInternalError クラス	DocumentBroker クラスライブラリの内部エラーを表す例外クラス
DbjIOException クラス	DocumentBroker クラスライブラリの IO エラーを表す例外クラスのスーパークラス
DbjIsMasterRenditionException クラス	マスタレンディションが指定された場合のエラーを表す例外クラス
DbjLastVersionException クラス	操作対象となるバージョン付きオブジェクトの最後の一つのバージョンを削除しようとした場合のエラーを表す例外クラス
DbjMasterRenditionNotSetException クラス	マスタレンディションのレンディションタイプが設定されていない場合のエラーを表す例外クラス
DbjNotAuthenticatedException クラス	ユーザ認証エラーを表す例外クラス
DbjNotCheckOutException クラス	チェックアウトされていない場合のエラーを表す例外クラス
DbjNotLoginException クラス	文書空間と接続（ログイン）されていない場合のエラーを表す例外クラス
DbjObjectNotFoundException クラス	操作対象となる文書管理オブジェクトが，メソッド実行時にすでに削除されていた場合，または存在しない場合のエラーを表す例外クラス
DbjOutOfMemoryError クラス	メモリ不足が発生したことを表す例外クラス
DbjPublicACLAlreadyBoundException クラス	指定したパブリック ACL がすでにバインドされている場合のエラーを表す例外クラス
DbjPublicACLNotBoundException クラス	指定したパブリック ACL がバインドされていない場合のエラーを表す例外クラス
DbjPublicACLNotFoundException クラス	指定したパブリック ACL が存在しない場合のエラーを表す例外クラス
DbjPublicACLOperationException クラス	パブリック ACL の操作エラーを表すスーパークラス。特定のパブリック ACL 操作のエラーにはサブクラスがスローされる例外クラス
DbjPublicACLOutOfRangeException クラス	オブジェクトに追加できるパブリック ACL の数が制限値を超えた場合のエラーを表す例外クラス

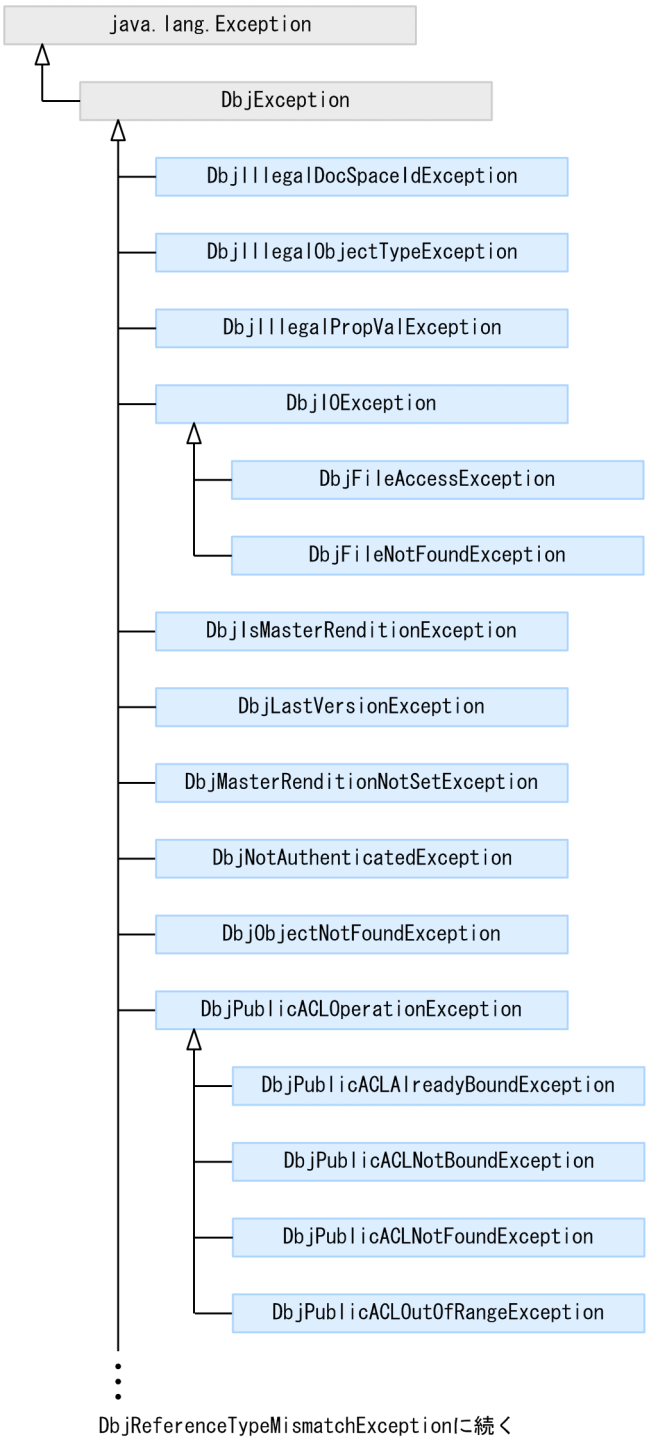
クラス	説明
DbjReferenceTypeMismatchException クラス	リファレンスファイル管理機能で，オブジェクトのリファレンス種別とメソッドが要求するリファレンス種別が一致しない場合のエラーを表す例外クラス
DbjRenditionConversionErrorExistedException クラス	変換エラーのレンディションが存在した場合のエラーを表す例外クラス
DbjRenditionConversionRequiringExistedException クラス	変換要求中のレイディションが存在した場合のエラーを表す例外クラス
DbjRenditionCountOutOfRangeException クラス	レンディション数が制限値を超えた場合のエラーを表す例外クラス
DbjRenditionIsEmptyException クラス	レンディションが空の場合のエラーを表す例外クラス
DbjRenditionNotConvertedException クラス	レンディションが更新処理されていない場合のエラーを表す例外クラス
DbjRenditionNotFoundException クラス	指定レンディションが存在しない場合のエラーを表す例外クラス
DbjRenditionTypeDuplicatedException クラス	レンディションタイプが重複している場合のエラーを表す例外クラス
DbjSessionException クラス	セッションについてのエラーを表す例外クラスのスーパークラス
DbjSessionNotConnectException クラス	文書空間との接続が行われていない場合のエラーを表す例外クラスのスーパークラス
DbjSubjectLengthOutOfRangeException クラス	ACE のサブジェクト長が制限値を超えた場合のエラーを表す例外クラス
DbjTargetContentPathNotSetException クラス	リファレンスファイル管理機能で，コンテンツ格納先ベースパスが設定されていない場合のエラーを表す例外クラス
DbjTargetContentPathInvalidException クラス	リファレンスファイル管理機能でベースパスが不正な場合のエラーを表す例外クラス
DbjUnknownError クラス	DocumentBroker クラスライブラリ内での未知のエラーの発生を表す例外クラス
DbjNotSupportedException クラス	サポートされていない操作を表す例外クラス
DbjVersionObjectNotFoundException クラス	バージョンオブジェクトが存在しない場合の例外クラス
DbjUnexpectedException クラス	内部矛盾が発生した場合の例外クラス

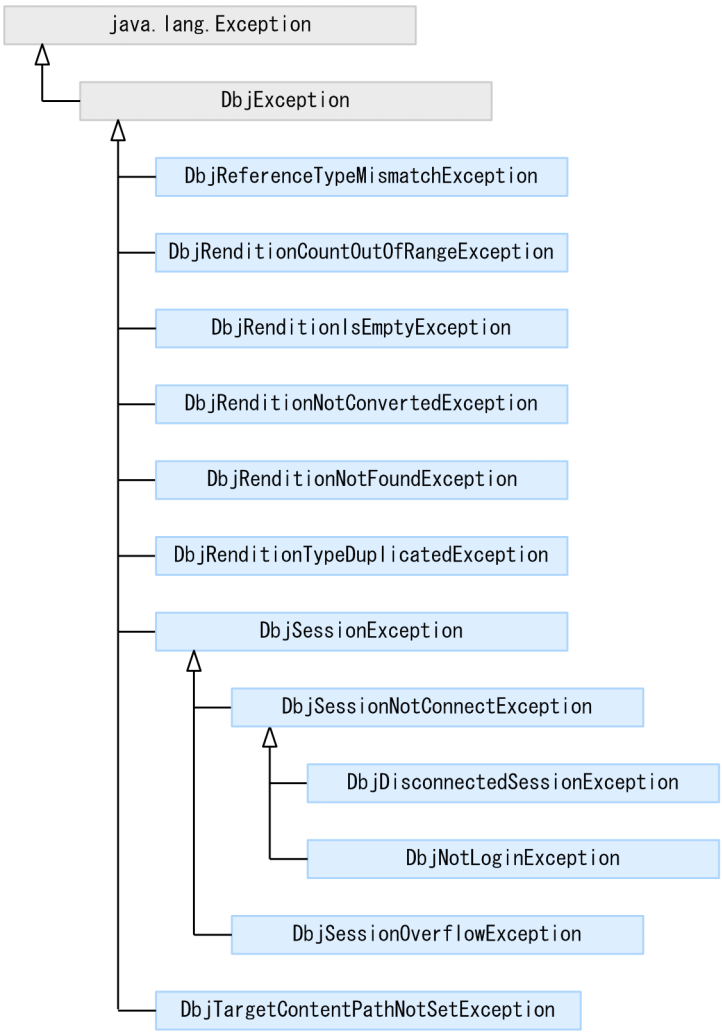
DocumentBroker クラスライブラリで提供しているこれらのクラスには継承関係があり，上位にあるクラスで定義したメソッドは，下位のクラスに継承されます。クラスの継承関係を次の図に示します。

図 3-3　例外クラスのインターフェースの継承関係




java.lang.Exception
DbjException
DbjAccessControlException
DbjAccessControlNotSupportedException
DbjACEOperationException
DbjACLOutOfRangeException
DbjSubjectLengthOutOfRangeException
DbjCheckedOutException
DbjAlreadyCheckOutException
DbjSubjectLengthOutOfRangeException
DbjContentNotRegisteredException
DbjContentTypeMismatchException
DbjDBException
DbjDBLockTimeoutException
DbjDBDeadLockException
DbjFileReferenceCurrentContentNotFoundException
DbjFileReferenceMismatchStatusException
DbjFileReferenceOperationFailedException
DbjIllegalDocSpaceIdExceptionに続く


java.lang.Exception
DbjException
DbjIllegalDocSpaceIdException
DbjIllegalObjectTypeException
DbjIllegalPropValException
DbjIOException
DbjFileAccessException
DbjFileNotFoundException
DbjIsMasterRenditionException
DbjLastVersionException
DbjMasterRenditionNotSetException
DbjNotAuthenticatedException
DbjObjectNotFoundException
DbjPublicACLOperationException
DbjPublicACLAlreadyBoundException
DbjPublicACLNotBoundException
DbjPublicACLNotFoundException
DbjPublicACLOutOfRangeException
DbjReferenceTypeMismatchExceptionに続く



3.5.2　例外クラスのスーパークラス

DocumentBroker クラスライブラリの例外クラスのスーパークラスについて次に説明します。

(1) DbjError クラス

DbjError クラスは java.lang.Error クラスを継承し，DocumentBroker クラスライブラリ固有の致命的なエラーを表すスーパークラスです。メモリ不足，DocumentBroker クラスライブラリの初期化エラー，DocumentBroker クラスライブラリ内で予期しない内部エラーが発生した時のエラーなど，ユーザアプリケーションプログラムで対処できないエラーを表します。

(2) DbjException クラス

DbjException クラスは java.lang.Exception クラスを継承し，DocumentBroker クラスライブラリ固有の

エラーを表すスーパークラスです。

3.5.3 例外クラスのコンストラクタ

DbjError クラスおよび DbjException クラスは，二つの public 属性のコンストラクタを保持します。一つは，引数を取らないコンストラクタで，もう一つは，String 型の引数を取るコンストラクタです。引数を取らないコンストラクタは，エラーメッセージに null が関連付けられた例外クラスのオブジェクトを生成します。また，String 型の引数を取るコンストラクタは，引数に指定されたエラーメッセージが関連付けられた例外クラスのオブジェクトを生成します。

3.6 定数定義クラスのカテゴリ一覧

定数定義クラスのカテゴリを次の表に示します。

表 3-37 定数定義クラスのカテゴリ一覧

カテゴリ	説明
DbjDef クラス	
ACL	検索実行時のアクセス制御モードを表す定数
DATATYPE	プロパティのデータ型を表す定数
DMA BOOLEAN	BOOL 型を表す定数
INDEXTYPE	全文検索インデクスの種別を表す定数
LINK	リンク種別を表す定数
LOCK	ロック種別を表す定数
OBJTYPE	オブジェクト種別を表す定数
OPERATEMODE	コンテンツのパス操作モードを表す定数
ORDER	バージョン情報の取得順序を表す定数
PERM	パーミッションを表す定数
PRIV	ユーザの特権を表す定数
REFERENCETYPE	リファレンス種別を表す定数
RELATIONEND	文書間リンク設定情報一覧の取得条件を表す定数
RENDSTATUS	レンディションステータスを表す定数
SUBJECTTYPE	サブジェクト種別を表す定数
SYSSUBJECT	システムサブジェクトを表す定数
CONTENTTYPE	コンテンツ種別を表す定数
Others	そのほか（上記以外）の内容を表す定数
DbjTraceDef クラス	
TRACELEVEL	トレースの出力レベルを表す定数
TRACEOUTPUT	トレースの出力先を表す定数

3.7 ライブラリ情報取得クラスのクラス，およびメソッド一覧

ライブラリ情報取得クラスのクラス，およびメソッドを一覧形式で説明します。

3.7.1 ライブラリ情報取得クラスのクラス

ライブラリ情報取得クラスのクラスを次の表に示します。なお，クラスの使用方法については，「1.14 ライブラリ情報の取得」を参照してください。

表 3-38 ライブラリ情報取得クラスのクラス

クラス	説明
ライブラリ情報を扱うクラス	
DbjLibInfo クラス	DocumentBroker のバージョン情報を扱うクラス

3.7.2 ライブラリ情報取得クラスのメソッド一覧

ライブラリ情報取得クラスの DbjLibInfo クラス（DocumentBroker のバージョン情報を扱うクラス）のメソッドを次の表に示します。

表 3-39 DbjLibInfo クラスのメソッド

メソッド	機能
バージョンを取得するメソッド	
getVersion	DocumentBroker のバージョンの取得

3.8　トレースクラスのクラス，およびメソッド一覧

トレースクラスのクラス，およびメソッドを一覧形式で説明します。

3.8.1　トレースクラスのクラス

トレースクラスのクラスを次の表に示します。なお，クラスの使用方法については，「1.15　ユーザアプリケーションプログラムのトレース情報の出力」を参照してください。

表 3-40　トレースクラスのクラス

クラス	説明
トレース情報を扱うクラス	
DbjTrace クラス	トレース情報を扱うクラス

3.8.2　トレースクラスのメソッド一覧

トレースクラスの DbjTrace クラス（トレース情報を扱うクラス）のメソッドを次の表に示します。

表 3-41　DbjTrace クラスのメソッド

メソッド	機能
トレース情報を出力するメソッド	
arg	パラメタ情報の出力
call	外部 API の呼び出し情報の出力
DbjTrace	コンストラクタ
enter	メソッドの入口情報の出力
error	エラー情報の出力
exit	メソッドの出口情報の出力
hint	下位のメソッドのエラー情報の出力
init	トレースクラスの初期化
msg	メッセージの出力
returned	外部 API からのリターン情報の出力

3.9 クラス，インターフェースおよびメソッドの説明形式

DocumentBroker クラスライブラリのクラス，インターフェースおよびメソッドについての説明形式について示します。各クラスまたは各インターフェースの説明では，そのクラスまたはインターフェースに定義されるメソッドについてだけ説明しています。上位のクラスまたはインターフェースで定義されているメソッドについては，定義元のクラスまたはインターフェースを参照してください。

3.9.1 メソッドで説明する項目

クラスおよびインターフェースの説明に続けて，そのクラスおよびインターフェースのメソッドを説明しています。上位のインターフェースから継承されるメソッドの詳細については，継承元のインターフェースのメソッドの説明を参照してください。

(1) 機能

メソッドの機能の詳細を説明しています。

(2) 形式

メソッドの形式を説明しています。複数の形式がある場合，「(a) 形式 1」，「(b) 形式 2」，「(c) 形式 3」などと表記します。

(3) 引数

引数について次のように説明しています。

引数（入力）
指定する値について説明しています。

引数（入出力）
指定した領域に設定される値について説明しています。

- 引数は「形式」で記述した文字列です。
- 省略できることを明記していない引数は，すべて指定する必要があります。
- （入力）は，値を指定することを示しています。
- （入出力）は，メソッドによって出力値が設定されるオブジェクトを指定することを示しています。

(4) 戻り値

メソッドの戻り値を説明しています。戻り値を返却しないメソッドでは，「なし」と記述します。

(5) 例外

発生する例外について説明しています。

3.9.2 パラメタクラスのインターフェースで説明する項目

パラメタクラスのインターフェースの説明では，次の項目について説明しています。

(1) 直接のスーパーインターフェース

該当するインターフェースの，直接のスーパーインターフェース名を記述しています。

なお，パラメタクラスのインターフェースは，java.util.Collection インターフェース，java.util.List イン

ターフェース，java.util.Map インターフェース，および java.lang.Cloneable インターフェースなどを継承しています。

(2) プロパティ一覧

クラスおよびインターフェースで定義されているプロパティ一覧を記述しています。プロパティ一覧の例を表 1-44 に示します。

表 3-42 プロパティ一覧の例

プロパティ名	データ型	getter メソッド	setter メソッド
aaa	xxx 型	getAaa	setAaa
bbb	yyy 型	getBbb	setBbb

プロパティ名
プロパティの名称を記述しています。

データ型
プロパティの値のデータ型を記述しています。インターフェースを扱う場合は，インターフェース名を記述しています。

getter メソッド
プロパティを取得するメソッド名を記述しています。

setter メソッド
プロパティを設定するメソッド名を記述しています。

3.9.3 文書管理オブジェクトのプロパティのデータ型

メソッドの説明で使用する文書管理オブジェクトのプロパティのデータ型と，DocumentBroker で扱うデータ型の対応を次の表に示します。

表 3-43 文書管理オブジェクトのプロパティのデータ型と DocumentBroker で扱うデータ型の対応

プロパティのデータ型	DocumentBroker で扱うデータ型
BOOL	Boolean 型
INT	Integer32 型
VARRAY	Object 型
STR	String 型
STRLIST	－

（凡例）
－：該当するデータ型がないことを示します。

DocumentBroker で扱うデータ型については，マニュアル「DocumentBroker Version 5　システム導入・運用ガイド」を参照してください。

4 ファクトリクラス詳細

この章では，ファクトリクラスのインターフェース，およびメソッドについて説明します。

4.1 DbjFactory0200 クラス

DbjFactory0200 クラスは，DocumentBroker クラスライブラリのファクトリインターフェース，およびメタマネージャーインターフェースを取得するクラスです。なお，このクラスはインスタンスを生成できません（コンストラクタは，private 属性です）。

DocumentBroker クラスライブラリは，このクラスのメソッドが実行される初回時に初期化されます。初期化によって，DocumentBroker クラスライブラリは動作環境の定義に従った内容で設定されます。ただし，初期化の実行時にエラーがあると，例外クラス DbjInitializeError がスローされるおそれがあります。

以降，DbjFactory0200 クラスのメソッドについて説明します。

4.1.1 getFactory（ファクトリインターフェースの取得）

(1) 機能

ファクトリインターフェースを取得します。

(2) 形式

```
static DbjFactory getFactory()
```

(3) 引数

なし

(4) 戻り値

ファクトリインターフェース（DbjFactory インターフェース）

(5) 例外

なし

4.1.2 getMetaManager（メタマネージャーインターフェースの取得）

(1) 機能

メタマネージャーインターフェースを取得します。

(2) 形式

```
static DbjMetaManager getMetaManager()
```

(3) 引数

なし

(4) 戻り値

メタマネージャーインターフェース（DbjMetaManager インターフェース）

(5) 例外

なし

4.2 DbjFactory インターフェース

DbjFactory インターフェースは，DocumentBroker クラスライブラリのオブジェクトを生成して，そのインターフェースを取得する機能を持つファクトリインターフェースです。DbjFactory インターフェースは，次の表に示すオブジェクトを作成してインターフェースを取得するメソッドを提供します。ユーザアプリケーションプログラムは，DbjFactory インターフェースを使用して DocumentBroker クラスライブラリのセッションオブジェクトを生成して，そのインターフェースを取得したり，パラメタクラスのオブジェクトを生成して，そのインターフェースを取得したりできます。

DbjFactory インターフェースで作成できるオブジェクト，そのオブジェクトを作成するメソッド，および取得するインターフェース名を次の表に示します。

表 4-1 DbjFactory インターフェースで作成できるオブジェクト，オブジェクトを作成するメソッド，および取得するインターフェース

オブジェクト	オブジェクトを作成するメソッド	取得するインターフェース
? パラメタ（BOOL 型）オブジェクト	createBooleanQParam	DbjBooleanQParam
? パラメタ（INT 型）オブジェクト	createInteger32QParam	DbjInteger32QParam
? パラメタ（Object 型）オブジェクト	createObjQParam	DbjObjQParam
? パラメタ（OIID）オブジェクト	createOIIDQParam	DbjOIIDQParam
? パラメタ（String 型）オブジェクト	createStringQParam	DbjStringQParam
? パラメタ（概念検索種文章）オブジェクト	createSeedDocQParam	DbjSeedDocQParam
ACE オブジェクト	createACE	DbjACE
アップロード情報オブジェクト	createUploadInfo	DbjUploadInfo
入力ストリームを使用したアップロード情報オブジェクト	createUploadInfoByStream	DbjUploadInfo
可変長配列オブジェクト	createVArray	DbjVArray
DocumentBroker のセッションオブジェクト	createSession	DbjSession
検索結果取得情報オブジェクト	createFetchInfo	DbjFetchInfo
参照型リンク設定情報オブジェクト	createSetRCRLinkInfo	DbjSetRCRLinkInfo
直接型リンク設定情報オブジェクト	createSetDCRLinkInfo	DbjSetDCRLinkInfo
パブリック ACL の OIID エレメントオブジェクト	createPublicACLIdElm	DbjPublicACLIdElm
プロパティ値集合オブジェクト	createPropSet	DbjPropSet
文書間リンク設定情報オブジェクト	createSetRelLinkInfo	DbjSetRelLinkInfo

オブジェクト	オブジェクトを作成するメソッド	取得するインターフェース
リファレンスファイル文書のアップロード情報オブジェクト	createReferenceUploadInfo	DbjReferenceUploadInfo
入力ストリームを使用したリファレンスファイル文書のアップロード情報オブジェクト	createReferenceUploadInfoByStream	DbjReferenceUploadInfo
リファレンスファイル文書のパス情報オブジェクト	createReferencePathInfo	DbjReferencePathInfo
コンテンツ情報オブジェクト	createConvertContentInfo	DbjConvertContentInfo

以降，DbjFactory インターフェースのメソッドについて説明します。

4.2.1 createACE（ACE の作成）

(1) 機能

ACE を作成し，そのインターフェース（DbjACE インターフェース）を取得します。

形式 1 では，初期値状態の ACE を作成します。形式 2 では，指定されたプロパティ値集合を ACE とみなして作成して propSet プロパティに設定します。この場合，値はコピーされません。形式 3 では，指定された引数を初期値として ACE を作成します。なお，形式 2 および形式 3 の引数に指定するプロパティ値集合の妥当性は，検証されません。

(2) 形式

(a) 形式 1

```
DbjACE createACE()
```

(b) 形式 2

```
DbjACE createACE(
     DbjPropSet  src
)
```

(c) 形式 3

```
DbjACE createACE(
     String      subject,
     int         subjectType,
     int         permission
)
```

(3) 引数

src（入力）
ACE として扱うプロパティ値集合を指定します。null を指定すると例外がスローされます。

subject（入力）
ACE のサブジェクトを指定します。

subjectType（入力）
ACE のサブジェクト種別を指定します。

permission（入力）
ACE のパーミッションを指定します。

(4) 戻り値

ACE インターフェース（DbjACE インターフェース）

(5) 例外

NullPointerException
引数 src に null を指定した場合

4.2.2 createBooleanQParam（BOOL 型の？パラメタの作成）

(1) 機能

BOOL 型の？パラメタを作成し，そのインターフェース（DbjBooleanQParam インターフェース）を取得します。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

```
DbjBooleanQParam createBooleanQParam(
     int     param
)
```

(3) 引数

param（入力）

パラメタの値を指定します。次に示す値のどれかを指定できます。

- DbjDef.DMA_TRUE
- DbjDef.DMA_FALSE
- DbjDef.DMA_UNKNOWN

(4) 戻り値

BOOL 型の？パラメタインターフェース（DbjBooleanQParam インターフェース）

(5) 例外

なし

4.2.3 createConvertContentInfo（コンテンツ情報オブジェクトの作成）

(1) 機能

コンテンツ種別変換機能で使用するコンテンツ情報オブジェクトを作成し，そのインターフェース（DbjConvertContentInfo）を取得します。

形式 1 ではオブジェクトのメンバはデフォルト値が入ります。形式 2 では，初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.5　DbjConvertContentInfo インターフェース」を参照してください。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

(a) 形式 1

```
DbjConvertContentInfo createConvertContentInfo()
```

(b) 形式 2

```
DbjConvertContentInfo createConvertContentInfo(
            int         convertType,
            int         executeMode,
            int         sourceScope,
            boolean     changeMaster,
            String      renditionType,
            boolean     checkRenditionStatus,
            boolean     investSourceComment,
            int         investMode,
            String      renditionComment,
            String      referenceTargetPath )
```

(3) 引数

convertType（入力）

convertType プロパティの初期値を指定します。

executeMode（入力）

executeMode プロパティの初期値を指定します。

sourceScope（入力）

sourceScope プロパティの初期値を指定します。

changeMaster（入力）

changeMaster プロパティの初期値を指定します。

renditionType（入力）

renditionType プロパティの初期値を指定します。

checkRenditionStatus（入力）

checkRenditionStatus プロパティの初期値を指定します。

investSourceComment（入力）

investSourceComment プロパティの初期値を指定します。

investMode（入力）

investMode プロパティの初期値を指定します。

renditionComment（入力）

renditionComment プロパティの初期値を指定します。

referenceTargetPath（入力）

referenceTargetPath プロパティの初期値を指定します。

（4） 戻り値

コンテンツ種別変換機能のコンテンツ情報インターフェース（DbjConvertContentInfo インターフェース）

（5） 例外

なし

4.2.4 createFetchInfo（検索結果取得情報オブジェクトの作成）

(1) 機能

検索結果取得情報オブジェクトを作成し，そのインターフェース（DbjFetchInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，初期値を引数で指定できます。形式 3 では，形式 2 で指定できる初期値に加えて，キャッシュ検索実行時に取得した結果のソートを制御するための引数 comp を指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.6　DbjFetchInfo インターフェース」を参照してください。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

(a) 形式 1

```
DbjFetchInfo createFetchInfo()
```

(b) 形式 2

```
DbjFetchInfo createFetchInfo(
     int               startIndex,
     int               fetchCount,
     int               maxFetchCount,
     String            cacheName,
     int               cacheKey
)
```

(c) 形式 3

```
DbjFetchInfo createFetchInfo(
     int               startIndex,
     int               fetchCount,
     int               maxFetchCount,
     Comparator        comp,
     String            cacheName,
     int               cacheKey
)
```

(3) 引数

startIndex（入力）

startIndex プロパティの初期値を指定します。

fetchCount（入力）

fetchCount プロパティの初期値を指定します。

maxFetchCount（入力）

maxFetchCount プロパティの初期値を指定します。

comp（入力）

comparator プロパティの初期値を指定します。

cacheName（入力）

cacheName プロパティの初期値を指定します。

cacheKey（入力）

cacheKey プロパティの初期値を指定します。

(4) 戻り値

検索結果取得情報インターフェース（DbjFetchInfo インターフェース）

(5) 例外

なし

4.2.5 createInteger32QParam（INT 型の？パラメタの作成）

(1) 機能

INT 型の？パラメタを作成し，そのインターフェース（DbjInteger32QParam インターフェース）を取得します。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

```
DbjInteger32QParam createInteger32QParam(
     int        param
)
```

(3) 引数

param（入力）

パラメタの値を指定します。

(4) 戻り値

INT 型の？パラメタインターフェース（DbjInteger32QParam インターフェース）

(5) 例外

なし

4.2.6 createOIIDQParam（OIID の？パラメタの作成）

(1) 機能

OIID の？パラメタを作成し，そのインターフェース（DbjOIIDQParam インターフェース）を取得します。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

```
DbjOIIDQParam createOIIDQParam(
     DbjObj     param
)
```

(3) 引数

param（入力）

パラメタの値を指定します。param.getOiid() の値が，？パラメタとして指定されます。

(4) 戻り値

OIID の？パラメタインターフェース（DbjOIIDQParam インターフェース）

(5) 例外

なし

4.2.7 createObjQParam（Object 型の？パラメタの作成）

(1) 機能

Object 型の？パラメタを作成し，そのインターフェース（DbjObjQParam インターフェース）を取得します。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

```
DbjObjQParam createObjQParam(
     DbjObj     param
)
```

(3) 引数

param（入力）

パラメタの値を指定します。

(4) 戻り値

Object 型の？パラメタインターフェース（DbjObjQParam インターフェース）

(5) 例外

なし

4.2.8 createPropSet（プロパティ値集合オブジェクトの作成）

（1） 機能

プロパティ値集合オブジェクトを作成し，そのインターフェース（DbjPropSet インターフェース）を取得します。

形式 1 では，空のプロパティ値集合オブジェクトを作成します。形式 2 では，指定されたマップをプロパティ値集合オブジェクトにマッピングして DbjPropSet インターフェースを取得します。

（2） 形式

（a） 形式 1

```
DbjPropSet createPropSet()
```

（b） 形式 2

```
DbjPropSet createPropSet(
     Map<String, Object>      src
)
```

（3） 引数

src（入力）

マッピングの元になるマップを指定します。要素キーにプロパティ名，要素の値にプロパティ値を設定しておきます。null を指定すると，要素数 0 の空のオブジェクトが仮定されます。

（4） 戻り値

プロパティ値集合インターフェース（DbjPropSet インターフェース）

（5） 例外

なし

4.2.9 createPublicACLIdElm（パブリック ACL の OIID エレメントの作成）

(1) 機能

パブリック ACL の OIID エレメントを作成し，そのインターフェース（DbjPublicACLIdElm インターフェース）を取得します。

形式 1 では，初期値状態のパブリック ACL の OIID エレメントを作成します。形式 2 では，指定されたプロパティ値集合をパブリック ACL の OIID エレメントとみなして取得して propSet プロパティに設定します（値はコピーされません）。形式 3 では，初期値に指定されたパブリック ACL の OIID を取得します。なお，形式 2 および形式 3 の引数に指定するプロパティ値集合の妥当性は検証されません。

(2) 形式

(a) 形式 1

```
DbjPublicACLIdElm createPublicACLIdElm()
```

(b) 形式 2

```
DbjPublicACLIdElm createPublicACLIdElm(
     DbjPropSet      src
)
```

(c) 形式 3

```
DbjPublicACLIdElm createPublicACLIdElm(
     String          publicACLId
)
```

(3) 引数

src（入力）

パブリック ACL の OIID エレメントとして扱うプロパティ値集合を指定します。null を指定した場合は，例外がスローされます。

publicACLId（入力）

パブリック ACL の OIID を指定します。null を指定した場合は，例外がスローされます。

(4) 戻り値

パブリック ACL の OIID エレメントインターフェース（DbjPublicACLIdElm インターフェース）

(5) 例外

NullPointerException

引数 src に null を指定した場合

4.2.10 createReferencePathInfo（リファレンスファイル文書のパス情報オブジェクトの作成）

（1） 機能

リファレンスファイル文書のパス情報オブジェクトを作成し，そのインターフェース（DbjReferencePathInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.14 DbjReferencePathInfo インターフェース」を参照してください。なお，引数に指定する値の妥当性は検証されません。

（2） 形式

（a） 形式 1

```
DbjReferencePathInfo createReferencePathInfo()
```

（b） 形式 2

```
DbjReferencePathInfo createReferencePathInfo(
        int           contentOperateMode,
        String        entry,
        String        targetPath,
        String        deleteRootPath )
```

（3） 引数

contentOperateMode（入力）

contentOperateMode プロパティの初期値を指定します。

entry（入力）

entry プロパティの初期値を指定します。

targetPath（入力）

targetPath プロパティの初期値を指定します。なお，文書空間で使用する文字コード種別が UTF-8 の場合に指定できる値は，印刷が可能な ASCII コードだけです。

deleteRootPath（入力）

deleteRootPath プロパティの初期値を指定します。なお，文書空間で使用する文字コード種別が UTF-8 の場合に指定できる値は，印刷が可能な ASCII コードだけです。

（4） 戻り値

リファレンスファイル文書のパス情報インターフェース（DbjReferencePathInfo インターフェース）

（5） 例外

なし

4.2.11 createReferenceUploadInfo（リファレンスファイル文書のアップロード情報オブジェクトの作成）

(1) 機能

リファレンスファイル文書のアップロード情報オブジェクトを作成し，そのインターフェース（DbjReferenceUploadInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.15 DbjReferenceUploadInfo インターフェース」を参照してください。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

(a) 形式 1

```
DbjReferenceUploadInfo createReferenceUploadInfo()
```

(b) 形式 2

```
DbjReferenceUploadInfo createReferenceUploadInfo(
      String                    filePath,
      String                    retrievalName,
      String                    renditionType,
      DbjPropSet                renditionPropSet,
      String                    indexPath,
      DbjReferencePathInfo      referencePathInfo )
```

(3) 引数

filePath（入力）

filePath プロパティの初期値を指定します。

retrievalName（入力）

retirievalName プロパティの初期値を指定します。

renditionType（入力）

renditionType プロパティの初期値を指定します。

renditionPropSet（入力）

renditionPropSet プロパティの初期値を指定します。

indexPath（入力）

indexPath プロパティの初期値を指定します。

referencePathInfo（入力）

referencePathInfo プロパティの初期値を指定します。

(4) 戻り値

リファレンスファイル文書のアップロード情報インターフェース（DbjReferenceUploadInfo インターフェース）

(5) 例外

なし

4.2.12 createReferenceUploadInfoByStream（入力ストリームを使用したリファレンスファイル文書のアップロード情報オブジェクトの作成）

（1） 機能

入力ストリームを使用してリファレンスファイル文書のアップロード情報オブジェクトを作成し，そのインターフェース（DbjReferenceUploadInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.15 DbjReferenceUploadInfo インターフェース」を参照してください。なお，引数に指定する値の妥当性は検証されません。

（2） 形式

（a） 形式 1

```
DbjReferenceUploadInfo createReferenceUploadInfoByStream()
```

（b） 形式 2

```
DbjReferenceUploadInfo createReferenceUploadInfoByStream(
      java.io.InputStream       fileStream,
      String                    retrievalName,
      String                    renditionType,
      DbjPropSet                renditionPropSet,
      java.io.InputStream       indexStream,
      DbjReferencePathInfo      referencePathInfo )
```

（3） 引数

fileStream（入力）

fileStream プロパティの初期値を指定します。

retrievalName（入力）

retirievalName プロパティの初期値を指定します。

renditionType（入力）

renditionType プロパティの初期値を指定します。

renditionPropSet（入力）

renditionPropSet プロパティの初期値を指定します。

indexStream（入力）

indexStream プロパティの初期値を指定します。

referencePathInfo（入力）

referencePathInfo プロパティの初期値を指定します。

（4） 戻り値

リファレンスファイル文書のアップロード情報インターフェース（DbjReferenceUploadInfo インターフェース）

（5） 例外

なし

4.2.13 createSeedDocQParam（種文章の？パラメタの作成）

(1) 機能

概念検索で指定する種文章の？パラメタを作成し，そのインターフェース（DbjSeedDocQParam インターフェース）を取得します。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

```
DbjSeedDocQParam createSeedDocQParam(
     String     param
)
```

(3) 引数

param（入力）

パラメタの値を指定します。

(4) 戻り値

概念検索の種文章の？パラメタインターフェース（DbjSeedDocQParam インターフェース）

(5) 例外

なし

4.2.14 createSession（DocumentBroker クラスライブラリのセッションオブジェクトの作成）

（1） 機能

指定した文書空間に対する DocumentBroker クラスライブラリのセッションオブジェクトを作成し，そのインターフェース（DbjSession インターフェース）を取得します。ただし，DbjSession インターフェースを取得した時点では文書空間にアクセスしません。

形式 1 では，セッションを開始する文書空間の文書空間識別子を指定します。形式 2 では，動作環境定義ファイルに指定したデフォルト文書空間（コンフィギュレーションキーは DefaultDocSpaceId）に接続するためのセッションオブジェクトを作成し，そのインターフェース（DbjSession インターフェース）を取得します。

（2） 形式

（a） 形式 1

```
DbjSession createSession(
     String     docspaceId
)
```

（b） 形式 2

```
DbjSession createSession()
```

（3） 引数

docspaceId（入力）

文書空間識別子を GUID 文字列で指定します。GUID 文字列は，16 進数「X」によって，「XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX（8 けた -4 けた -4 けた -4 けた -12 けた）」の形式で表されます。「X」は，0 ～ 9，a ～ f（小文字），および A ～ F（大文字）のどれかです。null を指定した場合は，例外がスローされます。

（4） 戻り値

DocumentBroker クラスライブラリのセッションインターフェース（DbjSession インターフェース）

（5） 例外

NullPointerException

引数 docspaceId に null を指定した場合

4.2.15 createSetDCRLinkInfo（直接型リンク設定情報の作成）

(1) 機能

直接型リンク設定情報オブジェクトを作成し，そのインターフェース（DbjSetDCRLinkInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメ LINKID=C18"> タの詳細については，「5.20　DbjSetDCRLinkInfo インターフェース」を参照してください。

(2) 形式

(a) 形式 1

```
DbjSetDCRLinkInfo createSetDCRLinkInfo()
```

(b) 形式 2

```
DbjSetDCRLinkInfo createSetDCRLinkInfo(
     DbjObj               obj,
     DbjPropSet           linkProps
)
```

(3) 引数

obj（入力）

リンク対象の文書管理オブジェクトを指定します。文書管理オブジェクトを未設定にする場合は，null を指定できます。初期値のデフォルト値は null です。

linkProps（入力）

リンクオブジェクトに設定するプロパティのプロパティ値集合を指定します。プロパティを未設定にする場合は，null を指定できます。初期値のデフォルト値は null です。

(4) 戻り値

直接型リンク設定情報インターフェース（DbjSetDCRLinkInfo インターフェース）

(5) 例外

なし

4.2.16 createSetRCRLinkInfo（参照型リンク設定情報の作成）

（1） 機能

参照型リンク設定情報オブジェクトを作成し，そのインターフェース（DbjSetRCRLinkInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.22 DbjSetRCRLinkInfo インターフェース」を参照してください。

（2） 形式

（a） 形式 1

```
DbjSetRCRLinkInfo createSetRCRLinkInfo()
```

（b） 形式 2

```
DbjSetRCRLinkInfo createSetRCRLinkInfo(
     DbjObj           obj,
     DbjPropSet       linkProps
)
```

（3） 引数

obj（入力）

リンク対象の文書管理オブジェクトを指定します。文書管理オブジェクトを未設定にする場合は，null を指定できます。初期値のデフォルト値は null です。

linkProps（入力）

リンクオブジェクトに設定するプロパティのプロパティ値集合を指定します。プロパティを未設定にする場合は，null を指定できます。初期値のデフォルト値は null です。

（4） 戻り値

参照型リンク設定情報インターフェース（DbjSetRCRLinkInfo インターフェース）

（5） 例外

なし

4.2.17 createSetRelLinkInfo（文書間リンク設定情報の作成）

(1) 機能

文書間リンク設定情報オブジェクトを作成し，そのインターフェース（DbjSetRelLinkInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.23 DbjSetRelLinkInfo インターフェース」を参照してください。

(2) 形式

(a) 形式 1

```
DbjSetRelLinkInfo createSetRelLinkInfo()
```

(b) 形式 2

```
DbjSetRelLinkInfo createSetRelLinkInfo (
     DbjObj           obj,
     DbjPropSet       linkProps
)
```

(3) 引数

obj（入力）

リンク対象の文書管理オブジェクト（バージョン付きオブジェクト）を指定します。文書管理オブジェクトを未設定にする場合は，null を指定できます。初期値のデフォルト値は null です。

linkProps（入力）

リンクオブジェクトに設定するプロパティのプロパティ値集合を指定します。プロパティを未設定にする場合は，null を指定できます。初期値のデフォルト値は null です。

(4) 戻り値

文書間リンク設定情報インターフェース（DbjSetRelLinkInfo インターフェース）

(5) 例外

なし

4.2.18 createStringQParam（String 型の？パラメタの作成）

（1）機能

String 型の？パラメタを作成し，そのインターフェース（DbjStringQParam インターフェース）を取得します。なお，引数に指定する値の妥当性は検証されません。

（2）形式

```
DbjStringQParam createStringQParam(
     String     param
)
```

（3）引数

param（入力）

？パラメタの値を指定します。

（4）戻り値

String 型の？パラメタ（DbjStringQParam インターフェース）

（5）例外

なし

4.2.19 createUploadInfo（文書のアップロード情報オブジェクトの作成）

(1) 機能

文書アップロード情報オブジェクトを作成し，そのインターフェース（DbjUploadInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.25 DbjUploadInfo インターフェース」を参照してください。なお，引数に指定する値の妥当性は検証されません。

(2) 形式

(a) 形式 1

```
DbjUploadInfo createUploadInfo()
```

(b) 形式 2

```
DbjUploadInfo createUploadInfo(
     String          filePath,
     String          retrievalName,
     String          renditionType,
     DbjPropSet      renditionPropSet,
     String          indexPath
)
```

(3) 引数

filePath（入力）

filePath プロパティの初期値を指定します。

retrievalName（入力）

retrievalName プロパティの初期値を指定します。

renditionType（入力）

renditionType プロパティの初期値を指定します。

renditionPropSet（入力）

renditionPropSet プロパティの初期値を指定します。

indexPath（入力）

indexPath プロパティの初期値を指定します。

(4) 戻り値

アップロード情報インターフェース（DbjUploadInfo インターフェース）

(5) 例外

なし

4.2.20 createUploadInfoByStream（入力ストリームを使用した文書のアップロード情報オブジェクトの作成）

（1） 機能

入力ストリームを使用した文書のアップロード情報オブジェクトを作成し，そのインターフェース（DbjUploadInfo インターフェース）を取得します。

形式 1 では，オブジェクトのメンバにデフォルト値が入ります。形式 2 では，オブジェクトの初期値を引数で指定できます。オブジェクトの初期値として設定できるパラメタの詳細については，「5.25 DbjUploadInfo インターフェース」を参照してください。なお，引数に指定する値の妥当性は検証されません。

（2） 形式

（a） 形式 1

```
DbjUploadInfo createUploadInfoByStream()
```

（b） 形式 2

```
DbjUploadInfo createUploadInfoByStream(
     java.io.InputStream      fileStream,
     String                   retrievalName,
     String                   renditionType,
     DbjPropSet               renditionPropSet,
     java.io.InputStream      indexStream
)
```

（3） 引数

fileStream（入力）

fileStream プロパティの初期値を指定します。

retrievalName（入力）

retrievalName プロパティの初期値を指定します。

renditionType（入力）

renditionType プロパティの初期値を指定します。

renditionPropSet（入力）

renditionPropSet プロパティの初期値を指定します。

indexStream（入力）

indexStream プロパティの初期値を指定します。

（4） 戻り値

アップロード情報インターフェース（DbjUploadInfo インターフェース）

（5） 例外

なし

4.2.21 createVArray（可変長配列オブジェクトの作成）

(1) 機能

可変長配列オブジェクトを作成し，そのインターフェース（DbjVArray インターフェース）を取得します。引数に指定されたメタプロパティを保持する要素数 0 の空の可変長配列オブジェクトを作成します。

(2) 形式

```
DbjVArray createVArray(
     Collection<String>     propDefs
)
```

(3) 引数

propDefs（入力）

各要素は，String 型のプロパティになります。可変長配列のメタプロパティのコレクションを指定します。指定を省略する場合は，各要素に null を指定できます。この場合，要素数 0 の空のオブジェクトが仮定されます。

(4) 戻り値

可変長配列インターフェース（DbjVArray インターフェース）

(5) 例外

なし

5 パラメタクラス詳細

この章では，パラメタクラスのインターフェース，およびメソッドについて説明します。

5.1 DbjACE インターフェース

DbjACE インターフェースは，ACE の値を扱うインターフェースです。ACE の値を取得，または設定するメソッドを提供します。また，サブジェクトと同時にサブジェクト種別を設定するメソッドも提供します。

なお，DocumentBroker クラスライブラリでは，ACL を可変長配列として扱い，ACE を可変長配列の要素として扱います。このため，ACL は DbjVArray インターフェースでも操作でき，ACE は DbjPropSet インターフェースでも操作できます。

プロパティ一覧

DbjACE インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-1 DbjACE インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
permission	int 型	getPermission	setPermission
propSet	DbjPropSet	propSet	setPropSet
subject	String 型	getSubject	setSubject
subjectType	int 型	getSubjectType	setSubjectType

プロパティ詳細

- permission プロパティ
  ACE のパーミッションを表します。
  このプロパティに対する操作は，プロパティ値集合の dbrProp_Permission プロパティに反映されます。
  初期値は DbjDef.PERM_NONE（アクセス権なし）です。
- propSet プロパティ
  DbjACE インターフェースで扱うプロパティ値の集合です。
  このプロパティ値集合に対する操作は，集合に含まれる各プロパティ値に反映されます。また，各プロパティ値に対する操作もこのプロパティ値集合に反映されます。
- subject プロパティ
  ACE のサブジェクトを表します。
  このプロパティに対する操作は，プロパティ値集合の dbrProp_Subject プロパティに反映されます。
  初期値は null です。
- subjectType プロパティ
  ACE のサブジェクト種別を表します。
  このプロパティに対する操作は，プロパティ値集合の dbrProp_SubjectType プロパティに反映されます。
  初期値は 0 です。

以降，DbjACE インターフェースのメソッドについて説明します。

5.1.1 getPermission（パーミッションの取得）

(1) 機能

ACE のパーミッションを取得します。

permission プロパティに値が設定されていない場合は，0 が返却されます。

(2) 形式

```
int getPermission()
```

(3) 引数

なし

(4) 戻り値

ACE のパーミッションが，次に示す定数で返却されます。

- DbjDef.PERM_NONE
  アクセス権なし
- DbjDef.PERM_PRIM_READ_PROPS
  基本プロパティ参照権
- DbjDef.PERM_PRIM_WRITE_PROPS
  基本プロパティ更新権
- DbjDef.PERM_PRIM_READ_CONTENTS
  基本コンテンツ参照権
- DbjDef.PERM_PRIM_WRITE_CONTENTS
  基本コンテンツ更新権
- DbjDef.PERM_PRIM_LINK
  基本リンク権
- DbjDef.PERM_PRIM_VERSION
  基本バージョン管理権
- DbjDef.PERM_PRIM_DELETE
  基本削除権
- DbjDef.PERM_CHANGE_PERM
  アクセス制御情報変更権
- DbjDef.PERM_CREATE
  オブジェクト作成権
- DbjDef.PERM_READ_PROPS
  プロパティ参照権
- DbjDef.PERM_READ
  参照権
- DbjDef.PERM_WRITE_PROPS
  プロパティ更新権
- DbjDef.PERM_READ_WRITE
  参照更新権
- DbjDef.PERM_DELETE
  削除権
- DbjDef.PERM_LINK
  リンク権

- DbjDef.PERM_VERSION
  バージョン権
- DbjDef.PERM_FULL_CONTROL
  フルコントロール

(5) 例外

なし

5.1.2 getSubject（サブジェクトの取得）

（1）機能

ACE のサブジェクトを取得します。

subject プロパティに値が設定されていない場合は，null が返却されます。

（2）形式

```
String getSubject()
```

（3）引数

なし

（4）戻り値

subject プロパティ

ACE のサブジェクトが返却されます。

（5）例外

なし

5.1.3 getSubjectType（サブジェクト種別の取得）

(1) 機能

ACE のサブジェクト種別を取得します。

subjectType プロパティに値が設定されていない場合は，0 が返却されます。

(2) 形式

```
int getSubjectType()
```

(3) 引数

なし

(4) 戻り値

ACE のサブジェクト種別が，次に示す定数で返却されます。

- DbjDef.SUBJECTTYPE_USR
  ユーザサブジェクト
- DbjDef.SUBJECTTYPE_GRP
  グループサブジェクト
- DbjDef.SUBJECTTYPE_SYS
  システムサブジェクト

(5) 例外

なし

5.1.4 propSet（ACE のプロパティ値集合の取得）

（1） 機能

ACE の値を要素としたプロパティ値集合を取得します。

取得するプロパティ値集合の各要素は，プロパティ値の参照です。なお，値が設定されていなくても，null は返却されません。

（2） 形式

```
DbjPropSet propSet()
```

（3） 引数

なし

（4） 戻り値

propSet プロパティ（DbjPropSet インターフェース）

（5） 例外

なし

5.1.5 setGroupSubject（サブジェクトをグループサブジェクトとして設定）

(1) 機能

指定したサブジェクトをグループサブジェクトとして設定します。subjectType プロパティに，DbjDef.SUBJECTTYPE_GRP（グループサブジェクト）が自動的に設定されます。

なお，指定した値の妥当性は検証されません。

(2) 形式

```
void setGroupSubject(
     String     grpSubject
)
```

(3) 引数

grpSubject（入力）

グループサブジェクトとして設定するサブジェクトを指定します。

(4) 戻り値

なし

(5) 例外

なし

5.1.6 setPermission（パーミッションの設定）

(1) 機能

ACE のパーミッションを設定します。

なお，指定した値の妥当性は検証されません。

(2) 形式

```
void setPermission(
     int       permission
)
```

(3) 引数

permission（入力）

ACE のパーミッションを指定します。次に示す定数のどれかを指定できます。

- DbjDef.PERM_NONE
  アクセス権なし
- DbjDef.PERM_PRIM_READ_PROPS
  基本プロパティ参照権
- DbjDef.PERM_PRIM_WRITE_PROPS
  基本プロパティ更新権
- DbjDef.PERM_PRIM_READ_CONTENTS
  基本コンテンツ参照権
- DbjDef.PERM_PRIM_WRITE_CONTENTS
  基本コンテンツ更新権
- DbjDef.PERM_PRIM_LINK
  基本リンク権
- DbjDef.PERM_PRIM_VERSION
  基本バージョン管理権
- DbjDef.PERM_PRIM_DELETE
  基本削除権
- DbjDef.PERM_CHANGE_PERM
  アクセス制御情報変更権
- DbjDef.PERM_CREATE
  オブジェクト作成権
- DbjDef.PERM_READ_PROPS
  プロパティ参照権
- DbjDef.PERM_READ
  参照権
- DbjDef.PERM_WRITE_PROPS
  プロパティ更新権
- DbjDef.PERM_READ_WRITE
  参照更新権
- DbjDef.PERM_DELETE
  削除権
- DbjDef.PERM_LINK
  リンク権
- DbjDef.PERM_VERSION

バージョン権
- DbjDef.PERM_FULL_CONTROL
  フルコントロール

(4) 戻り値

なし

(5) 例外

なし

5.1.7 setPropSet（ACE のプロパティ値集合の設定）

（1）機能

ACE の値に，指定したプロパティ値集合の内容をコピーして設定します。

（2）形式

```
void setPropSet(
     DbjPropSet  propSet
)
```

（3）引数

propSet（入力）

設定するプロパティ値集合を指定します。

（4）戻り値

なし

（5）例外

NullPointerException

引数 propSet に null を指定した場合

5.1.8 setSubject（サブジェクトの設定）

（1） 機能

ACE のサブジェクトを設定します。

なお，指定した値の妥当性は検証されません。

（2） 形式

```
void setSubject(
     String      subject
)
```

（3） 引数

subject（入力）

ACE のサブジェクトを指定します。

（4） 戻り値

なし

（5） 例外

なし

5.1.9 setSubjectType（サブジェクト種別の設定）

（1）機能

ACE のサブジェクト種別を設定します。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setSubjectType(
     int       subjectType
)
```

（3）引数

subjectType（入力）

ACE のサブジェクト種別を指定します。次に示す定数のどれかを指定できます。

- DbjDef.SUBJECTTYPE_USR
  ユーザサブジェクト
- DbjDef.SUBJECTTYPE_GRP
  グループサブジェクト
- DbjDef.SUBJECTTYPE_SYS
  システムサブジェクト

（4）戻り値

なし

（5）例外

なし

5.1.10 setSystemSubject（サブジェクトをシステムサブジェクトとして設定）

（1）機能

指定したサブジェクトをシステムサブジェクトとして設定します。subjectType プロパティに，DbjDef.SUBJECTTYPE_SYS が自動的に設定されます。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setSystemSubject(
     String      sysSubject
)
```

（3）引数

sysSubject（入力）

システムサブジェクトとして設定するサブジェクトを指定します。次に示す定数のどちらかを指定できます。

- DbjDef.SYSSUBJECT_SELF
  対象オブジェクトの所有者
- DbjDef.SYSSUBJECT_EVERYONE
  すべてのユーザ

（4）戻り値

なし

（5）例外

なし

5.1.11 setUserSubject（サブジェクトをユーザサブジェクトとして設定）

(1) 機能

指定したサブジェクトをユーザサブジェクトとして設定します。subjectType プロパティに，DbjDef.SUBJECTTYPE_USR が自動的に設定されます。

なお，指定した値の妥当性は検証されません。

(2) 形式

```
void setUserSubject(
     String      usrSubject
)
```

(3) 引数

usrSubject（入力）

ユーザサブジェクトとして設定するサブジェクトを指定します。

(4) 戻り値

なし

(5) 例外

なし

5.2 DbjBooleanQParam インターフェース

DbjBooleanQParam インターフェースは，BOOL 型を表す？パラメタを扱うインターフェースです。

直接のスーパーインターフェース

- DbjQParam

5.3 DbjCheckOutInfo インターフェース

DbjCheckOutInfo インターフェースは，バージョン付きオブジェクトのチェックアウト情報を扱うインターフェースです。バージョン付きオブジェクトのチェックアウト情報を取得するメソッドを提供します。チェックアウト中の場合は，チェックアウトしたユーザのユーザ識別子および仮のバージョン識別子を取得できます。

プロパティ一覧

DbjCheckOutInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドを次の表に示します。

表 5-2 DbjCheckOutInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド
checkOut	boolean 型	isCheckOut
checkOutUserId	String 型	getCheckOutUserId
checkOutVersionId	String 型	getCheckOutVersionId

プロパティ詳細

- checkOut プロパティ
  バージョン付きオブジェクトがチェックアウト中かどうかを表します。
  チェックアウト中の場合は true となり，チェックアウトしていない場合は false となります。
- checkOutUserId プロパティ
  チェックアウトしたユーザのユーザ識別子を表します。
  チェックアウト中の場合にだけ有効です。チェックアウトしていない場合は null となります。
- checkOutVersionId プロパティ
  チェックアウト中の仮のバージョン識別子を表します。
  チェックアウト中の場合にだけ有効です。チェックアウトしていない場合は null となります。

以降，DbjCheckOutInfo インターフェースのメソッドについて説明します。

5.3.1 getCheckOutUserId（チェックアウトしたユーザのユーザ識別子の取得）

(1) 機能

チェックアウトしたユーザのユーザ識別子を取得します。

チェックアウトしていない場合は，null が返却されます。

(2) 形式

```
String getCheckOutUserId()
```

(3) 引数

なし

(4) 戻り値

checkOutUserId プロパティ

(5) 例外

なし

5.3.2 getCheckOutVersionId（仮のバージョン識別子の取得）

(1) 機能

チェックアウト中の仮のバージョン識別子を取得します。

チェックアウトしていない場合は，null が返却されます。

(2) 形式

```
String getCheckOutVersionId()
```

(3) 引数

なし

(4) 戻り値

checkOutVersionId プロパティ

(5) 例外

なし

5.3.3 isCheckOut（チェックアウト中かどうかの判定）

(1) 機能

チェックアウト中かどうかを判定します。

(2) 形式

```
boolean isCheckOut()
```

(3) 引数

なし

(4) 戻り値

true
　チェックアウト中です。

false
　チェックアウトしていません。

(5) 例外

なし

5.4 DbjContentInfo インターフェース

DbjContentInfo インターフェースは，文書のコンテンツ情報を扱うインターフェースです。コンテンツのレンディションタイプ，ファイル名および入力ストリームを取得するメソッドを提供します。

プロパティー覧

DbjContentInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドを次の表に示します。

表 5-3 DbjContentInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド
renditionType	String 型	getRenditionType
retrievalName	String 型	getRetrievalName
InputStream	java.io.InputStream	getInputStream

プロパティ詳細

- renditionType プロパティ
  コンテンツのレンディションタイプを表します。文書オブジェクトの dbrProp_RenditionType プロパティと同じ値です。
- retrievalName プロパティ
  コンテンツのファイル名を表します。文書オブジェクトの dbrProp_RetrievalName プロパティと同じ値です。
- InputStream プロパティ
  コンテンツの入力ストリームを表します。

以降，DbjContentInfo インターフェースのメソッドについて説明します。

5.4.1 getRenditionType（コンテンツのレンディションタイプの取得）

(1) 機能

コンテンツのレンディションタイプを取得します。

(2) 形式

```
String getRenditionType()
```

(3) 引数

なし

(4) 戻り値

renditionType プロパティ

(5) 例外

なし

5.4.2 getRetrievalName（コンテンツのファイル名の取得）

（1）機能

コンテンツのファイル名を取得します。

（2）形式

```
String getRetrievalName()
```

（3）引数

なし

（4）戻り値

retrievalName プロパティ

（5）例外

なし

5.4.3 getInputStream（コンテンツの入力ストリームの取得）

(1) 機能

コンテンツを読み込むための入力ストリームを取得します。取得した入力ストリームが不要になった時点で，必ず DbjContentInfo#close() メソッドを実行してください。

(2) 形式

```
java.io.InputStream getInputStream()
```

(3) 引数

なし

(4) 戻り値

文書のコンテンツを読み込むための入力ストリーム

(5) 例外

DbjIOException

DocumentBroker クラスライブラリ固有の IO エラーの場合

5.4.4 getReferenceContentInfo（DbjReferenceContentInfo の getter）

（1） 機能

DbjObj#downloadContents の形式 3 を実行した結果，リファレンスファイル文書であった場合，リファレンスファイル文書のコンテンツ情報を取得します。リファレンスファイル文書以外の文書に対してこのメソッドを実行した場合は，null を返却します。

（2） 形式

```
DbjReferenceContentInfo getReferenceContentInfo()
```

（3） 引数

なし

（4） 戻り値

リファレンスファイル文書のコンテンツ情報（DbjReferenceContentInfo インタフェース）

（5） 例外

なし

5.4.5 close（コンテンツ情報のクローズ）

(1) 機能

コンテンツ情報が保持している入力ストリームをクローズします。また，DbjObj#downloadContents の形式 3 を実行したときに作成された一時ファイルを削除します。

(2) 形式

```
void close()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

なし

5.5 DbjConvertContentInfo インターフェース

DbjConvertContentInfo インターフェースは，コンテンツ変換機能を使用した文書のコンテンツ種別の変換時に使用するコンテンツ情報を扱うインターフェースです。

プロパティ一覧

DbjConvertContentInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-4 DbjConvertContentInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
convertType	int 型	getConvertType	setConvertType
executeMode	int 型	getExecuteMode	setExecuteMode
sourceScope	int 型	getSourceScope	setSourceScope
renditionType	String 型	getRenditionType	setRenditionType
investMode	int 型	getInvestMode	setInvestMode
renditionComment	String 型	getRenditionComment	setRenditionComment
referenceTargetPath	String 型	getReferenceTargetPath	setReferenceTargetPath
changeMaster	boolean 型	isChangeMaster	setChangeMaster
checkRenditionStatus	boolean 型	isCheckRenditionStatus	setCheckRenditionStatus
investSourceComment	boolean 型	isInvestSourceComment	setInvestSourceComment

プロパティ詳細

- convertType プロパティ
  変換する文書の変換先コンテンツ種別を設定します。
  初期値は DbjDef.CONTENTTYPE_REFERENCE（シングルファイル文書をリファレンス文書に変換する）です。
- executeMode プロパティ
  変換する文書の変換元レンディションを残すかどうかの変換モードを設定します。
  初期値は DbjConvertDef.EXECUTEMODE_REPLACE（変換元のレンディションを残さない）です。
- sourceScope プロパティ
  変換対象とするレンディションの範囲を設定します。
  初期値は DbjConvertDef.SOURCESCOPE_ALL（すべてのレンディションを変換対象とする）です。
- changeMaster プロパティ
  executeMode プロパティに DbjConvertDef.EXECUTE_VERBOSE（変換する文書の変換元レンディションを残す）を指定した場合で，マスタレンディションを変換するときに，変換後のレンディションをマスタレンディションに変更するかどうかを設定します。
  初期値は true（変換後のレンディションをマスタレンディションに変更する）です。
- renditionType プロパティ
  sourceScope プロパティに DbjConvertDef.SOURCESCOPE_SEPARATE（renditionType プロパティで指定したレンディションを変換対象とする）を指定した場合に，変換の対象とするレンディションタイプを設定します。
  sourceScope プロパティに DbjConvertDef.SOURCESCOPE_SEPARATE 以外が指定された場合

は，このプロパティの指定は無視されます。
初期値は null（マスタレンディション）です。

- checkRenditionStatus プロパティ
  変換対象のレンディションのステータスをチェックするかどうかを設定します。
  初期値は false（レンディションステータスをチェックしません）です。
- investSourceComment プロパティ
  renditionComment プロパティで指定するレンディションコメントを変換前のレンディションに付与するか，変換後のレンディションに付与するかを設定します。初期値は false（変換後のレンディションにコメントを付与する）です。
- investMode プロパティ
  コメントの付与方式を設定します。
  初期値は DbjConvertDef.INVESTMODE_REPLACE（コメント部分を指定したコメントで置き換える）です。
- renditionComment プロパティ
  レンディションに付与するレンディションコメントを設定します。
- referenceTargetPath プロパティ
  コンテンツの登録先となるサーバマシン上のディレクトリのパスを設定します。
  初期値は null です。

5.5.1 getConvertType（変換先コンテンツ種別の取得）

(1) 機能

変換先のコンテンツ種別を取得します。

(2) 形式

```
int getConvertType ()
```

(3) 引数

なし

(4) 戻り値

convertType プロパティ

変換先のコンテンツ種別が，次に示す定数で返却されます。

- DbjDef.CONTENTTYPE_REFERENCE
  シングル文書をリファレンスファイル文書に変換する
- DbjDef.CONTENTTYPE_CONTENT
  リファレンス文書をシングルファイル文書に変換する

(5) 例外

なし

5.5.2 getExecuteMode（変換モードの取得）

(1) 機能

変換する文書の変換元レンディションを残すかどうかの変換モードを取得します。

(2) 形式

```
int getExecuteMode ()
```

(3) 引数

なし

(4) 戻り値

executeMode プロパティ

変換する文書の変換元レンディションを残すかどうかの変換モードが，次に示す定数で返却されます。

- DbjConvertDef.EXECUTE_REPLACE
  変換元のレンディションを残さない
- DbjConvertDef.EXECUTE_VERBOSE
  変換元のレンディションを残す

(5) 例外

なし

5.5.3 getSourceScope（変換対象レンディション範囲の取得）

（1）機能

変換対象とするレンディションの範囲を取得します。

（2）形式

```
int getSourceScope ()
```

（3）引数

なし

（4）戻り値

sourceScope プロパティ

変換元レンディションを残すかどうかの変換モードが，次に示す定数で返却されます。

- DbjConvertDef.SOURCESCOPE_ALL
  すべてのレンディションを変換対象とする
- DbjConvertDef.SOURCESCOPE_SEPARATE
  renditionType プロパティで指定したレンディションを変換対象とする

（5）例外

なし

5.5.4 getRenditionType（変換対象レンディションタイプの取得）

(1) 機能

sourceScope プロパティに DbjConvertDef.SOURCESCOPE_SEPARATE（renditionType プロパティで指定したレンディションを変換対象とする）を設定している場合に，変換対象のレンディションタイプを取得します。

(2) 形式

```
String getRenditionType ()
```

(3) 引数

なし

(4) 戻り値

renditionType プロパティ

変換対象のレンディションタイプが返却されます。

(5) 例外

なし

5.5.5 getInvestMode（コメント付与方式の取得）

（1）機能

コメントの付与方式を取得します。

（2）形式

```
int getInvestMode ()
```

（3）引数

なし

（4）戻り値

investMode プロパティ

コメントの付与方式が，次に示す定数で返却されます。

- DbjConvertDef.INVESTMODE_APPEND
  レンディションタイプの末尾にコメントを追加する
- DbjConvertDef.INVESTMODE_REPLACE
  レンディションタイプの末尾に存在するコメントを指定したコメントで置き換える（複数のコメントが存在する場合は，末尾のコメントを置き換えます。コメントがない場合は，レンディションタイプの末尾にコメントを追加します。）

（5）例外

なし

5.5.6 getRenditionComment（レンディションコメントの取得）

(1) 機能

レンディションに付与されたコメントを取得します。

(2) 形式

```
String getRenditionComment ()
```

(3) 引数

なし

(4) 戻り値

renditionComment プロパティ

レンディションに付与されたコメントが返却されます。

(5) 例外

なし

5.5.7 getReferenceTargetPath（コンテンツ登録先ディレクトリパスの取得）

(1) 機能

コンテンツ登録先のサーバマシン上のディレクトリパスを取得します。

(2) 形式

```
String getReferenceTargetPath ()
```

(3) 引数

なし

(4) 戻り値

referenceTargetPath プロパティ

コンテンツ登録先のサーバマシン上のディレクトリパスが返却されます。

(5) 例外

なし

5.5.8 isChangeMaster（変換後のレンディション種別取得）

（1） 機能

executeMode プロパティに DbjConvertDef.EXECUTE_VERBOSE（変換元のレンディションを残す）を設定してマスタレンディションを変換する場合に，変換後のレンディションをマスタレンディションに変更するかどうかを取得します。

（2） 形式

```
boolean isChangeMaster ()
```

（3） 引数

なし

（4） 戻り値

true
　変換後のレンディションをマスタレンディションに変更する

false
　変換後のレンディションをマスタレンディションに変更しない

（5） 例外

なし

5.5.9 isCheckRenditionStatus（レンディションステータスのチェックの取得）

(1) 機能

変換対象のレンディションステータスをチェックするかどうかを取得します。

(2) 形式

```
boolean isCheckRenditionStatus ()
```

(3) 引数

なし

(4) 戻り値

true
　変換対象のレンディションステータスをチェックする

false
　変換対象のレンディションステータスをチェックしない

(5) 例外

なし

5.5.10 isInvestSourceComment（レンディションコメントの付与箇所の取得）

(1) 機能

レンディションコメントを変換前のレンディションに付与するか，変換後のレンディションに付与するかを取得します。

(2) 形式

```
boolean isInvestSourceComment ()
```

(3) 引数

なし

(4) 戻り値

true
　変換前のレンディションにコメントを付与する

false
　変換後のレンディションにコメントを付与する

(5) 例外

なし

5.5.11 setConvertType（変換先コンテンツ種別の設定）

（1）機能

変換先のコンテンツ種別を設定します。

（2）形式

```
void setConvertType (
     int       convertType
)
```

（3）引数

convertType（入力）

変換先のコンテンツ種別を指定します。次に示す定数のどちらかを指定できます。

- DbjDef.CONTENTTYPE_REFERENCE
  シングルファイル文書をリファレンスファイル文書に変換する
- DbjDef.CONTENTTYPE_CONTENT
  リファレンスファイル文書をシングルファイル文書に変換する

（4）戻り値

なし

（5）例外

なし

5.5.12 setExecuteMode（変換モードの設定）

(1) 機能

変換する文書の変換元レンディションを残すかどうかの変換モードを設定します。

(2) 形式

```
void setExecuteMode(
     int       executeMode
)
```

(3) 引数

executeMode（入力）

変換する文書の変換元レンディションを残すかどうかの変換モードを指定します。次に示す定数のどちらかを指定できます。

- DbjConvertDef.EXECUTE_REPLACE
  変換元のレンディションを残さない
- DbjConvertDef.EXECUTE_VERBOSE
  変換元のレンディションを残す

(4) 戻り値

なし

(5) 例外

なし

5.5.13 setSourceScope（変換対象レンディション範囲の設定）

（1）機能

変換対象とするレンディションの範囲を設定します。

（2）形式

```
void setsourceScope (
     int       sourceScope
)
```

（3）引数

sourceScope（入力）

変換対象とするレンディションの範囲を指定します。次に示す定数のどちらかを指定できます。

- DbjConvertDef.SOURCESCOPE_ALL
  すべてのレンディションを変換対象とする
- DbjConvertDef.SOURCESCOPE_SEPARATE
  renditionType で指定したレンディションを変換対象とする

（4）戻り値

なし

（5）例外

なし

5.5.14 setRenditionType（変換対象レンディションタイプの設定）

（1） 機能

sourceScope プロパティに DbjConvertDef.SOURCESCOPE_SEPARATE（renditionType プロパティで指定したレンディションを変換対象とする）を設定した場合，このプロパティに変換の対象とするレンディションタイプを設定します。

（2） 形式

```
void setRenditionType (
     String      renditionType
)
```

（3） 引数

renditionType（入力）

変換対象のレンディションタイプを指定します。レンディションタイプにマスタレンディションを指定する場合は，null を指定します。

空文字列（""）や存在しないレンディションタイプを指定した場合は，エラーとなります。

（4） 戻り値

なし

（5） 例外

なし

5.5.15 setInvestMode（コメント付与方式の設定）

（1）機能

コメントの付与方式を設定します。

（2）形式

```
void setInvestMode(
     int       investMode
)
```

（3）引数

investMode（入力）

コメントの付与方式を指定します。次に示す定数のどちらかを指定できます。

- DbjConvertDef.INVESTMODE_APPEND
  レンディションタイプの末尾にコメントを追加する
- DbjConvertDef.INVESTMODE_REPLACE
  レンディションタイプの末尾に存在するコメントを指定したコメントで置き換える（複数のコメントが存在する場合は，末尾のコメントを置き換えます。コメントがない場合は，レンディションタイプの末尾にコメントを追加します。）

（4）戻り値

なし

（5）例外

なし

5.5.16 setRenditionComment（レンディションコメントの設定）

(1) 機能

レンディションに付与するコメントを設定します。

コメントの指定には，コンテンツタイプを表す文字列を指定することを推奨します。レンディションタイプとコメントの間には1バイトのスペースが設定されます。

(2) 形式

```
void setRenditionComment (
     String      renditionComment
)
```

(3) 引数

renditionComment（入力）

レンディションに付与するコメントを指定します。
null を指定した場合は，DocumentBroker が自動的にコメント（"edms-cnv"）を付与します。空文字列（""）は指定できません。
指定するコメントは，変換するレンディションのレンディションタイプの文字列長とレンディションコメントの文字列長を合わせて 255 バイト以内で指定してください。255 バイトを超えた場合は，エラーとなります。
なお，レンディションコメントには，印刷可能な ASCII コードで記述できる値を指定してください。

(4) 戻り値

なし

(5) 例外

なし

5.5.17 setReferenceTargetPath（コンテンツ登録先ディレクトリパスの設定）

(1) 機能

コンテンツの登録先となるサーバマシン上のディレクトリパスを設定します。

(2) 形式

```
void setReferenceTargetPath (
     String      referenceTargetPath
)
```

(3) 引数

referenceTargetPath（入力）

コンテンツの登録先となるサーバマシン上のディレクトリパスを指定します。

(4) 戻り値

なし

(5) 例外

なし

5.5.18 setChangeMaster（変換後のレンディション種別設定）

（1）機能

executeMode プロパティに DbjConvertDef.EXECUTE_VERBOSE（変換元のレンディションを残す）を設定してマスタレンディションを変換する場合に，変換後のレンディションをマスタレンディションに変更するかどうかを設定できます。

（2）形式

```
void setChangeMaster (
     boolean      changeMaster
)
```

（3）引数

changeMaster（入力）

変換後のレンディションをマスタレンディションに変更するかどうかを指定します。次に示す boolean 型の定数のどちらかを指定できます。

- true
  変換後のレンディションをマスタレンディションに変更する
- false
  変換後のレンディションをマスタレンディションに変更しない

（4）戻り値

なし

（5）例外

なし

5.5.19 setCheckRenditionStatus（レンディションステータスのチェックの設定）

（1）機能

変換対象のレンディションステータスをチェックするかどうかを設定します。

（2）形式

```
void setCheckRenditionStatus (
      boolean      checkRenditionStatus
)
```

（3）引数

checkRenditionStatus（入力）

変換対象のレンディションステータスをチェックするかどうかを指定します。次に示す boolean 型の定数のどちらかを指定できます。

- true
  変換対象のレンディションステータスをチェックする
- false
  変換対象のレンディションステータスをチェックしない

（4）戻り値

なし

（5）例外

なし

5.5.20 setInvestSourceComment（レンディションコメントの付与箇所の設定）

(1) 機能

setrenditionComment で設定するレンディションコメントを変換前のレンディションに付与するか，変換後のレンディションに付与するかを設定します。

(2) 形式

```
void setInvestSourceComment (
      boolean      investSourceComment
)
```

(3) 引数

investSourceComment（入力）

レンディションコメントを変換前のレンディションに付与するか，変換後のレンディションに付与するかを指定します。次に示す boolean 型の定数のどちらかを指定できます。

- true
  変換前のレンディションにコメントを付与する
- false
  変換後のレンディションにコメントを付与する

(4) 戻り値

なし

(5) 例外

なし

5.6 DbjFetchInfo インターフェース

DbjFetchInfo インターフェースは，検索結果取得情報を扱うインターフェースです。検索結果の取得開始位置，取得件数および最大取得件数などを取得，設定するメソッドを提供します。また，キャッシュ検索の実行時に使用するキャッシュ名およびキャッシュキーを取得，設定できます。

キャッシュ検索を実行すると検索結果がキャッシュされ，以降の検索では検索結果キャッシュから検索結果を取得できます。検索結果キャッシュは，同一セッションの同一キャッシュ名に対して有効です。

直接のスーパーインターフェース

- java.lang.Cloneable
  DbjFetchInfo インターフェースの clone メソッドは，ディープコピーを実行します。

プロパティ一覧

DbjFetchInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-5 DbjFetchInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
cacheKey	int 型	getCacheKey	setCacheKey
cacheName	String 型	getCacheName	setCacheName
cacheTotal	int 型	getCacheTotal	setCacheTotal
comparator	Comparator	getComparator	setComparator
fetchCount	int 型	getFetchCount	setFetchCount
maxFetchCount	int 型	getMaxFetchCount	setMaxFetchCount
startIndex	int 型	getStartIndex	setStartIndex

プロパティ詳細

- cacheKey プロパティ
  キャッシュ検索の実行時に，検索結果キャッシュの整合性をチェックするためのキーです。キャッシュキーは INT 型の値で，再検索を実行して検索結果キャッシュが更新されるごとにインクリメントされます。キャッシュ検索を実行しない時，またはキャッシュ検索の初回実行時は無効です。
  指定したキャッシュキーと検索結果キャッシュに保持されているキーが一致した場合は，データベースへはアクセスしないで，検索結果キャッシュから検索結果を取得します。キャッシュキーが不一致の場合は，データベースを再検索して検索結果を取得します。このとき，検索結果キャッシュは更新されます。
  キャッシュ検索を実行しない場合（データベースを再検索する場合）は，キャッシュキーに DbjDef.INITIAL_KEY を指定してください。
  初期値は DbjDef.INITIAL_KEY です。
- cacheName プロパティ
  検索結果のキャッシュ名を表します。キャッシュ検索を実行する場合に指定します。null または「""」（空文字列）を指定すると，キャッシュ検索を実行しません（検索結果はキャッシュされません）。
  初期値は null です。
- cacheTotal プロパティ
  キャッシュ検索の実行時に設定される検索結果キャッシュの全件数を表します。

初期値は 0 です。

- comparator プロパティ
  検索結果取得時に検索結果キャッシュの内容をソートする場合に使用する，java.util.Comparator インターフェースを表します。null を指定すると，検索結果キャッシュの内容はソートされません。
  初期値は null です。
  検索結果キャッシュの内容をソートする際には，java.util.Comparator#compare メソッドを実装して，検索結果の要素を比較する必要があります。また，java.util.Comparator#equals メソッドを実装して，検索結果キャッシュに保持されているコンパレータと，今回の検索結果のコンパレータが等しいかどうかを判定する必要があります。コンパレータが等しいかどうかを判定する対象には，異なるインスタンスのコンパレータを指定してください。
  コンパレータが等しいかどうかを判定した結果，等しい場合は，検索結果キャッシュはソートされません。等しくない場合は，検索結果キャッシュがソートされます。なお，キャッシュキーが一致している場合で，コンパレータが等しくないときは，データベースへの再検索は実行されないで，検索結果キャッシュのソートだけが実行されます。
- fetchCount プロパティ
  検索結果の取得件数（検索結果を実際に何件取得するか）を表します。最大件数を取得したい場合は，DbjDef.MAX_NUM を指定します。
  キャッシュ検索では，検索結果キャッシュに保持している件数以上は取得できません。
  初期値は DbjDef.MAX_NUM です。
- maxFetchCount プロパティ
  検索結果の最大取得件数（検索結果を最大で何件まで取得するか）を表します。全件取得したい場合は，DbjDef.MAX_NUM を指定します。
  キャッシュ検索では，検索結果キャッシュに保持する最大件数を表します。
  初期値は DbjDef.MAX_NUM です。
- startIndex プロパティ
  検索結果の取得開始位置を表します。開始位置は 0 から始まるインデクスで指定します。
  キャッシュ検索では，検索結果キャッシュ上の開始位置を表します。
  初期値は 0 です。

以降，DbjFetchInfo インターフェースのメソッドについて説明します。

5.6.1 getCacheKey（キャッシュキーの取得）

(1) 機能

キャッシュキーを取得します。

(2) 形式

```
int getCacheKey()
```

(3) 引数

なし

(4) 戻り値

cacheKey プロパティ

(5) 例外

なし

5.6.2 getCacheName（キャッシュ名の取得）

(1) 機能

キャッシュ名を取得します。

(2) 形式

```
String getCacheName()
```

(3) 引数

なし

(4) 戻り値

cacheName プロパティ

(5) 例外

なし

5.6.3 getCacheTotal（検索結果キャッシュの全件数の取得）

(1) 機能

検索結果キャッシュの全件数を取得します。

(2) 形式

```
int getCacheTotal()
```

(3) 引数

なし

(4) 戻り値

cacheTotal プロパティ

(5) 例外

なし

5.6.4 getComparator（Comparator インターフェースの取得）

(1) 機能

java.util.Comparator インターフェースを取得します。

(2) 形式

```
Comparator getComparator()
```

(3) 引数

なし

(4) 戻り値

java.util.Comparator インターフェース

(5) 例外

なし

5.6.5 getFetchCount（検索結果の取得件数の取得）

(1) 機能

検索結果の取得件数（検索結果を実際に何件取得するか）を取得します。

(2) 形式

```
int getFetchCount()
```

(3) 引数

なし

(4) 戻り値

fetchCount プロパティ

(5) 例外

なし

5.6.6 getMaxFetchCount（検索結果の最大取得件数の取得）

(1) 機能

検索結果の最大取得件数（検索結果を最大で何件まで取得するか）を取得します。

(2) 形式

```
int getMaxFetchCount()
```

(3) 引数

なし

(4) 戻り値

maxFetchCount プロパティ

(5) 例外

なし

5.6.7 getStartIndex（検索結果の取得開始位置の取得）

(1) 機能

検索結果の取得開始位置を取得します。

(2) 形式

```
int getStartIndex()
```

(3) 引数

なし

(4) 戻り値

startIndex プロパティ

(5) 例外

なし

5.6.8 setCacheKey（キャッシュキーの設定）

（1） 機能

キャッシュキーを設定します。

なお，指定した値の妥当性は検証されません。

（2） 形式

```
void setCacheKey(
     int       cacheKey
)
```

（3） 引数

cacheKey（入力）

キャッシュキーを指定します。
キャッシュ検索を実行しない場合は，DbjDef.INITIAL_KEY を指定してください。

（4） 戻り値

なし

（5） 例外

なし

5.6.9 setCacheName（キャッシュ名の設定）

（1）機能

キャッシュ名を設定します。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setCacheName(
     String      cacheName
)
```

（3）引数

cacheName（入力）

キャッシュ名を指定します。
キャッシュ検索を実行しない場合は，null または「""」（空文字列）を指定してください。

（4）戻り値

なし

（5）例外

なし

5.6.10 setCacheTotal（検索結果キャッシュの全件数の設定）

(1) 機能

検索結果キャッシュの全件数を表す cacheTotal プロパティを設定します。

(2) 形式

```
void setCacheTotal(
     int              cacheTotal
)
```

(3) 引数

cacheTotal（入力）

cacheTotal プロパティに設定する値を指定します。

(4) 戻り値

なし

(5) 例外

なし

5.6.11 setComparator（Comparator インターフェースの設定）

（1） 機能

java.util.Comparator インターフェースを設定します。

なお，指定した値の妥当性は検証されません。

（2） 形式

```
void setComparator(
     Comparator      comp
)
```

（3） 引数

comp（入力）

java.util.Comparator インターフェースを指定します。

（4） 戻り値

なし

（5） 例外

なし

5.6.12 setFetchCount（検索結果の取得件数の設定）

（1） 機能

検索結果の取得件数（検索結果を実際に何件取得するか）を設定します。

なお，指定した値の妥当性は検証されません。

（2） 形式

```
void setFetchCount(
     int       fetchCount
)
```

（3） 引数

fetchCount（入力）

検索結果の取得件数を指定します。
最大件数取得したい場合は，DbjDef.MAX_NUM を指定してください。

（4） 戻り値

なし

（5） 例外

なし

5.6.13 setMaxFetchCount（検索結果の最大取得件数の設定）

（1） 機能

検索結果の最大取得件数（検索結果を最大で何件まで取得するか）を設定します。

なお，指定した値の妥当性は検証されません。

（2） 形式

```
void setMaxFetchCount(
     int       maxFetchCount
)
```

（3） 引数

maxFetchCount（入力）

検索結果の最大取得件数を指定します。
全件取得したい場合は，DbjDef.MAX_NUM を指定してください。

（4） 戻り値

なし

（5） 例外

なし

5.6.14 setStartIndex（検索結果の取得開始位置の設定）

（1） 機能

検索結果の取得を開始する位置を設定します。

なお，指定した値の妥当性は検証されません。

（2） 形式

```
void setStartIndex(
     int       startIndex
)
```

（3） 引数

startIndex（入力）

検索結果の取得開始位置を指定します。
0 から始まるインデクスで指定してください。

（4） 戻り値

なし

（5） 例外

なし

5.7 DbjInteger32QParam インターフェース

DbjInteger32QParam インターフェースは，INT 型を表す？パラメタを扱うインターフェースです。

直接のスーパーインターフェース

- DbjQParam

5.8 DbjObjQParam インターフェース

DbjObjQParam インターフェースは，文書管理オブジェクトを表す？パラメタを扱うインターフェースです。

直接のスーパーインターフェース

- DbjQParam

5.9 DbjOIIDQParam インターフェース

DbjOIIDQParam インターフェースは，OIID 文字列を表す？パラメタを扱うインターフェースです。

直接のスーパーインターフェース

- DbjQParam

5.10 DbjPropSet インターフェース

DbjPropSet インターフェースは，プロパティ値集合を扱うインターフェースです。プロパティ値を取得，設定するメソッド，およびプロパティ名を変更するメソッドを提供します。

プロパティ値集合とは，文書管理オブジェクトのプロパティ名とそのプロパティ値を一組にした要素の集合オブジェクトです。要素のキーを文書管理オブジェクトのプロパティ名とし，要素の値をそのプロパティ値とするマップで表します。

直接のスーパーインターフェース

- java.lang.Cloneable
  DbjPropSet インターフェースの clone メソッドは，ディープコピーを実行します。
- java.util.Map
  DbjPropSet インターフェースは，java.util.Map インターフェースを継承します。したがって，プロパティ値集合にプロパティ間（要素間）の順序性はなく，プロパティ名（要素のキー）は重複しません。また，マップに対する一般的な操作を使用して，プロパティ値集合を操作することもできます。

コーディング例

```
// ユーザアプリケーションプログラムのコーディング例
// mdmProp_Nameプロパティ（STR型），
// mdmProp_Numberプロパティ（INT型）について，
// name，numberで与えられた値でプロパティ値集合を作成します。

DbjPropSet MakeProps(String name, int number)
{
// ファクトリから空プロパティ値集合を取得
  DbjPropSet propset = DbjFactory0200.getFactory().createPropSet();

// mdmProp_Nameプロパティの設定
  propset.setPropVal("mdmProp_Name", name);

// mdmProp_Numberプロパティの設定
  propset.setPropVal("mdmProp_Number", number);

  return propset;
}
```

以降，DbjPropSet インターフェースのメソッドについて説明します。

5.10.1 changePropName（プロパティ名の変更）

（1）機能

指定したプロパティのプロパティ名を変更します。

変更後のプロパティ名がすでに存在していても，例外はスローされません。ただし，プロパティ名が変更前，変更後のどちらになるかは保証できません。

なお，指定したプロパティ名がプロパティ値集合にない場合は，false が返却されます。

（2）形式

```
boolean changePropName(
     String      oldName,
     String      newName
)
```

（3）引数

oldName（入力）

変更前のプロパティ名を指定します。
null を指定した場合は，例外がスローされます。

newName（入力）

変更後のプロパティ名を指定します。
null を指定した場合は，例外がスローされます。

（4）戻り値

true

変更されました。

false

変更されませんでした。

（5）例外

NullPointerException

引数 oldName または newName に null を指定した場合

5.10.2 changePropNames（プロパティ名の一括変更）

(1) 機能

指定したマップの内容に従って，プロパティ名を一括変更します。マップは，要素のキーに変更前のプロパティ名を，要素の値に変更後のプロパティ名を設定します。

変更後のプロパティ名がすでに存在していても，例外はスローされません。ただし，プロパティ名が変更前，変更後のどちらになるかは保証できません。

なお，変更前として指定したプロパティ名がプロパティ値集合にない場合は，false が返却されます。

(2) 形式

```
boolean changePropNames(
     Map<String, String>      changeMap
)
```

(3) 引数

changeMap（入力）
　変更前と変更後のプロパティ名を設定したマップを指定します。

(4) 戻り値

true
　一つ以上変更されました。

false
　変更されませんでした。

(5) 例外

NullPointerException
　指定したマップの任意のキーまたは値に null を指定した場合

5.10.3 getIntegerVal（Integer 型でのプロパティ値の取得）

（1）機能

指定したプロパティの値を Integer 型で取得します。

指定したプロパティがない場合，またはプロパティの値が NULL 値の場合は，null が返却されます。プロパティの値が INT 型でない場合は，例外がスローされます。

（2）形式

```
Integer getIntegerVal(
     String      propName
)
```

（3）引数

propName（入力）

プロパティ名を指定します。

（4）戻り値

プロパティ値（Integer 型）

（5）例外

ClassCastException

プロパティ値が INT 型でなかった場合

NullPointerException

引数 propName に null を指定した場合

5.10.4 getIntVal（int 型でのプロパティ値の取得）

（1） 機能

指定したプロパティの値を int 型で取得します。

指定したプロパティがない場合，またはプロパティの値が NULL 値の場合は，0 が返却されます。プロパティの値が INT 型でない場合は，java.lang.Object#toString メソッドの実行結果がさらに int 型の値に変換されて返却されます。

（2） 形式

```
int getIntVal(
     String     propName
)
```

（3） 引数

propName（入力）
　プロパティ名を指定します。

（4） 戻り値

プロパティ値（int 型）

（5） 例外

NullPointerException
　引数 propName に null を指定した場合

NumberFormatException
　数値変換に失敗した場合

5.10.5 getListRef（List 型でのプロパティ値の取得）

（1） 機能

指定したプロパティの値を List 型で取得します。このメソッドは，ログインユーザのユーザ情報（dbrProp_GroupList プロパティ）の参照を取得するのに使用します。

指定したプロパティがない場合，またはプロパティの値が NULL 値の場合は，null が返却されます。プロパティの値が STRLIST 型でない場合は，例外がスローされます。

（2） 形式

```
List<?> getListRef(
     String      propName
)
```

（3） 引数

propName（入力）

プロパティ名を指定します。

（4） 戻り値

プロパティ値のリスト（List インターフェース）

（5） 例外

ClassCastException

プロパティ値が STRLIST 型でなかった場合

NullPointerException

引数 propName に null を指定した場合

5.10.6 getStringVal（String 型でのプロパティ値の取得）

（1） 機能

指定したプロパティの値を String 型で取得します。

指定したプロパティがない場合，またはプロパティの値が NULL 値の場合は，null が返却されます。プロパティの値が STR 型でない場合は，java.lang.Object#toString メソッドを実行した結果が返却されます。

（2） 形式

```
String getStringVal(
     String      propName
)
```

（3） 引数

propName（入力）

プロパティ名を指定します。

（4） 戻り値

プロパティ値（String 型）

（5） 例外

NullPointerException

引数 propName に null を指定した場合

5.10.7 getVArrayRef（VARRAY 型でのプロパティ値の参照の取得）

（1） 機能

指定した VARRAY 型プロパティの値の参照を取得します。

指定したプロパティがない場合，またはプロパティの値が NULL 値の場合は，null が返却されます。プロパティの値が VARRAY 型でない場合は，例外がスローされます。

（2） 形式

```
DbjVArray getVArrayRef(
     String      propName
)
```

（3） 引数

propName（入力）

プロパティ名を指定します。

（4） 戻り値

VARRAY 型プロパティ値の参照（DbjVArray インターフェース）

（5） 例外

ClassCastException

プロパティ値が VARRAY 型でなかった場合

NullPointerException

引数 propName に null を指定した場合

5.10.8 getVArrayVal（VARRAY 型でのプロパティ値の取得）

(1) 機能

指定した VARRAY 型プロパティの値のコピーを取得します。

指定したプロパティがない場合，またはプロパティの値が NULL 値の場合は，null が返却されます。プロパティの値が VARRAY 型でない場合は，例外がスローされます。

なお，取得するプロパティ値は，次に示すメソッドの実行結果と同じです。ただし，getVArrayVal メソッドでは，プロパティ値が NULL 値の場合に null が返却されます。

```
DbjVArray getVArrayRef(propName).clone()
```

(2) 形式

```
DbjVArray getVArrayVal(
     String     propName
)
```

(3) 引数

propName（入力）

プロパティ名を指定します。

(4) 戻り値

VARRAY 型プロパティ値（DbjVArray インターフェース）

(5) 例外

ClassCastException

プロパティ値が VARRAY 型でなかった場合

NullPointerException

引数 propName に null を指定した場合

5.10.9 isNull（プロパティ値が NULL 値かどうかの判定）

（1）機能

指定したプロパティの値が NULL 値かどうかを判定します。

指定したプロパティがない場合は，true が返却されます。

（2）形式

```
boolean isNull(
     String     propName
)
```

（3）引数

propName（入力）

プロパティ名を指定します。

（4）戻り値

true

NULL 値です。または指定したプロパティがありません。

false

NULL 値以外です。

（5）例外

NullPointerException

引数 propName に null を指定した場合

5.10.10 setNull（NULL 値プロパティの設定）

（1） 機能

指定したプロパティの値に NULL 値を設定します。

同一プロパティがある場合は上書きします。

（2） 形式

```
void setNull(
     String    propName
)
```

（3） 引数

propName（入力）

プロパティ名を指定します。

（4） 戻り値

なし

（5） 例外

NullPointerException

引数 propName に null を指定した場合

5.10.11 setPropRef（プロパティ値の参照の設定）

（1）機能

指定したプロパティの値の参照を設定します。

同一プロパティがある場合は上書きします。

（2）形式

```
void setPropRef(
     String      propName,
     DbjVArray   ref
)
```

（3）引数

propName（入力）

プロパティ名を指定します。

ref（入力）

プロパティ値を指定します。null を指定できます。

（4）戻り値

なし

（5）例外

NullPointerException

引数 propName に null を指定した場合

5.10.12 setPropVal（プロパティ値の設定）

(1) 機能

指定したプロパティの値を設定します。

同一プロパティがある場合は上書きします。

形式 2 では，プロパティ値が INT 型のオブジェクトにラップされて設定されます。

形式 4 では，指定した VARRAY 型の値がコピーされて設定されます。実行結果は次に示すメソッドの実行結果と同じです。

```
setVArrayRef(propName,(DbjVArray)val.clone())
```

ただし，setPropVal メソッドの形式 4 では，val に null を指定した場合，次に示すメソッドの実行結果と同じになります。

```
setVArrayRef(propName,null)
```

(2) 形式

(a) 形式 1

```
void setPropVal(
     String     propName,
     String     val
)
```

(b) 形式 2

```
void setPropVal(
     String     propName,
     int        val
)
```

(c) 形式 3

```
void setPropVal(
     String     propName,
     Integer    val
)
```

(d) 形式 4

```
void setPropVal(
     String      propName,
     DbjVArray   val
)
```

(3) 引数

propName（入力）

プロパティ名を指定します。

val（入力）

プロパティ値を指定します。形式 1，形式 3 または形式 4 の場合は，null を指定するとプロパティに NULL 値を設定できます。

(4) 戻り値

なし

(5) 例外

NullPointerException

引数 propName に null を指定した場合

5.11 DbjPublicACLIdElm インターフェース

DbjPublicACLIdElm インターフェースは，文書管理オブジェクトにバインドするパブリック ACL の OIID 文字列を扱うインターフェースです。パブリック ACL の OIID 文字列を取得，設定するメソッドを提供します。

なお，DocumentBroker クラスライブラリでは，文書管理オブジェクトにバインドするパブリック ACL の OIID 文字列のリストを，DbjVArray インターフェースで操作できます。また，その要素であるパブリック ACL の OIID 文字列は DbjPropSet インターフェースでも操作できます。

プロパティー覧

DbjPublicACLIdElm インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-6 表 55DbjPublicACLIdElm インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
Id	String 型	getId	setId
propSet	DbjPropSet	propSet	setPropSet

プロパティ詳細

- Id プロパティ
  パブリック ACL の OIID 文字列を表します。
  このプロパティへのアクセスは，プロパティ値集合の dbrProp_PublicACLIdElem プロパティに反映されます。
  初期値は null です。
- propSet プロパティ
  DbjPublicACLIdElm インターフェースで扱うプロパティ値の集合です。
  このプロパティ値集合に対する操作は，集合に含まれる各プロパティ値に反映されます。また，各プロパティ値に対する操作もこのプロパティ値集合に反映されます。

以降，DbjPublicACLIdElm インターフェースのメソッドについて説明します。

5.11.1 getId（パブリック ACL の OIID 文字列の取得）

（1） 機能

パブリック ACL の OIID 文字列を取得します。

id プロパティに値が設定されていない場合は，null が返却されます。

（2） 形式

```
String getId()
```

（3） 引数

なし

（4） 戻り値

Id プロパティ

（5） 例外

なし

5.11.2 propSet（パブリック ACL の OIID 文字列のプロパティ値集合の取得）

(1) 機能

パブリック ACL の OIID 文字列を要素としたプロパティ値集合を取得します。

取得するプロパティ値集合の各要素は，プロパティ値の参照です。なお，値が設定されていなくても，null は返却されません。

(2) 形式

```
DbjPropSet propSet()
```

(3) 引数

なし

(4) 戻り値

propSet プロパティ（DbjPropSet インターフェース）

(5) 例外

なし

5.11.3 setId（パブリック ACL の OIID 文字列の設定）

（1）機能

パブリック ACL の OIID 文字列を設定します。

（2）形式

```
void setId(
     String     id
)
```

（3）引数

id（入力）

パブリック ACL の OIID 文字列を指定します。null を指定できます。

（4）戻り値

なし

（5）例外

なし

5.11.4 setPropSet（パブリック ACL の OIID 文字列のプロパティ値集合の設定）

（1）機能

パブリック ACL の OIID 文字列を要素としたプロパティ値集合に，指定したプロパティ値集合の内容をコピーして設定します。

（2）形式

```
void setPropSet(
     DbjPropSet      propSet
)
```

（3）引数

propSet（入力）

パブリック ACL の OIID 文字列のプロパティ値集合を指定します。

（4）戻り値

なし

（5）例外

NullPointerException

引数 propSet に null を指定した場合

5.12 DbjQParam インターフェース

DbjQParam インターフェースは，検索メソッドで指定する？パラメタリストの要素となるインターフェースのスーパーインターフェースです。

直接のスーパーインターフェース

- java.lang.Cloneable

5.12.1 getVal（? パラメタ値の取得）

(1) 機能

オブジェクトに設定されている ? パラメタの値を Object 型で取得します。

? パラメタの値が INT 型の場合は，INT 型オブジェクトの値が返却されます。値が設定されていない場合，または NULL 値の場合は，null が返却されます。

(2) 形式

```
Object getVal()
```

(3) 引数

なし

(4) 戻り値

? パラメタ値（Object 型）

(5) 例外

なし

5.13 DbjReferenceContentInfo インターフェース

DbjReferenceContentInfo インターフェースは，リファレンスファイル文書のコンテンツ情報を扱うインターフェースです。リファレンスファイル文書のコンテンツロケーションおよびリファレンス種別を取得するメソッドを提供します。

直接のスーパーインターフェース

- DbjContentInfo

プロパティ一覧

DbjReferenceContentInfo インターフェースで扱うプロパティ一覧と，各プロパティに対する getter メソッドを次の表に示します。

表 5-7 DbjReferenceContentInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド
contentLocation	String 型	getContentLocation
referenceType	int 型	getReferenceType

プロパティ詳細

- contentLocation プロパティ
  コンテンツロケーションを表します。
  文書オブジェクトの dbrProp_ContentLocation プロパティと同じ値です。
- referenceType プロパティ
  コンテンツのリファレンス種別を表します。
  文書オブジェクトの dbrProp_ReferenceType プロパティと同じ値です。

以降，DbjReferenceContentInfo インターフェースのメソッドについて説明します。

5.13.1 getContentLocation（コンテンツロケーションの取得）

(1) 機能

コンテンツロケーションを取得します。

(2) 形式

```
String getContentLocation()
```

(3) 引数

なし

(4) 戻り値

contentLocation プロパティ

(5) 例外

なし

5.13.2 getReferenceType（リファレンス種別の取得）

(1) 機能

リファレンス種別を取得します。

(2) 形式

```
int getReferenceType()
```

(3) 引数

なし

(4) 戻り値

referenceType プロパティ

(5) 例外

なし

5.14 DbjReferencePathInfo インターフェース

DbjReferencePathInfo インターフェースは，リファレンスファイル文書のパス情報を扱うインターフェースです。リファレンスファイル文書のパス情報を取得，設定するメソッドを提供します。

プロパティ一覧

DbjReferencePathInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-8 DbjReferencePathInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
contentOperateMode	int 型	getContentOperateMode	setContentOperateMode
deleteRootPath	String 型	getDeleteRootPath	setDeleteRootPath
entry	String 型	getEntry	setEntry
targetPath	String 型	getTargetPath	setTargetPath

プロパティ詳細

- contentOperateMode プロパティ
  コンテンツのパス操作モードを設定します。
  初期値は DbjDef.OPERATEMODE_NONE（コンテンツがないオブジェクトを作成する）です。
- deleteRootPath プロパティ
  リファレンスファイル文書を削除する場合に，コンテンツの削除と同時にコンテンツを格納したディレクトリを削除するときの，削除するディレクトリのルートパスを設定します。
  初期値は null（登録されているコンテンツ，および DocumentBroker がコンテンツを管理するためのディレクトリを削除する）です。
  なお，deleteRootPath プロパティの指定は，コンテンツが登録されているオブジェクトに対してだけ有効になります。
  また，deleteRootPath プロパティに指定できるのは，DbjSession#setReferencePath メソッドで指定したコンテンツ格納先ベースパスから，コンテンツ格納先ベースパスと targetPath プロパティで示されるコンテンツ格納先パスの間のパスです。コンテンツ格納先ベースパスより上位のパス，および DocumentBroker がコンテンツを管理するためのディレクトリのパスは指定できません。
- entry プロパティ
  登録するコンテンツのパスまたはダウンロード先のパスを設定します。
  初期値は null です。
  コンテンツをストリーム形式でアップロードする場合は，DbjUploadInfo の fileStream プロパティに設定してください。なお，このプロパティに値が設定されている場合は，DbjUploadInfo の fileStream プロパティの値は無視されます。
- targetPath プロパティ
  サーバ上のコンテンツの格納先となるディレクトリのパス（コンテンツ格納先パス）を設定します。
  初期値は null です。
  なお，コンテンツロケーション（dbrProp_ContentLocation プロパティ）には，targetPath プロパティで指定したパス，DocumentBroker がコンテンツを管理するためのディレクトリのパス，および entry プロパティで指定したファイル名を結合した値が格納されます。このとき，区切り文字には半角の「/」が設定されます。

以降，DbjReferencePathInfo インターフェースのメソッドについて説明します。

5.14.1 getContentOperateMode（コンテンツのパス操作モードの取得）

(1) 機能

コンテンツのパス操作モードを取得します。

(2) 形式

```
int getContentOperateMode()
```

(3) 引数

なし

(4) 戻り値

contentOperateMode プロパティ

コンテンツのパス操作モードが，次に示す定数で返却されます。

- DbjDef.OPERATEMODE_USER_RELATIVE_CONTENT
  コンテンツロケーションをコンテンツの相対パスで管理する
- DbjDef.OPERATEMODE_NONE
  コンテンツがないオブジェクトを作成する

(5) 例外

なし

5.14.2 getDeleteRootPath（削除するディレクトリのルートパスの取得）

（1）機能

リファレンスファイル文書を削除する場合に，コンテンツの削除と同時にコンテンツを格納したディレクトリを削除するときの，削除するディレクトリのルートパスを取得します。

（2）形式

```
String getDeleteRootPath()
```

（3）引数

なし

（4）戻り値

deleteRootPath プロパティ
　削除するディレクトリのルートパスが返却されます。

（5）例外

なし

5.14.3 getEntry（登録するコンテンツのパスまたはダウンロード先のパスの取得）

（1）機能

登録するコンテンツのパスまたはダウンロード先のパスを取得します。

（2）形式

```
String getEntry()
```

（3）引数

なし

（4）戻り値

entry プロパティ

登録するコンテンツのパスまたはダウンロード先のパスが返却されます。

（5）例外

なし

5.14.4 getTargetPath（コンテンツ格納先パスの取得）

(1) 機能

コンテンツ格納先パスを取得します。

(2) 形式

```
String getTargetPath()
```

(3) 引数

なし

(4) 戻り値

targetPath プロパティ
　コンテンツ格納先パスが返却されます。

(5) 例外

なし

5.14.5 setContentOperateMode（コンテンツのパス操作モードの設定）

（1）機能

コンテンツのパス操作モードを設定します。なお，指定した値の妥当性は，検証されません。

（2）形式

```
void setContentOperateMode(
                int     contentOperateMode
)
```

（3）引数

contentOperateMode（入力）

コンテンツのパス操作モードを指定します。次に示す定数のどちらかを指定できます。

- DbjDef.OPERATEMODE_NONE
  コンテンツロケーションをコンテンツの相対パスで管理する
- DbjDef.OPERATEMODE_USER_RELATIVE_CONTENT
  コンテンツロケーションがないオブジェクトを作成する

（4）戻り値

なし

（5）例外

なし

5.14.6 setDeleteRootPath（削除するディレクトリのルートパスの設定）

(1) 機能

リファレンスファイル文書を削除する場合に，コンテンツの削除と同時にコンテンツを格納したディレクトリを削除するときの，削除するディレクトリのルートパスを設定します。なお，指定した値の妥当性は，検証されません。

(2) 形式

```
void setDeleteRootPath(
                String  deleteRootPath
)
```

(3) 引数

deleteRootPath（入力）

リファレンスファイル文書を削除する場合に，コンテンツの削除と同時にコンテンツを格納したディレクトリを削除するときの，削除するディレクトリのルートパスを指定します。
削除の対象となるディレクトリは，コンテンツロケーション（dbrProp_ContentLocation プロパティ）で示されるパス上の格納ディレクトリから，deleteRootPath プロパティで指定したディレクトリの下位までの間にあるディレクトリです。最下位層のディレクトリから順に削除し，エラーとなった時点で処理を終了します。
deleteRootPath プロパティに指定したパスがコンテンツロケーションのパスと一致していない場合はエラーになります。
null を指定した場合，登録されているコンテンツ，および DocumentBroker がコンテンツを管理するためのディレクトリが削除されます。
なお，deleteRootPath プロパティの指定は，コンテンツが登録されているオブジェクトに対してだけ有効になります。
また，deleteRootPath プロパティに指定できるのは，DbjSession#setReferencePath メソッドで指定したコンテンツ格納先ベースパスから，コンテンツ格納先ベースパスと targetPath プロパティで示されるコンテンツ格納先パスの間のパスです。コンテンツ格納先ベースパスより上位のパス，および DocumentBroker がコンテンツを管理するためのディレクトリのパスは指定できません。

(4) 戻り値

なし

(5) 例外

なし

5.14.7 setEntry（登録するコンテンツのパスまたはダウンロード先のパスの設定）

（1）機能

登録するコンテンツのパスまたはダウンロード先のパスを設定します。なお，指定した値の妥当性は，検証されません。

（2）形式

```
void setEntry (
                String  entry
)
```

（3）引数

entry（入力）

登録するコンテンツのパスまたはダウンロード先のパスを指定します。

（4）戻り値

なし

（5）例外

なし

5.14.8 setTargetPath（コンテンツ格納先パスの設定）

（1） 機能

コンテンツ格納先パスを設定します。なお，指定した値の妥当性は，検証されません。

（2） 形式

```
void setTargetPath(
                String  targetPath
)
```

（3） 引数

targetPath（入力）

コンテンツ格納先パスを指定します。
コンテンツロケーション（dbrProp_ContentLocation プロパティ）には，targetPath プロパティで指定したパス，DocumentBroker がコンテンツを管理するためのディレクトリのパス，および entry プロパティで指定したファイル名を結合した値が格納されます。このとき，区切り文字には半角の「/」が設定されます。

（4） 戻り値

なし

（5） 例外

なし

5.15 DbjReferenceUploadInfo インターフェース

DbjReferenceUploadInfo インターフェースは，リファレンスファイル文書のアップロード情報を扱うインターフェースです。リファレンスファイル文書のアップロード情報を取得，設定するメソッドを提供します。

直接のスーパーインターフェース

- DbjUploadInfo

プロパティ一覧

DbjReferenceUploadInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-9 DbjReferenceUploadInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
referencePathInfo	DbjReferencePathInfo	getReferencePathInfo	setReferencePathInfo

プロパティ詳細

- referencePathInfo プロパティ
  リファレンスファイル文書のパス情報を設定します。

DbjUploadInfo インターフェースから継承するプロパティ詳細

- filePath プロパティ
  このインターフェースでは使用しません。設定した場合，設定値は無視されます。
- indexPath プロパティ
  登録するリファレンスファイル文書の全文検索インデクスを作成するファイルのフルパスを表します。
- renditionPropSet プロパティ
  登録するリファレンスファイル文書のレンディションプロパティ値集合です。
- renditionType プロパティ
  登録するリファレンスファイル文書のレンディションタイプを表します。
- retrievalName プロパティ
  登録するリファレンスファイル文書のクライアント側でのローカルファイル名を表します。
- fileStream プロパティ
  登録する文書の java.io パッケージで提供された入力ストリームを表します。
- indexStream プロパティ
  登録する文書の全文検索インデクスの java.io パッケージで提供された入力ストリームを表します。

以降，DbjReferenceUploadInfo インターフェースのメソッドについて説明します。

5.15.1 getReferencePathInfo（リファレンスファイル文書のパス情報の取得）

(1) 機能

リファレンスファイル文書のパス情報を取得します。

(2) 形式

```
DbjReferencePathInfo getReferencePathInfo()
```

(3) 引数

なし

(4) 戻り値

referencePathInfo プロパティ

リファレンスファイル文書のパス情報が返却されます。

(5) 例外

なし

5.15.2 setReferencePathInfo（リファレンスファイル文書のパス情報の設定）

（1）機能

リファレンスファイル文書のパス情報を設定します。なお，指定した値の妥当性は，検証されません。

（2）形式

```
void setReferencePathInfo(
                        DbjReferencePathInfo referencePathInfo
)
```

（3）引数

referencePathInfo（入力）

リファレンスファイル文書のパス情報を指定します。

（4）戻り値

なし

（5）例外

なし

5.16 DbjRenditionInfo インターフェース

DbjRenditionInfo インターフェースは，レンディション情報を扱うインターフェースです。レンディションタイプを取得するメソッドを提供します。

プロパティ一覧

DbjRenditionInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドを次の表に示します。

表 5-10 DbjRenditionInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド
propSet	DbjPropSet	propSet
renditionType	String 型	getRenditionType

プロパティ詳細

- propSet プロパティ
  オブジェクトに設定されているレンディションのプロパティ値集合です。
- renditionType プロパティ
  レンディションタイプを表します。

以降，DbjRenditionInfo インターフェースのメソッドについて説明します。

5.16.1 getRenditionType（レンディションタイプの取得）

(1) 機能

レンディションタイプを取得します。

(2) 形式

```
String getRenditionType()
```

(3) 引数

なし

(4) 戻り値

renditionType プロパティ

レンディションタイプが返却されます。

(5) 例外

なし

5.16.2 propSet（レンディションのプロパティ値集合の取得）

(1) 機能

レンディションのプロパティ値集合を取得します。

取得するプロパティ値集合の各要素は，プロパティ値の参照です。

(2) 形式

```
DbjPropSet propSet()
```

(3) 引数

なし

(4) 戻り値

propSet プロパティ（DbjPropSet インターフェース）
　レンディションのプロパティ値集合が返却されます。

(5) 例外

なし

5.17 DbjRenditionList インターフェース

DbjRenditionList インターフェースは，レンディション情報のリストを扱うインターフェースです。リストの要素であるレンディション情報（DbjRenditionInfo インターフェース）を取得するメソッドを提供します。また，各レンディション情報からレンディションタイプのリストを取得するメソッドも提供します。

直接のスーパーインターフェース

- java.util.List
  DbjRenditionList インターフェースは java.util.List インターフェースを継承します。リスト中の各要素は DbjRenditionInfo インターフェースです。

以降，DbjRenditionList インターフェースのメソッドについて説明します。

5.17.1 getRenditionInfo（レンディション情報の取得）

（1） 機能

指定した要素の DbjRenditionInfo インターフェースを取得します。

指定した要素は java.util.List#get メソッドでも取得できますが，この getRenditionInfo メソッドでは，ユーザアプリケーションプログラムがダウンキャストしないで要素を取得できます。

（2） 形式

```
DbjRenditionInfo getRenditionInfo(
     int     index
)
```

（3） 引数

index（入力）

レンディション情報を取得する要素のインデクスを指定します。
0 から始まるインデクスで指定してください。

（4） 戻り値

指定した要素のレンディション情報（DbjRenditionInfo インターフェース）

（5） 例外

ClassCastException

要素が DbjRenditionInfo インターフェースでなかった場合

IndexOutOfBoundsException

インデクスが不正の場合

5.17.2 getRenditionTypeList（レンディションタイプのリストの取得）

（1）機能

オブジェクトが保持するすべての要素のレンディションタイプを List 型で取得します。

リストが空の場合は，空リストが返却されます。

（2）形式

```
List<String> getRenditionTypeList()
```

（3）引数

なし

（4）戻り値

レンディションタイプのリスト（List 型（各要素は String 型））

（5）例外

なし

5.18 DbjResultSet インターフェース

DbjResultSet インターフェースは，検索結果集合を扱うインターフェースです。検索結果集合の情報を扱うメソッド，指定した行または列のデータを取得するメソッド，カーソルを操作するメソッドなどを提供します。

検索結果集合とは，要素にプロパティ値を保持する行と列の 2 次元データに，メタデータを付けたものです。メタデータとは，列のプロパティ値のデータ型と列名の集合のことです。DbjResultSet インターフェースは，検索結果集合の 2 次元データを，各行を要素とするリストとして扱います。また，この各行データを，各列の値を要素とするリストとして扱います。

列名をメタデータに含む検索結果集合を名前付き検索結果といい，列名をメタデータに含まない検索結果集合を名前なし検索結果といいます。ユーザアプリケーションプログラムで実行する検索メソッドの形式によって，取得する検索結果集合が名前付きか名前なしかが決まります。名前付き検索結果は，検索結果集合をプロパティ値集合にマッピングする際に利用できます。また，列名を指定して名前が重複する列を削除したり，列名の付いていない列だけを削除したりできます。

検索結果集合に対するデータのアクセスには，カーソルを使用します。このカーソルの初期位置は，検索結果集合の先頭行の前です。したがって，ある行のデータを取得する際には，まず DbjResultSet#next メソッドなどを実行して，カーソルを移動する必要があります。

直接のスーパーインターフェース

- java.util.List

以降，DbjResultSet インターフェースのメソッドについて説明します。

5.18.1 absolute（カーソルの絶対指定行への移動）

(1) 機能

検索結果集合のカーソルを絶対指定行に移動します。

(2) 形式

```
boolean absolute(
     int      rowIndex
)
```

(3) 引数

rowIndex（入力）

カーソルの移動先を行インデクスで指定します。

指定できる範囲は，-1 ～ n（n は行数）です。-1 は先頭行の前を表し，n は最終行の後ろを表します。

つまり，行インデクスは次のように表せます。

- -1：先頭行の前
- 0：先頭行
- n-1：最終行
- n：最終行の後ろ

(4) 戻り値

true

移動後が有効行です。

false

移動後が有効行ではありません。カーソルは移動していません。

(5) 例外

IndexOutOfBoundsException

行インデクスが範囲を超えた場合

5.18.2 afterLast（カーソルの最終行の後ろへの移動）

(1) 機能

検索結果集合のカーソルを最終行の後ろに移動します。

(2) 形式

```
void afterLast()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

なし

5.18.3 beforeFirst（カーソルの先頭行の前への移動）

(1) 機能

検索結果集合のカーソルを先頭行の前に移動します。

(2) 形式

```
void beforeFirst()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

なし

5.18.4 distinct（列の重複の排除）

（1） 機能

名前付き検索結果から，列名が重複している列を排除します。

なお，列名が重複している列のうち，どの列が残るかは保証できません。

（2） 形式

```
void distinct()
```

（3） 引数

なし

（4） 戻り値

なし

（5） 例外

IllegalStateException

名前なし検索結果に対してメソッドを実行した場合

5.18.5 findColumnName（列インデクスの取得）

（1）機能

検索結果集合から，指定した列名の列インデクスを取得します。

指定した列名が検索結果集合にない場合は，-1 が返却されます。

なお，指定した列名が重複している場合，どの列インデクスを取得するかは保証できません。

（2）形式

```
int findColumnName(
     String     columnName
)
```

（3）引数

columnName（入力）

列名を指定します。null を指定すると，列名が付いていない列のインデクスを取得できます。

（4）戻り値

列インデクス

（5）例外

なし

5.18.6 first（カーソルの先頭行への移動）

（1） 機能

検索結果集合のカーソルを先頭行に移動します。

（2） 形式

```
boolean first()
```

（3） 引数

なし

（4） 戻り値

true
　移動後が先頭行です。

false
　検索結果の件数が 0 件のため，先頭行がありません。カーソルは移動していません。

（5） 例外

なし

5.18.7 getColumnCount（列数の取得）

(1) 機能

検索結果集合が保持する列数を取得します。

検索結果の件数が 0 件の場合は，0 が返却されます。

(2) 形式

```
int getColumnCount()
```

(3) 引数

なし

(4) 戻り値

列数

(5) 例外

なし

5.18.8 getColumnMetaName（列名の取得（表名を含む））

(1) 機能

名前付き検索結果から，指定した列インデクスの列名を取得します。取得する列名は表名を含みます。

名前なし検索結果の場合は，null が返却されます。検索結果の件数が 0 件の場合は，メタデータも空となるため，例外がスローされます。

(2) 形式

```
String getColumnMetaName(
     int     columnIndex
)
```

(3) 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

(4) 戻り値

表名を含む列名

(5) 例外

IndexOutOfBoundsException

列インデクスが範囲を超えた場合

NullPointerException

検索結果の件数が 0 件の場合

5.18.9 getColumnName（列名の取得）

（1） 機能

名前付き検索結果から，指定した列インデクスの列名を取得します。取得する列名は表名（または相関名）を含みません。

名前なし検索結果の場合は，null が返却されます。検索結果の件数が 0 件の場合は，メタデータも空となるため，例外がスローされます。

（2） 形式

```
String getColumnName(
     int     columnIndex
)
```

（3） 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

（4） 戻り値

列名

（5） 例外

IndexOutOfBoundsException

列インデクスが範囲を超えた場合

NullPointerException

検索結果の件数が 0 件の場合

5.18.10 getColumnType（列のデータ型の取得）

（1） 機能

検索結果集合から，指定した列インデクスの列のデータ型を取得します。

検索結果の件数が 0 件の場合は，メタデータも空となるため，例外がスローされます。

（2） 形式

```
int getColumnType(
     int      columnIndex
)
```

（3） 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

（4） 戻り値

列のデータ型が，次に示す定数で返却されます。

- DbjDef.DATATYPE_BOOL
  BOOL 型
- DbjDef.DATATYPE_INT
  INT 型
- DbjDef.DATATYPE_STR
  STR 型
- DbjDef.DATATYPE_VARRAY
  VARRAY 型

（5） 例外

IndexOutOfBoundsException

列インデクスが範囲を超えた場合

NullPointerException

検索結果の件数が 0 件の場合

5.18.11 getColumnVals（列データの取得）

（1） 機能

検索結果集合から，指定した列データの全行をリストで取得します。取得した List オブジェクトは新規に作成され，List オブジェクトの各要素は列データ値のコピーとなります。

データを取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データを取得するかは保証できません。

（2） 形式

（a） 形式 1

```
List<Object> getColumnVals(
     int     columnIndex
)
```

（b） 形式 2

```
List<Object> getColumnVals(
     String     columnName
)
```

（3） 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

columnName（入力）

列名を指定します。null を指定すると，列名が付いていない列のデータを取得できます。

（4） 戻り値

列データのリスト

（5） 例外

IllegalArgumentException

指定した列名がなかった場合

IndexOutOfBoundsException

列インデクスが範囲を超えた場合

5.18.12 getIntegerVal（Integer 型での列データの取得）

（1） 機能

検索結果集合のカーソル行から，指定した列データを Integer 型で取得します。

列値が NULL 値の場合は，null が返却されます。列値が INT 型でない場合は，例外がスローされます。

データを取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データを取得するかは保証できません。

（2） 形式

（a） 形式 1

```
Integer getIntegerVal(
     int         columnIndex
)
```

（b） 形式 2

```
Integer getIntegerVal(
     String      columnName
)
```

（3） 引数

columnIndex（入力）
　0 から始まる列インデクスを指定します。

columnName（入力）
　列名を指定します。null を指定すると，列名が付いていない列の INT 型データを取得できます。

（4） 戻り値

列データ（Integer 型）

（5） 例外

ClassCastException
　列値が INT 型でなかった場合

IllegalArgumentException
　指定した列名がなかった場合

IllegalStateException
　カーソル状態が不正の場合

IndexOutOfBoundsException
　列インデクスが不正の場合

5.18.13 getIntVal（int 型での列データの取得）

（1） 機能

検索結果集合のカーソル行から，指定した列のデータを int 型で取得します。

列値が NULL 値の場合は，0 が返却されます。列値が INT 型でない場合は，java.lang.Object#toString メソッドの実行結果がさらに int 型の値に変換されて返却されます。列値が INT 型の場合は，そのまま java.langInteger#intValue メソッドの実行結果が返却されます。

データを取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データを取得するかは保証できません。

（2） 形式

（a） 形式 1

```
int getIntVal(
     int     columnIndex
)
```

（b） 形式 2

```
int getIntVal(
     String  columnName
)
```

（3） 引数

columnIndex（入力）
　0 から始まる列インデクスを指定します。

columnName（入力）
　列名を指定します。null を指定すると，列名が付いていない列のデータを int 型で取得できます。

（4） 戻り値

列データ（int 型）

（5） 例外

IllegalArgumentException
　指定した列名がなかった場合

IllegalStateException
　カーソル状態が不正の場合

IndexOutOfBoundsException
　列インデクスが不正の場合

NumberFormatException
　数値変換に失敗した場合

5.18.14 getObjectRef（データの参照の取得）

(1) 機能

検索結果集合のカーソル行から，指定した列の Object 型データの値の参照を取得します。列値が NULL 値の場合は，null が返却されます。

データの参照を取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データの参照を取得するかは保証できません。

(2) 形式

(a) 形式 1

```
Object getObjectRef(
     int      columnIndex
)
```

(b) 形式 2

```
Object getObjectRef(
     String   columnName
)
```

(3) 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

columnName（入力）

列名を指定します。null を指定すると，列名が付いていない列のデータの参照を取得できます。

(4) 戻り値

データ（Object 型）

(5) 例外

IllegalArgumentException

指定した列名がなかった場合

IllegalStateException

カーソル状態が不正の場合

IndexOutOfBoundsException

列インデクスが不正の場合

5.18.15 getObjectVal（データの取得）

（1） 機能

検索結果集合のカーソル行から，指定した列の Object 型データの値のコピーを取得します。列値が NULL 値の場合は，null が返却されます。

データを取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データを取得するかは保証できません。

（2） 形式

（a） 形式 1

```
Object getObjectVal(
     int        columnIndex
)
```

（b） 形式 2

```
Object getObjectVal(
     String     columnName
)
```

（3） 引数

columnIndex（入力）
　0 から始まる列インデクスを指定します。

columnName（入力）
　列名を指定します。null を指定すると，列名が付いていない列のデータ値のコピーを取得できます。

（4） 戻り値

データ（Object 型）

（5） 例外

IllegalArgumentException
　指定した列名がなかった場合

IllegalStateException
　カーソル状態が不正の場合

IndexOutOfBoundsException
　列インデクスが不正の場合

5.18.16 getPropSet（行データをプロパティ値集合として取得）

（1） 機能

名前付き検索結果から，指定した行のデータをプロパティ値集合として取得します。取得したプロパティ値集合は新規に作成され，値がコピーされます。取得後のプロパティ値集合のプロパティ名は，列名の「.」（ピリオド）以前の文字列が削除されます。

なお，指定した行のデータのうち，列名が null の列のデータは取得しません。

形式 1，3 はカーソル行が対象となり，形式 2，4 はインデクスで行を指定します。

形式 1，2 はすべての列が対象となりますが，形式 3，4 は指定した表名（または相関名）に関連する列名だけを対象とします。なお，形式 3，4 では，列名が「表名 . 列名」の形式で設定されている必要があります。

（2） 形式

（a） 形式 1

```
DbjPropSet getPropSet()
```

（b） 形式 2

```
DbjPropSet getPropSet(
     int        rowIndex
)
```

（c） 形式 3

```
DbjPropSet getPropSet(
     String     tableName
)
```

（d） 形式 4

```
DbjPropSet getPropSet(
     String     tableName,
     int        rowIndex
)
```

（3） 引数

rowIndex（入力）

0 から始まる行インデクスを指定します。

tableName（入力）

対象表名を指定します。null を指定すると，表指定なしとみなされます。

（4） 戻り値

行データ（DbjPropSet インターフェース）

（5） 例外

IllegalStateException

名前付き検索結果でない場合，またはカーソル状態が不正の場合

IndexOutOfBoundsException

行インデクスが不正の場合

5.18.17 getRow（カーソル行インデクスの取得）

(1) 機能

検索結果集合から，カーソル行のインデクスを取得します。

(2) 形式

```
int getRow()
```

(3) 引数

なし

(4) 戻り値

カーソル行のインデクス

(5) 例外

なし

5.18.18 getRowCount（行数の取得）

(1) 機能

検索結果集合が保持する行数を取得します。

(2) 形式

```
int getRowCount()
```

(3) 引数

なし

(4) 戻り値

行数

(5) 例外

なし

5.18.19 getRowVals（行データの取得）

（1） 機能

検索結果集合から，指定した行のデータをリストで取得します。各要素は列データの値となります。取得したリストは新規に作成されます。

形式 1 はカーソル行のデータを取得し，形式 2 はインデクスで指定した行のデータを取得します。

（2） 形式

（a） 形式 1

```
List<Object> getRowVals()
```

（b） 形式 2

```
List<Object> getRowVals(
     int       rowIndex
)
```

（3） 引数

rowIndex（入力）

0 から始まる行インデクスを指定します。

（4） 戻り値

行データのリスト

（5） 例外

IllegalStateException

カーソル状態が不正の場合

IndexOutOfBoundsException

行インデクスが不正の場合

5.18.20 getStringVal（String 型での列データの取得）

（1） 機能

検索結果集合のカーソル行から，指定した列のデータを String 型で取得します。列データが NULL 値の場合は，null が返却されます。列データが STR 型でない場合は，java.lang.Object#toString メソッドを実行した結果が返却されます。

データを取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データを取得するかは保証できません。

（2） 形式

（a） 形式 1

```
String getStringVal(
     int        columnIndex
)
```

（b） 形式 2

```
String getStringVal(
     String     columnName
)
```

（3） 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

columnName（入力）

列名を指定します。null を指定すると，列名が付いていない列のデータを String 型で取得できます。

（4） 戻り値

列データ（String 型）

（5） 例外

IllegalArgumentException

指定した列名がなかった場合

IllegalStateException

カーソル状態が不正の場合

IndexOutOfBoundsException

列インデクスが不正の場合

5.18.21 getTableName（表名の取得）

（1）機能

名前付き検索結果から，指定した列の表名（または相関名）を取得します。

指定した列の列名が表名を含まない場合は，null が返却されます。検索結果の件数が 0 件の場合は，メタデータも空となるため，例外がスローされます。

（2）形式

```
String getTableName(
     int      columnIndex
)
```

（3）引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

（4）戻り値

表名

（5）例外

IllegalStateException

検索結果の件数が 0 件の場合

IndexOutOfBoundsException

列インデクスが範囲を超えた場合

5.18.22 getVArrayRef（VARRAY 型でのデータの参照の取得）

（1） 機能

検索結果集合のカーソル行から，指定した列の VARRAY 型データの参照を取得します。列値が NULL 値の場合は，null が返却されます。列値が VARRAY 型でない場合は，例外がスローされます。

データの参照を取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列のデータの参照を取得するかは保証できません。

（2） 形式

（a） 形式 1

```
DbjVArray getVArrayRef(
     int        columnIndex
)
```

（b） 形式 2

```
DbjVArray getVArrayRef(
     String     columnName
)
```

（3） 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

columnName（入力）

列名を指定します。null を指定すると，列名が付いていない列の VARRAY 型データの参照を取得できます。

（4） 戻り値

VARRAY 型データ（DbjVArray インターフェース）

（5） 例外

ClassCastException

列値が VARRAY 型でなかった場合

IllegalArgumentException

指定した列名がなかった場合

IllegalStateException

カーソル状態が不正の場合

IndexOutOfBoundsException

列インデクスが不正の場合

5.18.23 getVArrayVal（VARRAY 型でのデータの取得）

（1） 機能

検索結果集合のカーソル行から，指定した列の VARRAY 型データのコピーを取得します。列データが NULL 値の場合は，null が返却されます。列データが VARRAY 型でない場合は，例外がスローされます。

データを取得する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データを取得するかは保証できません。

（2） 形式

（a） 形式 1

```
DbjVArray getVArrayVal(
     int        columnIndex
)
```

（b） 形式 2

```
DbjVArray getVArrayVal(
     String     columnName
)
```

（3） 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

columnName（入力）

列名を指定します。null を指定すると，列名が付いていない列の VARRAY 型データのコピーを取得できます。

（4） 戻り値

VARRAY 型データ（DbjVArray インターフェース）

（5） 例外

ClassCastException

列値が VARRAY 型でなかった場合

IllegalArgumentException

指定した列名がなかった場合

IllegalStateException

カーソル状態が不正の場合

IndexOutOfBoundsException

列インデクスが不正の場合

5.18.24 isAfterLast（カーソルが最終行の後ろかどうかの判定）

（1） 機能

検索結果集合のカーソルが最終行の後ろかどうかを判定します。

（2） 形式

```
boolean isAfterLast()
```

（3） 引数

なし

（4） 戻り値

true
　最終行の後ろです。または，検索結果の件数が 0 件のため最終行がありません。

false
　最終行の後ろではありません。

（5） 例外

なし

5.18.25 isBeforeFirst（カーソルが先頭行の前かどうかの判定）

（1）機能

検索結果集合のカーソルが先頭行の前かどうかを判定します。

（2）形式

```
boolean isBeforeFirst()
```

（3）引数

なし

（4）戻り値

true
　先頭行の前です。または，検索結果の件数が 0 件のため先頭行がありません。

false
　先頭行の前ではありません。

（5）例外

なし

5.18.26 isFirst（カーソルが先頭行かどうかの判定）

(1) 機能

検索結果集合のカーソルが先頭行かどうかを判定します。

(2) 形式

```
boolean isFirst()
```

(3) 引数

なし

(4) 戻り値

true
　先頭行です。

false
　先頭行ではありません。

(5) 例外

なし

5.18.27 isLast（カーソルが最終行かどうかの判定）

(1) 機能

検索結果集合のカーソルが最終行かどうかを判定します。

(2) 形式

```
boolean isLast()
```

(3) 引数

なし

(4) 戻り値

true
　最終行です。

false
　最終行ではありません。

(5) 例外

なし

5.18.28 isNamed（名前付き検索結果かどうかの判定）

(1) 機能

検索結果集合が名前付き検索結果かどうかを判定します。

(2) 形式

```
boolean isNamed()
```

(3) 引数

なし

(4) 戻り値

true
　名前付き検索結果です。

false
　名前付き検索結果ではありません。名前なし検索結果です。

(5) 例外

なし

5.18.29 isNull（NULL 値かどうかの判定）

（1） 機能

検索結果集合のカーソル行の，指定した列のデータが NULL 値かどうかを判定します。

NULL 値かどうか判定する列は，列インデクス（形式 1）または列名（形式 2）で指定します。なお，列名で指定した場合で，その列名が重複しているときは，どの列データを判定するかは保証できません。

（2） 形式

（a） 形式 1

```
boolean isNull(
     int        columnIndex
)
```

（b） 形式 2

```
boolean isNull(
     String     columnName
)
```

（3） 引数

columnIndex（入力）
　0 から始まる列インデクスを指定します。

columnName（入力）
　列名を指定します。null を指定すると，列名が付いていない列のデータが NULL 値かどうかを判定します。

（4） 戻り値

true
　NULL 値です。

false
　NULL 値ではありません。

（5） 例外

IllegalArgumentException
　指定した列名がなかった場合

IllegalStateException
　カーソル状態が不正の場合

IndexOutOfBoundsException
　列インデクスが不正の場合

5.18.30 last（カーソルの最終行への移動）

(1) 機能

検索結果集合のカーソルを最終行に移動します。

(2) 形式

```
boolean last()
```

(3) 引数

なし

(4) 戻り値

true
　移動後が最終行です。

false
　検索結果の件数が 0 件のため，最終行がありません。カーソルは移動していません。

(5) 例外

なし

5.18.31 next（カーソルの次の行への移動）

(1) 機能

検索結果集合のカーソルを次の行に移動します。

(2) 形式

```
boolean next()
```

(3) 引数

なし

(4) 戻り値

true
　移動後，末尾を越えていません。有効行です。

false
　移動後，末尾を越えました。カーソルは移動していません。

(5) 例外

なし

5.18.32 previous（カーソルの前の行への移動）

(1) 機能

検索結果集合のカーソルを前の行に移動します。

(2) 形式

```
boolean previous()
```

(3) 引数

なし

(4) 戻り値

true
　移動後が有効行です。

false
　移動後が有効行ではありません。カーソルは移動していません。

(5) 例外

なし

5.18.33 reduct（名前なし列の削除）

(1) 機能

名前付き検索結果の列から，名前の付いていない列（列名が null の列）を削除します。

(2) 形式

```
void reduct()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

IllegalStateException

名前なし検索結果に対してメソッドを実行した場合

5.18.34 setColumnMetaName（列名の設定）

(1) 機能

検索結果集合の，指定した列の列名を設定します。列名には表名（または相関名）を含めてもかまいません。列名は，ほかの列の列名と重複してもかまいませんが，できるだけ重複しないように設定することをお勧めします。

なお，名前なし検索結果に対して一つでも列名を設定すると，検索結果集合は名前付き検索結果となります。また，一度名前付き検索結果となった検索結果集合は，すべての列名を null に設定しても，名前付き検索結果のままです。つまり，すべての列名が null である検索結果集合でも，名前付き検索結果の場合があります。

(2) 形式

```
void setColumnMetaName(
     int         columnIndex,
     String      columnName
)
```

(3) 引数

columnIndex（入力）

0 から始まる列インデクスを指定します。

columnMetaName（入力）

列名を指定します。null を指定できます。

(4) 戻り値

なし

(5) 例外

IndexOutOfBoundsException

列インデクスが範囲を超えた場合

5.19 DbjSeedDocQParam インターフェース

DbjSeedDocQParam インターフェースは，種文章を表す？パラメタを扱うインターフェースです。

直接のスーパーインターフェース

- DbjQParam

5.20 DbjSetDCRLinkInfo インターフェース

DbjSetDCRLinkInfo インターフェースは，DbjObj#Link メソッドなどで設定する直接型リンクの設定情報を扱うインターフェースです。このインターフェースで扱うオブジェクトの linkType プロパティの値は，常に DbjDef.LINK_DCR となります。

直接のスーパーインターフェース

- DbjSetLinkInfo

5.21 DbjSetLinkInfo インターフェース

DbjSetLinkInfo インターフェースは，DbjObj#Link メソッドなどで設定するリンク設定情報を扱うインターフェースです。リンク種別およびリンク先オブジェクトを取得，設定するメソッドを提供します。

直接のスーパーインターフェース

- java.lang.Cloneable
  DbjSetLinkInfo インターフェースは，java.lang.Cloneable インターフェースを継承します。このインターフェースの clone メソッドは，ディープコピーを実行します。

プロパティ一覧

DbjSetLinkInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-11 DbjSetLinkInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
linkType	int 型	getLinkType	―
propSet	DbjPropSet	PropSet	setPropSet
targetObj	DbjObj	getTargetObj	setTargetObj

（凡例）
―：該当するメソッドがないことを示します。

プロパティ詳細

- linkType プロパティ
  リンク種別を表します。
- propSet プロパティ
  リンクオブジェクトのプロパティ値集合です。
- targetObj プロパティ
  リンク先オブジェクトを表します。

以降，DbjSetLinkInfo インターフェースのメソッドについて説明します。

5.21.1 getLinkType（リンク種別の取得）

(1) 機能

リンク種別を取得します。

(2) 形式

```
int getLinkType()
```

(3) 引数

なし

(4) 戻り値

リンク種別が，次に示す定数で返却されます。

- DbjDef.LINK_DCR
  直接型リンク
- DbjDef.LINK_RCR
  参照型リンク
- DbjDef.LINK_REL
  文書間リンク

(5) 例外

なし

5.21.2 getTargetObj（リンク先オブジェクトの取得）

(1) 機能

リンク先オブジェクトを取得します。

(2) 形式

```
DbjObj getTargetObj()
```

(3) 引数

なし

(4) 戻り値

targetObj プロパティ（DbjObj インターフェース）

(5) 例外

なし

5.21.3 propSet（リンクオブジェクトのプロパティ値集合の取得）

(1) 機能

リンクオブジェクトのプロパティ値集合を取得します。

取得するプロパティ値集合の各要素は，プロパティ値の参照です。なお，値が設定されていなくても，null は返却されません。

(2) 形式

```
DbjPropSet propSet()
```

(3) 引数

なし

(4) 戻り値

propSet プロパティ（DbjPropSet インターフェース）

(5) 例外

なし

5.21.4 setPropSet（リンクオブジェクトのプロパティ値集合の設定）

（1）機能

リンクオブジェクトのプロパティ値を要素としたプロパティ値集合に，指定したプロパティ値集合の内容をコピーして設定します。

（2）形式

```
void setPropSet(
     DbjPropSet     propSet
)
```

（3）引数

propSet（入力）

設定するリンクのプロパティ値集合を指定します。null を指定できます。

（4）戻り値

なし

（5）例外

なし

5.21.5 setTargetObj（リンク先オブジェクトの設定）

（1） 機能

リンク先オブジェクトを設定します。

（2） 形式

```
void setTargetObj(
     DbjObj      obj
)
```

（3） 引数

obj（入力）

リンク先となるオブジェクトを指定します。null を指定できます。

（4） 戻り値

なし

（5） 例外

なし

5.22 DbjSetRCRLinkInfo インターフェース

DbjSetRCRLinkInfo インターフェースは，DbjObj#Link メソッドなどで設定する参照型リンクの設定情報を扱うインターフェースです。このインターフェースで扱うオブジェクトの linkType プロパティの値は，常に DbjDef.LINK_RCR となります。

直接のスーパーインターフェース

- DbjSetLinkInfo

5.23 DbjSetRelLinkInfo インターフェース

DbjSetRelLinkInfo インターフェースは，DbjObj#Link メソッドなどで設定する文書間リンクの設定情報を扱うインターフェースです。このインターフェースで扱うオブジェクトの linkType プロパティの値は，常に DbjDef.LINK_REL となります。

直接のスーパーインターフェース

- DbjSetLinkInfo

5.24 DbjStringQParam インターフェース

DbjStringQParam インターフェースは，STR 型を表す？パラメタを扱うインターフェースです。

直接のスーパーインターフェース

- DbjQParam

5.25 DbjUploadInfo インターフェース

DbjUploadInfo インターフェースは，文書のアップロード情報を扱うインターフェースです。文書のアップロード情報を取得，設定するメソッドを提供します。

直接のスーパーインターフェース

- java.lang.Cloneable
  DbjUploadInfo インターフェースは，java.lang.Cloneable インターフェースを継承します。このインターフェースの clone メソッドは，ディープコピーを実行します。

プロパティー覧

DbjUploadInfo インターフェースで扱うプロパティの一覧と，各プロパティに対する getter メソッドおよび setter メソッドを次の表に示します。

表 5-12 DbjUploadInfo インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド	setter メソッド
filePath	String 型	getFilePath	setFilePath
indexPath	String 型	getIndexPath	setIndexPath
renditionPropSet	DbjPropSet	renditionPropSet	setRenditionPropSet
renditionType	String 型	getRenditionType	setRenditionType
retrievalName	String 型	getRetrievalName	setRetrievalName
fileStream	java.io.InputStream	getFileStream	setFileStream
indexStream	java.io.InputStream	getIndexStream	setIndexStream

プロパティ詳細

- filePath プロパティ
  登録する文書のローカルパス名を表します。ファイル名を含むフルパスです。
  次に示す場合には，このプロパティの値に null を指定できます。
  ・マルチレンディションを使用しない文書（サブレンディションを持たない，マスタレンディションだけの文書）を登録する場合
  ・登録する文書の入力ストリームを用いる場合（fileStream プロパティを有効とする場合）
  null を指定した場合は，登録する文書から全文検索インデクスを作成しません。indexPath プロパティは無視されます。
  初期値は null です。
- indexPath プロパティ
  登録する文書の全文検索インデクスを作成するファイルのフルパスを表します。
  filePath プロパティに指定したファイルから全文検索インデクスを作成する場合は，
  DbjDef.INDEXPATH_SAME を指定します。全文検索インデクスを作成しない場合は，null を指定します。
  初期値は null です。
- renditionPropSet プロパティ
  登録する文書のレンディションプロパティ値集合です。レンディションプロパティ値集合には，次のプロパティが指定できます。
  ・dbrProp_RetrievalName（レンディションに登録された文書のファイル名）

・dbrProp_RenditionStatus（レンディションの更新状態）
各レンディションプロパティの値に初期値を設定する場合は，null を指定します。
初期値は空プロパティ値集合です。

- renditionType プロパティ
登録する文書のレンディションタイプを表します。設定したレンディションタイプは登録文書オブジェクトの dbrProp_RenditionType プロパティに設定されます。
このプロパティ値に null を設定すると，retrievalName プロパティの値の拡張子に対応するレンディションタイプが設定されます。retrievalName プロパティの値も null の場合には，filePath プロパティの値の拡張子に対応するレンディションタイプが設定されます。この時，拡張子に対応するレンディションタイプは，レンディション定義ファイルの定義内容に従って取得されます。拡張子がレンディション定義ファイルに定義されていない場合は，レンディションタイプとして null が設定されます。なお，このプロパティ値に null を指定して，さらに retrievalName プロパティおよび filePath プロパティの値も null の場合には，レンディションタイプに null が設定されます。
初期値は null です。
- retrievalName プロパティ
登録する文書のクライアント側でのオリジナルファイル名を表します。設定したファイル名は登録文書オブジェクトの dbrProp_RetrievalName プロパティに設定されます。
filePath プロパティに指定したフルパス中のファイル名を設定したい場合は，null を指定します。
初期値は null です。
- fileStream プロパティ
登録する文書の java.io パッケージで提供された入力ストリームを表します。このプロパティを指定する場合，retrievalName プロパティに値を設定してください。filePath プロパティを指定した場合，filePath プロパティの値が有効となります。fileStream プロパティに指定した値を有効にするには，filePath プロパティに null を指定します。
次に示す場合には，このプロパティの値に null を指定できます。
・サブレンディションを追加（DbjObj#addRendition）する場合
・バージョンなし文書の作成（DbjDocSpace#createDocument）で，先頭のアップロード情報の場合
・バージョン付き文書の作成（DbjDocSpace#createVrDocument）で，先頭のアップロード情報の場合
null を指定した場合は，登録する文書から全文検索インデクスを作成しません。indexStream プロパティは無視されます。
初期値は null です。
- indexStream プロパティ
登録する文書の全文検索インデクスの java.io パッケージで提供された入力ストリームを表します。
マスタレンディション以外で指定した場合は，無視されます。
indexPath プロパティを指定した場合，indexPath プロパティの値が有効となります。
indexStream プロパティの値を有効にするには，indexPath プロパティに null を指定してください。
全文検索インデクスを作成しない場合は，null を指定します。
初期値は null です。

以降，DbjUploadInfo インターフェースのメソッドについて説明します。

5.25.1 getFilePath（登録する文書のフルパスの取得）

(1) 機能

登録する文書のフルパスを取得します。

(2) 形式

```
String getFilePath()
```

(3) 引数

なし

(4) 戻り値

filePath プロパティ

(5) 例外

なし

5.25.2 getIndexPath（全文検索インデクス作成用ファイルのフルパスの取得）

（1）機能

全文検索インデクス作成用ファイルのフルパスを取得します。

（2）形式

```
String getIndexPath()
```

（3）引数

なし

（4）戻り値

indexPath プロパティ

（5）例外

なし

5.25.3 getRenditionType（登録する文書のレンディションタイプの取得）

(1) 機能

登録する文書のレンディションタイプを取得します。

(2) 形式

```
String getRenditionType()
```

(3) 引数

なし

(4) 戻り値

renditionType プロパティ

(5) 例外

なし

5.25.4 getRetrievalName（登録する文書のファイル名の取得）

（1）機能

登録する文書のファイル名を取得します。

（2）形式

```
String getRetrievalName()
```

（3）引数

なし

（4）戻り値

retrievalName プロパティ

（5）例外

なし

5.25.5 renditionPropSet（レンディションプロパティ値集合の取得）

（1） 機能

レンディションプロパティ値集合を取得します。

取得するプロパティ値集合の各要素は，プロパティ値の参照です。なお，値が設定されていなくても，null は返却されません。

（2） 形式

```
DbjPropSet renditionPropSet()
```

（3） 引数

なし

（4） 戻り値

renditionPropSet プロパティ（DbjPropSet インターフェース）

（5） 例外

なし

5.25.6 getFileStream（fileStream プロパティの取得）

（1） 機能

登録する文書の入力ストリームを取得します。

（2） 形式

```
java.io.InputStream getFileStream()
```

（3） 引数

なし

（4） 戻り値

fileStream プロパティの値

（5） 例外

なし

5.25.7 getIndexStream（indexStream プロパティの取得）

（1） 機能

全文検索インデクス作成用ファイルの入力ストリームを取得します。

（2） 形式

```
java.io.InputStream getIndexStream()
```

（3） 引数

なし

（4） 戻り値

indexStream プロパティの値

（5） 例外

なし

5.25.8 setFilePath（登録する文書のフルパスの設定）

（1）機能

登録する文書のフルパスを設定します。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setFilePath(
     String     filePath
)
```

（3）引数

filePath（入力）

登録する文書のフルパスを指定します。

（4）戻り値

なし

（5）例外

なし

5.25.9 setIndexPath（全文検索インデクス作成用ファイルのフルパスの設定）

（1）機能

全文検索インデクス作成用ファイルのフルパスを設定します。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setIndexPath(
     String     indexPath
)
```

（3）引数

indexPath（入力）

全文検索インデクス作成用ファイルのフルパスを指定します。
filePath プロパティに指定したファイルから全文検索インデクスを作成したい場合は，DbjDef.INDEXPATH_SAME を指定します。
全文検索インデクスを作成しない場合は，null を指定します。

（4）戻り値

なし

（5）例外

なし

5.25.10 setRenditionPropSet（レンディションプロパティ値集合の設定）

（1）機能

レンディションプロパティ値集合に，指定したプロパティ値集合の内容をコピーして設定します。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setRenditionPropSet(
     DbjPropSet     renditionPropSet
)
```

（3）引数

renditionPropSet（入力）

設定するレンディションプロパティ値集合を指定します。

各レンディションプロパティの値に初期値を設定する場合は，null を指定します。

レンディションプロパティ値集合に指定できるプロパティを次に示します。

- dbrProp_RetrievalName
- dbrProp_RenditionStatus

（4）戻り値

なし

（5）例外

なし

5.25.11 setRenditionType（登録する文書のレンディションタイプの設定）

(1) 機能

登録する文書のレンディションタイプを設定します。

なお，指定した値の妥当性は検証されません。

(2) 形式

```
void setRenditionType(
     String     renditionType
)
```

(3) 引数

renditionType（入力）

レンディションタイプを指定します。
このプロパティ値に null を設定すると，filePath プロパティの値の拡張子に対応するレンディションタイプが自動的に設定されます。

(4) 戻り値

なし

(5) 例外

なし

5.25.12 setRetrievalName（登録する文書のファイル名の設定）

（1）機能

登録する文書のファイル名を設定します。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setRetrievalName(
     String      retrievalName
)
```

（3）引数

retrievalName（入力）

登録する文書のファイル名を指定します。
filePath プロパティに指定したフルパス中のファイル名を設定したい場合は，null を指定します。

（4）戻り値

なし

（5）例外

なし

5.25.13 setFileStream（fileStream プロパティの設定）

（1） 機能

登録する文書の入力ストリームを設定します。

なお，指定した値の妥当性は検証されません。

（2） 形式

```
void setFileStream(
     java.io.InputStream      fileStream
)
```

（3） 引数

fileStream（入力）

登録する文書の入力ストリームを指定します。

（4） 戻り値

なし

（5） 例外

なし

5.25.14 setIndexStream（indexStream プロパティの設定）

（1）機能

オブジェクトの indexStream プロパティに値を設定します。

なお，指定した値の妥当性は検証されません。

（2）形式

```
void setIndexStream(
     java.io.InputStream      indexStream
)
```

（3）引数

indexStream（入力）

全文検索インデクス作成用のファイルの入力ストリームを指定します。
全文検索インデクスを作成しない場合は，null を指定します。

（4）戻り値

なし

（5）例外

なし

5.26 DbjVArray インターフェース

DbjVArray インターフェースは，VARRAY 型プロパティの値である可変長配列オブジェクトを扱うインターフェースです。可変長配列オブジェクトの要素を扱うメソッド，可変長配列オブジェクトのプロパティを扱うメソッド，およびプロパティ名を変更するメソッドを提供します。

可変長配列オブジェクトは，プロパティ値集合を要素とするリストオブジェクトで，プロパティとしてメタプロパティを保持しています。可変長配列オブジェクトのメタプロパティとは，その可変長配列オブジェクトが保持するプロパティの，プロパティ名集合を表します。

可変長配列オブジェクトの任意の要素は，そのオブジェクトのメタプロパティが保持するプロパティを必ず保持します。例えば，メタプロパティがプロパティ "A"，"B"，"C" を保持する場合，任意の要素がプロパティ "A"，"B"，"C" を保持します。

また，可変長配列オブジェクトの要素は，メタプロパティが保持していないプロパティを要素ごとに保持できます。例えば，メタプロパティがプロパティ "A"，"B"，"C" を保持する場合，ある要素がプロパティ "A"，"B"，"C"，"X" を保持し，同時に別の要素がプロパティ "A"，"B"，"C"，"Y"，"Z" を保持できます。

- メタプロパティとプロパティの整合性について
  前述のメタプロパティとプロパティとの整合性は，DbjVArray インターフェースを使用して可変長配列オブジェクトの要素（プロパティ値集合）を追加する場合に保証されます。
  しかし，java.util.List インターフェースを直接使用して要素を追加した場合は，単に要素が追加されるだけとなるため，整合性は保証されません。

直接のスーパーインターフェース

- java.util.List
- java.lang.Cloneable
  DbjVArray インターフェースの clone メソッドは，ディープコピーを実行します。

コーディング例

```
// ユーザアプリケーションプログラムのコーディング例
Set<String> metaProps = new HashSet<String>();
metaProps.add("mdmProp_Name");
metaProps.add("mdmProp_Number");

DbjVArray varray = dbjfactory.createVArray(metaProps);

DbjPropSet props = dbjfactory.createPropSet();
props.setPropVal("mdmProp_Name", "muraoka");
props.setPropVal("mdmProp_Number", 0);

// 1行目の追加("muraoka", 0)
varray.addPropSet(props);

props.setPropVal("mdmProp_Name", "tanaka");
props.setPropVal("mdmProp_Number", 8);

// 2行目の追加("tanaka", 8)
varray.addPropSet(props);

// 表示
System.out.println("1st display");
for(int i = 0; i < varray.size(); i ++ ) {
System.out.print("[" + i + "]");
System.out.print(varray.propSet(i).getStringVal("mdmProp_Name") + ", ");
System.out.println(varray.propSet(i).getIntVal("mdmProp_Number"));
}
```

```
// 1行目のmdmProp_Numberプロパティ更新
props = varray.propSet(0);
props.setPropVal("mdmProp_Number", 7);

// 2行目のmdmPropNameプロパティ更新
varray.propSet(1).setPropVal("mdmProp_Name", "kanie");

// 表示
System.out.println("2nd display");
for(int i = 0; i < varray.size(); i++){
System.out.print("[" + i + "]");
System.out.print(varray.propSet(i).getStringVal("mdmProp_Name") + ",  ");
System.out.println(varray.propSet(i).getIntVal("mdmProp_Number"));
}
```

コーディング例の実行結果

```
1st display
[0]muraoka, 0
[1]tanaka, 8
2nd display
[0]muraoka, 7
[1]kanie, 8
```

以降，DbjVArray インターフェースのメソッドについて説明します。

5.26.1 addPropSet（プロパティ値集合（行）の追加）

(1) 機能

指定したプロパティ値集合を可変長配列オブジェクトの要素に追加します。

追加する際，メタプロパティに含まれているが指定したプロパティ値集合に含まれていないプロパティには，NULL 値が設定されます。一方，指定したプロパティ値集合に含まれているがメタプロパティに含まれていないプロパティは，値がそのまま追加されます。ただし，メタプロパティは変更されません。

(2) 形式

```
void addPropSet(
     DbjPropSet     propSet
)
```

(3) 引数

propSet（入力）
　プロパティ値集合を指定します。

(4) 戻り値

なし

(5) 例外

NullPointerException
　引数 propSet に null を指定した場合

5.26.2 addPropVals（プロパティ値（列）の追加）

（1）機能

可変長配列オブジェクトに，指定したリストの値をコピーして追加します。指定したプロパティ名がすでに存在していた場合は，そのプロパティの値を，指定したリストの値で上書きします。

指定したリストの要素が可変長配列の要素に対して不足していた場合は，リストの最後の要素が可変長配列オブジェクトの不足分に設定されます。

可変長配列オブジェクトが空の場合に限り，空のプロパティ値を追加できます。

メタプロパティは変更されません。

（2）形式

```
void addPropVals(
     String     propName,
     List<?>    vals
)
```

（3）引数

propName（入力）
　プロパティ名を指定します。

vals（入力）
　プロパティ値の List インターフェース（要素は可変長配列の各要素のプロパティ値）を指定します。

（4）戻り値

なし

（5）例外

IllegalArgumentException
　可変長配列要素が空でないのに空リストを指定した場合

NullPointerException
　引数 propName に null を指定した場合

5.26.3 changePropName（可変長配列オブジェクトのプロパティ名の変更）

(1) 機能

プロパティ名を変更します。

変更後のプロパティ名が可変長配列オブジェクトにすでに存在していても，例外はスローされません。ただし，プロパティ名が変更前，変更後のどちらになるかは保証できません。

また，変更前として指定したプロパティ名が可変長配列オブジェクトにない場合は，false が返却されます。

(2) 形式

```
boolean changePropName(
     String     oldName,
     String     newName
)
```

(3) 引数

oldName（入力）

変更前のプロパティ名を指定します。
null を指定した場合は，例外がスローされます。

newName（入力）

変更後のプロパティ名を指定します。
null を指定した場合は，例外がスローされます。

(4) 戻り値

true

変更されました。

false

変更されませんでした。

(5) 例外

NullPointerException

引数 oldName または newName に null を指定した場合

5.26.4 changePropNames（可変長配列オブジェクトのプロパティ名の一括変更）

(1) 機能

指定したマップの内容に従って，プロパティ名を一括変更します。マップは，要素のキーに変更前のプロパティ名を，要素の値に変更後のプロパティ名を設定します。

変更後のプロパティ名が可変長配列オブジェクトにすでに存在していても，例外はスローされません。ただし，プロパティ名が変更前，変更後のどちらになるかは保証できません。

また，変更前として指定したプロパティ名が可変長配列オブジェクトに存在しなくても，例外はスローされません。

(2) 形式

```
boolean changePropNames(
     Map<String, String>      changeMap
)
```

(3) 引数

changeMap（入力）
: 変更前と変更後のプロパティ名を設定したマップを指定します。

(4) 戻り値

true
: 一つ以上変更されました。

false
: 変更されませんでした。

(5) 例外

NullPointerException
: 指定したマップの任意のキーまたは値に null を指定した場合

5.26.5 getPropCount（メタプロパティ数の取得）

(1) 機能

可変長配列オブジェクトのメタプロパティ数を取得します。メタプロパティが設定されていない場合は，0が返却されます。

(2) 形式

```
int getPropCount()
```

(3) 引数

なし

(4) 戻り値

メタプロパティ数

(5) 例外

なし

5.26.6 getPropNameSet（メタプロパティの取得）

(1) 機能

可変長配列オブジェクトが保持するメタプロパティを取得します。戻り値は新規に作成されて返却されます。

メタプロパティが設定されていない場合は，空オブジェクトが返却されます。

(2) 形式

```
Set getPropNameSet()
```

(3) 引数

なし

(4) 戻り値

メタプロパティ（要素が String 型のプロパティ名集合）

(5) 例外

なし

5.26.7 getPropVals（可変長配列オブジェクトのプロパティ値のリストの取得）

（1）機能

可変長配列オブジェクトから，指定したプロパティの値のコピーをリストで取得し，そのインターフェースを取得します。

可変長配列オブジェクトの要素が空の場合は，空の List インターフェースを取得します。指定したプロパティ名がない場合は，null が返却されます。

（2）形式

```
List<?> getPropVals(
     String     propName
)
```

（3）引数

propName（入力）

プロパティ名を指定します。プロパティ名に null を指定した場合は，例外がスローされます。

（4）戻り値

プロパティ値のリストまたは null

（5）例外

NullPointerException

引数 propName に null を指定した場合

5.26.8 propSet（プロパティ値集合の取得）

(1) 機能

可変長配列オブジェクトから，指定した行のデータをプロパティ値集合として取得します。

(2) 形式

```
DbjPropSet propSet(
     int     rowIndex
)
```

(3) 引数

rowIndex（入力）

0 から始まる行インデクスを指定します。

指定できる範囲は，0 ～ n-1（n は行数）です。

(4) 戻り値

プロパティ値集合の参照（DbjPropSet インターフェース）

(5) 例外

IndexOutOfBoundsException

行インデクスが範囲を超えた場合

5.26.9 removeProp（プロパティの削除）

(1) 機能

可変長配列オブジェクトから，指定したプロパティを削除します。

指定したプロパティがメタプロパティに含まれている場合は，メタプロパティからも指定したプロパティを削除します。

(2) 形式

```
void removeProp(
     String     propName
)
```

(3) 引数

propName（入力）

プロパティ名を指定します。

(4) 戻り値

なし

(5) 例外

NullPointerException

引数 propName に null を指定した場合

6 文書管理クラス詳細

この章では，文書管理クラスのインターフェース，およびメソッドについて説明します。

6.1 DbjDocSpace インターフェース

DbjDocSpace インターフェースは，文書空間へのアクセスを扱うインターフェースです。なお，個別の文書管理オブジェクトへのアクセスには DbjObj インターフェースを使用し，複数の文書管理オブジェクトへのアクセスには DbjObjList インターフェースを使用します。DbjDocSpace インターフェースは，特定の文書管理オブジェクトに依存しないメソッド，および新規に文書管理オブジェクトを生成するメソッドを提供します。

文書管理オブジェクトは DbjDocSpace インターフェースのメソッドで作成できます。個別の文書管理オブジェクトにアクセスする場合は，Proxy オブジェクトのインターフェースを DbjDocSpace#createObjConnection メソッドまたは DbjDocSpace#createObjList メソッドで取得し，そのインターフェースを使用してアクセスします。

DbjDocSpace インターフェースは，接続している文書空間のメタ情報にアクセスできます。この場合，DbjDocSpace#getMeta メソッドを使用して文書空間のメタ情報にアクセスするインターフェースを取得できます。また，DbjDocSpace インターフェースの DbjDocSpace#isAccessControlMode メソッドを利用することで文書空間がアクセス制御モードで動作しているかどうかを調べることができます。

以降，DbjDocSpace インターフェースのメソッドについて説明します。

6.1.1 changeSearchACLMode（検索実行時のアクセス制御モードの変更）

（1）機能

検索（DbjDocSpace#executeSearch メソッド）の実行時のアクセス制御モードを変更します。アクセス制御モードの指定の有無によって，検索の実行時にアクセスできるオブジェクトが異なる場合があります。アクセス制御モードで動作している文書空間に対して検索を実行する場合，このメソッドを実行してアクセス制御モードなしの状態を設定して検索を実行できます。デフォルト値には，DbjDef.WITH_ACL が設定されています。なお，引数に指定する値の妥当性は検証されません。

（2）形式

```
void changeSearchACLMode(
      int             aclMode
)
```

（3）引数

aclMode（入力）

検索の実行時のアクセス制御モードを指定します。アクセス制御機能を使用している場合だけ有効です。次のどちらかの値を指定します。

- DbjDef.WITH_ACL
  アクセス制御機能付き検索モードで実行します。このモードの場合，ユーザはアクセス権がないオブジェクトを検索結果として取得できません。
- DbjDef.WITHOUT_ACL
  アクセス制御機能なし検索（アクセス権を無視した検索）モードで実行します。このモードの場合，ユーザはアクセス権がないオブジェクトでも検索結果として取得できます。

（4）戻り値

なし

（5）例外

DbjAccessControlNotSupportedException
　アクセス制御モードで動作していない場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.1.2 createDocument（バージョンなし文書の作成）

（1）機能

文書空間にバージョンなし文書を作成して，指定したリンク元フォルダにリンクを設定します。作成したバージョンなし文書の文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）を取得します。バージョンなし文書を作成する場合にバージョンなし文書を構成するクラス名，およびバージョンなし文書のプロパティの初期値を指定できます。

dbrProp_RetrievalName（レンディションに登録されたコンテンツのファイル名）プロパティを指定する方法と指定の優先順位を次の表に示します。

表 6-1 dbrProp_RetrievalName プロパティの指定方法と優先順位

dbrProp_RetrievalName プロパティの指定方法	優先順位
引数 uploadList が持つ要素の retrievalName プロパティに指定	1
引数 uploadList が持つ要素の renditionPropSet プロパティに dbrProp_RetrievalName を指定	2
引数 propSet に dbrProp_RetrievalName を指定（マスタレンディションだけ）	3

上記のいずれも指定しなかった場合，引数 uploadList の要素が持つ filePath プロパティに含まれるファイル名を設定します。

形式

```
DbjObj createDocument(
      String           className,
      DbjPropSet       propSet,
      List<? extends DbjUploadInfo>    uploadList,
      List<? extends DbjUploadInfo>    parentLinkList
)
```

（2）引数

className（入力）

作成するバージョンなし文書のクラス名を指定します。dmaClass_DocVersion クラス，またはそのサブクラスのクラス名を指定できます。null を指定するとデフォルトクラスである dmaClass_DocVersion クラスが仮定されます。

propSet（入力）

作成する文書のプロパティの初期値として設定するプロパティ値集合を指定します。プロパティの初期値を指定しない場合は null を指定します。

uploadList（入力）

要素は，DbjUploadInfo インターフェース，またはそのサブインターフェースです。作成する文書に登録する文書のアップロード情報のリストを指定します。アップロードするファイルを指定しない場合は null を指定します。

parentLinkList（入力）

要素は，DbjSetLinkInfo インターフェースのサブインターフェースです。作成する文書にリンクを設定するリンク元フォルダオブジェクト群およびリンク種別のリストを指定します。フォルダにリンクを設定しない場合は null を指定します。

(3) 戻り値

作成した文書の文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）

(4) 例外

ClassCastException
　引数 parentLinkList の要素が DbjSetLinkInfo インターフェースでなかった場合

DbjAccessControlException
　アクセス権エラーの場合

DbjACEOperationException
　指定 ACE についてのエラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
　指定プロパティ値が不正の場合

DbjIOException
　DocumentBroker クラスライブラリ固有の IO エラーの場合

DbjMasterRenditionNotSetException
　マスタレンディションのレンディションタイプが設定されていなかった場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
　指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException
　指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

IllegalArgumentException
　引数の指定が不正の場合

NullPointerException
　値が必要な引数が null の場合

6.1.3 createFolder（バージョンなしフォルダの作成）

(1) 機能

文書空間にバージョンなしフォルダを作成し，指定したリンク元フォルダにリンクを設定します。作成したバージョンなしフォルダの文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）を取得します。バージョンなしフォルダを作成する場合にバージョンなしフォルダを構成するクラス名，およびバージョンなしフォルダのプロパティの初期値を指定できます。

(2) 形式

```
DbjObj createFolder(
      String      className,
      DbjPropSet  propSet,
      List<? extends DbjSetLinkInfo>   parentLinkList
)
```

(3) 引数

className（入力）

作成するバージョンなしフォルダのクラス名を指定します。edmClass_ContainerVersion クラス，またはそのサブクラスのクラス名を指定できます。null を指定するとデフォルトクラスである edmClass_ContainerVersion クラスが仮定されます。

propSet（入力）

作成するバージョンなしフォルダのプロパティの初期値として設定するプロパティ値集合を指定します。プロパティの初期値を指定しない場合は null を指定します。

parentLinkList（入力）

要素は，DbjSetLinkInfo インターフェースのサブインターフェースです。作成するバージョンなしフォルダにリンクを設定するリンク元フォルダオブジェクト群のリストを指定します。フォルダにリンクを設定しない場合は null を指定します。

(4) 戻り値

作成したバージョンなしフォルダの文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）

(5) 例外

ClassCastException
　引数 parentLinkList の要素が DbjSetLinkInfo インターフェースでなかった場合

DbjAccessControlException
　アクセス権エラーの場合

DbjACEOperationException
　指定 ACE についてのエラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
指定プロパティ値が不正の場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException
指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

6.1.4 createIndependentData（独立データの作成）

（1）機能

文書空間に独立データを作成します。作成した独立データの文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）を取得します。独立データを作成する場合に独立データを構成するクラス名，および独立データのプロパティの初期値を指定できます。

（2）形式

```
DbjObj createIndependentData(
      String       className,
      DbjPropSet   propSet
)
```

（3）引数

className（入力）

作成する独立データのクラス名を指定します。edmClass_IndependentPersistence クラス，またはそのサブクラスのクラス名を指定できます。null を指定するとデフォルトクラスである edmClass_IndependentPersistence クラスが仮定されます。

propSet（入力）

作成する独立データのプロパティの初期値として設定するプロパティ値集合を指定します。プロパティの初期値を指定しない場合は null を指定します。

（4）戻り値

作成した独立データの文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）

（5）例外

DbjAccessControlException
　アクセス権エラーの場合

DbjACEOperationException
　指定 ACE についてのエラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
　指定プロパティ値が不正の場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
　指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException

指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

6.1.5 createLinkObjList（複数のリンクオブジェクトアクセスインターフェースの取得）

(1) 機能

複数のリンクオブジェクトにアクセスするインターフェース（DbjLinkObjList インターフェース）を取得します。このインターフェースを実装するオブジェクトは，要素を DbjLinkObj インターフェースとするリストオブジェクトです。ユーザアプリケーションプログラムは，このインターフェースを使用して，複数のリンクオブジェクトに対して一括してアクセスできます。

形式 1 では，要素数が 0 の空のリストオブジェクトである複数のリンクオブジェクトアクセスインターフェースを取得します。形式 2 では，要素が DbjLinkObj インターフェースであるコレクションインターフェースを引数（srcObjs）に指定します。各要素（DbjLinkObj インターフェース）は，リストオブジェクトの要素として設定されます。

(2) 形式

(a) 形式 1

```
DbjLinkObjList createLinkObjList()
```

(b) 形式 2

```
DbjLinkObjList createLinkObjList(
      Collection<DbjLinkObj>      srcObjs
)
```

(3) 引数

srcObjs（入力）

要素が DbjLinkObj インターフェースのコレクションインターフェースを指定します。null を指定すると例外がスローされます。

(4) 戻り値

複数のリンクオブジェクトアクセスインターフェース（DbjLinkObjList インターフェース）

(5) 例外

ClassCastException

引数 srcObjs の要素が DbjObj インターフェース以外の場合

NullPointerException

引数 srcObjs が null の場合

6.1.6 createObjConnection（文書管理オブジェクトアクセスインターフェースの取得）

（1） 機能

指定した文書管理オブジェクトにアクセスするインターフェース（DbjObj インターフェース）を取得します。アクセス対象の文書管理オブジェクトを OIID で指定します。なお，OIID の妥当性は検証されません。

このメソッドによって文書管理オブジェクトにアクセスするための Proxy オブジェクトを作成して，そのインターフェース（DbjObj インターフェース）を取得します。ユーザアプリケーションプログラムは，取得した DbjObj インターフェースを使用して対象の文書管理オブジェクトにアクセスします。ただし，このメソッドでは実際に文書管理オブジェクトに接続しません。

（2） 形式

```
DbjObj createObjConnection(
      String      oiid
)
```

（3） 引数

oiid（入力）
: アクセス対象の文書管理オブジェクトの OIID を指定します。null を指定すると例外がスローされます。

（4） 戻り値

文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）

（5） 例外

NullPointerException
: 引数 oiid が null の場合

6.1.7 createObjList（複数の文書管理オブジェクトアクセスインターフェースの取得）

（1） 機能

複数の文書管理オブジェクトにアクセスするインターフェース（DbjObjList インターフェース）を取得します。このインターフェースを実装するオブジェクトは，要素を DbjObj インターフェースとするリストオブジェクトです。ユーザアプリケーションプログラムは，このインターフェースを使用して複数の文書管理オブジェクトに対して一括してアクセスできます。

形式 1 では，要素数が 0 の空のリストオブジェクトである複数文書管理オブジェクトにアクセスするインターフェースを取得します。取得後，ユーザが List#add メソッドによって要素を追加して使用します。形式 2 では，要素が DbjObj インターフェースであるコレクションインターフェースを指定します。各要素（DbjObj インターフェース）は，リストオブジェクトの要素として設定されます。

形式 3 および形式 4 では，リストオブジェクトの各要素を検索結果集合の各行からリストオブジェクトの要素にマッピングして複数文書管理オブジェクトアクセスインターフェースを取得します。形式 3 では，OIID の入っている列を列インデクスで指定します。形式 4 では，OIID の入っている列を列名で指定します。さらに，検索結果集合で名前の付いている列（列名が null 以外のもの）は，各要素の Proxy オブジェクトのプロパティとして設定されます。名前なし検索結果集合を指定すると，列名はすべて null になります。よって，要素の Proxy オブジェクトは，すべてのプロパティは保持しません。また，OIID をプロパティとして指定した列は，対象外になります。形式 3 および形式 4 の tableName に null を指定すると，列名から表名（または相関名）部分が除かれた値がプロパティ名として設定されます。tableName に null 以外を指定すると，指定した表名（または相関名）を含む列名のプロパティだけが対象になります。

（2） 形式

（a） 形式 1

```
DbjObjList createObjList()
```

（b） 形式 2

```
DbjObjList createObjList(
     Collection<DbjObj>    srcObjs
)
```

（c） 形式 3

```
DbjObjList createObjList(
     DbjResultSet        result,
     int                 columnIndexOfOiid,
     String              tableName
)
```

（d） 形式 4

```
DbjObjList createObjList(
     DbjResultSet         result,
     String               columnNameOfOiid,
     String               tableName
)
```

（3） 引数

srcObjs（入力）

要素が DbjObj インターフェースのコレクションインターフェースを指定します。null を指定すると例外がスローされます。

result（入力）
検索結果集合の DbjResultSet インターフェースを指定します。null を指定すると例外がスローされます。

columnIndexOfOiid（入力）
検索結果集合のうち，OIID を表す列インデクスを指定します。

columnNameOfOiid（入力）
検索結果集合のうち，OIID を表す列名を指定します。null を指定すると例外がスローされます。

tableName（入力）
表名（または相関名）を指定します。

（4） 戻り値

複数の文書管理オブジェクトアクセスインターフェース（DbjObjList インターフェース）

（5） 例外

ClassCastException
引数 srcObjs の要素が DbjObj インターフェース以外の場合

IndexOutOfBoundsException
引数 columnIndexOfOiid が範囲外の場合

NullPointerException
引数 srcObjs，result，または columnNameOfOiid が null の場合

6.1.8 createPublicACL（パブリック ACL の作成）

（1）機能

文書空間にパブリック ACL を作成します。作成したパブリック ACL の文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）を取得します。パブリック ACL の作成時にパブリック ACL を構成するクラス名，およびパブリック ACL のプロパティの初期値を指定できます。また，パブリック ACL をバインドする文書管理オブジェクトも指定できます。

（2）形式

```
DbjObj createPublicACL(
     String          className,
     DbjPropSet      propSet,
     List<DbjObj>    bindObjectList
)
```

（3）引数

className（入力）

作成するパブリック ACL のクラス名を指定します。edmClass_PublicACL クラスのクラス名を指定できます。null を指定するとデフォルトクラスである edmClass_PublicACL クラスが仮定されます。

propSet（入力）

作成するパブリック ACL のプロパティの初期値として設定するプロパティ値集合を指定します。プロパティの初期値を指定しない場合は null を指定します。

bindObjectList（入力）

要素は DbjObj インターフェースです。作成するパブリック ACL にバインドする文書管理オブジェクトのリストを指定します。バインドする文書管理オブジェクトがない場合は null を指定します。

（4）戻り値

作成したパブリック ACL の文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）

（5）例外

ClassCastException
　引数 bindObjectList の要素が DbjObj インターフェースでなかった場合

DbjAccessControlException
　アクセス権エラーの場合

DbjACEOperationException
　指定 ACE についてのエラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
　指定プロパティ値が不正の場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjPublicACLAlreadyBoundException
　指定パブリック ACL はすでにバインドされていた場合

DbjPublicACLNotFoundException
　指定パブリック ACL は存在しなかった場合

DbjPublicACLOutOfRangeException
　指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.1.9 createVrDocument（バージョン付き文書の作成）

（1）機能

文書空間にバージョン付き文書を作成し，指定したリンク元フォルダにリンクを設定します。作成したバージョン付き文書の文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）を取得します。バージョン付き文書を作成する場合にバージョン付き文書を構成するクラス名，およびバージョン付き文書のプロパティの初期値を指定できます。

dbrProp_RetrievalName（レンディションに登録されたコンテンツのファイル名）プロパティを指定する方法と指定の優先順位を次の表に示します。

表 6-2　dbrProp_RetrievalName プロパティの指定方法と優先順位

dbrProp_RetrievalName プロパティの指定方法	優先順位
引数 uploadList が持つ要素の retrievalName プロパティに指定	1
引数 uploadList が持つ要素の renditionPropSet プロパティの dbrProp_RetrievalName に指定	2
引数 propSet にバージョニングオブジェクトのプロパティとして dbrProp_RetrievalName を指定（マスタレンディションだけ）	3
引数 propSet にバージョンオブジェクトのプロパティとして dbrProp_RetrievalName を指定（マスタレンディションだけ）	4

上記のいずれも指定しなかった場合，引数 uploadList の要素が持つ filePath プロパティに含まれるファイル名を設定します。

（2）形式

```
DbjObj createVrDocument(
     String         classNameVersioning,
     String         classNameVersion,
     DbjPropSet     propSet,
     List<? extends DbjUploadInfo>     uploadList,
     List<? extends DbjSetLinkInfo>     parentLinkList
)
```

（3）引数

classNameVersioning（入力）

作成するバージョン付き文書のバージョニングオブジェクトのクラス名を指定します。
dmaClass_ConfigurationHistory クラス，またはそのサブクラスのクラス名を指定できます。null を指定するとデフォルトクラスである dmaClass_ConfigurationHistory クラスが仮定されます。

classNameVersion（入力）

作成するバージョン付き文書のバージョンオブジェクトのクラス名を指定します。
dmaClass_DocVersion クラス，またはそのサブクラスのクラス名を指定できます。null を指定するとデフォルトクラスである edmClass_VersionTracedDocVersion クラスが仮定されます。

propSet（入力）

作成するバージョン付き文書のプロパティの初期値として設定するプロパティ値集合を指定します。バージョニングオブジェクトとバージョンオブジェクト（カレントバージョン）のプロパティを同時に設定できます。プロパティの初期値を指定しない場合は null を指定します。

uploadList（入力）

要素は，DbjUploadInfo インターフェースまたはそのサブインターフェースです。作成するバージョン付き文書に登録するファイル情報（アップロード情報）のリストを指定します。アップロードするファイルを指定しない場合は null を指定します。

parentLinkList（入力）

要素は，DbjSetLinkInfo インターフェースのサブインターフェースです。作成するバージョン付き文書にリンクを設定するリンク元フォルダオブジェクト群とリンク種別のリストを指定します。フォルダにリンクを設定しない場合は null を指定します。

（4） 戻り値

作成したバージョン付き文書の文書管理オブジェクトアクセスインターフェース（DbjObj インターフェース）

（5） 例外

ClassCastException

引数 parentLinkList の要素が DbjSetLinkInfo インターフェースでなかった場合

DbjAccessControlException

アクセス権エラーの場合

DbjACEOperationException

指定 ACE についてのエラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjIllegalPropValException

指定プロパティ値が不正の場合

DbjIOException

DocumentBroker クラスライブラリ固有の IO エラーの場合

DbjMasterRenditionNotSetException

マスタレンディションのレンディションタイプが設定されていなかった場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException

指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException

指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

IllegalArgumentException

引数の指定が不正の場合

NullPointerException
値が必要な引数が null の場合

6.1.10 executeSearch（検索の実行）

(1) 機能

文書空間に対して edmSQL 文による検索を実行し，検索結果集合オブジェクトインターフェース（DbjResultSet インターフェース）を取得します。

形式 1 および形式 2 では，名前なし検索結果集合を取得します。

形式 3 および形式 4 では，名前付き検索結果集合を取得します。

形式 2 および形式 4 では，ロック種別を指定できます。ロック種別のデフォルト値は，ロックなし指定（DbjDef.LOCK_NONE 指定に相当）です。

形式 3 および形式 4 の引数（eqlStatement）には，SELECT 句の項目部分に「$_」を使用して記述します。例えば，select $_ from mdmClass_Folder_c のように指定します。また，引数（selectItems）に指定した要素は，「,」（コンマ）で区切られて「$_」に置き換えられます。検索結果集合の列名は，selectItems で指定した要素になります。SELECT 句の項目指定部分は，「$_」だけ使用できます。例えば，select A,$_ from ... と指定すると例外がスローされます。なお，形式 3 および形式 4 では，取得される列数と selectItems の要素数が不一致の場合，例外がスローされます。

(2) 形式

(a) 形式 1

```
DbjResultSet executeSearch(
     String                      eqlStatement,
     List<? extends DbjQParam>   qParamList,
     DbjFetchInfo                fetchInfo
)
```

(b) 形式 2

```
DbjResultSet executeSearch(
     String                      eqlStatement,
     List<? extends DbjQParam>   qParamList,
     DbjFetchInfo                fetchInfo,
     int                         lockType
)
```

(c) 形式 3

```
DbjResultSet executeSearch(
     List<String>                selectItems,
     String                      eqlStatement,
     List<? extends DbjQParam>   qParamList,
     DbjFetchInfo                fetchInfo
)
```

(d) 形式 4

```
DbjResultSet executeSearch(
     List<String>                  selectItems,
     String                        eqlStatement,
     List<? extends DbjQParam>     qParamList,
     DbjFetchInfo                  fetchInfo,
     int                           lockType
)
```

(3) 引数

selectItems（入力）

要素のデータ型は String 型です。SELECT 句に指定する項目のリストを指定します。名前付き検索

結果集合の列名になります。null を指定すると例外がスローされます。

eqlStatement（入力）

edmSQL 文を指定します。null を指定すると例外がスローされます。

qParamList（入力）

? パラメタの値を表します。要素は，DbjQParam インターフェースのサブクラスのリストを指定します。NULL 値の場合は，要素の値に null を指定します。? パラメタの指定がない場合は，null を指定します。

fetchInfo（入出力）

検索結果取得情報の条件を指定します。全件を取得する場合，またはキャッシュ検索をしない場合は null を指定します。キャッシュ検索をすると，cacheKey プロパティと cacheTotal プロパティが設定されます。全件を取得しない場合，maxFetchCount プロパティに指定されている件数を取得した後に，comparator プロパティに指定したコンパレータによりソートされます。そのため，全件を取得する場合と，maxFetchCount プロパティを指定して検索した場合を比較すると，comparator プロパティに同じオブジェクトを指定しても結果が異なる場合があります。

lockType（入力）

検索結果集合に対して取得するロック種別を指定します。次に示す値のどれかを指定できます。

- DbjDef.LOCK_NONE
  ロックなし
- DbjDef.LOCK_READ
  read ロック
- DbjDef.LOCK_WRITE
  write ロック
- DbjDef.LOCK_READFORUPDATE
  READFORUPDATE ロック

（4） 戻り値

検索結果集合オブジェクトインターフェース（DbjResultSet インターフェース）

（5） 例外

DbjAccessControlNotSupportedException
　アクセス制御モードに対応していない場合にアクセス制御モードで検索しようとした場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

DbjIllegalCacheStartIndexException
　検索結果数より大きい値を引数 fetchInfo（DbjFetchInfo インターフェース）の startIndex プロパティに指定した場合

IllegalArgumentException
　引数が不正の場合

NullPointerException
　引数 eqlStatement が null の場合

6.1.11 getMeta（文書管理のメタ情報の取得）

（1）機能

接続している文書空間のメタ情報を取得します。

（2）形式

```
DbjMeta getMeta()
```

（3）引数

なし

（4）戻り値

メタ情報インターフェース（DbjMeta インターフェース）

（5）例外

なし

6.1.12 getSearchACLMode（検索実行時のアクセス制御モードの取得）

(1) 機能

DbjDocSpace#executeSearch メソッドの実行時のアクセス制御モードを取得します。アクセス制御モードが動作している場合だけ有効です。

(2) 形式

```
int getSearchACLMode()
```

(3) 引数

なし

(4) 戻り値

DbjDocSpace#executeSearch メソッドの実行時のアクセス制御モードとして次に示す値が返却されます。

DbjDef.WITH_ACL
　アクセス制御機能付き検索を実行します。

DbjDef.WITHOUT_ACL
　アクセス制御機能なし検索（アクセス権を無視した検索）を実行します。

(5) 例外

DbjAccessControlNotSupportedException
　アクセス制御モードで動作していない場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.1.13 isAccessControlMode（アクセス制御モードの取得）

(1) 機能

文書空間がアクセス制御モードで実行されているかどうかについての情報を取得します。

(2) 形式

```
boolean isAccessControlMode()
```

(3) 引数

なし

(4) 戻り値

true
　アクセス制御モードで実行しています。

false
　アクセス制御モードで実行していません。

(5) 例外

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.1.14 removeObjects（検索条件を指定したオブジェクトの削除）

(1) 機能

文書空間に対して edmSQL 文による検索を実行し，検索結果として取得したオブジェクトを削除します。このメソッドは，検索結果集合の 1 列目を，削除対象のオブジェクトの OIID とみなします。したがって，指定する edmSQL 文の主問い合わせの SELECT 句の第一項目には，削除対象とする表の dmaProp_OIID プロパティ（または削除対象の OIID 文字列が設定されているプロパティ）を必ず指定します。

(2) 形式

```
void removeObjects(
     String                          eqlStatement,
     List<? extends DbjQParam>       qParamList
)
```

(3) 引数

eqlStatement（入力）

edmSQL 文を指定します。null を指定すると例外がスローされます。

qParamList（入力）

? パラメタの値を表します。要素には，DbjQParam インターフェースのサブクラスのリストを指定します。NULL 値の場合は，要素の値に null を指定します。また，? パラメタの指定がない場合は，null を指定します。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException
アクセス権エラーの場合

DbjAccessControlNotSupportedException
アクセス制御モードに対応していない場合にアクセス制御モードで検索しようとした場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjLastVersionException
最後のバージョンを消そうとした場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

NullPointerException
引数 eqlStatement が null の場合

6.2 DbjLinkObj インターフェース

DbjLinkObj インターフェースは，一つのリンクオブジェクトにアクセスするインターフェースです。

以降，DbjLinkObj インターフェースのメソッドについて説明します。

6.2.1 getLinkId（リンク識別子の取得）

(1) 機能

リンク Proxy オブジェクトのアクセス対象のターゲットオブジェクトであるリンクオブジェクトのリンク識別子を取得します。

(2) 形式

```
String getLinkId()
```

(3) 引数

なし

(4) 戻り値

リンク識別子

(5) 例外

なし

6.2.2 getLinkType（リンク種別の取得）

(1) 機能

リンク Proxy オブジェクトのアクセス対象のターゲットオブジェクトであるリンクオブジェクトのリンク種別を取得します。

(2) 形式

```
int getLinkType()
```

(3) 引数

なし

(4) 戻り値

リンク種別

(5) 例外

なし

6.2.3 getOwnerObj（リンク元オブジェクトの取得）

(1) 機能

リンク Proxy オブジェクトのアクセス対象のターゲットオブジェクトであるリンクオブジェクトのリンク元オブジェクトを取得します。取得するのは，リンク元オブジェクトの Proxy オブジェクトのインターフェース（DbjObj インターフェース）です。

(2) 形式

```
DbjObj getOwnerObj()
```

(3) 引数

なし

(4) 戻り値

リンク元オブジェクトのインターフェース（DbjObj インターフェース）

(5) 例外

なし

6.2.4 getTargetObj（リンク先オブジェクトの取得）

（1）機能

リンク Proxy オブジェクトのアクセス対象のターゲットオブジェクトであるリンクオブジェクトのリンク先オブジェクトを取得します。取得するのは，リンク先オブジェクトの Proxy オブジェクトのインターフェース（DbjObj インターフェース）です。

（2）形式

```
DbjObj getTargetObj()
```

（3）引数

なし

（4）戻り値

リンク先オブジェクトのインターフェース（DbjObj インターフェース）

（5）例外

なし

6.2.5 propSet（リンク Proxy オブジェクトのプロパティ値集合インターフェースの取得）

(1) 機能

リンク Proxy オブジェクトのプロパティ値集合インターフェースを取得します。プロパティ値集合インターフェースは，参照権限で取得します。取得したインターフェースを使用して値を設定すると，Proxy オブジェクトのリンクプロパティ値も更新されます。なお，プロパティ値集合が設定されていなくても null は返却されません。

(2) 形式

```
DbjPropSet propSet()
```

(3) 引数

なし

(4) 戻り値

リンク Proxy オブジェクトのプロパティ値集合インターフェース（DbjPropSet インターフェース）

(5) 例外

なし

6.2.6 readProperties（リンクプロパティの取得）

(1) 機能

リンク Proxy オブジェクトのアクセス対象のターゲットオブジェクトであるリンクオブジェクトの指定プロパティを，各要素のリンク Proxy オブジェクトに読み込みます。

形式 1 では，リンク Proxy オブジェクトの保持しているプロパティ値集合でのすべてのプロパティをリンクオブジェクトから読み込みます。形式 2 では，読み込み対象のプロパティのコレクションを指定します。

(2) 形式

(a) 形式 1

```
void readProperties()
```

(b) 形式 2

```
void readProperties(
    Collection<String>     propDefs
)
```

(3) 引数

propDefs（入力）

要素は，リンクプロパティ名（String 型）です。取得するプロパティのプロパティ名のコレクションを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 propDefs が null の場合

6.2.7 removeObject（リンクオブジェクトの削除）

(1) 機能

リンク Proxy オブジェクトのアクセス対象のターゲットオブジェクトであるリンクオブジェクトを削除して，リンクを解除します。

(2) 形式

```
boolean removeObject()
```

(3) 引数

なし

(4) 戻り値

true
　リンクオブジェクトが削除されました。

false
　リンクオブジェクトが削除されませんでした。この値は，すでにリンクオブジェクトが削除されていた場合に返却されます。

(5) 例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.2.8 setPropSet（リンク Proxy オブジェクトのプロパティ値集合の設定）

（1）機能

リンク Proxy オブジェクトのプロパティ値集合に，指定したプロパティ値集合の内容をコピーして設定します。

（2）形式

```
void setPropSet(
     DbjPropSet     propSet
)
```

（3）引数

propSet（入力）

設定するプロパティ値集合インターフェースを指定します。null を指定するとオブジェクトのプロパティ値集合には，空値が仮定されます。

（4）戻り値

なし

（5）例外

なし

6.2.9 writeProperties（リンクプロパティ値の設定）

(1) 機能

リンク Proxy オブジェクトのアクセス対象のターゲットオブジェクトであるリンクオブジェクトのプロパティを一括して設定します。

形式 1 では，アクセス対象のターゲットオブジェクトであるリンクオブジェクトのプロパティを，指定したプロパティ値集合で設定します。形式 2 および形式 3 では，アクセス対象のターゲットオブジェクトであるリンクオブジェクトのプロパティを，リンク Proxy オブジェクトに設定されたプロパティ値集合で設定します。形式 3 では，リンク Proxy オブジェクトに設定されているプロパティのうち，アクセス対象のターゲットオブジェクトであるリンクオブジェクトに反映する対象のプロパティを限定して指定できます。

(2) 形式

(a) 形式 1

```
void writeProperties(
     DbjPropSet     propSet
)
```

(b) 形式 2

```
void writeProperties()
```

(c) 形式 3

```
void writeProperties(
     Collection<String>     propdef
)
```

(3) 引数

propSet（入力）
: リンクオブジェクトに設定するリンクプロパティ値集合を指定します。null を指定すると例外がスローされます。

propDefs（入力）
: 要素はリンクプロパティ名（String 型）です。リンクオブジェクトに設定する対象のリンクプロパティ値のコレクションを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException
: アクセス権エラーの場合

DbjDBException
: DB エラーの場合

DbjException
: DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalPropValException
: 指定プロパティ値が不正の場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

NullPointerException
引数 propSet，または propDefs が null の場合

6.3 DbjLinkObjList インターフェース

DbjLinkObjList インターフェースは，DbjLinkObj インターフェースを要素とするリストインターフェースを扱います。

スーパーインターフェース

- java.util.List

以降，DbjLinkObjList インターフェースのメソッドについて説明します。

6.3.1 getLinkIdList（ターゲットリンク識別子プロパティのリストの取得）

（1）機能

リンク Proxy オブジェクトが保持するリスト要素すべてのターゲットリンク識別子プロパティ値をリストで取得します。

（2）形式

```
List<String> getLinkIdList()
```

（3）引数

なし

（4）戻り値

ターゲットリンク識別子プロパティ値（String 型）のリスト

（5）例外

ClassCastException

リスト要素が DbjLinkObj インターフェースでなかった場合

6.3.2 getLinkObj（リスト要素の DbjLinkObj インターフェースの取得）

(1) 機能

インデクスで指定されたリスト要素の DbjLinkObj インターフェースを取得します。

(2) 形式

```
DbjLinkObj getLinkObj(
     int      index
)
```

(3) 引数

index（入力）

リスト要素のインデクスを指定します。

(4) 戻り値

リスト要素の DbjLinkObj インターフェース

(5) 例外

ClassCastException

インデクスで指定されたリスト要素が DbjLinkObj インターフェースでなかった場合

IndexOutOfBoundException

引数に指定したリスト要素のインデクスが不正の場合

6.3.3 getOwnerObjList（リンク元オブジェクトのリストの取得）

（1）機能

リンク元オブジェクトが保持するリスト要素すべてのオーナーオブジェクトのプロパティ値をリストで取得します。

（2）形式

```
DbjObjList getOwnerObjList()
```

（3）引数

なし

（4）戻り値

オーナーオブジェクトのプロパティ値のリスト（DbjObjList インターフェース）

（5）例外

ClassCastException

リスト要素が DbjLinkObj インターフェースでなかった場合

6.3.4 getTargetObjList（リンク先オブジェクトのリストの取得）

(1) 機能

リンク先オブジェクトが保持するリスト要素すべてのターゲットオブジェクトのプロパティ値をリストで取得します。

(2) 形式

```
DbjObjList getTargetObjList()
```

(3) 引数

なし

(4) 戻り値

ターゲットオブジェクトのプロパティ値のリスト（DbjObjList インターフェース）

(5) 例外

ClassCastException

リスト要素が DbjLinkObj インターフェースでなかった場合

6.3.5 readProperties（リンクプロパティ値の一括取得）

（1） 機能

リスト要素のターゲットオブジェクトであるリンクオブジェクトの指定プロパティを各要素のリンクProxy オブジェクトに一括して読み込みます。リストの要素である各 DbjLinkObj インターフェースに対して，DbjLinkObj#readProperties メソッドを実行します。

形式 1 では，Proxy オブジェクトの保持しているプロパティ値集合でのすべてのプロパティをターゲットオブジェクトであるリンクオブジェクトの指定プロパティから読み込みます。形式 2 では，読み込み対象のプロパティのコレクションを指定します。

（2） 形式

（a） 形式 1

```
void readProperties()
```

（b） 形式 2

```
void readProperties(
     Collection<String>     propDefs
)
```

（3） 引数

propDefs（入力）

要素はリンクプロパティ名（String 型）です。取得するプロパティのプロパティ名のコレクションを指定します。null を指定すると例外がスローされます。

（4） 戻り値

なし

（5） 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 propDefs が null の場合

6.3.6 removeObjects（リンクオブジェクトの一括削除）

(1) 機能

リストオブジェクト要素のアクセス対象のターゲットオブジェクトであるリンクオブジェクトを一括して削除します。リストの要素である各 DbjLinkObj インターフェースに対して，DbjLinkObj#removeObject メソッドを実行します。

(2) 形式

```
List<DbjLinkObj> removeObjects()
```

(3) 引数

なし

(4) 戻り値

削除されたリンクオブジェクトの DbjLinkObj インターフェースのリスト

(5) 例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.3.7 writeProperties（リンクプロパティ値の一括設定）

（1） 機能

リスト要素のアクセス対象のターゲットオブジェクトであるリンクオブジェクトのプロパティを一括して設定します。

形式 1 では，要素のアクセス対象のターゲットオブジェクトであるリンクオブジェクトのプロパティを，指定したプロパティ値集合で設定します。形式 2 および形式 3 では，要素のアクセス対象のターゲットオブジェクトであるリンクオブジェクトのプロパティを，各要素のリンク Proxy オブジェクトに設定されたプロパティ値集合で設定します。形式 3 では，リンク Proxy オブジェクトに設定されているプロパティのうち，ターゲットオブジェクトであるリンクオブジェクトに反映するプロパティを限定して指定できます。

（2） 形式

（a） 形式 1

```
void writeProperties(
     DbjPropSet     propSet
)
```

（b） 形式 2

```
void writeProperties()
```

（c） 形式 3

```
void writeProperties(
    Collection<String>     propdef
)
```

（3） 引数

propSet（入力）
: リンクオブジェクトに設定するリンクプロパティ値集合を指定します。null を指定すると例外がスローされます。

propDefs（入力）
: 要素はリンクプロパティ名（String 型）です。リンクオブジェクトに設定する対象のプロパティ値集合を指定します。null を指定すると例外がスローされます。

（4） 戻り値

なし

（5） 例外

DbjAccessControlException
: アクセス権エラーの場合

DbjDBException
: DB エラーの場合

DbjException
: DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalPropValException
: 指定プロパティ値が不正の場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

NullPointerException
引数 propSet，または propDefs が null の場合

6.4 DbjObj インターフェース

DbjObj インターフェースは，一つの文書管理オブジェクトへのアクセスを扱います。このインターフェースは，Proxy オブジェクトを通じて一つの文書管理オブジェクトにアクセスするメソッドを提供します。

以降，DbjObj インターフェースのメソッドについて説明します。

6.4.1 addRendition（レンディションの追加）

(1) 機能

Proxy オブジェクトのアクセス対象の文書にレンディションを追加します。すでに登録されているレンディションが指定された場合は，例外がスローされます。レンディションを追加できるアクセス対象の文書管理オブジェクトは，文書だけです。

(2) 形式

```
void addRendition(
     List<? extends DbjUploadInfo>      uploadList
)
```

(3) 引数

uploadList（入力）

要素は DbjUploadInfo インターフェースです。文書に登録するレンディションのファイル情報（アップロード情報）のリストを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjFileReferenceOperationFailedException

リファレンスファイル機能のコンテンツ操作でエラーが発生した場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjIllegalPropValException

指定レンディションプロパティ値が不正の場合

DbjIOException

DocumentBroker クラスライブラリ固有の IO エラーの場合

DbjMasterRenditionNotSetException

マスタレンディションのレンディションタイプが設定されていなかった場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjRenditionCountOutOfRangeException

レンディション数が範囲を超えた場合

DbjRenditionTypeDuplicatedException
　指定レンディションタイプが重複していた場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

IllegalArgumentException
　レンディションタイプが不正の場合

NullPointerException
　引数 uploadList が null の場合

6.4.2 bindPublicACL（パブリック ACL のバインド）

(1) 機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトに，指定したパブリック ACL をバインドします。ただし，Proxy オブジェクトのアクセス対象の文書管理オブジェクトには，パブリック ACL を指定しないでください。

(2) 形式

```
void bindPublicACL(
     List<DbjObj>     publicACLList
)
```

(3) 引数

publicACLIdList（入力）

要素は DbjObj インターフェースです。各要素には，バインドするパブリック ACL のリストを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

ClassCastException

引数 publicACLList の要素が DbjObj インターフェースでなかった場合

DbjAccessControlException

アクセス権エラーの場合

DbjAccessControlNotSupportedException

アクセス制御機能に対応していない文書空間だった場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjPublicACLAlreadyBoundException

指定パブリック ACL はすでにバインドされていた場合

DbjPublicACLNotFoundException

指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException

パブリック ACL 数が制限を超えた場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException
引数 publicACLIdList が null の場合

6.4.3 changeMasterRendition（マスタレンディションの変更）

(1) 機能

Proxy オブジェクトのアクセス対象の文書のマスタレンディションを，指定したレンディションタイプのレンディションに変更します。マスタレンディションを変更できるターゲットオブジェクトは，文書だけです。

(2) 形式

```
void changeMasterRendition(
     String       renditionType
)
```

(3) 引数

renditionType（入力）

マスタレンディションに変更するレンディションタイプを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjIsMasterRenditionException

すでにマスタレンディションだった場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjRenditionIsEmptyException

レンディションが空の場合

DbjRenditionNotConvertedException

指定レンディションが変換されていなかった場合

DbjRenditionNotFoundException

指定レンディションが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていない

NullPointerException
引数 renditionType が null の場合

6.4.4 cancelCheckOut（バージョン付きオブジェクトのチェックアウトの取り消し）

(1) 機能

Proxy オブジェクトのアクセス対象であるバージョン付きオブジェクトのチェックアウトを取り消します。チェックアウトを取り消すことができるターゲットオブジェクトは，バージョン付きオブジェクトだけです。

形式 1 および形式 2 では，取り消しと同時に Proxy オブジェクトに設定されているプロパティ値が，カレントバージョンのプロパティに設定されます。形式 2 では，設定するプロパティを限定できます（指定したプロパティだけが設定対象になります）。形式 3 では，Proxy オブジェクトのものに関係なく，指定したプロパティ値集合がカレントバージョンのプロパティを対象として設定されます。Proxy オブジェクトのターゲットバージョン識別子のプロパティは，カレントバージョン（null）になります。

(2) 形式

(a) 形式 1

```
void cancelCheckOut()
```

(b) 形式 2

```
void cancelCheckOut(
     Collection<String>     propDefs
)
```

(c) 形式 3

```
void cancelCheckOut(
     DbjPropSet     propSet
)
```

(3) 引数

propDefs（入力）

要素はプロパティ名（String 型）です。チェックアウトの取り消し後のカレントバージョンを対象としたバージョン付きオブジェクトに設定する Proxy オブジェクトのプロパティとして，どのプロパティを設定するかを指定します。null を指定するとプロパティを設定しません。

propSet（入力）

チェックアウトの取り消し後のカレントバージョンを対象としたバージョン付きオブジェクトに設定するプロパティ値集合を指定します。null を指定するとプロパティを設定しません。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjACEOperationException

指定 ACE についてのエラーの場合

DbjDBException

DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
　指定プロパティ値が不正の場合

DbjNotCheckOutException
　チェックアウトしていなかった場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
　指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException
　指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

NullPointerException
　引数 propSet が null の場合

6.4.5 checkIn（バージョン付きオブジェクトのチェックイン）

(1) 機能

Proxy オブジェクトのアクセス対象のバージョン付きオブジェクトをチェックインします。チェックインできるターゲットオブジェクトは，バージョン付きオブジェクトだけです。

形式 1 および形式 2 では，チェックインと同時に Proxy オブジェクトに設定されているプロパティ値集合が，チェックイン後のカレントバージョンを対象としてプロパティに設定されます。形式 2 では，設定するプロパティを限定できます（指定したプロパティだけが設定の対象になります）。形式 3 では，Proxy オブジェクトのものに関係なく，指定したプロパティ値集合がカレントバージョンを対象として設定されます。Proxy オブジェクトのターゲットバージョン識別子のプロパティは，新しいカレントバージョン（null）になります。

(2) 形式

(a) 形式 1

```
void checkIn()
```

(b) 形式 2

```
void checkIn(
     Collection<String>     propDefs
)
```

(c) 形式 3

```
void checkIn(
     DbjPropSet     propSet
)
```

(3) 引数

propDefs（入力）

要素はプロパティ名（String 型）です。チェックイン後のカレントバージョンを対象としたバージョン付きオブジェクトに設定する Proxy オブジェクトのプロパティとして，どのプロパティを設定するかを指定します。null を指定するとプロパティは設定されません。

propSet（入力）

チェックイン後のカレントバージョンを対象とした，バージョン付きオブジェクトに設定するプロパティ値集合を指定します。null を指定するとプロパティは設定されません。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjACEOperationException

指定 ACE についてのエラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
指定プロパティ値が不正の場合

DbjNotCheckOutException
チェックアウトしていなかった場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException
指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

NullPointerException
引数 propSet が null の場合

6.4.6 checkOut（バージョン付きオブジェクトのチェックアウト）

（1） 機能

Proxy オブジェクトのアクセス対象のバージョン付きオブジェクトをチェックアウトします。チェックアウトによって，チェックアウト中の仮のバージョン識別子を取得します。チェックアウトできるターゲットオブジェクトは，バージョン付きオブジェクトだけです。

形式 1 および形式 2 では，チェックアウトと同時に Proxy オブジェクトに設定されているプロパティ値集合がチェックアウト中の仮のバージョンを対象としてプロパティに設定されます。形式 2 では，設定するプロパティを限定できます（指定したプロパティだけが，プロパティ設定の対象になります）。形式 3 では，Proxy オブジェクトのものに関係なく，指定したプロパティ値集合が仮のバージョンを対象として設定されます。また，Proxy オブジェクトのターゲットバージョン識別子のプロパティは，チェックアウト中のバージョンを指します。

（2） 形式

（a） 形式 1

```
void checkOut()
```

（b） 形式 2

```
void checkOut(
     Collection<String>     propDefs
)
```

（c） 形式 3

```
void checkOut(
     DbjPropSet     propSet
)
```

（3） 引数

propDefs（入力）
: 要素はプロパティ名（String 型）です。チェックアウト後の仮のバージョンオブジェクトを対象としたバージョン付きオブジェクトに設定する Proxy オブジェクトのプロパティとして，どのプロパティを設定するかを指定します。null を指定するとプロパティは設定されません。

propSet（入力）
: チェックアウト後の仮のバージョンオブジェクトを対象としたバージョン付きオブジェクトに設定するプロパティ値集合を指定します。null を指定するとプロパティは設定されません。

（4） 戻り値

なし

（5） 例外

DbjAccessControlException
: アクセス権エラーの場合

DbjACEOperationException
: 指定 ACE についてのエラーの場合

DbjAlreadyCheckOutException
: すでにチェックアウトしている場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有エラーの場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
指定プロパティ値が不正の場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException
指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

NullPointerException
引数 propSet が null の場合

6.4.7 convertContentType（コンテンツ種別の変換）

（1） 機能

指定されたコンテンツの格納先（コンテンツ種別）を変換します。コンテンツ種別を変換できるターゲットオブジェクトは，文書だけです。

（2） 形式

```
void convertContentType(
      DbjConvertContentInfoconvertInfo
)
```

（3） 引数

convertInfo（入力）

コンテンツ種別の変換情報を指定します。null を指定すると例外がスローされます。

（4） 戻り値

なし

（5） 例外

DbjAccessControlException

アクセス権エラーの場合

DbjConvertContentTargetNotFoundException

変換対象のコンテンツが存在しなかった場合

DbjContentNotRegisteredException

リファレンスファイル管理機能を使用したオブジェクトをローカルファイルにダウンロードする場合に，オブジェクトにコンテンツが登録されていなかったとき

DbjContentTypeMismatchException

リファレンスファイル管理機能で，オブジェクトのコンテンツ種別とメソッドが要求するコンテンツ種別が一致しない場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有エラーの場合

DbjFileReferenceOperationFailedException

リファレンスファイル管理機能を使用するサーバでのコンテンツ操作が失敗した場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切な場合

DbjIOException

DocumentBroker クラスライブラリ固有の IO エラーの場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjRenditionCountOutOfRangeException
　レンディション数が範囲を超えた場合

DbjRenditionTypeDuplicatedException
　指定レンディションタイプが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

DbjTargetContentPathNotSetException
　リファレンスファイル管理機能で，コンテンツ格納先ベースパスが設定されていなかった場合

IllegalArguemntException
　指定した引数が不正の場合

NullPointerException
　引数 convertInfo が null の場合

6.4.8 deleteRendition（レンディションの削除）

(1) 機能

Proxy オブジェクトのアクセス対象の文書から指定したレンディションタイプのレンディションを削除します。レンディションの削除によって，存在しなかったレンディションタイプのリストが返却されます。レンディションを削除できるターゲットオブジェクトは，文書だけです。

(2) 形式

```
List deleteRendition(
     List<String>       renditionTypeList
)
```

(3) 引数

renditionTypeList（入力）

要素には，削除するレンディションタイプ（String 型）のリストを指定します。null を指定すると例外がスローされます。

(4) 戻り値

すでに削除されていたレンディションタイプ（String 型）のリスト

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjIsMasterRenditionException

マスタレンディションを削除しようとした場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 renditionTypeList が null の場合

6.4.9 deleteVersion（バージョンの削除）

（1）機能

Proxy オブジェクトのアクセス対象のバージョン付きオブジェクトから指定した複数のバージョンを削除します。バージョンを削除できるターゲットオブジェクトは，文書だけです。ただし，複数のバージョンを削除する場合，1バージョンは残す必要があります。もし，すべてのバージョンを指定した場合は，例外がスローされます。バージョンの削除によって，すでに削除されていたバージョンなど，削除されなかったバージョンのバージョン識別子のリストが返却されます。

（2）形式

```
List deleteVersion(
     List<String>    versionIds
)
```

（3）引数

versionIds（入力）

要素には，削除するバージョンのバージョン識別子（String 型）のリストを指定します。バージョン識別子に null を指定するとカレントバージョンになります。バージョン識別子が重複していてもエラーにはなりません。null を指定すると例外がスローされます。

（4）戻り値

すでに削除されていたオブジェクトのバージョンのバージョン識別子（String 型）のリスト

（5）例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjLastVersionException
　最後のバージョンを削除しようとした場合

DbjNotCheckOutException
　チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

NullPointerException
　引数 versionIds が null の場合

6.4.10 downloadContents（文書のコンテンツのダウンロード）

（1） 機能

文書のコンテンツをダウンロードします。

形式 1 では，レンディションタイプを指定した文書のコンテンツを，指定したローカルファイルにダウンロードします。リファレンスファイル文書に対しては実行できません。リファレンスファイル文書に対して実行した場合は，DbjException 例外がスローされます。形式 2 では，リファレンスファイル文書のコンテンツを指定したパスにダウンロードします。リファレンスファイル文書に対してだけ実行できます。リファレンスファイル文書以外にこのメソッドを実行した場合は DbjContentTypeMismatchException 例外がスローされます。形式 3 では，指定したレンディションタイプの文書のコンテンツを一時ディレクトリ下の一時ファイルにダウンロードします。DbjContentInfo#getInputStream の呼び出しにより入力ストリームで取得することができます。リファレンスファイル文書に対しても実行できます。形式 3 を使用する場合，文書空間構成定義ファイルの TempDirectory プロパティを指定してください。指定されたディレクトリ下に一時ファイルを作成します。作成した一時ファイルは，DbjContentInfo#close() メソッド実行時または DbjSession#logout() メソッド実行時に削除されます。

（2） 形式

（a） 形式 1

```
DbjContentInfo downloadContents(
     String     renditionType,
     String     filePath
)
```

（b） 形式 2

```
DbjReferenceContentInfo downloadContents(
     String                     renditionType,
     DbjReferencePathInfo       pathInfo
)
```

（c） 形式 3

```
DbjContentInfo downloadContents(
     String     renditionType,
)
```

（3） 引数

renditionType（入力）

レンディションタイプを指定します。null を指定するとマスタレンディションが仮定されます。

filePath（入力）

ダウンロード先のパス名（ローカルファイル，およびファイル名を含みます）を指定します。null を指定すると例外がスローされます。

pathInfo（入力）

ダウンロード先の情報を指定します。null を指定すると例外がスローされます。

（4） 戻り値

形式 1，形式 3 の場合

コンテンツ情報（DbjContentInfo インターフェース）

形式 2 の場合

リファレンスファイル機能用のコンテンツ情報（DbjReferenceContentInfo インタフェース）

(5) 例外

DbjAccessControlException
アクセス権エラーの場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjIOException
DocumentBroker クラスライブラリ固有の IO エラーの場合

DbjNotCheckOutException
チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjRenditionNotFoundException
指定レンディションが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

DbjContentTypeMismatchException
このメソッドでは操作できないコンテントを対象に実行した場合

DbjFileReferenceOperationFailedException
リファレンスファイル機能を使用するサーバのコンテント操作においてエラーが発生した場合

DbjTargetContentPathNotSetException
リファレンスファイル機能において相対パスのベースとなるパスが設定されていない場合

DbjContentNotRegisteredException
コンテンツが登録されていない場合

NullPointerException
引数 filePath または引数 pathInfo が null の場合

6.4.11 getBindObjectList（パブリック ACL にバインドしている文書管理オブジェクト一覧の取得）

(1) 機能

Proxy オブジェクトのアクセス対象のパブリック ACL にバインドされている文書管理オブジェクトの一覧情報を取得します。一覧情報を取得できるターゲットオブジェクトは，パブリック ACL だけです。取得対象とするオブジェクトのオブジェクト種別を指定できます。

バージョン付きオブジェクトが取得対象のオブジェクトに含まれる場合は，バージョニングオブジェクトのプロパティだけ取得でき，バージョンオブジェクトのプロパティは取得できません。

(2) 形式

```
DbjObjList getBindObjectList(
     Collection<String>      propDefs,
     int                     objType,
     DbjFetchInfo            fetchInfo
)
```

(3) 引数

propDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するオブジェクトのプロパティ名のコレクションを指定します。取得するプロパティがない場合は null を指定します。

objType（入力）

一覧で取得するオブジェクトのオブジェクト種別を指定します。次に示す定数のどれかを指定できます。なお，論理和を使用した指定もできます。

- DbjDef.OBJTYPE_FOLDER
  バージョンなしフォルダ
- DbjDef.OBJTYPE_DOC
  バージョンなし文書
- DbjDef.OBJTYPE_VRDOC
  バージョン付き文書
- DbjDef.OBJTYPE_IP
  独立データ
- DbjDef.OBJTYPE_ANY
  任意のオブジェクト種別

fetchInfo（入出力）

一覧で取得する場合の検索結果取得情報を指定します。全件を取得する場合，またはキャッシュ検索しない場合は null を指定します。全件を取得しない場合，maxFetchCount プロパティに指定されている件数を取得した後に，comparator プロパティに指定したコンパレータによりソートされます。そのため，全件を取得する場合と，maxFetchCount プロパティを指定して検索した場合を比較すると，comparator プロパティに同じオブジェクトを指定しても結果が異なる場合があります。

(4) 戻り値

バインドしているオブジェクト一覧情報（DbjObjList インターフェース）

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjAccessControlNotSupportedException
アクセス制御機能に対応していない文書だった場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalCacheStartIndexException
検索結果数より大きい値を引数 fetchInfo（DbjFetchInfo インターフェース）の startIndex プロパティに指定した場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalArgumentException
引数 fetchInfo の指定が不正の場合

6.4.12 getCheckOutStatus（バージョン付きオブジェクトのチェックアウト状態の取得）

(1) 機能

Proxy オブジェクトのアクセス対象のバージョン付きオブジェクトのチェックアウト状態を取得します。DbjCheckOutInfo インターフェースを使用して，チェックアウト状態，チェックアウトしているユーザ識別子，チェックアウトしている仮のバージョンのバージョン識別子などを取得できます。チェックアウト状態を取得できるターゲットオブジェクトは，バージョン付きオブジェクトだけです。

(2) 形式

```
DbjCheckOutInfo getCheckOutStatus()
```

(3) 引数

なし

(4) 戻り値

チェックアウト情報（DbjCheckOutInfo インターフェース）

(5) 例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.4.13 getChildList（フォルダのリンク先オブジェクト（下位オブジェクト）の取得）

（1）機能

Proxy オブジェクトのアクセス対象のフォルダにリンクが設定されているリンク先オブジェクト（下位オブジェクト）の一覧情報を取得します。取得対象とするリンク種別とリンク先オブジェクトのオブジェクト種別を指定できます。

ユーザアプリケーションプログラムの取得対象のオブジェクトに，リンク種別として直接型リンクまたは参照型リンクを指定し，バージョン付きオブジェクトが含まれる場合について説明します。この場合，取得対象のオブジェクトがバージョニングオブジェクトのときは，バージョニングオブジェクトのプロパティだけ取得でき，バージョンオブジェクトのプロパティは取得できません。また，取得対象のオブジェクトがバージョニングオブジェクトを構成するバージョンオブジェクトのときは，バージョンオブジェクトのプロパティだけ取得できます。ただし，プロパティの指定には@プレフィックスを付けないで指定します。

（2）形式

```
DbjLinkObjList getChildList(
     Collection<String>    propDefs,
     Collection<String>    linkPropDefs,
     int                   linkType,
     int                   objType,
     DbjFetchInfo          fetchInfo
)
```

（3）引数

propDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するリンク先オブジェクトのプロパティ名のコレクションを指定します。取得するプロパティがない場合は null を指定します。

linkPropDefs（入力）

要素はプロパティ名 (String 型) です。一覧で取得するリンクプロパティ名のコレクションを指定します。取得するリンクプロパティがない場合は null を指定します。

linkType（入力）

リンク種別を指定します。次に示す定数のどれかを指定できます。なお，論理和を使用した指定もできます。

- DbjDef.LINK_DCR
  直接型リンク
- DbjDef.LINK_RCR
  参照型リンク
- DbjDef.LINK_DCR | DbjDef.LINK_RCR
  直接型リンクまたは参照型リンク

objType（入力）

一覧で取得するリンク先オブジェクトのオブジェクト種別を指定します。次に示す定数のどれかを指定できます。なお，論理和を使用した指定もできます。

- DbjDef.OBJTYPE_FOLDER
  バージョンなしフォルダ
- DbjDef.OBJTYPE_DOC
  バージョンなし文書

- DbjDef.OBJTYPE_VRDOC
  バージョン付き文書
- DbjDef.OBJTYPE_ANY
  任意のオブジェクト種別

fetchInfo（入出力）

一覧で取得する場合の検索結果取得情報を指定します。全件を取得する場合，またはキャッシュ検索をしない場合は null を指定します。
全件を取得しない場合，maxFetchCount プロパティに指定されている件数を取得した後に，comparator プロパティに指定したコンパレータによりソートされます。そのため，全件を取得する場合と，maxFetchCount プロパティを指定して検索した場合を比較すると，comparator プロパティに同じオブジェクトを指定しても結果が異なる場合があります。

（4）戻り値

リンクオブジェクト一覧情報（DbjLinkObjList インターフェース）

（5）例外

DbjAccessControlException
アクセス権エラーの場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjIllegalCacheStartIndexException
検索結果数より大きい値を引数 fetchInfo（DbjFetchInfo インターフェース）の startIndex プロパティに指定した場合

DbjNotCheckOutException
チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalArgumentException
引数 linkType，fetchInfo の指定が不正の場合

6.4.14 getClassName（アクセス対象文書管理オブジェクトを構成するDocumentBroker クラス名の取得）

（1） 機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトの DocumentBroker クラス名を取得します。

バージョンオブジェクトについては，バージョニングオブジェクトおよびバージョンオブジェクトを構成する二つの DocumentBroker クラスのクラス名をこの順に返し，このほかのオブジェクトについては一つの DocumentBroker クラスのクラス名を返します。ただし，バージョン付きオブジェクトに対しては，アクセス権エラーが発生するおそれがあります。

（2） 形式

```
List<String> getClassName()
```

（3） 引数

なし

（4） 戻り値

要素を DocumentBroker クラスのクラス名（String 型）とするリストインターフェース

（5） 例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.4.15 getDCRParent（直接型リンクによるリンク元オブジェクトの取得）

(1) 機能

Proxy オブジェクトのアクセス対象のフォルダに直接型リンクが設定されているリンク元（上位）オブジェクトの情報を取得します。リンク元オブジェクトがない場合は，null が返却されます。リンクオブジェクトについての情報（リンク識別子）は取得できません。

(2) 形式

```
DbjObj getDCRParent(
     Collection<String>    propDefs
)
```

(3) 引数

propDefs（入力）

要素はプロパティ名（String 型）です。取得するリンク元オブジェクトのプロパティ名のコレクションを指定します。取得するプロパティがない場合は null を指定します。

(4) 戻り値

リンク元のリンクオブジェクト（DbjObj インターフェース）

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

6.4.16 getLockType（Proxy オブジェクトのアクセスロック種別の取得）

(1) 機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトにアクセスする場合のロック種別（Proxy オブジェクトのアクセスロック種別のプロパティ）を取得します。取得するロック種別は，参照系メソッドによって取得するロック種別です。

(2) 形式

```
int getLockType()
```

(3) 引数

なし

(4) 戻り値

次のどちらかのロック種別を取得します。

DbjDef.LOCK_READ
: read ロックを取得します。

DbjDef.LOCK_WRITE
: write ロックを取得します。

(5) 例外

なし

6.4.17 getObjType（アクセス対象文書管理オブジェクトのオブジェクト種別の取得）

(1) 機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトのオブジェクト種別を取得します。

(2) 形式

```
int getObjType()
```

(3) 引数

なし

(4) 戻り値

次のどれかのオブジェクト種別が返却されます。

DbjDef.OBJTYPE_DOC
　バージョンなし文書です。

DbjDef.OBJTYPE_FOLDER
　バージョンなしフォルダです。

DbjDef.OBJTYPE_VRDOC
　バージョン付き文書です。

DbjDef.OBJTYPE_IP
　独立データです。

DbjDef.OBJTYPE_PUBLICACL
　パブリック ACL です。

(5) 例外

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.4.18 getOiid（アクセス対象文書管理オブジェクトの OIID の取得）

（1）機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトの OIID を取得します。

（2）形式

```
String getOiid()
```

（3）引数

なし

（4）戻り値

Proxy オブジェクトのアクセス対象である文書管理オブジェクトの OIID 文字列（String 型）

（5）例外

なし

6.4.19 getParentList（フォルダのリンク元オブジェクトの取得）

（1）機能

Proxy オブジェクトのアクセス対象のフォルダにリンクが設定されているリンク元オブジェクトの一覧情報を取得します。取得対象のリンク種別およびリンク元オブジェクトのオブジェクト種別を指定できます。

（2）形式

```
DbjLinkObjList getParentList(
     Collection<String>       propDefs,
     Collection<String>       linkPropDefs,
     int                      linkType,
     int                      objType,
     DbjFetchInfo             fetchInfo
)
```

（3）引数

propDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するリンク元オブジェクトのプロパティ名のコレクションを指定します。取得するプロパティがない場合は null を指定します。

linkPropDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するリンクプロパティ名のコレクションを指定します。取得するリンクプロパティがない場合は null を指定します。

linkType（入力）

リンク種別を指定します。次に示す定数のどれかを指定できます。

- DbjDef.LINK_DCR
  直接型リンク
- DbjDef.LINK_RCR
  参照型リンク
- DbjDef.LINK_DCR | DbjDef.LINK_RCR
  直接型リンクまたは参照型リンク

objType（入力）

一覧で取得するリンク先オブジェクトのオブジェクト種別を指定します。次に示す定数のどれかを指定できます。なお，論理和を使用した指定もできます。

- DbjDef.OBJTYPE_FOLDER
  バージョンなしフォルダ
- DbjDef.OBJTYPE_ANY
  任意のオブジェクト種別

fetchInfo（入出力）

一覧で取得する場合の検索結果取得情報を指定します。全件を取得する場合，またはキャッシュ検索をしない場合は null を指定します。全件を取得しない場合，maxFetchCount プロパティに指定されている件数を取得した後に，comparator プロパティに指定したコンパレータによりソートされます。そのため，全件を取得する場合と，maxFetchCount プロパティを指定して検索した場合を比較すると，comparator プロパティに同じオブジェクトを指定しても結果が異なる場合があります。

（4）戻り値

リンクオブジェクト一覧情報（DbjLinkObjList インターフェース）

(5) 例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalCacheStartIndexException
　検索結果数より大きい値を引数 fetchInfo（DbjFetchInfo インターフェース）の startIndex プロパティに指定した場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

IllegalArgumentException
　引数 linkType，fetchInfo の指定が不正の場合

6.4.20 getPublicACLList（バインドしているパブリック ACL 一覧の取得）

(1) 機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトがバインドしているパブリック ACL の一覧情報を取得します。ただし，Proxy オブジェクトのアクセス対象の文書管理オブジェクトには，パブリック ACL を指定しないでください。

(2) 形式

```
DbjObjList getPublicACLList(
     Collection<String>     propDefs
)
```

(3) 引数

propDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するパブリック ACL のプロパティ名のコレクションを指定します。取得するプロパティがない場合は null を指定します。

(4) 戻り値

バインドしているパブリック ACL 一覧情報（DbjObjList インターフェース）

(5) 例外

DbjAccessControlException
アクセス権エラーの場合

DbjAccessControlNotSupportedException
アクセス制御機能に対応していない文書空間だった場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

6.4.21 getRelList（文書間リンク一覧の取得）

（1）機能

Proxy オブジェクトのアクセス対象の文書に文書間リンクが設定されている文書の一覧情報を取得します。一覧情報を取得できるターゲットオブジェクトは，文書だけです。この場合，取得対象とするオブジェクトのオブジェクト種別を指定できます。また，引数（relationendType）の指定によって，取得するオブジェクトの情報をフィルタリングできます。

（2）形式

```
DbjLinkObjList getRelList(
     Collection<String>     propDefs,
     Collection<String>     linkPropDefs,
     int                    relationendType,
     int                    objType,
     DbjFetchInfo           fetchInfo
)
```

（3）引数

propDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するリンク先オブジェクトのプロパティ名集合を指定します。取得するプロパティがない場合は null を指定します。

linkPropDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するリンクプロパティ名集合を指定します。取得するリンクプロパティがない場合は null を指定します。

relationendType（入力）

取得するオブジェクトの条件を指定します。次に示す定数のどれかを指定できます。
RELATIONEND_STATUS 定数を指定しない場合は，リンク先のオブジェクトはすべて取得します。
なお，DbjDef.RELATIONEND_HEAD または DbjDef.RELATIONEND_TAIL のどちらか一つは，必ず指定してください。

- DbjDef.RELATIONEND_HEAD
  リンク先オブジェクトを取得します。RELATIONEND_TAIL と同時に指定できません。
- DbjDef.RELATIONEND_TAIL
  リンク元オブジェクトを取得します。RELATIONEND_HEAD と同時に指定できません。
- DbjDef.RELATIONEND_STATUS_EXIST
  リンク先オブジェクトがあるオブジェクトを取得します。
- DbjDef.RELATIONEND_STATUS_NOT_EXIST
  リンク先オブジェクトが削除されて存在しないオブジェクトを取得します。

objType（入力）

一覧で取得するリンク元またはリンク先オブジェクトのオブジェクト種別を指定します。次に示す定数のどれかを指定できます。なお，論理和を使用した指定もできます。

- DbjDef.OBJTYPE_DOC
  バージョンなし文書
- DbjDef.OBJTYPE_VRDOC
  バージョン付き文書
- DbjDef.OBJTYPE_ANY
  任意のオブジェクト種別

fetchInfo（入出力）

一覧で取得する場合の検索結果取得情報を指定します。全件を取得する場合，またはキャッシュ検索をしない場合は null を指定します。全件を取得しない場合，maxFetchCount プロパティに指定されている件数を取得した後に，comparator プロパティに指定したコンパレータによりソートされます。そのため，全件を取得する場合と，maxFetchCount プロパティを指定して検索した場合を比較すると，comparator プロパティに同じオブジェクトを指定しても結果が異なる場合があります。

(4) 戻り値

リンクオブジェクト一覧情報（DbjLinkObjList インターフェース）

(5) 例外

DbjAccessControlException
アクセス権エラーの場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalCacheStartIndexException
検索結果数より大きい値を引数 fetchInfo（DbjFetchInfo インターフェース）の startIndex プロパティに指定した場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalArgumentException
引数 fetchInfo の指定が不正の場合

6.4.22 getRenditionList（レンディション情報一覧の取得）

（1）機能

Proxy オブジェクトのアクセス対象の文書が保持するレンディション情報の一覧をすべて取得します。レンディション情報を取得できるアクセス対象の文書管理オブジェクトは，文書だけです。

（2）形式

```
DbjRenditionList getRenditionList(
     Collection<String>     propDefs
)
```

（3）引数

propDefs（入力）

要素はレンディションプロパティ名（String 型）です。一覧で取得するレンディションのプロパティ名のコレクションを指定します。取得するレンディションプロパティがない場合は null を指定します。次に示すプロパティを指定できます。

- dbrProp_RetrievalName プロパティ
- dbrProp_RenditionStatus プロパティ
- dbrProp_ContentType プロパティ
- dbrProp_ContentLocation プロパティ
- dbrProp_ReferenceType プロパティ

（4）戻り値

レンディション一覧情報（DbjRenditionList インターフェース）

（5）例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjNotCheckOutException
　チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.4.23 getTargetVersion（アクセス対象ターゲットバージョンのバージョン識別子の取得）

（1） 機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトのバージョンのバージョン識別子を取得します。

（2） 形式

```
String getTargetVersion()
```

（3） 引数

なし

（4） 戻り値

Proxy オブジェクトのアクセス対象である文書管理オブジェクトのバージョンのバージョン識別子（String 型）が返却されます。なお，カレントバージョンの場合は，null が返却されることがあります。

（5） 例外

なし

6.4.24 getVersionId（バージョンオブジェクトのバージョン識別子の取得）

（1）機能

Proxy オブジェクトのアクセス対象オブジェクトがバージョン付きオブジェクトのバージョンオブジェクトの場合，そのバージョン識別子を取得します。アクセス対象の文書管理オブジェクトがバージョンオブジェクトでない場合は，null が返却されます。

（2）形式

```
String getVersionId()
```

（3）引数

なし

（4）戻り値

バージョンオブジェクトのバージョン識別子（String 型）

（5）例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.4.25 getVersioningInfo（バージョン付きオブジェクトのバージョニングオブジェクトの取得）

(1) 機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトがバージョン付きオブジェクトの場合，このバージョンオブジェクトに対応するバージョニングオブジェクトの Proxy オブジェクトインターフェース（DbjObj インターフェース）を取得します。該当するバージョンのバージョン識別子は，その Proxy オブジェクトインターフェースの DbjObj#getTargetVersion メソッドで取得できます。したがって，取得した DbjObj インターフェースが指す Proxy オブジェクトの対象バージョンは，Proxy オブジェクトのバージョン識別子を示します。

アクセス対象の文書管理オブジェクトがバージョンオブジェクトでない場合，null が返却されます。

(2) 形式

```
DbjObj getVersioningInfo()
```

(3) 引数

なし

(4) 戻り値

バージョン付きオブジェクトの Proxy オブジェクトインターフェース（DbjObj インターフェース）

(5) 例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.4.26 getVersionObjList（バージョン付きオブジェクトのバージョン一覧の取得）

（1）機能

Proxy オブジェクトのアクセス対象であるバージョン付きオブジェクトのバージョンオブジェクトの一覧情報を取得します。一覧情報を取得するアクセス対象の文書管理オブジェクトは，バージョン付きオブジェクトだけです。取得対象のバージョンオブジェクトから取得するプロパティを指定できます。また，一覧で取得するバージョンオブジェクトの並べ方を指定できます。

（2）形式

```
DbjVerObjList getVersionObjList(
     Collection<String>     propDefs,
     int                    order,
     DbjFetchInfo           fetchInfo
)
```

（3）引数

propDefs（入力）

要素はプロパティ名（String 型）です。一覧で取得するバージョンオブジェクトのプロパティ名のコレクションを指定します。取得するプロパティがない場合は null を指定します。

dmaProp_OIID プロパティを指定しない場合，dmaProp_OIID プロパティが自動的に追加されるものとみなします。

order（入力）

一覧で取得する場合の並べ方を指定します。次の定数のどれかを指定できます。

- DbjDef.ORDER_ASC
  古い順でソートする
- DbjDef.ORDER_DESC
  新しい順でソートする
- DbjDef.ORDER_NONE
  ソート方法を指定しない

fetchInfo（入出力）

一覧で取得する場合の検索結果取得情報を指定します。全件を取得する場合，またはキャッシュ検索をしない場合は null を指定します。

キャッシュ検索すると cacheKey プロパティと cacheTotal プロパティが API によって設定されます。全件を取得しない場合，maxFetchCount プロパティに指定されている件数を取得した後に，comparator プロパティに指定したコンパレータによりソートされます。そのため，全件を取得する場合と，maxFetchCount プロパティを指定して検索した場合を比較すると，comparator プロパティに同じオブジェクトを指定しても結果が異なる場合があります。

（4）戻り値

バージョンオブジェクトの一覧情報（DbjVerObjList インターフェース）

（5）例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalCacheStartIndexException
検索結果数より大きい値を引数 fetchInfo（DbjFetchInfo インターフェース）の startIndex プロパティに指定した場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalArgumentException
引数 fetchInfo の指定が不正の場合

6.4.27 link（リンク先オブジェクトとのリンク）

（1）機能

Proxy オブジェクトのアクセス対象オブジェクトのリンク元オブジェクトに，指定した複数のリンク先オブジェクトとのリンクを設定します。

形式 1 では，DbjSetLinkInfo インターフェースを要素とするリストオブジェクトで，リンク先オブジェクトとリンク種別を指定します。この場合，リンク先オブジェクトごとにリンク種別を選択できます。形式 2 では，リンク種別は一つだけ選択でき，DbjObj インターフェースのリストにリンク先オブジェクトを指定できます。

（2）形式

（a）形式 1

```
void link(
     List<? extends DbjSetLinkInfo>     childLinkList
)
```

（b）形式 2

```
void link(
     int               linkType,
     List<DbjObj>      childObjs
)
```

（3）引数

childLinkList（入力）

要素は DbjSetLinkInfo インターフェースのサブインターフェースです。リンクを設定するリンク先オブジェクトとそのリンクの種類のリストを指定します。null を指定すると例外がスローされます。

linkType（入力）

リンク種別を指定します。次のどれかを指定できます。

- DbjDef.LINK_DCR
  直接型リンク
- DbjDef.LINK_RCR
  参照型リンク
- DbjDef.LINK_REL
  文書間リンク

childObjs（入力）

要素は DbjObj インターフェースです。リンク対象のリンク先オブジェクトのリストを指定します。null を指定すると例外がスローされます。

（4）戻り値

なし

（5）例外

ClassCastException

リスト引数の要素が不適切の場合

DbjAccessControlException

アクセス権エラーの場合

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
指定オブジェクト種別が不適切の場合

DbjIllegalPropValException
指定プロパティ値が不正の場合

DbjNotCheckOutException
チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalArgumentException
引数 linkType が不正の場合

NullPointerException
引数 childLinkList，または childObjs が null の場合

6.4.28 lock（アクセスロック種別の異なる文書管理オブジェクトインターフェースの取得）

（1）機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトに，指定したロック種別でロックを取得する Proxy オブジェクトのインターフェース（DbjObj インターフェース）を取得します。

元の DbjObj インターフェース（this）でアクセスする際のロック種別は，変更されません。なお，引数に指定する値の妥当性は検証されません。

（2）形式

```
DbjObj lock(
     int     lockType
)
```

（3）引数

lockType（入力）

ロック種別を指定します。次に示す定数のどちらかを指定します。

- DbjDef.LOCK_READ
  read ロック
- DbjDef.LOCK_WRITE
  write ロック

（4）戻り値

DbjObj インターフェース（DbjObj インターフェース）

（5）例外

なし

6.4.29 move（直接型リンクが設定されている文書管理オブジェクトの移動）

(1) 機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトに対してリンク元オブジェクトから直接型リンクが設定されている場合，そのリンクを解除して代わりに指定したリンク元フォルダオブジェクトに直接型リンクを設定します。文書管理オブジェクトに直接型リンクが設定されていない場合は，指定したリンク元フォルダに直接型リンクを設定します。文書管理オブジェクトに直接型リンク以外のリンクが設定されている場合は，そのリンクは解除されます。

(2) 形式

```
void move(
     DbjObj     parentFolder
)
```

(3) 引数

parentFolder（入力）

自オブジェクトが移動するリンク元フォルダを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 parentFolder が null の場合

6.4.30 propSet（Proxy オブジェクトのプロパティ値集合インターフェースの取得）

（1） 機能

Proxy オブジェクトのプロパティ値集合インターフェースを取得します。プロパティ値集合インターフェースは，参照権限で取得します。取得したプロパティ値集合インターフェースを使用して値を設定すると Proxy オブジェクトのプロパティ値も更新されます。ただし，メソッド実行時に，プロパティ値集合インターフェースが設定されていなくても null は返却されません。

（2） 形式

```
DbjPropSet propSet()
```

（3） 引数

なし

（4） 戻り値

Proxy オブジェクトのプロパティ値集合インターフェース（DbjPropSet インターフェース）

（5） 例外

なし

6.4.31 readProperties（文書管理オブジェクトのプロパティ値の読み込み）

（1） 機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトの指定プロパティを，Proxy オブジェクトのプロパティに読み込みます。Proxy オブジェクトに読み込まれたプロパティ値集合は，DbjObj#propSet メソッドで DbjPropSet インターフェースを取得して参照できます。読み込まれたプロパティ値集合は，Proxy オブジェクトで保持しているプロパティ値集合に上書きして追加されます。

形式 1 では，Proxy オブジェクトの保持しているプロパティ値集合でのすべてのプロパティを文書管理オブジェクトから読み込みます。形式 2 では，読み込み対象のプロパティのコレクションを指定します。

（2） 形式

（a） 形式 1

```
void readProperties()
```

（b） 形式 2

```
void readProperties(
     Collection<String>     propDefs
)
```

（3） 引数

propDefs（入力）

要素はプロパティ名（String 型）です。取得するプロパティのプロパティ名のコレクションを指定します。null を指定すると例外がスローされます。

（4） 戻り値

なし

（5） 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 propDefs が null の場合

6.4.32 removeObject（文書管理オブジェクトの削除）

（1） 機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトを削除します。

形式 2 では，フォルダの削除の場合は直接型リンクが設定されているすべてのリンク先オブジェクトを削除します。

形式 3 では，リファレンスファイル管理機能を使用したオブジェクトを削除すると同時に，コンテンツを格納していたフォルダを削除できます。

なお，デッドロックを防止するため，リンク先オブジェクトを削除する場合は，リンク元オブジェクトを削除するまたはリンク元オブジェクトに write ロックします。このメソッドの実行時に，文書管理オブジェクトが存在しない場合，例外はスローされません。

（2） 形式

（a） 形式 1

```
boolean removeObject()
```

（b） 形式 2

```
boolean removeObject(
     boolean     isAlsoChildren
)
```

（c） 形式 3

```
boolean removeObject(

     DbjReferencePathInfo      pathInfo
)
```

（3） 引数

isAlsoChildren（入力）

リンク先オブジェクトの削除を指定します。true の場合，リンク先オブジェクトを削除します。false の場合，リンク先オブジェクトを削除しません。

pathInfo（入力）

リファレンスファイル管理機能を使用したオブジェクトの削除時に使用する，パス情報を設定します。null を指定すると例外がスローされます。

（4） 戻り値

次に示す値が返却されます。

true

リンク先オブジェクトが削除されました。

false

該当する文書管理オブジェクトが存在しませんでした。

（5） 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjLastVersionException
　最後のバージョンを削除しようとした場合

DbjNotCheckOutException
　チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

NullPointerException
　引数 pathInfo が null の場合

6.4.33 setPropSet（Proxy オブジェクトのプロパティ値集合の設定）

（1）機能

Proxy オブジェクトのプロパティ値集合に，指定したプロパティ値集合の内容をコピーして設定します。

（2）形式

```
void setPropSet(
     DbjPropSet     propSet
)
```

（3）引数

propSet（入力）

プロパティ値集合インターフェースを指定します。null を指定するとオブジェクトのプロパティ値集合には空値が仮定されます。

（4）戻り値

なし

（5）例外

なし

6.4.34 setTargetVersion（アクセス対象ターゲットバージョンのバージョン識別子の変更）

（1）機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトのバージョンのバージョン識別子を変更します。バージョン識別子の変更によって，元の DbjObj インターフェースを取得します。デフォルトでは，Proxy オブジェクトのアクセス対象のバージョンは，カレントバージョンになります。なお，バージョン識別子の妥当性は検証されません。

（2）形式

```
DbjObj setTargetVersion(
     String     versionId
)
```

（3）引数

versionId（入力）

変更する対象バージョンのバージョン識別子を指定します。null を指定するとカレントバージョンに変更されます。

（4）戻り値

元のオブジェクトインターフェース（this）

（5）例外

なし

6.4.35 unbindPublicACL（パブリック ACL のアンバインド）

（1）機能

Proxy オブジェクトのアクセス対象の文書管理オブジェクトから指定したパブリック ACL をアンバインドします。ただし，Proxy オブジェクトがアクセスする文書管理オブジェクトには，パブリック ACL を指定しないでください。

（2）形式

```
void unbindPublicACL(
     List<DbjObj>     publicACLList
)
```

（3）引数

publicACLIdList（入力）

要素は DbjObj インターフェースです。各要素に，アンバインドするパブリック ACL を指定します。null を指定すると例外がスローされます。

（4）戻り値

なし

（5）例外

ClassCastException

引数 publicACLList の要素が DbjObj インターフェースでなかった場合

DbjAccessControlException

アクセス権エラーの場合

DbjAccessControlNotSupportedException

接続先の文書空間がアクセス制御動作モードに対応していなかった場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjPublicACLNotBoundException

指定パブリック ACL がバインドされていない場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 publicACLIdList が null の場合

6.4.36 unlink（リンク先オブジェクトとのリンクの解除）

(1) 機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトのリンク元オブジェクトから，指定した複数のリンク先オブジェクトとのリンクを解除します。

リンクを解除するリンク先オブジェクトは，DbjLinkObj インターフェースまたは DbjObj インターフェースを要素とするリストで指定します。DbjLinkObj インターフェースと DbjObj インターフェースを同時に指定することもできます。DbjObj インターフェースを指定した場合，その DbjObj インターフェースが指す文書管理オブジェクトとの直接型リンク，参照型リンク，および文書間リンクがすべて解除されます。

なお，指定したリンクがすでに解除されていた場合は，そのリンクまたはリンク先オブジェクトのインターフェースを要素とするリストが戻り値として返却されます。

(2) 形式

```
List unlink(
     List<?>     childList
)
```

(3) 引数

childList（入力）

要素は DbjLinkObj インターフェースまたは DbjObj インターフェースです。
リストの要素として，リンクを解除するリンクオブジェクトのインターフェースまたはリンク先オブジェクトのインターフェースを指定します。null を指定すると例外がスローされます。

(4) 戻り値

すでにリンクが解除されていたインターフェースのリスト

(5) 例外

ClassCastException
　リスト引数の要素が不適切の場合

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException
　指定オブジェクト種別が不適切の場合

DbjNotCheckOutException
　チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

NullPointerException
　引数 childLinkList，または childObjs が null の場合

6.4.37 unlinkByLinkId（リンク識別子の指定によるリンク先オブジェクトとのリンクの解除）

(1) 機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトのリンク元オブジェクトから複数のリンク先オブジェクトのリンク識別子を指定してリンクを解除します。なお，すでにリンクが解除されていたため，メソッドの実行によってリンクが解除されなかったリンクのリンク識別子の要素が返却されます。

(2) 形式

```
List unlinkByLinkId(
     List<String>    linkIds
)
```

(3) 引数

linkIds（入力）

要素には，解除するリンク識別子（String 型）のリストを指定します。null を指定すると例外がスローされます。

(4) 戻り値

すでにリンクが解除されていたため，メソッドの実行によって，リンクが解除されなかったリンクのリンク識別子（String 型）のリスト

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 linkIds が null の場合

6.4.38 uploadContents（文書のコンテンツのアップロード）

（1）機能

指定したレンディションタイプのコンテンツをアップロードして更新します。アップロードの対象になるターゲットオブジェクトは，文書だけです。

アップロード情報に指定するレンディションタイプとアップロード対象に指定するレンディションタイプに異なるレンディションを指定した場合，アップロードするコンテンツのレンディションタイプを変更できます。この場合，アップロード情報に指定したレンディションタイプがすでに文書に存在していると，例外がスローされます。アップロード情報の指定でレンディションタイプに null を設定した場合は，レンディションタイプは変更されません。

（2）形式

```
void uploadContents(
     String               renditionType,
     DbjUploadInfo        uploadInfo
)
```

（3）引数

renditionType（入力）

更新対象のレンディションタイプを指定します。null を指定するとマスタレンディションが仮定されます。

なお，全文検索インデクスを作成できるのは，null を指定した場合だけです。null 以外を指定した場合，全文検索インデクスは作成されません。

uploadInfo（入力）

文書に登録するファイルのアップロード情報を指定します。null を指定すると例外がスローされます。

（4）戻り値

なし

（5）例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjIllegalPropValException

指定レンディションプロパティ値が不正の場合

DbjIOException

DocumentBroker クラスライブラリ固有の IO エラーの場合

DbjMasterRenditionNotSetException

マスタレンディションのレンディションタイプが設定されていなかった場合

DbjNotCheckOutException
チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjRenditionNotFoundException
指定レンディションが存在しなかった場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalArgumentException
引数に不正な値が指定されていた場合

NullPointerException
引数 uploadInfo が null の場合

6.4.39 writeProperties（文書管理オブジェクトのプロパティ値の設定）

（1） 機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトのプロパティ値を設定します。

形式 1 では，Proxy オブジェクトがアクセス対象とする文書管理オブジェクトのプロパティを，指定したプロパティ値集合で設定します。この場合，Proxy オブジェクトのプロパティ値集合は無視されます。形式 2 および形式 3 では，Proxy オブジェクトに設定されているプロパティ値集合で文書管理オブジェクトのプロパティを設定します。形式 2 では，DbjObj#readProperties メソッドで読み込んだプロパティ値集合のプロパティ値を書き換えて文書管理オブジェクトのプロパティ値を更新します。形式 3 では，Proxy オブジェクトに設定されているプロパティのうち，文書管理オブジェクトに反映するプロパティを限定して指定できます。なお，形式 3 の場合，指定した引数の値によって Proxy オブジェクトの保持するプロパティ値集合は変更されません。

（2） 形式

（a） 形式 1

```
void writeProperties(
     DbjPropSet     propSet
)
```

（b） 形式 2

```
void writeProperties()
```

（c） 形式 3

```
void writeProperties(
     Collection<String>     propDefs
)
```

（3） 引数

propSet（入力）
: 文書管理オブジェクトに設定するプロパティ値集合を指定します。null を指定すると例外がスローされます。

propDefs（入力）
: 要素はプロパティ名（String 型）です。文書管理オブジェクトに設定する対象になるプロパティ値のコレクションを指定します。Proxy オブジェクトのプロパティ値のコレクションに存在しないプロパティを指定した場合は無視されます。null を指定すると例外がスローされます。

（4） 戻り値

なし

（5） 例外

DbjAccessControlException
: アクセス権エラーの場合

DbjACEOperationException
: 指定 ACE についてのエラーの場合

DbjDBException
: DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalPropValException
　指定プロパティ値が不正の場合

DbjNotCheckOutException
　チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
　指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
　指定パブリック ACL が存在しなかった場合

DbjPublicACLOutOfRangeException
　指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

NullPointerException
　引数 propSet，または propDefs が null の場合

6.4.40 writeRenditionProperties（レンディションプロパティの設定）

（1）機能

Proxy オブジェクトのアクセス対象である文書の指定レンディションについて，レンディションプロパティを設定します。レンディションプロパティを設定できるターゲットオブジェクトは，文書だけです。

（2）形式

```
void writeRenditionProperties(
     String              renditionType,
     DbjPropSet          propSet
)
```

（3）引数

renditionType（入力）

設定対象のレンディションタイプを指定します。null を指定するとマスタレンディションが仮定されます。

propSet（入力）

設定するレンディションのプロパティ値集合を指定します。null を指定すると例外がスローされます。

（4）戻り値

なし

（5）例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjIllegalPropValException

指定プロパティ値が不正の場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjRenditionNotFoundException

指定レンディションが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 propSet が null の場合

6.5 DbjObjList インターフェース

DbjObjList インターフェースは，複数の文書管理オブジェクトに一括してアクセスするインターフェースです。要素は，DbjObj インターフェースです。

スーパーインターフェース

- java.util.List

以降，DbjObjList インターフェースのメソッドについて説明します。

6.5.1 getObj（リスト要素の取得）

(1) 機能

インデクスで指定したリスト要素（DbjObj インターフェース）を取得します。

(2) 形式

```
DbjObj getObj(
     int     index
)
```

(3) 引数

index（入力）
　要素のインデクスを指定します。

(4) 戻り値

DbjObj インターフェース

(5) 例外

ClassCastException
　リスト要素が DbjObj インターフェースでなかった場合

IndexOutOfBoundException
　指定したインデクスが不正の場合

6.5.2 lock（アクセスロック種別の異なる複数文書管理オブジェクトインターフェースの取得）

（1）機能

アクセス対象の文書空間に対して，指定したアクセスロック種別のロックを保持する DbjObjList インターフェースを取得します。取得されたインターフェースには，要素の文書管理オブジェクトインターフェースのロックの指定よりも優先してロックが指定されます。なお，デフォルトでは，各要素の文書管理オブジェクトインターフェースのロック種別が有効になります。

（2）形式

```
DbjObjList lock(
     int     lockType
)
```

（3）引数

lockType（入力）

ロック種別として，次に示す定数のどちらかを指定します。

- DbjDef.LOCK_READ
  read ロック
- DbjDef.LOCK_WRITE
  write ロック

（4）戻り値

複数文書管理オブジェクトインターフェース（DbjObjList インターフェース）

（5）例外

なし

6.5.3 move（直接型リンクが設定されている文書管理オブジェクトの一括移動）

(1) 機能

リスト要素の文書管理オブジェクトに対してリンク元フォルダから直接型リンクが設定されている場合，そのリンクを解除して引数で指定したリンク元フォルダに直接型リンクを設定します。リスト要素の文書管理オブジェクトに直接型リンクが設定されていない場合は，指定したリンク元フォルダに直接型リンクを設定します。リスト要素の文書管理オブジェクトに直接型リンク以外のリンクが設定されている場合は，そのリンクは解除されます。

(2) 形式

```
void move(
     DbjObj     parentFolder
)
```

(3) 引数

parentFolder（入力）

要素オブジェクトが移動するリンク元フォルダを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalObjectTypeException

指定オブジェクト種別が不適切の場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 parentFolder が null の場合

6.5.4 readProperties（複数の文書管理オブジェクトプロパティ値の一括取得）

（1） 機能

Proxy オブジェクトのアクセス対象である文書管理オブジェクトの指定プロパティを，リストの各要素の Proxy オブジェクトに読み込みます。リストの要素の各 DbjObj インターフェースに対して，同じ形式の DbjObj#readProperties メソッドを実行します。

形式 1 では，Proxy オブジェクトの保持しているプロパティ値集合でのすべてのプロパティを文書管理オブジェクトから読み込みます。形式 2 では，読み込み対象のプロパティのコレクションを指定します。

（2） 形式

（a） 形式 1

```
void readProperties()
```

（b） 形式 2

```
void readProperties(
    Collection<String>     propDefs
)
```

（3） 引数

propDefs（入力）

要素はプロパティ名（String 型）です。取得するプロパティのプロパティ名のコレクションを指定します。null を指定すると例外がスローされます。

（4） 戻り値

なし

（5） 例外

DbjAccessControlException

アクセス権エラーの場合

DbjDBException

DB エラーの場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjNotCheckOutException

チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException

指定オブジェクトが存在しなかった場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 propDefs が null の場合

6.5.5 removeObjects（文書管理オブジェクトの一括削除）

(1) 機能

リストオブジェクトの各要素の Proxy オブジェクトのアクセス対象の文書管理オブジェクトを一括して削除します。リストの要素の各 DbjObj インターフェースに対して，DbjObj#removeObject メソッドを実行します。

(2) 形式

(a) 形式 1

```
List<DbjObj> removeObjects()
```

(b) 形式 2

```
List<DbjObj> removeObjects(
     boolean       isAlsoChildren
)
```

(3) 引数

isAlsoChildren(**入力**)

リンク先の文書管理オブジェクトの削除を指定します。true を指定すると，リンク先の文書管理オブジェクトが削除されます。この場合，直接型リンクが設定されている下位のすべての文書管理オブジェクトが削除されます。false を指定すると，リンク先の文書管理オブジェクトは削除されません。

(4) 戻り値

削除されたリスト要素の各 DbjObj インターフェースのリスト

(5) 例外

DbjAccessControlException
　アクセス権エラーの場合

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjLastVersionException
　最後のバージョンを削除しようとした場合

DbjNotCheckOutException
　チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.5.6 setPropSet（要素のプロパティ値集合の設定）

（1）機能

Proxy オブジェクトのプロパティ値集合の全要素に，指定したプロパティ値集合の内容をコピーして設定します。

（2）形式

```
void setPropSet(
     DbjPropSet     propSet
)
```

（3）引数

propSet（入力）

プロパティ値集合インターフェースを指定します。null を指定すると，オブジェクトのプロパティ値集合には空値が仮定されます。

（4）戻り値

なし

（5）例外

なし

6.5.7 writeProperties（複数の文書管理オブジェクトプロパティ値の一括設定）

(1) 機能

リスト要素の Proxy オブジェクトのアクセス対象である文書管理オブジェクトのプロパティを一括して設定します。

形式 1 では，要素のアクセス対象である文書管理オブジェクトのプロパティを，指定したプロパティ値集合で設定します。この場合，要素の Proxy オブジェクトのプロパティ値集合は無視されます。形式 2 および形式 3 では，要素のアクセス対象である文書管理オブジェクトのプロパティを各要素の Proxy オブジェクトに設定されたプロパティ値集合で設定します。形式 2 では，DbjObj#readProperties メソッドで読み込んだプロパティ値集合のプロパティ値を書き換えて文書管理オブジェクトのプロパティ値を更新します。形式 3 では，Proxy オブジェクトに設定されているプロパティのうち，要素のアクセス対象である文書管理オブジェクトに反映するプロパティを限定して指定できます。なお，形式 3 の場合，指定した引数の値によって Proxy オブジェクトの保持するプロパティ値集合は変更されません。

(2) 形式

(a) 形式 1

```
void writeProperties(
     DbjPropSet     propSet
)
```

(b) 形式 2

```
void writeProperties()
```

(c) 形式 3

```
void writeProperties(
     Collection<String>     propdef
)
```

(3) 引数

propSet（入力）
: 文書管理オブジェクトに設定するプロパティ値集合を指定します。null を指定すると例外がスローされます。

propDefs（入力）
: 要素はプロパティ名（String 型）です。文書管理オブジェクトに設定する対象のプロパティ値のコレクションを指定します。null を指定すると例外がスローされます。

(4) 戻り値

なし

(5) 例外

DbjAccessControlException
: アクセス権エラーの場合

DbjACEOperationException
: 指定 ACE についてのエラーの場合

DbjDBException

DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalPropValException
指定プロパティ値が不正の場合

DbjNotCheckOutException
チェックアウトされていないオブジェクトに対して仮のバージョン識別子が指定された場合

DbjObjectNotFoundException
指定オブジェクトが存在しなかった場合

DbjPublicACLNotFoundException
指定パブリック ACL は存在しなかった場合

DbjPublicACLOutOfRangeException
指定パブリック ACL 数が範囲を超えた場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

NullPointerException
引数 propSet，または propDefs が null の場合

6.6 DbjSession インターフェース

DbjSession インターフェースは，DocumentBroker クラスライブラリのセッションを扱います。DocumentBroker クラスライブラリの一つのセッションオブジェクトは文書空間への一つの接続に対応し，その接続している期間をセッションといいます。DbjSession インターフェースは，同時に一つのセッションのセッション管理をサポートします。ユーザアプリケーションプログラムが複数のセッションを実現するためには，複数の DbjSession インターフェースを扱う必要があります。

以降，DbjSession インターフェースのメソッドについて説明します。

6.6.1 begin（トランザクションの開始）

(1) 機能

トランザクションを開始します。このメソッドを二重に実行した場合は例外がスローされます。

(2) 形式

```
void begin()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalStateException
DbjSession#begin メソッドが二重に実行された場合

6.6.2 checkSession（セッションが有効かどうかのチェック）

(1) 機能

接続中の文書空間に対して有効なセッションがあるかどうかを調べます。無効なセッションを次に示します。

- 切断（logout）しているセッション
- 確立（login）していないセッション
- 文書空間との接続がタイムアウトしているセッション

(2) 形式

```
boolean checkSession()
```

(3) 引数

なし

(4) 戻り値

true
　有効なセッションです。

false
　無効なセッションです。

(5) 例外

なし

6.6.3 commit（トランザクションの確定）

(1) 機能

トランザクションを確定します。トランザクションが開始されていない場合，例外がスローされます。

(2) 形式

```
void commit()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

DbjDBException
DB エラーの場合

DbjException
DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException
セッションが接続されていなかった場合

IllegalStateException
DbjSession#begin メソッドが実行されていないのに DbjSession#commit メソッドが実行された場合

6.6.4 getLoginUserInfo（ユーザ情報の取得）

（1） 機能

ログインしているユーザのユーザ情報を取得します。

（2） 形式

```
DbjPropSet getLoginUserInfo(
     Collection<String>     propDefs
)
```

（3） 引数

propDefs（入力）

要素はプロパティ名（String 型）です。取得したいユーザ情報プロパティのプロパティ名を指定します。

（4） 戻り値

ユーザ情報プロパティ値集合

（5） 例外

DbjAccessControlNotSupportedException

接続先の文書空間がアクセス制御動作モードに対応していなかった場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException

セッションが接続されていなかった場合

NullPointerException

引数 propDefs が null の場合

6.6.5 getReferencePath（コンテンツ格納先ベースパスの取得）

（1） 機能

リファレンスファイル管理機能を使用して，コンテンツを任意の領域に格納する場合，
DbjSession#setReferencePath メソッドで設定したコンテンツ格納先ベースパスを取得します。

（2） 形式

```
String getReferencePath()
```

（3） 引数

なし

（4） 戻り値

コンテンツの格納先となるベースパス

（5） 例外

DbjSessionNotConnectException
　セッションが接続されていなかった場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

6.6.6 login（ログイン）

（1） 機能

文書空間にログインしてセッションを確立し，同時にユーザ識別子およびパスワードの認証を実行して，文書空間インターフェース（DbjDocSpace インターフェース）を取得します。このメソッドは，一つの文書空間にアクセスするほかのメソッドの実行前に実行する必要があります。確立しているセッションの有効期限は，DbjSession#logout メソッドを実行するまで，または文書空間への接続をタイムアウト（DocumentBroker で設定される）するまでです。なお，ユーザ認証の失敗，および二重のログインがあった場合は，例外がスローされます。

（2） 形式

```
DbjDocSpace login(
    String       userName,
    String       passWord
)
```

（3） 引数

userName（入力）
　ログインするユーザのユーザ識別子を指定します。

passWord（入力）
　ログインするユーザのパスワードを指定します。

（4） 戻り値

文書管理オブジェクトアクセスインターフェース（DbjDocSpace インターフェース）

（5） 例外

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjIllegalDocSpaceIdException
　接続先の文書空間識別子が不正の場合

DbjNotAuthenticatedException
　認証エラーの場合

IllegalStateException
　すでにログインしていた場合

6.6.7 logout（ログアウト）

（1）機能

文書空間からログアウトしてセッションを切断します。セッションの切断によって，ログインセッション内で継続されて使用される内部データ（検索結果キャッシュなど）は消去されます。また，切断するセッションに未確定のトランザクションがある場合は，自動的にトランザクションが取り消されて最後にトランザクションが確定された状態まで戻ります。

（2）形式

```
void logout()
```

（3）引数

なし

（4）戻り値

なし

（5）例外

なし

6.6.8 rollback（トランザクションの取り消し）

(1) 機能

トランザクションを取り消して，最後に DbjSession#commit メソッドが実行された状態まで戻します。
トランザクションが開始されていなくても正常終了します。

(2) 形式

```
void rollback()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

DbjDBException
　DB エラーの場合

DbjException
　DocumentBroker クラスライブラリ固有のエラーの場合

DbjSessionNotConnectException
　セッションが接続されていなかった場合

6.6.9 setReferencePath（コンテンツ格納先ベースパスの設定）

（1）機能

リファレンスファイル管理機能を使用して，コンテンツを任意の領域に格納する場合に，コンテンツ格納先ベースパスを設定します。

（2）形式

```
void setReferencePath( String referencePath )
```

（3）引数

referencePath（入力）

リファレンスファイル管理機能で，コンテンツの格納先となるベースパスを指定します。

（4）戻り値

なし

（5）例外

DbjSessionNotConnectException

セッションが接続されていなかった場合

DbjException

DocumentBroker クラスライブラリ固有のエラーの場合

6.6.10 getConnection（JDBC コネクションの取得）

(1) 機能

ユーザプログラムで扱うデータと DocumentBroker が扱うデータの同期をとるために，トランザクションを開始している JDBC コネクションを取得します。トランザクションをコミットまたはロールバックする場合，DbjSession#commit または DbjSession#rollback メソッドを経由して行ってください。なお，ユーザプログラムで，このコネクションを使用して DocumentBroker が提供しているテーブルを操作しないでください。

(2) 形式

```
java.sql.Connection getConnection()
```

(3) 引数

なし

(4) 戻り値

JDBC コネクション

(5) 例外

DbjSessionNotConnectException
　セッションが接続されていなかった場合

IllegalStateException
　begin されていないのに呼び出された場合

6.6.11 changeUserPrivilege（文書空間に接続中のユーザの特権変更）

（1）機能

指定した特権の種別に従って，文書空間に接続中のユーザの特権を変更します。このメソッドを使用することで，クラスライブラリを使用した一連の処理中に，任意の回数／タイミングでユーザの特権を変更することができます。このメソッドで変更した特権は，DbjSession#logout メソッドが実行されるまで有効となります。

（2）形式

```
void changeUserPrivilege(int privilege)
```

（3）引数

privilege（入力）

特権の種別を指定します。次のどちらかの値を指定します。

- DbjDef.PRIV_SECURITY_ADMINISTRATOR
  セキュリティ管理者に変更します。
- DbjDef.PRIV_LOGIN_USER
  DbjSession#login 時のユーザの特権に変更します。

（4）戻り値

なし

（5）例外

DbjSessionNotConnectException

セッションが接続されていなかった場合

DbjAccessControlNotSupportedException

アクセス制御モードで動作していない場合

IllegalArgumentException

引数に不正な値を設定した場合

6.7 DbjVerObj インターフェース

DbjVerObj インターフェースは，一つの文書管理オブジェクトへのアクセスを扱うことに加えて，Proxy オブジェクトのターゲットオブジェクトであるバージョンオブジェクトのバージョン識別子を扱うインターフェースです。

スーパーインターフェース

- DbjObj

プロパティー覧

DbjVerObj インターフェースが扱うプロパティと getter メソッドを次の表に示します。

表 6-3 DbjVerObj インターフェースで扱うプロパティ

プロパティ名	データ型	getter メソッド
バージョン識別子	String	getVersionId

以降，DbjVerObj インターフェースのメソッドについて説明します。

6.7.1 getVersionId（文書管理オブジェクトのバージョン識別子の取得）

（1）機能

文書管理オブジェクトに設定されている Proxy オブジェクトのターゲットオブジェクトであるバージョンオブジェクトのバージョン識別子のプロパティ値（String 型）を取得します。

（2）形式

```
String getVersionId()
```

（3）引数

なし

（4）戻り値

バージョンオブジェクトのバージョン識別子のプロパティ値（versionId フィールドの値）

（5）例外

なし

6.8 DbjVerObjList インターフェース

DbjVerObjList インターフェースは，DbjVerObj インターフェースを要素とするリストインターフェースを扱います。

スーパーインターフェース

- DbjObjList

以降，DbjVerObjList インターフェースのメソッドについて説明します。

6.8.1 getVerObj（要素の DbjVerObj インターフェースの取得）

（1）機能

インデクスで指定された要素の DbjVerObj インターフェースを取得します。

（2）形式

```
DbjVerObj getVerObj(
     int      index
)
```

（3）引数

index（入力）

要素インデクスを指定します。

（4）戻り値

要素の DbjVerObj インターフェース

（5）例外

ClassCastException

指定要素が DbjObj インターフェースでなかった場合

IndexOutOfBoundException

指定したインデクスが不正の場合

6.8.2 getVersionIdList（文書管理オブジェクトのバージョン識別子リストの取得）

（1）機能

文書管理オブジェクトが保持する要素すべてについて，Proxy オブジェクトのターゲットオブジェクトであるバージョンオブジェクトのバージョン識別子のプロパティ値をリストで取得します。

（2）形式

```
List<String> getVersionIdList()
```

（3）引数

なし

（4）戻り値

バージョンオブジェクトのバージョン識別子のプロパティ値（versionId フィールドの値）のリスト（String 型のリスト）

（5）例外

ClassCastException
　指定要素が DbjObj インターフェースでなかった場合

7 メタクラス詳細

この章では，メタクラスのインターフェース，およびメソッドについて説明します。

7.1 DbjClassDesc インターフェース

DbjClassDesc インターフェースは，一つの文書管理オブジェクトクラスのメタ情報（クラスディスクリプション）を扱うインターフェースです。

以降，DbjClassDesc インターフェースのメソッドについて説明します。

7.1.1 getName（クラス名の取得）

(1) 機能

クラス名を取得します。

(2) 形式

```
String getName()
```

(3) 引数

なし

(4) 戻り値

クラス名

(5) 例外

なし

7.1.2 getProperties（プロパティディスクリプションの取得）

(1) 機能

クラスが保持するプロパティディスクリプションをすべて取得します。

(2) 形式

```
List<DbjPropDesc> getProperties()
```

(3) 引数

なし

(4) 戻り値

プロパティディスクリプション（DbjPropDesc インターフェース）を要素とするリスト

(5) 例外

なし

7.1.3 getSubClasses（サブクラスのクラスディスクリプションの取得）

(1) 機能

直接のサブクラスのクラスディスクリプションをすべて取得します。

サブクラスがない場合は，要素が空のリストが返却されます。

(2) 形式

```
List<DbjClassDesc> getSubClasses()
```

(3) 引数

なし

(4) 戻り値

クラスディスクリプション（DbjClassDesc インターフェース）を要素とするリスト

(5) 例外

なし

7.1.4 getSuperClass（スーパークラスのクラスディスクリプションの取得）

(1) 機能

直接のスーパークラスのクラスディスクリプションを取得します。

スーパークラスがない場合は，null が返却されます。

(2) 形式

```
DbjClassDesc getSuperClass()
```

(3) 引数

なし

(4) 戻り値

スーパークラスのクラスディスクリプション（DbjClassDesc インターフェース）

(5) 例外

なし

7.2 DbjMeta インターフェース

DbjMeta インターフェースは，一つの文書管理のメタ情報を扱うインターフェースです。

以降，DbjMeta インターフェースのメソッドについて説明します。

7.2.1 getClassDesc（指定したクラスのクラスディスクリプションの取得）

(1) 機能

指定したクラスのクラスディスクリプションを取得します。

(2) 形式

```
DbjClassDesc getClassDesc(
     String              className
)
```

(3) 引数

className（入力）

クラス名を指定します。
null を指定した場合は，例外がスローされます。

(4) 戻り値

指定したクラスのクラスディスクリプション（DbjClassDesc インターフェース）

(5) 例外

IllegalArgumentException
指定したクラスがなかった場合

NullPointerException
引数 className に null を指定した場合

7.2.2 getDocSpaceId（文書空間識別子の取得）

(1) 機能

文書空間識別子を取得します。

(2) 形式

```
String getDocSpaceId()
```

(3) 引数

なし

(4) 戻り値

文書空間識別子

(5) 例外

なし

7.2.3 getExtFromRenditionType（レンディションタイプに対応する拡張子の取得）

（1） 機能

レンディション定義ファイルの定義内容に従って，指定したレンディションタイプに対応する拡張子を取得します。

指定したレンディションタイプがレンディション定義ファイルに定義されていない場合は，null が返却されます。また，指定したレンディションタイプに対応する拡張子が複数定義されている場合は，その中のどれかの拡張子が返却されます。

（2） 形式

```
String getExtFromRenditionType(
     String                    renditionType
)
```

（3） 引数

renditionType（入力）

レンディションタイプを指定します。
null を指定した場合は，例外がスローされます。

（4） 戻り値

指定したレンディションタイプに対応する拡張子

（5） 例外

NullPointerException

引数 renditionType に null を指定した場合

7.2.4 getPropDataType（指定したプロパティのデータ型の取得）

(1) 機能

指定したプロパティのデータ型を取得します。

(2) 形式

```
int getPropDataType(
     String         propName
)
```

(3) 引数

propName（入力）

プロパティ名を指定します。
null を指定した場合は，例外がスローされます。

(4) 戻り値

指定したプロパティのデータ型が，次に示す定数で返却されます。

- DbjDef.DATATYPE_BOOL
  BOOL 型
- DbjDef.DATATYPE_INT
  INT 型
- DbjDef.DATATYPE_STR
  STR 型
- DbjDef.DATATYPE_STRLIST
  STRLIST 型
- DbjDef.DATATYPE_VARRAY
  VARRAY 型

(5) 例外

IllegalArgumentException

指定したプロパティがなかった場合

NullPointerException

引数 propName に null を指定した場合

7.2.5 getPropDesc（指定したプロパティのプロパティディスクリプションの取得）

（1）機能

指定したプロパティのプロパティディスクリプションを取得します。

（2）形式

```
DbjPropDesc getPropDesc(
     String            propName
)
```

（3）引数

propName（入力）

プロパティ名を指定します。
null を指定した場合は，例外がスローされます。

（4）戻り値

指定したプロパティのプロパティディスクリプション（DbjPropDesc インターフェース）

（5）例外

IllegalArgumentException

指定したプロパティがなかった場合

NullPointerException

引数 propName に null を指定した場合

7.2.6 getRenditionType（拡張子に対応するレンディションタイプの取得）

(1) 機能

レンディション定義ファイルの定義内容に従って，指定した拡張子に対応するレンディションタイプを取得します。

指定した拡張子に対応するレンディションタイプがレンディション定義ファイルに定義されていない場合は，null が返却されます。

(2) 形式

```
String getRenditionType(
     String            extension
)
```

(3) 引数

extension（入力）

ファイルの拡張子を指定します。指定する拡張子の先頭に「.」（ピリオド）は不要です。
null を指定した場合は，例外がスローされます。

(4) 戻り値

指定した拡張子に対応するレンディションタイプ（「MIME::」は含みません）

(5) 例外

NullPointerException

引数 extension に null を指定した場合

7.3 DbjMetaManager インターフェース

DbjMetaManager インターフェースは，文書空間のメタ情報を管理するためのインターフェースです。

以降，DbjMetaManager インターフェースのメソッドについて説明します。

7.3.1 getMeta（メタ情報の取得）

(1) 機能

指定した文書空間のメタ情報を取得します。メタ情報は DbjMeta インターフェースで返却されます。

(2) 形式

```
DbjMeta getMeta(
     String    docspaceId
)
```

(3) 引数

docspaceId（入力）

文書空間識別子を GUID 文字列で指定します。GUID 文字列は，16 進数「X」によって，「XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX（8 けた -4 けた -4 けた -4 けた -12 けた）」の形式で表されます。「X」は，0 ～ 9，a ～ f（小文字），および A ～ F（大文字）のどれかです。null を指定した場合は，例外がスローされます。

(4) 戻り値

指定した文書空間のメタ情報（DbjMeta インターフェース）

(5) 例外

IllegalArgumentException

指定した文書空間がなかった場合

NullPointerException

引数 docspaceId に null を指定した場合

7.4 DbjPropDesc インターフェース

DbjPropDesc インターフェースは，文書管理オブジェクトの一つのプロパティのメタ情報（プロパティディスクリプション）を扱うインターフェースです。

以降，DbjPropDesc インターフェースのメソッドについて説明します。

7.4.1 getDataType（プロパティのデータ型の取得）

(1) 機能

プロパティのデータ型を取得します。

(2) 形式

```
int getDataType()
```

(3) 引数

なし

(4) 戻り値

プロパティのデータ型が，次に示す定数で返却されます。

- DbjDef.DATATYPE_BOOL
  BOOL 型
- DbjDef.DATATYPE_INT
  INT 型
- DbjDef.DATATYPE_STR
  STR 型
- DbjDef.DATATYPE_STRLIST
  STRLIST 型
- DbjDef.DATATYPE_VARRAY
  VARRAY 型

(5) 例外

なし

7.4.2 getName（プロパティ名の取得）

(1) 機能

プロパティ名を取得します。

(2) 形式

```
String getName()
```

(3) 引数

なし

(4) 戻り値

プロパティ名

(5) 例外

なし

7.4.3 getVArrayClass（VARRAY 型プロパティを扱うクラスのクラスディスクリプションの取得）

(1) 機能

VARRAY 型プロパティを扱うクラスのクラスディスクリプションを取得します。

VARRAY 型プロパティを扱うクラスでない場合は，null が返却されます。

(2) 形式

```
DbjClassDesc getVArrayClass()
```

(3) 引数

なし

(4) 戻り値

VARRAY 型プロパティを扱うクラスのクラスディスクリプション（DbjClassDesc インターフェース）

(5) 例外

なし

8 例外クラス詳細

この章では，例外クラスの詳細について説明します。

8.1 例外クラスの詳細

DocumentBroker クラスライブラリの API でエラーが発生すると，例外がスローされます。例外クラスは，DocumentBroker クラスライブラリ固有の例外を扱うクラス群です。ここでは，それぞれの例外クラスの機能，例外クラスのスーパークラス，メソッド（コンストラクタ）一覧を説明します。なお，例外クラスのコンストラクタについては，「3.5.3　例外クラスのコンストラクタ」を参照してください。

8.2 DbjAccessControlException クラス

DbjAccessControlException クラスは，ユーザのメソッドの実行時に，その操作に対してユーザにアクセス権がない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjAccessControlException()

DbjAccessControlException(String s)

8.3 DbjAccessControlNotSupportedException クラス

DbjAccessControlNotSupportedException クラスは，文書空間がアクセス制御機能に対応していない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjAccessControlNotSupportedException()

DbjAccessControlNotSupportedException(String s)

8.4 DbjACEOperationException クラス

DbjACEOperationException クラスは，ACE の操作エラーを表すスーパークラスです。特定の ACE 操作のエラーは，サブクラスがスローされます。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjACEOperationException()

DbjACEOperationException(String s)

8.5 DbjACLOutOfRangeException クラス

DbjACLOutOfRangeException クラスは，文書管理オブジェクトの ACE の個数が制限値を超えて追加された場合のエラーを表します。

(1) スーパークラス

DbjACEOperationException

(2) コンストラクター覧

DbjACLOutOfRangeException()

DbjACLOutOfRangeException(String s)

8.6 DbjAlreadyCheckOutException クラス

DbjAlreadyCheckOutException クラスは，すでにチェックアウトされている場合のエラーを表します。

(1) スーパークラス

DbjCheckOutException

(2) コンストラクター覧

DbjAlreadyCheckOutException()

DbjAlreadyCheckOutException(String s)

8.7 DbjCheckOutException クラス

DbjCheckOutException クラスは，チェックアウトについてのエラーを表すスーパークラスです。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjCheckOutException()

DbjCheckOutException(String s)

8.8 DbjContentNotRegisteredException クラス

DbjContentNotRegisteredException クラスは，リファレンスファイル機能を使用したオブジェクトに DbjObj#downloads() を実行したとき，対象オブジェクトにコンテンツが登録されていない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjContentNotRegisteredException ()

DbjContentNotRegisteredException (String s)

8.9 DbjContentTypeMismatchException クラス

DbjContentTypeMismatchException クラスはリファレンスファイル機能で，コンテンツの種別がメソッドが要求するコンテンツの種別と異なる場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjContentTypeMismatch()

DbjContentTypeMismatch (String s)

8.10 DbjConvertContentTargetNotFoundException クラス

DbjConvertContentTargetNotFoundException クラスは，変換対象のコンテンツが存在しない場合のエラーを表します。

(1) スーパークラス

DbjDBException

(2) コンストラクター覧

DbjConvertContentTargetNotFoundException ()

DbjConvertContentTargetNotFoundException (String s)

8.11 DbjDBDeadLockException クラス

DbjDBDeadLockException クラスは，DB のデッドロックエラーを表します。

(1) スーパークラス

DbjDBException

(2) コンストラクター一覧

DbjDBDeadLockException()

DbjDBDeadLockException(String s)

8.12 DbjDBException クラス

DbjDBException クラスは，DB エラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjDBException()

DbjDBException(String s)

8.13 DbjDBLockTimeoutException クラス

DbjDBLockTimeoutException クラスは，DB のロックタイムアウトエラーを表します。

(1) スーパークラス

DbjDBException

(2) コンストラクター一覧

DbjDBLockTimeoutException()

DbjDBLockTimeoutException(String s)

8.14 DbjDisconnectedSessionException クラス

DbjDisconnectedSessionException クラスは，文書空間との接続が切断された場合のエラーを表します。

(1) スーパークラス

DbjSessionNotConnectException

(2) コンストラクター覧

DbjContentTypeMismatch()

DbjContentTypeMismatch (String s)

8.15 DbjError クラス

DbjError クラスは java.lang.Error クラスを継承し，DocumentBroker クラスライブラリ固有の致命的なエラーを表すスーパークラスです。メモリ不足，DocumentBroker クラスライブラリの初期化エラー，DocumentBroker クラスライブラリ内で予期しない内部エラーが発生した場合のエラーなど，ユーザアプリケーションプログラムで対処できないエラーを表します。

DbjError クラスに関連付けられたエラーメッセージは次の内容を含みます。

- エラー内容，または原因コード
- エラー発生メソッド
- エラー発生個所（ソース行番号，ソースファイル名）

(1) スーパークラス

java.lang.Error

(2) コンストラクター覧

DbjError()

DbjError(String s)

8.16 DbjException クラス

DbjException クラスは java.lang.Exception クラスを継承し，DocumentBroker クラスライブラリ固有のエラーを表すスーパークラスです。

(1) スーパークラス

java.lang.Exception

(2) コンストラクター覧

DbjException()

DbjException(String s)

8.17 DbjFileAccessException クラス

DbjFileAccessException クラスは，ファイルのアクセス権がない場合のエラーを表します。

(1) スーパークラス

DbjIOException

(2) コンストラクター一覧

DbjFileAccessException()

DbjFileAccessException(String s)

8.18 DbjFileNotFoundException クラス

DbjFileNotFoundException クラスは，指定ファイルが存在しない場合のエラーを表します。

(1) スーパークラス

DbjIOException

(2) コンストラクター一覧

DbjFileNotFoundException()

DbjFileNotFoundException(String s)

8.19 DbjFileReferenceCurrentContentNotfoundException クラス

DbjFileReferenceCurrentContentNotfoundException クラスは，リファレンスファイル管理機能を使用したバージョン付きの文書オブジェクトで，カレントバージョンにコンテンツが存在しない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjFileReferenceCurrentContentNotfoundException()

DbjFileReferenceCurrentContentNotfoundException(String s)

8.20 DbjFileReferenceMismatchStatusException クラス

DbjFileReferenceMismatchStatusException クラスは，リファレンスファイル管理機能を使用するサーバでのコンテンツ操作でエラーが発生し，オブジェクトとコンテンツが不整合な状態になった場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjFileReferenceMismatchStatusException()

DbjFileReferenceMismatchStatusException(String s)

8.21 DbjFileReferenceOperationFailedException クラス

DbjFileReferenceOperationFailedException クラスは，リファレンスファイル管理機能を使用するサーバでのコンテンツ操作が失敗した場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjFileReferenceOperationFailedException()

DbjFileReferenceOperationFailedException(String s)

8.22 DbjIllegalCacheStartIndexException クラス

DbjIllegalCacheStartIndexException クラスは，キャッシュ検索時に検索結果より大きい値を DbjFetchInfo インターフェースの startIndex プロパティへ指定した場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjIlleaglCacheStartIndexException()

DbjIlleaglCacheStartIndexException(String s)

8.23 DbjIllegalDocSpaceIdException クラス

DbjIllegalDocSpaceIdException クラスは，不正な文書空間識別子が指定された場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjIllegalDocSpaceIdException()

DbjIllegalDocSpaceIdException(String s)

8.24 DbjIllegalObjectTypeException クラス

DbjIllegalObjectTypeException クラスは，メソッドによる操作に不適切な文書管理オブジェクト種別が指定された場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjIllegalObjectTypeException()

DbjIllegalObjectTypeException(String s)

8.25 DbjIllegalPropValException クラス

DbjIllegalPropValException クラスは，不正なプロパティ値を指定した場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjIllegalPropValException()

DbjIllegalPropValException(String s)

8.26 DbjInitializeError クラス

DbjInitializeError クラスは，DocumentBroker クラスライブラリの初期化を実行した時のエラーを表します。

(1) スーパークラス

DbjError

(2) コンストラクター覧

DbjInitializeError()

DbjInitializeError(String s)

8.27 DbjInternalError クラス

DbjInternalError クラスは，DocumentBroker クラスライブラリの内部エラーを表します。

(1) スーパークラス

DbjError

(2) コンストラクター一覧

DbjInternalError()

DbjInternalError(String s)

8.28 DbjIOException クラス

DbjIOException クラスは，DocumentBroker クラスライブラリ固有の IO エラーを表すスーパークラスです。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjIOException()

DbjIOException(String s)

8.29 DbjIsMasterRenditionException クラス

DbjIsMasterRenditionException クラスは，マスタレンディションが指定された場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjIsMasterRenditionException()

DbjIsMasterRenditionException(String s)

8.30 DbjLastVersionException クラス

DbjLastVersionException クラスは，操作対象となるバージョン付きオブジェクトの最後の 1 バージョンを削除しようとした場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjLastVersionException()

DbjLastVersionException(String s)

8.31 DbjMasterRenditionNotSetException クラス

DbjMasterRenditionNotSetException クラスは，マスタレンディションのレンディションタイプが設定されていない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjMasterRenditionNotSetException()

DbjMasterRenditionNotSetException(String s)

8.32 DbjNotAuthenticatedException クラス

DbjNotAuthenticatedException クラスは，ユーザ認証エラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjNotAuthenticatedException()

DbjNotAuthenticatedException(String s)

8.33 DbjNotCheckOutException クラス

DbjNotCheckOutException クラスは，チェックアウトされていない場合のエラーを表します。

(1) スーパークラス

DbjCheckOutException

(2) コンストラクター一覧

DbjNotCheckOutException()

DbjNotCheckOutException(String s)

8.34 DbjNotLoginException クラス

DbjNotLoginException クラスは，文書空間にログインされていない場合のエラーを表します。

(1) スーパークラス

DbjSessionNotConnectException

(2) コンストラクター覧

DbjNotLoginException()

DbjNotLoginException(String s)

8.35 DbjNotSupportedException クラス

DbjNotSupportedException クラスは，サポートされていない操作を表します。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjNotSupportedException ()

DbjNotSupportedException (String s)

8.36 DbjObjectNotFoundException クラス

DbjObjectNotFoundException クラスは，操作対象となる文書管理オブジェクトがメソッド実行にすでに削除されていた場合，または存在しない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjObjectNotFoundException()

DbjObjectNotFoundException(String s)

8.37 DbjOutOfMemoryError クラス

DbjOutOfMemoryError クラスは，メモリ不足が発生した場合のエラーを表します。

(1) スーパークラス

DbjError

(2) コンストラクター一覧

DbjOutOfMemoryError()

DbjOutOfMemoryError(String s)

8.38 DbjPublicACLAlreadyBoundException クラス

DbjPublicACLAlreadyBoundException クラスは，指定したパブリック ACL がすでにバインドされている場合のエラーを表します。

(1) スーパークラス

DbjPublicACLOperationException

(2) コンストラクター一覧

DbjPublicACLAlreadyBoundException()

DbjPublicACLAlreadyBoundException(String s)

8.39 DbjPublicACLNotBoundException クラス

DbjPublicACLNotBoundException クラスは，指定したパブリック ACL がバインドされていない場合のエラーを表します。

(1) スーパークラス

DbjPublicACLOperationException

(2) コンストラクター覧

DbjPublicACLNotBoundException()

DbjPublicACLNotBoundException(String s)

8.40 DbjPublicACLNotFoundException クラス

DbjPublicACLNotFoundException クラスは，指定したパブリック ACL が存在しない場合のエラーを表します。

(1) スーパークラス

DbjPublicACLOperationException

(2) コンストラクター覧

DbjPublicACLNotFoundException()

DbjPublicACLNotFoundException(String s)

8.41 DbjPublicACLOperationException クラス

DbjPublicACLOperationException クラスは，パブリック ACL の操作エラーを表すスーパークラスです。特定のパブリック ACL 操作のエラーには，サブクラスがスローされます。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjPublicACLOperationException()

DbjPublicACLOperationException(String s)

8.42 DbjPublicACLOutOfRangeException クラス

DbjPublicACLOutOfRangeException クラスは，文書管理オブジェクトのパブリック ACL の個数が制限値を超えて追加された場合のエラーを表します。

(1) スーパークラス

DbjPublicACLOperationException

(2) コンストラクター一覧

DbjPublicACLOutOfRangeException()

DbjPublicACLOutOfRangeException(String s)

8.43 DbjReferenceTypeMismatchException クラス

DbjReferenceTypeMismatchException クラスは，リファレンスファイル管理機能で，オブジェクトのリファレンス種別とメソッドが要求するリファレンス種別が一致しない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjReferenceTypeMismatchException()

DbjReferenceTypeMismatchException(String s)

8.44 DbjRenditionConversionErrorExistedException クラス

DbjRenditionConversionErrorExistedException クラスは，変換エラーのレンディションが存在した場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjRenditionConversionErrorExistedException ()

DbjRenditionConversionErrorExistedException (String s)

8.45 DbjRenditionConversionRequiringExistedException クラス

DbjRenditionConversionRequiringExistedException クラスは，変換要求中のレイディションが存在した場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjRenditionConversionRequiringExistedException ()

DbjRenditionConversionRequiringExistedException (String s)

8.46 DbjRenditionCountOutOfRangeException クラス

DbjRenditionCountOutOfRangeException クラスは，レンディション数が制限値を超えた場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjRenditionCountOutOfRangeException()

DbjRenditionCountOutOfRangeException(String s)

8.47 DbjRenditionIsEmptyException クラス

DbjRenditionIsEmptyException クラスは，レンディションが空の場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjRenditionIsEmptyException()

DbjRenditionIsEmptyException(String s)

8.48 DbjRenditionNotConvertedException クラス

DbjRenditionNotConvertedException クラスは，レンディションが更新処理されていない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjRenditionNotConvertedException()

DbjRenditionNotConvertedException(String s)

8.49 DbjRenditionNotFoundException クラス

DbjRenditionNotFoundException クラスは，指定レンディションが存在しない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjRenditionNotFoundException()

DbjRenditionNotFoundException(String s)

8.50 DbjRenditionTypeDuplicatedException クラス

DbjRenditionTypeDuplicatedException クラスは，レンディションタイプが重複している場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjRenditionTypeDuplicatedException()

DbjRenditionTypeDuplicatedException(String s)

8.51 DbjSessionException クラス

DbjSessionException クラスは，セッションについてのエラーを表すスーパークラスです。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjSessionException()

DbjSessionException(String s)

8.52 DbjSessionNotConnectException クラス

DbjSessionNotConnectException クラスは，文書空間と接続されていない場合のエラーを表すスーパークラスです。

(1) スーパークラス

DbjSessionException

(2) コンストラクター覧

DbjSessionNotConnectException()

DbjSessionNotConnectException(String s)

8.53 DbjSubjectLengthOutOfRangeException クラス

DbjSubjectLengthOutOfRangeException クラスは，ACE のサブジェクトの長さが制限値を超えた場合のエラーを表します。

(1) スーパークラス

DbjACEOperationException

(2) コンストラクター覧

DbjSubjectLengthOutOfRangeException()

DbjSubjectLengthOutOfRangeException(String s)

8.54 DbjTargetContentPathInvalidException クラス

DbjTargetContentPathInvalidException クラスは，リファレンスファイル管理機能で，コンテンツ格納先ベースパスが不正な場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjTargetContentPathInvalidException ()

DbjTargetContentPathInvalidException (String s)

8.55 DbjTargetContentPathNotSetException クラス

DbjTargetContentPathNotSetException クラスは，リファレンスファイル管理機能で，コンテンツ格納先ベースパスが設定されていない場合のエラーを表します。

(1) スーパークラス

DbjException

(2) コンストラクター覧

DbjTargetContentPathNotSetException()

DbjTargetContentPathNotSetException(String s)

8.56 DbjUnknownError クラス

DbjUnknownError クラスは，DocumentBroker クラスライブラリ内での未知のエラーの発生を表します。

(1) スーパークラス

DbjError

(2) コンストラクター覧

DbjUnknownError()

DbjUnknownError(String s)

8.57 DbjUnexpectedException クラス

DbjUnexpectedException クラスは，内部矛盾が発生した場合のエラーを表すクラスです。

(1) スーパークラス

DbjException

(2) コンストラクター一覧

DbjUnexpectedException ()

DbjUnexpectedException (String s)

8.58 DbjVersionObjectNotFoundException クラス

DbjVersionObjectNotFoundException クラスは，バージョンオブジェクトが存在しない場合のエラーを表すクラスです。

(1) スーパークラス

DbjObjectNotFoundException

(2) コンストラクター覧

DbjVersionObjectNotFoundException ()

DbjVersionObjectNotFoundException (String s)

9 定数定義クラス詳細

この章では，定数定義クラスの定義内容について説明します。

9.1 DbjDef クラス

DbjDef クラスは，DocumentBroker クラスライブラリで使用する定数を定義したクラスです。

以降，DbjDef クラスの定数定義内容について，カテゴリごとに説明します。

なお，DbjDef クラスで定義されている定数のアクセス指定子は，すべて「public」です。

9.1.1 Category：ACL

検索実行時のアクセス制御モードを表す定数です。アクセス制御機能付き検索を実行するか，実行しないかを表します。

検索実行時のアクセス制御モードを表す定数を次の表に示します。

表 9-1 検索実行時のアクセス制御モードを表す定数

定数	データ型	意味
WITH_ACL	int 型	アクセス制御機能付き検索モード
WITHOUT_ACL	int 型	アクセス制御機能なし検索モード

9.1.2 Category：CONTENTTYPE

コンテンツ種別を表す定数です。

コンテンツ種別を表す定数を次の表に示します。

表 9-2 コンテンツ種別を表す定数

定数	データ型	意味
CONTENT	int 型	シングルファイル文書であるコンテンツ
REFERENCE	int 型	リファレンスファイル文書であるコンテンツ
OTHER	String 型	上記以外のコンテンツを持つオブジェクト

9.1.3 Category：DATATYPE

文書管理オブジェクトのプロパティのデータ型を表す定数です。

文書管理オブジェクトのプロパティのデータ型を表す定数を次の表に示します。

表 9-3 文書管理オブジェクトのプロパティのデータ型を表す定数

定数	データ型	意味
DATATYPE_BOOL	int 型	BOOL 型
DATATYPE_INT	int 型	INT 型
DATATYPE_STR	int 型	STR 型
DATATYPE_STRLIST	int 型	STRLIST 型
DATATYPE_VARRAY	int 型	VARRAY 型
DATATYPE_UNKNOWN	int 型	未サポートのデータ型，またはデータ型不明

9.1.4 Category：DMA BOOLEAN

BOOL 型を表す定数です。

BOOL 型を表す定数を次の表に示します。

表 9-4 BOOL 型を表す定数

定数	データ型	意味
DMA_TRUE	int 型	真
DMA_FALSE	int 型	偽
DMA_UNKNOWN	int 型	不定

9.1.5 Category：INDEXTYPE

全文検索インデクスの種別を表す定数です。

全文検索インデクスの種別を表す定数を次の表に示します。

表 9-5 全文検索インデクスの種別を表す定数

定数	データ型	意味
INDEXTYPE_NOTHING	int 型	全文検索インデクスを作成しません。インデクスデータは出力されません。

9.1.6 Category：LINK

リンク種別を表す定数です。

リンク種別を表す定数を次の表に示します。

表 9-6 リンク種別を表す定数

定数	データ型	意味
LINK_DCR	int 型	直接型リンク
LINK_RCR	int 型	参照型リンク
LINK_REL	int 型	文書間リンク
LINK_ALL	int 型	すべてのリンク種別を組み合わせたリンク
LINK_NONE	int 型	リンクなし

9.1.7 Category：LOCK

ロック種別を表す定数です。

ロック種別を表す定数を次の表に示します。

表 9-7 ロック種別を表す定数

定数	データ型	意味
LOCK_READ	int 型	read ロック
LOCK_WRITE	int 型	排他的な write ロック
LOCK_READFORUPDATE	int 型	read ロック（更新または削除ができます）
LOCK_NONE	int 型	ロックなし

9.1.8 Category：OBJTYPE

オブジェクト種別を表す定数です。

オブジェクト種別を表す定数を次の表に示します。

表 9-8 オブジェクト種別を表す定数

定数	データ型	意味
OBJTYPE_ANY	int 型	任意のオブジェクト種別の論理和
OBJTYPE_NVTFOLDER	int 型	構成管理できないバージョンなしフォルダ
OBJTYPE_DOC	int 型	バージョンなし文書
OBJTYPE_VRDOC	int 型	バージョン付き文書
OBJTYPE_FOLDER	int 型	バージョンなしフォルダ
OBJTYPE_IP	int 型	独立データ
OBJTYPE_PUBLICACL	int 型	パブリック ACL
OBJTYPE_UNKNOWN	int 型	オブジェクト種別不明

9.1.9 Category：OPERATEMODE

コンテンツのパス操作モードを表す定数です。

コンテンツのパス操作モードを表す定数を次の表に示します。

表 9-9 コンテンツのパス操作モードを表す定数

定数	データ型	意味
OPERATEMODE_NONE	int 型	コンテンツがないオブジェクトを作成します。
OPERATEMODE_USER_RELATIVE_CONTENT	int 型	コンテンツロケーションをコンテンツの相対パスで管理します。

9.1.10 Category：ORDER

バージョン情報の取得順序を表す定数です。

バージョン情報の取得順序を表す定数を次の表に示します。

表 9-10 バージョン情報の取得順序を表す定数

定数	データ型	意味
ORDER_ASC	int 型	最も古いバージョンから順に取得します。
ORDER_DESC	int 型	最も新しいバージョンから順に取得します。
ORDER_NONE	int 型	順序を指定しません。

9.1.11 Category：PERM

パーミッションを表す定数です。

パーミッションを表す定数を次の表に示します。

表 9-11 パーミッションを表す定数

定数	データ型	意味
PERM_NONE	int 型	アクセス権なし
PERM_PRIM_READ_PROPS	int 型	基本プロパティ参照権
PERM_PRIM_WRITE_PROPS	int 型	基本プロパティ更新権
PERM_PRIM_READ_CONTENTS	int 型	基本コンテンツ参照権
PERM_PRIM_WRITE_CONTENTS	int 型	基本コンテンツ更新権
PERM_PRIM_LINK	int 型	基本リンク権
PERM_PRIM_VERSION	int 型	基本バージョン管理権
PERM_PRIM_DELETE	int 型	基本削除権
PERM_CHANGE_PERM	int 型	アクセス制御情報変更権
PERM_CREATE	int 型	オブジェクト作成権
PERM_READ_PROPS	int 型	プロパティ参照権 次のパーミッションと等価 • PERM_PRIM_READ_PROPS
PERM_READ	int 型	参照権 次のパーミッションの論理和と等価 • PERM_PRIM_READ_PROPS • PERM_PRIM_READ_CONTENTS
PERM_WRITE_PROPS	int 型	プロパティ更新権 次のパーミッションの論理和と等価 • PERM_PRIM_READ_PROPS • PERM_PRIM_WRITE_PROPS
PERM_READ_WRITE	int 型	参照更新権 次のパーミッションの論理和と等価 • PERM_READ • PERM_PRIM_WRITE_PROPS • PERM_PRIM_WRITE_CONTENTS
PERM_DELETE	int 型	削除権 次のパーミッションの論理和と等価 • PERM_PRIM_READ_PROPS • PERM_PRIM_DELETE
PERM_LINK	int 型	リンク権 次のパーミッションの論理和と等価 • PERM_PRIM_READ_PROPS • PERM_PRIM_LINK
PERM_VERSION	int 型	バージョン権 次のパーミッションの論理和と等価 • PERM_READ_WRITE • PERM_PRIM_VERSION

定数	データ型	意味
PERM_FULL_CONTROL	int 型	フルコントロール 次のパーミッションの論理和と等価 • PERM_READ_WRITE • PERM_PRIM_LINK • PERM_PRIM_VERSION • PERM_PRIM_DELETE

9.1.12 Category：PRIV

文書管理に対するユーザの特権を表す定数です。

文書空間に対するユーザの特権を表す定数を次の表に示します。

表 9-12 文書空間に対するユーザの特権を表す定数

定数	データ型	意味
PRIV_NONE	int 型	セキュリティ管理者ではありません。
PRIV_SECURITY_ADMINISTRATOR	int 型	セキュリティ管理者です。
PRIV_LOGIN_USER	int 型	DbjSession#login 時のユーザです。

9.1.13 Category：REFERENCETYPE

リファレンス種別を表す定数です。

リファレンス種別を表す定数を次の表に示します。

表 9-13 リファレンス種別を表す定数

定数	データ型	意味
REFERENCETYPE_NONE	int 型	コンテンツはありません。
REFERENCETYPE_USER_RELATIVE_CONTENT	int 型	ユーザ管理領域に格納したコンテンツ（相対パス）です。

9.1.14 Category：RELATIONEND

文書間リンク設定情報一覧の取得条件を表す定数です。

文書間リンク設定情報一覧の取得条件を表す定数を次の表に示します。

表 9-14 文書間リンク設定情報一覧の取得条件を表す定数

定数	データ型	意味
RELATIONEND_HEAD	int 型	文書間リンクで関連付けているリンク先オブジェクトの一覧を取得します。
RELATIONEND_TAIL	int 型	文書間リンクで関連付けられているリンク元オブジェクトの一覧を取得します。
RELATIONEND_STATUS_EXIST	int 型	関連付けの対象オブジェクトが存在しているリンク設定情報の一覧を取得します。
RELATIONEND_STATUS_NOT_EXIST	int 型	すでに関連付けの対象オブジェクトが削除されていて，リンクだけが残っているリンク設定情報の一覧を取得します。 関連付けの対象オブジェクトが削除されているリンク設定情報はすべて取得されます。
RELATIONEND_STATUS_ALL	int 型	すべてのリンク設定情報の一覧を取得する場合に指定します。 関連付けの対象オブジェクトが削除されているリンク設定情報はすべて取得されます。

9.1.15 Category：RENDSTATUS

レンディションステータスを表す定数です。

レンディションステータスを表す定数を次の表に示します。

表 9-15 レンディションステータスを表す定数

定数	データ型	意味
RENDSTATUS_NO_SUBREND	int 型	指定したサブレンディションに対応するマスタレンディションのコンテンツは登録済みですが，サブレンディションのコンテンツが未登録です。
RENDSTATUS_SUBREND_EXIST	int 型	指定したサブレンディションに対応するマスタレンディションおよびサブレンディションのコンテンツが登録されています。
RENDSTATUS_MASTERREND_UPDATE	int 型	指定したサブレンディションに対応するマスタレンディションが更新されていますが，サブレンディションが更新されていないため，マスタレンディションとサブレンディションの内容が不一致です。
RENDSTATUS_CONVERT_NOTREQUIRED	int 型	レンディション変換が不要な状態です。この状態の場合，レンディション変換機能によるレンディション変換の対象になりません。
RENDSTATUS_CONVERT_REQUIRED	int 型	レンディション変換が必要な状態です。この状態の場合，レンディション変換機能によるレンディション変換の対象になります。
RENDSTATUS_CONVERT_ERROR	int 型	レンディション変換機能によるレンディション変換時にエラーが発生した状態です。この状態になると，レンディション変換機能によるレンディション変換はできなくなります。
RENDSTATUS_MASTERD	int 型	指定したレンディションがマスタレンディションの場合，設定されます。

9.1.16 Category：SUBJECTTYPE

サブジェクト種別を表す定数です。

サブジェクト種別を表す定数を次の表に示します。

表 9-16 サブジェクト種別を表す定数

定数	データ型	意味
SUBJECTTYPE_USR	int 型	ユーザサブジェクトです。サブジェクトに設定した内容はユーザ識別子です。
SUBJECTTYPE_GRP	int 型	グループサブジェクトです。サブジェクトに設定した内容はグループ識別子です。
SUBJECTTYPE_SYS	int 型	システムサブジェクトです。サブジェクトに設定した内容は，システムサブジェクトを表す定数です。

9.1.17 Category：SYSSUBJECT

システムサブジェクトを表す定数です。

システムサブジェクトを表す定数を次の表に示します。

表 9-17 システムサブジェクトを表す定数

定数	データ型	意味
SYSSUBJECT_SELF	String 型	対象オブジェクトの所有者を表すサブジェクトです。 • 対象オブジェクトがパブリック ACL の ACL の場合は，そのパブリック ACL をバインドしている文書やフォルダを表すオブジェクトの所有者を示します。 • 対象オブジェクトが文書やフォルダを表すオブジェクトのローカル ACL およびセキュリティ ACL の場合は，そのオブジェクトの所有者を示します。 • 対象オブジェクトがパブリック ACL のセキュリティ ACL の場合は，パブリック ACL の所有者を示します。

9.1.18 Category：Others

そのほかの内容を表す定数です。

そのほかの内容を表す定数を次の表に示します。

表 9-18 そのほかの内容を表す定数

定数	データ型	意味
MAX_NUM	int 型	検索結果の取得件数を最大とします。
INITIAL_KEY	int 型	キャッシュ検索を実行しません。
INDEXPATH_SAME	String 型	全文検索インデクス作成用ファイルのパスを DbjUploadInfo#setIndexPath メソッドの filePath プロパティに指定したファイルのパスとします。
CURRENT_VERSION	String 型	アクセス対象のバージョンをカレントバージョンとします。
DEFAULT_DOCSPCID	String 型	文書空間識別子のデフォルト値（673d2be0-d1fd-11d0-ab59-08002be29e1d）です。

9.2 DbjTraceDef クラス

DbjTraceDef クラスは，DocumentBroker クラスライブラリのトレースクラスで使用する定数を定義したクラスです。

以降，DbjTraceDef クラスの定数定義内容について，カテゴリごとに説明します。

なお，DbjTraceDef クラスで定義されている定数のアクセス指定子は，すべて「public」です。

9.2.1 Category：TRACELEVEL

トレース出力時のトレースレベルを表す定数です。トレース出力時のトレースレベルを表す定数を次の表に示します。

表 9-19 トレース出力時のトレースレベルを表す定数

定数	データ型	意味
ERROR	int 型	障害監視レベル
MANAGE	int 型	通常運用レベル
HINT	int 型	障害調査レベル
DEBUG	int 型	デバッグレベル

9.2.2 Category：TRACEOUTPUT

トレース出力時のトレース出力先を表す定数です。トレース出力時のトレース出力先を表す定数を次の表に示します。

表 9-20 トレース出力時のトレース出力先を表す定数

定数	データ型	意味
PROMPT	int 型	コマンドプロンプト（標準出力・標準エラー出力）に出力します。
TRACE	int 型	アプリケーショントレースファイルに出力します。
ERRORLOG	int 型	アプリケーションエラーログファイルに出力します。

10 ライブラリ情報取得クラス詳細

この章では，ライブラリ情報取得クラスの DbjLibInfo クラス，およびメソッドについて説明します。

10.1 DbjLibInfo クラス

10.1 DbjLibInfo クラス

DbjLibInfo クラスは，DocumentBroker のバージョン情報を扱うクラスです。このクラスはインスタンスを生成できません（コンストラクタは，private 属性です）。

以降，DbjLibInfo クラスのメソッドについて説明します。

10.1.1　getVersion（バージョンの取得）

（1）機能

DocumentBroker のバージョン番号およびリビジョン番号を取得します。バージョン番号およびリビジョン番号は，次に示す計算式によって値を求めた結果の整数値で取得します。なお，バージョン番号およびリビジョン番号が XX-YY の場合，XX をバージョン番号，YY をリビジョン番号とします。ただし，リビジョン番号は 2 けたまでです。

計算式

(A × 100)+B

（凡例）

A：バージョン番号

B：リビジョン番号

例えば，バージョン番号－リビジョン番号が 02-10 の場合，バージョン番号は 2，リビジョン番号は 10 になります。したがって，このメソッドの実行によって 210 が返却されます。

（2）形式

```
static int getVersion()
```

（3）引数

なし

（4）戻り値

バージョン番号およびリビジョン番号

（5）例外

なし

11 トレースクラス詳細

この章では，トレースクラスのクラスとそのメソッドについて説明します。

11.1 DbjTrace クラス

11.1 DbjTrace クラス

DbjTrace クラスは，トレース情報を扱うクラスです。以降，DbjTrace クラスのメソッドについて説明します。

11.1.1 arg（パラメタ情報の出力）

(1) 機能

メソッドの入り口での入力パラメタ，または外部 API に渡すパラメタ情報を出力します。引数で指定したパラメタ名およびパラメタに設定した値を出力します。このメソッドは，メソッドの入り口または外部 API の実行前に実行してください。

(2) 形式

int 型のパラメタ情報を出力する場合は，形式 1 ～形式 3 の形式で指定してください。String 型のパラメタ情報を出力する場合は，形式 4 ～形式 6 の形式で指定してください。

(a) 形式 1

```
void arg(
    String        name,
    int           value
    DbjTraceDef   level,
    int           output
)
```

(b) 形式 2

```
void arg(
    String        name,
    int           value
    DbjTraceDef   level
)
```

形式 2 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

(c) 形式 3

```
void arg(
    String        name,
    int           value
)
```

形式 3 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.DEBUG になります。

(d) 形式 4

```
void arg(
    String        name,
    String        value
    DbjTraceDef   level,
    int           output
)
```

(e) 形式 5

```
void arg(
    String        name,
    String        value
    DbjTraceDef   level
)
```

形式 5 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

(f) 形式 6

```
void arg(
    String        name,
    String        value
```

）

形式 6 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.DEBUG になります。

（3） 引数

name（入力）

メソッド入力のパラメタ名，または外部 API に渡すパラメタ名を指定します。

value（入力）

パラメタに設定した値を指定します。

level（入力）

トレースレベルを指定します。次に示すトレースレベルのうち，どれかを指定してください。

- DbjTraceDef.ERROR
  障害監視レベル
- DbjTraceDef.MANAGE
  通常運用レベル
- DbjTraceDef.HINT
  障害調査レベル
- DbjTraceDef.DEBUG
  デバッグレベル

output（入力）

トレース情報の出力先を指定します。次に示す出力先を指定できます。

- DbjTraceDef.PROMPT
  コマンドプロンプト（標準出力・標準エラー出力）に出力します。
- DbjTraceDef.TRACE
  アプリケーショントレースファイルに出力します。
- DbjTraceDef.ERRORLOG
  アプリケーションエラーログファイルに出力します。

前記以外の出力先を指定した場合，トレース情報は出力されません（実行したメソッドは正常終了します）。
複数の出力先を指定する場合は，OR 演算子を用いて定数を連結してください。例えば，DbjTraceDef.TRACE と DbjTraceDef.ERRRORLOG に同時出力する場合は，DbjTraceDef.TRACE | DbjTraceDef.ERRORLOG と指定してください。

（4） 戻り値

なし

（5） 例外

なし

11.1.2 call（外部 API の呼び出し情報の出力）

(1) 機能

外部 API の呼び出し時の情報を出力します。引数で指定した呼び出し元のクラス名およびメソッド名を出力します。このメソッドは，外部 API の呼び出し時に実行してください。

(2) 形式

クラス名およびメソッド名を出力する場合は，形式 1 ～形式 3 の形式で指定してください。メソッド名だけを出力する場合は，形式 4 ～形式 6 の形式で指定してください。

(a) 形式 1

```
void call(
    String        className,
    String        methodName,
    DbjTraceDef   level,
    int           output
)
```

(b) 形式 2

```
void call(
    String        className,
    String        methodName,
    DbjTraceDef   level
)
```

形式 2 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

(c) 形式 3

```
void call(
    String        className,
    String        methodName
)
```

形式 3 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.DEBUG になります。

(d) 形式 4

```
void call(
    String        methodName,
    DbjTraceDef   level,
    int           output
)
```

(e) 形式 5

```
void call(
    String        methodName,
    DbjTraceDef   level
)
```

形式 5 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

(f) 形式 6

```
void call(
    String        methodName
)
```

形式 6 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.DEBUG になります。

(3) 引数

className（入力）

呼び出しクラスの名称を 256 バイト以内で指定します。256 バイトを超えた名称を指定した場合は，256 バイトまでの文字列が設定されます。

methodName（入力）

呼び出しメソッドの名称を 256 バイト以内で指定します。256 バイトを超えた名称を指定した場合は，256 バイトまでの文字列が設定されます。

level（入力）

トレースレベルを指定します。次に示すトレースレベルのうち，どれかを指定してください。

- DbjTraceDef.ERROR
  障害監視レベル
- DbjTraceDef.MANAGE
  通常運用レベル
- DbjTraceDef.HINT
  障害調査レベル
- DbjTraceDef.DEBUG
  デバッグレベル

output（入力）

トレース情報の出力先を指定します。次に示す出力先を指定できます。

- DbjTraceDef.TRACE
  アプリケーショントレースファイルに出力します。

前記以外の出力先を指定した場合，トレース情報は出力されません（実行したメソッドは正常終了します）。

(4) 戻り値

なし

(5) 例外

なし

11.1.3 DbjTrace（コンストラクタ）

（1）機能

DbjTrace オブジェクトを作成します。

（2）形式

```
DbjTrace(
    String      upName,
    String      className,
    String      methodName
)
```

（3）引数

upName（入力）

アプリケーション名を指定します。
ここで指定したアプリケーション名を init メソッドの引数 upName に指定してください。異なるアプリケーション名を指定した場合，トレース情報は出力されません。また，アプリケーション名に null を指定した場合もトレース情報は出力されません。

className（入力）

クラス名を 256 バイト以内で指定します。256 バイトを超えた名称を指定した場合は，256 バイトまでの文字列が設定されます。
null を指定した場合，クラス名は設定されません。この場合，メソッド名だけが出力されます。

methodName（入力）

メソッド名を 256 バイト以内で指定します。256 バイトを超えた名称を指定した場合は，256 バイトまでの文字列が設定されます。
null を指定した場合，enter または exit メソッドを実行しても，トレース情報は出力されません。

（4）戻り値

なし

（5）例外

なし

11.1.4 enter（メソッドの入り口情報の出力）

（1） 機能

メソッドの入り口情報を出力します。DbjTrace メソッドで指定したクラス名およびメソッド名を出力します。このメソッドは，メソッドの入り口で実行してください。

（2） 形式

（a） 形式 1

```
void enter(
    DbjTraceDef     level,
    int             output
)
```

（b） 形式 2

```
void enter(
    DbjTraceDef     level
)
```

形式 2 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

（c） 形式 3

```
void enter()
```

形式 3 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.DEBUG になります。

（3） 引数

level（入力）

トレースレベルを指定します。次に示すトレースレベルのうち，どれかを指定してください。

- DbjTraceDef.ERROR
  障害監視レベル
- DbjTraceDef.MANAGE
  通常運用レベル
- DbjTraceDef.HINT
  障害調査レベル
- DbjTraceDef.DEBUG
  デバッグレベル

output（入力）

トレース情報の出力先を指定します。次に示す出力先を指定できます。

- DbjTraceDef.PROMPT
  コマンドプロンプト（標準出力・標準エラー出力）に出力します。
- DbjTraceDef.TRACE
  アプリケーショントレースファイルに出力します。

前記以外の出力先を指定した場合，トレース情報は出力されません（実行したメソッドは正常終了します）。
複数の出力先を指定する場合は，OR 演算子を用いて定数を連結してください。例えば，DbjTraceDef.PROMPT と DbjTraceDef.TRACE に同時出力する場合は，DbjTraceDef.PROMPT | DbjTraceDef.TRACE と指定してください。

(4) 戻り値

なし

(5) 例外

なし

11.1.5 error（エラー情報の出力）

(1) 機能

ユーザアプリケーションプログラムにエラーが発生したときのエラー情報を出力します。引数で指定したメッセージまたは例外クラスを出力します。このメソッドは，ユーザアプリケーションプログラムのエラー発生時に実行してください。

(2) 形式

エラー情報をメッセージにして出力する場合は，形式 1 ～形式 3 の形式で指定してください。通常の例外の情報を出力する場合は，形式 4 ～形式 6 の形式で指定してください。DocumentBroker クラスライブラリの例外の情報を出力する場合は，形式 7 ～形式 9 の形式で指定してください。

(a) 形式 1

```
void error(
    String        message,
    DbjTraceDef   level,
    int           output
)
```

(b) 形式 2

```
void error(
    String        message,
    DbjTraceDef   level
)
```

形式 2 の場合，トレース情報の出力先はアプリケーショントレースファイルおよびアプリケーションエラーログファイルになります。

(c) 形式 3

```
void error(
    String        message
)
```

形式 3 の場合，トレース情報の出力先はアプリケーショントレースファイルおよびアプリケーションエラーログファイルになります。また，トレースレベルは DbjTraceDef.ERROR になります。

(d) 形式 4

```
void error(
    Exception     ex,
    DbjTraceDef   level,
    int           output
)
```

(e) 形式 5

```
void error(
    Exception     ex,
    DbjTraceDef   level
)
```

形式 5 の場合，トレース情報の出力先はアプリケーショントレースファイルおよびアプリケーションエラーログファイルになります。

(f) 形式 6

```
void error(
    Exception     ex
)
```

形式 6 の場合，トレース情報の出力先はアプリケーショントレースファイルおよびアプリケーションエラーログファイルになります。また，トレースレベルは DbjTraceDef.ERROR になります。

(g) 形式 7

```
void error(
    DbjException   ex,
    DbjTraceDef    level,
    int            output
)
```

(h) 形式 8

```
void error(
    DbjException   ex,
    DbjTraceDef    level
)
```

形式 8 の場合，トレース情報の出力先はアプリケーショントレースファイルおよびアプリケーションエラーログファイルになります。

(i) 形式 9

```
void error(
    DbjException   ex
)
```

形式 9 の場合，トレース情報の出力先はアプリケーショントレースファイルおよびアプリケーションエラーログファイルになります。また，トレースレベルは DbjTraceDef.ERROR になります。

(3) 引数

message（入力）

出力するメッセージを指定します。

ex（入力）

発生した例外クラスを指定します。
Exception クラスおよびそのサブクラスを指定した場合は，ex.getMessage() および ex.printStackTrace() の内容が出力されます。

level（入力）

トレースレベルを指定します。次に示すトレースレベルのうち，どれかを指定してください。

- DbjTraceDef.ERROR
  障害監視レベル
- DbjTraceDef.MANAGE
  通常運用レベル
- DbjTraceDef.HINT
  障害調査レベル
- DbjTraceDef.DEBUG
  デバッグレベル

output（入力）

トレース情報の出力先を指定します。次に示す出力先を指定できます。

- DbjTraceDef.PROMPT
  コマンドプロンプト（標準出力・標準エラー出力）に出力します。
- DbjTraceDef.TRACE
  アプリケーショントレースファイルに出力します。
- DbjTraceDef.ERRORLOG

アプリケーションエラーログファイルに出力します。

前記以外の出力先を指定した場合，トレース情報は出力されません（実行したメソッドは正常終了します）。
複数の出力先を指定する場合は，OR 演算子を用いて定数を連結してください。例えば，DbjTraceDef.TRACE と DbjTraceDef.ERRRORLOG に同時出力する場合は，DbjTraceDef.TRACE | DbjTraceDef.ERRORLOG と指定してください。

（4） 戻り値

なし

（5） 例外

なし

11.1.6 exit（メソッドの出口情報の出力）

(1) 機能

メソッドの出口情報を出力します。DbjTrace メソッドで指定したクラス名およびメソッド名を出力します。このメソッドは，メソッドの出口で実行してください。

クラス名およびメソッド名については，DbjTrace メソッドで指定した値が設定されます。トレースレベルおよび出力先については，enter メソッドで指定した値が設定されます。したがって，enter メソッドを実行する前に exit メソッドを実行した場合，トレース情報は出力されません。

(2) 形式

```
void exit()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

なし

11.1.7 hint（下位のメソッドのエラー情報の出力）

（1） 機能

下位のメソッドでエラーが発生したときのエラー情報を出力します。引数で指定したメッセージを出力します。このメソッドは，下位のメソッドのエラー発生時に実行してください。

（2） 形式

（a） 形式 1

```
void hint(
    String        message,
    DbjTraceDef   level,
    int           output
)
```

（b） 形式 2

```
void hint(
    String        message,
    DbjTraceDef   level
)
```

形式 2 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

（c） 形式 3

```
void hint(
    String        message
)
```

形式 3 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.HINT になります。

（3） 引数

message（入力）

出力するメッセージを指定します。

level（入力）

トレースレベルを指定します。次に示すトレースレベルのうち，どれかを指定してください。

- DbjTraceDef.ERROR
  障害監視レベル
- DbjTraceDef.MANAGE
  通常運用レベル
- DbjTraceDef.HINT
  障害調査レベル
- DbjTraceDef.DEBUG
  デバッグレベル

output（入力）

トレース情報の出力先を指定します。次に示す出力先を指定できます。

- DbjTraceDef.TRACE
  アプリケーショントレースファイルに出力します。

前記以外の出力先を指定した場合，トレース情報は出力されません（実行したメソッドは正常終了します）。

(4) 戻り値

なし

(5) 例外

なし

11.1.8 init（トレースクラスの初期化）

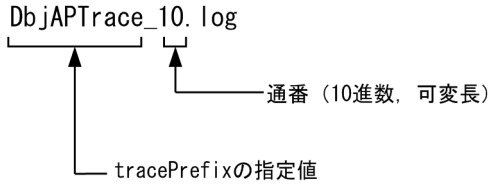
(1) 機能

トレースクラスを初期化します。

このメソッドで指定する引数 tracePrefix を基に，アプリケーショントレースファイル名およびアプリケーションエラーログファイル名が次のように決まります。

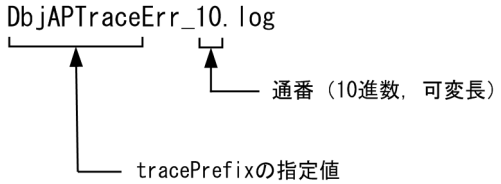
●アプリケーショントレースファイルの場合

アプリケーショントレースファイル名の例を次に示します。



●アプリケーションエラーログファイルの場合

アプリケーションエラーログファイル名の例を次に示します。



! 注意事項

- DbjTrace クラスを使用してトレース情報を出力するには，最初に init メソッドを実行してトレースクラスを初期化してください。init メソッドの実行前や，init メソッドで例外が送出された場合に，init 以外のメソッドを実行すると，そのメソッドは処理を行わないで正常終了します。
- init メソッドが正常終了したあとに，同じ upName および tracePrefix を指定した init メソッドを実行できません。同じ upName および tracePrefix を指定した init メソッドを実行した場合，処理を行わないで正常終了します。

(2) 形式

```
void init(
    String      upName
    String      tracePrefix
)
```

(3) 引数

upName（入力）

アプリケーション名を 16 バイト以内で指定します。16 バイトを超えた名称を指定した場合は，16 バイトまでの文字列が設定されます。null を指定した場合は，「DbjAP」が設定されます。

tracePrefix（入力）

アプリケーショントレースファイル名およびアプリケーションエラーログファイル名のプリフィクス（先頭部分）を 128 バイト以内で指定します。128 バイトを超えた名称を指定した場合は，128 バイト

までの文字列が設定されます。null を指定した場合は，「DbjAPTrace」が設定されます。

（4） 戻り値

なし

（5） 例外

IllegalArgumentException

指定した引数に誤りがある場合，この例外が送出されます。例えば，次に示す場合にこの例外が送出されます。

- 正常終了した init メソッドと同じ upName または tracePrefix を指定し，再度 init メソッドを実行した場合

IOException

ファイル操作に失敗した場合，この例外が送出されます。例えば，次に示す場合にこの例外が送出されます。

- アプリケーショントレースファイルまたはアプリケーションエラーログファイルの格納先ディレクトリがない場合
- アプリケーショントレースファイルまたはアプリケーションエラーログファイルを作成する権限がない場合
- アプリケーショントレースファイルまたはアプリケーションエラーログファイルのパス長（格納先ディレクトリとファイルのパス長の合計）が最大パス長を超えている場合

11.1.9 msg（メッセージの出力）

(1) 機能

メッセージを出力します。引数で指定したメッセージ ID およびメッセージを出力します。このメソッドは，メッセージの出力が必要なときに実行してください。

(2) 形式

メッセージ ID およびメッセージを出力する場合は，形式 1 ～形式 3 の形式で指定してください。メッセージだけを出力する場合は，形式 4 ～形式 6 の形式で指定してください。

(a) 形式 1

```
void msg(
    String        messageID,
    String        message,
    DbjTraceDef   level,
    int           output
)
```

(b) 形式 2

```
void msg(
    String        messageID,
    String        message,
    DbjTraceDef   level
)
```

形式 2 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

(c) 形式 3

```
void msg(
    String        messageID,
    String        message
)
```

形式 3 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.ERROR になります。

(d) 形式 4

```
void msg(
    String        message,
    DbjTraceDef   level,
    int           output
)
```

(e) 形式 5

```
void msg(
    String        message,
    DbjTraceDef   level
)
```

形式 5 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。

(f) 形式 6

```
void msg(
    String        message
)
```

形式 6 の場合，トレース情報の出力先はアプリケーショントレースファイルになります。また，トレースレベルは DbjTraceDef.ERROR になります。

(3) 引数

messageID（入力）

出力するメッセージ ID を 11 バイト以内で指定します。11 バイトを超えた場合は，11 バイトまでの文字列が設定されます。null を指定した場合，メッセージ ID は設定されません。

message（入力）

出力するメッセージを指定します。

level（入力）

トレースレベルを指定します。次に示すトレースレベルのうち，どれかを指定してください。

- DbjTraceDef.ERROR
  障害監視レベル
- DbjTraceDef.MANAGE
  通常運用レベル
- DbjTraceDef.HINT
  障害調査レベル
- DbjTraceDef.DEBUG
  デバッグレベル

output（入力）

トレース情報の出力先を指定します。次に示す出力先を指定できます。

- DbjTraceDef.PROMPT
  コマンドプロンプト（標準出力・標準エラー出力）に出力します。
- DbjTraceDef.TRACE
  アプリケーショントレースファイルに出力します。
- DbjTraceDef.ERRORLOG
  アプリケーションエラーログファイルに出力します。

前記以外の出力先を指定した場合，トレース情報は出力されません（実行したメソッドは正常終了します）。
複数の出力先を指定する場合は，OR 演算子を用いて定数を連結してください。例えば，DbjTraceDef.TRACE と DbjTraceDef.ERRRORLOG に同時出力する場合は，DbjTraceDef.TRACE | DbjTraceDef.ERRORLOG と指定してください。

(4) 戻り値

なし

(5) 例外

なし

11.1.10 returned（外部 API からのリターン情報の出力）

(1) 機能

外部 API から制御が戻ってきたときの情報を出力します。call メソッドの引数で指定したクラス名およびメソッド名を出力します。このメソッドは，外部 API から制御が戻ってきたときに実行してください。

クラス名，メソッド名，トレースレベルおよび出力先については，call メソッドで指定した値が設定されます。したがって，call メソッドを実行する前に returned メソッドを実行した場合，トレース情報は出力されません。

(2) 形式

```
void returned()
```

(3) 引数

なし

(4) 戻り値

なし

(5) 例外

なし

付録

付録 A　このマニュアルの参考情報

付録 A.1　関連マニュアル

このマニュアルの関連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。

なお，本文に記載のマニュアル名称は，「uCosminexus DocumentBroker」を「DocumentBroker」と表記しています。

DocumentBroker のマニュアル

- uCosminexus DocumentBroker Version 5　**システム導入・運用ガイド**（3021-3-401）
  DocumentBroker Version 5 を使用する環境を定義，管理および運用する場合に参照してください。
- uCosminexus DocumentBroker Version 5　**概説**（3021-3-402）
  DocumentBroker Version 5 の機能について知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 5　**メッセージ**（3021-3-405）
  DocumentBroker が出力するメッセージについて知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 5　**サンプル Web アプリケーション**（3021-3-404）
  DocumentBroker Developer が提供しているサンプル Web アプリケーションの機能と使用方法，およびサンプル Web アプリケーションを参考にした Web アプリケーションの開発方法について知りたい場合に参照してください。

関連製品のマニュアル（HiRDB）

- **ノンストップデータベース** HiRDB Version 9 SQL **リファレンス**（3020-6-457）
- **ノンストップデータベース** HiRDB Version 9 **メッセージ**（3020-6-458）
- HiRDB XML **拡張機能** HiRDB XML Extension Version 9（3020-6-480）
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 9（3020-6-481）

関連マニュアルの略称

このマニュアルで使用する関連マニュアルの略称を次に示します。

マニュアル名	略称
HiRDB Version 9 メッセージ	HiRDB　メッセージ
HiRDB Version 9 SQL リファレンス	HiRDB　SQL リファレンス
HiRDB Text Search Plug-in Version 9	HiRDB Text Search Plug-in
HiRDB XML Extension Version 9	HiRDB XML Extension

付録 A.2　このマニュアルでの表記

このマニュアルでは，製品名称を次に示す略称で表記しています。

製品名称	略称
uCosminexus Application Server Version 9	Cosminexus
uCosminexus Developer Version 9	
uCosminexus DocumentBroker Developer Version 5	DocumentBroker
uCosminexus DocumentBroker Platform Version 5	
uCosminexus DocumentBroker Runtime Library Version 5	
uCosminexus DocumentBroker Developer Version 5	DocumentBroker Developer

製品名称	略称
uCosminexus DocumentBroker Platform Version 5	DocumentBroker Platform
uCosminexus DocumentBroker Runtime Library Version 5	DocumentBroker Runtime Library
HiRDB Server Version 9	HiRDB
HiRDB Text Search Plug-in Conceptual Extension Version 7	HiRDB Text Search Plug-in Conceptual Extension
HiRDB Text Search Plug-in Version 9	HiRDB Text Search Plug-in
HiRDB XML Extension Version 9	HiRDB XML Extension
HiRDB/Run Time Version 9	HiRDB/Run Time
Java(TM) Platform Standard Edition Development Kit 6.0	JDK 6.0

このほか，このマニュアルでは，次に示す表記方法を使用しています。

- Java™ を Java と表記します。
- DocumentBroker が提供する Java 言語対応のクラスライブラリを DocumentBroker クラスライブラリと表記します。

付録 A.3　英略語

このマニュアルで使用する英略語を次に示します。

英略語	英字での表記
ACE	Access Control Element
ACFlag	Access Control Flag
ACL	Access Control List
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BLOB	Binary Large Object
BNF	Backus Normal Form
CR	Carriage Return
DB	Database
DBMS	Database Management System
DIT	Directory Information Tree
DMA	Document Management Alliance
DN	Distinguished Name
EOF	End of File
GIF	Graphics Interchange Format
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
ID	Identifier
JIS	Japanese Industrial Standards
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LF	Line Feed

英略語	英字での表記
MIME	Multipurpose Internet Mail Extensions
OIID	Object Instance Identifier
OS	Operating System
PC	Personal Computer
PDF	Portable Document Format
SGML	Standard Generalized Markup Language
URL	Uniform Resource Locator
UTF-8	8-bit UCS Transformation Format
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

付録 A.4　KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

付録 B　用語解説

DocumentBroker クラスライブラリで使用する用語について説明します。

ここで説明していない DocumentBroker クラスライブラリの用語については，マニュアル「DocumentBroker Version5　システム導入・運用ガイド」を参照してください。

（記号）

? パラメタ

edmSQL 文中のパラメタ値を，検索条件を指定するときには固定しないでおいて，検索実行時にアプリケーションから値を設定するために使用するパラメタです。edmSQL 文中の ? パラメタによって値を渡す個所には，定数を指定する代わりに「?」を指定しておきます。
DocumentBroker クラスライブラリでは，DbjQParam インターフェースのサブインターフェースによって，? パラメタの値を設定できます。

（英字）

ACE

Access Control Element の略です。アクセス制御エレメントのことです。

ACFlag

Access Control Flag の略です。アクセス制御フラグのことです。

ACL

Access Control List の略です。アクセス制御リストのことです。

AND 条件

検索条件同士の論理積を求める結合条件です。例えば，「著者が『日立太郎』で，文書のコンテンツ中に『コンピュータ』という文字列を含む文書を検索する」というような場合に使用できます。

API（Application Programming Interface）

アプリケーションプログラムとのインターフェースです。

Container オブジェクト

DocumentBroker クラスの dmaClass_Container クラス，または edmClass_ContainerVersion クラス以外の dmaClass_Container クラスのサブクラスを基に作成された DocumentBroker オブジェクトです。

dmaClass_ConfigurationHistory クラス

バージョン管理に使用する DocumentBroker クラスです。
dmaClass_ConfigurationHistory クラスまたはそのサブクラスは，バージョニングオブジェクトのトップオブジェクトクラスになります。

dmaClass_Container クラス

オブジェクトをディレクトリに格納するイメージでリンクしてまとめて管理したり，オブジェクトに分類を付けるイメージでリンクして関連付けて管理したりできるフォルダを表す DocumentBroker クラスです。
dmaClass_Container クラスまたはそのサブクラスは，構成管理できないバージョンなしフォルダのトップオブジェクトクラスになります。

dmaClass_DocVersion クラス

文書を表す DocumentBroker クラスです。
dmaClass_DocVersion クラスまたはそのサブクラスは，バージョンなし文書のトップオブジェクトクラスになります。また，構成管理できないバージョン付き文書のバージョンオブジェクトのトップオブジェクトクラスになります。

DocumentBroker オブジェクト

DocumentBroker クラスを基に作成されたオブジェクトです。

DocumentBroker クラス

DocumentBroker のオブジェクトモデルに基づいたクラス，DocumentBroker で定義しているクラスおよびそのサブクラスです。DocumentBroker の文書管理オブジェクトクラスのプロパティは，DocumentBroker クラスに定義します。

DocumentBroker クラスライブラリ

DocumentBroker にアクセスするユーザアプリケーションプログラムを Java 言語で作成するために，DocumentBroker が提供しているインタフェースです。

DocVersion オブジェクト

DocumentBroker クラスの dmaClass_DocVersion クラス，または dmaClass_DocVersion クラスのサブクラスを基に作成された DocumentBroker オブジェクトです。

edmClass_ContainerVersion クラス

オブジェクトのまとまりをバージョン管理できるフォルダを表す DocumentBroker クラスです。

edmClass_IndependentPersistence クラス

独立データを表す DocumentBroker クラスです。
edmClass_IndependentPersistence クラスまたはそのサブクラスは，独立データのトップオブジェクトクラスになります。

edmClass_PublicACL クラス

パブリック ACL を表す DocumentBroker クラスです。文書やフォルダなどのオブジェクトとは独立に作成・参照でき，複数の文書やフォルダなどのオブジェクトから共有できるアクセス制御情報を表すクラスです。
パブリック ACL のトップオブジェクトクラスになります。

edmClass_VersionTracedDocVersion クラス

文書を表す DocumentBroker クラスです。
edmClass_VersionTracedDocVersion クラスまたはそのサブクラスは，バージョン付き文書のバージョンオブジェクトのトップオブジェクトクラスになります。

edmSQL

文書管理オブジェクトを検索する場合に使用する，検索条件式を表現するための文法です。SQL の文法に基づいています。

edmSQL 検索

検索条件に，SQL の文法に基づいた文法で記述できる edmSQL 文を指定して実行する検索のことです。

getter メソッド

パラメタクラスのインターフェースで扱うオブジェクトのプロパティの値を取得するためのメソッドです。

GUID

Globally Unique Identifier の略です。DocumentBroker のクラス，プロパティなどに与えるユニークな識別子です。
GUID は，「X」を 0 ～ 9，a ～ f（小文字），および A ～ F（大文字）で表される 16 進数とした「XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX（8 けた -4 けた -4 けた -4 けた -12 けた）」の形式で表されます。

MIME 形式

MIME::text/plain，MIME::text/html など，コンテンツのレンディションタイプを表す形式です。なお，DocumentBroker では，「MIME::」は省略して指定します。

NOT 条件

指定したキーワードとの不一致を求める検索条件です。例えば，「作成者が『日立』ではない文書を検索する」というような場合に使用できます。

OIID

Object Instance Identifier の略です。データベースに格納されたオブジェクトの存在や格納位置などを明確にするために，各オブジェクトに与えるユニークな識別子です。

OR 条件

検索条件同士の論理和を求める結合条件です。例えば，「作成者が『日立太郎』である文書管理オブジェクトか，作成者の所属が『日立製作所』である文書管理オブジェクトを検索する」という場合，「文書のコンテンツ中に『コンピュータ』という文字列を含むか，『インターネット』という文字列を含む文書を検索する」という場合に使用できます。

Proxy オブジェクト

文書管理オブジェクトを操作する場合に使用する代理オブジェクトです。DocumentBroker クラスライブラリでは，文書管理オブジェクトを操作するとき，文書管理オブジェクトを直接操作するのではなく，Proxy オブジェクトを操作することで間接的に操作します。

Proxy オブジェクトはメモリ空間に存在します。不要になった場合は，Java のガベージコレクションによって削除されます。

setter メソッド

パラメタクラスのインターフェースで扱うオブジェクトのプロパティの値を設定するためのメソッドです。

VARRAY 型

文書管理オブジェクトのプロパティのデータ型の一つです。VARRAY 型のプロパティは，プロパティのデータ型に従った複数の値を可変長な一次元配列として持ちます。

VersionTracedDocVersion オブジェクト

DocumentBroker クラスの edmClass_VersionTracedDocVersion クラスまたはそのサブクラスを基に作成された DocumentBroker オブジェクトです。

W3C（World Wide Web Consortium）

HTML，WWW に関する技術の標準化を推進する非営利団体です。

（ア行）

アクセス権

文書管理オブジェクトを作成したり，すでに作成されているオブジェクトにアクセスしたりする権利です。

アクセス制御エレメント（ACE：Access Control Element）

アクセス制御リスト（ACL）の要素です。一つのサブジェクトと一つのパーミッションの組で構成され，指定されたサブジェクトに対して指定されたパーミッションの範囲のアクセス権を与えることを示す情報です。

アクセス制御機能

DocumentBroker の文書空間での文書管理オブジェクトの作成や，管理されている文書やフォルダなどの文書管理オブジェクトに対する操作を，ユーザやグループごとに許可または制限する機能です。

アクセス制御機能付き検索

アクセス制御機能を利用した文書空間で検索を実行した場合に，ユーザにアクセス権がない文書管理オブジェクトを検

索結果として取得しない検索です。

アクセス制御情報

アクセス制御されている文書空間で，ユーザがメソッドを実行する際に，アクセス権の判定に使用される情報です。

アクセス制御情報変更権

文書管理オブジェクトに設定されているアクセス制御情報（ACFlag および ACL）を変更する権利です。また，パブリック ACL をアクセス制御対象の文書管理オブジェクトに設定することを許可する権利も含みます。なお，パブリック ACL のアクセス情報変更権には，パブリック ACL のユーザ定義プロパティの値を変更する権利を含みます。

アクセス制御フラグ（ACFlag：Access Control Flag）

文書管理オブジェクトの所有者，プライマリグループおよび全ユーザという区分でパーミッションを設定できるアクセス制御情報の一つです。

アクセス制御リスト（ACL：Access Control List）

任意のユーザまたはグループにパーミッションを設定できるアクセス制御情報の一つです。アクセス制御エレメント（ACE）のリストで構成されます。

アンバインド

パブリック ACL とのバインドを解除することです。

異表記展開検索

全文検索条件として指定する検索タームまたは検索タームの異表記を含む文書を検索する全文検索のオプションです。例えば，検索タームとして「バイオリン」を指定した場合に，「ヴァイオリン」という検索タームの異表記を含む文書も検索できます。

インターフェース

DocumentBroker クラスライブラリの機能を実行するためのメソッドが定義されている，クラスの仕様です。DocumentBroker クラスライブラリでは，オブジェクトを操作するとき，まず，インターフェースを取得して，そのインターフェースで定義されているメソッドを実行するという手順で操作します。

永続オブジェクト

データベースに格納されたオブジェクトです。

永続プロパティ

データベースに存在するプロパティです。

オーナーオブジェクトプロパティ

リンク Proxy オブジェクトのプロパティです。ターゲットリンクオブジェクトによってリンクが設定されている文書管理オブジェクトのうち，リンク元である文書管理オブジェクトの Proxy オブジェクトが設定されます。

オブジェクト作成権

オブジェクト作成権限で，文書管理オブジェクトを作成する権利を与えるパーミッションです。ユーザ権限定義ファイルに指定します。

オブジェクト作成権限

文書管理オブジェクトを作成する権限で，ユーザ権限の一つです。ユーザ権限定義ファイルで定義します。オブジェクト作成権限を与えられたユーザおよびグループに属するユーザは，文書管理オブジェクトを作成するメソッドを実行できます。

オブジェクト種別

文書管理オブジェクトの種類を表す定数です。オブジェクト種別には，バージョンなし文書，バージョン付き文書，バージョンなしフォルダ，独立データおよびパブリック ACL があります。

オブジェクト種別プロパティ

Proxy オブジェクトのプロパティです。ターゲットオブジェクトのオブジェクト種別が設定されます。

オブジェクト操作権限

文書空間内のすべての文書管理オブジェクトを，与えられた権限の範囲で操作する権利です。ユーザ権限の一つです。ユーザ権限定義ファイルで定義します。例えば，オブジェクト操作権限として基本プロパティ参照権を与えられたユーザおよびグループに属するユーザは，文書空間内のすべての文書管理オブジェクトのプロパティを参照できます。

オブジェクトリファレンス

DocumentBroker オブジェクトへのリファレンスを示すプロパティの値です。例えば，dmaProp_ParentContainer プロパティの値がこれに当たります。DocumentBroker クラスライブラリのメソッドでこの値を使用することはできませんが，edmSQL 文中では使用することができます。

（カ行）

下位オブジェクト

フォルダを使用した文書管理をする場合に，フォルダからリンクを設定されている，リンク先オブジェクトです。例えば，文書をフォルダに格納する管理をしている場合は，文書が下位オブジェクトです。

概念検索

ユーザが任意に指定した文章や文字列を手がかりにして，その条件と似た概念を持つ文書を検索する方法です。全文検索の一種です。概念検索で指定する条件のことを種文章といいます。

可変長配列

VARRAY 型のプロパティの値です。DocumentBroker クラスライブラリでは，DbjVArray インターフェースで扱います。

仮のバージョン識別子

チェックアウト中の仮のバージョンを識別するための識別子です。
チェックアウト中のオブジェクトを参照・更新するときに使用します。この識別子はチェックアウト時に DocumentBroker によって設定される識別子であり，仮のバージョンに該当するオブジェクトの OIID とは異なります。

カレントバージョン

バージョン付き文書が持つ複数のバージョンの中の，最新バージョンのことです。

基本コンテンツ更新権

基本パーミッションの一つで，バージョンなし文書またはバージョン付き文書のコンテンツを更新する権利を与えるパーミッションです。基本プロパティ参照権を含みます。また，全文検索インデクスを作成，削除する権利も含みます。

基本コンテンツ参照権

基本パーミッションの一つで，バージョンなし文書またはバージョン付き文書のコンテンツを参照する権利を与えるパーミッションです。基本プロパティ参照権を含みます。

基本削除権

基本パーミッションの一つで，文書管理オブジェクトを削除する権利を与えるパーミッションです。基本プロパティ参照権を含みます。

基本バージョン管理権

基本パーミッションの一つで，バージョン管理されている文書管理オブジェクトのバージョンを追加，削除する権利を与えるパーミッションです。基本プロパティ参照権を含みます。

基本パーミッション

ユーザおよびグループが文書管理オブジェクトに対して実行できる操作の範囲を定めるパーミッションの基本単位です。

オブジェクト操作権限，アクセス制御フラグおよびアクセス制御エレメントで，ユーザおよびグループに許可する操作の範囲を定める場合に使用します。例えば，あるユーザに対して，文書の更新と削除を許可する場合は，更新と削除を許可するために，基本コンテンツ更新権と基本削除権という二つのパーミッションを設定します（一つのパーミッションで一つの権利を与えます）。基本パーミッションには，基本プロパティ参照権，基本プロパティ更新権，基本コンテンツ参照権，基本コンテンツ更新権，基本リンク権，基本バージョン管理権および基本削除権があります。

基本プロパティ更新権

基本パーミッションの一つで，文書管理オブジェクトのプロパティを更新する権利を与えるパーミッションです。基本プロパティ参照権を含みます。

基本プロパティ参照権

基本パーミッションの一つで，文書管理オブジェクトのプロパティを参照する権利を与えるパーミッションです。そのほかのすべての基本パーミッションに含まれます。フォルダに対して設定すると，フォルダのリンクをたどることも許可します。

基本リンク権

基本パーミッションの一つで，リンクを設定・解除する権利を与えるパーミッションです。基本プロパティ参照権を含みます。

キャッシュキー

キャッシュ付き検索を実行する場合に，2 回目以降の検索条件が前回の検索条件と同じかどうかを判断するためのキーです。検索条件が同じ場合，検索結果キャッシュに対して検索が実行されます。検索条件が異なる場合，検索結果キャッシュは破棄されて，文書空間に対して検索が再度実行されます。

キャッシュ検索

検索結果キャッシュを使用する検索です。文書空間に対して検索を実行したときに，取得した検索結果をキャッシュに保存しておき，同じ検索条件の 2 回目以降の検索では検索結果キャッシュから検索結果を取得します。同じ検索条件で複数回検索を実行する場合に，検索結果取得処理が高速になります。例えば，10,000 件検索結果がある場合に，これを検索結果キャッシュとして保存しておき，ユーザアプリケーションプログラムではこれを 100 件ずつ取得する，という検索ができます。

キャッシュ名

検索結果キャッシュの名称です。検索結果キャッシュは，キャッシュ名によって識別されます。

近傍条件検索

全文検索条件に複数の検索タームを指定した場合に，検索ターム間の距離を条件にして検索する方法です。例えば，「『文書管理』という検索タームと『ドキュメント』という検索タームを含み，これらの検索タームがこのとおりの順序で出現し，かつ検索ターム間に入る文字が 5 文字以内である文書を検索する」というような場合に使用できます。

組み合わせパーミッション

基本パーミッションを複数組み合わせた権利を与えるパーミッションの単位です。アクセス制御フラグおよびアクセス制御エレメントでユーザおよびグループに許可する操作の範囲を定める場合に使用します。組み合わせパーミッションには，プロパティ参照権，参照権，プロパティ更新権，参照更新権，削除権，リンク権，バージョン管理権およびフルコントロールがあります。例えば，あるユーザに対して，ある文書の参照更新権という組み合わせパーミッションを設定すると，そのユーザは基本プロパティ参照権，基本プロパティ更新権，基本コンテンツ参照権および基本コンテンツ更新権が設定されたのと同じ範囲の操作を，その文書に対して実行できます。

クラスディスクリプション

DocumentBroker クラスに関するメタ情報です。DocumentBroker クラスごとに定義されています。

継承

既存のクラスを利用して新しいクラスを定義するオブジェクト指向の技術です。

検索結果キャッシュ

キャッシュ付き検索実行時に作成される検索結果のキャッシュです。キャッシュ付き検索では，例えば，10,000 件の検索結果がある場合に，これを検索結果キャッシュとして保存しておき，ユーザアプリケーションプログラムではこれを

100 件ずつ取得する，という検索ができます。

検索結果集合
検索実行時に作成される，検索条件に一致する文書管理オブジェクトの集合です。要素がプロパティ値である，行と列の二次元データに，列名や列のデータ型などのメタデータを付けたものとして表されます。検索結果集合は，DbjResultSet インターフェースで扱います。

検索結果取得情報
キャッシュ付き検索実行時に，検索結果を何件検索結果キャッシュとして保存するか，また，検索結果キャッシュから何件ユーザアプリケーションプログラムに返却するかを指定する情報です。DbjFetchInfo インターフェースで扱います。

更新系メソッド
操作対象になるオブジェクトの状態を変化させるメソッドです。文書のコンテンツの更新や，文書やフォルダのプロパティの更新を実行するメソッドなどが該当します。

コレクション
java.util.Collection インターフェースの仕様に従うオブジェクトです。要素オブジェクトの集合を表します。

コンテンツ
文書のデータ部分を指します。DocumentBroker で規定しているコンテンツ管理モデルに従ってアクセスされるオブジェクトの実体（例えば，report.doc，document.htm など）です。

コンテンツ格納先パス
リファレンスファイル管理機能を使用する場合に，コンテンツ格納先ベースパスを基点とする相対パスのことです。

コンテンツ格納先ベースパス
リファレンスファイル管理機能を使用する場合に，コンテンツ格納先の基点となるディレクトリパスのことです。

コンテンツ種別
レンディションのコンテンツが，シングルファイル文書のコンテンツか，またはリファレンスファイル文書のコンテンツかを示します。

コンテンツ情報
文書のコンテンツ管理モデルで管理されているコンテンツに関する情報です。ファイル名およびレンディションタイプが管理されています。DbjContentInfo インターフェースで扱います。

コンテンツロケーション
リファレンスファイル管理機能を使用する場合に，コンテンツの格納先を示す情報のことです。

（サ行）

削除権
組み合わせパーミッションの一つです。基本削除権と同じ操作を許可するパーミッションです。

サブインターフェース
あるインターフェースから派生するインターフェースのことです。上位インターフェースで定義されているメソッドを継承します。

サブクラス
あるクラスから派生するクラスのことです。または，それ自身がサブクラスとして参照されているクラスのことです。

サブジェクト
アクセス権を与えるユーザまたはグループです。

サブジェクト種別
アクセス権を与えるサブジェクトが，ユーザなのか，グループなのかまたはシステムなのかを識別するための情報です。

サブレンディション
マルチレンディション文書のマスタレンディション以外のレンディションです。なお，サブレンディションは，登録後にマスタレンディションに変更できます。

参照型リンク
一つのオブジェクトから複数のバージョンなしフォルダを親として関連付けるリンクの種別です。一つの文書に対して複数の分類を付けるイメージでの管理を実現します。

参照系メソッド
操作対象になるオブジェクトの状態を変化させないメソッドです。文書のコンテンツの参照や，文書やフォルダのプロパティの参照を実行するメソッドなどが該当します。

参照権
組み合わせパーミッションの一つです。基本コンテンツ参照権と同じ操作を許可するパーミッションです。

参照更新権
組み合わせパーミッションの一つです。基本プロパティ参照権，基本プロパティ更新権，基本コンテンツ参照権および基本コンテンツ更新権を組み合わせたパーミッションです。すなわち，参照更新権を設定することで，プロパティを参照，更新する権利とコンテンツを参照，更新する権利を設定できます。

サンプル Web アプリケーション
DocumentBroker がサンプルとして提供しているアプリケーションプログラムです。DocumentBroker を使用して開発されたコンポーネントで構成されています。
ユーザアプリケーションプログラムを開発する際に，アプリケーションプログラムのアーキテクチャを参考にできます。また，DocumentBroker のクラス，インターフェース，メソッドなどの具体的な使用方法について参考にできます。

システム管理者
DocumentBroker の運用，管理および保守をするユーザです。DocumentBroker の実行環境を設定することができます。

上位オブジェクト
フォルダを使用した文書管理をする場合のリンク元オブジェクトです。つまり，フォルダのことです。

状態フラグ
マルチレンディション文書で，マスタレンディションに対するサブレンディションのコンテンツの状態を表すフラグです。マスタレンディションとサブレンディションのコンテンツの状態が一致している，マスタレンディションのコンテンツが更新されたのに対してサブレンディションのコンテンツが更新されていない，またはサブレンディションのコンテンツが存在しない，という 3 種類の状態が表されます。dbrProp_RenditionStatus プロパティの下位 2 バイトに設定されます。

所有者
文書管理オブジェクトの所有者として設定されているユーザです。アクセス制御フラグでアクセス権を与えられます。所有者に設定されているユーザは，その文書管理オブジェクトのアクセス制御フラグで所有者に与えられたパーミッションの範囲の操作を，その文書管理オブジェクトに対して実行できます。また，その文書管理オブジェクトの所有者およびセキュリティ ACL の値を変更できます。

スーパーインターフェース
あるインターフェースのインターフェース定義に使用されたインターフェースです。派生したインターフェースでは，スーパーインターフェースで定義されているメソッドを継承します。

スーパークラス
あるクラスのクラス定義に使用されたクラスを，派生したクラスのスーパークラスといいます。

セキュリティ ACL

文書管理オブジェクトに設定されたアクセス制御情報へのアクセスを制御するためのアクセス制御リストです。任意のユーザまたはグループにアクセス制御情報変更権を設定できます。

セキュリティ管理者

アクセス制御機能を利用した文書空間で，アクセス権判定を受けることなく，すべての文書管理オブジェクトに自由にアクセスする特権を持ち，文書空間のすべての文書管理オブジェクトを保守するユーザです。セキュリティ定義ファイルに定義します。

セッション

文書空間に接続している間のことです。文書空間に接続することを，セッションの確立といいます。文書空間との接続を解除することを，セッションの切断といいます。

セッションオブジェクト

セッションを確立する機能とセッション内のトランザクションを制御する機能を持つ，DocumentBroker クラスライブラリのオブジェクトです。DbjSession インターフェースの機能を実行できます。

セット

java.util.Set インターフェースの仕様に従うオブジェクトです。重複のない，要素オブジェクトの集合を表します。

全文検索

文書中に含まれるキーワードを条件（全文検索条件）として，キーワードを含む文書を検索する方法です。

全文検索インデクス

文書を全文検索するために，データベースに登録するインデクスです。全文検索の対象になるテキストデータに対応する edmProp_TextIndex プロパティ，edmProp_ConceptTextIndex プロパティに相当します。

全文検索機能付き文書クラス

dmaClass_DocVersion クラスのサブクラスに全文検索に必要なプロパティを追加したサブクラスです。プロパティの追加は，ユーザが行います。
バージョンなし文書作成時のトップオブジェクトクラスの DocumentBroker クラス名として，またはバージョン付き文書作成時のバージョンオブジェクトのトップオブジェクトクラスの DocumentBroker クラス名として指定します。
edmSQL で全文検索を実行するときには，この DocumentBroker クラスを検索対象として FROM 句に指定します。

属性検索

文書管理オブジェクトのプロパティを対象にした検索です。例えば，「『文書名』が『X』で『作成者』が『A』の文書」のような条件を設定して検索する方法です。

（タ行）

ターゲット OIID プロパティ

Proxy オブジェクトのプロパティです。ターゲットオブジェクトの OIID が設定されます。

ターゲットオブジェクト

Proxy オブジェクトが対象にする文書管理オブジェクトのことです。

ターゲットオブジェクトプロパティ

リンク Proxy オブジェクトのプロパティです。ターゲットリンクオブジェクトが関連付けている二つの文書管理オブジェクトのうち，リンク先オブジェクトの Proxy オブジェクトが設定されます。
例えば，フォルダと文書間の直接型リンクをターゲットリンクオブジェクトとするリンク Proxy オブジェクトの場合，ターゲットオブジェクトプロパティには文書の Proxy オブジェクトが設定されます。

ターゲットバージョン
バージョン付きオブジェクトの複数のバージョンオブジェクトのうち，操作の対象になるバージョンです。

ターゲットバージョン識別子プロパティ
Proxy オブジェクトのプロパティです。ターゲットバージョンであるバージョンオブジェクトのバージョン識別子が設定されます。

ターゲットプロパティ値集合プロパティ
Proxy オブジェクトのプロパティです。文書管理オブジェクトからロードしたプロパティ値集合が設定されます。

ターゲットリンクオブジェクト
リンク Proxy オブジェクトが対象にするリンクオブジェクトのことです。

ターゲットリンク識別子プロパティ
リンク Proxy オブジェクトのプロパティです。ターゲットリンクオブジェクトのリンク識別子が設定されます。

種文章
概念検索の検索条件に指定する文章です。概念検索では，種文章を特徴付ける単語が種文章から抽出され，さらに抽出された特徴タームの中から種文章の概念を表す（実際の検索に使用する）タームが選出されます。このタームが，検索タームとして使用されます。

チェックアウト
バージョン付き文書にバージョンを追加するために，仮のバージョンを追加することです。

チェックアウト情報
バージョン付きオブジェクトのチェックアウトに関する情報です。チェックアウトしているか，チェックアウトしている場合はだれがチェックアウトしているか，また，チェックアウトしたときに設定された仮のバージョン識別子は何かについての情報です。DbjCheckOutInfo インターフェースで扱います。

チェックイン
バージョン付き文書をチェックアウトして追加した仮のバージョンを最新バージョンとして確定することです。

直接型リンク
一つのフォルダを親として複数のオブジェクトを関連付けるリンクの種別です。ディレクトリにファイルを格納するイメージでの管理を実現します。

定数定義クラス
DocumentBroker クラスライブラリのメソッドで指定する定数が定義されている DbjDef クラスのことです。

ディレクトリサービス
ネットワーク上にあるユーザや組織の情報などの資源とその属性を記憶し，検索できるようにしたシステムです。DocumentBroker では，Active Directory や Oracle Directory Server Enterprise Edition などの製品を使用した LDAP 対応のディレクトリサービスと連携できます。

同義語展開検索
全文検索条件として指定する検索タームまたは検索タームの同義語を含む文書を検索する方法です。例えば，検索タームとして「パソコン」を指定した場合に，「電子計算機」，「パーソナルコンピュータ」，「PC」など，検索タームと同じ意味を持つ単語を含む文書も検索できます。

独立データ
独立したデータを表すオブジェクトです。プロパティだけを持つことができる文書管理オブジェクトです。edmClass_IndependentPersistence クラスまたはそのサブクラスを基に作成した DocumentBroker オブジェクトをトップオブジェクトとする文書管理オブジェクトです。

特権
アクセス制御機能を利用した文書空間で，アクセス権判定を受けることなく，すべてのオブジェクトに自由にアクセス

する権利です。セキュリティ定義ファイルにセキュリティ管理者として定義されたユーザに与えられます。特権の有無は，ログイン時にセキュリティ定義ファイルが参照され，ログインユーザごとに作成されるユーザ情報に保持されます。

トップオブジェクト

文書管理オブジェクトを構成する DocumentBroker オブジェクトのうち，代表的な DocumentBroker オブジェクトです。例えば，バージョンなし文書の場合は，DocVersion オブジェクトです。
また，バージョン付きオブジェクトの場合，バージョニングオブジェクトのトップオブジェクトとバージョンオブジェクトのトップオブジェクトがあります。例えば，バージョン付き文書の場合は，ConfigurationHistory オブジェクトと VersionTracedDocVersion オブジェクト（または DocVersion オブジェクト）がトップオブジェクトです。

トップオブジェクトクラス

トップオブジェクトの基になる DocumentBroker クラスです。文書管理オブジェクトのプロパティの定義元になります。

トランザクション

文書管理オブジェクトの処理単位です。トランザクション単位で，文書管理オブジェクトへの操作を確定または取り消すことができます。
トランザクションの単位は，ユーザが明示的に指定できます。指定しなかった場合は，メソッド単位でトランザクションが分割されます。

トレースクラス

DocumentBroker クラスライブラリを使用してユーザアプリケーションプログラムを開発・運用する上で必要なトレース情報を出力するメソッドを提供しています。

トレース情報

DocumentBroker クラスライブラリで発生した障害の原因を追求するための情報です。

（ナ行）

名前付き検索結果

列名が付いている検索結果集合です。列名には，edmSQL 文で SELECT 句に指定した項目の名前（プロパティ名など）が設定されます。

名前なし検索結果

列名が付いていない検索結果集合です。名前なし検索結果は，あとから列名を追加して，名前付き検索結果集合にすることができます。

（ハ行）

バージョニングオブジェクト

バージョン付きオブジェクトの一連のバージョンを統括する文書管理オブジェクトです。DocumentBroker オブジェクトの ConfigurationHistory オブジェクトをトップオブジェクトとする文書管理オブジェクトです。

バージョンオブジェクト

バージョン付きオブジェクトの個々のバージョンを表す文書管理オブジェクトです。
バージョン付き文書の場合は，DocumentBroker オブジェクトの VersionTracedDocVersion オブジェクト（または DocVersion オブジェクト）をトップオブジェクトとする文書管理オブジェクトです。

バージョン管理

文書管理オブジェクトを更新する場合に，古いオブジェクトを残して新しい状態のオブジェクトを追加して，一連の履歴を管理することです。

バージョン管理権
組み合わせパーミッションの一つです。基本プロパティ参照権，基本プロパティ更新権，基本コンテンツ参照権，基本コンテンツ更新権および基本バージョン管理権を組み合わせたパーミッションです。

バージョン識別子
複数のバージョンを持つ文書管理オブジェクトの，バージョンを識別するための識別子です。特定のバージョンを指定して操作するときに使用します。この識別子は DocumentBroker によってバージョンごとに設定される識別子であり，バージョンオブジェクトの OIID とは異なります。

バージョン付き文書
複数のバージョンを保持できる文書を表す文書管理オブジェクトです。

バージョンなしフォルダ
バージョン管理しないフォルダを表す文書管理オブジェクトです。

バージョンなし文書
バージョン管理しない文書を表す文書管理オブジェクトです。

パーミッション
文書管理オブジェクトの作成，プロパティ参照，コンテンツ更新などの実行できる操作の範囲を表す値です。オブジェクト作成権限を与えるパーミッション，オブジェクトの操作の範囲を定めるパーミッションがあります。

バインド
文書やフォルダからパブリック ACL を参照することです。

パブリック ACL
文書管理オブジェクトとして存在するアクセス制御情報です。複数の文書管理オブジェクトで共有できます。

パラメタクラス
DocumentBroker クラスライブラリ固有のデータをメソッドで受け渡す場合に使用するインターフェースの総称です。

ファクトリオブジェクト
DbjFactory インターフェースの機能を実行するためのオブジェクトです。パラメタクラスのオブジェクトおよびセッションオブジェクトが作成できます。

ファクトリクラス
パラメタクラスのオブジェクトの生成，セッションオブジェクトの生成，および文書空間メタ情報アクセスインターフェースの取得を実行する，クラスならびにインターフェースです。

プライマリグループ
アクセス制御フラグ（ACFlag）でパーミッションを与えるグループです。

フラッシュ
Proxy オブジェクトのターゲットプロパティ値集合プロパティに設定した値を，文書管理オブジェクトのプロパティに反映することです。

フルコントロール
組み合わせパーミッションの一つです。すべての基本パーミッションを組み合わせたパーミッションです。文書管理オブジェクトに対するすべての操作を許可します。

プロパティ
オブジェクトに関する付加情報です。プロパティに設定される値をプロパティ値といいます。

プロパティ更新権
組み合わせパーミッションの一つです。基本プロパティ更新権と同じ操作を許可するパーミッションです。

プロパティ参照権
組み合わせパーミッションの一つです。基本プロパティ参照権と同じ操作を許可するパーミッションです。

プロパティ値集合
プロパティ名とプロパティ値の対応を要素として持つ集合です。DbjPropSet インターフェースで扱います。
DbjPropSet インターフェースは，java.util.Map インターフェースを継承しています。キーをプロパティ名，値をプロパティ値とするマップとして表されます。
Proxy オブジェクトにロードしたプロパティの値を操作したり，文書管理オブジェクトに設定するためのプロパティの値を設定したりするときに使用します。

プロパティ定義元の DocumentBroker クラス
DocumentBroker クラスライブラリで扱う文書管理オブジェクトのプロパティが実際に定義されている DocumentBroker のクラスです。

プロパティディスクリプション
文書管理オブジェクトのプロパティについてのメタ情報です。プロパティごとに定義されています。

文書
dmaClass_DocVersion クラスまたはそのサブクラスをトップオブジェクトクラスとする文書管理オブジェクトです。コンテンツを保持できます。

文書管理オブジェクト
DocumentBroker の文書管理で使用する，文書空間に存在するオブジェクトの総称です。バージョンなし文書，バージョン付き文書，バージョンなしフォルダ，独立データおよびパブリック ACL があります。

文書管理クラス
DocumentBroker の文書管理モデルに基づいた管理を実現するための機能を提供するインターフェースの総称です。

文書間リンク
文書と文書を関連付けるリンクの種別です。

文書空間
DocumentBroker オブジェクトモデルを実装するリポジトリです。

文書空間アクセスオブジェクト
セッションを確立した文書空間を，概念的なオブジェクトとして扱うためのオブジェクトです。DbjDocSpace インターフェースの機能を実行できます。

文書空間構成定義ファイル
アクセス制御の運用情報を定義するファイルです。セキュリティ管理者，ユーザ権限定義ファイル名，および文書管理オブジェクト作成時にアクセス制御フラグにデフォルトで設定されるパーミッションを定義します。

文書空間識別子
セッションを確立する文書空間を特定するための識別子です。GUID 文字列で表されます。

文書のアップロード情報
文書のコンテンツを更新するときに，コンテンツとして登録するファイルのパス名やファイル名，レンディションタイプなどを設定します。DbjUploadInfo インターフェースで扱います。

変換フラグ
マルチレンディション文書のサブレンディションのコンテンツを，レンディション変換の対象にするかどうかを表すフラグです。レンディション変換機能を使用してレンディション変換を実行する場合に使用します。また，レンディション変換機能によるレンディション変換でエラーが発生した場合には，エラーを示すフラグとしても使われます。
dbrProp_RenditionStatus プロパティの上位 2 バイトに設定されます。

（マ行）

マスタレンディション
マルチレンディション文書の主要なレンディションです。マルチレンディション文書の参照・更新時には，レンディション形式を指定しますが，レンディション形式を指定しない場合は，マスタレンディションが対象になります。なお，マスタレンディションとして扱うレンディションは，登録後に変更できます。

マップ
java.util.Map インターフェースの仕様に従うオブジェクトです。キーと値が対応付けられた要素オブジェクトの集合を表します。

マルチレンディション機能
一つの文書に，同一内容の複数の異なる形式のコンテンツを登録する機能です。

マルチレンディション文書
複数のレンディションを登録している文書のことです。一つの同じ内容を表す複数の形式のコンテンツを保持する文書です。バージョンなし文書クラスまたはバージョン付き文書クラスを使用して操作します。

メソッド
データを操作するために定義されている方法です。DocumentBroker クラスライブラリでは，インターフェースおよびクラスごとにメソッドが定義されています。

メタクラス
文書空間のメタ情報を扱うためのインターフェースの総称です。

メッセージ情報
DocumentBroker クラスライブラリのメソッドを実行してエラーが発生したときに取得できるエラーメッセージです。

（ヤ行）

ユーザ管理システム
DocumentBroker にログインするユーザのユーザ名や所属グループなどの情報を管理しているシステムです。LDAP 対応のディレクトリサービスなどが使用できます。
なお，ログイン時の認証に必要な情報，およびアクセス制御機能に必要な情報を取得する場合は，DocumentBroker を介してユーザ管理システムにアクセスします。この場合にどのユーザ管理システムと連携するかについては，DocumentBroker で定義されています。

ユーザ権限
文書空間にオブジェクトを作成する権利（オブジェクト作成権限）と，文書空間内のすべてのオブジェクトに対する操作の範囲（オブジェクト操作権限）をユーザまたはグループ単位で定めるアクセス制御情報の一つです。ユーザ権限定義ファイルに定義します。ユーザ権限の内容は，ログイン時にユーザ権限定義ファイルが参照され，ログインユーザごとに作成されるユーザ情報に保持されます。

ユーザ権限定義ファイル
ユーザ権限（オブジェクト作成権限およびオブジェクト操作権限）を定義するためのファイルです。

ユーザ情報
ログインユーザのユーザ識別子，所属グループ，特権およびユーザ権限を表す情報です。ログイン時にユーザごとに生成され，アクセス権判定に使用されます。

ユーザ定義プロパティ
ユーザが業務に応じて追加するプロパティです。

（ラ行）

ライブラリ情報取得クラス
DocumentBroker のバージョン情報を返却するメソッドを提供するクラスです。

リスト
java.util.List インターフェースの仕様に従うオブジェクトです。順序付けのある要素オブジェクトの集合を表します。

リファレンスファイル管理機能
DocumentBroker が存在するマシンから接続可能なファイルシステムの任意のディレクトリで文書のコンテンツを管理し，文書のプロパティおよびコンテンツの格納先の情報をデータベースで管理する機能です。

リファレンスファイル文書
DocumentBroker が存在するマシンから接続可能なファイルシステムの任意のディレクトリに格納されているファイルをコンテンツとして持つ文書のことです。データベースでは，文書のプロパティおよびコンテンツの格納先の情報を管理しています。

リレーション種別
文書間リンクで関連付けられた文書をたどる場合に，リンク先の文書を取得するか，リンク元の文書を取得するかを指定するための種別です。

リンク Proxy オブジェクト
リンクオブジェクトを操作する場合に使用する代理オブジェクトです。
リンク Proxy オブジェクトは，メモリ空間に存在します。不要になった場合は，Java のガベージコレクションによって削除されます。

リンクオブジェクト
文書管理オブジェクトの関連付けに使用するオブジェクトです。文書空間およびデータベースに存在する永続オブジェクトです。

リンク権
組み合わせパーミッションの一つです。基本リンク権と同じ操作を許可するパーミッションです。

リンク先オブジェクト
リンクを設定する先になる文書管理オブジェクトです。

リンク識別子
文書管理オブジェクト間（フォルダとフォルダ間）のリンクを識別するための識別子です。リンクを解除したり，リンクのプロパティの参照または更新したりする場合に使用します。
この識別子は関連付けをしたときに DocumentBroker によって設定される識別子です。同じフォルダに，同じオブジェクトを 2 度リンク付けした場合は，それぞれ異なるリンク識別子が設定されます。

リンク種別
リンクの種類です。直接型リンク，参照型リンク，文書間リンクがあります。

リンク設定情報
リンクの設定に関する情報です。リンク種別，リンクオブジェクトのプロパティ，リンク先になるオブジェクトに関する情報などがあります。DbjSetLinkInfo インターフェースのサブインターフェースで表されます。なお，DbjSetLinkInfo インターフェースのサブインターフェースは，リンク種別ごとに存在します。

リンクプロパティ値集合プロパティ
リンク Proxy オブジェクトのプロパティです。リンクオブジェクトからロードしたプロパティ値集合が設定されます。

リンク元オブジェクト
リンクを設定する元になる文書管理オブジェクトです。

例外クラス
DocumentBroker クラスライブラリで発生する例外のうち，DocumentBroker クラスライブラリ固有の例外を扱うクラスの総称です。java.lang.Exception クラスまたは java.lang.Error クラスを継承しています。

列名
検索結果集合の列に付ける名前です。edmSQL 文の SELECT 句に指定したプロパティ名が設定されます。

レンディション
文書のコンテンツの形式およびそのコンテンツを併せた概念です。

レンディション情報
レンディションの情報です。レンディションタイプおよびレンディションのプロパティです。DbjRenditionInfo インターフェースで扱います。

レンディション情報リスト
レンディション情報を要素としたリストです。DbjRenditionList インターフェースで扱います。

レンディションタイプ
Word などのアプリケーションで編集したファイル，GIF などの画像データのファイルのように，登録した文書のコンテンツのファイルの形式を表す文字列です。レンディションごとに設定します。DocumentBroker では，レンディションタイプとして，MIME 形式を指定することを推奨しています。

レンディション定義ファイル
ファイルの拡張子とレンディションタイプ（MIME 形式）の対応を定義するファイルです。DocumentBroker クラスライブラリがコンテンツとして登録するファイルの拡張子によってレンディションタイプ（MIME 形式）を自動的に判別して設定する場合に参照されます。

レンディション変換
マルチレンディション文書のマスタレンディションのコンテンツを変換して，サブレンディションのコンテンツを作成，登録することです。

ローカル ACL
文書管理オブジェクトごとに設定できるアクセス制御リスト（ACL）です。VARRAY 型のプロパティとして設定されます。

ロード
文書管理オブジェクトのプロパティを，Proxy オブジェクトのターゲットプロパティ値集合プロパティに読み込むことです。

ログアウト
文書空間とのセッションを切断することです。

ログイン
文書空間とのセッションを確立することです。ログインするときには，ユーザ識別子とパスワードによる認証処理も実行されます。

ロック指定検索
検索結果集合として取得した文書管理オブジェクトに，指定した種別のロックを設定する検索です。

ロック種別
排他制御を実行するために設定する，ロックの種類です。read ロックと write ロックがあります。

ロック種別プロパティ
Proxy オブジェクトのプロパティです。その Proxy オブジェクトのインターフェースのメソッドで設定するロック種別が設定されます。

（ワ行）

ワイルドカード

文字の代わりに指定する記号です。検索する単語の一部しかわからない場合，わかっている部分にワイルドカードを付けて検索条件のキーワードとして指定します。

索引

記号

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

な

に

は

ひ

ふ

へ

ま

め

も

ゆ

よ

ら

り

る

れ

ろ

わ