HITACHI
Inspire the Next

uCosminexus DocumentBroker Version 3
クラスライブラリ　Java　解説

解説書

3000-3-F16-20

■対象製品

R-1M95D-43※ uCosminexus DocumentBroker Development Kit Version 3 03-60（適用 OS：AIX 5L V5.1，AIX 5L V5.2，AIX 5L V5.3）
R-1595D-43 uCosminexus DocumentBroker Development Kit Version 3 03-70（適用 OS：Windows Server 2003，Windows Server 2003 R2，Windows Server 2003 R2 x64 Edition，Windows Server 2008 x86，Windows Server 2008 R2，Windows Server 2012，Windows XP，Windows Vista，Windows 7，Windows 8）
R-1M95D-53※ uCosminexus DocumentBroker Runtime Version 3 03-60（適用 OS：AIX 5L V5.1，AIX 5L V5.2，AIX 5L V5.3）
R-1595D-53 uCosminexus DocumentBroker Runtime Version 3 03-70（適用 OS：Windows Server 2003，Windows Server 2003 R2，Windows Server 2003 R2 x64 Edition，Windows Server 2008 x86，Windows Server 2008 R2，Windows Server 2012，Windows XP，Windows Vista，Windows 7，Windows 8）
これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。
※印の製品については，サポート時期をご確認ください。

■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

■発行

2013 年 9 月 3000-3-F16-20

■著作権

変更内容

変更内容（3000-3-F16-20）DocumentBroker Development Kit Version 3　03-70（**適用** OS：Windows Server 2003，Windows Server 2003 R2，Windows Server 2003 R2 x64 Edition，Windows Server 2008 x86，Windows Server 2008 R2，Windows Server 2012，Windows XP，Windows Vista，Windows 7，Windows 8），DocumentBroker Runtime Version 3　03-70（**適用** OS：Windows Server 2003，Windows Server 2003 R2，Windows Server 2003 R2 x64 Edition，Windows Server 2008 x86，Windows Server 2008 x64，Windows Server 2008 R2，Windows Server 2012，Windows XP，Windows Vista，Windows 7，Windows 8）

追加・変更内容	変更個所
次の前提 OS をサポートしました。 • Windows Server 2012 • Windows 8	1.2，1.3.1，1.3.2，1.3.3，

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 Java クラスライブラリの機能および環境設定の方法について説明したものです。

- R-1M95D-43　uCosminexus DocumentBroker Development Kit Version 3
- R-1595D-43　uCosminexus DocumentBroker Development Kit Version 3
- R-1M95D-53　uCosminexus DocumentBroker Runtime Version 3
- R-1595D-53　uCosminexus DocumentBroker Runtime Version 3

■対象読者

このマニュアルは，uCosminexus DocumentBroker Development Kit および uCosminexus DocumentBroker Runtime が提供する Java 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。

- UNIX または Windows に関する知識
- Java^{TM} 言語に関する知識
- XML 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　Java クラスライブラリの概要

Java 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　文書管理で使用する概念

文書管理で使用するオブジェクト，クラス，インターフェースなどの概念について説明しています。

第 3 章　文書管理モデル

Java クラスライブラリで実現できる文書管理モデルについて説明しています。

第 4 章　検索機能

Java クラスライブラリが提供する機能によってオブジェクトを検索する方法について説明しています。

第 5 章　edmSQL の文法

edmSQL の文法について説明しています。

第 6 章　Java クラスライブラリの機能

Java クラスライブラリの機能について説明しています。

第 7 章　環境の設定

Java クラスライブラリを使用する際の環境設定，注意事項などについて説明しています。

第 8 章　障害対策

障害が発生したときに原因を追求して対処するために使用するトレース情報とメッセージ情報について説明しています。

付録 A　文書空間オブジェクトのプロパティ一覧

Java クラスライブラリで扱うことができるプロパティの種類について説明しています。

付録 B　用語解説

Java 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。

■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。

uCosminexus DocumentBroker **のマニュアル**

uCosminexus DocumentBroker Version 3　**クラスライブラリ**　Java　**リファレンス**（3000-3-F17）

Java 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知りたい場合に参照してください。

uCosminexus DocumentBroker Version 3　**クラスライブラリ**　Java　**サンプル** Web **アプリケーション**（3000-3-F18）

uCosminexus DocumentBroker Development Kit および uCosminexus DocumentBroker Runtime が提供している Java クラスライブラリを使用したサンプル 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。

DocumentBroker Version 3　**システム導入・運用ガイド**（3000-3-D01）

uCosminexus DocumentBroker Version 3　**システム導入・運用ガイド**（3020-3-U71）

uCosminexus DocumentBroker 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。
UNIX の場合は，資料番号が 3000-3-D01 のマニュアルを参照してください。
Windows の場合は，資料番号が 3020-3-U71 のマニュアルを参照してください。

uCosminexus DocumentBroker Version 3　**メッセージ**（3000-3-F12）

uCosminexus 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 Text Search Index Loader Version 3（3020-3-U72）

uCosminexus DocumentBroker に格納されている文書からテキスト情報を抽出して，全文検索用のインデクスファイルを作成し，これを全文検索インデクスに登録するユティリティについて知りたい場合に参照してください。

DocumentBroker Rendering Option Version 3（3020-3-N47）

Adobe Acrobat Distiller と連携した DocumentBroker Rendering Option と連携して，レンディション変換要求機能を使用するために必要な DocumentBroker Rendering Option の知識およびコマンドの文法について知りたい場合に参照してください。

DocumentBroker Rendering Option Version 3（**活文** PDFstaff **編**）（3020-3-N49）

PDFstaff Runtime および PDFstaff SDK と連携した DocumentBroker Rendering Option と連携して，レンディション変換要求機能を使用するために必要な DocumentBroker Rendering Option の知識およびコマンドの文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）

- **スケーラブルデータベースサーバ**　HiRDB Version 6　UAP **開発ガイド**（UNIX(R)/Windows(R) **用**）（3000-6-236）※1
- **スケーラブルデータベースサーバ**　HiRDB Version 6　UAP **開発ガイド**（Windows(R) **用**）（3020-6-126）※1
- **スケーラブルデータベースサーバ**　HiRDB Version 7　UAP **開発ガイド**（UNIX(R)/Windows(R) **用**）（3000-6-276）※1

- **スケーラブルデータベースサーバ** HiRDB Version 7 UAP **開発ガイド**（Windows(R) **用**）（3020-6-276）※1
- **スケーラブルデータベースサーバ** HiRDB Version 8 UAP **開発ガイド**（3020-6-356）※1
- **スケーラブルデータベースサーバ** HiRDB Version 6 SQL **リファレンス**（UNIX(R)/Windows(R) **用**）（3000-6-237）※2
- **スケーラブルデータベースサーバ** HiRDB Version 6 SQL **リファレンス**（Windows(R) **用**）（3020-6-127）※2
- **スケーラブルデータベースサーバ** HiRDB Version 7 SQL **リファレンス**（UNIX(R)/Windows(R) **用**）（3000-6-277）※2
- **スケーラブルデータベースサーバ** HiRDB Version 7 SQL **リファレンス**（Windows(R) **用**）（3020-6-277）※2
- **スケーラブルデータベースサーバ** HiRDB Version 8 SQL **リファレンス**（3020-6-357）※2
- **スケーラブルデータベースサーバ** HiRDB Version 6 **メッセージ**（UNIX(R)/Windows(R) **用**）（3000-6-238）※3
- **スケーラブルデータベースサーバ** HiRDB Version 6 **メッセージ**（Windows(R) **用**）（3020-6-128）※3
- **スケーラブルデータベースサーバ** HiRDB Version 7 **メッセージ**（UNIX(R)/Windows(R) **用**）（3000-6-278）※3
- **スケーラブルデータベースサーバ** HiRDB Version 7 **メッセージ**（Windows(R) **用**）（3020-6-278）※3
- **スケーラブルデータベースサーバ** HiRDB Version 8 **メッセージ**（3020-6-358）※3
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 2（3000-6-245）（3020-6-140）※4
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 7（3000-6-288）※4
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 8（3020-6-375）※4

注※1

このマニュアルでは，これらのマニュアルを「HiRDB UAP 開発ガイド」と表記しています。

注※2

このマニュアルでは，これらのマニュアルを「HiRDB SQL リファレンス」と表記しています。

注※3

このマニュアルでは，これらのマニュアルを「HiRDB メッセージ」と表記しています。

注※4

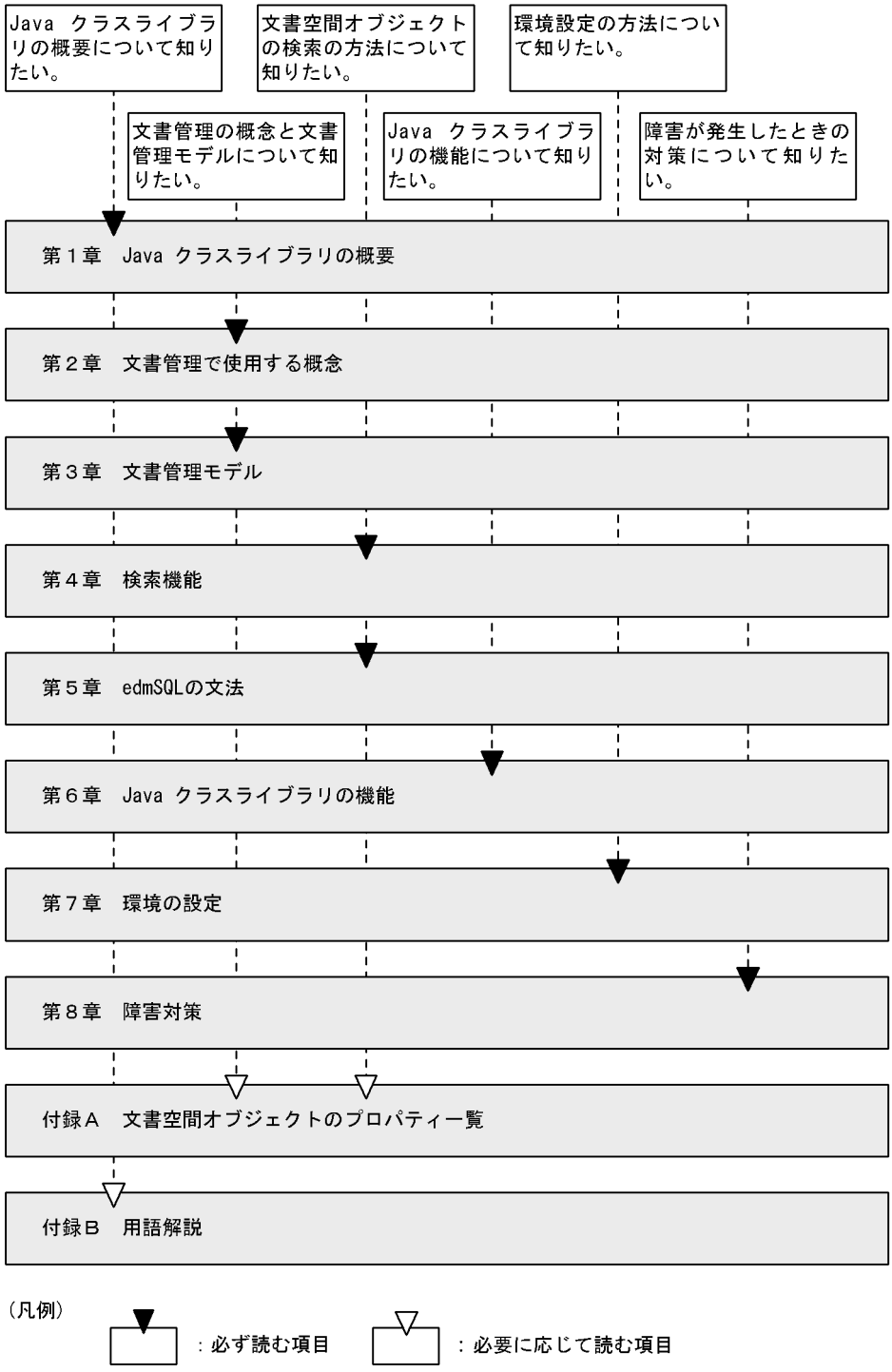
このマニュアルでは，これらのマニュアルを「HiRDB Text Search Plug-in」と表記しています。

関連製品のマニュアル（その他）

- Hitachi Web Server（3000-3-350）

■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。利用目的別に，次の流れに従ってお読みいただくことをお勧めします。



■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
AIX 5L V5.1	AIX
AIX 5L V5.2	
AIX 5L V5.3	

製品名称	略称
Cosminexus Studio Version 5	Cosminexus
Cosminexus Application Server Version 5	
Cosminexus Developer Version 5	
Cosminexus Application Server Version 6	
Cosminexus Developer Light Version 6	
Cosminexus Developer Standard Version 6	
Cosminexus Developer Professional Version 6	
uCosminexus Application Server Standard Version 6.7	
uCosminexus Application Server Enterprise Version 6.7	
uCosminexus Developer Light Version 6.7	
uCosminexus Developer Standard Version 6.7	
uCosminexus Developer Professional Version 6.7	
uCosminexus Application Server Standard Version 7	
uCosminexus Application Server Enterprise Version 7	
uCosminexus Developer Standard Version 7	
uCosminexus Developer Professional Version 7	
uCosminexus Application Server Standard Version 8	
uCosminexus Application Server Enterprise Version 8	
uCosminexus Developer Standard Version 8	
uCosminexus Developer Professional Version 8	
uCosminexus DocumentBroker Development Kit Version 3	DocumentBroker
uCosminexus DocumentBroker Runtime Version 3	
uCosminexus DocumentBroker Server Version 3	
uCosminexus DocumentBroker Development Kit Version 3	DocumentBroker Development Kit
uCosminexus DocumentBroker Runtime Version 3	DocumentBroker Runtime
uCosminexus DocumentBroker Server Version 3	DocumentBroker Server
uCosminexus DocumentBroker Development Kit Version 3	DocumentBroker クライアント
uCosminexus DocumentBroker Runtime Version 3	
uCosminexus DocumentBroker Server Version 3	DocumentBroker サーバ
DABroker	DABroker
DABroker for C++	
HiRDB/Single Server Version 6	HiRDB
HiRDB/Single Server Version 7	
HiRDB/Single Server Version 8	
HiRDB/Parallel Server Version 6	
HiRDB/Parallel Server Version 7	
HiRDB/Parallel Server Version 8	
HiRDB/Single Server - Object Option Version 6	HiRDB Object Option
HiRDB/Parallel Server - Object Option Version 6	
HiRDB/Single Server - Object Option Version 7	

製品名称	略称	
HiRDB/Parallel Server - Object Option Version 7		
HiRDB Text Search Plug-in Version 2	HiRDB Text Search Plug-in	
HiRDB Text Search Plug-in Version 7		
HiRDB Text Search Plug-in Version 8		
IBM VisualAge C++ Professional for AIX V5	VisualAge C++ Professional for AIX	
IBM VisualAge C++ Professional for AIX V6		
IBM XL C/C++ Enterprise Edition V7 for AIX	IBM XL C/C++ Enterprise Edition for AIX	
IBM XL C/C++ Enterprise Edition V8 for AIX		
iPlanet™ Web Server Enterprise Edition	iPlanet Web Server Enterprise Edition	
JBuilder(TM) 2005 Enterprise	JBuilder	
JBuilder(TM) 9 Enterprise		
Microsoft(R) Active Directory(R)	Active Directory	
Microsoft(R) Internet Explorer(R)	Internet Explorer	
Microsoft(R) Internet Information Services	Internet Information Services	
Microsoft(R) Office Word	Word	
Microsoft(R) Word		
Microsoft(R) Windows Server(R) 2003, Enterprise Edition 日本語版	Windows Server 2003, Enterprise Edition	Windows Server 2003
Microsoft(R) Windows Server(R) 2003, Standard Edition 日本語版	Windows Server 2003, Standard Edition	
Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition 日本語版	Windows Server 2003 R2, Enterprise Edition	Windows Server 2003 R2
Microsoft(R) Windows Server(R) 2003 R2, Standard Edition 日本語版	Windows Server 2003 R2, Standard Edition	
Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition 日本語版	Windows Server 2003 R2, Enterprise Edition (x64)	Windows Server 2003 R2 または Windows Server 2003 R2 x64 Edition
Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition 日本語版	Windows Server 2003 R2, Standard Edition (x64)	
Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit 日本語版	Windows Server 2008 Enterprise Edition x86	Windows Server 2008 または Windows Server 2008 x86
Microsoft(R) Windows Server(R) 2008 Standard 32-bit 日本語版	Windows Server 2008 Standard Edition x86	
Microsoft(R) Windows Server(R) 2008 R2 Enterprise 日本語版	Windows Server 2008 R2 Enterprise	Windows Server 2008 R2
Microsoft(R) Windows Server(R) 2008 R2 Standard 日本語版	Windows Server 2008 R2 Standard	
Microsoft(R) Windows Server(R) 2012 Standard 日本語版	Windows Server 2012 Standard	Windows Server 2012
Microsoft(R) Windows Server(R) 2012 Datacenter 日本語版	Windows Server 2012 Datacenter	
Microsoft(R) Windows(R) XP Professional Operating System	Windows XP Professional	Windows XP
Microsoft(R) Windows Vista(R) Ultimate 日本語版 (32 ビット版)	Windows Vista Ultimate	Windows Vista

製品名称	略称	
Microsoft(R) Windows Vista(R) Business 日本語版 (32 ビット版)	Windows Vista Business	
Microsoft(R) Windows Vista(R) Enterprise 日本語版 (32 ビット版)	Windows Vista Enterprise	
Microsoft(R) Windows(R) 7 Professional 日本語版 (32 ビット版)	Windows 7 Professional x86	Windows 7
Microsoft(R) Windows(R) 7 Enterprise 日本語版 (32 ビット版)	Windows 7 Enterprise x86	
Microsoft(R) Windows(R) 7 Ultimate 日本語版 (32 ビット版)	Windows 7 Ultimate x86	
Microsoft(R) Windows(R) 7 Professional 日本語版 (64 ビット版)	Windows 7 Professional x64	
Microsoft(R) Windows(R) 7 Enterprise 日本語版 (64 ビット版)	Windows 7 Enterprise x64	
Microsoft(R) Windows(R) 7 Ultimate 日本語版 (64 ビット版)	Windows 7 Ultimate x64	
Windows(R) 8 Pro 日本語版（32 ビット版）	Windows 8 Pro	Windows 8
Windows(R) 8 Pro 日本語版（64 ビット版）		
Windows(R) 8 Enterprise 日本語版（32 ビット版）	Windows 8 Enterprise	
Windows(R) 8 Enterprise 日本語版（64 ビット版）		
Sun Java(TM) System Directory Server 5.1	Sun Java System Directory Server	
Sun Java(TM) System Directory Server 5.2		
Sun Java(TM) System Directory Server 6.3		
TPBroker Developer	TPBroker	
TPBroker for C++		
TPBroker		
活文 PDFstaff Runtime	PDFstaff Runtime	
活文 PDFstaff SDK	PDFstaff SDK	

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。

- uCosminexus DocumentBroker Development Kit および uCosminexus DocumentBroker Runtime が提供する Java 言語対応のクラスライブラリを Java クラスライブラリと表記します。
- AIX を UNIX と表記することがあります。
- Windows Server 2003，Windows Server 2003 R2，Windows Server 2003 R2 x64 Edition，Windows Server 2008 x86，Windows Server 2008 R2，Windows Server 2012，Windows XP，Windows Vista，Windows 7，および Windows 8 を合わせて Windows と表記することがあります。
- Java^{TM} を Java と表記します。
- Enterprise JavaBeans^{TM} を Enterprise JavaBeans と表記します。
- JavaBeans^{TM} を JavaBeans と表記します。
- Java Naming and Directory Interface^{TM} を JNDI と表記します。
- JavaServer Pages^{TM} を JSP と表記します。
- Java^{TM} Servlet を Servlet と表記します。
- Java^{TM} 2 Platform, Enterprise Edition を Java 2 Platform, Enterprise Edition または J2EE と表記します。
- Java^{TM} 2 Runtime Environment, Standard Edition を Java 2 Runtime Environment, Standard Edition と表記します。
- Java^{TM} 2 Software Development Kit, Standard Edition を Java 2 SDK, Standard Edition と表記します。

■ uCosminexus DocumentBroker のマニュアルで使用する略語

uCosminexus DocumentBroker 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
AIIM	Association for Information and Image Management International
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BES	Back End Server
BLOB	Binary Large Object
BMP	Bit Map
BNF	Backus Normal Form
BOA	Basic Object Adapter
CD-ROM	Compact Disc Read Only Memory
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CR	Carriage Return
CSV	Comma Separated Value
DAP	Directory Access Protocol
DAT	Digital Audio Tape
DB	Database
DBMS	Database Management System
DCD	Document Content Description
DDE	Dynamic Data Exchange
DIT	Directory Information Tree
DLL	Dynamic Linking Library
DMA	Document Management Alliance
DN	Distinguished Name
DTD	Document Type Definition
EOF	End of File
ESIS-B	Element Structure Information Set-Binary Format
EUC	Extended UNIX Code
GIF	Graphics Interchange Format
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol

英略語	英字での表記
IANA	Internet Assigned Numbers Authority
ID	Identifier
IPF	Itanium(R) Processor Family
ISO	International Organization for Standardization
JIS	Japanese Industrial Standards
JPEG	Joint Photographic Expert Group
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LF	Line Feed
MFC	Microsoft Foundation Class
MIME	Multipurpose Internet Mail Extensions
OCR	Optical Character Reader
OIID	Object Instance Identifier
OLE	Object Linking and Embedding
OMG	Object Management Group
ORB	Object Request Broker
ORDB	Object Relational Database
OS	Operating System
OTS	Object Transaction Service
PC	Personal Computer
PDF	Portable Document Format
RDB	Relational Database
RDN	Relative Distinguished Name
RFC	Request for Comment
RTF	Rich Text Format
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TIFF	Tag Image File Format
UCS-2	Universal Character Set coded in 2 octets
UCS-4	Universal Character Set coded in 4 octets
UNC	Universal Naming Convention
UOC	User Own Coding
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
UTF-16	16-bit UCS Transformation Format
UTF-8	8-bit UCS Transformation Format

英略語	英字での表記
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

■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。
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを示します。項目が横に並べられ，記号｜で区切られている場合は，そのうちの一つを選択します。 （例） {A ｜ B ｜ C} A，B または C のどれかを指定することを示します。
〔 〕	この記号で囲まれている項目は省略してもよいことを意味します。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。
＿	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。

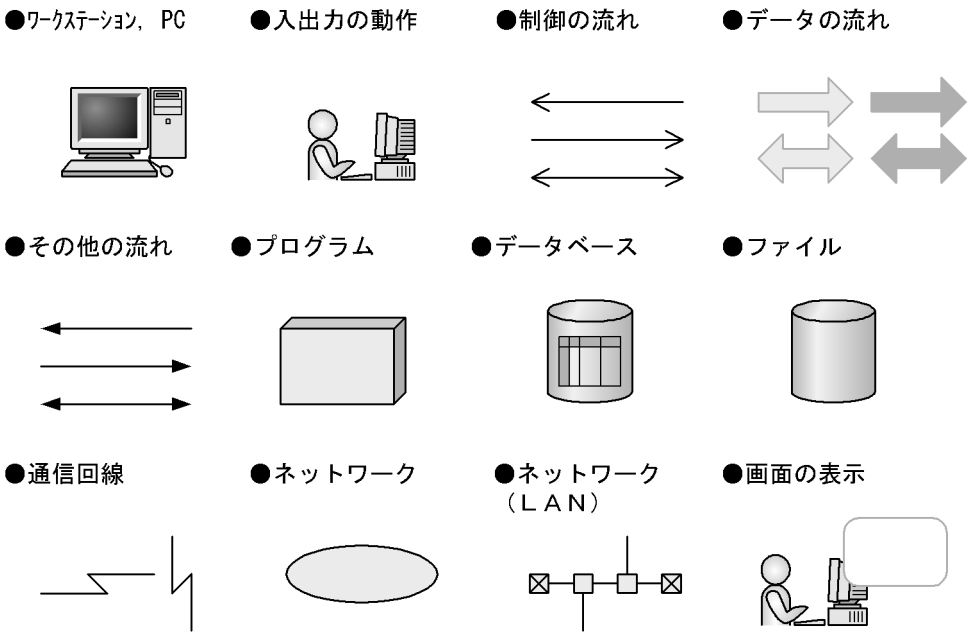
構文要素記号	意味
英字	A ～ Z　a ～ z
英小文字	a ～ z
英大文字	A ～ Z
数字	0 ～ 9
英数字	A ～ Z　a ～ z　0 ～ 9
16 進数字	0 ～ 9　A ～ F　a ～ f

構文要素記号	意味
記号	! " # $ % & ' ( ) + , _ ― ・ . / : ; < = > @ [ ] ^ ― { } ¯ 空白 ¥

注　すべて半角文字を使用してください。

■このマニュアルの図中で使用する記号

このマニュアルの図中で使用する記号を，次のように定義します。



■常用漢字以外の漢字の使用について

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

個所（かしょ）

■ KB（キロバイト）などの単位表記について

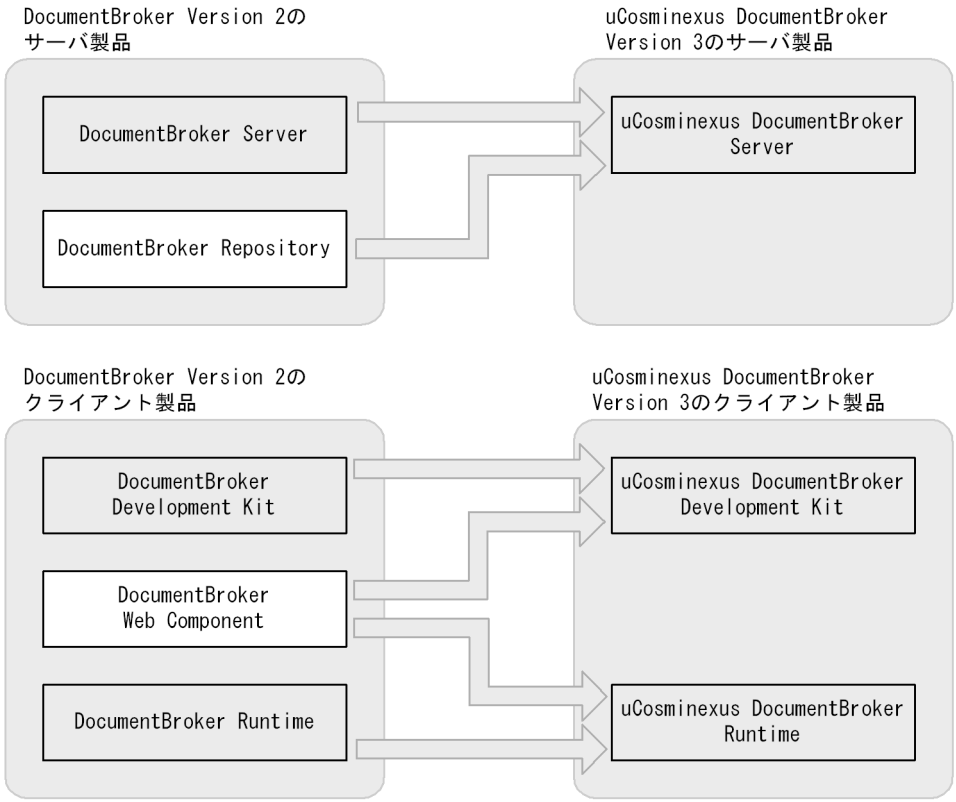
1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

■ DocumentBroker Version 2 と uCosminexus DocumentBroker Version 3 の製品体系の違い

uCosminexus DocumentBroker Version 3 では次のように製品の統合を行いました。

- DocumentBroker Repository を uCosminexus DocumentBroker Server に統合しました。
- DocumentBroker Web Component を uCosminexus DocumentBroker Development Kit および uCosminexus DocumentBroker Runtime に統合しました。

DocumentBroker Version 2 と uCosminexus DocumentBroker Version 3 の製品体系の違いを次に示します。



■ DocumentBroker Version 2 と uCosminexus DocumentBroker Version 3 のマニュアルの対応

バージョンアップおよび製品体系の変更に伴い，uCosminexus DocumentBroker Version 3 では次に示すようにマニュアル名称を変更しました。

Version 2 のマニュアル名称	Version 3 のマニュアル名称
DocumentBroker Version 2　システム導入・運用ガイド	uCosminexus DocumentBroker Version 3　システム導入・運用ガイド
DocumentBroker Version 2　クラスライブラリ　解説	uCosminexus DocumentBroker Version 3　クラスライブラリ　C++　解説
DocumentBroker Version 2　クラスライブラリ　リファレンス　基本機能編	uCosminexus DocumentBroker Version 3　クラスライブラリ　C++　リファレンス　基本機能編
DocumentBroker Version 2　クラスライブラリ　リファレンス　概念 SGML 文書管理機能編	廃版
DocumentBroker Version 2　オブジェクト操作ツール	uCosminexus DocumentBroker Version 3　オブジェクト操作ツール
DocumentBroker Version 2　統計解析ツール	uCosminexus DocumentBroker Version 3　統計解析ツール
DocumentBroker Version 2　メッセージ	uCosminexus DocumentBroker Version 3　メッセージ
DocumentBroker Web Component Version 2　解説	uCosminexus DocumentBroker Version 3　クラスライブラリ　Java　解説
DocumentBroker Web Component Version 2　リファレンス	uCosminexus DocumentBroker Version 3　クラスライブラリ　Java　リファレンス

Version 2 のマニュアル名称	Version 3 のマニュアル名称
DocumentBroker Web Component Version 2　サンプル Web アプリケーション	uCosminexus DocumentBroker Version 3　クラスライブラリ　Java　サンプル Web アプリケーション
DocumentBroker Text Search Index Loader Version 2	uCosminexus DocumentBroker Text Search Index Loader Version 3
DocumentBroker Rendering Option　システム導入・運用ガイド	uCosminexus DocumentBroker Rendering Option Version 3
DocumentBroker Object Loader Version 2	uCosminexus DocumentBroker Object Loader Version 3

目次

1 Java クラスライブラリの概要

この章では，Java 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，および処理の流れについて説明します。

1.1 Java クラスライブラリとは

DocumentBroker Development Kit および DocumentBroker Runtime は，文書を管理するユーザアプリケーションプログラムを開発・実行する際に使用できる，Java 言語に対応したクラスライブラリ（Java クラスライブラリ）を提供しています。

この Java クラスライブラリを使用して開発するユーザアプリケーションプログラムは，Java 環境を経由して DocumentBroker サーバにアクセスし，文書の作成，参照，更新などの操作ができます。

1.1.1 Java クラスライブラリの特長

Java クラスライブラリの特長を次に示します。

● **ユーザアプリケーションプログラムをコンポーネント化して再利用できます。**
Java のコンポーネント化技術である JavaBeans 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。ユーザアプリケーションプログラムのコンポーネント化による再利用性の向上と文書管理システムの構築期間の短縮化が図れます。

● **OS に依存しないユーザアプリケーションプログラムを開発できます。**
Java 言語を使用して開発するユーザアプリケーションプログラムは，Java 仮想マシン上で動作します。このため，OS に依存しないユーザアプリケーションプログラムを開発できます。

● **サンプル Web アプリケーションを提供しています。**
Java クラスライブラリを使用して開発したアプリケーションプログラムを，サンプルとして提供しています。このサンプルを，**サンプル Web アプリケーション**といいます。サンプル Web アプリケーションは，文書の検索，ダウンロード，登録，削除，フォルダを使用した管理，アクセス制御など，基本的な文書管理機能を備えたアプリケーションプログラムです。
サンプル Web アプリケーションは，Servlet，JSP，JavaBeans，そのほかの Java クラスなどのコンポーネントによって構成されています。サンプル Web アプリケーションのそれぞれのコンポーネントでは，Java 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の具体的な使用方法が示されています。
ユーザアプリケーションプログラムを開発する際に，サンプル Web アプリケーションで示されているアーキテクチャを参考に設計・構築すると，効率良くユーザアプリケーションプログラムが開発できます。
なお，使用する文字コード種別が UTF-8 の場合，サンプル Web アプリケーションは使用できません。
サンプル Web アプリケーションの詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　サンプル Web アプリケーション」を参照してください。

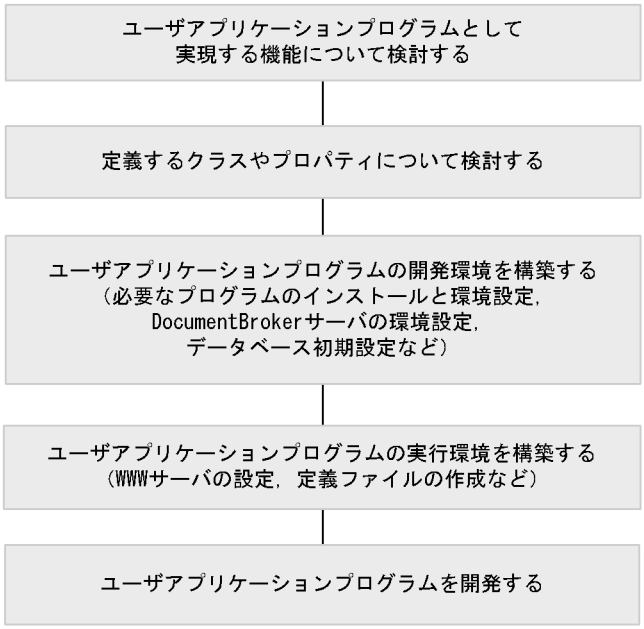
1.1.2 ユーザアプリケーションプログラムの開発の流れ

Java クラスライブラリを使用すると，業務に対応する文書管理体系を DocumentBroker サーバに定義して，定義に従って構築されたデータベースを操作するためのユーザアプリケーションプログラムを開発できます。

ユーザアプリケーションプログラム開発時には，どのような業務に対応するユーザアプリケーションプログラムを開発するのか，どのような文書管理体系にするのか，また，そのためにはどのような定義でデータベースを構築する必要があるのかを，詳細に検討しておく必要があります。

ユーザアプリケーションプログラムの開発の流れを次の図に示します。

図 1-1　ユーザアプリケーションプログラムの開発の流れ



それぞれの手順での詳細は，次の記述個所を参照してください。

● ユーザアプリケーションプログラムとして実現する機能について検討する場合
- 「3.　文書管理モデル」
- 「4.　検索機能」
- 「6.　Java クラスライブラリの機能」

● 定義するクラスやプロパティについて検討する場合
- 「2.　文書管理で使用する概念」

● DocumentBroker サーバの環境設定，クラスやプロパティの定義方法，データベースの構築方法などを確認する場合
- マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」

なお，ユーザアプリケーションプログラムのアーキテクチャや，Java クラスライブラリの具体的な使用方法については，サンプル Web アプリケーションを参考にすることができます。サンプル Web アプリケーションについては，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　サンプル Web アプリケーション」を参照してください。

1.2 Java クラスライブラリで実現する機能

この節では，Java クラスライブラリによって実現する文書管理機能について説明します。なお，次の場合は XML 文書管理機能を使用できません。

- Windows Server 2008，Windows Server 2008 R2，Windows Server 2012，Windows XP，Windows Vista，Windows 7，および Windows 8 を使用する場合
- AIX を使用する場合
- 文書空間で使用する文字コード種別が UTF-8 の場合

1.2.1 文書空間（データベース）に接続する機能

DocumentBroker では，文書および文書を管理する仕組みをオブジェクトの概念で表現し，オブジェクトを操作することによって，文書の参照，更新などの文書管理を実現しています。オブジェクトに対する操作を設計したユーザアプリケーションプログラムは，DocumentBroker サーバの文書空間にアクセスして，データベースに格納されているオブジェクトを操作できます。

Java クラスライブラリでは，ユーザアプリケーションプログラムと，DocumentBroker サーバの文書空間を接続する機能を提供しています。文書空間に接続してオブジェクトを作成すると，その時点で，データベースの内容も更新されます。この更新を確定またはキャンセルしたり，文書空間との接続を確認したり，トランザクションを制御したりする機能も提供しています。

なお，DocumentBroker サーバの文書空間では，Shift-JIS または UTF-8 のどちらかの文字コード種別を使用できます。ただし，使用する文字コード種別が UTF-8 の場合，DocumentBroker で使用できるクラス，インターフェースおよびメソッドに制限があります。文書空間で使用する文字コード種別が UTF-8 の場合に使用できるクラス，インターフェースおよびメソッドについては，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」を参照してください。

1.2.2 文書を作成，管理する機能

Java クラスライブラリで管理する文書には，次の種類があります。

- 一つのファイルから構成される文書
- 一つの文書の中に複数のバージョンに対応するファイルを持つ文書
- 一つの文書の中に複数の形式に対応するファイルを持つ文書

また，論理構造を持つ文書として，XML 文書を管理できます（ただし，AIX の場合を除きます）。

文書を管理する方法には，次の種類があります。

- 文書と文書を関連付けて管理する方法
- 文書とフォルダを関連付けて管理する方法
- 文書と構成管理できるフォルダを関連付けて管理する方法
- 文書やフォルダの属性を管理する方法
- 文書やフォルダを一括して管理する方法

(1) 一つのファイルから構成される文書を管理する

一つのファイルから構成される文書は，最も基本的な文書になります。

例えば，Word で作成したファイル「報告書 .doc」を，DocumentBroker の文書のコンテンツ（Word やテキストエディタなどのアプリケーションプログラムで作成された文書ファイル）として管理できます。

(2) 一つの文書の中に複数のバージョンに対応するファイルを持つ文書を管理する

文書を更新する時に，履歴を残して管理できます。文書に対して，第 1 版，第 2 版などのバージョンを付けておくことで，過去のある時点での文書を取り出したり，その文書を基に新たな文書を作成したりできます。また，その文書が幾つのバージョンを持っているかということも管理できます。

この機能を，**バージョン管理機能**といいます。

(3) 一つの文書の中に複数の形式に対応するファイルを持つ文書を管理する

複数の形式に対応するファイルを一つの文書のコンテンツとして管理できます。

この機能は，一つのファイルの内容を，対応するアプリケーションプログラムごとの複数の形式に変換した場合などに使用できます。

例えば，文書「報告書」のコンテンツとして，Word で作成したファイル「報告書 .doc」のほかに，「報告書 .doc」を PDF 形式に変換したファイル「報告書 .pdf」も一緒に登録できます。これによって，ユーザの目的やユーザが使用するマシンの環境などの利用形態に応じて，Word 形式のファイルをダウンロードしたり，PDF 形式のファイルをダウンロードしたりできます。

この機能を，**マルチレンディション機能**といいます。

(4) ファイルの実体が任意のディレクトリに登録されている文書を管理する

文書の実体であるコンテンツを，DocumentBroker サーバが存在するマシンから接続できるファイルシステムの任意のディレクトリに格納し，文書のプロパティとコンテンツの格納先の情報（**コンテンツロケーション**）をデータベースに登録して管理します。コンテンツロケーションには，コンテンツの格納先の基点となるディレクトリパス（**コンテンツ格納先ベースパス**）からの相対パスが登録されます。コンテンツの格納先を移行する場合などは，データベースに影響を与えることなく，コンテンツ格納先ベースパスを変更するだけで，コンテンツの格納領域を変更できます。また，コンテンツをデータベースに格納しないため，データベースの容量を削減できます。

この機能を，**リファレンスファイル管理機能**といいます。

(5) XML 文書を管理する

XML 文書とは，XML 形式のファイル（XML ファイル）を文書のコンテンツとして登録した文書です。XML ファイルは，タグによって定義された論理構造を持っています。Java クラスライブラリでは，任意のタグに囲まれた文字列を文書のプロパティの値として割り当てて管理したり，特定の構造に含まれる文字列を指定した検索を実行したりできます。

この機能を，XML **文書管理機能**といいます。

(6) 文書と文書を関連付けて管理する

文書をほかの文書と関連付けて管理できます。ある文書の中で，ほかの文書を参照する記述がある場合などに，文書間にリンクを設定しておくことで，参照元の文書から参照先の文書をたどることができます。また，参照先の文書から参照元の文書をたどることもできます。

Java クラスライブラリでは，文書と文書，文書とフォルダなどの関連づけを表すリンクをオブジェクト

（リンクオブジェクト）として管理できます。

(7) 文書とフォルダを関連付けて管理する

文書は，フォルダに格納してまとめて管理したり，フォルダに格納した文書を，別の観点から分類して管理したりできます。

Java クラスライブラリでは，複数の文書をまとめて管理するために，フォルダというオブジェクトを使用します。フォルダと文書にリンクを設定することによって，文書をフォルダにまとめて格納して管理したり，文書を複数の観点から分類して管理したりできます。設定するリンクの種類によって，フォルダと文書は 1:n（n は 1 以上の整数）の関係で関連付けたり，m:n（m，n は 1 以上の整数）の関係で関連付けたりできます。

例えば，文書をプロジェクト単位でまとめて管理したい場合，プロジェクトごとにフォルダを作成して，1:n の関係でフォルダと文書のリンクを設定すれば，プロジェクトごとに文書を管理できます。また，フォルダに格納された文書を作成者とテーマで分類して管理したい場合，作成者を表すフォルダとテーマを表すフォルダをそれぞれ作成して，m:n の関係でフォルダと文書のリンクを設定すれば，文書を作成者やテーマごとに分類して管理できます。

また，フォルダとフォルダを関連付けることによって，フォルダや分類に階層を持たせることもできます。

(8) 文書を構成管理する

フォルダでは，文書の特定のバージョンを管理することもできます。文書やフォルダのバージョンを指定して管理するには，**構成管理機能**を持つフォルダを使用します。構成管理機能とは，管理している文書やフォルダの，ある時点でのバージョン構成の状態を管理する機能です。例えば，管理する複数の文書が，それぞれ異なるタイミングでバージョンアップする場合などに，構成管理機能を使用します。

構成管理機能を持つフォルダでは，それぞれの文書の最新のバージョンを参照できるようにしたり，文書のバージョンアップに関係なく常にバージョン 1 やバージョン 2 など特定のバージョンの文書を参照できるようにしたりできます。ある文書は常に最新のバージョンを参照して，別の文書は特定のバージョンを参照し続ける，ということもできます。

また，このフォルダをバージョンアップすることで，フォルダごとに関連付けている文書やフォルダのバージョン構成をまとめて管理できます。例えば，バージョン 1 のフォルダからたどる文書をバージョン 1 で固定したり，バージョン 2 のフォルダで管理する文書は常に最新のバージョンをたどるようにしたりできます。

(9) 文書やフォルダの属性を管理する

Java クラスライブラリでは，文書やフォルダにさまざまな属性を付けて管理できます。この属性をプロパティといいます。文書やフォルダにプロパティを定義すると，プロパティを指定して文書やフォルダを検索できるので便利です。

プロパティには，システムによって定義されるプロパティと，ユーザが任意に追加定義するプロパティがあります。例えば，「報告書 .doc」を管理する場合，ユーザが追加したプロパティの値として，「作成者」，「担当部署」，「コメント」などの情報も一緒に管理できます。

バージョン管理できる文書およびフォルダでは，バージョンに共通するプロパティと，個々のバージョンのプロパティを個別に管理できます。また，リンクオブジェクトにプロパティを設定して管理することもできます。

(10) 文書やフォルダを一括して管理する

Java クラスライブラリでは，複数の文書やフォルダを一括して管理できます。このため，複数の文書のプ

ロパティを一括して取得したり，複数の文書を一括して削除したりできます。

1.2.3 文書やフォルダを検索する機能

すでに登録されている文書やフォルダは，検索によって取得できます。

Java クラスライブラリでは，文書やフォルダに設定されているプロパティの値を使用した属性検索や，文書に含まれる文字列を指定した全文検索ができます。

全文検索では，文章を指定して，その文章と類似した概念を持つ文書を検索（概念検索）できます。

全文検索を実行するには，前提プログラムとして HiRDB Text Search Plug-in が必要です。

検索を実行する場合の検索条件は，edmSQL によって指定できます。edmSQL は DocumentBroker が提供する機能であり，SQL に基づいた構文で記述できます。このため，SQL の知識を活用して検索を実行できます。

1.2.4 独立データを管理する機能

文書やフォルダの体系に関係なく，独立したデータを作成，管理できます。また，独立データには，プロパティを定義して管理できます。

1.2.5 そのほかの機能

そのほかの機能について説明します。

(1) アクセス制御機能

Java クラスライブラリで管理している文書やフォルダに対して参照や更新などのアクセスをする場合，そのユーザが持つアクセス権に応じてアクセスを制御できます。例えば，作成した文書をほかのユーザに参照はさせても更新はさせないようにしたい場合などに，この機能を使用します。これを，**アクセス制御機能**といいます。アクセスを制御するためのアクセス権は，ユーザやグループごとに設定できます。

(2) 排他制御機能

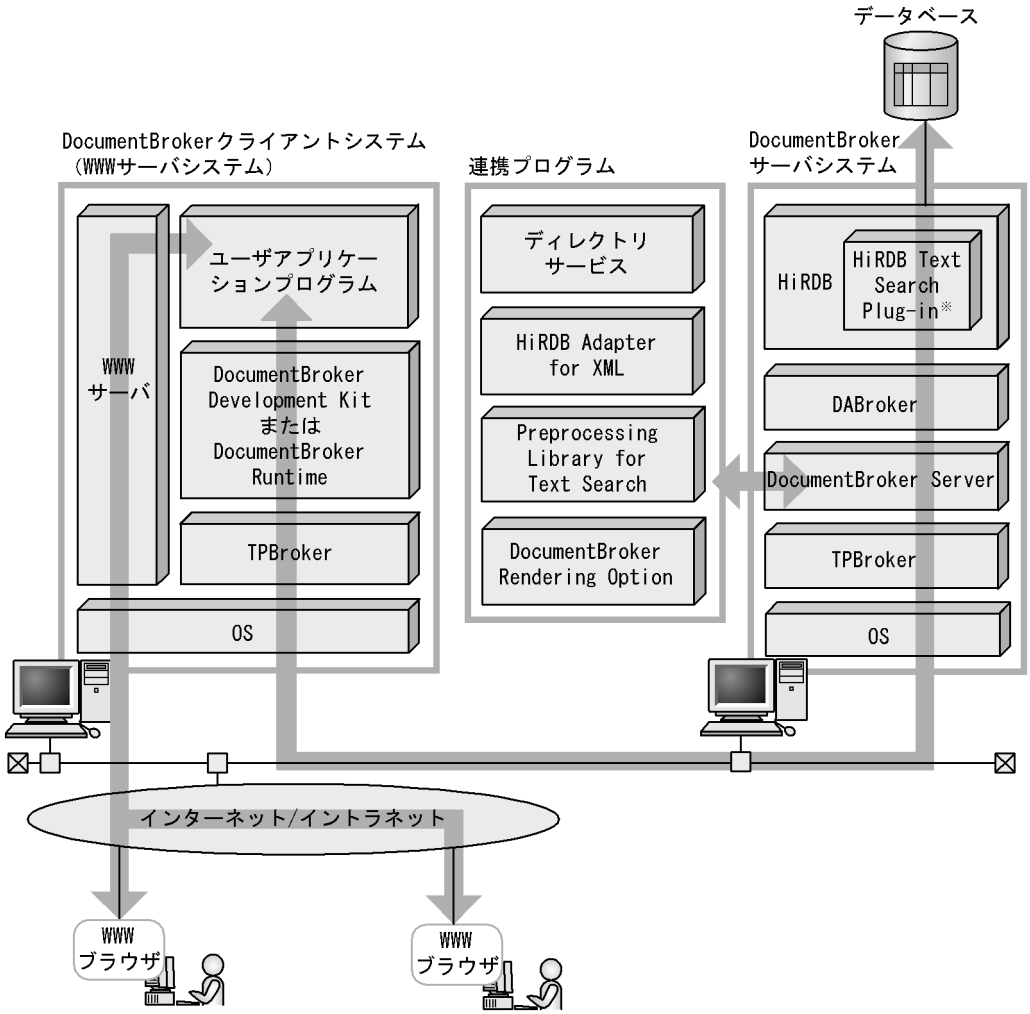
Java クラスライブラリで管理している文書やフォルダにアクセスする場合，排他制御が実行されます。排他制御とは，あるユーザが文書やフォルダにアクセスしているときに，ほかのユーザによってその文書やフォルダの状態が変更されたり，削除されたりしないようにロックを設定して制御する機能です。

Java クラスライブラリの排他制御には，メソッドによって暗黙的に設定されるロックと，ユーザが明示的に設定するロックがあります。

1.3 Java クラスライブラリを使用する場合のシステム構成

この節では，DocumentBroker クライアントシステムを DocumentBroker Development Kit および DocumentBroker Runtime の Java クラスライブラリを使用して開発する場合のシステム構成について説明します。システム構成を次の図に示します。

図 1-2 Java クラスライブラリを使用する場合のシステム構成



注※ 全文検索機能を使用する場合は，HiRDB Text Search Plug-inが必要です。さらに概念検索機能を使用する場合は，HiRDB Text Search Plug-in Conceptual Extensionが必要です。

DocumentBroker サーバシステムと DocumentBroker クライアントシステムは別マシンで運用することをお勧めします。なお，DocumentBroker サーバシステムと DocumentBroker クライアントシステムを別マシンで運用する場合は，ファイル転送機能を使用します。ファイル転送機能については，「7.4.5 ファイル転送機能を使用するための環境設定」を参照してください。

また，サンプル Web アプリケーションを使用する場合のシステム構成および前提プログラムについては，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　サンプル Web アプリケーション」を参照してください。

1.3.1 DocumentBroker サーバシステムの前提プログラム

DocumentBroker サーバシステムには，次に示すプログラムが必要です。

● OS

DocumentBroker サーバシステムでは，次に示すオペレーティングシステムを使用します。

- **AIX の場合：**
  AIX 5L V5.1，AIX 5L V5.2，または AIX 5L V5.3 を使用してください。
- **Windows Server 2003 の場合：**
  Windows Server 2003，Enterprise Edition または Windows Server 2003，Standard Edition を使用してください。
- **Windows Server 2003 R2 の場合：**
  Windows Server 2003 R2，Enterprise Edition または Windows Server 2003 R2，Standard Edition を使用してください。
- **Windows Server 2003 R2 x64 Edition の場合：**
  Windows Server 2003 R2，Enterprise x64 Edition，または Windows Server 2003 R2，Standard x64 Edition を使用してください。
- **Windows Server 2008 x86 の場合：**
  Windows Server 2008 Enterprise x86 Edition，または Windows Server 2008 Standard x86 Edition を使用してください。
- **Windows Server 2008 R2 の場合：**
  Windows Server 2008 R2 Enterprise，または Windows Server 2008 R2 Standard を使用してください。
- **Windows Server 2012 の場合：**
  Windows Server 2012 Standard，または Windows Server 2012 Datacenter を使用してください。

● TPBroker

DocumentBroker は，CORBA 仕様に準拠した ORB を使用する分散ネットワーク環境で運用されます。DocumentBroker では，ORB に TPBroker を使用します。TPBroker は，CORBA を使用したクライアントとサーバ間の通信機能を提供するプログラムです。

● DocumentBroker Server

サーバシステムのサーバプログラムです。DocumentBroker のサーバ機能を提供します。

● DABroker

DocumentBroker サーバシステムでは，DABroker を使用してデータベースにアクセスします。

● HiRDB

文書は，テキストファイルのように文章を編集したファイルだけでなく，図や画像ファイルも組み合わせて構成される場合があります。そのため，DocumentBroker では ORDB をデータベースに使用します。DocumentBroker サーバシステムでは，データベースの ORDB マネージメントシステムに HiRDB を使用します。

また，HiRDB Object Option※が必要です。なお，管理する文書を全文検索する場合は，HiRDB Text Search Plug-in が必要です。また，さらに概念検索機能を使用する場合は，HiRDB Text Search Plug-in Conceptual Extension が必要です。

注※

HiRDB Version 6 または HiRDB Version 7 を使用する場合に HiRDB Object Option が必要です。
HiRDB Version 8 以降を使用する場合，HiRDB Object Option は必要ありません。

1.3.2 DocumentBroker クライアントシステムの前提プログラム

DocumentBroker クライアントシステムには，次に示すプログラムが必要です。なお，WWW サーバ，ユーザアプリケーションプログラム，DocumentBroker Development Kit（または DocumentBroker Runtime）は同じマシン上に構築する必要があります。

● OS

DocumentBroker クライアントシステムでは，次に示すオペレーティングシステムを使用します。

- AIX **の場合：**
  AIX 5L V5.1，AIX 5L V5.2，または AIX 5L V5.3 を使用してください。
- Windows Server 2003 **の場合：**
  Windows Server 2003，Enterprise Edition または Windows Server 2003，Standard Edition を使用してください。
- Windows Server 2003 R2 **の場合：**
  Windows Server 2003 R2，Enterprise Edition または Windows Server 2003 R2，Standard Edition を使用してください。
- Windows Server 2003 R2 x64 Edition **の場合：**
  Windows Server 2003 R2，Enterprise x64 Edition，または Windows Server 2003 R2，Standard x64 Edition を使用してください。
- Windows Server 2008 x86 **の場合：**
  Windows Server 2008 Enterprise x86 Edition，または Windows Server 2008 Standard x86 Edition を使用してください。
- Windows Server 2008 R2 **の場合：**
  Windows Server 2008 R2 Enterprise，または Windows Server 2008 R2 Standard を使用してください。
- Windows Server 2012 **の場合：**
  Windows Server 2012 Standard，または Windows Server 2012 Datacenter を使用してください。
- Windows XP **の場合：**
  Windows XP Professional を使用してください。
- Windows Vista **の場合：**
  Windows Vista Ultimate，Windows Vista Business，または Windows Vista Enterprise を使用してください。
- Windows 7 **の場合：**
  Windows 7 Professional x86，Windows 7 Enterprise x86，Windows 7 Ultimate x86，Windows 7 Professional x64，Windows 7 Enterprise x64，または Windows 7 Ultimate x64 を使用してください。
- Windows 8 **の場合：**
  Windows 8 Pro，または Windows 8 Enterprise を使用してください。

● WWW **サーバ**

DocumentBroker クライアントシステムでは，次に示す WWW サーバを使用します。

- UNIX **の場合：**
  iPlanet Web Server Enterprise Edition，または Hitachi Web Server を使用してください。
- Windows **の場合：**
  Hitachi Web Server または Internet Information Services を使用してください。

● **ユーザアプリケーションプログラム**

ユーザが，Java クラスライブラリを使用して開発するアプリケーションプログラムです。ユーザアプ

リケーションプログラムの開発および実行に必要な Java の環境については，「1.3.3　ユーザアプリケーションプログラムの開発および実行に必要な Java の環境」を参照してください。

● DocumentBroker Development Kit または DocumentBroker Runtime
Java 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および実行するために使用するプログラムです。

● TPBroker
DocumentBroker Development Kit または DocumentBroker Runtime の ORB として使用するプログラムです。

1.3.3　ユーザアプリケーションプログラムの開発および実行に必要な Java の環境

Java クラスライブラリでユーザアプリケーションプログラムを開発する場合およびユーザアプリケーションプログラムを実行する場合に必要な Java の環境は，Cosminexus を使用して構築してください。

Cosminexus は，Java 2 Platform, Enterprise Edition に準拠した Java の開発環境および実行環境を提供するアプリケーションサーバであり，Java クラスライブラリで必要な Java の環境を提供しています。このほかに，Enterprise JavaBeans などの各種 API も提供しています。さらに Cosminexus は，DocumentBroker クライアントシステムの WWW サーバとして使用できる Hitachi Web Server（UNIX の場合）や，ビジュアルな環境でのユーザアプリケーションプログラムの開発を支援するツールである JBuilder なども提供しています。JBuilder からウィザードを起動することによって，JSP や JavaBeans を生成できます。このため，Java クラスライブラリの API を使用した Java コンポーネントを開発できます。また，Cosminexus が提供する JSP/Servlet エンジン上で Java クラスライブラリの API を使用した Java コンポーネントを実行できます。

なお，Windows Server 2008，Windows Server 2008 R2，および Windows Server 2012 を使用している場合，DocumentBroker のクライアントアプリケーションは管理者特権で実行する必要があります。

Windows Server 2003 R2 x64 Edition，Windows Server 2008，Windows Server 2008 R2，および Windows Server 2012 を使用している場合，ユーザアプリケーションプログラムの開発はできません。

1.3.4　連携できるプログラム

Java クラスライブラリを使用して連携できるプログラムを次に示します。必要に応じて使用してください。

(1) ユーザ認証やアクセス制御情報の取得に必要なプログラム

● **ディレクトリサービス**
アクセス制御機能を使用する場合，LDAP 対応のディレクトリサービスを使用したユーザ管理システムと連携することをお勧めします。LDAP 対応のディレクトリサービスとして使用できる製品については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

ログイン時のユーザ認証に必要な情報，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 Server を介してユーザ管理システムにアクセスします。DocumentBroker Server は，LDAP 対応のディレクトリサービスや UNIX のパスワードファイルなどを使用したユーザ認証機能を提供しています。DocumentBroker Server が LDAP 対応のディレクトリサービスを使用したユーザ管理システムと連携している場合，アクセス制御機能に必要な情報は，LDAP 対応のディレクトリサービスで管理

されている情報を基に生成されます。

なお，ユーザアプリケーションプログラムを開発する際に，JNDI を使用して LDAP 対応のディレクトリサービスと連携すれば，LDAP 対応のディレクトリサービスで管理されている情報を直接取得できます。

アクセス制御については，「2.12　アクセス制御」または「3.10　アクセス制御モデル」を参照してください。

(2) XML 文書を管理する場合に必要なプログラム

AIX の場合，XML 文書を管理する機能は使用できないため，ここで説明するプログラムは不要です。

● HiRDB Adapter for XML
XML 文書を解析して，XML のタグを RDB のテーブルやカラムに対応付け，XML 文書に含まれる文字列を適切なデータ型に変換して RDB に登録する XML-RDB マッピング機能を持つプログラムです。XML プロパティマッピング機能および XML インデクスデータ作成機能を使用して XML 形式の文書を管理する場合は，このプログラムを使用します。

● Preprocessing Library for Text Search
XML 文書から構造を指定した全文検索に必要なインデクスデータを生成する機能を持つプログラムです。XML インデクスデータ作成機能を使用する場合は，このプログラムを使用します。XML インデクスデータ作成機能によって作成されたインデクスデータで全文検索インデクスを登録した XML 文書は，XML の構造（タグ間の文字列や属性）をキーにした全文検索の対象にできます。

XML 文書の管理については，「2.7　XML 文書」および「3.4　XML 文書の管理モデル」を参照してください。

(3) レンディション変換要求機能を使用する場合に必要なプログラム

● DocumentBroker Rendering Option
マルチレンディション文書のコンテンツの登録・更新状態によって，DocumentBroker から出された要求に従い，レンディション変換を実行する機能を持つプログラムです。レンディション変換とは，登録済みのマスタレンディションのコンテンツを基に，ほかの形式のファイルを作成し，サブレンディションのコンテンツとして登録することです。マルチレンディション文書を管理する場合は，このプログラムと連携すると，マスタレンディションとサブレンディションの内容の同期が取りやすくなります。

マルチレンディションおよびレンディション変換要求機能については，「2.5　レンディション管理」および「3.2　レンディション管理モデル」を参照してください。

1.3.5　使用できる WWW ブラウザ

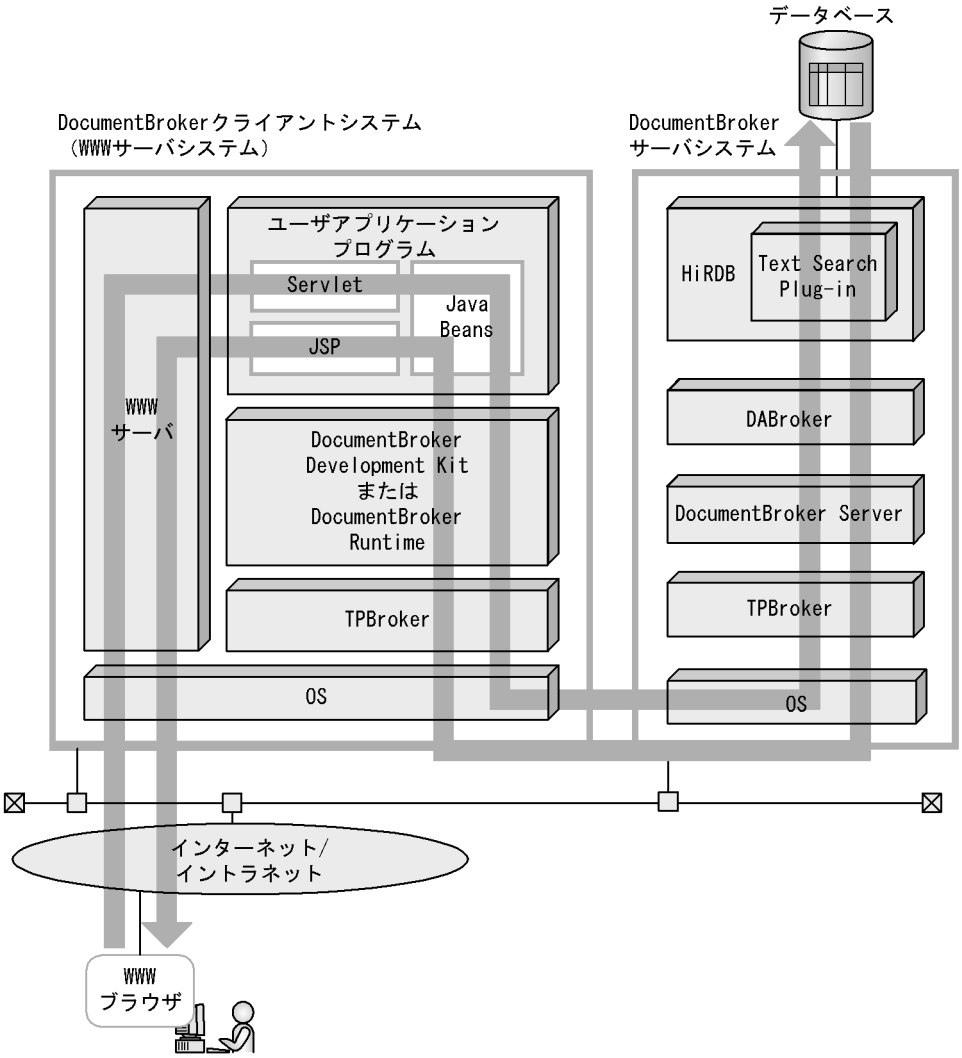
Java クラスライブラリを使用して開発したユーザアプリケーションプログラムを実行できる WWW ブラウザを次に示します。

● Internet Explorer

1.4 Java クラスライブラリを使用する場合の処理の流れ

この節では，DocumentBroker クライアントシステムを DocumentBroker Development Kit および DocumentBroker Runtime の Java クラスライブラリを使用して開発する場合の処理の流れについて説明します。処理の流れの例を次の図に示します。

図 1-3 Java クラスライブラリを使用する場合の処理の流れの例



この例では，JavaBeans，Servlet，および JSP を使用してユーザアプリケーションプログラムを作成しています。このユーザアプリケーションプログラムでは，Servlet と JSP の処理を，JavaBeans を経由して実行しています。

例えば，検索を実行する画面が WWW ブラウザ上に表示されているとします。ユーザは，WWW ブラウザ上に表示された画面に検索条件を入力し，「検索実行」ボタンを押して，検索を実行します。ユーザからの処理要求は Servlet が受け付け，Servlet はその処理を JavaBeans に渡します。JavaBeans は DocumentBroker Server 経由でデータベースにアクセスして検索処理を実行し，その処理結果を JSP に渡します。JSP は JavaBeans から受け取った処理結果を基に WWW ページを生成して，WWW ブラウザ

上に検索結果の画面を表示します。

2 文書管理で使用する概念

この章では，文書管理で使用する概念と各概念の詳細について説明します。

2.1　Java クラスライブラリの操作で使用する概念

この節では，DocumentBroker Development Kit および DocumentBroker Runtime で提供する Java 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ために理解しておく必要がある，オブジェクト，クラス，プロパティおよびメソッドについて説明します。

2.1.1　オブジェクト

Java クラスライブラリでは，次の 2 種類のオブジェクトを区別して理解しておく必要があります。

- Java オブジェクト
- 文書空間オブジェクト

(1) Java オブジェクト

Java 言語上のオブジェクトです。Java オブジェクトは，インターフェースを使用するための実体として，メモリ空間上に作成される概念的なオブジェクトです。

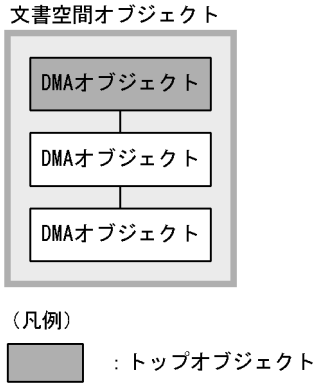
例えば，Java オブジェクトには，セッションを管理するセッションオブジェクト，文書空間へのアクセスを管理する文書空間アクセスオブジェクト，文書空間オブジェクトの代理オブジェクトである Proxy オブジェクトなどがあります。

Java オブジェクトの詳細については，「6.　Java クラスライブラリの機能」を参照してください。

(2) 文書空間オブジェクト

データベース上に実体が存在するオブジェクトです。文書空間オブジェクトは，DMA オブジェクトによって構成されます。文書空間オブジェクトは，文書管理の用途に応じて，複数の DMA オブジェクトを，一つのまとまったオブジェクトとして扱います。文書空間オブジェクトの構成を次の図に示します。

図 2-1　文書空間オブジェクトの構成



DMA オブジェクト

DMA が提唱するオブジェクトモデルに基づいたオブジェクトです。DocumentBroker では，DMA オブジェクトを組み合わせて使用することによって文書を管理しています。DMA オブジェクトは，文書空間オブジェクトの構成要素となります。DMA オブジェクトには，文書を表すオブジェクト，文書をまとめるフォルダの役目をするオブジェクト，文書のバージョンを管理するためのオブジェクトなどがあります。一つの DMA オブジェクトは，一つの DMA クラスを基に作成されます。
なお，このマニュアルでは，文書空間オブジェクトを構成する DMA オブジェクトのうち，代表的な

DMA オブジェクトを**トップオブジェクト**といいます。また，トップオブジェクトのクラスを**トップオブジェクトクラス**といいます。

2.1.2　クラス

Java クラスライブラリでは，次の 2 種類のクラスを区別して理解しておく必要があります。

- Java クラス
- 文書空間オブジェクトを操作するためのクラス

(1) Java クラス

Java 言語上のクラスです。また，Java クラスライブラリが提供する Java のクラスおよびインターフェースの総称です。Java クラスは，メソッドを実装しています。

Java クラスの詳細については，「6.　Java クラスライブラリの機能」を参照してください。

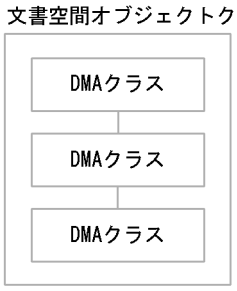
(2) 文書空間オブジェクトを操作するためのクラス

データベース上のクラスです。文書空間オブジェクトに対応するクラスです。文書空間オブジェクトを操作するためのクラスには次の 2 種類があります。

- 文書空間オブジェクトクラス
- DMA クラス

文書空間オブジェクトクラスは，DMA クラスによって構成されます。文書空間オブジェクトクラスの構成を次の図に示します。

図 2-2　文書空間オブジェクトクラスの構成



文書空間オブジェクトクラス

一つまたは複数の DMA クラスから構成される概念的なクラスです。

文書空間オブジェクトクラスは，DMA オブジェクトをまとめて操作するための機能を持ったクラスです。文書管理を実現する機能ごとに，必要な DMA オブジェクトを操作する機能を持ったクラスが定義されています。文書空間オブジェクトクラスには，バージョンなし文書クラス，バージョン付き文書クラス，バージョンなしフォルダクラスなどがあります。文書やフォルダを作成して操作するためのクラスは，文書やフォルダに必要な複数の DMA クラスを構成要素として持っています。

DMA クラス

文書空間オブジェクトクラスの構成要素です。DMA オブジェクトに対応するクラスです。DMA クラスは，データベースの表に対応します。DMA クラスには，文書を表すオブジェクトのクラス，フォルダを表すオブジェクトのクラスなどがあります。

DocumentBroker では，DMA が規定する文書管理に必要なオブジェクトを作成するための DMA クラスに加えて，拡張した機能を持つ DMA クラスを独自に提供しています。例えば，構成管理機能を

持つ DMA オブジェクトは，DocumentBroker が拡張した DMA クラスを基に作成されます。

文書空間オブジェクトクラスと，その構成要素である DMA クラスの対応については，「2.2.2　文書空間オブジェクトクラスと DMA クラスの対応」を参照してください。

なお，文書空間オブジェクトクラスには，メソッドは定義されていません。文書空間オブジェクトは，Java クラスのメソッドを通して操作します。

2.1.3　プロパティ

オブジェクトの属性のことを**プロパティ**といいます。プロパティには次の 2 種類があります。

Java オブジェクトのプロパティ

Java オブジェクトが持つ属性です。Java クラスに定義されています。

文書空間オブジェクトのプロパティ

文書空間オブジェクトが持つ属性です。文書空間オブジェクトクラスに定義されています。また，文書空間オブジェクトのプロパティとして，ユーザが任意のプロパティを追加定義できます。

2.1.4　メソッド

Java クラスライブラリのメソッドは，クラスごとまたはインターフェースごとに定義されています。メソッドによってプロパティの値を設定・取得したり，文書空間オブジェクトを操作したりできます。

操作対象の Java オブジェクトまたは文書空間オブジェクトの状態を変化させるメソッドを**更新系メソッド**といいます。また，操作対象の Java オブジェクトまたは文書空間オブジェクトの状態を変化させないメソッドを**参照系メソッド**といいます。

なお，このマニュアルでは，メソッドの名前を「インターフェースの名前 # メソッドの名前」と表記しています。例えば，「DbjObj#readProperties メソッド」は，「DbjObj インターフェース」の「readProperties メソッド」になります。

2.2　文書空間オブジェクト

この節では，Java クラスライブラリでの文書空間オブジェクトと DMA オブジェクトの関係，オブジェクトとデータベースの関係，文書空間オブジェクトの種類および文書空間オブジェクトクラスと DMA クラスの対応について説明します。

2.2.1　文書空間オブジェクトとは

文書空間オブジェクトは，複数の DMA オブジェクトを包含した形で構成されるオブジェクトです。Java クラスライブラリでは，文書空間オブジェクトを操作することによって，複数の DMA オブジェクトをまとめて操作します。

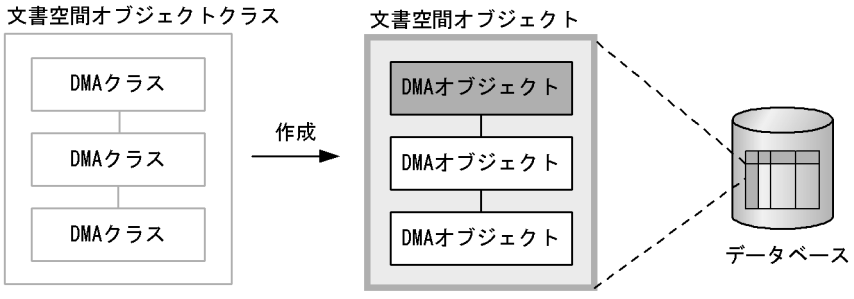
(1) 文書空間オブジェクトと DMA オブジェクトの関係

DMA オブジェクトには，文書を表すオブジェクト，文書の表現形式を管理するオブジェクト，文書の実体であるコンテンツ（文書ファイル）に相当するオブジェクトなどがあります。一つの文書を操作するためには，これらの DMA オブジェクトをすべて操作する必要があります。DMA オブジェクトは，プロパティの値を設定，変更することで操作します。文書管理の操作を実現するためには，文書を構成する複数の DMA オブジェクトに対して，それぞれ適切なプロパティを設定する必要があります。

Java クラスライブラリでは，文書管理に使用する複数の DMA オブジェクトを，文書やフォルダなどのまとまった一つの文書空間オブジェクトとして扱います。文書空間オブジェクトに対応するクラスが，文書空間オブジェクトクラスです。文書空間オブジェクトクラスは，文書管理を実現する機能ごとに，一つまたは複数の DMA クラスから構成されています。文書空間オブジェクトクラスを基に文書空間オブジェクトを作成することによって，その機能を実現するために必要な DMA オブジェクトが作成されます。また，Java クラスライブラリでは，文書管理に必要な操作をクラスごとまたはインターフェースごとにメソッドとして提供しています。Java クラスライブラリのメソッドを実行することによって，その操作に必要なプロパティが DMA オブジェクトに設定されます。したがって，文書を表す文書空間オブジェクトを操作することで，文書を表すオブジェクト，文書の表現形式を管理するオブジェクト，文書の実体であるコンテンツに相当するオブジェクトなどの複数の DMA オブジェクトを包含した，一つの文書が操作できます。

文書空間オブジェクトと DMA オブジェクトの関係を次の図に示します。

図 2-3　文書空間オブジェクトと DMA オブジェクトの関係



(2) オブジェクトとデータベースの関係

データベースは DMA のオブジェクトモデルと対応して構成されています。DMA クラスは，データベースの表に対応します。文書空間オブジェクトに対してメソッドを実行すると，複数の DMA オブジェクトへのメソッドとして配置され，該当するデータベースの表が操作されます。

オブジェクトを生成する DMA クラスは，通常，それぞれが HiRDB の一つの表と対応します。DMA オブジェクトは各表の行（レコード）に相当し，各 DMA クラスが持つプロパティのうち，永続プロパティ（データベースに値が格納されているプロパティ）は各表の列（カラム）に相当します。また，DMA オブジェクト同士の関連は，各オブジェクトが属性として保持する参照先の OIID によって表されます。

オブジェクトとデータベースの関係の詳細については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

(3) 文書空間オブジェクトの種類

文書空間オブジェクトの種類を次に示します。

● バージョンなし文書
一つのバージョンだけを持つ文書です。

● バージョン付き文書
複数のバージョンを持つ文書です。

● バージョンなしフォルダ
一つのバージョンだけを持ち，複数の文書をまとめたり，分類したりして構成管理するフォルダです。

● バージョン付きフォルダ
複数のバージョンを持ち，複数の文書をまとめたり，分類したりして構成管理するフォルダです。

● 独立データ
独立したデータを扱う文書空間オブジェクトです。

● パブリック ACL
複数の文書空間オブジェクトで共有するアクセス制御情報を保持する文書空間オブジェクトです。

● リンクオブジェクト
複数の文書空間オブジェクト間をリンク付ける文書空間オブジェクトです。なお，リンクオブジェクトは，文書またはフォルダに従属するオブジェクトであり，単独では作成できません。

なお，このマニュアルでは，それぞれの文書空間オブジェクトをまとめて，次のような総称で表記することがあります。

文書
「バージョンなし文書」および「バージョン付き文書」の総称として使用します。また，「文書オブジェクト」と表記することもあります。

フォルダ
「バージョンなしフォルダ」および「バージョン付きフォルダ」の総称として使用します。また，「フォルダオブジェクト」と表記することもあります。

リファレンスファイル文書
リファレンスファイル管理機能を使用して管理している文書のことです。
この文書では，任意のディレクトリに登録されているコンテンツを管理しています。データベースでは，そのコンテンツロケーションを管理しています。

バージョンなしオブジェクト
「バージョンなし文書」および「バージョンなしフォルダ」の総称として使用します。

バージョン付きオブジェクト
「バージョン付き文書」および「バージョン付きフォルダ」の総称として使用します。なお，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オブジェクトを**バージョニングオブジェクト**，個々のバージョンを表すオブジェクトを**バージョンオブジェクト**といいます。

2.2.2 文書空間オブジェクトクラスと DMA クラスの対応

文書空間オブジェクトクラスと，その構成要素である DMA クラスのトップオブジェクトクラスの対応を次の表に示します。トップオブジェクトクラスとして，次の表に示す DMA クラスまたはそのサブクラスを指定してください。

表 2-1 文書空間オブジェクトクラスとトップオブジェクトクラスの対応

文書空間オブジェクトクラス（文書空間オブジェクト）	トップオブジェクトクラス（トップオブジェクト）
バージョンなし文書クラス （バージョンなし文書）	dmaClass_DocVersion クラス （DocVersion オブジェクト）
バージョン付き文書クラス （バージョン付き文書）	dmaClass_ConfigurationHistory クラス （ConfigurationHistory オブジェクト）
バージョン付き文書クラス （バージョン付き文書のバージョン※1）	edmClass_VersionTracedDocVersion クラス※2 （VersionTracedDocVersion オブジェクト）
バージョンなしフォルダクラス （バージョンなしフォルダ）	edmClass_ContainerVersion クラス※3 （ContainerVersion オブジェクト）
バージョン付きフォルダクラス （バージョン付きフォルダ）	dmaClass_ConfigurationHistory クラス （ConfigurationHistory オブジェクト）
バージョン付きフォルダクラス （バージョン付きフォルダのバージョン※4）	edmClass_ContainerVersion クラス （ContainerVersion オブジェクト）
独立データクラス （独立データ）	edmClass_IndependentPersistence クラス （IndependentPersistence オブジェクト）
パブリック ACL クラス （パブリック ACL）	edmClass_PublicACL クラス※5 （PublicACL オブジェクト）
リンクオブジェクトクラス （リンクオブジェクト）	• dmaClass_DirectContainmentRelationship クラス （DirectContainmentRelationship オブジェクト） • dmaClass_ReferentialContainmentRelationship クラス （ReferentialContainmentRelationship オブジェクト） • edmClass_VersionTraceableContainmentRelationship クラス （VersionTraceableContainmentRelationship オブジェクト） • edmClass_Relationship クラス※5 （Relationship オブジェクト）

注※1　バージョン付き文書の 1 バージョンとして管理するバージョンなし文書です。
注※2　トップオブジェクトクラスとして，dmaClass_DocVersion クラス，または edmClass_VersionTracedDocVersion クラス以外の dmaClass_DocVersion クラスのサブクラスを指定した場合，そのバージョン付き文書は構成管理の対象になりません。
注※3　トップオブジェクトクラスとして，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指定した場合，そのバージョンなしフォルダは構成管理型リンクを使用できません。
注※4　バージョン付きフォルダの 1 バージョンとして管理するバージョンなしフォルダです。
注※5　edmClass_PublicACL クラスと edmClass_Relationship クラスは，サブクラスを作成できません。

なお，このマニュアルでは，それぞれの文書空間オブジェクトクラスをまとめて，次のような総称で表記することがあります。

バージョンなしオブジェクトクラス

「バージョンなし文書クラス」および「バージョンなしフォルダクラス」の総称として使用します。

バージョン付きオブジェクトクラス
「バージョン付き文書クラス」および「バージョン付きフォルダクラス」の総称として使用します。

文書オブジェクトクラス
「バージョンなし文書クラス」および「バージョン付き文書クラス」の総称として使用します。

フォルダオブジェクトクラス
「バージョンなしフォルダクラス」および「バージョン付きフォルダクラス」の総称として使用します。

2.3 文書

この節では，Java クラスライブラリでの文書の概念について説明します。

Java クラスライブラリでは，文書を表すオブジェクト，文書の表現形式を管理するオブジェクト，文書の実体であるコンテンツ（Word やテキストエディタなどのアプリケーションプログラムで作成された文書ファイル）に相当するオブジェクトなどの複数の DMA オブジェクトを，まとまった一つの文書として扱います。

Java クラスライブラリでは，次の 2 種類の文書を表す文書空間オブジェクトを扱います。

- バージョンなし文書
- バージョン付き文書

2.3.1 バージョンなし文書

バージョンなし文書は，一つのバージョンだけを持つ文書です。版管理をして更新履歴などを残す必要がない（バージョンを管理する必要がない）文書を管理する場合に使用します。

2.3.2 バージョン付き文書

バージョン付き文書は，複数のバージョンを持つ文書です。バージョンを付けて管理することで，過去のある時点での文書を参照したり，その文書を流用して新たに文書を作成するのに使用したりできます。

Java クラスライブラリでは，複数のバージョンを持つ文書を一つの文書（バージョン付き文書）として管理できます。

2.4 バージョン管理

この節では，Java クラスライブラリでのバージョン管理の概念について説明します。

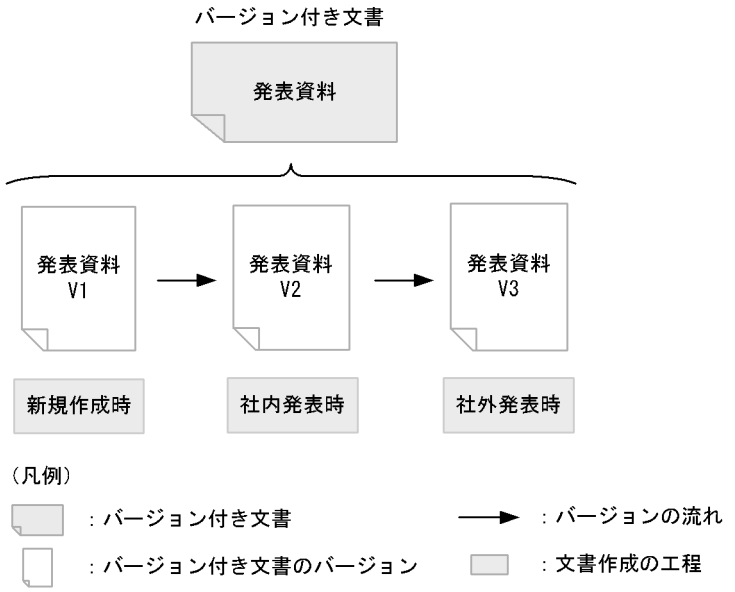
Java クラスライブラリでは，次の 2 種類の文書空間オブジェクトでバージョンを管理できます。

- バージョン付き文書
- バージョン付きフォルダ

バージョン付きオブジェクトでは，オブジェクトにバージョンを付けて管理します。新しく文書（またはフォルダ）を作成した時点からバージョンを付け始め，更新のたびにバージョンを追加して，一連のバージョンを管理します。なお，各バージョンから枝分かれしたバージョンを管理することはできません。

バージョン付きオブジェクトの管理例を次の図に示します。

図 2-4 バージョン付きオブジェクトの管理例



この例では，バージョン付き文書「発表資料」を作成して，社内発表時および社外発表時にバージョンを追加して，管理しています。

バージョン付きオブジェクトを更新する場合は，古いバージョンを破棄して新しいバージョンと置き換えるのではなく，古いバージョンを残して新しいバージョンを追加（バージョンアップ）して登録します。バージョンを追加する場合は，チェックアウトとチェックインという操作が必要になります。

チェックアウトは，新しいバージョンの追加を予約するために，最新バージョンの次に仮のバージョンを追加する操作です。また，チェックインは，チェックアウトで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操作です。

なお，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空間オブジェクトを**バージョニングオブジェクト**，個々のバージョンを表す文書空間オブジェクトを**バージョンオブジェクト**といいます。Java クラスライブラリでは，バージョン付きオブジェクトを操作する場合，バージョン全体を対象とするときはバージョニングオブジェクト，個々のバージョンを対象とするときはバージョンオブジェクトを操作します。

2.5　レンディション管理

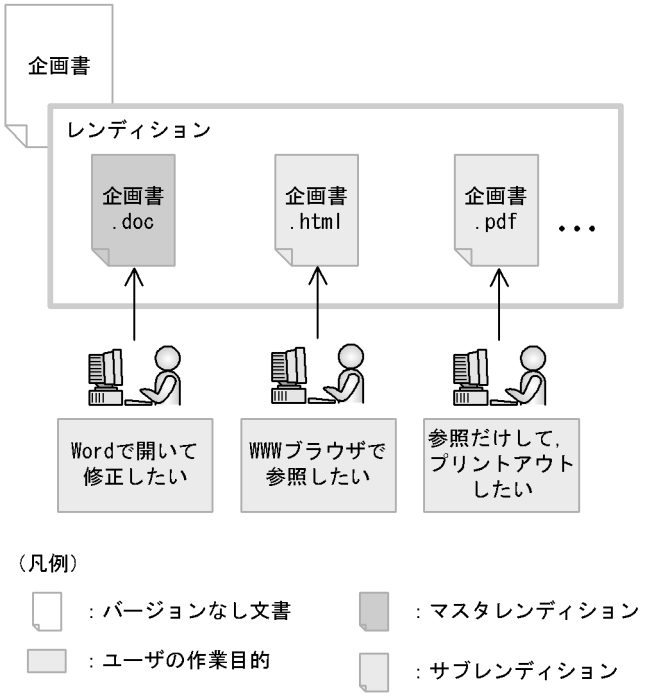
この節では，Java クラスライブラリでのレンディション管理の概念について説明します。

文書の実体であるコンテンツ（Word やテキストエディタなどのアプリケーションプログラムで作成された文書ファイル）と，その表現形式を表すレンディションタイプ（MIME **形式**）などの情報をあわせて**レンディション**といいます。

Java クラスライブラリでは，バージョンなし文書またはバージョン付き文書に 1 個または複数のレンディションを登録して管理します。複数のレンディションを登録した文書を特に**マルチレンディション文書**といいます。マルチレンディションとは，一つの文書に対して同じ内容を表す複数の異なる形式のレンディションを登録する機能です。なお，レンディションのうち，主要なレンディションを**マスタレンディション**，マスタレンディション以外のレンディションを**サブレンディション**といって区別します。

例えば，同じ内容を表す Word 形式のファイル，HTML 形式のファイルおよび PDF 形式のファイルをマルチレンディション文書のコンテンツとして登録して管理すると，ユーザの目的やユーザが使用するマシンの環境などの利用形態に応じて，Word 形式のファイルをダウンロードしたり，HTML 形式のファイルをダウンロードしたり，PDF 形式のファイルをダウンロードしたりできます。マルチレンディションによる文書の管理例を次の図に示します。

図 2-5　マルチレンディションによる文書の管理例



Java クラスライブラリでは，バージョンなし文書またはバージョン付き文書をマルチレンディション文書として管理して，一つの文書に対して，1 個のマスタレンディションおよび最大 9 個のサブレンディションを登録できます。

また，DocumentBroker Rendering Option と連携すると，登録済みのマスタレンディションのコンテンツを基に，別のレンディションタイプのファイルを自動作成してサブレンディションのコンテンツとして登録することもできます。

2.6　リファレンスファイル管理

この節では，Java クラスライブラリでのリファレンスファイル管理の概念について説明します。

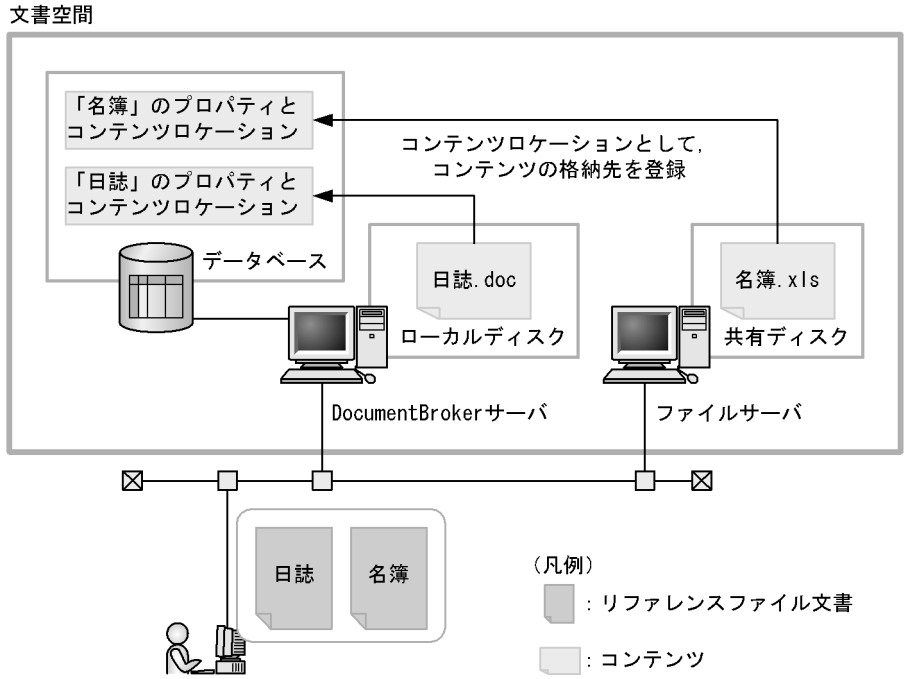
リファレンスファイル管理機能は，コンテンツを任意のディレクトリに格納し，文書のプロパティとコンテンツロケーションだけをデータベースで管理する機能です。データベースとコンテンツを切り離すことによって，ユーザによるコンテンツ格納先の選択やディスク移行時のコンテンツ格納ディレクトリの変更など，柔軟にコンテンツ格納先の管理ができます。また，Windows の場合は UNC 形式のパスでコンテンツを操作することもできます。

リファレンスファイル管理機能を使用して管理している文書を，**リファレンスファイル文書**といいます。

DocumentBroker では，バージョンなし文書およびバージョン付き文書をリファレンスファイル文書として管理できます。

リファレンスファイル文書の管理例を次の図に示します。

図 2-6　リファレンスファイル文書の管理例



2.7 XML 文書

この節では，Java クラスライブラリでの XML 文書の概念と XML 文書を管理するための機能について説明します。

XML は，W3C によって標準化が進められている構造化文書の定義言語です。XML では，ユーザが設定した独自のタグによって，論理構造を持つ文書を記述できます。このため，WWW 上のデータ交換フォーマットとして，多くの業務で適用されています。Java クラスライブラリでは，XML 形式で記述されたファイルを，バージョンなし文書およびバージョン付き文書のコンテンツとして登録して管理できます。

なお，このマニュアルでは，XML 形式で記述されたファイルを XML **ファイル**といいます。また，XML ファイルをコンテンツに持つバージョンなし文書およびバージョン付き文書を特に XML **文書**といいます。

XML 文書から参照される画像ファイルなどの複数のファイルは，バージョンなしフォルダまたはバージョン付きフォルダでまとめて管理したり，文書間リンクで関連づけたりすることによって管理できます。文書間リンクについては，「2.8　リンク」を参照してください。

Java クラスライブラリでは，次の二つの機能を使用して XML 文書を管理できます。

- XML プロパティマッピング機能
- XML インデクスデータ作成機能

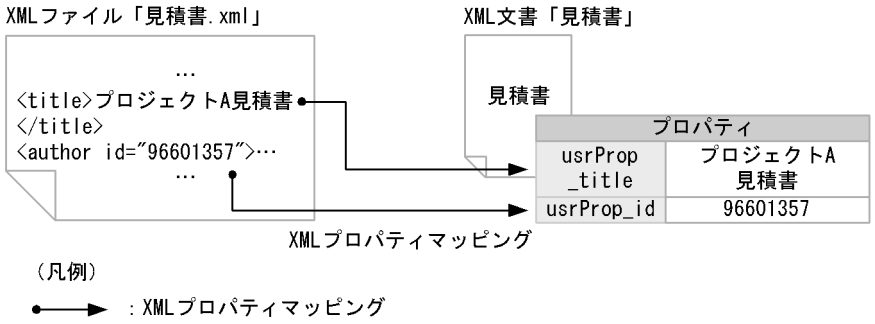
XML **プロパティマッピング機能**は，XML ファイル中のタグ間の文字列やタグの属性値を，XML 文書のプロパティに割り当てて管理する機能です。

例えば，次のような内容の XML ファイルがあります。このファイルは，プロジェクト A の見積書です。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<doc>
<title>プロジェクトA見積書</title>
<date>2001.1.10</date>
<author id="96601357">
  <name>日立太郎</name>
  <organization>営業部</organization>
</author>
<content>

</content>
</doc>
```

このとき，XML ファイルの <title> タグ間の情報「プロジェクト A 見積書」や，<author> タグの属性値「96601357」などを，XML 文書のプロパティとしてマッピングできます。XML プロパティマッピングの例を次の図に示します。

図 2-7 XML プロパティマッピングの例



XML プロパティマッピング機能を使用する場合は，前提プログラムとして，HiRDB Adapter for XML が必要です。

XML インデクスデータ作成機能は，XML 文書の構文解析を実行して，インデクスデータを作成し，タグ情報などを削除したプレーンテキスト形式の全文検索インデクスまたは構造指定検索用の全文検索インデクスを登録する機能です。

XML インデクスデータ作成機能を使用する場合は，前提プログラムとして HiRDB Adapter for XML および Preprocessing Library for Text Search が必要です。

2.8　リンク

この節では，Java クラスライブラリでのリンクの概念について説明します。

文書またはフォルダは，DocumentBroker に登録されているほかの複数の文書またはフォルダと関連づけて管理できます。Java クラスライブラリでは，このオブジェクト間のリンクをオブジェクト（**リンクオブジェクト**）として扱います。

Java クラスライブラリでは，このリンクオブジェクトを使用して，複数の文書空間オブジェクト間をリンク付けたり，文書空間オブジェクトを検索したりできます。

リンクには次の 4 種類があります。

- **直接型リンク**
- **参照型リンク**
- **構成管理型リンク**
- **文書間リンク**

この節では，文書間リンクについて説明します。直接型リンク，参照型リンクおよび構成管理型リンクについては，「2.9　フォルダ」を参照してください。

2.8.1　文書間リンク

文書間リンクとは，文書と文書を関連づけて管理する機能です。

文書間リンクは，次のような場合に使用できます。

● 参考文献のある論文などの文書を，参考文献とともに管理したい場合

● 別文書として登録しているテキストと図データを関連がわかるように管理したい場合

Java クラスライブラリでは，二つの文書を文書間リンクという概念で関連づけます。このとき，文書間リンクを設定するリンク元の文書を**リンク元文書**，文書間リンクが設定されるリンク先の文書を**リンク先文書**といいます。

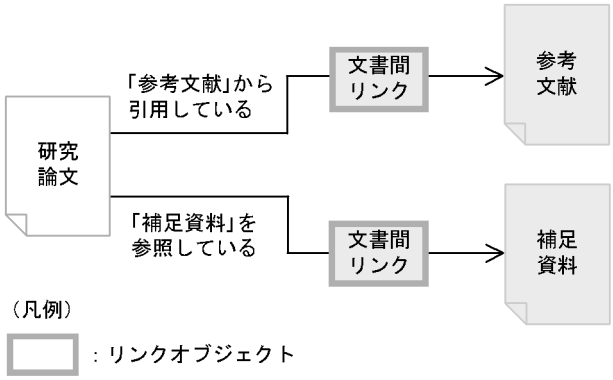
Java クラスライブラリでは，次のどちらかの文書空間オブジェクト間に文書間リンクを設定できます。

- バージョンなし文書
- バージョン付き文書

一つのリンク元文書は，複数のリンク先文書と関連づけることができます。また，一つのリンク先文書も，複数のリンク元文書から関連づけられることができます。したがって，m:n（m，n は 1 以上の整数）のリンクを設定することで，m 個と n 個の文書が文書間リンクによって関連づけられます。

文書間リンクによる文書の管理例を次の図に示します。

図 2-8 文書間リンクによる文書の管理例



この例では，文書「研究論文」は，文書「参考文献」から記述を引用し，文書「補足資料」を本文中で参照しています。このため，「研究論文」と「参考文献」，「研究論文」と「補足資料」を，それぞれ文書間リンクで関連づけています。このように文書間を関連づけておくことによって，必要な資料をまとめて管理できます。また，関連づけている「参考文献」が更新された場合や，「補足資料」が削除された場合には，「研究論文」から確認できます。

2.9 フォルダ

この節では，Java クラスライブラリでのフォルダの概念について説明します。

Java クラスライブラリでは，複数の文書をまとめたり，関連づけて分類したりするためにフォルダという概念を使用します。フォルダを使用すると，複数の文書を目的に応じて一つにまとめて管理したり，一つの文書を複数の分類に関連づけたりできます。また，フォルダを別のフォルダに関連づけることもできます。

Java クラスライブラリでは，次の 2 種類のフォルダを表す文書空間オブジェクトを扱います。

- バージョンなしフォルダ
- バージョン付きフォルダ

2.9.1 バージョンなしフォルダ

バージョンなしフォルダは，一つのバージョンだけを持つフォルダです。フォルダ自身のバージョンを管理する必要がない場合に使用します。

バージョンなしフォルダには次の三つの機能があります。

● 直接型リンク
一つのフォルダに複数の文書空間オブジェクト（文書またはフォルダ）をリンク付けて，ディレクトリのイメージで文書をまとめて管理する機能です。

● 参照型リンク
一つの文書空間オブジェクト（文書またはフォルダ）を複数のフォルダにリンク付けて，文書を分類管理する機能です。

● 構成管理型リンク
下位にリンク付けているバージョン付きオブジェクトのバージョンをトレースしてリンク付ける機能です。

2.9.2 バージョン付きフォルダ

バージョン付きフォルダは，複数のバージョンを持つフォルダです。バージョンなしフォルダの機能に加えて，フォルダ自身のバージョンを管理する機能を持ちます。フォルダ自身のバージョンを管理することによって，フォルダ自身のバージョンごとに，下位のバージョン付きオブジェクトのバージョンを管理できます。

バージョン付きフォルダには次の四つの機能があります。

● バージョン管理
バージョン付きフォルダのバージョンと，下位のバージョン付きオブジェクトを，リンク付けたままバージョンアップできる機能です。複数のバージョンを持つフォルダとして機能します。

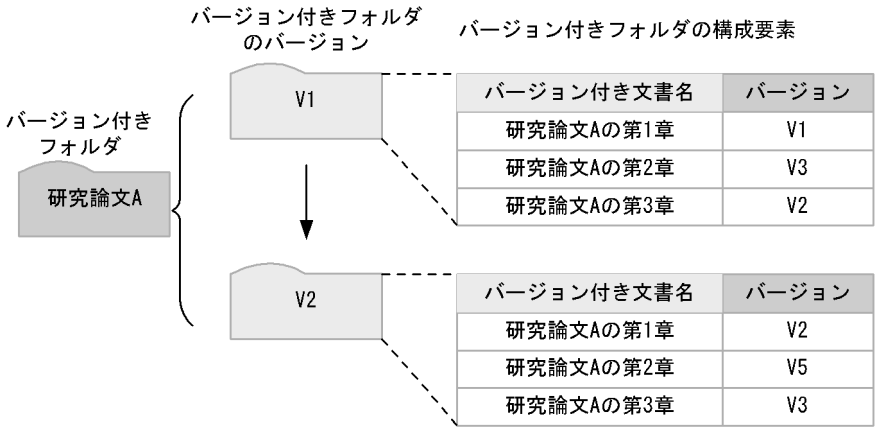
● 構成管理型リンク

● 直接型リンク

● 参照型リンク

バージョン付きフォルダによる管理の考え方
バージョン付きフォルダが持つバージョン管理および構成管理型リンクの機能を組み合わせると，次のような管理が考えられます。バージョン付きフォルダの管理例を次の図に示します。

図 2-9 バージョン付きフォルダの管理例



この例では，一つの「研究論文 A」という文書を複数の章に分割して，各章をバージョン付き文書として管理しています。そして，これらの複数の章を表すバージョン付き文書を「研究論文 A」というバージョン付きフォルダにまとめてリンク付けています。構成管理型リンクの機能を使用すると，バージョン付きオブジェクトをバージョン単位でリンク付けできるため，バージョン付きフォルダのバージョンごとに，構成要素であるバージョン付きオブジェクトのバージョン構成を管理できます。例えば，一つの原稿を複数の担当者で分担して編集し，原稿が完成した時点，審査を受けて指摘を反映した時点など，作業の区切りでバージョンを管理する場合には，バージョン付きフォルダを使用すると便利です。

2.10 独立データ

この節では，Java クラスライブラリでの独立データの概念について説明します。

独立データは，独立したデータを扱う文書空間オブジェクトです。このため，文書のようにまとめたり，フォルダに関連づけたり，バージョンを管理したり，フォルダのように上位・下位の階層構造で管理したりすることはできません。

2.11 文書空間オブジェクトのプロパティ

この節では，文書空間オブジェクトのプロパティについて説明します。

2.11.1 文書空間オブジェクトのプロパティの種類

文書空間オブジェクトのプロパティには次の 3 種類があります。

- **DMA または DocumentBroker サーバが規定したプロパティ（DMA プロパティ）**
- **Java クラスライブラリが定義したプロパティ（Java クラスライブラリ固有のプロパティ）**
- **ユーザが定義したプロパティ（ユーザ定義プロパティ）**

なお，データベースに値が格納されているプロパティを特に**永続プロパティ**といいます。

（1） DMA または DocumentBroker サーバが規定したプロパティ（DMA プロパティ）

DMA では，DMA クラスごとにプロパティを規定しています。また，DocumentBroker で拡張したクラスには，DocumentBroker 独自のプロパティも定義されています。DMA プロパティは，「dmaProp_」または「edmProp_」で始まる識別子を持つプロパティです。DMA プロパティは，文書空間オブジェクトクラスの構成要素である DMA クラスのうち，トップオブジェクトクラスに定義されています。トップオブジェクトクラスについては，「2.2.2　文書空間オブジェクトクラスと DMA クラスの対応」を参照してください。

DMA プロパティについては，「付録 A　文書空間オブジェクトのプロパティ一覧」を参照してください。

（2） Java クラスライブラリが定義したプロパティ（Java クラスライブラリ固有のプロパティ）

Java クラスライブラリ固有のプロパティとは，DMA クラスに定義されていない情報や，トップオブジェクトクラス以外の DMA クラスに定義されている情報を，文書空間オブジェクトのプロパティとして扱えるように定義したものです。

Java クラスライブラリでは，DMA クラスに定義されていない情報を，Java クラスライブラリ固有のプロパティとして扱うことができます。例えば，あるフォルダにリンク付けられている文書とフォルダの数，文書が保持するバージョンの数などは，DMA オブジェクト上には存在しない情報です。Java クラスライブラリでは，このような情報に対して「dbrProp_」で始まるプロパティ識別子を付けて，文書空間オブジェクトのプロパティとして扱えるようにしています。「バージョンなしフォルダがリンク付けている文書の数を知りたい」場合などに，このプロパティの値を取得するメソッドを実行すれば，必要な値を得ることができます。実際には，メソッドを実行した段階で，Java クラスライブラリの内部で必要な処理や演算が実行されて，値が作成され，バージョンなしフォルダのプロパティの値としてユーザに返却されます。なお，リンクオブジェクトクラスには，Java クラスライブラリ固有のプロパティはありません。

また，Java クラスライブラリでは，トップオブジェクトクラス以外の DMA クラスに定義されている情報を，Java クラスライブラリ固有のプロパティとして扱うことができます。例えば，バージョンなし文書のコンテンツのファイル名についての情報は，トップオブジェクト上でなく，バージョンなし文書を構成するほかの DMA オブジェクト上にプロパティとして存在しています。Java クラスライブラリでは，このような情報に対して「dbrProp_」で始まるプロパティ識別子を付けて，プロパティの値を取得できるようにしています。このプロパティの値を取得するメソッドを実行すれば，必要な値を得ることができます。実際には，メソッドを実行した段階で，Java クラスライブラリの内部で該当するプロパティが参照され，バージョンなし文書のプロパティとしてユーザに返却されます。

Java クラスライブラリ固有のプロパティについては，「付録 A　文書空間オブジェクトのプロパティ一覧」を参照してください。

注意事項

Java クラスライブラリ固有のプロパティは，検索条件には指定できません。アクセス制御機能を使用する場合に定義されるプロパティについては，指定方法によって，検索の対象にできるプロパティがあります。アクセス制御機能を使用する場合に定義されるプロパティについては，「3.10.5　アクセス制御モデルで使用するプロパティ」を参照してください。

(3) ユーザが定義したプロパティ（ユーザ定義プロパティ）

ユーザ定義プロパティは，ユーザが業務に応じて追加定義するプロパティです。ユーザ定義プロパティは，トップオブジェクトクラスにサブクラスを作成して追加定義できます。

ユーザ定義プロパティの追加については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

(4) 文書空間オブジェクトのクラスとプロパティの識別

Java クラスライブラリでは，文書空間オブジェクトクラスの識別に**クラスの名前**を使用します。クラスの名前とは，DocumentBroker サーバのクラス定義情報ファイルで，各クラスの dmaProp_DisplayName の値として指定されている文字列です。

また，Java クラスライブラリでは，文書空間オブジェクトのプロパティの識別に**プロパティの名前**を使用します。プロパティの名前とは，DocumentBroker サーバのクラス定義情報ファイルで，各プロパティの dmaProp_DisplayName の値として指定されている文字列です。

クラス定義情報ファイルについては，「7.6.1　クラス定義情報ファイル」を参照してください。

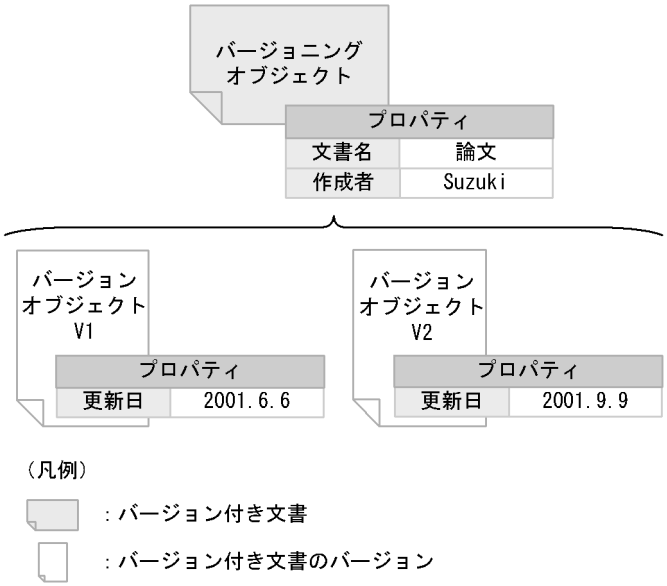
(5) バージョン付きオブジェクトのプロパティ

バージョン付きオブジェクトのプロパティは，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のプロパティとして管理します。

- **バージョニングオブジェクトのプロパティ**
  複数のバージョンに共通するプロパティです。
- **バージョンオブジェクトのプロパティ**
  特定のバージョン固有のプロパティです。

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次の図に示します。

図 2-10 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



この例では，バージョニングオブジェクトのプロパティとして文書名および作成者を，バージョンオブジェクトのプロパティとして更新日を設定しています。

2.11.2 文書空間オブジェクトのプロパティのデータ型

文書空間オブジェクトのプロパティのデータ型と DocumentBroker サーバで扱うデータ型の対応を次の表に示します。

表 2-2 文書空間オブジェクトのプロパティのデータ型と DocumentBroker サーバで扱うデータ型の対応

プロパティのデータ型（定数）	説明	DocumentBroker サーバで扱うデータ型
BOOL 型 （DbjDef.DATATYPE_BOOL）	真偽値です。次のどれかの値を取ります。 • DbjDef.DMA_TRUE（=0） • DbjDef.DMA_FALSE（=1） • DbjDef.DMA_UNKNOWN（=2）	Boolean 型
INT 型 （DbjDef.DATATYPE_INT）	数値です。	Integer32 型
VARRAY 型 （DbjDef.DATATYPE_VARRAY）	複数の値を保持する可変長配列です。	Object 型
STR 型 （DbjDef.DATATYPE_STR）	文字列です。	String 型
STRLIST 型 （DbjDef.DATATYPE_STRLIST）	要素が String 型である Java のリスト型です。ログインユーザのユーザ情報（dbrProp_GroupList プロパティ）を取得する場合に使用します。	－

（凡例）

－：該当するデータ型がないことを示します。

DocumentBroker サーバで扱うデータ型については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

2.11.3　文書空間オブジェクトのプロパティの用途

文書空間オブジェクトのプロパティには次のような用途があります。

● 検索

文書空間オブジェクトを操作する場合，目的の文書空間オブジェクトは検索によって取得できます。文書空間オブジェクトのプロパティは，検索条件として指定できます。

このため，ユーザがプロパティを追加定義したり，プロパティに値を設定したりする場合には，どのような検索をしたいかを考慮して，検索のキーになるようなプロパティを追加定義したり，プロパティの値を設定したりしてください。

なお，ユーザ定義プロパティのほか，DMA プロパティ（dmaProp_OIID など）も検索に使用できます。

● ユーザ管理

ユーザが文書空間オブジェクトの管理情報として保持したい情報を，プロパティとして文書空間オブジェクトごとに設定できます。

● 文書空間オブジェクト間のリンクの管理

文書空間オブジェクト間を関連づけるリンクオブジェクトに対して，リンクの重要度などを示す値を設定するプロパティを定義して運用できます。リンクオブジェクトにプロパティを定義して運用すると，検索に使用したり，リンク付けた文書空間オブジェクトの一覧を取得する場合に，重要度などの値に応じてソートして，文書空間オブジェクトに順序性を持たせたりできるので便利です。

2.12 アクセス制御

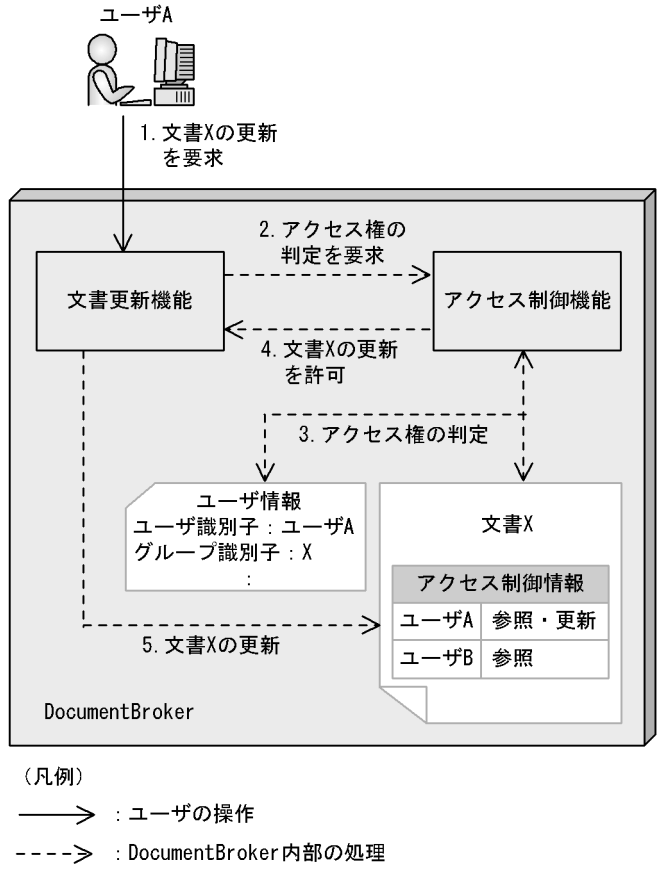
この節では，Java クラスライブラリでのアクセス制御の概念について説明します。

ユーザごとに文書やフォルダに対するアクセスの範囲を限定することを**アクセス制御**といいます。Java クラスライブラリで作成，管理する文書空間オブジェクトは，アクセス制御をしない状態では，DocumentBroker サーバにログインしたユーザであれば，だれでも参照したり更新したりできます。例えば，あるユーザが作成した文書を登録した場合，ほかのユーザがその文書を自由に参照したり，更新したり，削除したりできます。これでは，文書空間オブジェクトに対して，作成者以外のユーザが悪意を持って変更したり，誤って削除してしまったりすることが考えられます。

こうしたことを防ぐために，文書空間オブジェクトに対して，「どのユーザはどの操作が可能」というアクセスの範囲を限定する情報を設定しておく必要があります。Java クラスライブラリでは，文書空間オブジェクトごとに，「だれに対して」「どのような操作を許可するか」という**アクセス制御情報**を設定できます。

また，ユーザが DocumentBroker サーバにログインした時，「ログインしたユーザがだれなのか」という**ユーザ情報**が生成されます。ユーザが文書やフォルダに対する操作を要求すると，DocumentBroker サーバは，操作を要求したユーザのユーザ情報，要求された操作の種類および操作対象のオブジェクトに設定されているアクセス制御情報を比較します。比較した結果，ユーザが操作対象のオブジェクトに，要求した操作を実行する権利（**アクセス権**）があると判定できた場合だけ，要求を処理します。操作を要求したユーザに要求した範囲のアクセス権がない場合は，エラーを返却して要求を処理しません。アクセス制御の概要を次の図に示します。

図 2-11 アクセス制御の概要



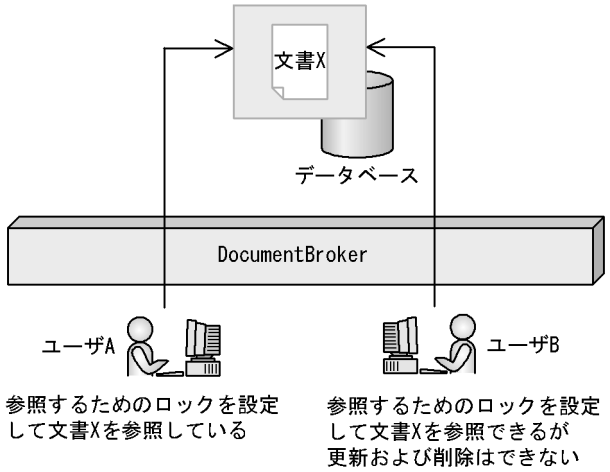
この例では，文書 X に対して，「ユーザ A からは参照と更新ができて，ユーザ B からは参照だけができるようにする」というアクセス制御情報が設定されています。ユーザ A が文書 X を更新しようとすると，ユーザ A のユーザ情報と文書 X のアクセス制御情報が比較され，文書 X の更新についてユーザ A のアクセス権が判定されます。文書 X のアクセス制御情報には，ユーザ A の「参照・更新」のアクセス権が設定されているため，アクセス権の判定の結果，ユーザ A による文書 X の更新が許可されて，文書 X が更新されます。

2.13 排他制御

この節では，Java クラスライブラリでの排他制御の概念について説明します。

Java クラスライブラリでは，文書空間オブジェクトを操作する場合には，ほかのユーザから同時にオブジェクトに対する操作が実行されないようにするために，排他制御をする必要があります。文書空間オブジェクトに対してロックを設定することによって，その文書空間オブジェクトの操作についてユーザ間の排他制御が実現します。ロックの設定例を次の図に示します。

図 2-12 ロックの設定例



この例では，ユーザ A は，参照するためのロックを設定して文書 X を参照しています。このため，ユーザ B は，同じロックを設定して文書 X を参照することはできますが，更新および削除はできません。

なお，Java クラスライブラリの排他制御には，メソッドによって暗黙的に設定されるロックと，ユーザが明示的に設定するロックがあります。

3 文書管理モデル

この章では，Java クラスライブラリでの文書管理体系で使用する各文書管理モデルについて説明します。モデルは複数組み合わせて導入できます。

3.1 バージョン管理モデル

この節では，文書空間オブジェクトのバージョン管理モデルについて説明します。

バージョンを管理できる文書空間オブジェクトは，バージョン付き文書およびバージョン付きフォルダです。

バージョン管理モデルは，ほかの文書管理モデルと組み合わせて使用できます。

ほかの文書管理モデルとの組み合わせ

- バージョン付き文書は，バージョン管理モデルとレンディション管理モデルを組み合わせて管理できます。レンディション管理モデルについては，「3.2　レンディション管理モデル」を参照してください。
- バージョン付きフォルダは，バージョン管理モデルとバージョン付きフォルダによる管理モデルを組み合わせて管理できます。バージョン付きフォルダによる管理モデルについては，「3.7　バージョン付きフォルダによる管理モデル」を参照してください。

ここでは，バージョン付きオブジェクトによるバージョン管理の概要と，バージョン付き文書によるバージョン管理について説明します。

3.1.1 バージョン管理の概要

ここでは，バージョン付きオブジェクトによるバージョン管理の概要について説明します。

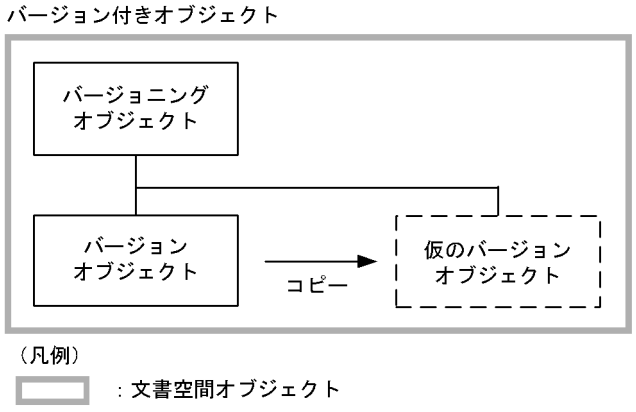
(1) バージョンの追加

バージョンを追加する場合には，チェックアウトとチェックインという操作が必要になります。

チェックアウト

チェックアウトは，新しいバージョンの追加を予約するために，最新バージョンの次に仮のバージョンを追加する操作です。チェックアウトによって，仮のバージョンのバージョンオブジェクトとして，最新バージョンのバージョンオブジェクトがコピーされて追加されます。バージョン付きオブジェクトのチェックアウトを次の図に示します。

図 3-1　バージョン付きオブジェクトのチェックアウト



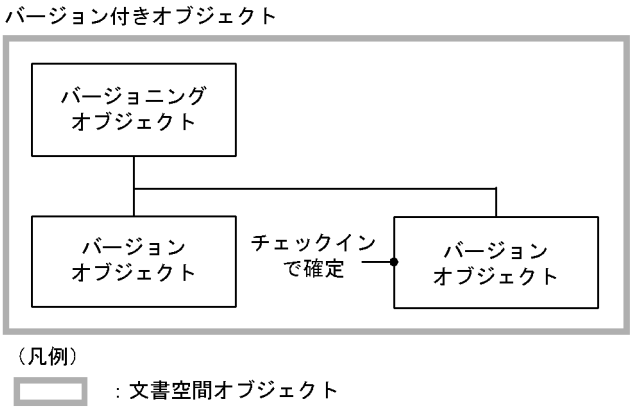
チェックアウトによって仮のバージョンが追加されたバージョン付きオブジェクトは，ほかのユーザによってチェックアウトできなくなります。チェックアウトは，チェックアウトされたバージョン付きオブジェクトがチェックインされるかまたはチェックアウトが取り消されるまで有効です。チェックアウトを取り消すと仮のバージョンは削除されます。なお，バージョン付き文書をチェックアウト

すると，追加された仮のバージョンには最新バージョンのマスタレンディションのコンテンツがコピーされて登録されますが，サブレンディションのコンテンツは，仮のバージョンには登録されません。このため，必要に応じてサブレンディションのコンテンツを追加してください。レンディションについては，「3.2　レンディション管理モデル」を参照してください。

チェックイン

チェックインは，チェックアウトで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操作です。チェックインすることでチェックアウトが解除され，バージョンが追加されます。バージョン付きオブジェクトのチェックインを次の図に示します。

図 3-2　バージョン付きオブジェクトのチェックイン



(2) バージョンの順序性とカレントバージョン

バージョンには順序性があります。バージョンの順序とは，それぞれのバージョンをチェックインした順序です。なお，最新のバージョンのことを**カレントバージョン**といいます。

(3) バージョンの指定方法

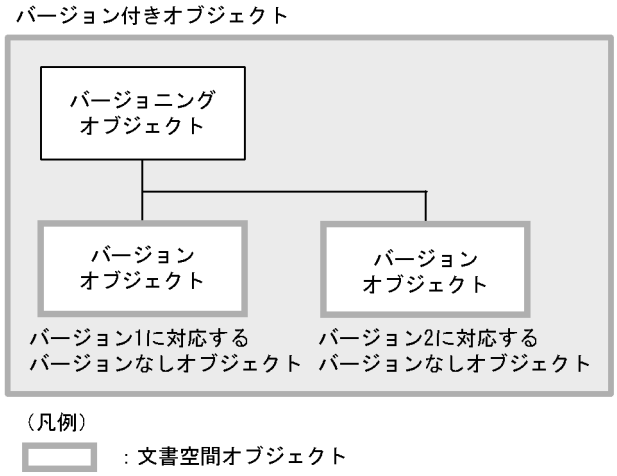
バージョン付きオブジェクトの個々のバージョンは，バージョン識別子によって指定できます。チェックアウトしている文書またはフォルダのバージョンを指定する場合は，チェックアウト時に取得した仮のバージョン識別子を指定します。また，バージョンのチェックアウト状態を確認して，すでにチェックアウトされている文書空間オブジェクトの仮のバージョン識別子を取得することもできます。

仮のバージョン識別子は，チェックイン後のバージョンの識別子とは異なりますので，注意してください。バージョンをチェックアウトしたあとで，チェックアウトを取り消した場合は，仮のバージョンオブジェクトおよび仮のバージョン識別子は削除されます。

(4) バージョン付きオブジェクトとバージョンなしオブジェクトの関係

バージョン付きオブジェクトでは，バージョン付きオブジェクトの１バージョンとしてバージョンなしオブジェクトを管理できます。バージョンなし文書はバージョン付き文書の１バージョンに対応し，バージョンなしフォルダはバージョン付きフォルダの１バージョンに対応します。つまり，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は，バージョンなしオブジェクトに対応します。バージョン付きオブジェクトとバージョンなしオブジェクトの関係を次の図に示します。

図 3-3　バージョン付きオブジェクトとバージョンなしオブジェクトの関係



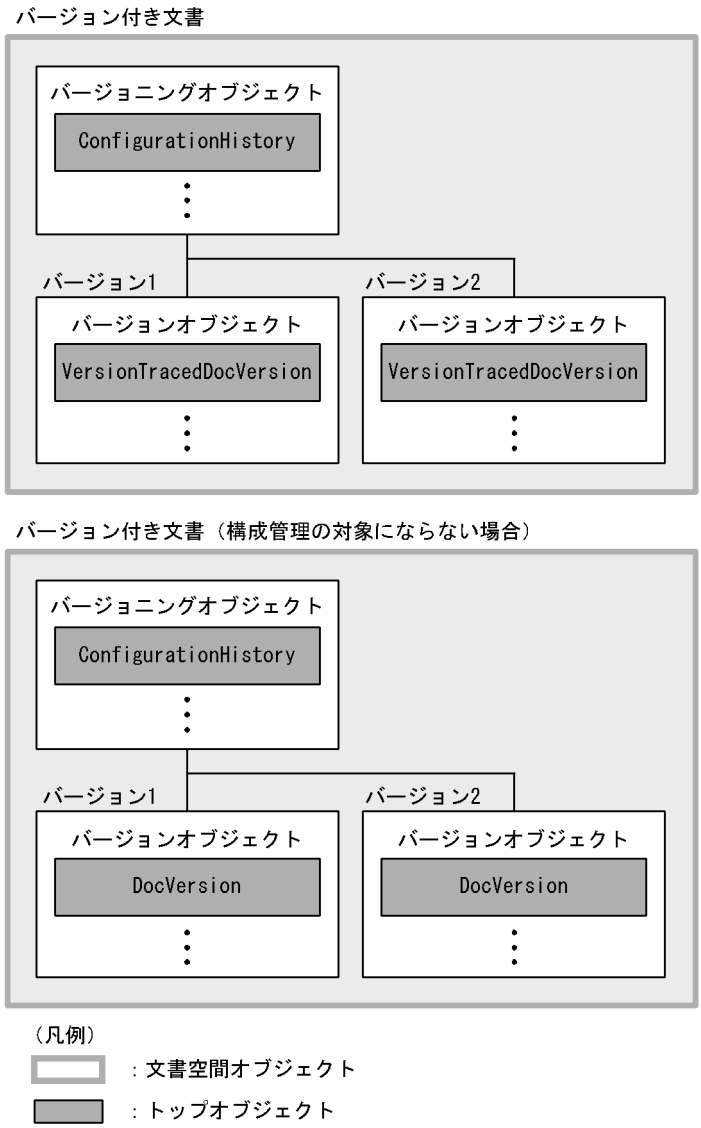
このため，バージョン付きオブジェクトのバージョンオブジェクトをバージョンなしオブジェクトとみなして，バージョンなしオブジェクトクラスの機能を使用して操作できます。また，バージョンなしオブジェクトから，自身をバージョンオブジェクトとして管理しているバージョニングオブジェクトをたどることもできます。

例えば，バージョン付き文書を全文検索する場合，実際に全文検索の対象となるコンテンツは個々のバージョンごとに存在します。このため，検索はバージョンオブジェクトであるバージョンなし文書が対象になり，検索結果としてバージョンオブジェクトのプロパティが取得できます。このとき，例えば，「全文検索で取得したバージョンなし文書を更新して，新しいバージョンとして追加したい」という場合などには，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をたどることもできます。

3.1.2　バージョン付き文書を構成する DMA オブジェクト

ここでは，バージョン付き文書を構成する DMA オブジェクトについて説明します。バージョン付き文書を構成する DMA オブジェクトを次の図に示します。

図 3-4 バージョン付き文書を構成する DMA オブジェクト



バージョン付き文書の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は，dmaClass_ConfigurationHistory クラスまたはそのサブクラスです。

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は，dmaClass_DocVersion クラスまたはそのサブクラスです。edmClass_VersionTracedDocVersion クラスまたはそのサブクラスを指定した場合，そのバージョン付き文書は構成管理の対象になります。dmaClass_DocVersion クラス，または edmClass_VersionTracedDocVersion クラス以外の dmaClass_DocVersion クラスのサブクラスを指定した場合，そのバージョン付き文書は構成管理の対象になりません。

バージョン付き文書を構成する DMA オブジェクトについて説明します。

● **ConfigurationHistory オブジェクト**

バージョン付き文書のバージョニングオブジェクトのトップオブジェクトです。この DMA オブジェクトのプロパティがバージョン付き文書クラスのデフォルトのプロパティとなり，バージョニングオブジェクトのプロパティとして使用できます。また，この DMA オブジェクトの OIID がバージョン付き文書の OIID となります。

● VersionTracedDocVersion **オブジェクト**

バージョンオブジェクトのトップオブジェクトです（トップオブジェクトクラスは，edmClass_VersionTracedDocVersion クラスまたはそのサブクラス）。この DMA オブジェクトで構成されるバージョン付き文書は，構成管理の対象になります。

この DMA 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として使用できます。

バージョン付き文書の個々のバージョンの識別には，この DMA オブジェクトの OIID ではなく，**バージョン識別子**を使用します。

● DocVersion **オブジェクト**

バージョンオブジェクトのトップオブジェクトです（トップオブジェクトクラスは，dmaClass_DocVersion クラス，または edmClass_VersionTracedDocVersion クラス以外の dmaClass_DocVersion クラスのサブクラス）。この DMA オブジェクトで構成されるバージョン付き文書は，構成管理の対象になりません。

この DMA 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として使用できます。

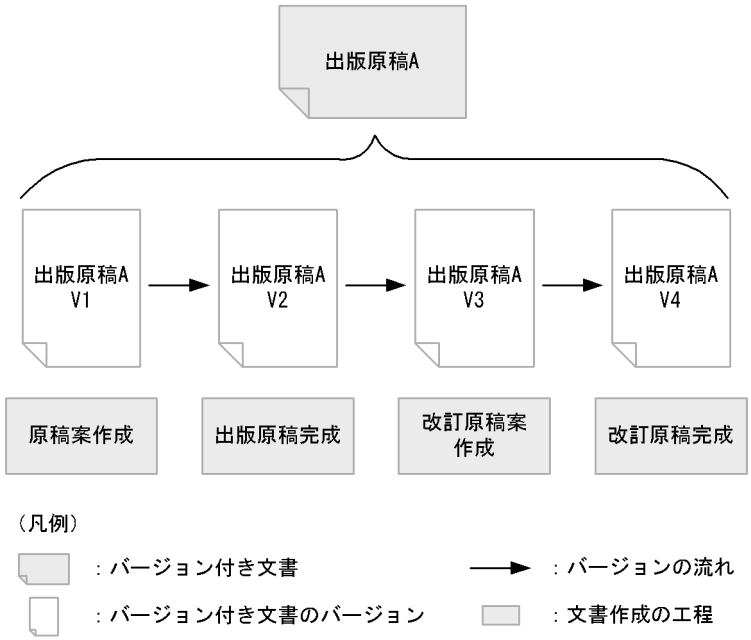
バージョン付き文書の個々のバージョンの識別には，この DMA オブジェクトの OIID ではなく，**バージョン識別子**を使用します。

3.1.3 バージョン付き文書による文書管理

ここでは，バージョン付き文書による文書管理について説明します。なお，バージョン付き文書を管理するための操作の詳細については，「6.8.10　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。

バージョン付き文書の管理例を次の図に示します。

図 3-5　バージョン付き文書の管理例



この例では，文書作成の工程に従って「出版原稿 A」という文書のバージョンを管理しています。この例では，原稿案を作成した時点で「出版原稿 A」を登録しています。そして，出版原稿が完成した時点，改訂版の原稿案を作成した時点および改訂版の出版原稿が完成した時点で「出版原稿 A」を更新してバージョンを追加しています。このように，新しく文書を登録した時点でバージョンを付け始め，履歴を残して一連の複数のバージョンを管理できます。Java クラスライブラリでは，このように連続する複数のバー

ジョンを持つ文書を一つの文書（バージョン付き文書）として管理できます。

また，バージョン付き文書と個々のバージョンには，プロパティを設定して管理できます。文書のプロパティとして「作成者」などを設定しておくと，文書の検索で使用できます。プロパティの管理については，「3.9　属性管理モデル」を参照してください。

3.2 レンディション管理モデル

この節では，レンディション管理モデルについて説明します。

3.2.1 レンディション管理の概要

ここでは，レンディション管理の概要について説明します。

(1) レンディションの種類

Java クラスライブラリでは，バージョンなし文書またはバージョン付き文書に 1 個または複数のレンディションを登録して管理します。レンディション管理モデルでは，次の二つのレンディションを区別して扱います。

- マスタレンディション
- サブレンディション

なお，このマニュアルでは，2 種類のレンディションを合わせてレンディションといいます。レンディションの種類によって説明が異なる場合は，それぞれ「マスタレンディション」，「サブレンディション」と記述して区別します。それぞれについて説明します。

マスタレンディション
主要なレンディションです。レンディションを 1 個だけ登録した場合は，そのレンディションがマスタレンディションになります。

サブレンディション
マスタレンディション以外のレンディションです。

文書をマルチレンディション文書として管理する場合，一つの文書に対して，1 個のマスタレンディションおよび最大 9 個のサブレンディションを登録できます。Java クラスライブラリでは，文書の作成時，コンテンツの更新時またはレンディションの追加時に，レンディションを登録できます。また，文書の作成時またはレンディションの追加時には，複数のレンディションを一括して登録できます。

サブレンディションは，削除できます。ただし，マスタレンディションは削除できません。また，サブレンディションをマスタレンディションに変更することもできます。この場合，それまでマスタレンディションだったレンディションはサブレンディションに変更されます。

(2) レンディションのコンテンツの管理

文書のコンテンツ（Word やテキストエディタなどのアプリケーションプログラムで作成された文書ファイル）は，プロパティおよびレンディションタイプ（MIME 形式）の情報をあわせて，レンディションとして文書に登録して管理します。

文書のコンテンツに対しては，次の操作ができます。

● 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して参照します。

● コンテンツのアップロード
文書のコンテンツを登録します。また，マスタレンディションのコンテンツの登録時には，登録するコンテンツの全文検索インデクスを作成することもできます。

(3) マルチレンディション文書の用途

バージョンなし文書またはバージョン付き文書に複数のレンディションを登録して，マルチレンディショ

ン文書として管理すると，次のような場合に使用できます。

- あるアプリケーションプログラムで作成したファイルを，そのアプリケーションプログラムをインストールしていないマシンでも参照できるように，同じ内容の HTML 形式および PDF 形式のファイルを作成して一緒に登録したいという場合
- 同じ拡張子のファイルでも，保存状態やアプリケーションプログラムのバージョンの違いごとに複数登録したい場合（例えば，拡張子が同じ「.gif」であっても解像度の違うファイルや，拡張子が同じ「.doc」であっても Word97 形式で保存されたファイルと Word2000 形式で保存されたファイルなど）
- アクセス制御モデルとの連携で，アクセス権によって参照できるファイルの形式を限定するために，更新できる形式のファイルと更新できない形式のファイル（例えば，同じ内容の Word 形式のファイルと PDF 形式のファイル）を両方登録したい場合

アクセス制御モデルについては，「3.10　アクセス制御モデル」を参照してください。

（4） DocumentBroker Rendering Option との連携

DocumentBroker Rendering Option と連携すると，登録済みのマスタレンディションのコンテンツを基に，別のレンディションタイプのファイルを自動作成してサブレンディションのコンテンツとして登録できます。この機能を**レンディション変換**といいます。また，DocumentBroker Rendering Option にレンディション変換を要求する機能を**レンディション変換要求機能**といいます。レンディションにコンテンツが登録されていない場合や，サブレンディションのコンテンツが登録されたあとにマスタレンディションのコンテンツが更新された場合などにレンディション変換要求機能を使用すれば，コンテンツの同期を取りながら複数のレンディションを管理できます。

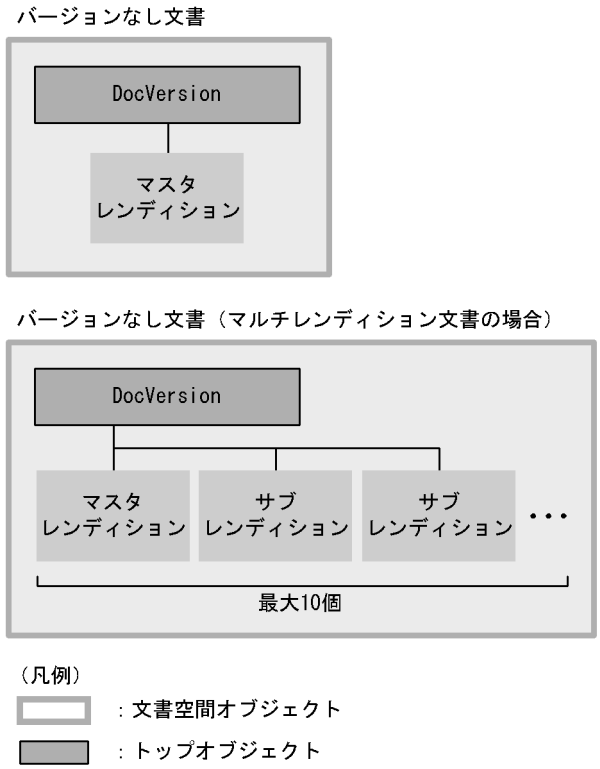
DocumentBroker Rendering Option の変換対象のレンディションタイプ，対応するファイルの拡張子とアプリケーションプログラムのバージョンなど，DocumentBroker Rendering Option の詳細については，マニュアル「DocumentBroker Rendering Option　システム導入・運用ガイド」を参照してください。

3.2.2　レンディションを管理する文書を構成する DMA オブジェクト

ここでは，レンディションを管理するバージョンなし文書およびバージョン付き文書を構成する DMA オブジェクトについて説明します。

レンディションを登録したバージョンなし文書を構成する DMA オブジェクトを次の図に示します。

図 3-6 レンディションを登録したバージョンなし文書を構成する DMA オブジェクト



バージョンなし文書のトップオブジェクトクラスは，dmaClass_DocVersion クラスまたはそのサブクラスです。

バージョンなし文書を構成する DMA オブジェクトについて説明します。

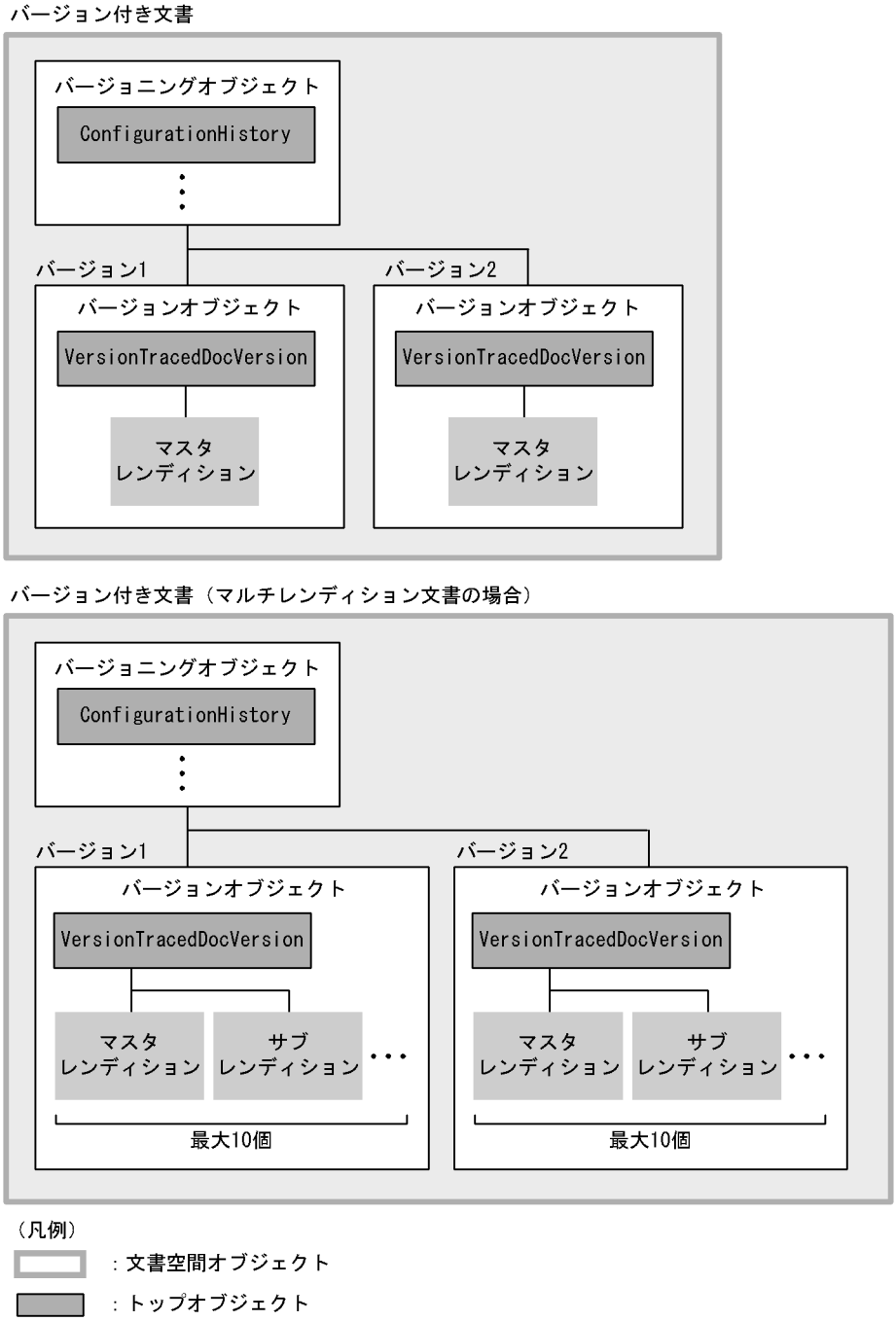
● **DocVersion オブジェクト**

バージョンなし文書のトップオブジェクトです。

バージョンなし文書をマルチレンディション文書として管理する場合は，一つの DocVersion オブジェクトに複数のレンディションを登録できます。

次に，バージョン付き文書について説明します。レンディションを登録したバージョン付き文書を構成する DMA オブジェクトを次の図に示します。

図 3-7　レンディションを登録したバージョン付き文書を構成する DMA オブジェクト



バージョン付き文書を構成する DMA オブジェクトの詳細については，「3.1.2　バージョン付き文書を構成する DMA オブジェクト」を参照してください。

バージョン付き文書をマルチレンディション文書として管理する場合は，一つの VersionTracedDocVersion オブジェクト（または DocVersion オブジェクト）に複数のレンディションを登録します。

3.2.3　レンディションによる文書管理

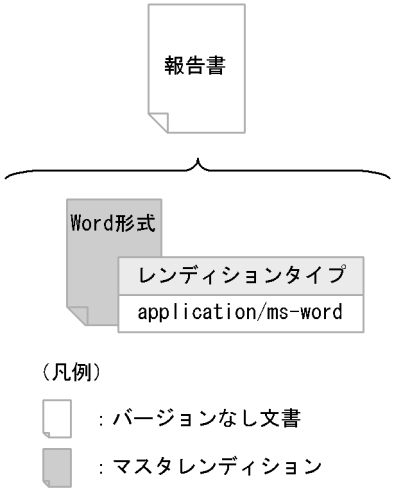
ここでは，レンディションによる文書管理について説明します。なお，マルチレンディション文書を管理

するための操作の詳細については，「6.8.11　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。文書のコンテンツを管理するための操作の詳細については，「6.8.9　文書のコンテンツの操作」を参照してください。

(1) レンディションの登録

バージョンなし文書に 1 個のレンディションを登録して管理している例を次の図に示します。

図 3-8　バージョンなし文書のレンディションの管理例

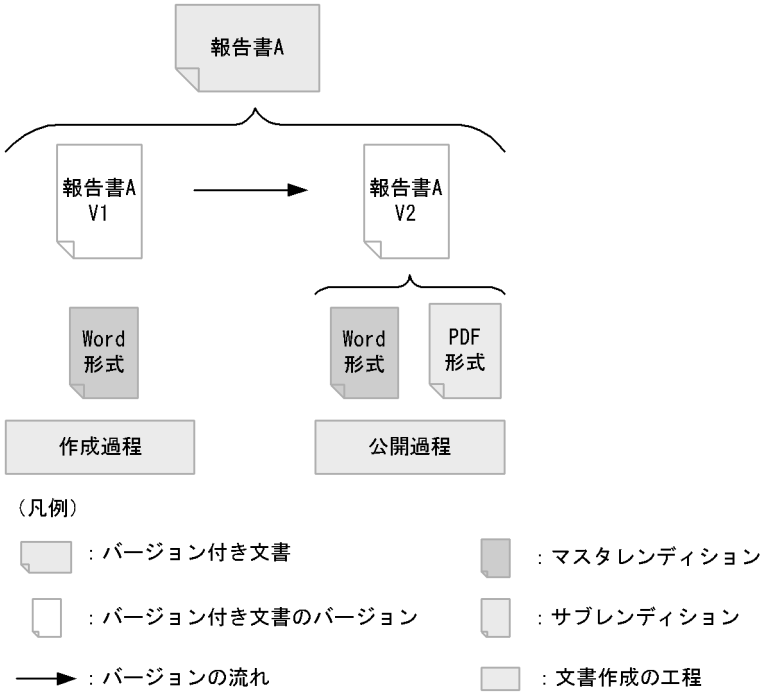


この例では，バージョンなし文書「報告書」に，1 個のレンディションを登録して管理しています。バージョンなし文書「報告書」には，コンテンツとして Word 形式のファイルと，レンディションタイプとしてコンテンツを作成したアプリケーションプログラムの名前を登録しています。

(2) レンディションの追加

バージョン付き文書にレンディションを追加して，マルチレンディション文書として管理する例を次の図に示します。

図 3-9 マルチレンディション文書のレンディションの追加例



この例では，文書の作成過程で，文書の作成者が更新用にコンテンツをダウンロードできるように，コンテンツを作成しているアプリケーションプログラム（この例の場合 Word）の形式のレンディションを登録しています。この Word 形式のレンディションがマスタレンディションとなります。そして，文書が完成して公開過程になったタイミングでバージョンを追加して，追加したバージョンをほかのユーザが参照しやすいように，PDF 形式のレンディションを追加登録しています。この追加した PDF 形式のレンディションがサブレンディションとなります。

なお，マルチレンディション文書をチェックアウトすると，最新バージョンのマスタレンディションが仮のバージョンにコピーされます。サブレンディションはコピーされません。例えば，図 3-9 の場合，「報告書 A」にさらにバージョンを追加するときは，チェックアウトすると，Word 形式のレンディションはコピーされますが，PDF 形式のレンディションはコピーされません。このため，仮のバージョンにサブレンディションのコンテンツを登録してからチェックインするなど，必要に応じてサブレンディションのコンテンツを追加してください。

(3) レンディションの参照

マルチレンディション文書は，レンディションタイプを指定して参照できます。例えば，図 3-9 の場合，「報告書 A」のバージョン 2 を参照するときは，レンディション一覧を取得して，Word 形式か PDF 形式かを指定して参照します。文書の作成者が文書をバージョンアップしたいときには，Word でファイルを編集するために Word 形式を指定してダウンロードしたり，Word をインストールしていない環境のユーザは PDF 形式を指定してダウンロードしたりするなど，目的や，コンテンツを参照するマシン環境に応じた参照ができます。

また，レンディション管理モデルとアクセス制御モデルを組み合わせて，マルチレンディション文書を管理することもできます。例えば，Word 形式のコンテンツと PDF 形式のコンテンツを登録した文書がある場合に，「基本コンテンツ更新権のあるユーザには Word 形式のコンテンツを参照させる」，「基本コンテンツ参照権しかないユーザには PDF 形式のコンテンツを参照させる」というように，ユーザの保持するアクセス権によって，ユーザが参照できるコンテンツを限定するという運用が考えられます。このとき，

Java クラスライブラリでは文書ごとのアクセス制御しかできないため，さらにレンディションごとのアクセス制御をユーザアプリケーションプログラムで処理する必要があります。アクセス制御モデルについては，「3.10　アクセス制御モデル」を参照してください。

(4) 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の更新

マルチレンディション文書のコンテンツを更新する場合，レンディションタイプを指定することによって，レンディションを指定して更新できます。例えば，600dpi の GIF ファイルをマスタレンディション，300dpi の GIF ファイルをサブレンディションとして管理している場合に，300dpi の GIF ファイルを更新するときは，該当するレンディションのレンディションタイプを指定してください。

マスタレンディションのコンテンツを更新した場合には，サブレンディションのコンテンツも更新してください。

コンテンツの更新時には，レンディションタイプを変更できます。また，マスタレンディションのコンテンツの更新時には，全文検索インデクスを作成できます。

(5) レンディションの削除

マルチレンディション文書のレンディションを削除する場合，削除対象のレンディションのレンディションタイプを指定することによって，複数のレンディションを一括して削除できます。

マスタレンディションは削除できません。なお，マルチレンディション文書自体を削除した場合は，すべてのレンディションが削除されます。

(6) レンディションの検索

レンディションタイプを表す dbrProp_RenditionType プロパティは，検索条件には指定できません。例えば，「マスタレンディションのレンディションタイプが "application/ms-word" である文書を検索する」のような指定はできません。また，検索結果として，レンディションタイプを取得することもできません。レンディションタイプを参照したい場合は，ほかの検索条件によって参照したい文書を特定してから取得してください。

3.2.4 レンディションのプロパティ

ここでは，レンディションのプロパティについて説明します。なお，この説明に出てくるメソッドについては，「6.　Java クラスライブラリの機能」およびマニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」を参照してください。

レンディションには，プロパティを設定して管理できます。ただし，レンディションのプロパティとしてユーザ定義プロパティは設定できません。レンディションに設定されているプロパティは，次の 4 種類です。

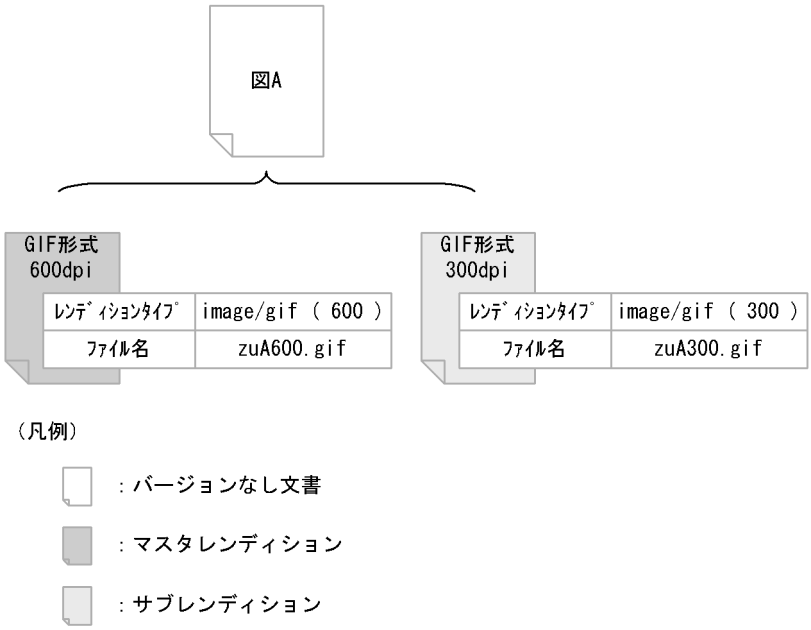
- dbrProp_RenditionType プロパティ
  レンディションタイプを表すプロパティです。
- dbrProp_RetrievalName プロパティ
  レンディションに登録されたコンテンツのファイル名を表すプロパティです。
- dbrProp_RenditionStatus プロパティ
  レンディション状態を表すプロパティです。
- dbrProp_ContentType プロパティ
  レンディションのコンテンツ種別を表すプロパティです。

これらのプロパティは，属性検索条件として指定したり，検索結果一覧の情報として取得したりすること

はできません。

レンディションタイプとコンテンツのファイル名を設定したレンディションのプロパティの管理例を次の図に示します。

図 3-10　レンディションのプロパティの管理例



次に，レンディションの各プロパティについて，説明します。

(1) レンディションタイプを表すプロパティ（dbrProp_RenditionType プロパティ）

レンディション管理モデルでは，同じレンディションタイプを重複して登録できません。ただし，" タイプ / サブタイプ (コメント)" などの形式でレンディションタイプを指定できるため，同じ拡張子のファイルでもレンディションタイプを重複させないで登録できます。例えば，図 3-10 の場合，「図 A」というバージョンなし文書のマスタレンディションとサブレンディションに，同じ GIF 形式でも異なる解像度（マスタレンディションは 600dpi，サブレンディションは 300dpi）で保存したコンテンツを登録しています。このため，"image/gif (600)" および "image/gif (300)" のように，コメントまで指定したレンディションタイプを設定して二つの GIF 形式のレンディションを登録しています。

レンディションタイプは，文書の作成時，コンテンツの更新時またはレンディションの追加時に指定します。このとき指定したレンディションタイプが，dbrProp_RenditionType プロパティの値として設定されます。

文書の作成時，コンテンツの更新時またはレンディションの追加時に，レンディションタイプの指定を省略すると，Java クラスライブラリは，レンディション定義ファイルの定義に従ってコンテンツのファイルの拡張子から判断して，レンディションタイプを自動的に設定します。レンディション定義ファイルについては，「7.6.2　レンディション定義ファイル」を参照してください。ただし，この例のように拡張子が同じでもファイルの保存状態などによって別々のレンディションに登録したい場合は，ファイルの拡張子からは判断できないので，コメントまで指定したレンディションタイプを明示的に指定してください。

なお，dbrProp_RenditionType プロパティは，STR 型のプロパティです。

(2) ファイル名を表すプロパティ（dbrProp_RetrievalName プロパティ）

レンディションタイプのほかに，各レンディションには，ファイル名を表す dbrProp_RetrievalName プロパティを設定できます。

ファイル名は，文書の作成時，コンテンツの更新時またはレンディションの追加時に指定します。このとき指定したファイル名が，dbrProp_RetrievalName プロパティの値として設定されます。図 3-10 の例では，マスタレンディションには zuA600.gif，サブレンディションには zuA300.gif という値が設定されています。

なお，dbrProp_RetrievalName プロパティは，STR 型のプロパティです。

(3) レンディション状態を表すプロパティ（dbrProp_RenditionStatus プロパティ）

マルチレンディション文書では，同じ内容を持つ複数のレンディションタイプのコンテンツを，まとめて一つの文書に登録して管理できます。このため，マスタレンディションとサブレンディションは，常に同じ内容にしておく必要があります。マスタレンディションのコンテンツを更新した場合には，サブレンディションのコンテンツも更新する必要があります。

マスタレンディションとサブレンディションの dbrProp_RenditionStatus プロパティの値について説明します。

マスタレンディションの場合

マスタレンディションの dbrProp_RenditionStatus プロパティの値は常に DbjDef.RENDSTATUS_MASTERREND です。

サブレンディションの場合

サブレンディションの dbrProp_RenditionStatus プロパティの値は，メソッドの実行によって遷移します。また，サブレンディションの dbrProp_RenditionStatus プロパティの値は，状態フラグと変換フラグの組み合わせで表されます。

状態フラグ

マスタレンディションに対するサブレンディションの更新状態を示す値です。状態フラグの値は，DocumentBroker によって自動的に設定されます。ユーザは状態フラグの値を変更できません。

変換フラグ

DocumentBroker Rendering Option を使用する場合に，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によって自動作成するかどうかを示す値です。また，変換フラグには，レンディション変換でエラーが発生したことを示す値も設定されます。変換フラグの値は，DocumentBroker によって自動的に設定されます。ユーザは変換フラグの値を変更できます。なお，変換フラグは，DocumentBroker Rendering Option を使用しない場合は使用しません。

なお，dbrProp_RenditionStatus プロパティは，INT 型のプロパティです。

次に，サブレンディションの状態フラグと変換フラグの値について説明します。

(a) 状態フラグの値

サブレンディションの状態フラグの値は，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の更新時に，DocumentBroker によって自動的に設定されます。サブレンディションの状態フラグの値を次の表に示します。

表 3-1 サブレンディションの状態フラグの値

状態フラグを表す定数	説明
DbjDef.RENDSTATUS_NO_SUBREND	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登録されていません。
DbjDef.RENDSTATUS_SUBREND_EXIST	マスタレンディションとサブレンディションの更新状態が一致しています。
DbjDef.RENDSTATUS_MASTERREND_UPDATE	マスタレンディションとサブレンディションの更新状態が不一致です。マスタレンディションのコンテンツは更新されていますが，サブレンディションのコンテンツが更新されていません。

(b) 変換フラグの値

サブレンディションの変換フラグの値は，文書の作成時，コンテンツの更新時またはレンディションの追加時に，コンテンツのファイルパスの指定の有無に応じて DocumentBroker が自動的に設定します。DocumentBroker Rendering Option によるレンディション変換実行時にエラーが発生した場合には，変換エラーを示す値が設定されます。

変換フラグの値をユーザが変更する場合は，レンディションのプロパティを更新する DbjObj#writeRenditionProperties メソッドを使用します。

サブレンディションの変換フラグの値を次の表に示します。

表 3-2 サブレンディションの変換フラグの値

変換フラグを表す定数	説明
DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED	DocumentBroker Rendering Option によるレンディション変換の対象になりません。 レンディション変換の対象にしないサブレンディションに対して指定してください。
DbjDef.RENDSTATUS_CONVERT_REQUIRED	DocumentBroker Rendering Option によるレンディション変換の対象になります。 レンディション変換の対象にする場合に指定してください。
DbjDef.RENDSTATUS_CONVERT_ERROR	DocumentBroker Rendering Option で変換エラーが発生しました。 DocumentBroker Rendering Option が設定する値です。

なお，状態フラグの値によって，変換フラグが取る値が異なります。

状態フラグと変換フラグの組み合わせについて，次の表に示します。

表 3-3 状態フラグと変換フラグの組み合わせ

状態フラグを表す定数	変換フラグを表す定数
DbjDef.RENDSTATUS_SUBREND_EXIST	• DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED • DbjDef.RENDSTATUS_CONVERT_REQUIRED
DbjDef.RENDSTATUS_NO_SUBREND	• DbjDef.RENDSTATUS_CONVERT_REQUIRED • DbjDef.RENDSTATUS_CONVERT_ERROR
DbjDef.RENDSTATUS_MASTERREND_UPDATE	• DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED • DbjDef.RENDSTATUS_CONVERT_REQUIRED • DbjDef.RENDSTATUS_CONVERT_ERROR

例えば，サブレンディションの文書のアップロード情報でコンテンツのファイルパスを指定しなかった（null を指定した）場合，DocumentBroker Rendering Option によるレンディション変換を要求しない変換フラグ（DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED）を指定することはできません。

また，文書のアップロード情報でコンテンツのファイルパスを指定して更新したいサブレンディションに対しては，マスタレンディションとサブレンディションの更新状態が不一致（DbjDef.RENDSTATUS_MASTERREND_UPDATE）でも，DocumentBroker Rendering Option によるレンディション変換を要求しない変換フラグ（DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED）を指定できます。

（4） dbrProp_RenditionStatus プロパティとメソッドの関係

dbrProp_RenditionStatus プロパティの値とメソッドの関係について説明します。

次のメソッドについて説明します。

- dbrProp_RenditionStatus プロパティの値を参照するメソッド
- dbrProp_RenditionStatus プロパティの値を遷移させるメソッド
- dbrProp_RenditionStatus プロパティの変換フラグを変更するメソッド
- dbrProp_RenditionStatus プロパティの値によって制限されるメソッド

（a） dbrProp_RenditionStatus プロパティの値を参照するメソッド

dbrProp_RenditionStatus プロパティの値は，次のメソッドによるレンディション一覧の取得時に参照できます。

● DbjObj#getRenditionList メソッド
特定のレンディションの dbrProp_RenditionStatus プロパティを参照したい場合は，
DbjRenditionList インターフェースおよび DbjRenditionInfo インターフェースを使用してください。

（b） dbrProp_RenditionStatus プロパティの値を遷移させるメソッド

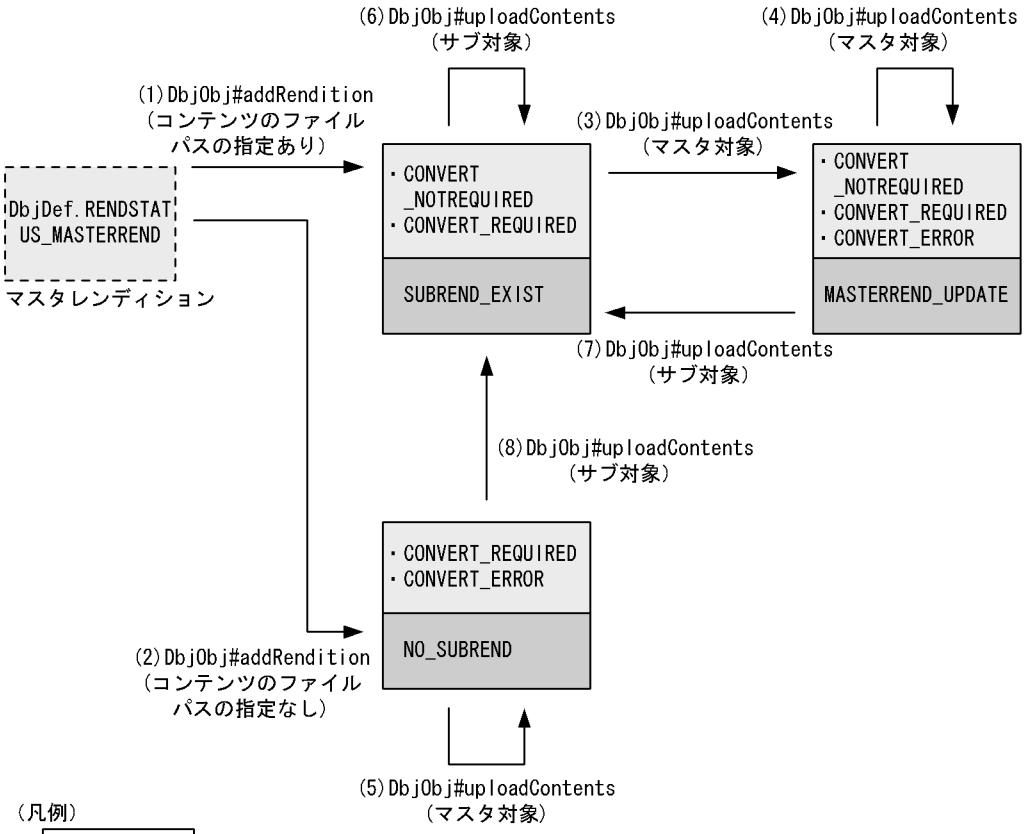
dbrProp_RenditionStatus プロパティの値は，次のメソッドの実行時に，DocumentBroker が自動的に値を設定することによって，遷移します。

- DbjObj#changeMasterRendition メソッド
- DbjObj#addRendition メソッド
- DbjObj#uploadContents メソッド

DbjObj#changeMasterRendition メソッドによる dbrProp_RenditionStatus プロパティの値の遷移については，「(d) dbrProp_RenditionStatus プロパティの値によって制限されるメソッド」を参照してください。

DbjObj#addRendition メソッドおよび DbjObj#uploadContents メソッドによる dbrProp_RenditionStatus プロパティの値の遷移を，次の図に示します。図中の (1) から (8) の番号は，表 3-4 から表 3-6 の説明中の番号と対応しています。

図 3-11 メソッドによる dbrProp_RenditionStatus プロパティの値の遷移



変換フラグ（上位2バイト）

CONVERT_NOTREQUIRED：DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED（変換不要）
CONVERT_REQUIRED：DbjDef.RENDSTATUS_CONVERT_REQUIRED（変換要）
CONVERT_ERROR：DbjDef.RENDSTATUS_CONVERT_ERROR（変換エラー）

状態フラグ（下位2バイト）

NO_SUBREND：DbjDef.RENDSTATUS_NO_SUBREND
（サブレンディションのコンテンツなし）
SUBREND_EXIST：DbjDef.RENDSTATUS_SUBREND_EXIST
（サブレンディションのコンテンツあり。マスタレンディションの状態と一致）
MASTERREND_UPDATE：DbjDef.RENDSTATUS_MASTERREND_UPDATE
（サブレンディションのコンテンツあり。マスタレンディションの状態と不一致）

マスタ対象：マスタレンディションに対してメソッドを実行することを示します。
サブ対象　：サブレンディションに対してメソッドを実行することを示します。

図で示した遷移の詳細を，メソッドごとに説明します。

● **DbjObj#addRendition メソッド**

追加したサブレンディションのプロパティには，コンテンツのファイルパスの指定に応じて，次の表に示す値が設定されます。

表 3-4 DbjObj#addRendition メソッドによって設定される dbrProp_RenditionStatus プロパティの値

コンテンツのファイルパスの指定	変換フラグ	状態フラグ
あり（(1) の遷移）	DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED	DbjDef.RENDSTATUS_SUBREND_EXIST

コンテンツのファイルパスの指定	変換フラグ	状態フラグ
なし（(2) の遷移） （null を指定した場合）	DbjDef.RENDSTATUS_CONVERT_REQUIRED	DbjDef.RENDSTATUS_NO_SUBREND

● DbjObj#uploadContents メソッド（マスタレンディションを更新する場合）

メソッドを実行した時点の状態フラグに応じて，次の表に示すように遷移します。

表 3-5 DbjObj#uploadContents メソッドによる dbrProp_RenditionStatus プロパティの値の遷移（マスタレンディションを更新する場合）

メソッドを実行した時点の状態フラグ	変換フラグの遷移	メソッドを実行したあとの状態フラグ
DbjDef.RENDSTATUS_SUBREND_EXIST（(3) の遷移）	DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED の場合： DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED （遷移しません）	DbjDef.RENDSTATUS_MASTERREND_UPDATE
	DbjDef.RENDSTATUS_CONVERT_REQUIRED の場合： DbjDef.RENDSTATUS_CONVERT_REQUIRED （遷移しません）	
DbjDef.RENDSTATUS_MASTERREND_UPDATE（(4) の遷移）	DbjDef.RENDSTATUS_CONVERT_ERROR の場合： DbjDef.RENDSTATUS_CONVERT_REQUIRED ※	DbjDef.RENDSTATUS_MASTERREND_UPDATE （遷移しません）
	DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED の場合： DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED （遷移しません）	
	DbjDef.RENDSTATUS_CONVERT_REQUIRED の場合： DbjDef.RENDSTATUS_CONVERT_REQUIRED （遷移しません）	
DbjDef.RENDSTATUS_NO_SUBREND（(5) の遷移）	DbjDef.RENDSTATUS_CONVERT_ERROR の場合： DbjDef.RENDSTATUS_CONVERT_REQUIRED ※	DbjDef.RENDSTATUS_NO_SUBREND （遷移しません）
	DbjDef.RENDSTATUS_CONVERT_REQUIRED の場合： DbjDef.RENDSTATUS_CONVERT_REQUIRED （遷移しません）	

注※ DocumentBroker Rendering Option でレンディション変換エラーが起きたあとにマスタレンディションを更新し直す場合は，更新するコンテンツにエラーの要因が残っていないかどうか確認してください。または，マスタレンディションのコンテンツを基に，別のレンディションタイプのファイルをユーザが作成して，サブレンディションのコンテンツとして登録してください。

● DbjObj#uploadContents メソッド（サブレンディションを更新する場合）

メソッドを実行した時点の状態フラグに応じて，次の表に示すように遷移します。

表 3-6 DbjObj#uploadContents メソッドによる dbrProp_RenditionStatus プロパティの値の遷移（サブレンディションを更新する場合）

メソッドを実行した時点の状態フラグ	変換フラグの遷移	メソッドを実行したあとの状態フラグ
DbjDef.RENDSTATUS_SUBREND_EXIST（(6) の遷移）	DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED の場合： DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED （遷移しません）	DbjDef.RENDSTATUS_SUBREND_EXIST （遷移しません）
	DbjDef.RENDSTATUS_CONVERT_REQUIRED の場合： DbjDef.RENDSTATUS_CONVERT_REQUIRED （遷移しません）	
DbjDef.RENDSTATUS_MASTERREND_UPDATE（(7) の遷移）	DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED	DbjDef.RENDSTATUS_SUBREND_EXIST
DbjDef.RENDSTATUS_NO_SUBREND（(8) の遷移）	DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED	DbjDef.RENDSTATUS_SUBREND_EXIST

（c） dbrProp_RenditionStatus プロパティの変換フラグを変更するメソッド

変換フラグの値をユーザが変更する場合は，DbjObj#writeRenditionProperties メソッドを使用します。

設定できる値は次のどちらかです。

- DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED
- DbjDef.RENDSTATUS_CONVERT_REQUIRED

ただし，次の場合は値を設定できません。

DbjObj#writeRenditionProperties メソッドで変換フラグを設定できない場合

- 状態フラグが DbjDef.RENDSTATUS_NO_SUBREND の場合，変換フラグに DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED は設定できません。
  状態フラグが DbjDef.RENDSTATUS_NO_SUBREND の場合は，サブレンディションのコンテンツを DbjObj#uploadContents メソッドで更新すると，変換フラグを DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED に遷移させることができます。
- 変換フラグが DbjDef.RENDSTATUS_CONVERT_ERROR の場合，変換フラグに DbjDef.RENDSTATUS_CONVERT_NOTREQUIRED または DbjDef.RENDSTATUS_CONVERT_REQUIRED は設定できません。
  変換フラグが DbjDef.RENDSTATUS_CONVERT_ERROR の場合は，エラーの要因を取り除いたコンテンツを準備して，マスタレンディションを DbjObj#uploadContents メソッドで再び更新してください。または，マスタレンディションのコンテンツを基に，別のレンディションタイプのファイルをユーザが作成して，DbjObj#uploadContents メソッドでサブレンディションのコンテンツとして登録してください。

（d） dbrProp_RenditionStatus プロパティの値によって制限されるメソッド

状態フラグと変換フラグの組み合わせによっては，DbjObj#changeMasterRendition メソッドの実行が制限されます。制限されるのは，次の場合です。

DbjObj#changeMasterRendition メソッドの実行が制限される場合

- 状態フラグが DbjDef.RENDSTATUS_NO_SUBREND の場合，DbjObj#changeMasterRendition メソッドは実行できません。

- 状態フラグが DbjDef.RENDSTATUS_MASTERREND_UPDATE の場合，変換フラグが DbjDef.RENDSTATUS_CONVERT_REQUIRED または DbjDef.RENDSTATUS_CONVERT_ERROR のときは，DbjObj#changeMasterRendition メソッドは実行できません。

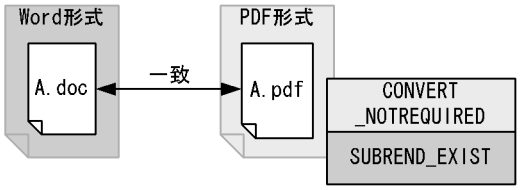
これ以外の組み合わせでは，DbjObj#changeMasterRendition メソッドを実行してマスタレンディションを変更すると，それまでサブレンディションだったレンディションはマスタレンディションに変更されて，そのレンディションの dbrProp_RenditionStatus プロパティの値は DbjDef.RENDSTATUS_MASTERREND に遷移します。

（e） DocumentBroker Rendering Option によるサブレンディション更新の例

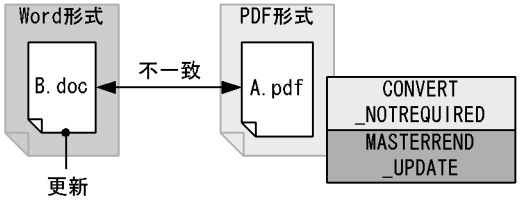
dbrProp_RenditionStatus プロパティの値が遷移する例として，DocumentBroker Rendering Option によるサブレンディション更新の例を次の図に示します。

図 3-12　DocumentBroker Rendering Option によるサブレンディション更新の例

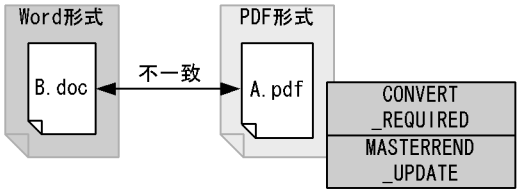
1. マスタレンディションとサブレンディションの内容が一致している。



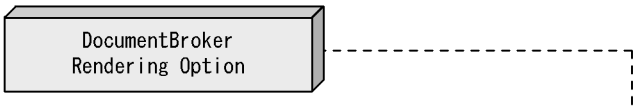
2. マスタレンディションに対してDbjObj#uploadContentsメソッドを実行して，
マスタレンディションのコンテンツを更新する。



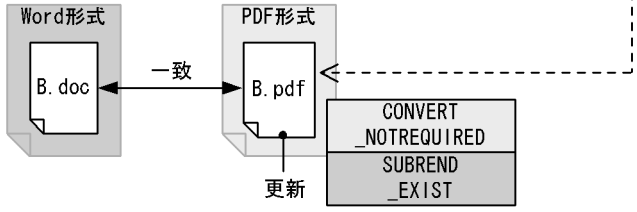
3. サブレンディションに対してDbjObj#writeRenditionPropertiesメソッドを
実行して，変換フラグにDbjDef.RENDSTATUS_CONVERT_REQUIREDを設定する。

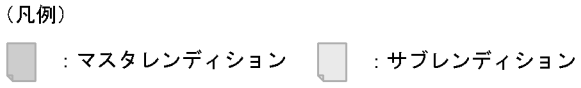


4. レンディション変換を実行する。



5. DocumentBroker Rendering Optionによってサブレンディション
のコンテンツが更新される。





図について説明します。

1. 図中の文書は，マスタレンディションのコンテンツとして Word 形式のファイル「A.doc」，サブレンディションのコンテンツとして PDF 形式のファイル「A.pdf」を持つ，バージョンなし文書です。このとき，マスタレンディションとサブレンディションの内容は一致しています。状態フラグは DbjDef.RENDSTATUS_SUBREND_EXIST です。
2. DbjObj#uploadContents メソッドによって，マスタレンディションのコンテンツを「B.doc」に更新します。このとき，マスタレンディションとサブレンディションの状態が不一致になるため，サブレンディションの状態フラグが DbjDef.RENDSTATUS_MASTERREND_UPDATE に遷移します。
3. DbjObj#writeRenditionProperties メソッドによって，サブレンディションの dbrProp_RenditionStatus プロパティの変換フラグに，

DbjDef.RENDSTATUS_CONVERT_REQUIRED を設定します。

4. DocumentBroker Rendering Option によってレンディション変換を実行します。3. で変換フラグに DbjDef.RENDSTATUS_CONVERT_REQUIRED を設定したサブレンディションは，レンディション変換の対象になります。レンディション変換は，マスタレンディションのコンテンツ「B.doc」を基に実行されます。
5. サブレンディションのコンテンツとして，マスタレンディションの「B.doc」を基にレンディション変換したコンテンツ「B.pdf」が登録されます。このとき，サブレンディションの状態フラグは，DbjDef.RENDSTATUS_SUBREND_EXIST に遷移します。

(5) レンディションのコンテンツ種別を表すプロパティ（dbrProp_ContentType プロパティ）

レンディションごとのコンテンツの種別は，dbrProp_ContentType プロパティによって知ることができます。コンテンツの種別から，レンディションのコンテンツが次のどれかを知ることができます。

- シングルファイル文書のコンテンツ
- リファレンスファイル文書のコンテンツ
- 上記以外のコンテンツ

このプロパティは，Integer32 型の 4 バイトの値として表されます。

ユーザは dbrProp_ContentType プロパティを設定できません。dbrProp_ContentType プロパティを参照する場合は，DbjObj#getRenditionList メソッドを使用します。

dbrProp_ContentType の値とコンテンツ種別の関係を次の表に示します。

表 3-7 dbrProp_ContentType の値とコンテンツ種別の関係

dbrProp_ContentType の値	対応する値	コンテンツ種別
DBR_CONTENTTYPE_CONTENT	0	ContentTransfer オブジェクト（シングルファイル文書であるコンテンツ）です。
DBR_CONTENTTYPE_REFERENCE	3	ContentReference オブジェクト（リファレンスファイル文書であるコンテンツ）です。
DBR_CONTENTTYPE_OTHER	-1	上記以外のコンテンツを持つオブジェクトです。

3.3 リファレンスファイル文書の管理モデル

この節では，リファレンスファイル文書の管理モデルについて説明します。

リファレンスファイル文書を管理するための操作の詳細については，「6.8.12 リファレンスファイル文書の操作」を参照してください。

3.3.1 リファレンスファイル文書の管理の概要

ここでは，リファレンスファイル文書を管理するための，リファレンスファイル管理機能の概要について説明します。

リファレンスファイル管理機能とは，文書の実体であるコンテンツを任意のディレクトリに格納し，文書のプロパティとコンテンツロケーションだけをデータベースで管理する機能です。

この機能には，次のような特長があります。

- ディスク移行時など，コンテンツ格納ディレクトリを変更する場合，データベースに影響を与えないで，管理するコンテンツ格納先の基点となるディレクトリパス（**コンテンツ格納先ベースパス**）だけの変更で対応できます。
- データベース容量を削減できます。

リファレンスファイル管理機能では，バージョンなし文書およびバージョン付き文書をリファレンスファイル文書として管理できます。

コンテンツを任意のディレクトリに登録し，そのコンテンツの格納先を，コンテンツロケーションとしてデータベースに登録します。コンテンツロケーションには，ユーザが管理するコンテンツのコンテンツ格納先ベースパスを基点とする相対パス（**コンテンツ格納先パス**）を登録します。コンテンツは，DocumentBroker の機能で操作します。

リファレンスファイル管理機能を使用する場合，データベースとコンテンツの整合性を考慮する必要があります。このため，1 メソッド 1 トランザクションとすることを前提としています。文書作成を n 件連続で実行し，エラーが発生したときは，ロールバックしたあとにコンテンツだけが存在する状態となります。また，文書削除を n 件連続で実行し，エラーが発生したときは，ロールバックしたあとにコンテンツの存在しない文書が残ります。

なお，リファレンスファイル管理機能では，オブジェクトとコンテンツが不整合となる場合があります。不整合となる場合の内容を，次の表に示します。

表 3-8 オブジェクトとコンテンツが不整合となる場合

機能	エラー発生個所	不整合の内容	対処
文書の作成	コンテンツを登録したあとのエラー	登録コンテンツが残ります。	再度実行して文書を作成してください。
文書の更新	更新前のコンテンツを削除したあと，更新後のコンテンツを登録するまでのエラー	コンテンツの存在しないオブジェクトが残ります。	文書を削除したあと，更新後のコンテンツを使用して文書を作成してください。
	更新後のコンテンツを登録したあとのエラー	コンテンツの存在しないオブジェクトが残り，登録コンテンツが残ります。	文書を削除したあと，更新後のコンテンツを使用して文書を作成してください。
文書の削除	コンテンツを削除したあとのエラー	コンテンツの存在しないオブジェクトが残ります。	再度実行して文書を削除してください。

機能	エラー発生個所	不整合の内容	対処
バージョンのチェックアウト	コンテンツを複写したあとのエラー	複写したコンテンツが残ります。	チェックアウトを取り消し，コンテンツを削除したあと，ロールバックを実行してください。
バージョンの削除	コンテンツを削除したあとのエラー	コンテンツの存在しないオブジェクトが残ります。	再度実行してバージョンを削除してください。
チェックアウトの取り消し	コンテンツを削除したあとのエラー	コンテンツの存在しないオブジェクトが残ります。	再度実行してチェックアウトを取り消してください。

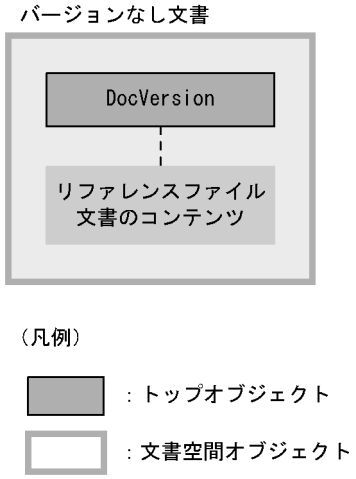
3.3.2 リファレンスファイル文書を構成する DMA オブジェクト

ここでは，リファレンスファイル文書を構成する DMA オブジェクトについて説明します。

リファレンスファイル管理機能では，バージョンなし文書およびバージョン付き文書をリファレンスファイル文書として管理できます。

リファレンスファイル文書を管理するバージョンなし文書を構成する DMA オブジェクトを次の図に示します。

図 3-13 リファレンスファイル文書を管理するバージョンなし文書を構成する DMA オブジェクト



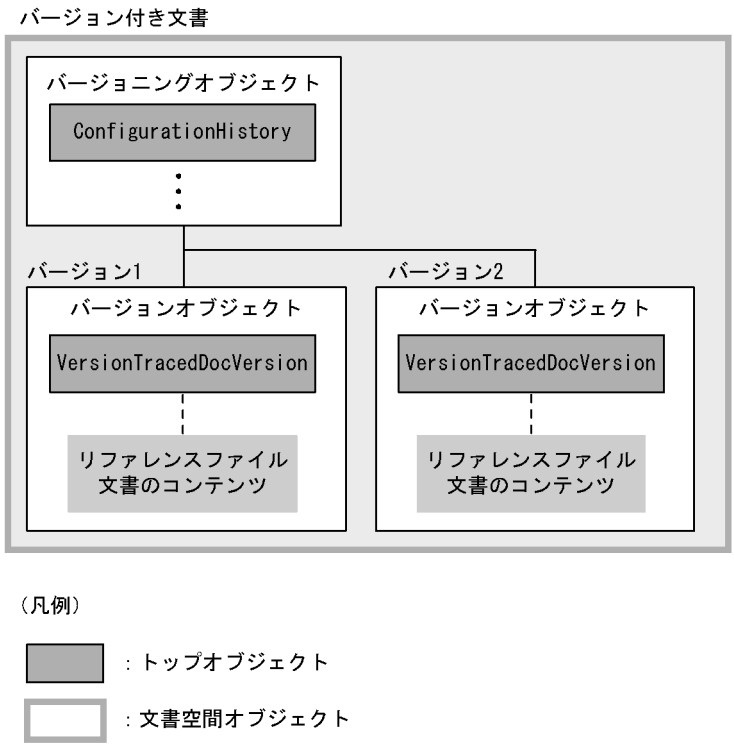
リファレンスファイル文書を管理するバージョンなし文書を構成する DMA オブジェクトについて説明します。

● DocVersion オブジェクト

バージョンなし文書のトップオブジェクトです。

次に，バージョン付き文書について説明します。リファレンスファイル文書を管理するバージョン付き文書を構成する DMA オブジェクトを次の図に示します。

図 3-14　リファレンスファイル文書を管理するバージョン付き文書を構成する DMA オブジェクト



リファレンスファイル文書を管理するバージョン付き文書を構成する DMA オブジェクトの詳細については,「3.1.2　バージョン付き文書を構成する DMA オブジェクト」を参照してください。

3.4 XML 文書の管理モデル

この節では，XML 文書の管理モデルについて説明します。

XML 文書は，ほかのレンディションタイプのファイルをコンテンツに持つ文書と同様に，バージョンなし文書またはバージョン付き文書として管理できます。バージョンなし文書およびバージョン付き文書の管理については，「3.1　バージョン管理モデル」および「3.2　レンディション管理モデル」を参照してください。

ここでは，XML 文書の管理で独自に使用できる機能について説明します。

XML 文書の管理で使用できるのは，次の二つの機能です。

● XML プロパティマッピング機能

● XML インデクスデータ作成機能

3.4.1 XML 文書の管理の概要

ここでは，XML 文書を管理するための XML プロパティマッピング機能および XML インデクスデータ作成機能について説明します。なお，XML 文書を管理するための操作の詳細については，「6.10　XML 文書を管理するための操作」を参照してください。

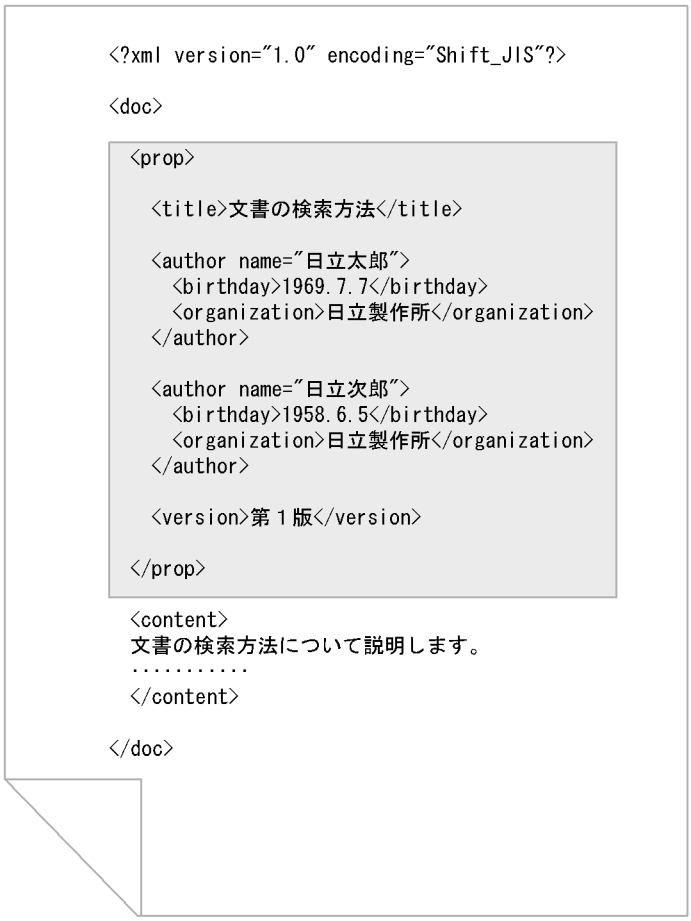
(1) XML プロパティマッピング機能

XML プロパティマッピング機能とは，XML 文書を登録する場合に，XML ファイル中のタグ間の文字列やタグの属性値を抽出して，XML 文書のプロパティに割り当てて設定する機能です。タグ間の文字列やタグの属性値を XML 文書のプロパティにマッピングすることによって，例えば，次のような操作ができるようになります。

- プロパティの一覧として情報を取得する
- 属性検索の対象にする
- 検索結果をソートするためのキーにする

XML プロパティマッピング機能を使用する XML ファイルの例を次の図に示します。

図 3-15　XML プロパティマッピング機能を使用する XML ファイルの例

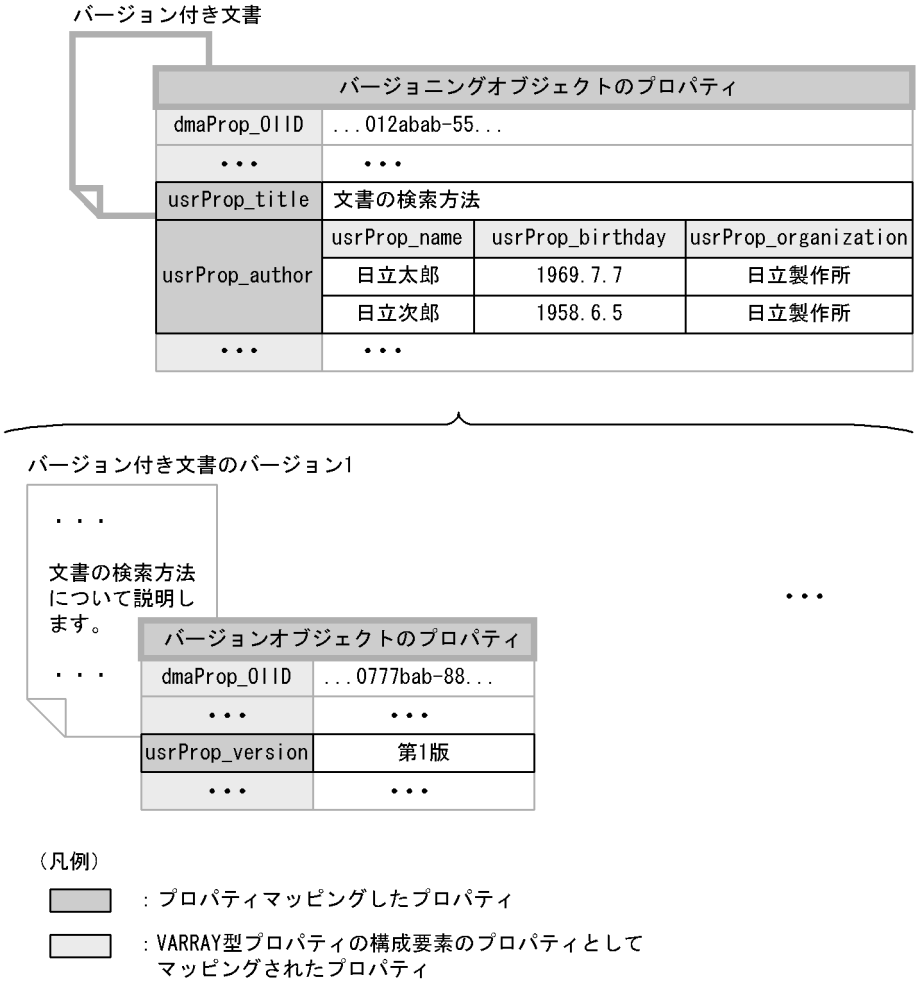


<prop> タグで囲まれた内容には，タイトルや著者，著者についての情報，バージョン情報などが，それぞれのタグ内の文字列または属性値として記述されています。

ここでは，XML ファイルをバージョン付き文書のコンテンツとして登録する場合について説明します。タイトルと著者の情報は，バージョニングオブジェクトのプロパティにマッピングします。また，バージョン情報は，バージョンオブジェクトのプロパティにマッピングします。

図 3-15 の XML ファイルに含まれる情報を XML 文書のプロパティにマッピングした例を次の図に示します。

図 3-16　XML プロパティマッピング機能によって XML 文書のプロパティにマッピングした例



この例では，<title> タグ間の文字列をバージョニングオブジェクトの usrProp_title プロパティにマッピングしています。<author> タグ間の文字列は複数の情報を表すタグを含むので，バージョニングオブジェクトの usrProp_author という VARRAY 型のプロパティの構成要素のプロパティにそれぞれマッピングしています。<version> タグ間の文字列はバージョンオブジェクトの usrProp_Version プロパティにマッピングしています。

なお，XML プロパティマッピング機能を使用する場合は，前提プログラムとして HiRDB Adapter for XML が必要です。また，登録する XML ファイルの論理構造を明確にして，どのタグをどのプロパティにマッピングするかという対応を定義しておく必要があります。定義に必要なファイルには，ユーザが作成するファイルと，DocumentBroker および HiRDB Adapter for XML によって作成されるファイルがあります。XML プロパティマッピング機能を使用するための定義については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

XML プロパティマッピング機能は，次の文字コードで記述された XML ファイルを対象に実行できます。

- Shift-JIS
- UTF-8
- UTF-16

ただし，XML プロパティマッピング機能でプロパティに設定される文字列は，Shift-JIS になります。

(2) XML インデクスデータ作成機能

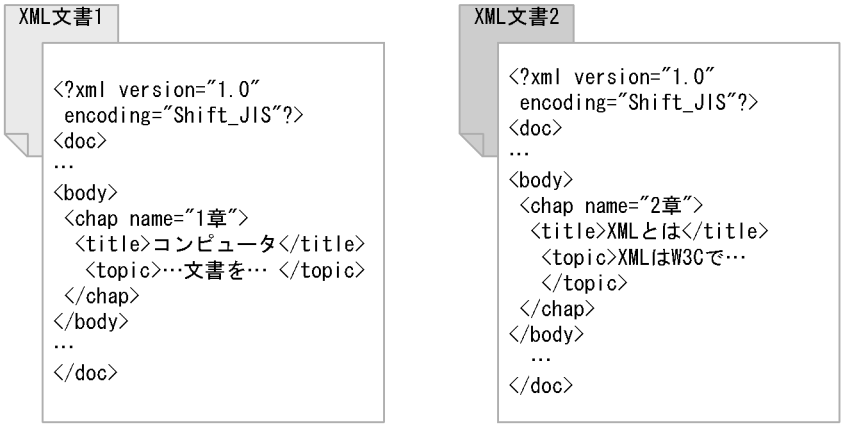
XML インデクスデータ作成機能とは，XML 文書の作成時や更新時に，XML ファイルの構文解析を実行して，インデクスデータを作成し，XML 文書の全文検索インデクスを登録する機能です。次の全文検索インデクスを登録できます。

- タグ情報などを削除したプレーンテキスト形式の全文検索インデクス
- 構造を指定した全文検索の実行に必要な構造指定検索用の全文検索インデクス

インデクスデータ作成時には，XML ファイルから不要なタグを削除したり，不要なタグ間のテキストをタグごと削除したりできます。この場合，削除するタグについての定義は，フィルタリング定義ファイルとして作成しておく必要があります。フィルタリング定義ファイルについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

XML インデクスデータ作成機能を使用して全文検索インデクスを登録した XML 文書は，構造指定検索の対象になります。構造指定検索について，次の図で説明します。

図 3-17　構造指定検索で使用する XML 文書の例



構造指定検索では，この例の XML 文書 1 および XML 文書 2 に対して，「構造 <chap> の下の構造 <topic> の中に『W3C』が含まれる文書を検索する」などの検索ができます。この例の場合，XML 文書 2 が検索結果として取得できます。また，「構造 <chap> に設定されている属性『name』の値が『1 章』である文書を検索する」というような場合にも使用できます。この例の場合，XML 文書 1 が検索結果として取得できます。このように，構造を指定した全文検索のことを構造指定検索といい，XML 文書の構造指定検索のことを **XML 構造指定検索**といいます。検索の詳細については，「4.　検索機能」を参照してください。

なお，XML インデクスデータ作成機能を使用する場合は，前提プログラムとして，HiRDB Adapter for XML および Preprocessing Library for Text Search が必要です。また，全文検索を実行する場合は，HiRDB Text Search Plug-in が必要です。XML インデクスデータは，登録する文書の情報に，XML 文書のアップロード情報を指定することで作成できます。

XML インデクスデータ作成機能は，次の文字コードで記述された XML ファイルを対象に実行できます。

- Shift-JIS
- UTF-8
- UTF-16

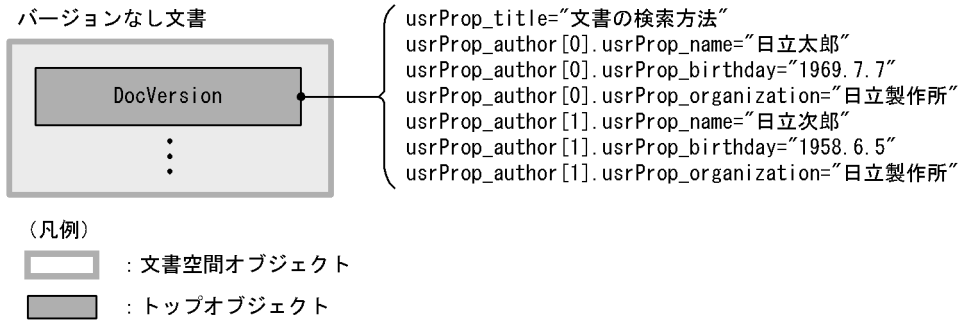
ただし，全文検索インデクスには Shift-JIS で登録されるため，全文検索の検索条件はすべて Shift-JIS で記述する必要があります。

3.4.2 XML 文書を構成する DMA オブジェクト

ここでは，XML 文書を構成する DMA オブジェクトについて説明します。

バージョンなし文書の場合，XML プロパティマッピング機能によって抽出した情報は，バージョンなし文書のトップオブジェクトである DocVersion オブジェクトのプロパティにマッピングできます。バージョンなし文書に対するプロパティマッピングの例を次の図に示します。

図 3-18 バージョンなし文書に対するプロパティマッピングの例

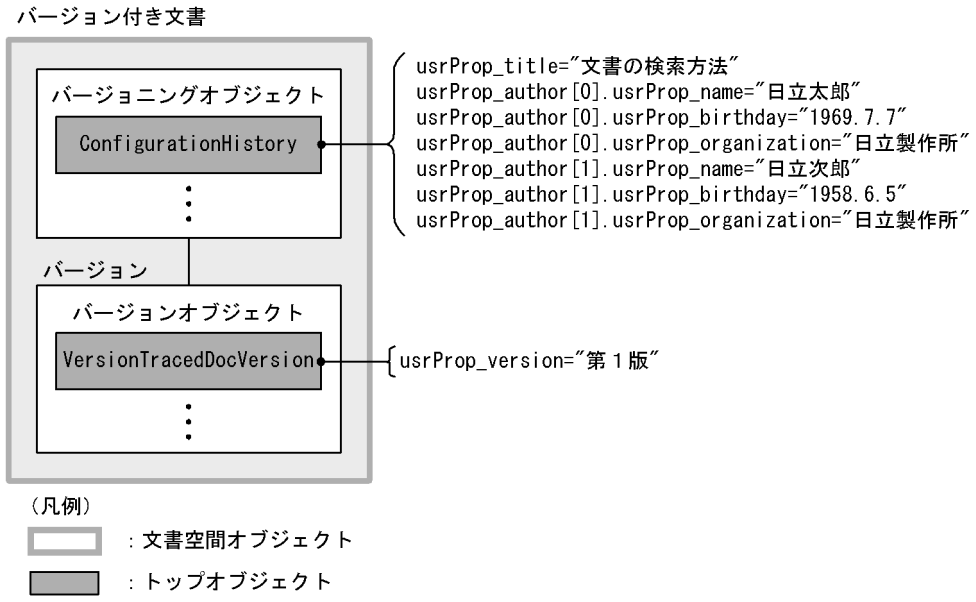


また，バージョンなし文書の場合，XML インデクスデータ作成機能で作成したインデクスデータを基に，トップオブジェクトである DocVersion オブジェクトの全文検索インデクス用プロパティが設定できます。

バージョン付き文書の場合，XML プロパティマッピング機能によって抽出した情報は，バージョニングオブジェクトのトップオブジェクトである ConfigurationHistory 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トップオブジェクトである VersionTracedDocVersion オブジェクト（または DocVersion オブジェクト）のプロパティにマッピングできます。

例えば，タイトルや著者などバージョン共通の情報を表すタグの情報は，バージョニングオブジェクトのプロパティにマッピングできます。バージョンや作成日などバージョンごとに異なる情報を表すタグの情報は，バージョンオブジェクトのプロパティにマッピングできます。バージョン付き文書に対するプロパティマッピングの例を次の図に示します。

図 3-19 バージョン付き文書に対するプロパティマッピングの例



また，バージョン付き文書の場合，XML インデクスデータ作成機能で作成したインデクスデータを基に，バージョンオブジェクトの全文検索インデクス用プロパティが設定できます。

3.4.3 Java クラスライブラリが管理できる XML 文書

ここでは，Java クラスライブラリが XML 文書のコンテンツとして管理できる XML ファイルについて説明します。

(1) XML ファイルの形式

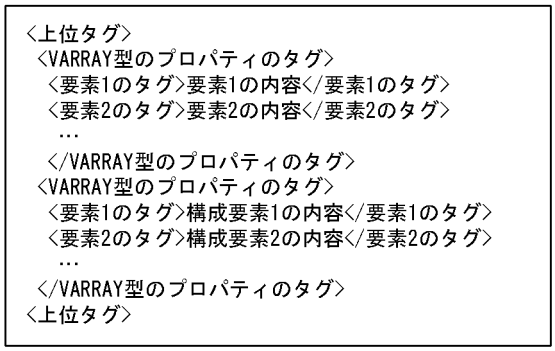
Java クラスライブラリでは，次のような XML ファイルを XML 文書のコンテンツとして管理できます。

- 外部参照を含まない，単一ファイルで構成された XML ファイル
- イメージファイルなど，構文解析の対象外の外部参照を含む XML ファイル
  ただし，この場合も，XML ファイルそのものは単一ファイルで構成される必要があります。

また，XML プロパティマッピング機能を使用する場合は，XML ファイルの構造は，次の条件を満たしている必要があります。

- XML 宣言が記載されている
- 一つのタグで文書全体が囲まれている
- プロパティにマッピングするタグが，1 度だけ出現する（VARRAY 型のプロパティにマッピングするタグは除く）
- VARRAY 型のプロパティにマッピングするタグの構造が，次の図に示す形式である

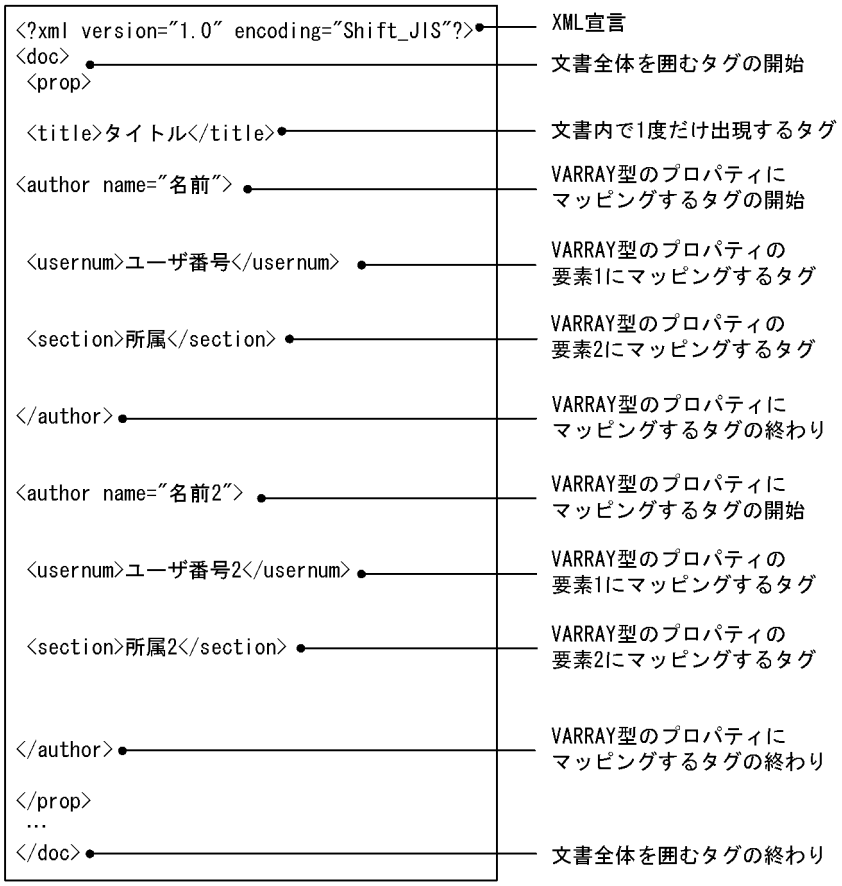
図 3-20 VARRAY 型のプロパティにマッピングできるタグの構造



(2) XML プロパティマッピング機能を使用できる XML ファイルの例

XML プロパティマッピング機能を使用できる XML ファイルの例を，次の図に示します。なお，それぞれのタグ名は任意です。

図 3-21 XML プロパティマッピング機能を使用できる XML ファイルの例



次に XML プロパティマッピング機能を使用できない XML ファイルの形式の例と，使用できる形式への変更例を説明します。

使用できない例 1：プロパティにマッピングするタグが複数回出現する XML ファイル

例えば，次の図のような XML ファイルです。

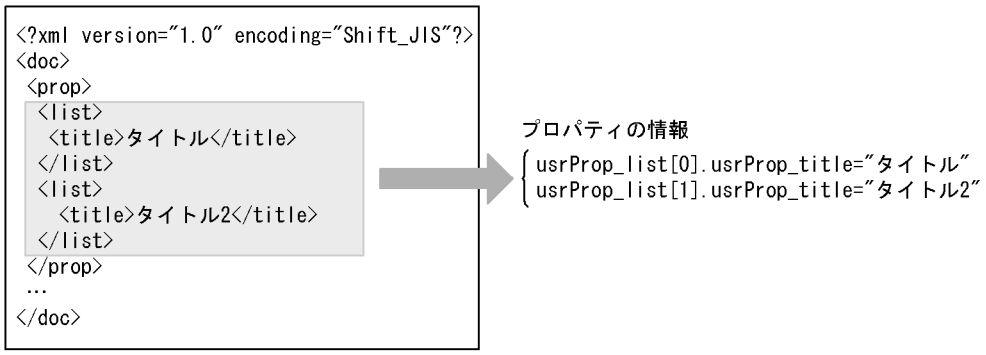
図 3-22 マッピングできない例 1：プロパティにマッピングするタグが複数回出現する XML ファイルの例

```
<?xml version="1.0" encoding="Shift_JIS"?>
<doc>
 <prop>
 <title>タイトル</title>
 <title>タイトル2</title>
 </prop>
 …
</doc>
```

例 1 の XML ファイルには，<title> タグで囲まれた情報が複数あるため，一つの値を取るプロパティにマッピングすることはできません。

複数回出現するタグの情報をプロパティにマッピングするためには，VARRAY 型のプロパティとしてマッピングする方法があります。例えば，次の図のように複数回出現するタグの前後に VARRAY 型のプロパティを表すタグを記述すれば，<title> タグの情報をプロパティにマッピングできます。

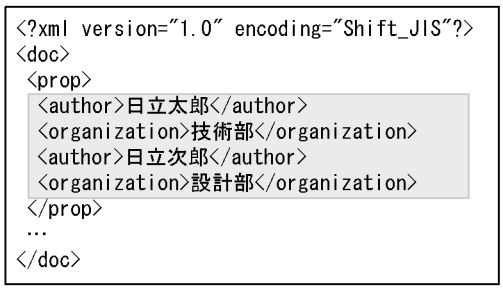
図 3-23 例 1 を登録できる構造に変更した例



使用できない例 2：要素が繰り返し出現する場合に VARRAY 型のプロパティを表すタグがない XML ファイル

例えば，次の図のような XML ファイルです。

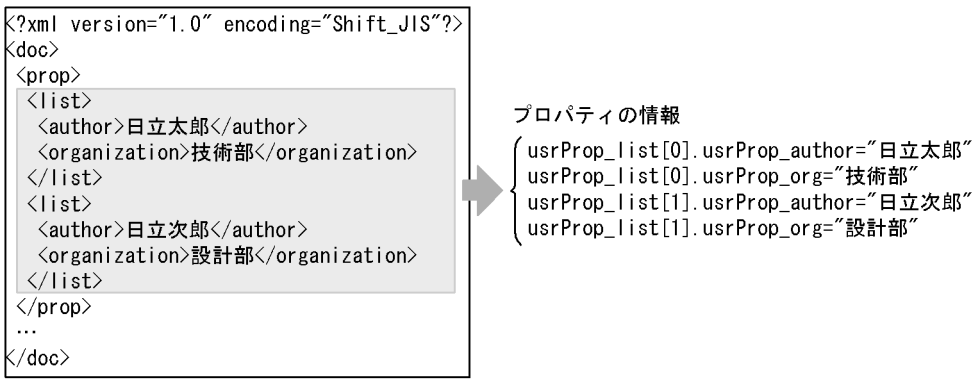
図 3-24 マッピングできない例 2：要素が繰り返し出現する場合に VARRAY 型のプロパティを表すタグがない XML ファイルの例



この例では，<author> タグと <organization> タグを繰り返していますが，出現順序が明確にならないため，VARRAY 型のプロパティにマッピングできません。

この場合，次の図のように VARRAY 型のプロパティを表す <list> タグを追加して構造を変更すれば，VARRAY 型のプロパティとして登録できます。

図 3-25 例 2 を登録できる構造に変更した例



3.5 リンクモデル

この節では，文書空間オブジェクト間を関連付けるリンクモデルについて説明します。

3.5.1 リンクによる文書管理の概要

Java クラスライブラリでは，リンクオブジェクトを使用して，複数の文書空間オブジェクト間をリンク付けたり，複数の文書空間オブジェクト間のリンクを一括して削除したりできます。リンクの一覧を取得して，関連付けられている文書空間オブジェクトを検索できます。また，リンクオブジェクトにはプロパティも設定できます。

リンクには次の 4 種類があります。

- 直接型リンク
- 参照型リンク
- 構成管理型リンク
- 文書間リンク

リンクモデルでは，リンクを設定するリンク元の文書空間オブジェクトを**リンク元オブジェクト**，リンクが設定されるリンク先の文書空間オブジェクトを**リンク先オブジェクト**といいます。なお，フォルダを使用したリンクの場合は，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。

リンクオブジェクトは，文書空間オブジェクトの作成時に，同時に既存の文書空間オブジェクトとのリンクを設定した場合に作成されます。また，既存の文書空間オブジェクトからほかの文書空間オブジェクトに対してリンクを設定した場合にも作成されます。リンクを解除すると，リンクオブジェクトは削除されます。また，リンクオブジェクトを削除することによって，リンクを解除することもできます。

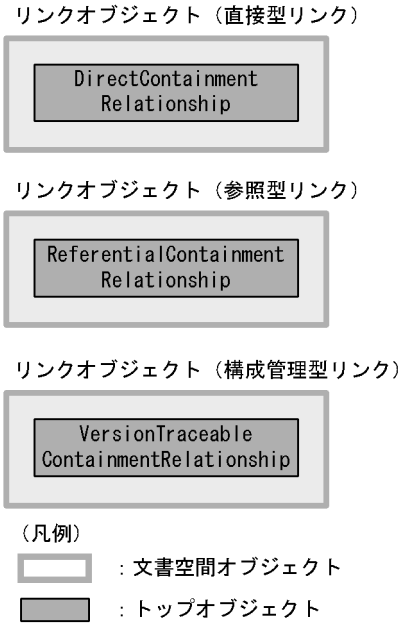
3.5.2 リンクオブジェクトを構成する DMA オブジェクト

ここでは，リンクオブジェクトを構成する DMA オブジェクトについて説明します。

(1) 直接型リンク，参照型リンクおよび構成管理型リンク

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，参照型リンクおよび構成管理型リンクについて説明します。直接型リンク，参照型リンクおよび構成管理型リンクを表すリンクオブジェクトを構成する DMA オブジェクトを次の図に示します。

図 3-26 直接型リンク，参照型リンクおよび構成管理型リンクを表すリンクオブジェクトを構成する DMA オブジェクト



各リンクオブジェクトを構成する DMA オブジェクトについて説明します。

● DirectContainmentRelationship **オブジェクト**
直接型リンクを表すリンクオブジェクトのトップオブジェクトです。トップオブジェクトクラスは，dmaClass_DirectContainmentRelationship クラスまたはそのサブクラスです。

● ReferentialContainmentRelationship **オブジェクト**
参照型リンクを表すリンクオブジェクトのトップオブジェクトです。トップオブジェクトクラスは，dmaClass_ReferentialContainmentRelationship クラスまたはそのサブクラスです。

● VersionTraceableContainmentRelationship **オブジェクト**
構成管理型リンクを表すリンクオブジェクトのトップオブジェクトです。トップオブジェクトクラスは，edmClass_VersionTraceableContainmentRelationship クラスまたはそのサブクラスです。

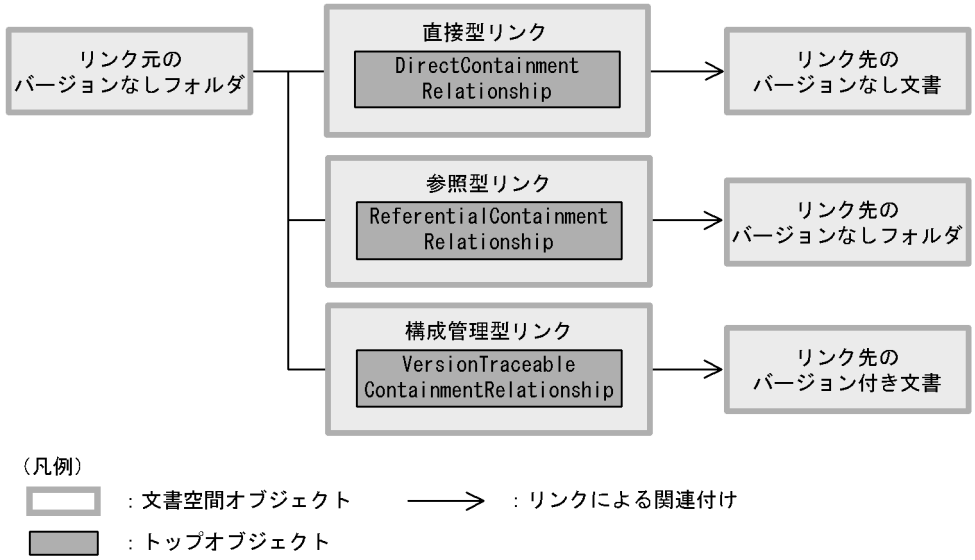
これらの DMA オブジェクトの識別には**リンク識別子**を使用します。上位オブジェクトまたは下位オブジェクトとの関連付けを表すリンクオブジェクトの一覧の取得時に，リンク識別子を取得できます。

これらのリンクオブジェクトは，文書またはフォルダ（下位オブジェクト）の作成時に，同時にリンク付ける既存のフォルダ（上位オブジェクト）を指定した場合に作成されます。また，既存のフォルダ（上位オブジェクト）からほかのフォルダまたは文書（下位オブジェクト）に対してリンクを設定した場合にも作成されます。リンクオブジェクトは，上位オブジェクトまたはリンクオブジェクト自身のインターフェースで操作します。また，リンクオブジェクトには，プロパティも設定できます。

上位オブジェクトまたは下位オブジェクトを削除すると，設定されていたリンクオブジェクトも削除されます。ただし，リンクの種類によって，リンクオブジェクトが削除されない場合もあります。詳細については，「3.7.3(4) リンクの解除，上位オブジェクトおよび下位オブジェクトの削除」を参照してください。

直接型リンクおよび参照型リンクは，フォルダと文書，フォルダとフォルダを関連付けます。構成管理型リンクは，フォルダとバージョン付きオブジェクトを関連付けます。直接型リンク，参照型リンクおよび構成管理型リンクの設定例を次の図に示します。

図 3-27 直接型リンク，参照型リンクおよび構成管理型リンクの設定例



この例では，直接型リンク，参照型リンクおよび構成管理型リンクを使用して，バージョンなしフォルダからほかの文書空間オブジェクトを関連付けています。

(2) 文書間リンク

ここでは，文書間を関連付ける場合の，文書間リンクについて説明します。文書間リンクを表すリンクオブジェクトを構成する DMA オブジェクトを次の図に示します。

図 3-28 文書間リンクを表すリンクオブジェクトを構成する DMA オブジェクト

文書間リンクを表すリンクオブジェクトを構成する DMA オブジェクトについて説明します。

● **Relationship オブジェクト**

文書間リンクを表すリンクオブジェクトのトップオブジェクトです。トップオブジェクトクラスは，edmClass_Relationship クラスです。この DMA オブジェクトの識別には**リンク識別子**を使用します。リンク元文書またはリンク先文書との関連付けを表すリンクオブジェクトの一覧の取得時に，リンク識別子を取得できます。

文書間リンクを表すリンクオブジェクトは，文書（リンク元文書）の作成時に，同時に既存の文書（リンク先文書）に対して文書間リンクを設定した場合に作成されます。また，既存の文書（リンク元文書）からほかの文書（リンク先文書）に対して文書間リンクを設定した場合にも作成されます。リンクオブジェクトは，リンク元文書またはリンクオブジェクト自身のインターフェースで操作します。また，リンクオブジェクトには，プロパティも設定できます。

なお，文書間リンクを表すリンクオブジェクトは，リンク元文書に従属する文書空間オブジェクトです。

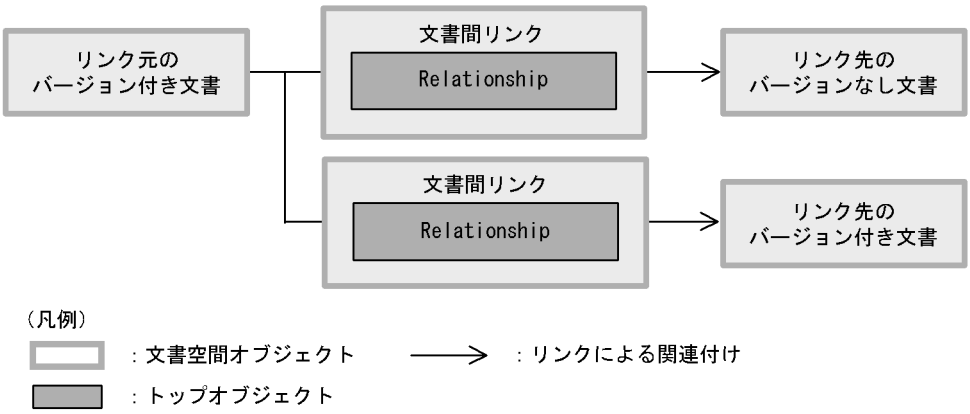
このため，リンク元文書を削除すると，設定されていたリンクオブジェクトも削除されますが，リンク先文書を削除しても，リンク元文書に設定されたリンクオブジェクトは削除されません。したがって，リンクを設定している文書を削除した場合は，リンク元文書からのリンクを解除することをお勧めします。

文書間リンクを表すリンクオブジェクトは，文書を表す次の文書空間オブジェクト間の関連付けに使用できます。また，バージョン付き文書の１バージョンに対してリンクを設定することもできます。

- バージョンなし文書
- バージョン付き文書

文書間リンクの設定例を次の図に示します。

図 3-29　文書間リンクの設定例



この例では，文書間リンクを使用して，バージョン付き文書からほかの文書を関連付けています。

3.5.3　リンクによる文書管理

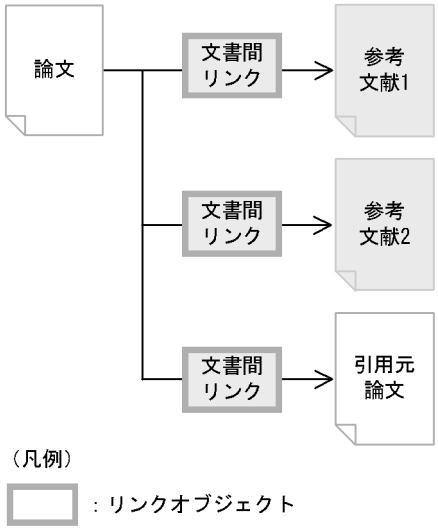
ここでは，文書間リンクを設定した文書管理について説明します。直接型リンク，参照型リンクおよび構成管理型リンクを設定した文書管理については，「3.6　バージョンなしフォルダによる管理モデル」および「3.7　バージョン付きフォルダによる管理モデル」を参照してください。

なお，リンクを管理するための操作の詳細については，「6.8.13　リンクの操作」を参照してください。

（1）文書間リンクの設定

文書間リンクを設定した文書の例を次の図に示します。

図 3-30 文書間リンクを設定した文書の例



この例では，「論文」をリンク元文書，「参考文献 1」，「参考文献 2」および「引用元論文」をリンク先文書として文書間リンクを設定しています。

なお，設定したリンクオブジェクトは，リンク元文書が管理します。したがって，リンクオブジェクトのプロパティを設定したり，リンクを解除したりする場合は，リンク元文書またはリンクオブジェクト自身のインターフェースで操作します。

(2) 文書間リンクをたどる参照

文書間リンクによって関連付けられた文書は，双方向に参照できます。例えば，図 3-30 の場合に，リンク元文書「論文」からリンク先文書「参考文献 1」，「参考文献 2」および「引用元論文」の一覧を取得して参照できます。また，リンク先文書「参考文献 1」からリンク元文書「論文」を参照できます。

リンク元文書またはリンク先文書を参照する場合は，リンク元文書またはリンク先文書との関連付けを表すリンクオブジェクトの一覧を取得します。リンクオブジェクトをたどってリンク元文書またはリンク先文書を参照します。また，リンクオブジェクト取得時には，次の情報も取得できます。

- リンク元文書またはリンク先文書のオブジェクト種別
- リンク識別子
- リンク元文書またはリンク先文書のプロパティ
- リンクオブジェクトのプロパティ

また，一覧取得時には，すでにリンク先文書が削除されているリンクオブジェクトも取得できます。

(3) リンクの解除，リンク元文書およびリンク先文書の削除

ここでは，リンクの解除と，リンク元文書およびリンク先文書の削除の関係について説明します。

(a) リンクの解除

リンク元文書からリンクを解除するか，リンクオブジェクト自身のインターフェースでリンクオブジェクトを削除します。

(b) リンク元文書の削除

リンク元文書を削除すると，設定されていたリンクオブジェクトも削除されます。また，バージョン付き文書の 1 バージョンに当たるバージョンなし文書がリンク元文書の場合に，バージョン付き文書からその

バージョンを削除したときも，バージョンなし文書に設定していたリンクは解除されて，リンクオブジェクトも削除されます。ただし，リンク先文書は削除されません。

(c) リンク先文書の削除

リンク先文書を削除した場合，設定されていたリンクオブジェクトは削除されません。したがって，リンク元文書には，リンク先文書が削除されているリンクオブジェクトも残ります。この場合，リンク元文書からリンクオブジェクトを削除してください。

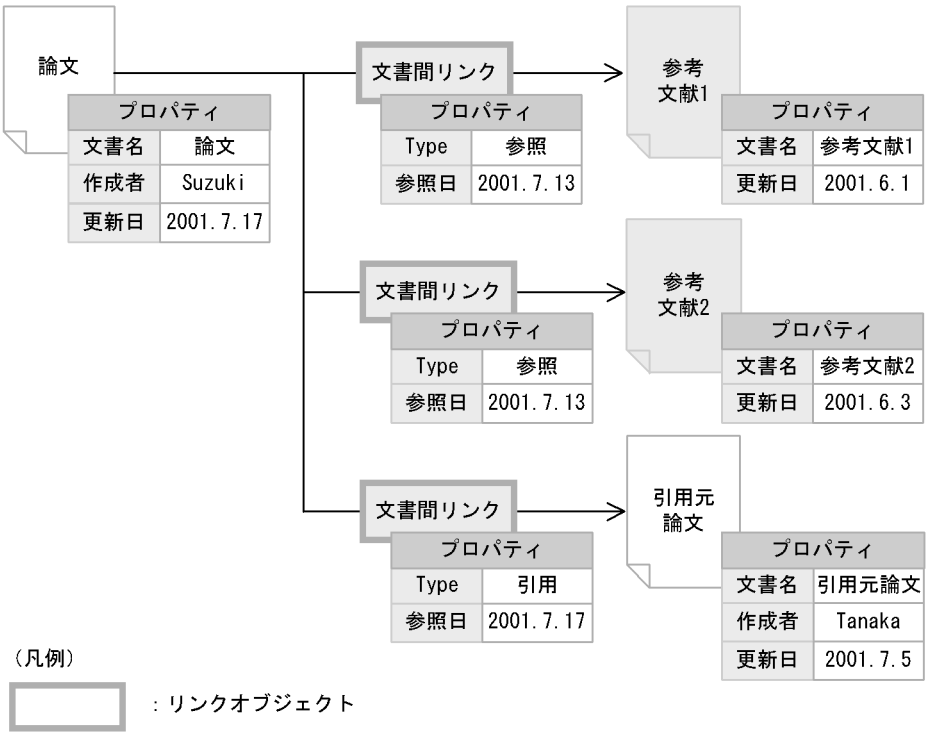
リンク先文書との関連付けを表すリンクオブジェクトの一覧を取得すれば，リンク先文書が削除されていないかを確認できます。

(4) 文書間リンクのプロパティの設定

リンク元文書，リンク先文書および文書のリンクには，ユーザ定義プロパティを設定して管理できます。リンクのプロパティとは，リンクオブジェクトに設定するプロパティです。リンクオブジェクトのプロパティは，リンク元文書やリンク先文書を分類したり，リンク元文書やリンク先文書のプロパティと組み合わせた管理に使用したりできます。リンクオブジェクトのプロパティは，リンク元文書またはリンクオブジェクト自身のインターフェースを使用して設定します。また，リンクオブジェクトのプロパティは検索に使用できます。検索では，ユーザ定義プロパティのほか，文書間リンクを表す DMA の Relationship オブジェクトのプロパティとして設定されている，DMA が規定したプロパティ（dmaProp_ または edmProp_ で始まる識別子を持つプロパティ）も使用できます。

プロパティを設定した文書間リンクの例を次の図に示します。

図 3-31 プロパティを設定した文書間リンクの例



この例では，リンクオブジェクトにユーザ定義プロパティとして，「Type」と「参照日」を設定しています。例えば，リンクオブジェクトのプロパティ「Type」によって，リンク先文書を「参照」と「引用」に分類したりできます。また，リンクオブジェクトのプロパティ「参照日」とリンク先文書のプロパティ「更新日」を比較して，参照後に文書が更新されたかどうかを確認したりできます。また，これらのプロパ

ティを使用して,「リンクオブジェクトの『Type』が『参照』であり,『文書名』が『論文』である文書を検索する」というような edmSQL 検索を実行することもできます。edmSQL 検索については,「4. 検索機能」を参照してください。edmSQL 検索に指定する edmSQL の文法については,「5. edmSQL の文法」を参照してください。

3.6 バージョンなしフォルダによる管理モデル

この節では，バージョンなしフォルダによる管理モデルについて説明します。

3.6.1 バージョンなしフォルダによる文書管理の概要

バージョンなしフォルダでは，複数の文書を目的に応じて一つにまとめたり，一つの文書を複数の分類に関連付けたりして文書を管理できます。また，バージョンなしフォルダを別のバージョンなしフォルダに関連付けて管理できます。

フォルダによる文書管理では，二つの文書空間オブジェクトを想定して，一方を「包含するオブジェクト」，もう一方を「包含されるオブジェクト」として考えます。このマニュアルでは，オブジェクトを包含するオブジェクトを**上位オブジェクト**，オブジェクトに包含されるオブジェクトを**下位オブジェクト**といいます。

例えば，バージョンなしフォルダに文書を格納することを考えた場合，バージョンなしフォルダが上位オブジェクト，文書が下位オブジェクトに相当します。また，バージョンなしフォルダの下位にバージョンなしフォルダを作成することを考えた場合，上位のバージョンなしフォルダが上位オブジェクト，下位のバージョンなしフォルダが下位オブジェクトに相当します。フォルダによる文書管理は，このような上位オブジェクトと下位オブジェクトとのリンク関係で表せます。リンクを解除すると，二つの文書空間オブジェクトの関連付けがなくなります。

バージョンなしフォルダでは，次の 3 種類のリンクで上位オブジェクトと下位オブジェクトをリンク付けます。

直接型リンク
上位オブジェクトと下位オブジェクトを 1:n（n は 1 以上の整数）の関係でリンク付けて，文書空間オブジェクトをまとめて管理できます。
なお，上位のコンテナを下位以下のコンテナの下位のオブジェクトとして関連づけることはできません。

参照型リンク
上位オブジェクトと下位オブジェクトを n:m（n，m は 1 以上の整数）の関係でリンク付けて，文書空間オブジェクトを分類して管理できます。

構成管理型リンク
下位オブジェクトがバージョン付きオブジェクトの場合，バージョン付きオブジェクトのバージョンをトレースして管理できます。構成管理型リンクについては，「3.7　バージョン付きフォルダによる管理モデル」を参照してください。

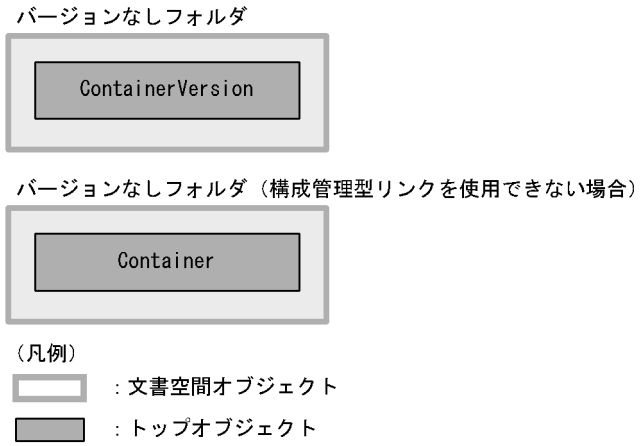
バージョンなしフォルダは，バージョン付きフォルダの 1 バージョンとして管理することもできます。バージョン付きフォルダの 1 バージョンとして管理するバージョンなしフォルダについては，「3.7　バージョン付きフォルダによる管理モデル」を参照してください。

なお，バージョンなしフォルダの直接型リンク，参照型リンクおよび構成管理型リンクでは，文書空間オブジェクト間をリンクオブジェクトによってリンク付けます。リンクオブジェクトについては，「3.5　リンクモデル」を参照してください。

3.6.2 バージョンなしフォルダを構成する DMA オブジェクト

ここでは，バージョンなしフォルダ構成する DMA オブジェクトについて説明します。バージョンなしフォルダを構成する DMA オブジェクトを次の図に示します。

図 3-32　バージョンなしフォルダを構成する DMA オブジェクト



バージョンなしフォルダのトップオブジェクトクラスは，dmaClass_Container クラスまたはそのサブクラスです。edmClass_ContainerVersion クラスまたはそのサブクラスを指定した場合，そのバージョンなしフォルダは構成管理型リンクを使用できます。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指定した場合，そのバージョンなしフォルダは構成管理型リンクを使用できません。

バージョンなしフォルダを構成する DMA オブジェクトについて説明します。

● ContainerVersion **オブジェクト**
バージョンなしフォルダのトップオブジェクトです（トップオブジェクトクラスは，edmClass_ContainerVersion クラスまたはそのサブクラス）。この DMA オブジェクトで構成されるバージョンなしフォルダは，構成管理型リンクを使用できます。

● Container **オブジェクト**
バージョンなしフォルダのトップオブジェクトです（トップオブジェクトクラスは，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）。この DMA オブジェクトで構成されるバージョンなしフォルダは，構成管理型リンクを使用できません。

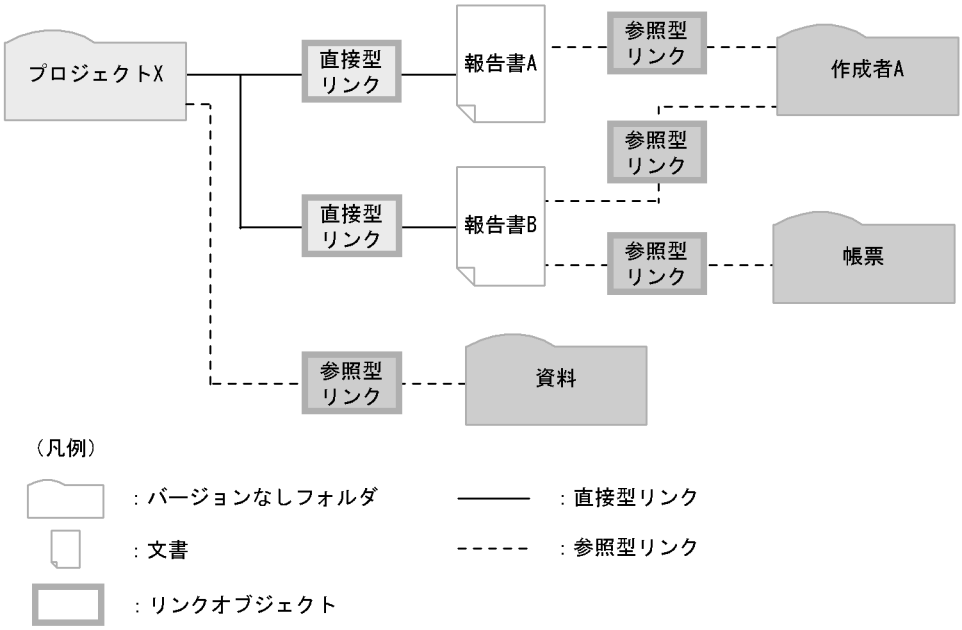
3.6.3　バージョンなしフォルダによる文書管理

ここでは，バージョンなしフォルダによる文書管理について説明します。なお，バージョンなしフォルダのリンクを管理するための操作の詳細については，「6.8.13　リンクの操作」を参照してください。

(1) リンクの設定

バージョンなしフォルダから文書およびフォルダに，直接型リンクおよび参照型リンクを設定した例を次の図に示します。なお，構成管理型リンクについては，「3.7　バージョン付きフォルダによる管理モデル」を参照してください。

図 3-33 バージョンなしフォルダによるリンクの設定例



この例では，バージョンなしフォルダ「プロジェクト X」は，直接型リンクで文書をリンク付け，参照型リンクでフォルダをリンク付けています。直接型リンクでは，上位オブジェクトと下位オブジェクトの関係は 1:n になるため，例えば，文書「報告書 A」または「報告書 B」は，フォルダ「プロジェクト X」と別のフォルダに対して，直接型リンクで同時にリンク付けることはできません。このため，文書「報告書 A」または「報告書 B」は，参照型リンクを使用して，フォルダ「作成者 A」または「帳票」にリンク付けています。

(2) リンクをたどる文書空間オブジェクトの参照

リンクによって関連付けられた上位オブジェクトと下位オブジェクトは，双方向に参照できます。例えば，図 3-33 の場合は，フォルダ「プロジェクト X」からリンクを設定している文書「報告書 A」，「報告書 B」およびフォルダ「資料」を参照できます。また，文書「報告書 A」からフォルダ「プロジェクト X」および「作成者 A」を参照できます。

上位オブジェクトまたは下位オブジェクトを参照する場合は，上位オブジェクトまたは下位オブジェクトとの関連付けを表すリンクオブジェクトの一覧を取得します。リンクオブジェクトをたどって上位オブジェクトまたは下位オブジェクトを参照します。また，リンクオブジェクト取得時には，次の情報も取得できます。

- 上位オブジェクトまたは下位オブジェクトのオブジェクト種別
- リンク識別子
- 上位オブジェクトまたは下位オブジェクトのプロパティ
- リンクオブジェクトのプロパティ

(3) リンクの解除，上位オブジェクトおよび下位オブジェクトの削除

ここでは，リンクの解除と，上位オブジェクトおよび下位オブジェクトの削除の関係について説明します。

(a) リンクの解除

上位オブジェクトからリンクを解除するか，リンクオブジェクト自身のインターフェースでリンクオブジェクトを削除します。

(b) 上位オブジェクトの削除

上位オブジェクトを削除すると，設定されていたリンクオブジェクトも削除されます。

(c) 下位オブジェクトの削除

下位オブジェクトを削除すると，設定されていたリンクオブジェクトも削除されます。ただし，リンクの種類によって，リンクオブジェクトが削除されない場合もあります。詳細については，「3.7.3(4) リンクの解除，上位オブジェクトおよび下位オブジェクトの削除」を参照してください。

(4) バージョンなしフォルダのプロパティの設定

バージョンなしフォルダおよびバージョンなしフォルダのリンクには，ユーザ定義プロパティを設定して管理できます。リンクのプロパティとは，リンクオブジェクトに設定するプロパティです。フォルダのプロパティとして「作成者」などを設定しておくと，フォルダの検索で使用できます。また，リンクオブジェクトのプロパティは，検索に使用したり，重要度などの値に応じてソートして，バージョンなしフォルダの下位オブジェクトに順序性を持たせたりできるので便利です。リンクオブジェクトのプロパティは，バージョンなしフォルダを指定して，上位オブジェクトまたはリンクオブジェクト自身のインターフェースを使用して設定します。

3.7 バージョン付きフォルダによる管理モデル

この節では，バージョン付きフォルダによる管理モデルについて説明します。

バージョン付きフォルダを管理する場合，バージョン付きフォルダの管理モデルとバージョン管理モデルを組み合わせて使用できます。バージョン管理モデルについては，「3.1　バージョン管理モデル」を参照してください。

3.7.1 バージョン付きフォルダによる文書管理の概要

バージョン付きフォルダでは，バージョン付きフォルダ自身のバージョンを管理して，バージョン付きフォルダのバージョンと下位オブジェクトとをリンク付けたままバージョンアップできます。また，バージョン付きフォルダでは，次の 3 種類のリンクで上位オブジェクトと下位オブジェクトをリンク付けます。

- **直接型リンク**
- **参照型リンク**
- **構成管理型リンク**

構成管理型リンクは，下位にリンク付けているバージョン付きオブジェクトのバージョンをトレースするリンクです。

なお，構成管理型リンクでは，下位にリンク付けてバージョン付きフォルダによって包含される下位オブジェクトを**構成要素**といいます。

構成管理型リンクには，FLOATING モードと FIX モードという 2 種類のモードがあります。この 2 種類のモードを**構成管理モード**といいます。

- FLOATING **モード**
  バージョン付きオブジェクトのバージョンが追加されるたびに，最新バージョンをトレースするモードです。このモードを設定すると，構成要素がバージョンアップするたびに，最新バージョンの構成要素にリンクをつなぎ替えます。
- FIX **モード**
  バージョン付きオブジェクトのバージョンを固定してリンク付けるモードです。このモードを設定すると，構成要素がバージョンアップしても，リンクは特定のバージョンに固定されます。

このように，バージョン付きフォルダは，構成要素であるバージョン付きオブジェクトのバージョン単位でリンク付けることができます。

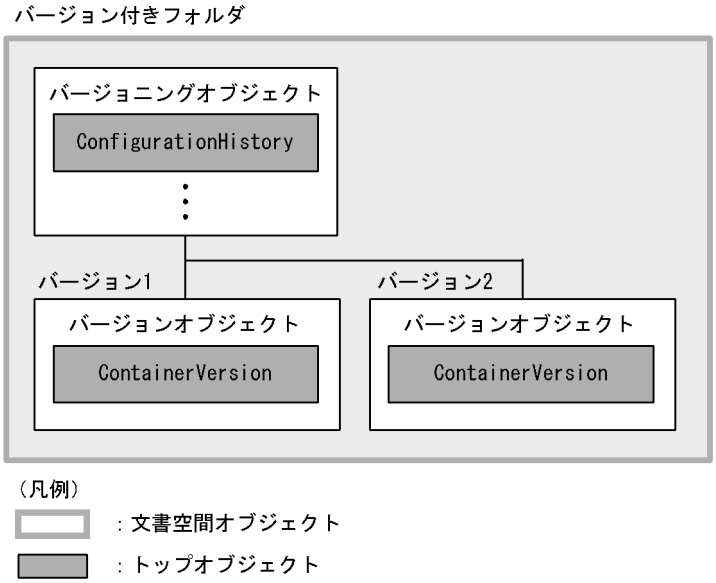
また，バージョン付きフォルダでは，バージョン付きフォルダの個々のバージョンとしてバージョンなしフォルダを管理できます。詳細については，「3.1.1(4) バージョン付きオブジェクトとバージョンなしオブジェクトの関係」を参照してください。

なお，バージョン付きフォルダの直接型リンク，参照型リンクおよび構成管理型リンクでは，文書空間オブジェクト間をリンクオブジェクトによってリンク付けます。リンクオブジェクトについては，「3.5　リンクモデル」を参照してください。

3.7.2 バージョン付きフォルダを構成する DMA オブジェクト

ここでは，バージョン付きフォルダを構成する DMA オブジェクトについて説明します。バージョン付きフォルダを構成する DMA オブジェクトを次の図に示します。

図 3-34　バージョン付きフォルダを構成する DMA オブジェクト



バージョン付きフォルダの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は，dmaClass_ConfigurationHistory クラスまたはそのサブクラスです。

バージョン付きフォルダ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は，dmaClass_ContainerVersion クラスまたはそのサブクラスです。

バージョン付きフォルダを構成する DMA オブジェクトについて説明します。

● ConfigurationHistory **オブジェクト**

バージョン付きフォルダのバージョニングオブジェクトのトップオブジェクトです。
この DMA オブジェクトのプロパティがバージョン付きフォルダクラスのデフォルトのプロパティとなり，バージョニングオブジェクトのプロパティとして使用できます。また，この DMA オブジェクトの OIID がバージョン付きフォルダの OIID となります。

● ContainerVersion **オブジェクト**

バージョン付きフォルダのバージョンオブジェクトのトップオブジェクトです。
この DMA 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として使用できます。
バージョン付きフォルダの個々のバージョンの識別には，この DMA オブジェクトの OIID ではなく，**バージョン識別子**を使用します。

3.7.3　バージョン付きフォルダによる文書管理

ここでは，バージョン付きフォルダによる文書管理について説明します。なお，バージョン付きフォルダを管理するための操作の詳細については，「6.8.10　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。バージョン付きフォルダのリンクを管理するための操作の詳細については，「6.8.13　リンクの操作」を参照してください。

(1) バージョン付きフォルダの構成管理機能

ここでは，バージョン付きフォルダの構成管理機能について説明します。

- バージョン付きフォルダの構成要素のバージョンアップ
- バージョン付きフォルダの構成要素のバージョンの確定

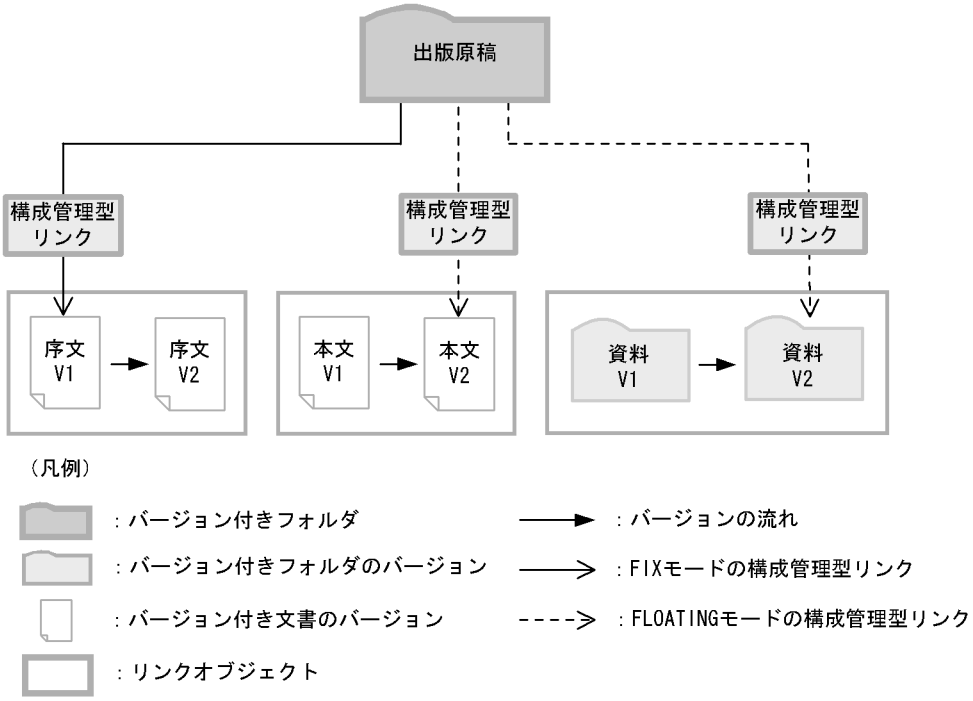
- バージョン付きフォルダのバージョンアップ
- バージョン付きフォルダのバージョンアップと構成要素および構成管理モードの関係

各内容について説明します。

(a) バージョン付きフォルダの構成要素のバージョンアップ

バージョン付きフォルダの構成要素がバージョンアップした場合，設定している構成管理モードによって，バージョン付きフォルダが管理する構成要素のバージョンが異なります。バージョン付きフォルダの構成要素のバージョンアップの例を次の図に示します。

図 3-35 バージョン付きフォルダの構成要素のバージョンアップの例

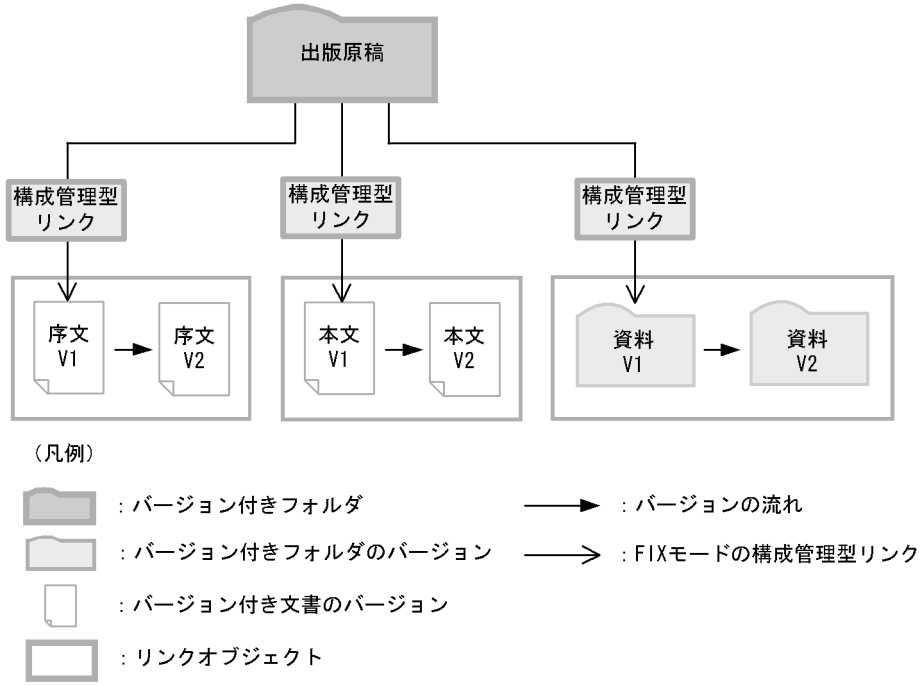


この例では，バージョン付きフォルダ「出版原稿」は FIX モードと FLOATING モードを使用して構成要素を管理しています。バージョン付きフォルダ「出版原稿」では，バージョン付き文書「序文」は FIX モードでリンク付けているため，「序文」のバージョンが追加されても，バージョン 1 の「序文」を管理しています。一方，バージョン付き文書「本文」およびバージョン付きフォルダ「資料」は FLOATING モードでリンク付けているため，最新バージョンであるバージョン 2 の「本文」および「資料」にリンクをつなぎ替えて管理しています。

(b) バージョン付きフォルダの構成要素のバージョンの確定

バージョン付きフォルダの構成要素の編集が終了した場合など，バージョン付きフォルダで管理する構成要素のバージョンを確定したいときは，構成管理モードを FLOATING モードから FIX モードに変更します。バージョンを確定したあとで構成要素のバージョンが追加されても，バージョン付きフォルダは確定したバージョンの構成要素を管理します。バージョン付きフォルダの構成要素のバージョンを確定した例を次の図に示します。

図 3-36 バージョン付きフォルダの構成要素のバージョンを確定した例

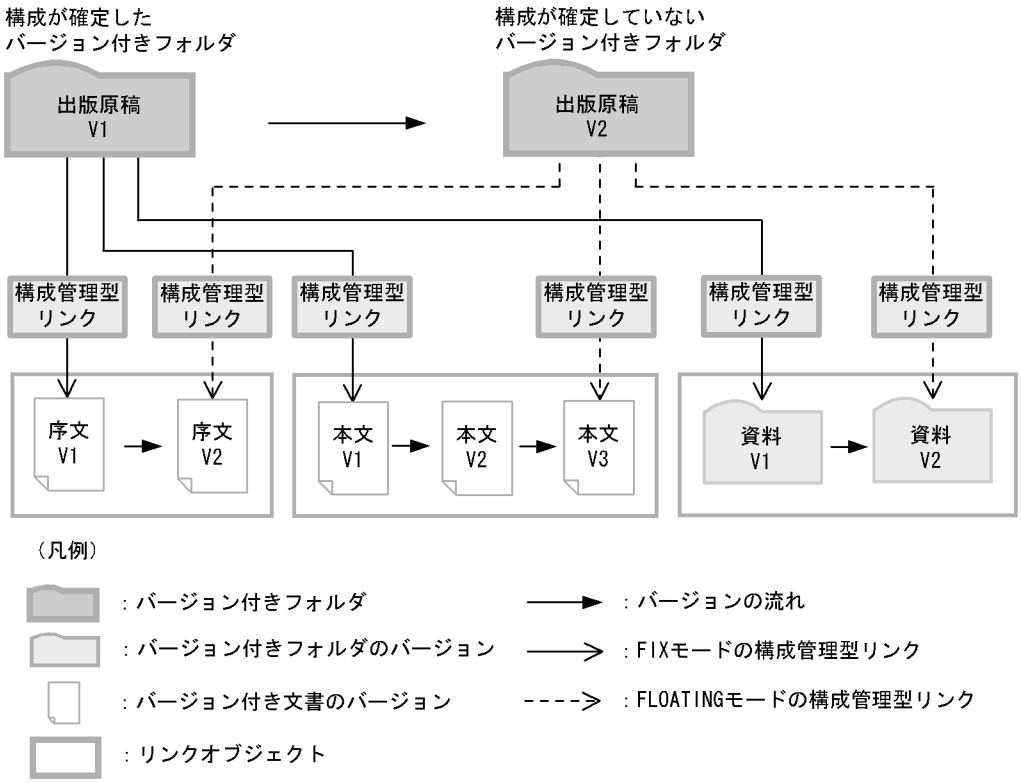


この例では，バージョン付きフォルダ「出版原稿」は，バージョン 1 で編集が終了したバージョン付き文書「序文」,「本文」およびバージョン付きフォルダ「資料」を FIX モードで管理しています。このため，各構成要素のバージョンが追加されても，バージョン付きフォルダ「出版原稿」が管理するのは最新バージョンであるバージョン 2 の構成要素ではなく，バージョン 1 の構成要素になります。

(c) バージョン付きフォルダのバージョンアップ

バージョン付きフォルダでは，バージョン付きフォルダのバージョンごとに構成要素のバージョンを管理できます。例えば，バージョン付きフォルダの構成要素を確定したあとでさらに構成要素を編集したい場合，バージョン付きフォルダをバージョンアップして，確定したバージョンを管理するバージョン付きフォルダとは別に，最新バージョンを管理するバージョン付きフォルダを新しく作成できます。バージョン付きフォルダのバージョンアップの例を次の図に示します。

図 3-37 バージョン付きフォルダのバージョンアップの例



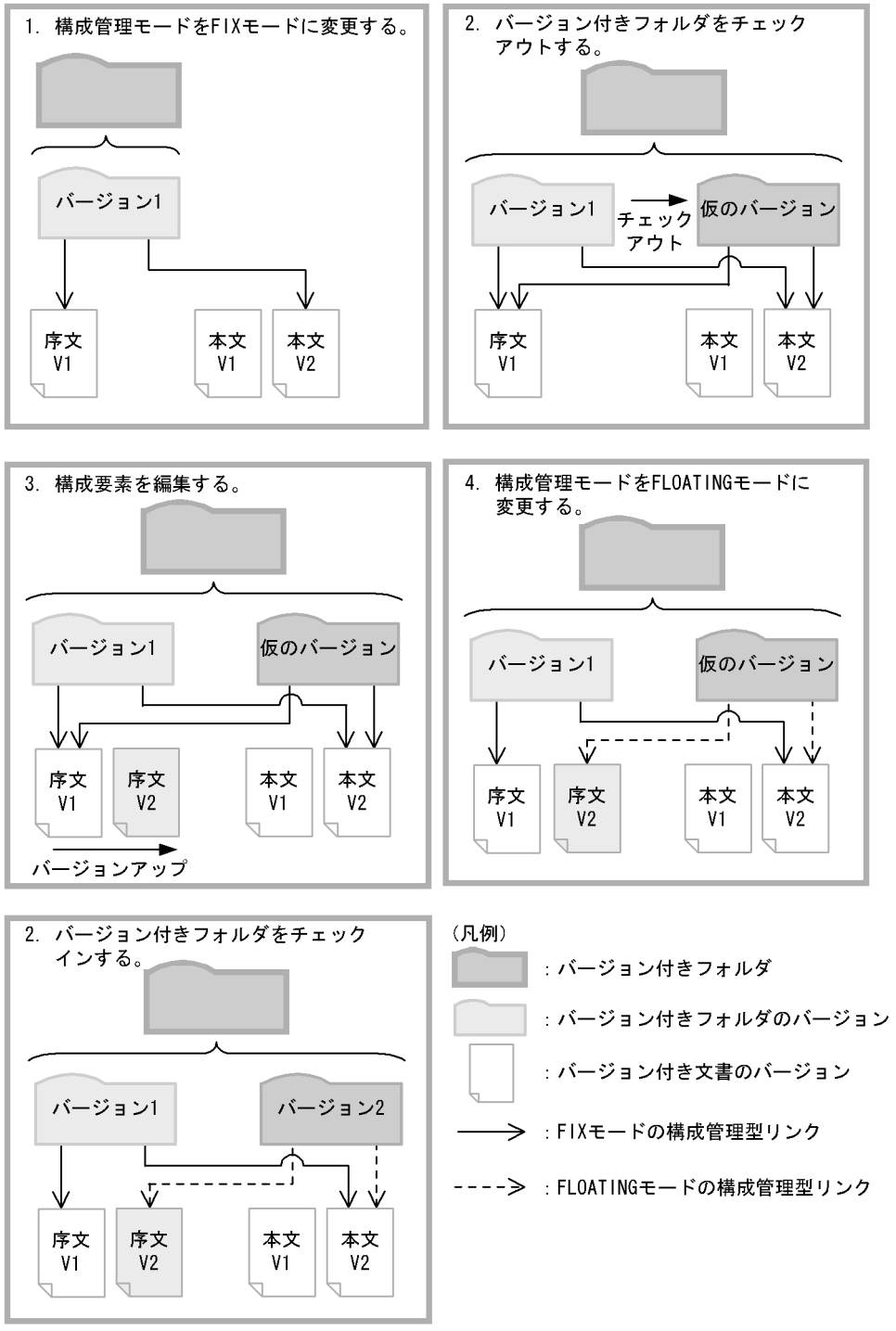
この例では，バージョン付きフォルダ「出版原稿」のバージョンごとに，構成要素のバージョンを管理しています。バージョン 1 のバージョン付きフォルダ「出版原稿」では，バージョン付き文書「序文」,「本文」およびバージョン付きフォルダ「資料」を FIX モードでリンク付けているため，各構成要素がバージョンアップしても，各構成要素のバージョン 1 を管理しています。一方，バージョン 2 のバージョン付きフォルダ「出版原稿」では，バージョン付き文書「序文」,「本文」およびバージョン付きフォルダ「資料」を FLOATING モードでリンク付けているため，各構成要素の最新バージョン（「序文」のバージョン 2,「本文」のバージョン 3,「資料」のバージョン 2）を管理しています。

(d) バージョン付きフォルダのバージョンアップと構成要素および構成管理モードの関係

バージョン付きフォルダをバージョンアップする場合，最新バージョンに対応する，バージョンなしフォルダがコピーされます。このとき，構成要素はコピーされません。また，バージョンアップするバージョン付きフォルダにリンクが設定されている場合，参照型リンクを表すリンクオブジェクトおよび構成管理型リンクを表すリンクオブジェクトもコピーされます。このため，参照型リンクおよび構成管理型リンクでリンク付けられる構成要素は，バージョンアップしたバージョン付きフォルダでも管理できます。ただし，直接型リンクを表すリンクオブジェクトはコピーされません。したがって，直接型リンクでリンク付けられる構成要素は，バージョンアップしたバージョン付きフォルダでは管理できません。

バージョン付きフォルダをバージョンアップするには，文書のバージョンアップと同様に，チェックアウトおよびチェックインを実行します。このとき，構成要素を管理している構成管理モードを適切に変更することによって，バージョン付きフォルダごとに構成要素のバージョンの履歴管理ができるようになります。バージョン付きフォルダのバージョンごとに構成要素の履歴を管理する場合，次の図に示す手順でバージョンアップしてください。

図 3-38 バージョン付きフォルダのバージョンごとの構成要素の履歴管理の手順



バージョン付きフォルダのバージョンごとに構成要素の履歴を管理する場合の注意事項を次に示します。

注意事項

バージョンアップ後のバージョン付きフォルダの構成要素を編集する場合，バージョンアップ前のバージョン付きフォルダの構成管理モードを FIX モードに変更したあとで，構成要素を編集してください。構成管理モードが FLOATING モードの状態で構成要素を編集した場合，バージョンアップ前のバージョン付きフォルダが管理する構成要素も編集後の構成要素に切り替わってしまうため，バージョン付きフォルダのバージョンごとの構成要素の履歴管理ができません。

(2) バージョン付きフォルダによる構成管理の運用例

バージョン付きフォルダの管理モデルでは，バージョン付きフォルダのバージョン管理機能と構成管理型リンクを使用して次のような二つの管理方法が考えられます。

- 文書収集型の管理方法
- 文書提出型の管理方法

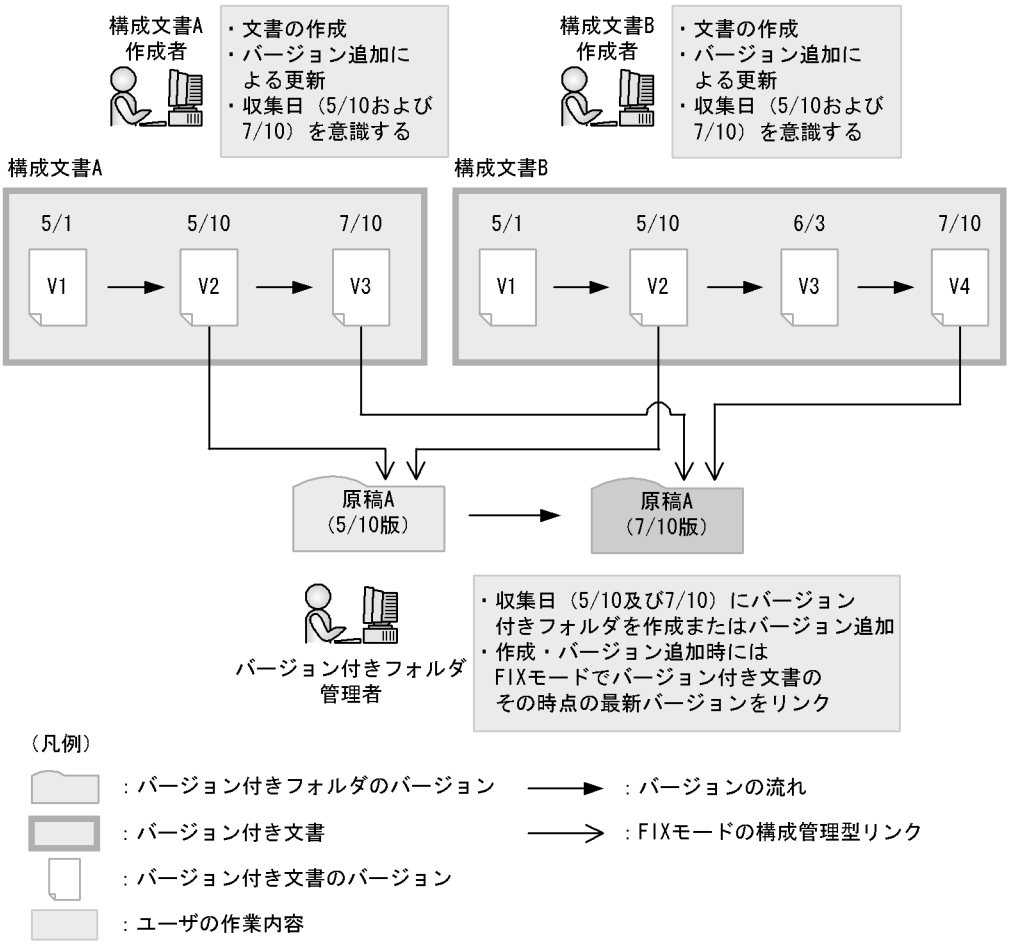
各管理方法について説明します。また，バージョン付きフォルダの複合リンクによる管理例についても説明します。

(a) 文書収集型の管理方法

文書収集型の管理方法とは，構成管理モードの FIX モードだけを使用する管理方法です。フォルダの管理者は，バージョン付き文書の収集期限を示しておき，収集期限がきたら，各バージョン付き文書の最新バージョンを FIX モードでリンク付けてバージョン付きフォルダを新規に作成またはバージョンを追加して管理します。

文書収集型の管理例を次の図に示します。

図 3-39 文書収集型の管理例



この例では，バージョン付きフォルダ「原稿 A」にリンク付けられているバージョン付き文書「構成文書 A」および「構成文書 B」の作成者は，文書収集日までにバージョン付き文書のバージョンを追加しながら更新していきます。バージョン付きフォルダ管理者は，最初の収集日（5 月 10 日）に，その時点の最新バージョンの「構成文書 A」および「構成文書 B」をバージョン付きフォルダ「原稿 A」の構成文書とし

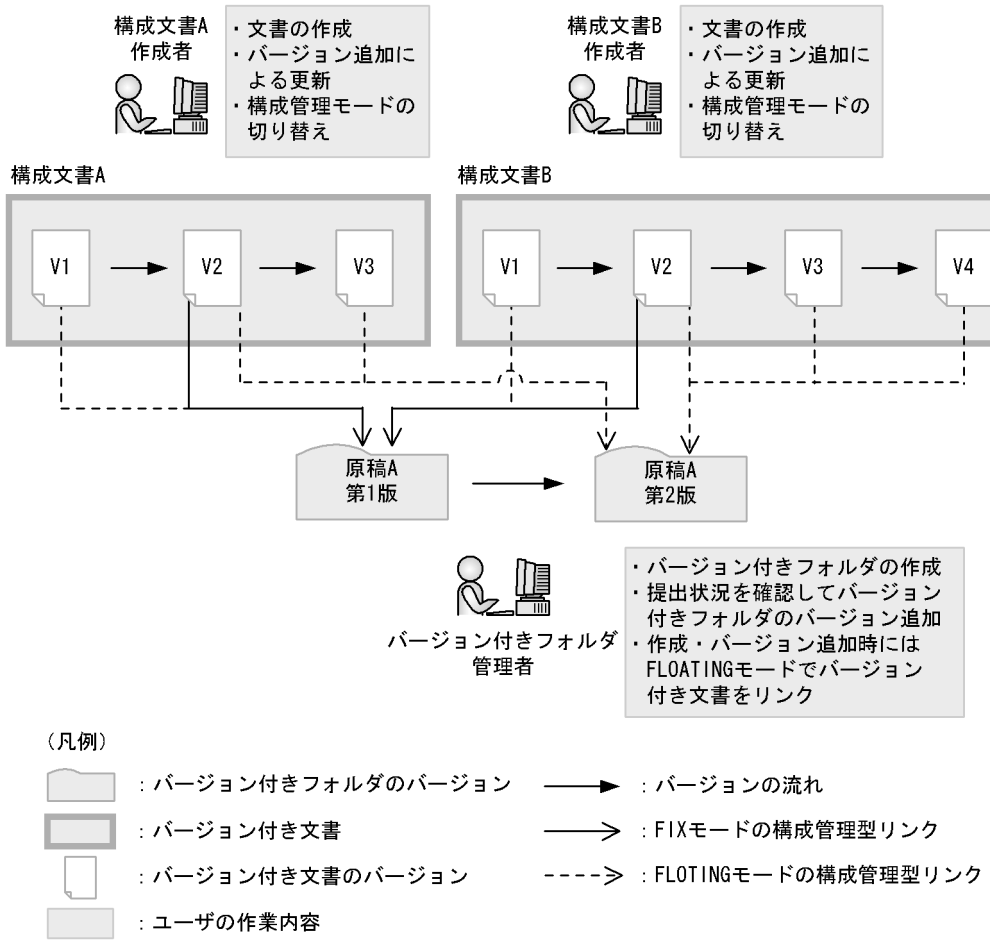
て FIX モードでリンク付けて作成します。そして，次の収集日（7 月 10 日）に，その時点で最新のバージョンの「構成文書 A」および「構成文書 B」を FIX モードでリンク付けてバージョン付きフォルダ「原稿 A」のバージョンを追加しています。

(b) 文書提出型の管理方法

文書提出型の管理方法とは，構成管理モードの FIX モードおよび FLOATING モードを使用する管理方法です。まず，各バージョン付き文書を FLOATING モードでリンク付けてバージョン付きフォルダを作成し，バージョン付き文書の編集を終えた時点で文書作成者が FIX モードに切り替えてバージョンを固定します。

文書提出型の管理例を次の図に示します。

図 3-40 文書提出型の管理例



この例では，バージョン付きフォルダ管理者は，バージョン付き文書「構成文書 A」および「構成文書 B」を FLOATING モードでリンク付けて，「原稿 A」をまとめるバージョン付きフォルダを作成します。そして，各構成文書の作成者が第 1 版としてのバージョン付き文書の編集を終えた時点で FIX モードに切り替えています。バージョン付きフォルダの管理者は，提出状況を確認して（どちらも FIX モードに切り替えられているかどうかを確認），バージョン付きフォルダのバージョンを追加して各バージョン付き文書を FLOATING モードでリンク付けています。

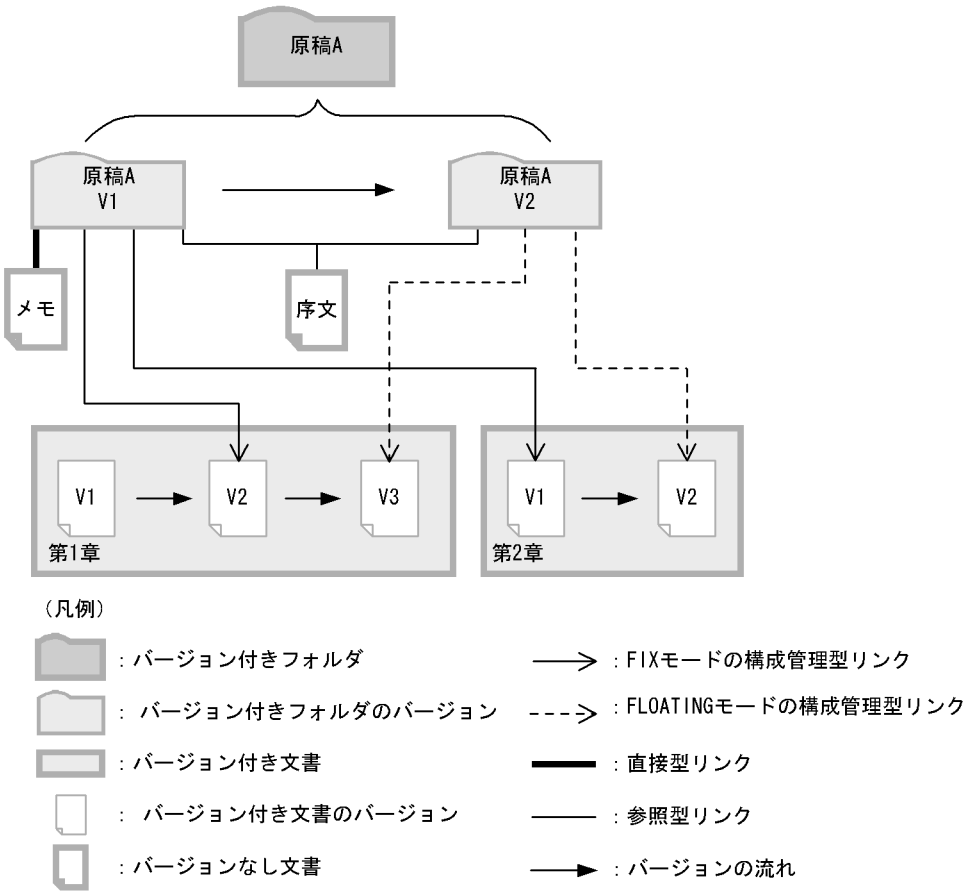
(c) バージョン付きフォルダでの複合リンクによる文書管理

バージョン付きフォルダで，直接型リンク，参照型リンクおよび構成管理型リンクを組み合わせて使用す

る文書の管理例について説明します。

バージョン付きフォルダで使用できる 3 種類のリンクをすべて使用した，バージョン付きフォルダの複合リンクによる文書の管理例を次の図に示します。

図 3-41 バージョン付きフォルダの複合リンクによる文書の管理例



この例では，バージョン付きフォルダにリンク付ける文書の性質を考慮して 3 種類のリンクを使い分けて管理しています。この例では，各文書が，次のような性質の文書であることを想定しています。

●「メモ」
バージョン付きフォルダ「原稿 A」のバージョン 1 だけに関連のあるメモ資料で，改訂の必要がなく，ほかのバージョンには関連のない文書

●「序文」
バージョン付きフォルダ「原稿 A」のすべてのバージョンにわたって共通に関連する文書で，改訂する必要のない文書

●「第 1 章」および「第 2 章」
バージョン管理され，ある時点ではバージョンを固定して，編集中は常に最新バージョンをトレースしたい文書

これらの文書の性質を考慮して，それぞれの文書をバージョン付きフォルダにリンク付ける場合のリンクの種類を次のように使い分けることができます。

●「メモ」
上位フォルダを一つだけ持つことができる直接型リンクでバージョン付きフォルダ「原稿 A」にリンク

付ける。

●「序文」
上位フォルダを複数持つことができる参照型リンクでバージョン付きフォルダ「原稿 A」のすべてのバージョンにリンク付ける。

●「第 1 章」および「第 2 章」
構成管理型リンクを使用し，構成管理モードを使い分けてバージョン付きフォルダ「原稿 A」にリンク付ける。

(3) リンクをたどる文書空間オブジェクトの参照

リンクによって関連付けられた上位オブジェクトと下位オブジェクトは，双方向に参照できます。

上位オブジェクトまたは下位オブジェクトを参照する場合は，上位オブジェクトまたは下位オブジェクトとの関連付けを表すリンクオブジェクトの一覧を取得します。リンクオブジェクトをたどって上位オブジェクトまたは下位オブジェクトを参照します。また，リンクオブジェクト取得時には，次の情報も取得できます。

- 上位オブジェクトまたは下位オブジェクトのオブジェクト種別
- リンク識別子
- 上位オブジェクトまたは下位オブジェクトのプロパティ
- リンクオブジェクトのプロパティ

(4) リンクの解除，上位オブジェクトおよび下位オブジェクトの削除

ここでは，リンクの解除と，上位オブジェクトおよび下位オブジェクトの削除の関係について説明します。

(a) リンクの解除

上位オブジェクトからリンクを解除するか，リンクオブジェクト自身のインターフェースでリンクオブジェクトを削除します。

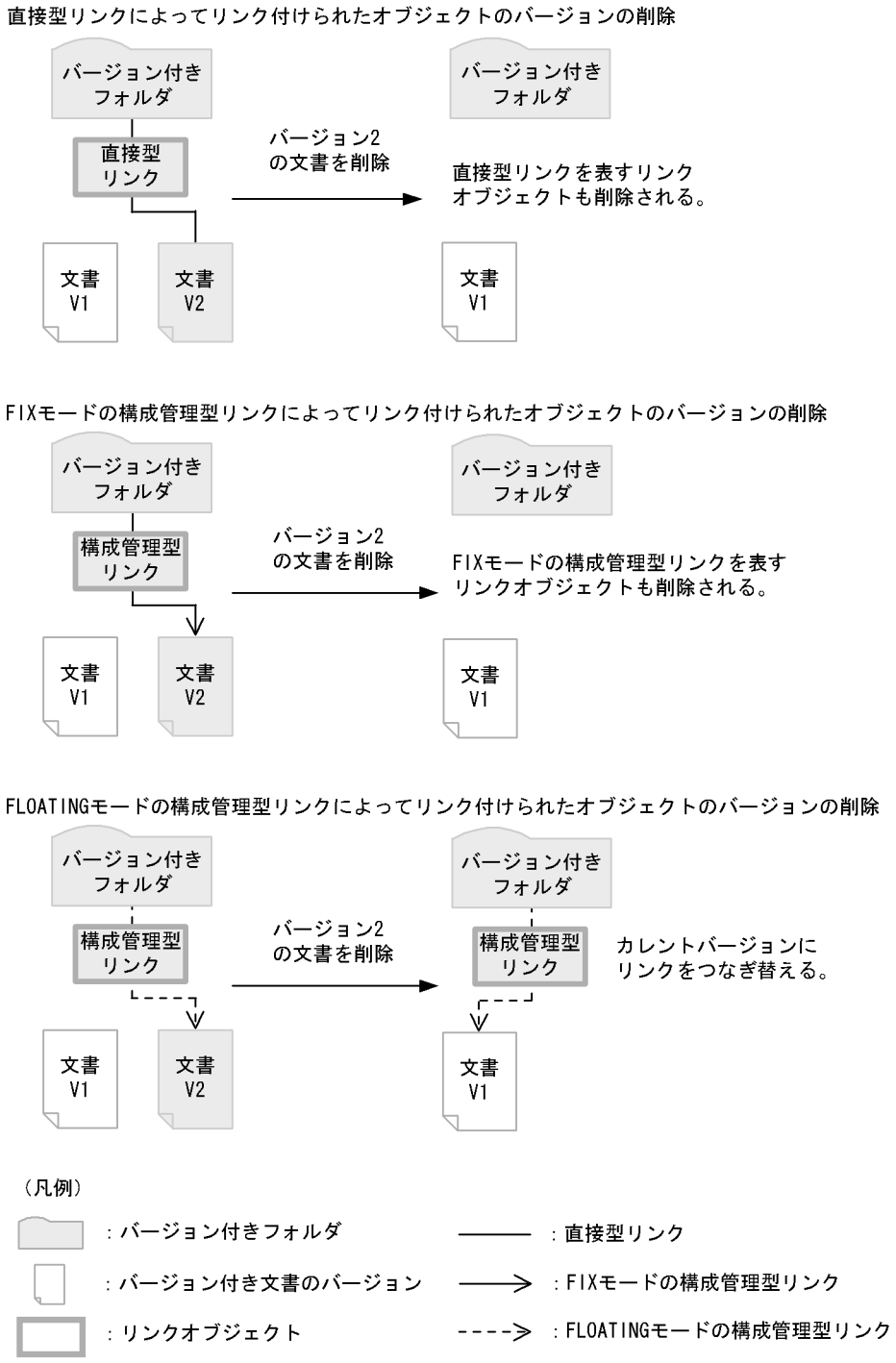
(b) 上位オブジェクトの削除

上位オブジェクトを削除すると，設定されていたリンクオブジェクトも削除されます。

(c) 下位オブジェクトの削除

下位オブジェクトを削除すると，設定されていたリンクオブジェクトも削除されます。ただし，構成管理モードが FLOATING モードでリンク付けられている下位オブジェクトのバージョンを削除する場合，構成管理型リンクを表すリンクオブジェクトは削除されません。バージョン削除後のカレントバージョンにリンクをつなぎ替えます。リンクの種類ごとに削除されるリンクオブジェクトについて次の図に示します。

図 3-42 リンクの種類ごとに削除されるリンクオブジェクト

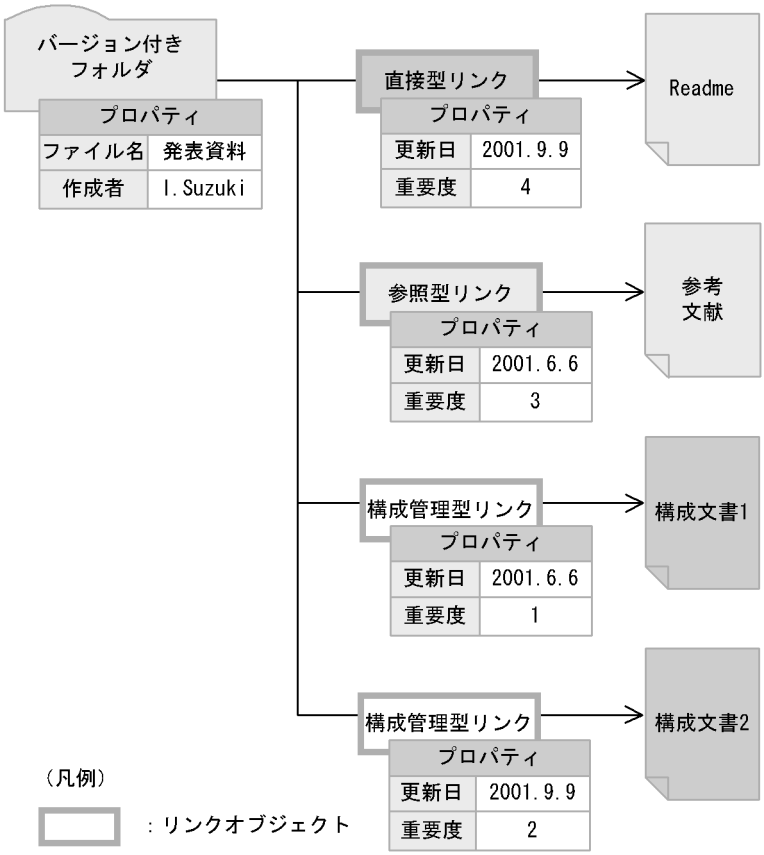


(5) バージョン付きフォルダのプロパティの設定

バージョン付きフォルダおよびバージョン付きフォルダのリンクには，ユーザ定義プロパティを設定して管理できます。リンクのプロパティとは，リンクオブジェクトに設定するプロパティです。フォルダのプロパティとして作成者などを設定しておくと，フォルダの検索で使用できます。また，リンクオブジェクトのプロパティは，検索に使用したり，重要度などの値に応じてソートして，バージョン付きフォルダの下位オブジェクトに順序性を持たせたりできるので便利です。リンクオブジェクトのプロパティは，上位オブジェクトまたはリンクオブジェクト自身のインターフェースを使用して設定します。

プロパティを設定したバージョン付きフォルダとリンクオブジェクトの例を次の図に示します。

図 3-43 プロパティを設定したバージョン付きフォルダとリンクオブジェクトの例



この例では，バージョン付きフォルダにユーザ定義プロパティとして「ファイル名」と「作成者」を設定しています。また，リンクオブジェクトに，ユーザ定義プロパティとして「更新日」と「重要度」を設定しています。例えば，リンクオブジェクトのプロパティ「更新日」の値によって，リンク付けた日付を管理したり，プロパティ「重要度」の値によって，バージョン付きフォルダの下位オブジェクト一覧を取得する場合に，重要度の昇順または降順にソートして情報を取得したりできます。

3.8 独立データ管理モデル

この節では，独立データ管理モデルについて説明します。

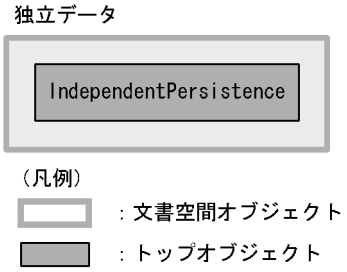
独立データは，独立したデータを扱う文書空間オブジェクトです。単なる表として扱いたいデータなどは独立データとして管理できます。独立データには，ユーザ定義プロパティを設定できます。

独立データは，文書のようにフォルダにリンク付けたり，バージョンを管理したり，フォルダのように上位・下位の階層構造で管理したりすることはできません。

3.8.1 独立データを構成する DMA オブジェクト

独立データを構成する DMA オブジェクトを次の図に示します。

図 3-44 独立データを構成する DMA オブジェクト



独立データを構成する DMA オブジェクトについて説明します。

● **IndependentPersistence オブジェクト**
独立データのトップオブジェクトです。トップオブジェクトクラスは，edmClass_IndependentPersistence クラスまたはそのサブクラスです。

3.9 属性管理モデル

この節では，属性管理モデルについて説明します。

Java クラスライブラリで管理する文書空間オブジェクトは，プロパティの値を DocumentBroker サーバやユーザが設定することによって作成されます。文書空間オブジェクトはプロパティに設定された値によって区別されます。文書空間オブジェクトのプロパティの値を参照してその状態（作成者はだれか，作成日はいつかなど）を確認したり，プロパティをキーにして検索したりできます。

なお，文書空間オブジェクトのプロパティを管理するための操作の詳細については，「6.8.7　文書空間オブジェクトのプロパティの操作」を参照してください。

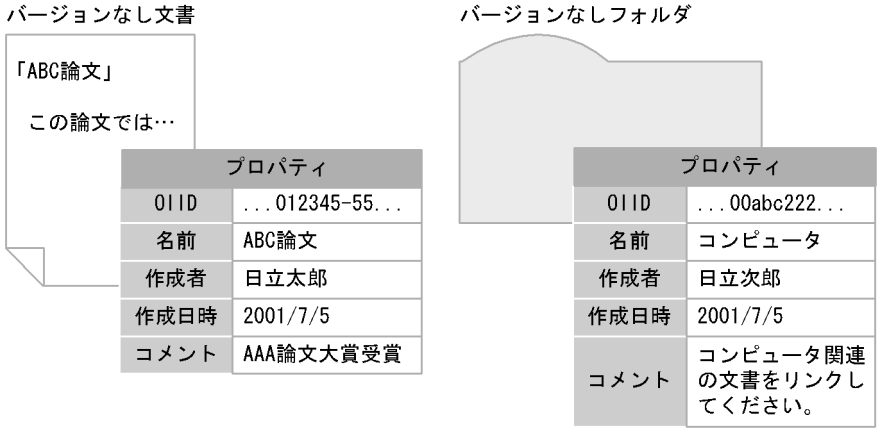
3.9.1 プロパティの管理例

ここでは，文書空間オブジェクトのプロパティとして，ユーザが設定できるプロパティの管理例を示します。

(1) バージョンなし文書およびバージョンなしフォルダのプロパティの管理例

バージョンなし文書およびバージョンなしフォルダのプロパティの管理例を次の図に示します。なお，バージョンを持たない独立データおよびパブリック ACL も，同様にプロパティを管理できます。

図 3-45　バージョンなし文書およびバージョンなしフォルダのプロパティの管理例

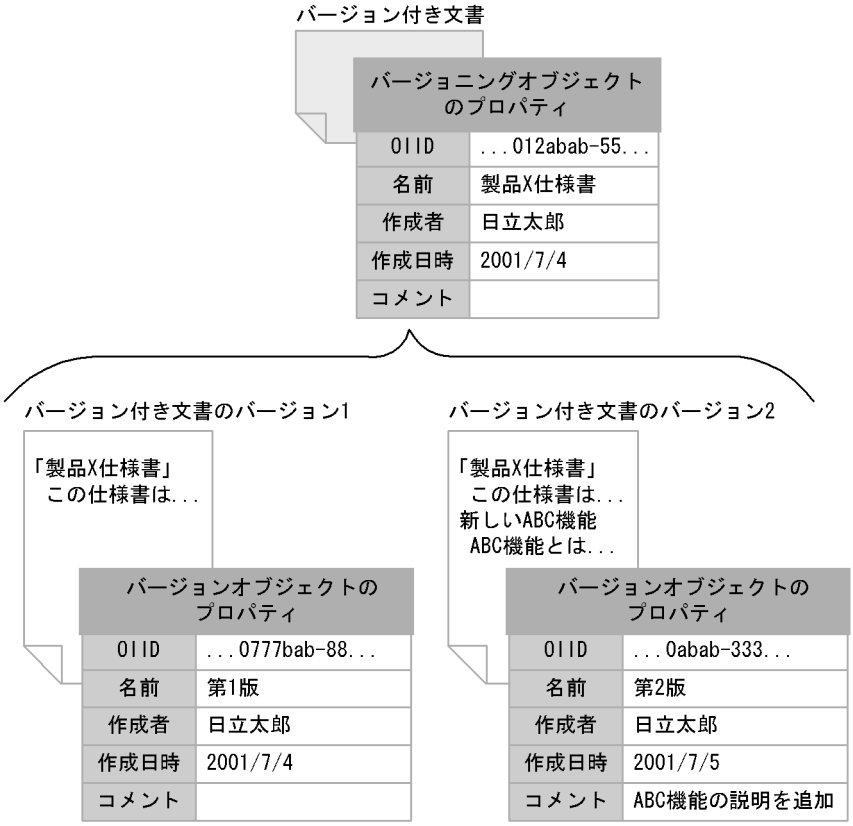


この例では，バージョンなし文書やバージョンなしフォルダには，文書空間オブジェクトの名前（userProp_Name），作成者（userProp_Creator），作成日時（userProp_CreateTime）およびコメント（userProp_Comment）などを表すプロパティをユーザ定義プロパティとして追加定義しています。OIID を表すプロパティ（dmaProp_OIID）は DMA が規定したプロパティであり，文書空間オブジェクト登録時に Java クラスライブラリが設定します。このようにプロパティの値を設定しておくことによって，文書空間オブジェクトを名前で区別したり，設定されている値をキーに目的の文書空間オブジェクトを検索したりできます。例えば，「『作成日時』が『2001 年 7 月以降』で，『作成者』は『日立太郎』の，『名前』に『論文』を含むバージョンなし文書」のような条件で検索できます。OIID は，各文書空間オブジェクトにユニークに与えられる ID であるため，文書空間オブジェクトを特定して DocumentBroker サーバに操作を要求する場合に使用します。

(2) バージョン付きオブジェクトのプロパティの管理例

バージョン付きオブジェクトのプロパティの管理例を，バージョン付き文書の例で次の図に示します。

図 3-46 バージョン付き文書のプロパティの管理例

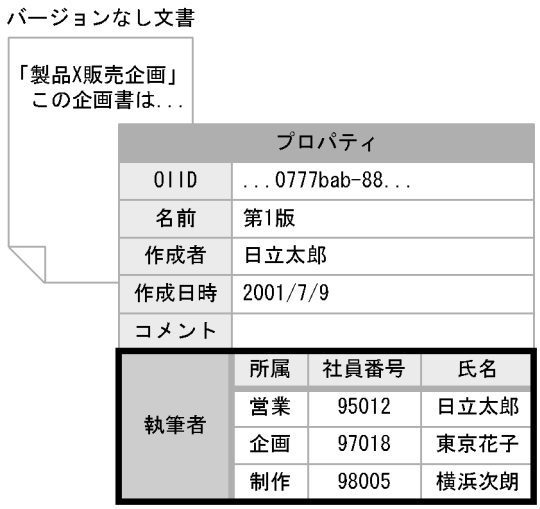


バージョン付きオブジェクトでは，複数バージョンの共通のプロパティをバージョニングオブジェクトのプロパティとして，各バージョン固有のプロパティをバージョンオブジェクトのプロパティとして設定できます。この例では，バージョン付き文書の共通の名称として「名前」（userProp_Name）に「製品 X 仕様書」という値を設定しています。各バージョンには，バージョン固有の名称として「名前」（userProp_Name）に「第 1 版」,「第 2 版」という値を設定し,「第 2 版」には「コメント」（userProp_Comment）として「第 1 版」からの変更内容を設定して管理しています。

(3) VARRAY 型のプロパティの管理例

VARRAY 型のプロパティの管理例を，バージョンなし文書の例で次の図に示します。

図 3-47　VARRAY 型のプロパティの管理例



この例では，VARRAY 型のプロパティとして「執筆者」というプロパティを定義して管理しています。VARRAY 型のプロパティでは，複数のプロパティを 1 組として，その組を複数個管理できます。要素となる複数のプロパティのデータ型は同一でなくてもかまいません。この例では，「所属」，「社員番号」および「氏名」という三つのプロパティを定義して，それらを 1 組にして「執筆者」プロパティの要素とし，複数人の「執筆者」の情報を管理しています。このとき，「所属」のデータ型が STR 型で，「社員番号」のデータ型が INT 型でもかまいません。

「所属」，「社員番号」および「氏名」のように VARRAY 型のプロパティの要素となるプロパティのことを VARRAY 型のプロパティの構成要素のプロパティといいます。

(4) LDAP 対応のディレクトリサービスとの連携を使用したプロパティの管理例

ユーザアプリケーションプログラムを開発する際に，JNDI を使用して LDAP 対応のディレクトリサービスと連携すれば，LDAP 対応のディレクトリサービスで管理されている情報を直接取得できます。

例えば，GUI に文書一覧などを表示する際，文書の所有者の情報をユーザ識別子で表示しても，人物を特定するのが困難です。この場合，ユーザ識別子でなく漢字の氏名などを表示するとわかりやすくなります。また，属性検索でも，所有者を検索条件にして検索する場合，ユーザ識別子を検索条件に指定するよりも，漢字の氏名を検索条件に指定する方がわかりやすくなります。

このような場合に，JNDI を使用して LDAP 対応のディレクトリサービスと連携していれば，ユーザ識別子やグループ識別子の値を検索条件にして，対応する漢字の氏名などのわかりやすい名前を，LDAP 対応のディレクトリサービスから検索して取得できます。ユーザ識別子やグループ識別子が設定されるプロパティについては，LDAP 対応のディレクトリサービスからユーザにわかりやすい名前を検索して，その名前を設定するプロパティを対応付けて定義および管理すると便利です。

LDAP 対応のディレクトリサービスとの連携については，ご使用になる LDAP 対応のディレクトリサービスのマニュアルを参照してください。

3.10 アクセス制御モデル

この節では，Java クラスライブラリでのアクセス制御モデルについて説明します。なお，アクセス制御を管理するための操作の詳細については，「6.9 アクセス制御に関する操作」を参照してください。

DocumentBroker では，ユーザがログインする時に生成されるユーザ情報や文書空間オブジェクトに設定されたアクセス制御情報などからユーザのアクセス権を判定してアクセスを制御します。

次のようにアクセスが制御されます。

1. **ユーザが文書空間にログインします。**
2. **ログイン時にユーザ情報が生成されます。**
   ユーザ情報には，ログインユーザを識別する情報や，ログインユーザのユーザ権限や特権の有無についての情報が設定されます。
3. **ログインユーザが操作を要求するとユーザ情報が参照されます。**
   - ユーザ情報に特権を示す情報を保持している場合，アクセス権の判定は成功したものとして，要求が処理されます。
   - ユーザ権限として，要求した操作を許可する情報を保持している場合，要求が処理されます。
4. **特権または要求した操作を許可するユーザ権限がユーザ情報に保持されていない場合，操作対象の文書空間オブジェクトに設定されているアクセス制御情報が参照されます。**
   - 操作対象の文書空間オブジェクトに，要求した操作をそのユーザに許可するアクセス制御情報が設定されている場合，要求が処理されます。
   - 操作対象の文書空間オブジェクトに，要求した操作をそのユーザに許可するアクセス制御情報が設定されていない場合，要求は処理されないで，エラーが返却されます。

なお，ログイン時のユーザ認証に必要な情報，およびアクセス権の判定に必要なユーザ情報を取得する場合は，DocumentBroker サーバ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を使用するかについては，DocumentBroker サーバに定義されています。

ユーザ情報，特権およびユーザ権限の詳細については，「3.10.1 アクセス権の判定に使用される情報」を参照してください。

3.10.1 アクセス権の判定に使用される情報

ここでは，アクセス権の判定に使用される情報について説明します。

DocumentBroker では，ユーザのアクセス権の判定に次の情報を使用します。

- ユーザ情報
- 文書空間オブジェクトごとのアクセス制御情報

(1) ユーザ情報

DocumentBroker では，アクセス権の判定に，ユーザ情報を使用します。ユーザ情報は，文書空間にログインしたときに Java クラスライブラリのセッションオブジェクトに関連づけられて保持されます。ユーザ情報は，次の情報で構成されます。

- ユーザ識別子
- 所属グループ
- 特権
- ユーザ権限

それぞれについて説明します。

(a) ユーザ識別子

ユーザが文書空間にログインした時に使用したユーザ管理システムから取得される，ログインユーザを識別する情報です。設定される値は，使用したユーザ管理システムによって異なります。ユーザ管理システムが LDAP 対応のディレクトリサービスの場合は，DocumentBroker サーバの DocumentSpace 構成定義ファイルに定義した属性の値が設定されます。UNIX のパスワードファイルの場合は，UNIX のユーザ ID が設定されます。

DocumentBroker サーバの DocumentSpace 構成定義ファイルについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。また，LDAP 対応のディレクトリサービスについては，ご使用になる LDAP 対応のディレクトリサービスのマニュアルを参照してください。

(b) 所属グループ

ユーザが文書空間にログインした時に使用したユーザ管理システムから取得される，ログインユーザが所属するグループの数と，各グループを識別するための**グループ識別子**の一覧情報です。設定される値は，使用したユーザ管理システムによって異なります。ユーザ管理システムが LDAP 対応のディレクトリサービスの場合は，DocumentBroker サーバの DocumentSpace 構成定義ファイルに定義した属性の値が設定されます。UNIX のパスワードファイルの場合は，UNIX のグループ ID が設定されます。

DocumentBroker サーバの DocumentSpace 構成定義ファイルについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。また，LDAP 対応のディレクトリサービスについては，ご使用になる LDAP 対応のディレクトリサービスのマニュアルを参照してください。

(c) 特権

アクセス権の判定を受けないですべての文書空間オブジェクトに対して自由にアクセスできる権限のことを**特権**といいます。DocumentBroker では，すべての文書空間オブジェクトや，各文書空間オブジェクトに設定されたアクセス制御情報の保守を担当する，特権を持つユーザを設定できます。このユーザを**セキュリティ管理者**といいます。セキュリティ管理者は，DocumentBroker サーバの**セキュリティ定義ファイル**に定義しておきます。ログイン時にセキュリティ定義ファイルが参照され，ログインするユーザがセキュリティ管理者の場合，ユーザ情報に「特権のあるセキュリティ管理者」を示す値が設定されます。ログインユーザがセキュリティ管理者でない場合は「特権なし」を示す値が設定されます。

セキュリティ管理者の定義方法については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

(d) ユーザ権限

ユーザ権限とは，DocumentBroker サーバのユーザ権限定義ファイルに定義した情報を基に，ユーザ，グループまたはすべてのユーザに対して設定できる権限です。**ユーザ権限定義ファイル**には，すべての文書空間オブジェクトに対するアクセス権を定義します。

ユーザ権限には次の 2 種類があります。

- オブジェクト作成権限
- オブジェクト操作権限

オブジェクト作成権限

オブジェクト作成権限とは，文書空間にオブジェクトを作成する権利を任意のユーザまたはグループに対して許可するアクセス制御情報です。ログイン時にユーザ権限定義ファイルが参照されて，ログ

インユーザにオブジェクト作成権が設定されている場合，ユーザ情報に「オブジェクトを作成する権利」を示すパーミッションが設定されます。
オブジェクト作成権限の定義方法については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

オブジェクト操作権限

オブジェクト操作権限とは，文書空間内のすべてのオブジェクトに対してどの範囲の操作を許可するかを定義するアクセス制御情報です。オブジェクト操作権限は，任意のユーザまたはグループごとに設定できます。例えば，「ユーザ A にプロパティを参照する権利を与える」と定義しておくと，「ユーザ A」は文書空間内のすべてのオブジェクトのプロパティを参照できます。ログイン時にユーザ権限定義ファイルが参照されて，ログインするユーザにオブジェクト操作権限としてプロパティ参照権が定義されていた場合，ユーザ情報に「プロパティを参照する権利」を示すパーミッションが設定されます。
オブジェクト操作権限の設定については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

(2) 文書空間オブジェクトごとのアクセス制御情報

文書空間オブジェクトに対するアクセスを制御するための情報です。文書空間オブジェクトごとに「どのユーザまたはグループに」，「どのような操作を許可するか」を表すアクセス制御情報を設定しておくことで，各ユーザまたはグループに対して文書空間オブジェクトごとのアクセス権を設定できます。文書空間オブジェクトのアクセス制御情報の種類については，「3.10.3　文書空間オブジェクトごとのアクセス制御情報の種類」を参照してください。

アクセス制御情報を設定できる文書空間オブジェクトのオブジェクト種別は，次のとおりです。

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

3.10.2 アクセス制御情報に設定できるパーミッションの種類

ここでは，アクセス制御情報に設定できるパーミッションの種類について説明します。

「どのような操作を許可するか」を表す値を**パーミッション**といいます。例えば，ある文書に対して，「ユーザ A に，プロパティの参照だけを許可する」，「ユーザ B に，プロパティの参照およびコンテンツの更新を許可する」のように設定して，許可する操作の範囲を限定できます。

(1) パーミッションの種類

アクセス制御情報として設定するパーミッションの種類を次の表に示します。

表 3-9　パーミッションの種類

パーミッションの用途	パーミッションの種類
個々の文書空間オブジェクトに対する操作を許可する	基本パーミッション 組み合わせパーミッション
アクセス制御情報に対する操作を許可する	アクセス制御情報変更権

パーミッションの用途	パーミッションの種類
文書空間オブジェクトの作成を許可する	オブジェクト作成権

各パーミッションについて説明します。

基本パーミッションと組み合わせパーミッション

個々の文書空間オブジェクトに対する操作を許可するパーミッション（ACFlag および ACL に指定するパーミッション）です。パーミッションは複数指定できます。例えば，複数の基本パーミッションを指定したり，複数の組み合わせパーミッションを指定したり，基本パーミッションと組み合わせパーミッションを混在させて指定したりできます。この場合，パーミッションとして設定される内容は，指定したパーミッションの論理和になります。

基本パーミッションの詳細については，「(2) 基本パーミッション」を参照してください。組み合わせパーミッションの詳細については，「(3) 組み合わせパーミッション」を参照してください。また，ACFlag および ACL については，「3.10.3　文書空間オブジェクトごとのアクセス制御情報の種類」を参照してください。

アクセス制御情報変更権

アクセス制御情報に対する操作を許可するパーミッションです。セキュリティ ACL だけに指定します。アクセス制御情報変更権およびセキュリティ ACL については，「3.10.4　アクセス制御情報の管理」を参照してください。

オブジェクト作成権

文書空間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。ユーザ権限のオブジェクト作成権限を定義する時に，ユーザ権限定義ファイルに指定します。パーミッションを表す定数は「DbjDef.PERM_CREATE」です。ユーザ権限定義ファイルで定義するパーミッションについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

なお，パーミッションを個々の文書空間オブジェクトに設定する方法については，「3.10.3　文書空間オブジェクトごとのアクセス制御情報の種類」を参照してください。

(2) 基本パーミッション

基本パーミッションは，文書空間オブジェクトを操作する範囲を限定するために，DocumentBroker が提供する基本的なパーミッションです。基本パーミッションは，ユーザ権限のオブジェクト操作権限の定義およびオブジェクトごとのアクセス制御情報の定義で指定します。基本パーミッションの種類を次の表に示します。

表 3-10　基本パーミッションの種類

パーミッションの種類	説明
基本プロパティ参照権 （DbjDef.PERM_PRIM_READ_PROPS）	文書空間オブジェクトのプロパティを参照する権利を与える基本パーミッションです。 このパーミッションを設定されたユーザは，文書やフォルダの OIID，ユーザ定義プロパティを参照できます。プロパティを指定した属性検索を実行する場合も，このパーミッションが必要です。また，このパーミッションを設定されないユーザに対しては，その文書空間オブジェクトの存在もわからないようにできます。 このパーミッションには，フォルダに設定されたリンクオブジェクトのプロパティを参照したり，フォルダとリンク付けられている文書空間オブジェクトのプロパティを参照したりする権利も含まれます。

パーミッションの種類	説明
基本プロパティ更新権 （DbjDef.PERM_PRIM_WRITE_PROPS）	文書空間オブジェクトのプロパティを更新する権利を与える基本パーミッションです。基本プロパティ参照権を含みます。 このパーミッションを設定されたユーザは，文書やフォルダに設定されたユーザ定義プロパティ，そのほかの読み取り専用以外のプロパティを更新できます。
基本コンテンツ参照権 （DbjDef.PERM_PRIM_READ_CONTENTS）	文書のコンテンツを参照する権利を与える基本パーミッションです。基本プロパティ参照権を含みます。 全文検索インデクスを使用した検索を実行する場合は，このパーミッションが必要です。
基本コンテンツ更新権 （DbjDef.PERM_PRIM_WRITE_CONTENTS）	文書のコンテンツを更新する権利を与える基本パーミッションです。基本プロパティ参照権を含みます。 全文検索インデクスを作成および削除する場合は，このパーミッションが必要です。
基本リンク権 （DbjDef.PERM_PRIM_LINK）	リンクに関する操作を実行する権利を与える基本パーミッションです。基本プロパティ参照権を含みます。 リンクに関する操作とは，リンク（直接型リンク，参照型リンク，構成管理型リンクおよび文書間リンク）を設定および解除する権利です。このパーミッションには，リンクオブジェクトを削除する権利，リンクオブジェクトのプロパティを設定および更新する権利も含まれます。また，フォルダの場合は，構成管理モードを変更する権利も含まれます。
基本バージョン管理権 （DbjDef.PERM_PRIM_VERSION）	バージョン管理に関する操作を実行する権利を与える基本パーミッションです。基本プロパティ参照権を含みます。 バージョン管理に関する操作とは，バージョン付きオブジェクトのバージョンを追加および削除する権利です。また，バージョン付きフォルダの場合は，構成管理モードを変更する権利も含まれます。
基本削除権 （DbjDef.PERM_PRIM_DELETE）	文書空間オブジェクトを削除する権利を与える基本パーミッションです。基本プロパティ参照権を含みます。

注 「DbjDef.PERM_」で始まる定数は，パーミッションを表す定数です。

基本パーミッションでは，それぞれのパーミッションは独立であり，包含関係はありません。ただし，基本プロパティ参照権は，すべての基本パーミッションに含まれます。

基本パーミッションは，複数組み合わせて使用できます。また，組み合わせパーミッションを使用すると，複数の基本パーミッションを指定した場合と同じパーミッションが設定できます。この場合，パーミッションとして設定される内容は，指定したパーミッションの論理和になります。組み合わせパーミッションについては，「(3) 組み合わせパーミッション」を参照してください。

基本パーミッションの設定例

基本パーミッションの設定例を示します。

プロパティの参照を可能にする場合

次のパーミッションを設定します。

- 基本プロパティ参照権

フォルダの階層をたどったり，フォルダ名，文書名などの文書空間オブジェクトのプロパティの値を参照したりできます。

プロパティとコンテンツの参照を可能にする場合

次のパーミッションを設定します。

- 基本コンテンツ参照権

フォルダの階層をたどったり，フォルダ名，文書名などの文書空間オブジェクトのプロパティの値を参照したり，コンテンツをダウンロードしたりできます。

プロパティとコンテンツの更新を可能にする場合

次のパーミッションを設定します。
- 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

文書またはフォルダのプロパティとコンテンツを参照可能にした場合の操作に加えて，プロパティ，コンテンツの更新が可能になります。

削除を可能にする場合

次のパーミッションを設定します。
- 基本削除権

文書またはフォルダのプロパティの参照と削除ができます。使用されていない文書またはフォルダを削除してデータベースを保守するユーザなどに与えるパーミッションです。

リンクを可能にする場合

次のパーミッションを設定します。
- 基本リンク権

フォルダに文書またはほかのフォルダをリンク付けたり，文書間リンクを設定したりできます。あるユーザがフォルダ体系を決定し，別のユーザには，そのフォルダ体系に従って文書をリンク付けさせたり，文書間リンクを設定させたりしたい場合，文書のリンク付けや文書間リンクの設定だけをするユーザには基本リンク権だけを設定します。

バージョンアップを可能にする場合

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に次のパーミッションを設定します。
- 基本バージョン管理権

また，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に次のパーミッションを設定します。
- 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

バージョン付きオブジェクトをバージョンアップし，追加したバージョンのバージョン名を設定したり，バージョン付き文書の場合はコンテンツを更新したりできます。

なお，文書空間オブジェクトごとにアクセス制御情報を設定する場合，文書空間オブジェクトのオブジェクト種別によって，アクセス制御情報に設定できるパーミッションの種類が異なります。パーミッションを設定する対象オブジェクトと基本パーミッションの対応を次の表に示します。

表 3-11　パーミッションを設定する対象オブジェクトと基本パーミッションの対応

基本パーミッションの種類	オブジェクト種別				
	D	VD	F	VF	IP
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	×	○	×	○	×
基本削除権	○	○	○	○	○

（凡例）
D：バージョンなし文書および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およびバージョン付きフォルダのバージョンオブジェクト
VF：バージョン付きフォルダ
IP：独立データ
○：設定できることを示します。
×：設定できないことを示します。

(3) 組み合わせパーミッション

組み合わせパーミッションは，基本パーミッションを組み合わせて定義されたパーミッションです。ある文書空間オブジェクトに対するパーミッションを設定する場合に，一般的に同時に指定すると考えられる複数の基本パーミッションが，まとめて一つのパーミッションとして指定できるように定義されています。組み合わせパーミッションの種類を次の表に示します。

表 3-12 組み合わせパーミッションの種類

パーミッションの種類	説明
プロパティ参照権 （DbjDef.PERM_READ_PROPS）	文書空間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。 このパーミッションは，文書空間オブジェクトのプロパティを参照したり，階層をたどったりする操作を許可するユーザに対して設定します。
プロパティ更新権 （DbjDef.PERM_WRITE_PROPS）	文書空間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。 このパーミッションは，文書空間オブジェクトのプロパティを参照して，そのプロパティの値の更新を許可するユーザに対して設定します。
参照権 （DbjDef.PERM_READ）	文書空間オブジェクトを参照する権利を与えるパーミッションです。 このパーミッションは，文書空間オブジェクトのプロパティを参照した上で，コンテンツの参照も許可するユーザに対して設定します。
参照更新権 （DbjDef.PERM_READ_WRITE）	文書空間オブジェクトを参照および更新する権利を与えるパーミッションです。 このパーミッションは，文書空間オブジェクトのプロパティやコンテンツの参照および更新を許可するユーザに対して設定します。
削除権 （DbjDef.PERM_DELETE）	文書空間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。 このパーミッションは，文書空間オブジェクトのプロパティを参照した上で，その文書空間オブジェクトの削除も許可するユーザに対して設定します。 使用されていない文書を削除してデータベースを保守するユーザなどに設定します。
リンク権 （DbjDef.PERM_LINK）	文書やフォルダのリンクを設定する権利を与えるパーミッションです。 このパーミッションは，文書やフォルダの名前などは変更させないで，文書やフォルダのリンクの設定を許可するユーザに対して設定します。 すでにフォルダの体系が決定されている場合に，あるユーザにその体系に従って文書を格納させたいときなどに設定します。
バージョン権 （DbjDef.PERM_VERSION）	バージョンを管理する権利を与えるパーミッションです。 バージョンの管理を許可するユーザに対して設定します。ただし，バージョン付きオブジェクトをバージョンアップする場合などには，バージョン管理の対象になる文書またはフォルダに対してこのパーミッションを与えると同時に，個々のバージョンに対応する文書またはフォルダに対して参照更新権が必要です。
フルコントロール （DbjDef.PERM_FULL_CONTROL）	文書空間オブジェクトに対するすべての操作を実行する権利を与えるパーミッションです。 このパーミッションは，文書空間オブジェクトの参照，更新および削除などのすべての操作を許可するユーザに対して設定します。

注 「DbjDef.PERM_」で始まる定数は，パーミッションを表す定数です。

組み合わせパーミッションと基本パーミッションの対応を次の表に示します。

表 3-13 組み合わせパーミッションと基本パーミッションの対応

組み合わせパーミッションの種類	対応する基本パーミッションの種類						
	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	基本削除権
プロパティ参照権	○	－	－	－	－	－	－
プロパティ更新権	○	○	－	－	－	－	－
参照権	○	－	○	－	－	－	－
参照更新権	○	○	○	○	－	－	－
削除権	○	－	－	－	－	－	○
リンク権	○	－	－	－	○	－	－
バージョン権	○	○	○	○	－	○	－
フルコントロール	○	○	○	○	○	○	○

（凡例）
○：組み合わせパーミッションに対応する基本パーミッションです。
－：組み合わせパーミッションに対応しない基本パーミッションです。

組み合わせパーミッションを使用すると，一つの組み合わせパーミッションを指定することによって，複数の基本パーミッションを指定した場合と同じパーミッションが設定できます。

例えば，あるユーザに対して文書のコンテンツの更新を許可したい場合，運用によっては，「検索によって文書を取得し，既存のコンテンツの内容を参照した上で，修正，更新する」という操作の流れが考えられます。また，「コンテンツを更新する場合には，更新した日時を表すプロパティを更新する」という操作があるとします。この場合，基本パーミッションを使用すると，「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」の 3 種類のパーミッションをそのユーザに設定する必要があります。これに対して，組み合わせパーミッションを使用すると，「参照更新権」を指定するだけで，3 種類の基本パーミッションを指定した場合と同じパーミッションが設定できます。

また，組み合わせパーミッションと基本パーミッションを組み合わせて指定することもできます。例えば，「参照更新権」を与えたユーザに対して，「文書空間オブジェクトを削除する権利も与えたい」という場合は，組み合わせパーミッションの「参照更新権」と，基本パーミッションの「基本削除権」をあわせて指定できます。この場合，パーミッションとして設定される内容は，指定したパーミッションの論理和になります。

3.10.3 文書空間オブジェクトごとのアクセス制御情報の種類

ここでは，アクセス制御情報のうち，文書空間オブジェクトごとに設定するアクセス制御情報について説明します。文書空間オブジェクトごとに設定するアクセス制御情報の種類を次に示します。

- 所有者
- プライマリグループ
- ACFlag
- ACL（ローカル ACL，パブリック ACL，セキュリティ ACL）

文書空間オブジェクトごとのアクセス制御情報は，ACFlag または ACL として設定します。文書空間オブジェクトごとに設定されたアクセス制御情報を管理するための情報も，ACL として設定します。また，文書空間オブジェクトごとに設定する所有者およびプライマリグループのアクセス権は，ACFlag として設

定します。

ACFlag として設定する情報，ACL として設定する情報について説明します。

(1) ACFlag として設定する情報

ACFlag は，文書空間オブジェクトの所有者，プライマリグループおよびすべてのユーザという区切りで，それぞれにパーミッションを設定できるアクセス制御情報です。

ACFlag は，文書空間オブジェクトのプロパティとして直接設定します。したがって，ACL を使用したアクセス権の判定に比べて，高速なアクセス権の判定ができるのが特長です。ACFlag は，次の三つのプロパティで表されます。

- dbrProp_OwnerPermission プロパティ
  所有者のアクセス権を設定するプロパティです。
- dbrProp_PrimaryGroupPermission プロパティ
  プライマリグループに所属するユーザのアクセス権を設定するプロパティです。
- dbrProp_EveryonePermission プロパティ
  すべてのユーザのアクセス権を設定するプロパティです。

プロパティの詳細については，「3.10.5(2) アクセス制御情報を設定するためのプロパティ」を参照してください。

ACFlag によってアクセス権を設定できる所有者，プライマリグループおよびすべてのユーザについて説明します。

所有者

アクセス制御機能を使用している場合，文書空間オブジェクトには必ず所有者が存在します。所有者とは，文書空間オブジェクトに設定されているアクセス制御情報を変更できるユーザです。文書空間オブジェクト作成時には，作成したユーザが所有者として登録されます。所有者以外のユーザがアクセス制御情報を変更できるようにしたい場合，まず所有者によってアクセス制御情報を変更する権利を与える必要があります。また，所有者は，所有者自身を変更できます。
ACFlag では，アクセス制御情報と所有者の変更以外に，所有者が実行できる操作を設定できます。
所有者は，dbrProp_OwnerId プロパティに設定されたユーザ識別子によって表されます。
dbrProp_OwnerId プロパティの値は，文書空間オブジェクト作成時に設定されます。所有者またはセキュリティ管理者は，dbrProp_OwnerId プロパティの値を変更することによって，所有者を変更できます。なお，所有者がアクセス制御情報やセキュリティ ACL の値を参照する場合は，基本プロパティ参照権が必要です。セキュリティ ACL については，「3.10.4　アクセス制御情報の管理」を参照してください。

プライマリグループ

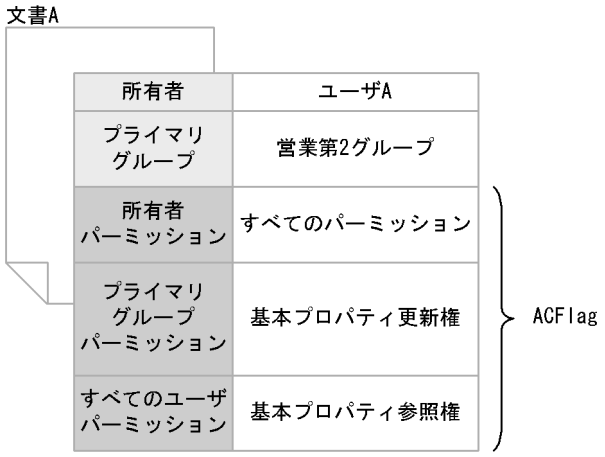
ACFlag でパーミッションを設定してアクセス権を与える一つのグループです。プライマリグループは，dbrProp_PrimaryGroupId プロパティに設定されたグループ識別子によって表されます。文書空間オブジェクト作成時にこのプロパティに設定される値は，ユーザが文書空間にログインした時に使用したユーザ管理システムによって異なります。ユーザ管理システムが LDAP 対応のディレクトリサービスの場合は，文書空間オブジェクト作成時には設定されません。文書空間オブジェクトに頻繁にアクセスするグループのグループ識別子をユーザが設定します。UNIX のパスワードファイルの場合は，文書空間オブジェクト作成時に所有者が所属するグループ（UNIX のパスワードファイルのグループ識別子）が設定されます。

すべてのユーザ

所有者，プライマリグループに所属するユーザ以外のすべてのユーザです。

ACFlag の概要を次の図に示します。

図 3-48 ACFlag の概要



この例では，「ユーザ A」が作成した「文書 A」を ACFlag によってアクセス制御する場合のパーミッションの設定方法を示しています。このように ACFlag を設定すると，所有者である「ユーザ A」にすべてのアクセス権，プライマリグループに設定されている「営業第 2 グループ」に所属するユーザに基本プロパティ更新権（基本プロパティ参照権を含む），そのほかのすべてのユーザに基本プロパティ参照権だけを設定するというアクセス制御が可能になります。

(2) ACL として設定する情報

ACL は，任意のユーザまたは任意のグループごとにパーミッションを設定してアクセス権を与えるためのリストです。

ACL の概要を次の図に示します。

図 3-49 ACL の概要

この例のように，ACL には，複数の ACE（アクセス制御エレメント）を設定できます。ACE は 64 個まで設定できます。個々の ACE には「アクセス権を与えるユーザまたはグループ」と「アクセスの範囲を決定するパーミッション」を対にして設定します。アクセス権を与えるユーザまたはグループのことを特に**サブジェクト**といいます。この例では，サブジェクトとしてユーザ A，ユーザ B，営業 2 課および経理 1 課を指定して，それぞれにパーミッションを設定しています。

ACL を使用すると，複数のユーザやグループに対して個別にパーミッションを設定してアクセス権を与えられますが，アクセス権の判定には，ACFlag の場合よりも時間がかかります。

DocumentBroker では，目的に応じて次の 2 種類の ACL を使用して，アクセス制御情報を設定できます。

- ローカル ACL
- パブリック ACL

また，次の ACL を使用して，アクセス制御情報に対する操作を許可するアクセス制御情報変更権を管理できます。

- セキュリティ ACL

ここでは，ローカル ACL およびパブリック ACL について説明します。アクセス制御情報変更権およびセキュリティ ACL については，「3.10.4　アクセス制御情報の管理」を参照してください。

(a) ローカル ACL

文書空間オブジェクトごとのアクセス権を設定するための ACL です。個々の文書空間オブジェクトの dbrProp_ACL プロパティとして設定します。図 3-49 で示したように，ローカル ACL は，ACE を表すオブジェクトを構成要素とした，VARRAY 型のプロパティです。dbrProp_ACL プロパティについては，「3.10.5(2) アクセス制御情報を設定するためのプロパティ」を参照してください。

(b) パブリック ACL

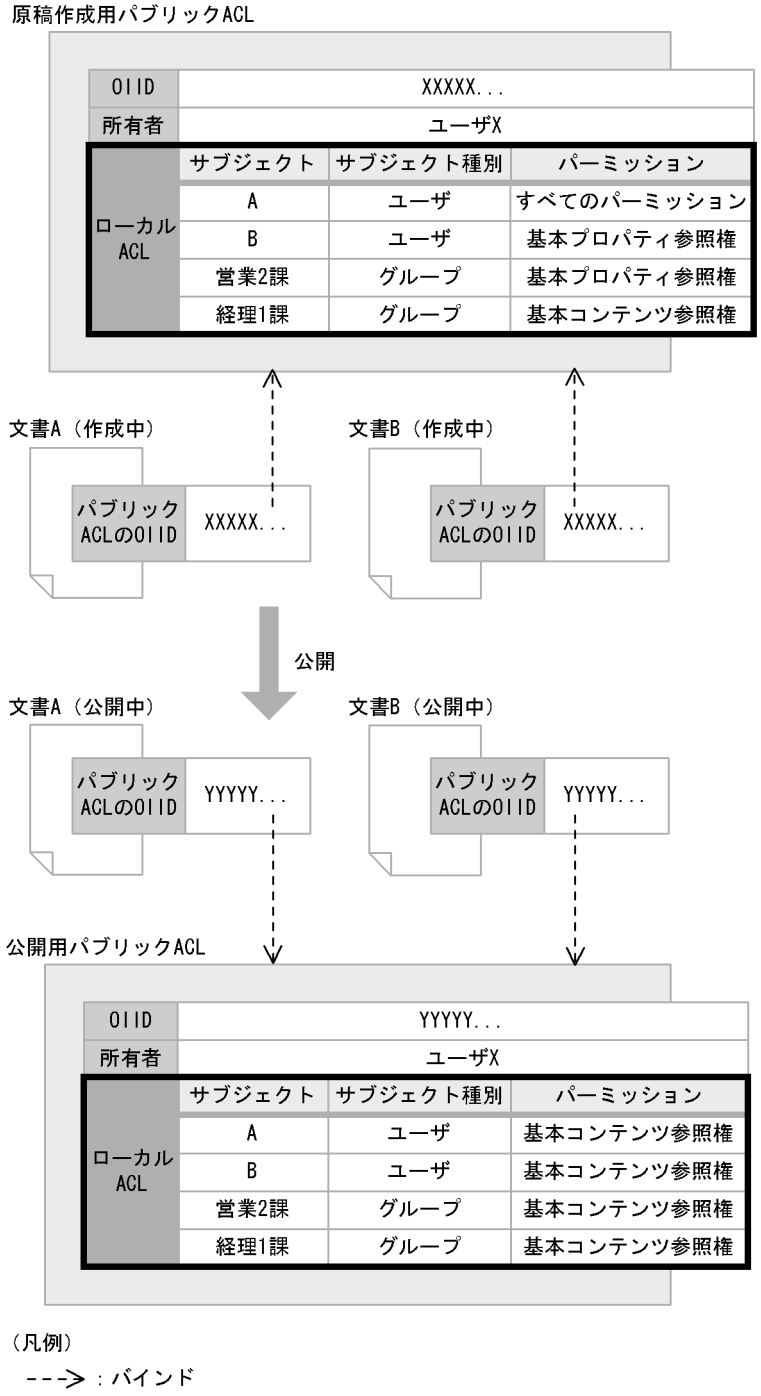
アクセス制御情報の一括管理に使用できる ACL です。パブリック ACL を使用すると，複数の文書空間オブジェクト間でアクセス制御情報を共有できます。

パブリック ACL は，ACL を共有するために使用する独立した文書空間オブジェクトです。共有したい ACL をパブリック ACL に設定して，文書やフォルダなどのほかの文書空間オブジェクトから参照させます。これによって，パブリック ACL に設定した ACL の値を，複数の文書空間オブジェクトに一括して適用させることができます。パブリック ACL は一つの文書空間オブジェクトに対して 10 個まで設定できます。

なお，文書やフォルダを表す文書空間オブジェクトからパブリック ACL を参照することを**バインド**といいます。また，パブリック ACL のバインドを解除することを**アンバインド**といいます。

パブリック ACL の概要を次の図に示します。

図 3-50　パブリック ACL の概要



この例では，作成中の段階には文書 A と文書 B から「原稿作成用パブリック ACL」をバインドさせて，原稿を公開する段階になった時点で，「公開用パブリック ACL」をバインドさせるように切り替えてアクセス制御しています。

パブリック ACL は，複数の文書空間オブジェクトに同じ ACL を設定して，一括してアクセス制御する場合に便利です。また，パブリック ACL のローカル ACL の設定を更新すると，そのパブリック ACL をバインドしているすべての文書空間オブジェクトに更新が反映されるため，グループ名やグループを構成するメンバの変更が頻繁にある場合の保守が容易になります。

パブリック ACL は文書空間オブジェクトとして作成します。パブリック ACL の文書空間オブジェクトクラスは，パブリック ACL クラスです。パブリック ACL として，複数の文書空間オブジェクト間で共有するために設定するアクセス制御情報は，パブリック ACL の dbrProp_ACL プロパティとして設定します。

パブリック ACL にユーザ定義プロパティを設定して，検索で使用できます。パブリック ACL のプロパティについては，「3.10.5(4) パブリック ACL のプロパティ」を参照してください。

なお，パブリック ACL には，自身をアクセス制御するための ACFlag やローカル ACL はありません。パブリック ACL に対するアクセス権を次の表に示します。

表 3-14　パブリック ACL に対するアクセス権

実行できる操作	対　象
作成	文書空間にオブジェクト作成権限を持つユーザ
参照	すべてのユーザ
ローカル ACL とユーザ定義プロパティの変更	セキュリティ ACL でアクセス制御情報変更権が与えられているユーザ，パブリック ACL の所有者およびセキュリティ管理者
セキュリティ ACL の変更	パブリック ACL の所有者およびセキュリティ管理者
削除	パブリック ACL の所有者およびセキュリティ管理者

3.10.4　アクセス制御情報の管理

ここでは，アクセス制御情報の管理について説明します。

DocumentBroker では，ACFlag や ACL を使用して，文書空間オブジェクトに対するユーザのアクセスを制御します。文書空間オブジェクトに対してアクセス権を保持しないユーザでも，ACFlag や ACL を変更すればアクセス権を取得できます。このため，ACFlag や ACL は，適切な権利を持つユーザだけが変更でき，それ以外のユーザからは変更されないように制御する必要があります。アクセス制御情報を変更するための権利を**アクセス制御情報変更権**といいます。DocumentBroker では，デフォルトの状態では，文書空間オブジェクトの所有者およびセキュリティ管理者だけがアクセス制御情報変更権を持っていますが，所有者およびセキュリティ管理者が必要に応じて，アクセス制御情報変更権を任意のユーザやグループに対して与えることもできます。

(1) アクセス制御情報へのアクセス権の設定

文書空間オブジェクトごとに「ユーザ A およびグループ B に」,「アクセス制御情報を変更する権利を与える」のように設定して，その文書空間オブジェクトのアクセス制御情報を変更する権利を任意のユーザまたはグループに対して与えられます。アクセス制御情報変更権を設定・更新できるのは，その文書空間オブジェクトの所有者およびセキュリティ管理者だけです。ただし，所有者が，設定されている値を参照する場合は，基本プロパティ参照権が必要です。

(a) アクセス制御情報変更権（DbjDef.PERM_CHANGE_PERM）

アクセス制御情報変更権は，アクセス制御情報に対する操作を許可するパーミッションです。アクセス制御情報変更権は，セキュリティ ACL に設定します。

アクセス制御情報変更権を与えられたユーザまたはグループは，基本プロパティ更新権を与えられていなくても，文書空間オブジェクトの ACFlag の値の更新，ローカル ACL の値の更新および文書空間オブジェクトのパブリック ACL へのバインド・アンバインドの権利を持ちます。ただし，値を参照する場合は，基本プロパティ参照権が必要です。なお，バインドされているパブリック ACL の内容を更新する権利は，パブリック ACL の所有者，パブリック ACL のセキュリティ ACL でアクセス制御情報変更権を設

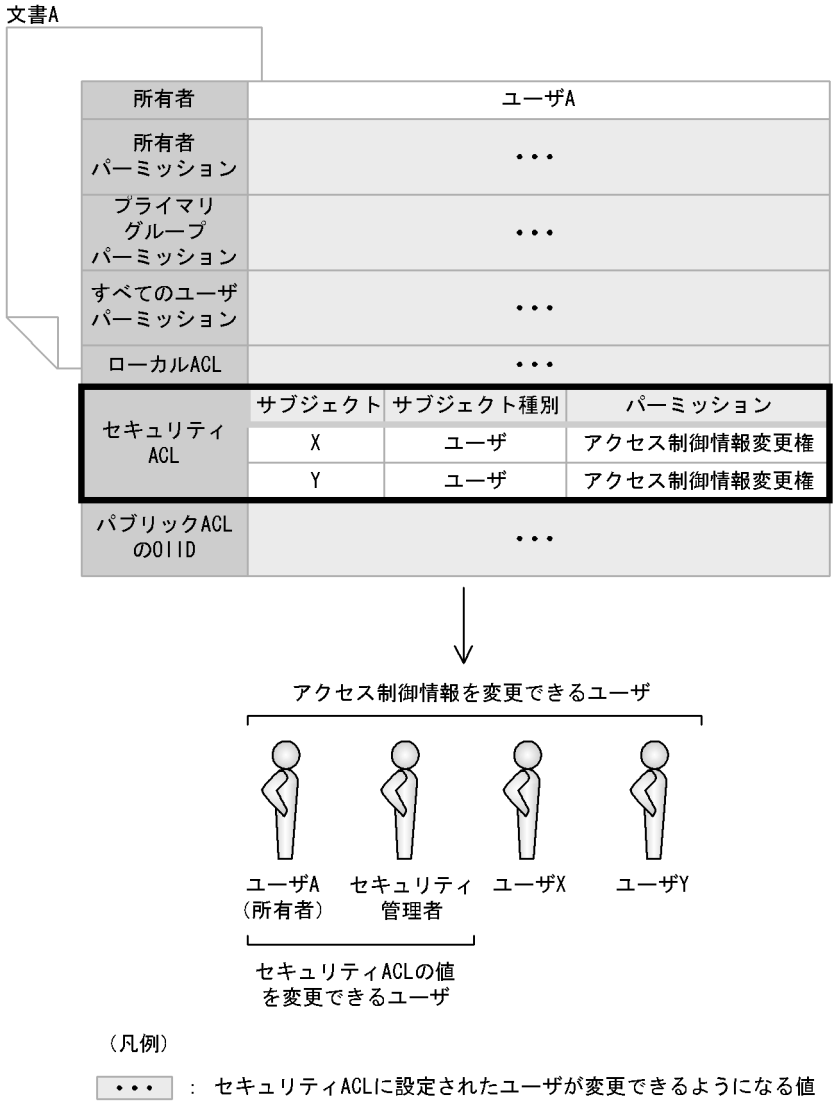
定されているユーザおよびセキュリティ管理者にあります。

(b) セキュリティ ACL

文書空間オブジェクトに設定されている ACFlag の更新，ローカル ACL の各 ACE の値の更新およびパブリック ACL へのバインド・アンバインドを実行するためのパーミッション（アクセス制御情報変更権）を設定する ACL です。セキュリティ ACL は，ACE を表すオブジェクトを構成要素とした，VARRAY 型のプロパティです。個々の文書空間オブジェクトの dbrProp_SACL プロパティとして設定します。dbrProp_SACL プロパティについては，「3.10.5(2) アクセス制御情報を設定するためのプロパティ」を参照してください。

セキュリティ ACL の ACE のパーミッションとして指定できるのはアクセス制御情報変更権だけです。また，セキュリティ ACL を設定できるのは，その文書空間オブジェクトの所有者およびセキュリティ管理者だけです。セキュリティ ACL を使用するとオブジェクトの所有者以外にもアクセス制御情報変更権を与えることができます。セキュリティ ACL の概要を次の図に示します。

図 3-51 セキュリティ ACL の概要



この例では，文書 A の所有者であるユーザ A がセキュリティ ACL を使用して，ユーザ X とユーザ Y にも

アクセス制御情報変更権を与えています。このため，文書 A の所有者であるユーザ A，セキュリティ管理者，ユーザ X およびユーザ Y の 4 人が，文書 A のアクセス制御情報を変更できます。ただし，セキュリティ ACL の値を変更できるのは，文書 A の所有者であるユーザ A およびセキュリティ管理者の 2 人だけです。

なお，パブリック ACL のセキュリティ ACL でアクセス制御情報変更権が設定されているユーザまたはグループは，パブリック ACL 内のローカル ACL の設定・更新，およびユーザ定義プロパティの値の設定・更新ができます。ただし，値を参照する場合は，基本プロパティ参照権が必要です。

(c) セキュリティ管理者およびセキュリティ運用者の設定

DocumentBroker では，アクセス権の判定を受けないで文書空間のすべてのオブジェクトに対して自由にアクセスする特権を持ち，文書空間のすべてのオブジェクトや，個々の文書空間オブジェクトに設定されたアクセス制御情報の保守を担当するユーザを設定できます。このユーザを**セキュリティ管理者**といいます。セキュリティ管理者は，DocumentBroker サーバのセキュリティ定義ファイルに定義しておきます。

また，DocumentBroker では，セキュリティ定義ファイルを保守するユーザを設定できます。このユーザを**セキュリティ運用者**といいます。セキュリティ運用者は，セキュリティ定義ファイルに，「セキュリティ管理者」，「文書空間オブジェクト作成時に ACFlag にデフォルトで設定されるパーミッションの値」および「ユーザ権限定義ファイル名」を定義します。セキュリティ定義ファイルの定義方法については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

3.10.5 アクセス制御モデルで使用するプロパティ

ここでは，アクセス制御モデルで使用するプロパティについて説明します。

(1) ユーザ情報を表すプロパティ

アクセス制御モデルでは，ユーザがログインした際に生成されるユーザ情報からログインユーザを識別したり，DocumentBroker サーバで定義されているログインユーザが保持するユーザ権限の情報を取得したりできます。ユーザ情報には，次のものがあります。

- ユーザ識別子
- 所属グループ
  - 所属するグループの数
  - 所属するグループのグループ識別子の一覧
- 特権
- ユーザ権限

各情報の詳細については，「3.10.1(1) ユーザ情報」を参照してください。それぞれのユーザ情報には，プロパティの名前が付けられています。ユーザ情報を表すプロパティの一覧を次の表に示します。

表 3-15　ユーザ情報を表すプロパティの一覧

意味	プロパティの名前	データ型
ユーザ識別子	dbrProp_UserId	STR 型
所属するグループの数	dbrProp_GroupCount	INT 型
所属するグループのグループ識別子の一覧	dbrProp_GroupList	STRLIST 型
特権	dbrProp_UserPrivilege	INT 型
ユーザ権限	dbrProp_UserPermission	INT 型

(2) アクセス制御情報を設定するためのプロパティ

個々の文書空間オブジェクトにアクセス制御情報を設定するためのプロパティを次の表に示します。なお，VARRAY 型のプロパティについては，構成要素のプロパティを「VARRAY 型のプロパティの名前：構成要素のプロパティの名前」と表記します。

表 3-16 アクセス制御情報を設定するためのプロパティの一覧

意味		プロパティの名前	データ型	制限値	値を更新できるユーザ
所有者のユーザ識別子		dbrProp_OwnerId	STR 型	1 ～ n※1 バイト	所有者※3 セキュリティ管理者
プライマリグループのグループ識別子		dbrProp_PrimaryGroupId	STR 型	0 ～ n※2 バイト	所有者※3 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3※4
ACFlag の所有者のパーミッション		dbrProp_OwnerPermission	INT 型	4 バイト※5	所有者※3 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3※4
ACFlag のプライマリグループのパーミッション		dbrProp_PrimaryGroupPermission	INT 型	4 バイト※5	所有者※3 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3※4
ACFlag のすべてのユーザのパーミッション		dbrProp_EveryonePermission	INT 型	4 バイト※5	所有者※3 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3※4
ローカル ACL	サブジェクト	dbrProp_ACL:dbrProp_Subject	STR 型	ACE は 64 個まで	所有者※3 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3※4
	サブジェクト種別	dbrProp_ACL:dbrProp_SubjectType	INT 型		
	パーミッション	dbrProp_ACL:dbrProp_Permission	INT 型		
セキュリティ ACL	サブジェクト	dbrProp_SACL:dbrProp_Subject	STR 型	ACE は 64 個まで	所有者※3 セキュリティ管理者
	サブジェクト種別	dbrProp_SACL:dbrProp_SubjectType	INT 型		
	パーミッション	dbrProp_SACL:dbrProp_Permission	INT 型		
バインドしているパブリック ACL の個数		dbrProp_PublicACLCount	INT 型	10 個	―※6
バインドしているパブリック ACL の OIID のリスト	バインドしているパブリック ACL の OIID	dbrProp_PublicACLIds:dbrProp_ACLIdElem	STR 型	―※7	所有者※3 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3※4
ログインユーザのパーミッション		dbrProp_UserPermission	INT 型	―	―※6

（凡例） －：該当しないことを示します。

注※ 1

n は，ユーザ識別子の最大長になります。通常，ユーザ識別子の最大長は 254 バイトですが，次に示すコマンドでユーザ識別子の最大長を拡張できます。

- メタ情報の初期設定コマンド（EDMInitMeta）
  この場合，-n オプションの指定値がユーザ識別子の最大長になります。
- 文書空間の定義コマンド（EDMCDefDocSpace）
  この場合，-s オプションに指定する文書空間情報ファイル中の UserIDMaxLength エントリの指定値がユーザ識別子の最大長になります。

メタ情報の初期設定コマンド（EDMInitMeta）および文書空間の定義コマンド（EDMCDefDocSpace）については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

注※ 2

n は，グループ識別子の最大長になります。通常，グループ識別子の最大長は 254 バイトですが，次に示すコマンドでグループ識別子の最大長を拡張できます。

- メタ情報の初期設定コマンド（EDMInitMeta）
  この場合，-g オプションの指定値がグループ識別子の最大長になります。
- 文書空間の定義コマンド（EDMCDefDocSpace）
  この場合，-s オプションに指定する文書空間情報ファイル中の GroupIDMaxLength エントリの指定値がグループ識別子の最大長になります。

メタ情報の初期設定コマンド（EDMInitMeta）および文書空間の定義コマンド（EDMCDefDocSpace）については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

注※ 3

値を更新する場合，基本プロパティ更新権は必要ありません。ただし，値を参照する場合は，基本プロパティ参照権が必要です。

注※ 4

セキュリティ ACL でアクセス制御情報変更権（DbjDef.PERM_CHANGE_PERM）を与えられているユーザを示します。

注※ 5

パーミッションを表す定数を指定します。パーミッションに指定する値については，「3.10.2　アクセス制御情報に設定できるパーミッションの種類」を参照してください。

注※ 6

読み取り専用のプロパティです。

注※ 7

一つの文書空間オブジェクトにバインドできるパブリック ACL は 10 個までになります。また，個々のパブリック ACL で設定できる ACE は 64 個までになります。

各プロパティの詳細について説明します。

所有者のユーザ識別子（dbrProp_OwnerId）

文書空間オブジェクト作成時に，作成したユーザのユーザ識別子が設定されるプロパティです。この

プロパティに設定されているユーザ識別子のユーザには，所有者として，ACFlag でアクセス権を与えることができます。また，所有者は，基本プロパティ更新権の有無に関係なく，その文書空間オブジェクトのアクセス制御情報およびアクセス制御情報変更権を設定・更新できます。ただし，値を参照する場合は，基本プロパティ参照権が必要です。
基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。また，属性検索条件にも指定できます。例えば，「A さんが所有する文書を検索する」という条件で検索できます。ただし，検索条件に指定する場合は，「edmProp_OwnerId」という名前で指定してください。
このプロパティの値を更新できるのは，所有者およびセキュリティ管理者だけです。所有者およびセキュリティ管理者以外のユーザは，基本プロパティ更新権があっても，このプロパティの値を更新できません。

プライマリグループのグループ識別子（dbrProp_PrimaryGroupId）

ACFlag でアクセス権を与えるグループのグループ識別子を設定するプロパティです。設定できるグループは一つだけです。このプロパティに設定されているグループ識別子のグループは，その文書空間オブジェクトに対して「ACFlag のプライマリグループのパーミッション」で設定された範囲のアクセス権を保持します。ユーザ管理システムが LDAP 対応のディレクトリサービスの場合は，オブジェクトを作成したあとに，その文書空間オブジェクトの所有者またはセキュリティ管理者がユーザ情報のグループ識別子の中から一つ選択して，設定してください。UNIX のパスワードファイルの場合は，文書空間オブジェクトを作成するときに UNIX のパスワードファイルに記述されているグループ識別子が設定されます。
基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。また，属性検索条件にも指定できます。例えば，「プライマリグループが A グループの文書を検索する」という条件で検索できます。ただし，検索条件に指定する場合は，「edmProp_PrimaryGroupId」という名前で指定してください。
このプロパティの値を更新できるのは，所有者およびセキュリティ管理者だけです。所有者およびセキュリティ管理者以外のユーザは，基本プロパティ更新権があっても，このプロパティの値を更新できません。

ACFlag **の所有者のパーミッション**（dbrProp_OwnerPermission）

「所有者のユーザ識別子」に設定されているユーザに対して与えるアクセス権の範囲を定めるパーミッションを設定するプロパティです。文書空間オブジェクト作成時には，セキュリティ定義ファイルに設定されているデフォルトのパーミッションが設定されます。セキュリティ定義ファイルについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。
基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。ただし，属性検索条件には指定できません。
このプロパティの値を更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，このプロパティの値を更新できません。

ACFlag **のプライマリグループのパーミッション**（dbrProp_PrimaryGroupPermission）

「プライマリグループのグループ識別子」に設定されているグループに対して与えるアクセス権の範囲を定めるパーミッションを設定するプロパティです。文書空間オブジェクト作成時には，セキュリティ定義ファイルに設定されているデフォルトのパーミッションが設定されます。セキュリティ定義ファイルについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。
「プライマリグループのグループ識別子」が設定されていない場合は，このプロパティに値を設定しても無視されます。
基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。ただし，属性検索条件

には指定できません。
このプロパティの値を更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，このプロパティの値を更新できません。

ACFlag のすべてのユーザのパーミッション（dbrProp_EveryonePermission）
すべてのユーザに対して与えるアクセス権の範囲を定めるパーミッションを設定するプロパティです。文書空間オブジェクト作成時には，セキュリティ定義ファイルに設定されているデフォルトのパーミッションが設定されます。セキュリティ定義ファイルについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。
基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。ただし，属性検索条件には指定できません。
このプロパティの値を更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，このプロパティの値を更新できません。

ローカル ACL（dbrProp_ACL）
ローカル ACL を設定するプロパティです。ローカル ACL は「サブジェクト」,「サブジェクト種別」および「パーミッション」という三つの要素を持つ ACE を複数管理する VARRAY 型のプロパティです。ローカル ACL の構成要素である ACE のプロパティについて説明します。

サブジェクト（dbrProp_ACL:dbrProp_Subject）
ローカル ACL の ACE でアクセス権を与えるユーザの識別子，グループの識別子またはシステムサブジェクトを表す定数を設定するプロパティです。

サブジェクト種別（dbrProp_ACL:dbrProp_SubjectType）
ローカル ACL の ACE でアクセス権を与えるユーザまたはグループの種別を設定するプロパティです。

パーミッション（dbrProp_ACL:dbrProp_Permission）
「サブジェクト」に設定したユーザまたはグループに与えるアクセス権の範囲を定めるパーミッションを設定するプロパティです。

これらのプロパティに設定する値の詳細については，「(3) アクセス制御情報の構成要素のプロパティ」を参照してください。
基本プロパティ参照権を持つユーザは，これらのプロパティの値を参照できます。ただし，属性検索条件には指定できません。また，これらのプロパティの値を設定・更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，これらのプロパティの値を更新できません。

セキュリティ ACL（dbrProp_SACL）
セキュリティ ACL を設定するプロパティです。セキュリティ ACL は「サブジェクト」,「サブジェクト種別」および「パーミッション」という三つの要素を持つ ACE を複数管理する VARRAY 型のプロパティです。セキュリティ ACL の構成要素である ACE のプロパティについて説明します。

サブジェクト（dbrProp_SACL:dbrProp_Subject）
セキュリティ ACL の ACE でアクセス制御情報変更権のパーミッションを設定してアクセス制御情報へのアクセス権を与えるユーザの識別子，グループの識別子またはシステムサブジェクトを表す定数を設定するプロパティです。

サブジェクト種別（dbrProp_SACL:dbrProp_SubjectType）

セキュリティ ACL の ACE でアクセス制御情報変更権のパーミッションを設定してアクセス制御情報へのアクセス権を与えるユーザまたはグループの種別を設定するプロパティです。

パーミッション（dbrProp_SACL:dbrProp_Permission）

「サブジェクト」に指定したユーザまたはグループに与えるアクセス権の範囲を定めるパーミッションを設定するプロパティです。このプロパティに設定できる値は，アクセス制御情報変更権を表す定数「DbjDef.PERM_CHANGE_PERM」です。

「サブジェクト」および「サブジェクト種別」に設定する値の詳細については「(3) アクセス制御情報の構成要素のプロパティ」を参照してください。

基本プロパティ参照権を持つユーザは，これらのプロパティの値を参照できます。ただし，属性検索条件には指定できません。また，これらのプロパティの値を設定・更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，これらのプロパティの値を更新できません。

バインドしているパブリック ACL **の個数**（dbrProp_PublicACLCount）

文書空間オブジェクトがバインドしているパブリック ACL の個数が設定されるプロパティです。

基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。ただし，属性検索条件には指定できません。

バインドしているパブリック ACL **の** OIID **のリスト**（dbrProp_PublicACLIds）

文書空間オブジェクトがバインドしているパブリック ACL の OIID のリストを表す VARRAY 型のプロパティです。構成要素であるパブリック ACL の OIID のプロパティについて説明します。

バインドしているパブリック ACL **の** OIID（dbrProp_PublicACLIds:dbrProp_ACLIdElem）

基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。ただし，属性検索条件には指定できません。

このプロパティの値を設定・更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，このプロパティの値を更新できません。

ログインユーザのパーミッション（dbrProp_UserPermission）

文書空間オブジェクトに対するログインユーザのアクセス権の範囲を定めるパーミッションが設定されるプロパティです。文書空間オブジェクトの ACFlag，ローカル ACL およびバインドしているパブリック ACL のアクセス制御情報から，文書空間オブジェクトに操作要求を出したログインユーザまたはログインユーザが所属するグループに与えられているアクセス権の範囲を定めるパーミッションの論理和が求められて設定されます。

(3) アクセス制御情報の構成要素のプロパティ

ここでは，アクセス制御情報のうち，ACL の構成要素である ACE のプロパティについて説明します。

ACE のプロパティを次の表に示します。

表 3-17 ACE のプロパティ

意味	プロパティの名前	データ型	制限値
サブジェクト	dbrProp_Subject	STR 型	サブジェクト種別に応じて制限値が異なります。 • ユーザ識別子の場合 1 ～ユーザ識別子の最大長※（単位はバイト）。 • グループ識別子の場合 1 ～グループ識別子の最大長※（単位はバイト）。 • システムサブジェクトの場合 1 ～ 254 バイト
サブジェクト種別	dbrProp_SubjectType	INT 型	4 バイト
パーミッション	dbrProp_Permission	INT 型	4 バイト

注※

通常，ユーザ識別子およびグループ識別子の最大長は 254 バイトですが，次に示すコマンドでユーザ識別子とグループ識別子の最大長を拡張できます。

- メタ情報の初期設定コマンド（EDMInitMeta）
  この場合，-n オプションの指定値がユーザ識別子の最大長に，-g オプションの指定値がグループ識別子の最大長になります。
- 文書空間の定義コマンド（EDMCDefDocSpace）
  この場合，-s オプションに指定する文書空間情報ファイル中の UserIDMaxLength エントリの指定値がユーザ識別子の最大長に，GroupIDMaxLength エントリの指定値がグループ識別子の最大長になります。

メタ情報の初期設定コマンド（EDMInitMeta）および文書空間の定義コマンド（EDMCDefDocSpace）については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

(a) サブジェクト（dbrProp_Subject）

次のどれかを指定します。

● ユーザ識別子

ユーザ単位でパーミッションを設定する場合に指定します。

● グループ識別子

グループ単位でパーミッションを設定する場合に指定します。

● システムサブジェクトを表す定数

システムサブジェクトに対してパーミッションを設定する場合に指定します。
システムサブジェクトは，DocumentBroker によって設定された情報に対してパーミッションを設定する場合に指定します。このサブジェクトは，主にパブリック ACL に指定して使用します。
システムサブジェクトの定数と意味を次の表に示します。

表 3-18 システムサブジェクトの定数と意味

定 数	意 味
DbjDef.SYSSUBJECT_SELF	対象の文書空間オブジェクトの所有者を表すサブジェクトです。 • パブリック ACL の ACL に設定した場合，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文書空間オブジェクトの所有者を示します。 • 文書やフォルダを表す文書空間オブジェクトのローカル ACL およびセキュリティ ACL に設定した場合は，その文書空間オブジェクトの所有者を示します。 • パブリック ACL のセキュリティ ACL に設定した場合は，パブリック ACL の所有者を示します。
DbjDef.SYSSUBJECT_EVERYONE	すべてのユーザを表すサブジェクトです。

(b) サブジェクト種別（dbrProp_SubjectType）

「サブジェクト」に指定した内容に応じて，種別を示す定数を指定します。

● ユーザ識別子を指定した場合
定数「DbjDef.SUBJECTTYPE_USR」を指定します。

● グループ識別子を指定した場合
定数「DbjDef.SUBJECTTYPE_GRP」を指定します。

● システムサブジェクトを表す定数を指定した場合
定数「DbjDef.SUBJECTTYPE_SYS」を指定します。

(c) パーミッション（dbrProp_Permission）

「サブジェクト」に指定したユーザまたはグループに対するパーミッションを表す定数を指定します。

パーミッションに指定する値については，「3.10.2 アクセス制御情報に設定できるパーミッションの種類」を参照してください。

(4) パブリック ACL のプロパティ

パブリック ACL のプロパティを次の表に示します。なお，VARRAY 型のプロパティについては，構成要素のプロパティを「VARRAY 型のプロパティの名前：構成要素のプロパティの名前」と表記します。

表 3-19 パブリック ACL のプロパティの一覧

意味		プロパティの名前	データ型	制限値	値を更新できるユーザ
オブジェクトの識別子		dmaProp_OIID	STR 型	－	＿※1
所有者のユーザ識別子		dbrProp_OwnerId	STR 型	1 ～ n※2 バイト	所有者 セキュリティ管理者
ローカル ACL	サブジェクト	dbrProp_ACL:dbrProp_Subject	STR 型	ACE は 64 個まで	所有者 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3
	サブジェクト種別	dbrProp_ACL:dbrProp_SubjectType	INT 型		
	パーミッション	dbrProp_ACL:dbrProp_Permission	INT 型		
セキュリティ ACL	サブジェクト	dbrProp_SACL:dbrProp_Subject	STR 型	ACE は 64 個まで	所有者 セキュリティ管理者

意味		プロパティの名前	データ型	制限値	値を更新できるユーザ
	サブジェクト種別	dbrProp_SACL:dbrProp_SubjectType	INT 型		
	パーミッション	dbrProp_SACL:dbrProp_Permission	INT 型		
パブリック ACL をバインドしている文書空間オブジェクトの個数		dbrProp_BindObjectCount	INT 型	－	＿※1
ログインユーザのパーミッション		dbrProp_UserPermission	INT 型	－	＿※1
ユーザ定義プロパティ		（ユーザ定義プロパティの識別子）	－	－	所有者 セキュリティ管理者 DbjDef.PERM_CHANGE_PERM※3

（凡例） －：該当しないことを示します。

注※ 1

読み取り専用のプロパティです。

注※ 2

n は，ユーザ識別子の最大長になります。通常，ユーザ識別子の最大長は 254 バイトですが，次に示すコマンドでユーザ識別子の最大長を拡張できます。

- メタ情報の初期設定コマンド（EDMInitMeta）
  この場合，-n オプションの指定値がユーザ識別子の最大長になります。
- 文書空間の定義コマンド（EDMCDefDocSpace）
  この場合，-s オプションに指定する文書空間情報ファイル中の UserIDMaxLength エントリの指定値がユーザ識別子の最大長になります。

メタ情報の初期設定コマンド（EDMInitMeta）および文書空間の定義コマンド（EDMCDefDocSpace）については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

注※ 3

セキュリティ ACL でアクセス制御情報変更権（DbjDef.PERM_CHANGE_PERM）を与えられているユーザを示します。

各プロパティの詳細について説明します。

オブジェクトの識別子（dmaProp_OIID）

パブリック ACL を識別するための OIID 文字列が設定されるプロパティです。パブリック ACL 作成時に，DocumentBroker サーバが値を設定します。この OIID を使用して，パブリック ACL をほかの文書空間オブジェクトからバインドさせたり，すでに作成されているパブリック ACL を指定してプロパティの値を設定・参照したりできます。また，属性検索条件にも指定できます。

所有者のユーザ識別子（dbrProp_OwnerId）

パブリック ACL を作成したユーザのユーザ識別子が設定されるプロパティです。このプロパティにユーザ識別子が設定されているユーザは，アクセス権の有無に関係なく，次の操作ができます。

- パブリック ACL のアクセス制御情報の設定・更新
- アクセス制御情報変更権の設定・更新
- ユーザ定義プロパティの設定・更新

- パブリック ACL の削除

このプロパティの値は，属性検索条件にも指定できます。例えば，「A さんが所有するパブリック ACL を検索する」という条件で検索できます。ただし，検索条件に指定する場合は，「edmProp_OwnerId」という名前で指定してください。
このプロパティの値を設定・更新できるのは，所有者およびセキュリティ管理者だけです。

ローカル ACL（dbrProp_ACL）

複数の文書空間オブジェクト間で共有したいアクセス制御情報を設定するプロパティです。ローカル ACL は「サブジェクト」,「サブジェクト種別」および「パーミッション」という三つの要素を持つ ACE を複数管理する VARRAY 型のプロパティです。ローカル ACL の構成要素である ACE のプロパティについて説明します。

サブジェクト（dbrProp_ACL:dbrProp_Subject）

アクセス権を与えるユーザの識別子，グループの識別子またはシステムサブジェクトを表す定数を設定するプロパティです。

サブジェクト種別（dbrProp_ACL:dbrProp_SubjectType）

アクセス権を与えるユーザまたはグループの種別を設定するプロパティです。

パーミッション（dbrProp_ACL:dbrProp_Permission）

「サブジェクト」に指定したユーザまたはグループに与えるアクセス権の範囲を定めるパーミッションを設定するプロパティです。

これらのプロパティに設定する値の詳細については，「(3) アクセス制御情報の構成要素のプロパティ」を参照してください。
基本プロパティ参照権を持つユーザは，これらのプロパティの値を参照できます。ただし，属性検索条件には指定できません。また，これらのプロパティの値を設定・更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，これらのプロパティの値を更新できません。

セキュリティ ACL（dbrProp_SACL）

パブリック ACL のローカル ACL およびユーザ定義プロパティの値を更新する権利を設定するプロパティです。このプロパティでアクセス制御情報変更権を与えられたユーザまたはグループは，パブリック ACL 内のローカル ACL の設定・更新およびユーザ定義プロパティの設定・更新ができます。セキュリティ ACL は「サブジェクト」,「サブジェクト種別」および「パーミッション」という三つの要素を持つ ACE を複数管理する VARRAY 型のプロパティです。セキュリティ ACL の構成要素である ACE のプロパティについて説明します。

サブジェクト（dbrProp_SACL:dbrProp_Subject）

パブリック ACL のローカル ACL およびユーザ定義プロパティの値を更新する権利を与えるユーザの識別子，グループの識別子またはシステムサブジェクトを表す定数を設定するプロパティです。

サブジェクト種別（dbrProp_SACL:dbrProp_SubjectType）

パブリック ACL を更新する権利を与えるユーザまたはグループの種別を設定するプロパティです。

パーミッション（dbrProp_SACL:dbrProp_Permission）

「サブジェクト」に指定したユーザまたはグループに与えるアクセス権の範囲を定めるパーミッションを設定するプロパティです。このプロパティに設定できる値は，アクセス制御情報変更権を表す定数「DbjDef.PERM_CHANGE_PERM」です。アクセス制御情報変更権を設定された

ユーザまたはグループは，パブリック ACL のローカル ACL の設定・更新およびユーザ定義プロパティの設定・更新ができます。

「サブジェクト」および「サブジェクト種別」に設定する値の詳細については，「(3) アクセス制御情報の構成要素のプロパティ」を参照してください。

基本プロパティ参照権を持つユーザは，これらのプロパティの値を参照できます。ただし，属性検索条件には指定できません。また，これらのプロパティの値を設定・更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザ以外のユーザは，基本プロパティ更新権があっても，これらのプロパティの値を更新できません。

パブリック ACL をバインドしている文書空間オブジェクトの個数（dbrProp_BindObjectCount）

パブリック ACL をバインドしている文書空間オブジェクトの個数が設定されるプロパティです。

基本プロパティ参照権を持つユーザは，このプロパティの値を参照できます。ただし，属性検索条件には指定できません。

ログインユーザのパーミッション（dbrProp_UserPermission）

パブリック ACL に対するログインユーザのアクセス権の範囲を定めるパーミッションが設定されるプロパティです。セキュリティ ACL から，パブリック ACL に操作要求を出したログインユーザまたはログインユーザが所属するグループに与えられているアクセス権の範囲を定めるパーミッションの論理和が求められて設定されます。

ユーザ定義プロパティ

パブリック ACL クラスには，ユーザ定義プロパティを追加定義できます。

ユーザ定義プロパティは，属性検索条件としても指定できます。

ユーザ定義プロパティの値を設定・更新できるのは，所有者，セキュリティ管理者およびセキュリティ ACL でアクセス制御情報変更権を設定されているユーザです。

3.10.6 アクセス制御モデルの運用例

ここでは，アクセス制御情報の使い分けとアクセス制御モデルの運用例を示します。

(1) アクセス制御情報の使い分け

ここでは，これまでに説明してきたアクセス制御情報の使い分けの考え方について説明します。アクセス権の判定に使用されるアクセス制御情報の処理順序や特長を考慮して，組織に合わせたアクセス制御モデルを構築してください。アクセス制御情報の処理順序と特長を次の表に示します。

表 3-20 アクセス制御情報の処理順序と特長

種類	処理順序	特長
ユーザ権限	1	• アクセス権の判定で最初に参照される情報なので，判定が高速です。 • あるユーザに対して，文書空間内で同一の操作を許可する場合に使用すると便利です。
ACFlag	2	• 所有者，プライマリグループの情報と組み合わせて判定されます。 • ユーザ権限の次に参照される情報なので，ローカル ACL やパブリック ACL に比べて早い時点で判定されます。 • 所有者，プライマリグループまたはすべてのユーザに対して許可する操作だけ設定できます。

種類	処理順序	特長
パブリック ACL	3	複数の文書空間オブジェクトに対して共通のアクセス制御情報（ローカル ACL と同様にユーザまたはグループ単位で個別に設定できる）が設定できます。したがって，アクセス権を一括して管理でき，アクセス権の保守が容易になります。 例えば，「企画部」というグループに対して参照を許可するパブリック ACL を作成して，複数の文書空間オブジェクトからこのパブリック ACL をバインドしていたとします。この場合，途中で「企画部」に対して更新も許可するような運用に変更するときは，パブリック ACL で「企画部」に対して更新を許可するパーミッションを追加すれば，このパブリック ACL をバインドしているすべての文書空間オブジェクトに変更を反映できます。
ローカル ACL	4	文書空間オブジェクト固有のアクセス制御情報をユーザまたはグループ単位で個別に設定できます。個々の文書空間オブジェクトに，状況に応じて柔軟にアクセス制御情報を設定できます。

(2) ACFlag を使用したアクセス制御の運用例

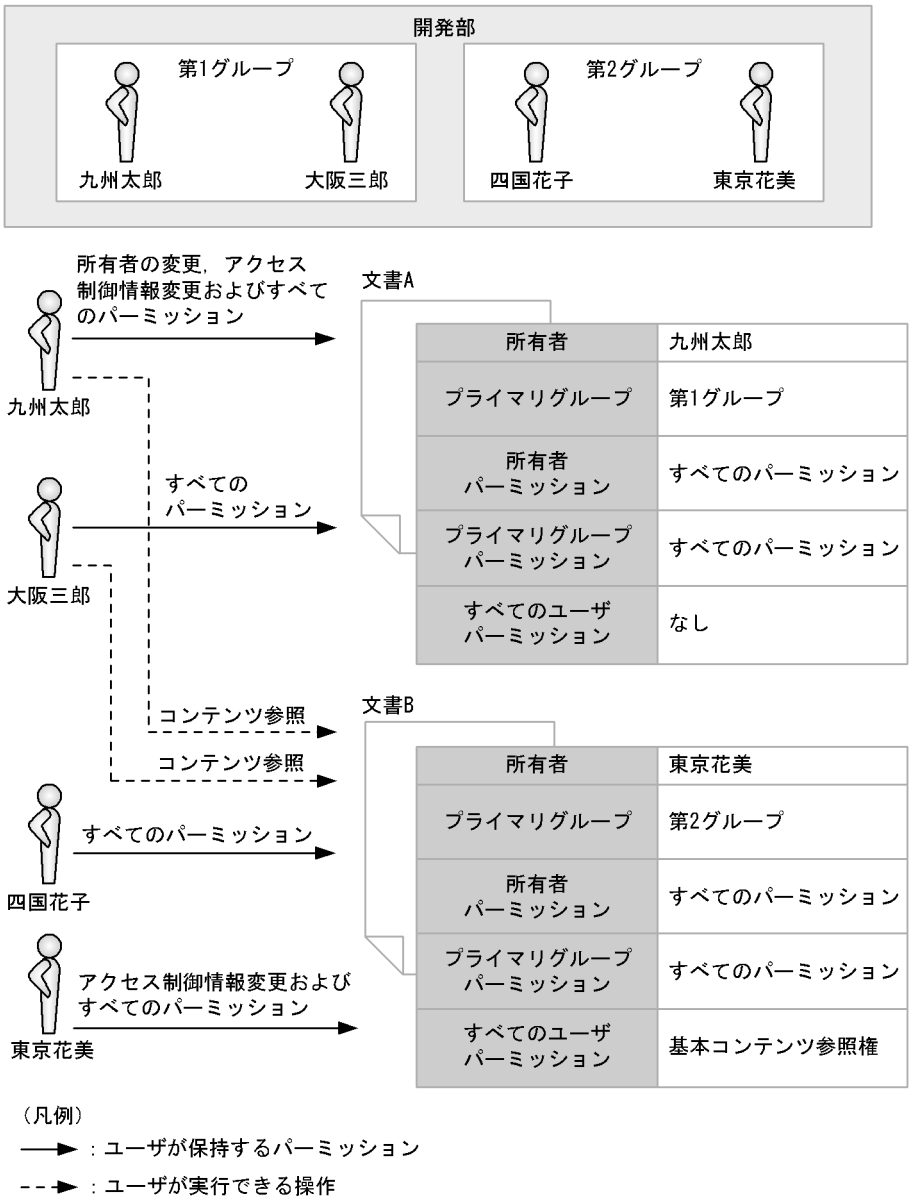
ACFlag を使用したアクセス制御の運用例について説明します。

なお，この例では，次のことを前提としています。

- すべてのユーザは文書を作成できます。
- ユーザは，所属するグループのユーザが作成した文書に対してすべてのパーミッションを組み合わせたアクセス権を与えられます。
- 文書の作成者は，文書の所有者としてアクセス制御情報変更権を保持し，作成した文書のアクセス権を設定・更新して管理します。

ACFlag を使用したアクセス制御の運用例を次の図に示します。

図 3-52　ACFlag を使用したアクセス制御の運用例



この例では，文書のライフサイクルの段階に応じて次のようにアクセスを制御することが考えられます。

文書が完成するまでの段階

文書 A のように所有者が ACFlag のすべてのユーザパーミッションを「なし」に設定しておき，所属グループ内だけに公開して作業します。

文書が完成した段階

文書 B のように所有者が ACFlag のすべてのユーザパーミッションに「基本コンテンツ参照権」を設定して，ほかのグループにも完成した文書の内容の参照を許可します。

(3) ACFlag およびローカル ACL を使用した運用例

ACFlag を使用したアクセス制御モデルで，常にグループのグループリーダーを所有者に設定して運用する運用例と，ローカル ACL もあわせて設定する運用例について説明します。

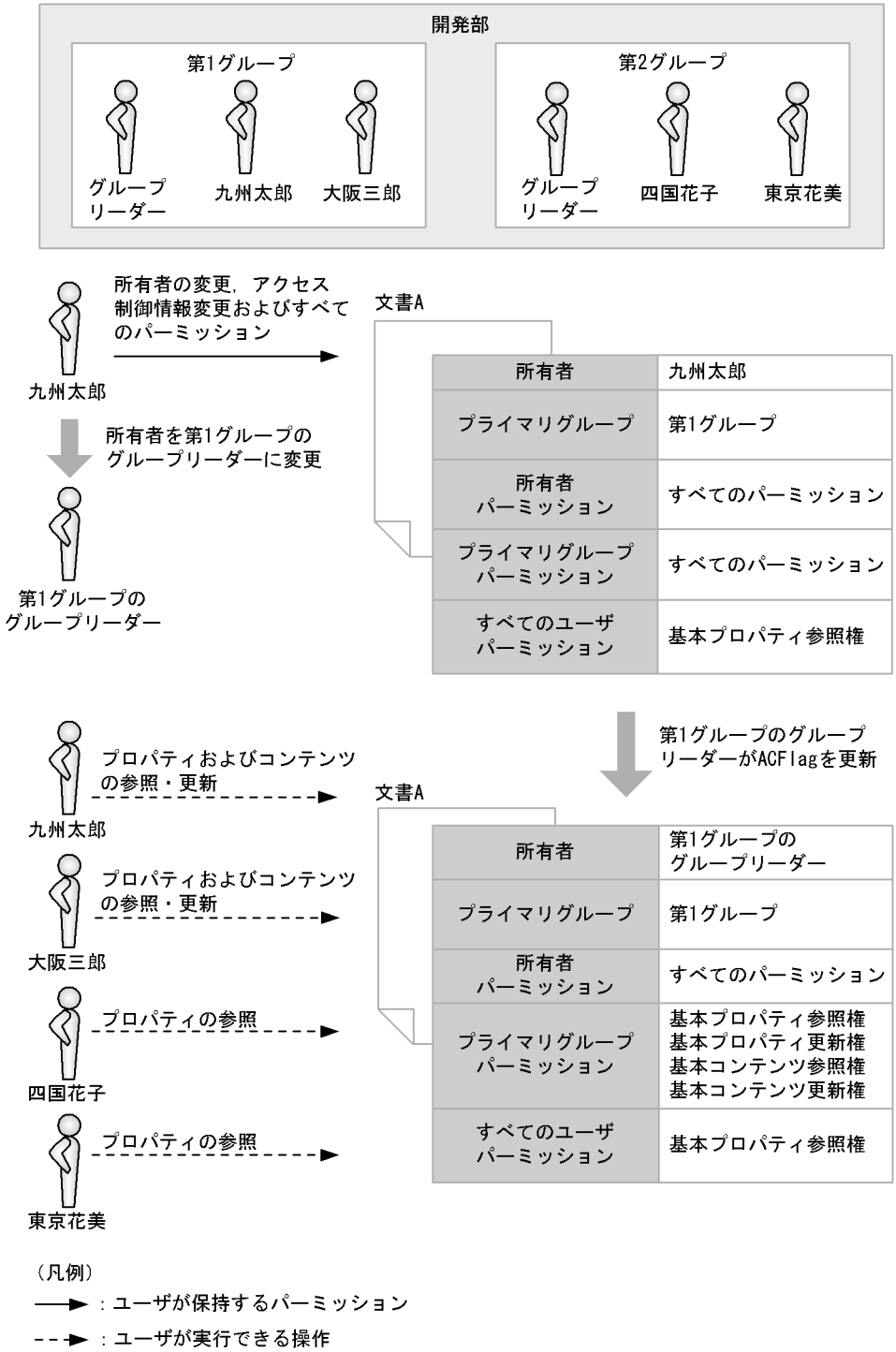
グループリーダーに管理を任せる ACFlag を使用した運用例

なお，この例では，次のことを前提としています。

- すべてのユーザは文書を作成，参照および更新できます。
- 文書の作成者は，作成後に文書の所有者をグループリーダーに変更します。
- グループリーダーはすべてのパーミッションを組み合わせたアクセス権を保持し，作成された文書を管理します。
- グループリーダー以外のユーザは作成者であっても文書を削除したり，フォルダにリンク付けたり，バージョンを管理したりすることはできません。

グループリーダーに管理を任せる ACFlag を使用した運用例を次の図に示します。

図 3-53 グループリーダーに管理を任せる ACFlag を使用した運用例



この例では，グループリーダーの判断で，ライフサイクルに応じた適切なパーミッションを設定して文書を管理しています。

ローカル ACL を併用した運用例

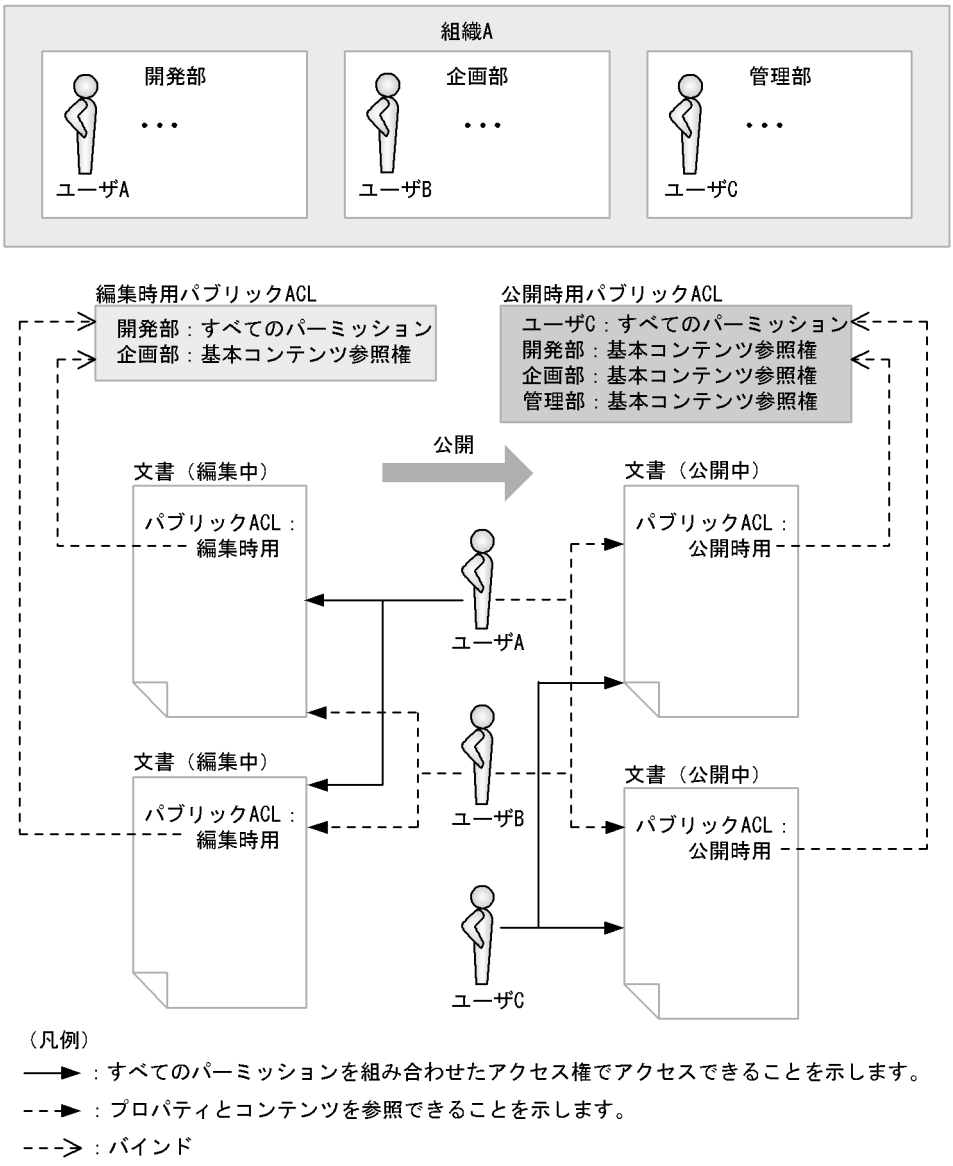
例えば，この例の運用上，文書 A について，「一定期間だけ他部署の部長（ユーザ X）にもコンテンツを参照させる必要ができた」という場合があったとします。ACFlag では特定ユーザに対するアクセス情報は設定できません。したがって，この場合は，ローカル ACL に「ユーザ X にコンテンツの

参照権を与える」と設定する運用が考えられます。

(4) パブリック ACL を使用した運用例

パブリック ACL を使用した運用例について説明します。パブリック ACL は，複数の文書やフォルダに対してアクセス制御情報を共通して設定したい場合に使用します。パブリック ACL を使用した運用例を次の図に示します。

図 3-54 パブリック ACL を使用した運用例



例えば，組織 A で，「『開発部』のユーザが作成した文書は，『開発部』内のすべてのユーザがすべてのパーミッションを組み合わせたアクセス権を与えられて更新や削除ができるようにして，『企画部』内のすべてのユーザがコンテンツの参照だけできるようにしたい」という場合があったとします。このとき，パブリック ACL を一つ作成して，「グループ『開発部』に対してすべてのパーミッションを組み合わせたアクセス権を与える」という ACE と「グループ『企画部』に対してコンテンツの参照権を与える」という ACE を設定します。『開発部』が作成したすべての文書に対してこのパブリック ACL を設定すれば，すべての文書に同じアクセス制御情報が設定できます。

なお，パブリック ACL は複数の文書空間オブジェクトで共有されますので，作成時に明確な運用方法を決めてからアクセス制御情報を設定してください。複数の文書空間オブジェクトからパブリック ACL をバインドしている場合に，特定の文書空間オブジェクトのアクセス制御情報を変更する必要が生じたときは，そのパブリック ACL の内容を変更するのではなく，アクセス制御情報を変更する必要がある文書空間オブジェクトだけ別のパブリック ACL にバインドし直すという運用をお勧めします。

例えば，「文書が完成するまでの文書の編集段階では『開発部』にすべてのパーミッションを組み合わせたアクセス権を与え，『企画部』にコンテンツの参照だけを許可したいが，一度公開したあとでは『管理部』内の特定のユーザ（ユーザ C）以外には更新させないようにしたい」というような場合は，公開時にパブリック ACL の内容を変更するのではなく，編集時用のパブリック ACL と，公開時用のパブリック ACL をそれぞれ作成しておく運用が考えられます。ある文書を公開したら，その文書にバインドするパブリック ACL を，編集時用から公開時用にバインドし直すことで，アクセス制御情報を変更できます。

（5）セキュリティ ACL を使用した運用例

アクセス制御情報は，デフォルトでは，その文書空間オブジェクトの所有者およびセキュリティ管理者だけが設定・更新できます。セキュリティ ACL を使用すると，文書やフォルダに設定された ACFlag の更新，ローカル ACL の更新およびパブリック ACL のバインド・アンバインドの権利を所有者およびセキュリティ管理者以外のユーザにも与えることができます。

例えば，パブリック ACL を使用した運用で，文書のライフサイクルの段階に応じてパブリック ACL をバインドし直すという運用をしている場合，文書空間オブジェクトの所有者以外のユーザにパブリック ACL のバインド・アンバインドの管理を任せるために，セキュリティ ACL を使用することが考えられます。例えば，「『開発部』が作成した文書を，『管理課』の特定ユーザ（ユーザ Y）が，公開時に『公開時用パブリック ACL』をバインドして公開する」という場合，開発部が作成する文書のセキュリティ ACL に「ユーザ Y にアクセス制御情報変更権を与える」と設定しておきます。

3.11 排他制御モデル

この節では，Java クラスライブラリでの排他制御について説明します。

Java クラスライブラリでは，HiRDB のロック機能を使用して文書空間オブジェクトに対する排他制御を実現します。このため，文書空間オブジェクトに設定されるロックは，一つのトランザクション内だけで有効です。通常，一つのメソッドが一つのトランザクションに対応し，メソッド実行時に，Java クラスライブラリが文書空間オブジェクトに対してロックを設定します。また，ユーザがトランザクションの開始と終了を指定することもできます。文書空間オブジェクトに設定されるロック種別は，使用するインターフェースおよびメソッドによって異なりますが，ロック種別をユーザが明示的に指定してオブジェクトにアクセスすることもできます。

なお，ロックの操作の詳細については，「6.8.6 アクセス方法に関する情報の取得と変更」を参照してください。

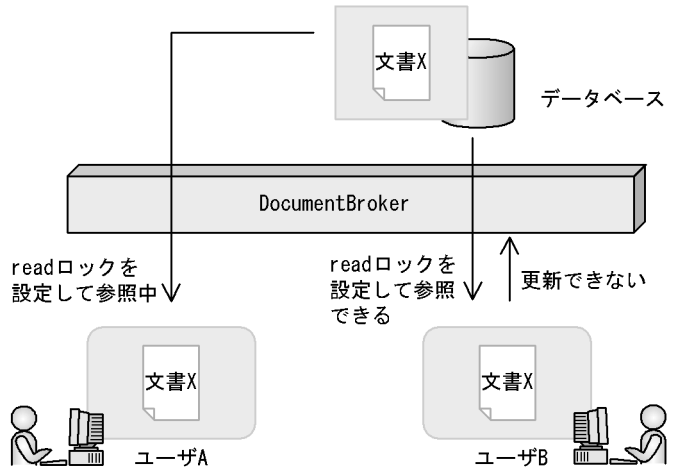
3.11.1 ロック種別

Java クラスライブラリでは，次に示すロック種別があります。

● read ロック

文書空間オブジェクトを参照する場合に設定されるロックです。あるユーザが read ロックを設定して取り出した文書空間オブジェクトに対して，ほかのユーザは write ロックを設定してその文書空間オブジェクトを更新，削除することはできません。ただし，read ロックを設定してその文書空間オブジェクトを参照することはできます。read ロックを設定した文書空間オブジェクトの操作を次の図に示します。

図 3-55 read ロックを設定した文書空間オブジェクトの操作



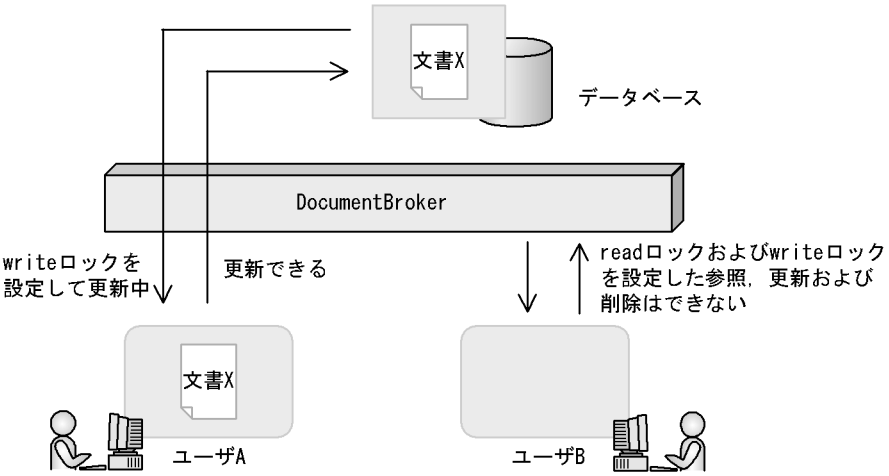
ユーザ A がほかのユーザより先に文書 X に read ロックを設定して参照している場合，ほかのユーザは文書 X に read ロックを設定して参照することはできますが，write ロックを設定して更新および削除することはできません。

● write ロック

文書空間オブジェクトを更新または削除する場合に設定されるロックです。あるユーザが write ロックを設定して取り出した文書空間オブジェクトに対して，ほかのユーザは read ロックまたは write ロックを設定できません。したがって，write ロックを設定した以外のユーザが，read ロックまたは write

ロックを設定してその文書空間オブジェクトを参照，更新または削除することはできません。write ロックを設定した文書空間オブジェクトの操作を次の図に示します。

図 3-56 write ロックを設定した文書空間オブジェクトの操作



ユーザ A がほかのユーザより先に文書 X に write ロックを設定して更新または削除しようとしている場合，ほかのユーザは文書 X に read ロックを設定して参照したり，write ロックを設定して更新および削除したりすることはできません。

3.11.2 ロックによる排他制御

あるユーザ（ユーザ A）が，ある文書空間オブジェクト（文書 X）に対してロックを設定している状態を，「ユーザ A が文書 X のロックを取得している」と表現します。Java クラスライブラリでは，同じ文書空間オブジェクトに対して，あるユーザ（ユーザ A）がすでに取得しているロックと，ほかのユーザ（ユーザ B）が取得するロックが競合した場合，ユーザ B は，ユーザ A が取得しているロック種別によって，次の表で示すように排他制御されます。

表 3-21 ロックが競合した場合のロックの関係

ユーザ A が取得したロック		read	write
ユーザ B が取得できるロック	read	○	×
	write	×	×

（凡例）
○：ロックを設定するメソッドを実行できます。
×：ロックを設定できないため，実行するメソッドは，ロックが解除されるまで待ち状態（WAIT モード）になります。

3.11.3 ロックが設定されているオブジェクトに対する接続

Java クラスライブラリでは，WAIT モードでメソッドを実行します。例えば，ユーザ A がある文書空間オブジェクトのロックを取得しようとした場合に，すでにユーザ B がその文書空間オブジェクトのロックを取得していたとします。このとき，ロックが解除されるまで，ユーザ A が実行しようとしたメソッドは待ち状態となります。ただし，タイムアウトやデッドロックが発生した場合は，処理を終了します。なお，ロックは次の時点まで継続します。

- セッションの切断

- トランザクションの終了

なお，WAIT モードでのタイムアウト時間は HiRDB に設定します。HiRDB のタイムアウト時間の設定については，マニュアル「HiRDB　UAP 開発ガイド」を参照してください。

3.11.4 ロック種別の変更

メソッド実行時に文書空間オブジェクトに設定されるロック種別は，使用するインターフェースおよびメソッドによって異なります。参照系メソッドを実行する場合，デフォルトの設定では，文書空間オブジェクトに対して read ロックが設定されます。Java クラスライブラリでは，このロック種別を write ロックに変更して文書空間オブジェクトにアクセスできます。

3.11.5 ロックに関する注意事項

ここでは，ロックに関する注意事項について説明します。

(1) ロックが設定される範囲

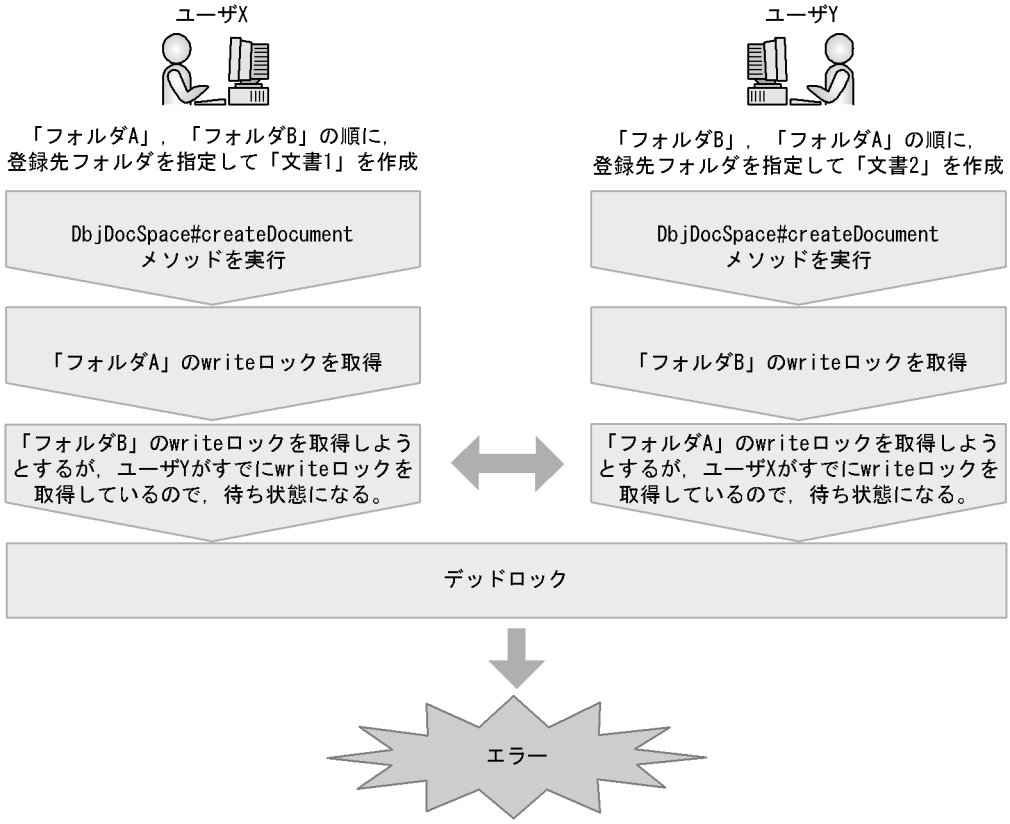
Java クラスライブラリでは，メソッド実行時に，文書空間オブジェクトを構成する DMA オブジェクトのトップオブジェクトに対してロックが設定されます。また，トップオブジェクト以外の DMA オブジェクトについては，メソッドの処理に必要なオブジェクトに対してロックが設定されます。このため，次の場合，トップオブジェクト以外の DMA オブジェクトに対するロックを，ほかのユーザが取得できてしまうことがあります。

- 文書空間オブジェクトを構成する DMA オブジェクトのロックを取得していても，ほかのユーザが，DbjDocSpace#executeSearch メソッドを使用して検索結果に対してロックを取得すると，トップオブジェクト以外の DMA オブジェクトのロックを取得できてしまう場合があります。
- バージョニングオブジェクトに対してロックを取得していても，ほかのユーザは，バージョンオブジェクトのロックを取得できてしまう場合があります。

(2) 複数のオブジェクトに write ロックを設定するメソッドの使用上の注意

複数の文書空間オブジェクトに write ロックを設定することがあるメソッドを使用して，複数の文書空間オブジェクトを操作する場合，パラメタの指定によっては，デッドロックが発生することがあります。
デッドロックの例を次の図に示します。

図 3-57　デッドロックの例



この例では，操作対象の文書空間オブジェクトとして二つのバージョンなしフォルダ「フォルダ A」と「フォルダ B」があります。ユーザ X が，バージョンなし文書「文書 1」の作成と同時に，その文書の登録先フォルダを「フォルダ A」,「フォルダ B」の順に指定して，バージョンなし文書を作成するメソッド（DbjDocSpace#createDocument メソッド）を実行したとします。このとき，ユーザ Y も，バージョンなし文書「文書 2」の作成と同時に，その文書の登録先フォルダを「フォルダ B」,「フォルダ A」の順に指定して同じメソッドを実行したとします。この場合，お互いに途中で待ち状態（WAIT モード）のままになってデッドロックが発生し，デッドロックが検知された時点でエラーになることがあります。

デッドロックを発生しにくくするには，次のような方法が考えられます。例えば，文書やフォルダなどの文書空間オブジェクトの一覧を取得する場合に，文書空間オブジェクトを一定の順序にソートしておいて，更新系メソッドに指定するバージョン識別子またはリンク識別子を，一定の順序で指定します。これによって，図 3-57 の例のような指定順が原因であるデッドロックが発生する可能性を軽減できます。

4 検索機能

この章では，Java クラスライブラリの検索機能である，edmSQL を使用して文書空間オブジェクトを検索する方法の概要について説明します。
DocumentBroker では，SQL に準じた構文規則を持つ edmSQL を使用してオブジェクトを検索できます。この検索では，プロパティの値を指定した検索や，検索タームを指定した検索が実行できます。
edmSQL の指定方法の詳細については，「5. edmSQL の文法」を参照してください。また，検索を実行するメソッドの使用方法については，「6. Java クラスライブラリの機能」を参照してください。

4.1 検索の概要

この節では，Java クラスライブラリで提供する検索機能の概要について説明します。

4.1.1 Java クラスライブラリでの検索

Java クラスライブラリで提供している検索機能には，次の二種類があります。

● **検索条件を指定するメソッドを実行する方法**
edmSQL という SQL に準じた構文で検索条件を指定するメソッドを実行する方法です。取得したいオブジェクトに関して知っている情報（プロパティの値など）を基に，文書空間オブジェクトを検索できます。検索結果として，文書空間オブジェクトの任意のプロパティが取得できます。OIID を表すプロパティを取得することもできます。

● **一覧を取得するメソッドを実行する方法**
一つの文書空間オブジェクトから，関連があるほかの文書空間オブジェクトの一覧を取得するメソッドを実行する方法です。例えば，フォルダからリンク付けている文書の一覧を取得したり，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からバージョンオブジェクトの一覧を取得したりして，文書空間オブジェクトを検索します。

これらの検索によって，検索結果として取得した文書空間オブジェクトを操作するためのインターフェースが取得できます。

なお，この章では，主に，検索条件を指定するメソッドを実行する方法について説明します。

4.1.2 検索条件と検索結果

検索を実行する場合に指定する検索条件と，取得する検索結果について説明します。

(1) 検索条件

検索条件を指定するメソッドの検索条件は，edmSQL 文として指定します。

edmSQL とは，文書空間オブジェクトを検索するための手段として DocumentBroker が提供する検索用言語です。SQL の文法に関する知識を活用して，文書空間オブジェクトの検索ができます。なお，これ以降，このマニュアルでは，edmSQL を使用した検索を，edmSQL **検索**といいます。また，edmSQL に従って記述した条件式を，edmSQL **文**といいます。

edmSQL には，次のような特長があります。

● **文書管理で使用する概念に基づいた検索条件を指定できます。**
edmSQL では，DocumentBroker の文書管理で使用する文書空間オブジェクトを構成する DMA クラスのクラス名やプロパティ名を使用して，検索条件を指定できます。

● **データベース標準問い合わせ言語である** SQL **に基づいた文法です。**
edmSQL の文法は，標準的な SQL の文法に基づいています。SQL の文法に関する知識を活用して edmSQL 文を作成できます。

● Java **クラスライブラリと違和感なく組み合わせて使用するためのインターフェースが提供されています。**
メソッドの引数に edmSQL 文を指定することによって，Java クラスライブラリのほかのインターフェースと違和感なく組み合わせて使用できます。

(2) 検索結果

ここでは，次の項目について説明します。

● 検索結果集合

● 検索結果キャッシュ

● 検索結果取得情報

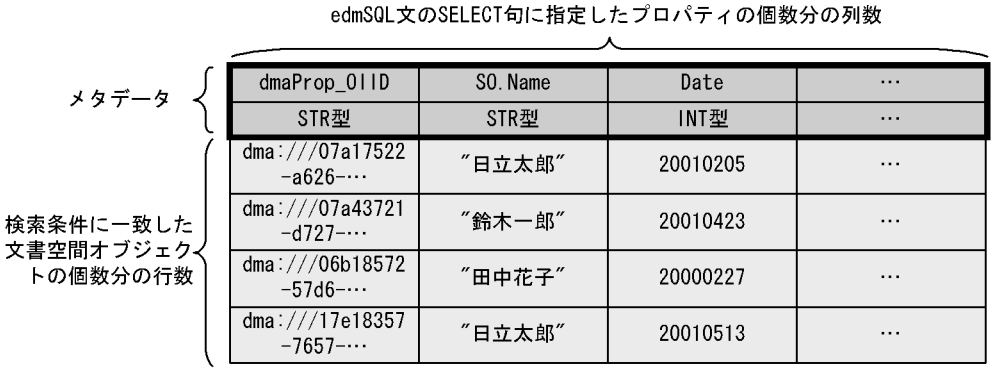
(a) 検索結果集合

検索条件を指定するメソッドを実行すると，検索結果集合が作成され，メソッドの戻り値として返却されます。**検索結果集合**は，検索条件に一致する文書空間オブジェクトの集合です。

検索結果集合は，要素がプロパティ値である，行と列の二次元データとして表すことができます。さらに，Java クラスライブラリでは，この二次元データにメタデータを付けて表すことができます。メタデータとは，列のプロパティ値のデータ型，および列名の集合です。

Java クラスライブラリで扱う検索結果集合を次の図に示します。

図 4-1 検索結果集合



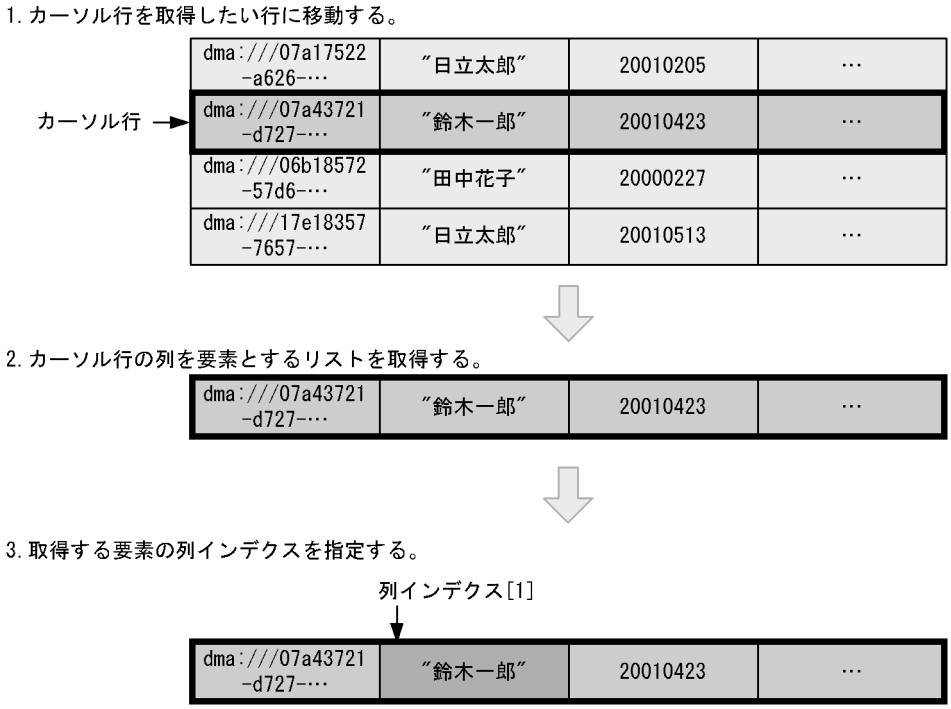
なお，メタデータとして列名（プロパティ名）が付けられるのは，名前付き検索結果の場合です。名前付き検索結果の詳細については，「4.3.4　名前付き検索結果を取得する検索」を参照してください。

検索結果集合は，検索実行メソッドの戻り値として返却される，DbjResultSet インターフェースを使用して操作します。DbjResultSet インターフェースでは，検索結果集合を，各行を要素とするリストとして扱います。さらに，行の各要素を，各列の値を要素とするリストとして扱います。

検索結果集合の行は，カーソル行を移動して特定します。各行の列の要素は，列の値を要素とするリストに対して，列インデクスを指定することによって特定します。列インデクスは，0 から始まる整数値であり，edmSQL 文の SELECT 句に指定した順に，0，1，…n と割り当てられます。

検索結果集合の要素を特定する例を次に示します。

図 4-2 検索結果集合の要素の特定



(b) 検索結果キャッシュ

検索結果は，いったんキャッシュに保存して，複数回に分けて取得することができます。このキャッシュを**検索結果キャッシュ**といいます。検索結果キャッシュは，検索条件を指定するメソッドおよび一覧を取得するメソッドで使用できます。検索結果キャッシュを使用するための情報は，検索結果取得情報に設定します。

検索結果キャッシュを使用すると，１回の検索でキャッシュから取得する検索結果の行数を指定したり，キャッシュ上の取得開始位置を指定して任意の行から検索結果を取得したりできます。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。このため，キャッシュ名が異なる複数の検索結果キャッシュを保持できます。キャッシュ名を指定しないと，ユーザアプリケーションプログラムが検索結果を取得し終わった時点で，検索結果キャッシュは無効になります。また，キャッシュ名は，同一セッション内だけで有効です。

なお，検索結果キャッシュから検索結果を取得できるのは，キャッシュ名が同じ検索結果キャッシュに対して，同じ検索条件で検索を実行する場合だけです。キャッシュを使用する検索では，次の条件を満たす場合，検索条件が同じであると判断されます。

- **同じセッション内で，同じキャッシュ名の検索結果キャッシュに対して検索を実行している**
  ログアウトすると，検索結果キャッシュは破棄されます。
- **検索結果キャッシュに保持されているキャッシュキーと検索結果取得情報に設定されているキャッシュキーが一致する**
  1 回目の検索では，**キャッシュキー**には，定数 DbjDef.INITIAL_KEY を指定します。2 回目以降は，前回の検索結果と一緒に取得したキャッシュキーを指定します。なお，DbjDef.INITIAL_KEY を指定すると，必ずキャッシュキーが不一致になるため，検索結果キャッシュは破棄されて新しい検索が実行されます。
- **検索結果キャッシュを作成した時と同じメソッドで検索を実行している**
  例えば，一覧を取得メソッドで取得した検索結果キャッシュに対して検索条件を指定するメソッドを実

行した場合などには，検索結果は破棄されます。

(c) 検索結果取得情報

検索結果の取得方法は，**検索結果取得情報**として指定します。検索結果取得情報には，検索結果集合からユーザアプリケーションプログラムに返却する検索結果の最大件数，取得開始位置と件数，並び順などが指定できます。また，検索結果キャッシュを使用する場合には，キャッシュに取得する最大件数，キャッシュ名およびキャッシュキーも指定できます。検索結果取得情報は，検索条件を指定するメソッドおよび一覧を取得するメソッドで指定できます。

取得する検索結果の並び順は，**コンパレータ**を使用して指定します。コンパレータを設定した検索結果取得情報を指定して検索を実行すると，コンパレータによってソートされた検索結果が取得できます。また，検索結果キャッシュに対して検索を実行する場合は，検索結果キャッシュの内容が並べ替えられます。並べ替えは，検索結果キャッシュに対して，異なるコンパレータを設定した検索結果取得情報を指定して検索を実行する時に実行されます。例えば，A というコンパレータでソートしていた検索結果キャッシュに対して，B というコンパレータを設定した検索結果取得情報を指定して検索を実行すると，検索結果キャッシュの内容が並べ替えられます。ただし，コンパレータとして null 値を指定した場合は，前と同じコンパレータを指定したものとして処理されます。このため，並べ替えは実行されません。

4.2 検索の種類

ここでは，edmSQL で実行できる検索の種類について説明します。edmSQL では，次に示す検索が実行できます。

edmSQL では，次の 3 種類の検索が実行できます。

- 属性検索
- 全文検索
- 全文検索機能付き文字列型プロパティを使用した全文検索

これらの検索は，組み合わせて指定することもできます。ここでは，それぞれの検索の概要について説明します。

4.2.1 属性検索

属性検索とは，文書空間オブジェクトの一つまたは複数個のプロパティの値を指定して実行する検索です。例えば，「文書名」がプロパティとして設定されている文書に対して，「文書名が『SQL 入門』である文書を探す」などの検索が実行できます。複数個のプロパティの値を指定する場合は，論理条件を指定して，「文書名が『SQL 入門』で，かつ，出版社が『ABC 出版』である文書を探す」という検索もできます。

検索の対象として，VARRAY 型のプロパティも指定できます。例えば，著者が複数人存在する可能性がある文書を管理するために，「著者」を VARRAY 型のプロパティで管理して，それぞれの「名前」をその要素のプロパティとして管理している場合，「『田中』という名前の著者が含まれる文書を探す」などの条件を指定した検索が実行できます。

4.2.2 全文検索

ここでは，全文検索について説明します。

(1) 全文検索とは

全文検索とは，登録したコンテンツを対象に，キーワードとなる文字列（**検索ターム**）を一つまたは複数個指定して実行する検索です。例えば，「文書のコンテンツ中に『コンピュータ』という文字列が含まれる文書を探す」などの検索が実行できます。

なお，全文検索には，次の 2 種類の検索が含まれます。

- **検索タームを指定する全文検索（狭義の全文検索）**
  HiRDB Text Search Plug-in の機能で実行できる，コンテンツに対して検索タームを指定して実行できる全文検索です。
- **概念検索**
  HiRDB Text Search Plug-in と HiRDB Text Search Plug-in Conceptual Extension の機能で実行できる，検索タームを文章（**種文章**）から抽出して実行できる全文検索です。この検索では，検索条件に指定した文章に近い概念を持つ（同じ単語が多く含まれる）文書が検索できます。

このマニュアルでは，これらの二つの検索を合わせて，「全文検索」といいます。それぞれの検索に特有の説明をする場合は，「検索タームを指定する全文検索」または「概念検索」と区別して説明します。

全文検索の対象にする文書は，全文検索機能付き文書クラスを基に作成されている必要があります。全文検索機能付き文書クラスの作成方法については，「4.5.2　全文検索機能付き文書クラスの作成」を参照してください。

なお，edmSQL では，全文検索の次の機能を実現するための関数を提供しています。

● プレーンテキストおよび構造を持った文書に対する全文検索（検索タームを指定する全文検索）

● プレーンテキストおよび構造を持った文書に対する概念検索

● 検索結果にスコアを付ける全文検索

● 検索結果にハイライトタグの埋め込みをして SGML 形式で出力する全文検索

ここでは，これらの検索の概要と，使用する edmSQL の関数について説明します。関数についての詳細は，「5.9　関数指定の構文規則」を参照してください。

また，全文検索では，HiRDB Text Search Plug-in の機能によって，検索タームや種文章をそのまま指定する検索だけでなく，同義語や異表記を同時に検索したり，検索タームに重みを指定したりすることもできます。これらの全文検索で指定できる機能の概要についても説明します。

なお，AIX の場合は，XML 文書が管理できません。このため，構造指定検索はできません。

(2) 全文検索の種類

ここでは，全文検索の種類について説明します。

(a) プレーンテキストに対する全文検索（検索タームを指定する全文検索）

プレーンテキストとは，構造を持たないテキスト形式の文書のことです。文書のコンテンツに対して，一つまたは複数個の検索タームを指定した検索が実行できます。複数個の検索タームを指定する場合，それらの論理条件も指定できます。

例えば，「文書のコンテンツに『コンピュータ』または『データベース』という文字列を含む文書」を検索したりできます。また，属性検索条件と組み合わせて，「文書のタイトル（プロパティ）が『データベース入門』であり，文書のコンテンツに『コンピュータ』という文字列を含む文書」のような検索も実行できます。

この検索には，contains 関数を使用します。

(b) 構造を持った文書に対する全文検索

プレーンテキストに対して実行できる全文検索に加えて，XML 文書のような構造を持った文書に対しては，その構造を条件に指定した全文検索ができます。このような検索を**構造指定検索**といいます。

なお，AIX の場合，構造指定検索は実行できません。

構造指定検索では，XML 文書のように論理構造を持っている文書に対して，その構造を指定した検索が実行できます。例えば，「文書・章・節」という構造で定義された XML 文書に対して，「構造『章』の中に『XML』という単語が含まれる文書」などが検索できます。

また，**構造に付けられた属性の値を指定した検索**もできます。例えば，「文書・タイトル・本文」という構造で定義され，構造「タイトル」が「分類」という属性を持っている XML 文書に対して，「構造『タイトル』の属性『分類』が『政治経済』である文書」などが検索できます。

さらに，複数の検索条件を指定する場合に，それらの検索条件が特定の構造内に含まれることを指定することもできます。例えば，「文書・著者」という構造に，「所属」という構造と「名前」という構造が含まれる XML 文書に対して，「特定の『文書・著者』という構造の中の，構造『所属』に『設計部』が含まれ，構造『名前』に『山田』が含まれる文書」などが検索できます。

この検索には，contains 関数を使用します。

(c) プレーンテキストに対する概念検索

全文検索実行時に，特定の検索タームを基に検索する以外に，「ある文章に似た内容の文書を検索したい」というような検索が実行できます。例えば，「リサイクルを含む環境問題」について書かれた文書を探したい場合に，「ゴミのリサイクル」に関する文章を基に検索が実行できます。

概念検索は，edmSQL などの検索条件に種文章を指定することで実行できます。種文章として指定できる容量は，5 メガバイト以内です。

ここでは，種文章に「…ゴミのリサイクルによって，資源が有効活用され，ゴミが減量化されます。…」という文字列を指定した場合の，概念検索の実行の流れを説明します。

1. HiRDB Text Search Plug-in Conceptual Extension の機能によって，種文章から特徴タームが抽出されます。
   特徴タームとは，種文章に指定した文章から，漢字，かたかなおよびアルファベットを抽出した単語です。
   この例の場合は，「ゴミ」，「リサイクル」，「資源」，「有効活用」，「減量化」というような単語が抽出されます。
2. HiRDB Text Serach Plug-in Conceptual Extension によって，1. の特徴タームに優先順位が与えられた**検索用特徴ターム**が選出されます。
   検索用特徴タームの選出基準については，HiRDB Text Search Plug-in の環境定義ファイルに指定します。詳細は，マニュアル「HiRDB Text Search Plug-in」を参照してください。概念検索では，この検索用特徴タームを基に，検索が実行されます。
3. 検索用特徴タームを検索タームとして，全文検索が実行されます。さらに，種文章との適合度順（検索用特徴タームを多く含む文章順）に，スコア値が設定されて，検索結果が出力されます。

なお，概念検索条件に指定する種文章は，? パラメタで指定する必要があります。

この検索には，concept_with_score 関数を使用します。

(d) 構造を持った文書に対する概念検索

構造を持った文書に対して概念検索を実行する場合も，検索条件に構造が指定できます。例えば，「文書・章・節」という構造が定義されている文書に対して，「構造『章』の中に『…ゴミのリサイクルによって，資源が有効に活用され，…』という文章に似た概念を含んでいる文書を検索する」というような検索が実行できます。

この検索には，concept_with_score 関数を使用します。

(e) 検索結果をスコア順にソートして取得できる全文検索

全文検索の検索結果には，スコアが付けられます。スコアの算出方法は，検索タームを指定する全文検索と，種文章を指定する概念検索で異なります。

検索タームを指定する全文検索の場合

スコアは，「検索結果として取得する文書の中に含まれる検索タームの数×検索タームに指定した重み×構造に指定した重み」の値として算出されます。ただし，検索条件に複数の検索タームの論理条件を指定したり，近傍検索条件を指定したりする場合，スコアの算出方法は異なります。
スコアの算出方法の詳細については，マニュアル「HiRDB Text Search Plug-in」を参照してください。
この検索には，contains_with_score 関数を使用します。スコア値は score 関数を使用して取得します。

種文章を指定する概念検索の場合

スコアは，種文章との適合度によって算出されます。適合度とは，種文章から抽出した検索用特徴

タームが，検索結果に含まれる割合のことです。
また，種文章を検索した場合のスコアを 100 として，スコアを正規化して算出することもできます。スコアの算出方法の詳細については，マニュアル「HiRDB Text Search Plug-in」を参照してください。
この検索には，concept_with_score 関数を使用します。スコア値は score_concept 関数を使用して取得します。

(f) 検索結果にハイライトタグの埋め込みをして SGML 形式で出力する検索

全文検索実行時に，検索結果として検索タームがハイライト表示される SGML 形式のファイルを出力します。例えば，「ネットワーク」という検索タームが含まれるテキスト形式の文書を検索する場合に，検索結果と同時に「<STRONG>」というハイライト表示用タグが埋め込まれた SGML 形式のファイルを出力できます。なお，検索結果として取得するハイライト表示用の文書は，検索対象である文書とは異なる文書として作成されます。

次に，検索対象文書とハイライト表示用の文書の例を示します。

検索対象文書

```
<!DOCTYPE body SYSTEM "DOC.dtd">
<body>
<p>
連絡事項
<note>
ネットワークの変更については，添付資料を参照してください。
</body>
```

検索結果として取得するハイライト表示用の文書

```
<!DOCTYPE body SYSTEM "DOC.dtd">
<body>
<p>
連絡事項
<note>
<STRONG>ネットワーク</STRONG>の変更については，添付資料を参照してください。
</body>
```

(3) 全文検索で指定できる機能

edmSQL では，全文検索条件として，HiRDB Text Search Plug-in の規則に従った検索条件が指定できます。

HiRDB Text Search Plug-in では，単純文字列としての検索タームおよび種文章のほか，検索タームを展開したり，重みを設定したりする指定ができます。

edmSQL では，これらの指定は全文検索を実行する関数の引数として指定します。これらの関数は，HiRDB Text Search Plug-in で提供されている関数と対応しています。指定方法については，マニュアル「HiRDB Text Search Plug-in」を参照してください。

ここでは，全文検索条件に指定できる機能の概要について説明します。

(a) 検索タームの異表記を検索対象にした検索（異表記展開指定）

異なる表記方法で表記された検索タームを含む文書も検索条件に指定できる検索です。

異表記展開を指定した場合は，検索タームに指定したかたかなの異表記を含む文書を検索したり，全角アルファベットの大文字・小文字の異表記を含む文書を検索したり，アルファベットの全角・半角の異表記を含む文書を検索したりできます。例えば，検索ターム「バイオリン」を検索する場合に，「ヴァイオリン」のように表記されている文書も検索できます。また，検索ターム「Ski」を検索する場合に，「ski」や「Ｓｋｉ」のように表記されている文書も検索できます。

異表記展開検索では，データベースで決められた規則に従って，検索タームの異表記を含む文書を検索します。

(b) 検索タームの同義語を検索対象にした検索（同義語展開指定）

検索タームと同じ意味を持つ単語（同義語）を含む文書も検索条件に指定できる検索です。

例えば，検索ターム「Ski」を検索する場合に，「スキー」を含む文書も一緒に検索できます。同義語展開検索では，あらかじめ作成してある同義語辞書を基に，検索タームの同義語を含む文書を検索します。

(c) 複数の検索ターム間の距離条件や出現順序を指定した検索（近傍条件指定）

複数の検索タームを指定した場合に，検索タームが出現する距離（文字数）を検索条件に指定できる検索です。

例えば，「『最新』と『技術』という検索タームを含み，これらの二つの単語が 10 文字以内に存在する文書」などが検索できます。この場合には，「最新印刷技術」や「最新の技術」などの文字列を含む文書が検索されます。

(d) 検索タームおよび構造に重みを付ける検索（重み指定）

検索タームに**重み**（重要度）を付ける検索です。重みを付けた検索とは，「『インターネット』および『データベース』が含まれる文書が検索したい。ただし，『インターネット』をより多く含む文書から取り出したいので，『インターネット』に『5』，『データベース』に『2』の重みを付けて検索する」のように指定する検索です。重みは構造にも付けられます。重みを付けた場合，スコアは重みによって算出されます。

概念検索では，重みは構造だけに指定できます。

(e) 種文章のスコアを 100 として，検索結果に相対的なスコアを付ける検索（概念検索のスコアオプション指定）

概念検索で取得するスコア値を，正規化して取得する指定ができる検索です。正規化した値は，種文章を検索した場合を 100 とした，相対的な値として算出されます。

(f) 構造の属性値を指定した検索（属性値指定）

構造を指定する場合に，その属性値も指定する検索です。

例えば，構造「文章」の属性「Author」に「Tanaka」を含む文書などが検索できます。

なお，概念検索では，属性値指定はできません。

(g) 複数の構造を対象にした検索条件を満たす構造が，ある特定の構造に含まれていることを指定する検索（特定構造検索指定）

複数の構造を対象にした検索を実行する場合に，その検索条件を満たす構造が，ある特定の構造に含まれていることを指定する検索です。構造「文章」の下位構造として「作成者」と「所属」がある場合に，特定の構造「文章」の構造「作成者」に「Tanaka」が含まれ，かつ構造「所属」に「営業部」が含まれる文書などが検索できます。

なお，概念検索では，特定構造検索指定はできません。

4.2.3 全文検索機能付き文字列型プロパティを使用した全文検索

ここでは，全文検索機能付き文字列型プロパティを使用した全文検索について説明します。なお，全文検索機能付き文字列型プロパティを使用するには，HiRDB Text Search Plug-in が必要です。

(1) 全文検索機能付き文字列型プロパティとは

全文検索が実行できる文字列型プロパティのことを，全文検索機能付き文字列型プロパティといいます。全文検索機能付き文字列型プロパティを使用すると，edmSQL で次に示す検索条件を指定して文書の検索ができます。

1. 指定した文字列がプロパティの値に含まれているか
2. 指定した文字列を異表記展開した文字列がプロパティの値に含まれているか
3. 指定した文字列を同義語展開した文字列がプロパティの値に含まれているか
4. 指定した文字列の距離（文字数）が指定の距離内にあるか

また，これらの検索条件を AND または OR で結ぶことができます。

(2) 全文検索機能付き文字列型プロパティの全文検索

edmSQL を使用して全文検索機能付き文字列型プロパティの値を取得する場合は，SELECT 句に extracts 関数を指定してください。

また，全文検索機能付き文字列型プロパティを全文検索する場合は，WHERE 句に contains 関数を指定してください。

全文検索機能付き文字列型プロパティを使用した全文検索の例を次に示します。

(例)

全文検索機能付き文字列型プロパティ usrProp_DocSummary に対応する文書中に，「コンピュータ」を同義語展開した文字列があるかどうかを検索し，文字列があればそのテキストデータを抽出します。

edmSQL の指定例

```
SELECT extracts(usrProp_DocSummary)
  FROM   usrClass_PropTextSearch
 WHERE  contains(usrProp_DocSummary,'{SYNONYM(myDic, "コンピュータ")}') is
true
```

注　usrClass_PropTextSearch は dmaClass_DocVersion クラスのサブクラスです。

! 注意事項

全文検索機能付き文字列型プロパティは，extracts 関数または contains 関数の第一引数に指定してください。それ以外の個所には指定できません。

(3) 全文検索機能付き文字列型プロパティに対応しているインターフェースおよびメソッド

全文検索機能付き文字列型プロパティに対応しているインターフェースおよびメソッドを次の表に示します。

表 4-1　全文検索機能付き文字列型プロパティに対応しているインターフェースおよびメソッド

対応しているインターフェース	対応しているメソッド
DbjDocSpace インターフェース	• createDocument • createFolder • createIndependentData • createVrDocument • createVrFolder • executeSearch

対応しているインターフェース	対応しているメソッド
DbjObj インターフェース	• cancelCheckOut • checkIn • checkOut • getDCRParent • readProperties • uploadContents • writeProperties • writeRenditionProperties

4.3 検索の実行方法の種類

ここでは，検索の実行方法の種類について説明します。

edmSQL 検索では，検索条件に？パラメタを指定して実行できます。また，検索実行時には，検索結果集合の取得方法や，検索結果に対する排他制御，アクセス制御などの設定ができます。

4.3.1 ？パラメタを使用した検索

業務によっては，検索条件のうち，一部の定数値だけを変更して繰り返し同じ検索を実行したい場合があります。このような場合に，？パラメタを使用できます。

例えば，文書の作成日がプロパティとして設定されている場合に，同じ検索条件で，作成日の値だけを「20010603」,「20010604」,「20010605」と変更して検索したい場合があります。このような，検索条件の特定の個所だけ変更して繰り返し検索を実行する場合に，このパラメタが使用できます。この検索では，検索条件式の作成日の値には「?」を設定しておきます。実行するたびに変更する値については別のパラメタとして指定します。これによって，複数の検索に対して一つの検索条件式を繰り返し使用でき，2 回目以降の検索にかかる処理時間を短縮できます。このパラメタを，**？パラメタ**といいます。また，概念検索で検索条件に指定する種文章は，必ず？パラメタを使用して指定します。

4.3.2 ロック指定検索

検索を実行する場合に，検索結果として取得する文書空間オブジェクトに対してロックを設定することができます。この検索を**ロック指定検索**といいます。これによって，検索結果集合に含まれる文書空間オブジェクトに対して，ほかのユーザが参照または更新することを防ぐことができます。ロック指定検索を実行すると，検索結果として取得する文書空間オブジェクト全体に，指定した種別のロックが設定されます。ただし，ロック指定検索は，ロックを指定しない場合に比べて処理に時間がかかります。必要に応じて使い分けてください。

なお，ロックを指定しない形式のメソッドを実行した場合，またはロック種別に DbjDef.LOCK_NONE を指定してメソッドを実行した場合は，検索結果集合に含まれる文書空間オブジェクトにロックは設定されません。

ロックについての詳細は，「3.11　排他制御モデル」を参照してください。

4.3.3 アクセス制御機能付き検索

アクセス制御機能を使用している環境の場合，検索結果として取得する文書空間オブジェクトのアクセス制御情報を有効にするか無効にするかを選択できます。アクセス制御情報は，検索対象の文書空間オブジェクトごとに設定されています。また，アクセス制御情報を有効にするか無効にするかの選択は，検索ごとに指定できます。

検索結果集合のアクセス制御情報を有効にする検索を，**アクセス制御機能付き検索**といいます。

アクセス制御機能付き検索を実行すると，アクセス制御機能を使用しない検索に比べて，処理に時間がかかります。このため，検索結果集合のアクセス制御情報を有効にするかどうかは，業務に応じて決定してください。

また，アクセス制御機能付き検索を実行する場合，次の制限があります。

- アクセス制御の対象になる文書空間オブジェクトは，主問い合わせの検索対象である文書空間オブジェクトだけです。したがって，副問い合わせの検索対象である文書空間オブジェクトには，アクセス制御

は適用されません。

- アクセス制御は，文書空間オブジェクトを対象に実行されます。この場合は，オブジェクトの識別性に基づいた検索になります。したがって，重複排除のようなオブジェクトのプロパティ値に基づく問い合わせや，集合関数のような統計演算をする検索を実行することはできません。

アクセス制御情報の詳細については，「3.10　アクセス制御モデル」を参照してください。

4.3.4　名前付き検索結果を取得する検索

検索結果集合を取得する時に，検索結果集合を表す表に，列名を付けることができます。列名が付いた検索結果集合を，**名前付き検索結果**といいます。また，列名が付いていない検索結果集合を，**名前なし検索結果**といいます。なお，名前なし検索を取得した場合も，あとから列名を設定して，名前付き検索結果に変更できます。

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場合に使用できます。例えば，検索結果集合の 1 行目をプロパティ値集合として取得して，ほかの文書空間オブジェクトのプロパティを更新するために使用したりできます。また，名前付き検索結果を取得すると，検索結果から列名を取得したり，列の値を取得する場合などに列インデクスではなく列名を指定したりすることができます。

名前付き検索結果の列名には，edmSQL 文の SELECT 句の項目名が設定されます。例えば，SELECT 句に，一つ目の項目として「dmaProp_OIID」を，二つ目の項目として「usrProp_Author」というプロパティ名を指定した場合は，取得する検索結果集合の列インデクス [0] の列に「dmaProp_OIID」，列インデクス [1] の列に「usrProp_Author」という列名が設定されます。

なお，名前付き検索結果を取得する検索を実行する場合，SELECT 句の選択項目は edmSQL 文に直接記述しないで，選択項目のリストを edmSQL 文とは別に作成する必要があります。edmSQL 文の SELECT 句には，「$_」を指定します。

名前付き検索結果を取得する検索の詳細な指定方法については，「6.7.4　文書空間オブジェクトの検索」を参照してください。また，edmSQL 文の指定方法については，「5.　edmSQL の文法」を参照してください。

4.3.5　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

同じ検索条件の検索を複数回繰り返す場合，**検索結果キャッシュ**を使用することができます。検索結果キャッシュを使用する検索を，**キャッシュ検索**といいます。また，検索結果キャッシュを使用しない検索は，**キャッシュなし検索**といいます。

キャッシュ検索では，1 回目の検索で取得した検索結果集合を，いったんキャッシュに保管します。ユーザアプリケーションプログラムには，ユーザアプリケーションプログラムで指定した件数分だけ返却します。2 回目以降の検索では，検索結果を，検索結果キャッシュからユーザアプリケーションプログラムに返却します。

例えば，検索結果が 1,000 件ある検索を実行する時に，「画面に 20 件ずつ表示したいので，検索結果は 20 件ずつ取得したい」という指定ができます。この場合，1 回目の検索で 1,000 件分の検索結果集合がキャッシュに保存され，ユーザアプリケーションプログラムには，そのうち先頭から 20 件分の検索結果だけが返却されます。2 回目以降の検索では，検索結果キャッシュからユーザアプリケーションプログラムに返却されます。「検索結果の 21 件目から 20 件を取得したい」という指定をして検索すれば，検索結果キャッシュの 21 件目から 40 件目の検索結果が，ユーザアプリケーションプログラムに返却されます。

また，検索結果が 10,000 件ある検索を実行した場合に，「検索結果としては 1,000 件までしか必要ない」

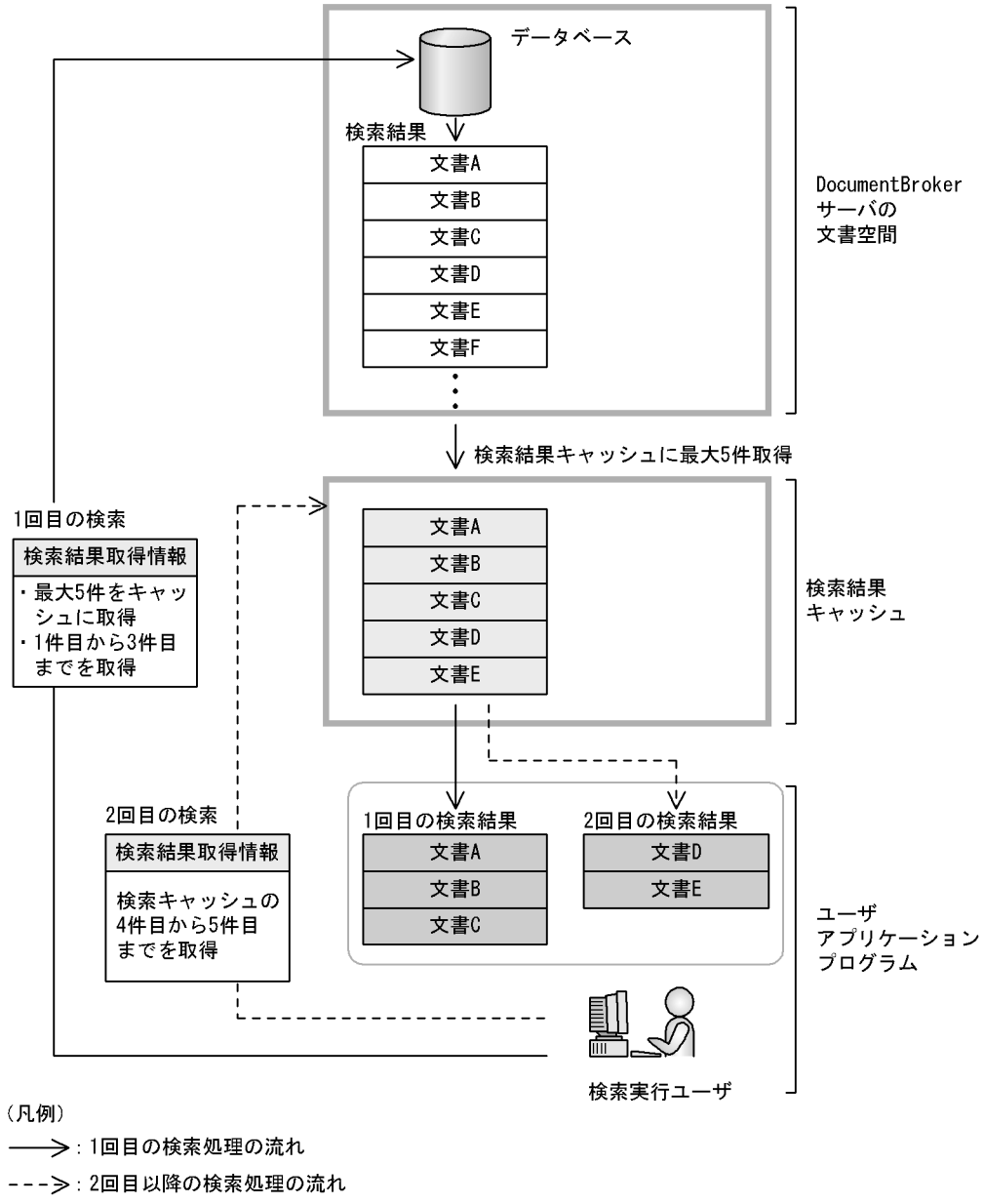
という場合は，検索結果キャッシュとして取得する件数を 1,000 件までに限定することもできます。

検索結果キャッシュは，**キャッシュ名**を付けて管理します。キャッシュ名を指定しない場合，検索結果はキャッシュに保持されません。また，検索結果キャッシュはキャッシュ名によって識別されるため，一度に複数の検索結果キャッシュを保持できます。

検索結果キャッシュを使用する場合，「検索結果の何件目から何件目までを取得する」という情報は，**検索結果取得情報**として指定します。検索結果取得情報に指定した件数分の検索結果が，検索結果キャッシュからユーザアプリケーションプログラムに返却されます。また，キャッシュに取得する件数も，検索結果取得情報に指定します。

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の例を，次の図に示します。

図 4-3　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の例



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の詳細な指定方法については，「6.7.4　文書空間オブジェクトの検索」を参照してください。

また，一覧を取得するメソッドを実行する場合にも，検索結果キャッシュを作成して，検索結果取得情報を指定できます。これによって，例えば，「フォルダに関連付けている文書の一覧を 10 件ずつ取得する」というような処理が実現できます。

4.4 検索対象になる文書空間オブジェクト

この節では，edmSQL 検索の対象になる文書空間オブジェクトと，検索できるプロパティの詳細について説明します。

なお，edmSQL 文の詳細については，「5. edmSQL の文法」を参照してください。

4.4.1 検索対象になる DMA クラス

検索対象は，edmSQL 文の FROM 句に指定する項目です。

edmSQL の検索条件には，検索する文書空間オブジェクトの構成要素である DMA クラスを検索対象として指定します。DMA クラスは，データベースの表と対応しています。DMA クラスを指定すると，文書空間に存在する，その DMA クラスを構成要素とするすべての文書空間オブジェクトが，検索の対象になります。

例えば，DMA クラスの usrClass_DocVersion クラス（dmaClass_DocVersion クラスのサブクラス）を検索対象に指定すると，その DMA クラスを基に作成された DocVersion オブジェクトを構成要素とする，バージョン付き文書およびバージョンなし文書（またはバージョン付き文書の 1 バージョン）がすべて検索対象になります。

検索対象には，複数の DMA クラスを結合して指定することもできます。edmSQL では，次の種類の結合ができます。

- 内部結合（INNER JOIN）
- 左外部結合（LEFT OUTER JOIN）

検索対象として指定できるのは，次の DMA クラスまたはそのサブクラスとしてユーザがデータベースに登録したクラスです。

● dmaClass_ConfigurationHistory クラス

● dmaClass_Container クラス

● dmaClass_DirectContainmentRelationship クラス

● dmaClass_DocVersion クラス

● dmaClass_ReferentialContainmentRelationship クラス

● edmClass_ComponentDocVersion クラス

● edmClass_ContainerVersion クラス

● edmClass_ContentSearch クラス

● edmClass_Relationship クラス

● edmClass_VersionTraceableContainer クラス

● edmClass_VersionTraceableContainmentRelationship クラス

● edmClass_VersionTracedComponentDocVersion クラス

● edmClass_VersionTracedDocVersion クラス

● edmClass_PublicACL クラス（アクセス制御機能を使用している環境の場合）

これらのクラスは，検索対象として，DocumentBroker サーバのメタ情報ファイル（edms.ini）に検索可能なクラスとして登録されています。

4.4.2 検索結果として取得できるプロパティ

検索を実行すると，検索結果として，検索条件に合った文書空間オブジェクトのプロパティが取得できます。

取得できるプロパティは，検索対象に指定した DMA クラスに定義されている，DMA が規定したプロパティ（dmaProp_ または edmProp_ で始まるプロパティ）およびユーザが定義したプロパティなどです。ただし，オブジェクトリファレンス※を表すデータ型のプロパティは取得できません。

検索結果として取得するプロパティは，edmSQL 文の SELECT 句に指定します。これによって，SELECT 句に指定したプロパティを列の要素とする検索結果集合が取得できます。データベース上のデータである検索結果は，データ型ごとに対応する Java クラスライブラリのデータ型に変換されます。これらの値は，副問い合わせの評価値として，述語や演算子にも使用できます。

検索結果として取得できるプロパティは，永続プロパティ（データベースに存在するプロパティ）または VARRAY 型のプロパティで，かつプロパティ定義に「dmaProp_IsSelectable = bool = 1」と指定されているプロパティです。

注※

オブジェクトリファレンスとは，DocumentBroker サーバで定義されている，データ型が Object 型で基本単位が VariableArray 型以外であるプロパティを表すデータです。Java クラスライブラリでは扱いません。

4.4.3 検索条件に指定できるプロパティ

FROM 句に指定した検索対象に対して，どのような条件を満たす文書空間オブジェクトを取得するかは，検索条件として指定します。また，複数の DMA クラスを検索対象として指定する場合に，どのような条件で結合するかの指定も，検索条件として指定します。

検索条件は，edmSQL 文の WHERE 句や FROM 句の ON 節に指定できます。検索条件には，検索条件の対象になるデータ型に従って，四則演算などを指定するための演算，比較演算子などを指定するための述語，全文検索などを指定するための関数を指定します。

検索条件に指定できるプロパティは，永続プロパティ（データベースに存在するプロパティ）または VARRAY 型のプロパティで，かつプロパティ定義に「dmaProp_IsSearchable = bool = 1」と指定されているプロパティです。

4.5 全文検索の対象になる文書の作成

この節では，全文検索の対象になる文書の作成方法について説明します。

DocumentBroker で全文検索の対象になるのは，次のように作成された文書です。

- 全文検索に必要なプロパティを持った DMA クラス（**全文検索機能付き文書クラス**）を基に作成された文書
- 文書の内容に対応するテキストデータを基に作成された全文検索用のインデクス（**全文検索インデクス**）を持つ文書

文書のレンディションタイプによって，全文検索の実行が制限されることはありません。ただし，全文検索用のインデクスの作成方法は，文書のレンディションタイプによって異なります。レンディションタイプごとの全文検索インデクスの作成方法については，「4.5.3　全文検索インデクスの作成」で説明します。

なお，全文検索インデクスは，全文検索に必要なプロパティの一つとして管理されます。

4.5.1 全文検索対象文書の作成手順

全文検索の対象となる文書は，次の手順で作成します。

1. **全文検索機能付き文書クラスを作成します。**
   全文検索機能付き文書クラスは，DMA クラスの dmaClass_DocVersion クラスのサブクラスとして作成します。作成方法の詳細については，「4.5.2　全文検索機能付き文書クラスの作成」を参照してください。
2. **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には，文書のコンテンツとして登録するファイルのほか，全文検索用のインデクスの作成に使用するファイルも指定します。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によっては，自動的にインデクスを作成することもできます。詳細は，「4.5.3　全文検索インデクスの作成」を参照してください。
3. **1. で作成した全文検索機能付き文書クラスを指定して，バージョンなし文書またはバージョン付き文書を作成します。メソッド実行時には，2. で作成した文書のアップロード情報も指定します。**
   バージョンなし文書を作成する場合は，文書のトップオブジェクトクラスを指定する引数に，全文検索機能付き文書クラスのクラス名を指定してメソッドを実行します。
   バージョン付き文書を作成する場合は，バージョンオブジェクトのトップオブジェクトクラスを指定する引数に，全文検索機能付き文書クラスのクラス名を指定してメソッドを実行します。
   バージョンなし文書またはバージョン付き文書の作成方法の詳細は，「6.7.3　文書空間オブジェクトの作成」を参照してください。

4.5.2 全文検索機能付き文書クラスの作成

ここでは，全文検索機能付き文書クラスの作成方法について説明します。

(1) 全文検索機能付き文書クラスとは

全文検索機能付き文書クラスとは，全文検索に必要な機能を持ったプロパティを追加した dmaClass_DocVersion クラスのサブクラスです。全文検索は，全文検索機能付き文書クラスを基に作成された文書に対して実行できます。

また，全文検索のうち，概念検索の対象になるクラスについては，全文検索用の機能を持ったプロパティのほか，概念検索用の機能を持ったプロパティも追加する必要があります。このマニュアルでは，全文検

索機能付き文書クラスのうち，概念検索に必要な機能を持ったプロパティを追加した dmaClass_DocVersion クラスのサブクラスを特に，**概念検索機能付き文書クラス**といいます。

Java クラスライブラリで全文検索の対象になる文書空間オブジェクトを作成する場合，全文検索機能付き文書クラスまたは概念検索機能付き文書クラスを，次に示す文書空間オブジェクトの構成要素として指定します。

- バージョンなし文書
- バージョン付き文書

(2) 全文検索機能付き文書クラスに追加するプロパティ

全文検索機能付き文書クラスを作成する場合，dmaClass_DocVersion クラスのサブクラスに，次のプロパティを追加する必要があります。

- 全文検索インデクス用プロパティ
- edmProp_Score プロパティ
- edmProp_RawScore プロパティ
- edmProp_DocLength プロパティ
- edmProp_ContentIndexStatus プロパティ

プロパティの追加方法については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

次に，全文検索インデクス用プロパティと，edmProp_ContentIndexStatus プロパティについて説明します。

全文検索インデクス用プロパティ

全文検索インデクス用プロパティには，表 4-2 に示す 4 種類があります。作成する文書で実現したい全文検索の種類によって，これらのプロパティを使い分けてください。なお，このマニュアルでは，以降，これらのプロパティとして生成される全文検索用のインデクスを，**全文検索インデクス**といいます。

全文検索インデクス用プロパティの種類と，それぞれの全文検索インデクス用プロパティで実行できる全文検索の種類を，次の表に示します。

表 4-2　全文検索インデクス用プロパティで実行できる全文検索の種類

プロパティ	全文検索※	構造指定検索	概念検索
edmProp_TextIndex （プレーンテキスト検索用全文検索インデクス）	○	×	×
edmProp_StIndex （構造指定検索用全文検索インデクス）	○	○	×
edmProp_ConceptTextIndex （プレーンテキスト検索用概念検索インデクス）	○	×	○
edmProp_ConceptStIndex （構造指定検索用概念検索インデクス）	○	○	○

（凡例）

○：実行できます。

×：実行できません。

注※

プレーンテキストに対して検索タームを指定して実行する全文検索を示します（構造指定検索および

概念検索を含みません)。

edmProp_ContentIndexStatus **プロパティ**

全文検索インデクスの作成状態を表すプロパティです。INT 型のデータが設定されます。このプロパティの値は，DocumentBroker Text Search Index Loader Version 3 で使用されます。

edmProp_ContentIndexStatus プロパティの値と意味を次の表に示します。

表 4-3　edmProp_ContentIndexStatus プロパティの値と意味

値	意　味
0	文書が登録されていないことを示す値です。
1	文書は登録されていますが，全文検索インデクスは作成されていないことを示す値です。
2	文書が登録されており，対応する全文検索インデクスも作成されていることを示す値です。
3	文書が更新されていますが，全文検索インデクスが更新されていないため，文書と全文検索インデクスが一致しないことを示す値です。
100 以上	エラー情報など，DocumentBroker Text Search Index Loader Version 3 が使用する値です。

(3) 概念検索機能付き文書クラスに追加するプロパティ

概念検索機能付き文書クラスを作成する場合には，次のプロパティを追加する必要があります。

- 全文検索インデクス用プロパティ（edmProp_ConceptTextIndex プロパティまたは edmProp_ConceptStIndex プロパティ）
- edmProp_Score プロパティ
- edmProp_RawScore プロパティ
- edmProp_DocLength プロパティ
- edmProp_ContentIndexStatus プロパティ
- edmProp_ScoreConcept プロパティ

プロパティの追加方法については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

4.5.3 全文検索インデクスの作成

ここでは，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**とは，全文検索機能付き文書クラスに追加した全文検索インデクス用プロパティに格納される，全文検索の対象になるテキストデータです。

全文検索インデクスは，どのレンディションタイプを持つ文書に対して作成するかによって，作成方法が異なります。

(1) 全文検索インデクスの作成時点

全文検索インデクスは，文書作成時や更新時など，文書のコンテンツを登録する時に同時に作成します。作成方法の詳細については，「6.7.3　文書空間オブジェクトの作成」および「6.8.9(2) コンテンツのアップロード」を参照してください。

なお，新規に作成した文書の全文検索インデクスを，全文検索対象文書全体の全文検索インデクスとは別に，一時的なインデクスとして作成する方法もあります。ここで作成したインデクスを**差分インデクス**といい，差分インデクスを作成する機能を**差分インデクス作成機能**といいます。

差分インデクス作成機能を使用すると，全文検索インデクスの追加登録が高速に処理できます。ただし，個々の検索処理は差分インデクスを使用しない場合に比べて若干遅くなります。また，差分インデクスは，容量が増えた段階で，まとめて全体のインデクスに反映させる必要があります。

差分インデクスの作成には，HiRDB Text Search Plug-in の差分インデクス作成機能を使用します。差分インデクス作成機能を使用する場合は，インデクス情報ファイルに差分インデクス作成を定義する必要があります。インデクス情報ファイルの定義については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。差分インデクス作成機能の詳細については，マニュアル「HiRDB Text Search Plug-in」を参照してください。

(2) レンディションタイプと全文検索インデクスの作成方法との対応

DocumentBroker では，文書のレンディションタイプに関係なく，全文検索インデクスを持つすべての文書に対して全文検索を実行できます。ただし，全文検索インデクスの作成方法は，レンディションタイプによって異なります。

レンディションタイプごとの全文検索インデクスの作成方法について説明します。なお，文書がマルチレンディション文書である場合には，マスタレンディションの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によって，全文検索インデクスの作成方法が異なります。

● 特定のレンディションタイプを持つテキストファイルの全文検索インデクスの作成

特定のレンディションタイプとは，次のレンディションタイプです。

- 先頭が「text/」で始まるレンディションタイプ
  例えば，「text/plain」や「text/xml」などです。

これらのレンディションタイプを持つ文書の場合，文書の登録時に，登録した文書のコンテンツから全文検索インデクスが作成できます。

● 特定のレンディションタイプを持たない文書の全文検索インデクスの作成

前に示したレンディションタイプ以外の形式を持つ文書です。ビットマップなどのイメージデータや，Word などのアプリケーションプログラムで作成した文書などが該当します。

これらの文書の場合，登録した文書のコンテンツから全文検索インデクスを作成することはできません。この場合は，ユーザアプリケーションプログラムなどで別に作成した全文検索インデクス作成用のテキストデータを文書のコンテンツと同時に登録して，全文検索インデクスを作成する必要があります。

4.5.4 全文検索インデクスの削除

全文検索インデクスだけを削除することはできません。文書空間オブジェクトを削除すると，全文検索インデクスも一緒に削除されます。

5 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。

5.1 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

5.1.1 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 >，<ID 文字列 >，< 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，「5.4　字句規則」を参照してください。

5.1.2 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，「5.5　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，「5.6　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，「5.7　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，「5.8　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 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，「5.9　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，「5.10　データ操作の構文規則」を参照してください。

5.2 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。

5.3　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。

5.3.1　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

● 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，Java クラスライブラリでは String 型のデータとして扱います。

なお，DocumentBroker サーバで扱うプロパティのデータ型である Id 型は，edmSQL のデータ型としては提供しませんが，クラスやプロパティを ID で指定することはできます。

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

● Java クラスライブラリのデータ型との値の受け渡しのインターフェース

注意事項

edmSQL では，文書空間オブジェクトのトップオブジェクトクラスである DMA クラスに対して検索を実行します。検索条件に指定するプロパティも，すべて DocumentBroker サーバで DMA クラス上に定義されているプロパティです。このため，この章では，プロパティのデータ型はすべて DocumentBroker サーバで定義されているプロパティのデータ型で説明します。DocumentBroker サーバで定義されているプロパティのデータ型については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。Java クラスライブラリで扱うプロパティのデータ型との対応については，「2.11.2　文書空間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。

5.3.2　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，Java 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 5-1 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 5-2 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 5-3 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 （dmaProp_DataType=DMA_DATATYPE_BOOLEAN と定義されたプロパティ） 格納値：0，1，2

値	定義
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 5-4 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 5-5 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 5-6 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) Java 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，Java 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) Java 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 Java 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 5-7 edmSQL の論理型の値と Java 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	Java 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 Java 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，Java 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。

5.3.3 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，Java 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 5-8 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 （dmaProp_DataType=DMA_DATATYPE_INTEGER32 と定義されたプロパティ） 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 5-9 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 5-10 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) Java クラスライブラリで扱うデータ型との対応

Java 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 5-11 edmSQL の整数型と Java 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	Java 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

5.3.4 オブジェクト型

オブジェクト型の値には，次の 2 種類があります。

- 基本単位が VariableArray 型のプロパティの値
- 基本単位が VariableArray 型以外のプロパティの値（オブジェクトリファレンスの値）

オブジェクトリファレンスとは，DMA 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 DMA 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 DMA 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，「5.9.3(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，「6.5 パラメタの操作」およびマニュアル「DocumentBroker Version 3 クラスライブラリ Java リファレンス」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，「5.9.3(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，Java 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 5-12 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 （dmaProp_DataType=DMA_DATATYPE_OBJECT dmaProp_Cardinality=255 と定義されたプロパティ） またはオブジェクトリファレンスを表すプロパティ

(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 5-13 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	=
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) Java クラスライブラリで扱うデータ型との対応

Java 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 5-14 edmSQL のオブジェクト型と Java 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	Java クラスライブラリで扱うデータ型	値
オブジェクト型 （基本単位が VariableArray 型のプロパティ）	DbjVArray	次のように定義されているプロパティの値としてだけ対応します。 • dmaProp_DataType = DMA_DATATYPE_OBJECT • dmaProp_Cardinality=255

オブジェクト型のデータ型の値として基本単位が VariableArray 型以外のプロパティの値（オブジェクトリファレンス）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。

5.3.5 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，「5.4.1 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，Java 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 5-15 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，「5.4.6 < 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 （dmaProp_DataType=DMA_DATATYPE_STRING と定義されたプロパティ） 格納値： 定義長（dmaProp_MaximumLengthString の値）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 5-16 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 5-17 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) Java クラスライブラリで扱うデータ型との対応

Java 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 5-18 edmSQL の文字列型と Java 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	Java 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

5.3.6 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，「5.9 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 5-19 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) Java クラスライブラリで扱うデータ型との対応

Java 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 5-20 edmSQL のバイナリ型と Java 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	Java 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 Java クラスライブラリでは String 型の値として取得できます。

5.4 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。

5.4.1 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 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 3　システム導入・運用ガイド」を参照してください。

5.4.2 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 5-21　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
16 進数字	0 ～ 9，a ～ f，および A ～ F

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，「5.4.1　文字コードセットとの対応」を参照してください。

5.4.3 < 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。

5.4.4 < 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，「5.4.5 < キーワード >」，「5.4.6 < リテラル >」および「5.4.7 < 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 5-22 < 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	5.4.7 < 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	5.4.5 < キーワード >
符号なし数値リテラル	符号なしの数字の値	5.4.6 < 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	5.9 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)	5.4.7 < 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)	5.4.6 < リテラル >
特殊文字	edmSQL で使用できる特殊文字	5.4.2 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 5-23 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 > に囲まれた区切られた識別子の中には，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，「< □ =」は，演算子「<=」ではなく，二つの演算子「<」と「=」と識別されるため，構文解析エラーになります。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。
% & : (?? ??) ^ | { } -> ::

5.4.5 < 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。

● DMA の検索モデルで定義されている演算子のうち，キーワードとして表現されているものは，edmSQL では使用しないものも含めてすべて < 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 5-24 < 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	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	
DMA の検索モデルで使用するキーワード	ID	
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	

種　類	値	特　徴
DocumentBroker の文書モデルで使用するキーワード	ACL CARDINALITY CLASS CONTAINER CONTENT DOCUMENT EVERYONE FOLDER FOR GROUP LINK LOCK OBJECT OF OWNER PARENT PRIVATE PROPERTY PUBLIC READ TO TYPE UNLINK UPDATE VARRAY VERSION VERSIONABLE WRITE	

(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。

5.4.6 < 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 5-25 < 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 5-26 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
'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 5-27　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，Java 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 Java 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
  ```

5.4.7　< 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 5-28 ＜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。

5.4.8 ＜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＞と<ID 文字列＞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 5-29 ＜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＞ • <ID 文字列＞	• 5.4.7 ＜識別子＞ • 5.4.9 <ID 文字列＞
プロパティ名	• ＜識別子＞ • <ID 文字列＞ • ＜特殊なプロパティ＞	• 5.4.7 ＜識別子＞ • 5.4.9 <ID 文字列＞ • 5.4.10 ＜特殊なプロパティ＞

種　類	値	参照先
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	• 5.9　関数指定の構文規則
相関名	＜識別子＞	• 5.4.7　＜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 DMA クラスを識別するための名称（dmaProp_DisplayName の値），または GUID 値の ID 文字列表現です。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。

一つの edmSQL 文の中で，同じクラスを識別するための＜クラス名＞の表現に，名称と ID 文字列を混用することはできません。一つの edmSQL 文では，＜クラス名＞は，名称または ID 文字列で統一して表記してください。例えば，<FROM 句＞の＜クラス名＞に ID 文字列を指定した場合は，<WHERE 句＞などでプロパティを修飾する＜クラス名＞にも ID 文字列を指定する必要があります。

＜識別子＞および<ID 文字列＞の詳細は，「5.4.7　＜識別子＞」および「5.4.9　<ID 文字列＞」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（dmaProp_DisplayName の値），または GUID 値の ID 文字列表現です。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。

一つの edmSQL 文の中で，同じプロパティを識別するための＜プロパティ名＞の表現に，名称と ID 文字列を混用することはできません。なお，そのプロパティがどの DMA 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。

一つの edmSQL 文では，＜プロパティ名＞は，名称または ID 文字列で統一して表記してください。例えば，＜選択項目＞の＜プロパティ名＞に ID 文字列を指定した場合は，ORDER BY 句で指定する＜プロパティ名＞にも，ID 文字列を指定する必要があります。

＜識別子＞，<ID 文字列＞および＜特殊なプロパティ＞の詳細は，「5.4.7　＜識別子＞」，「5.4.9　<ID 文字列＞」および「5.4.10　＜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，「5.9　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。

5.4.9　<ID 文字列 >

<ID 文字列 > の定義について説明します。<ID 文字列 > は，< クラス名 > や < プロパティ名 > を，名称（dmaProp_DisplayName に定義した値）以外で表記するための文字列です。

Java クラスライブラリでは，DMA クラスやプロパティは，名称で指定します。この名称は，メタ情報ファイルの定義内容に従って GUID の値に変換されて，DocumentBroker の文書空間に渡されます。DocumentBroker の文書空間では，DMA クラスやプロパティは GUID によって定義され，識別されています。

edmSQL では，GUID で DMA クラスおよびプロパティを指定できます。

edmSQL で GUID 値を表現する場合，<ID 文字列 > という記述形式で表現します。これは，< クラス名 > や < プロパティ名 > と同じ扱いになるため，<ID 文字列 > に <? パラメタ > によって値を指定することはできません。

(1) <ID 文字列 > として指定できる字句

<ID 文字列 > は，< シングルクォート > で囲まれた GUID 文字列として指定します。

GUID 文字列とは，「X」を 0 ～ 9 および a ～ f（小文字）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）」の形式の文字列です。

なお，<ID 文字列 > に指定する「ID」と <ID 本体 > の間に，区切り文字は指定できません。

(2) 規約詳細

<ID 文字列 > の詳細について説明します。

(a) <ID 文字列 > が区切られる単位についての詳細

ID 文字列の「ID」は，< 予約されたキーワード > です。したがって，< 非区切りトークン > として扱われます。また，<ID 本体 > は，< 区切りトークン > として扱われます。このため，<ID 文字列 > の前に < 区切り文字 > なしで < 非区切りトークン > は指定できませんが，<ID 文字列 > の後ろに < 区切り文字 > な

しで < 非区切りトークン > は指定できます。

<ID 文字列 > の表記例と edmSQL 検索実行時の値の扱われ方を，次の表に示します。

表 5-30　<ID 文字列 > の表記例と edmSQL 検索実行時の値の扱われ方

<ID 文字列 > の表記	扱われ方
ID'FA7DA34B-CFF3-11d3-A03E-00A0C9967923'	正しい <ID 文字列 > として扱われます。
ID □ 'FA7DA34B-CFF3-11d3-A03E-00A0C9967923'	「ID」と <ID 本体 > の間に区切り文字が入っているため，字句解析エラーになります。
ID'FA7DA34B-CFF3-11d3-A03E-00A0C9967923'FROM	<ID 文字列 > と「FROM」として扱われます。
SELECTID'FA7DA34B-CFF3-11d3-A03E-00A0C9967923'	「SELECTID」と文字リテラルとして扱われます。

（凡例）
□：区切り文字または < 空白 >

5.4.10　< 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 5-31　< 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_StIndex	edmProp_StIndex 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 >
edmProp_ConceptStIndex	edmProp_ConceptStIndex 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，「4.5.2　全文検索機能付き文書クラスの作成」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，DMA の検索モデルを拡張するための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，edmProp_StIndex 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，edmProp_ConceptStIndex 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，「5.9　関数指定の構文規則」を参照してください。

5.4.11　<? 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。

5.4.12　<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，Java 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，Java 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，Java 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 文中に 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，「5.9.3(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 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，「6.5　パラメタの操作」およびマニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」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 = ?
```

5.5 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，「5.6　問い合わせ式の構文規則」を参照してください。

5.5.1 <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。

5.5.2 <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。

5.6 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 DMA 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，「5.8　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，「5.10　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,「5.8　述語の構文規則」を参照してください。

5.6.1　< 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。

5.6.2　< 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 DMA 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，「5.9　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 > に基本単位が 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。

5.6.3　< 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。

5.6.4　<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 DMA クラスを指定します。複数の DMA 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 DMA 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 DMA 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（DMA 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 DMA 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 DMA 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 DMA 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 DMA 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 DMA 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 DMA 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 DMA 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。

5.6.5 <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，「5.8　述語の構文規則」を参照してください。

5.6.6 < 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，「5.8　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 > に基本単位が 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。

5.6.7 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，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。

5.6.8 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。

5.7 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>
```

5.7.1 < 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 DMA 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 5-32 < 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。
ID'FA7DA34B-CFF3-11d3-A03E-00A0C9967923'	<ID 文字列 > 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。
myClass_XX.ID'FA7DA34B-CFF3-11d3-A03E-00A0C9967923'	<ID 文字列 > によって表現した < 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。

5.7.2 < 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，基本単位が 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。

5.7.3 < 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，基本単位が 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，基本単位が 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，基本単位が VariableArray 型のプロパティの < プロパティ名 > を指定します。例えば，「基本単位が 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
```

5.7.4 < 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，「5.9　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,「5.4.4 < 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

- < 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 5-33 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。

5.7.5　＜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）

5.7.6　＜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。

5.7.7 < 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 5-34 ＜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。

5.8 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>
```

5.8.1 < 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 5-35 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 5-36 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 5-37 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

5.8.2 < 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>
```

5.8.3 < 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。
- 基本単位が 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，基本単位が 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 5-38 < 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

5.8.4 < 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 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
```

5.8.5 <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。

5.8.6 <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。

5.8.7 <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。

5.8.8 <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。

5.8.9 <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。

5.9 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。

5.9.1 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 **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。

5.9.2 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> <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」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 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 および 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 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」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_StIndex プロパティ
- edmProp_TextIndex プロパティ
- edmProp_ConceptTextIndex プロパティ
- edmProp_ConceptStIndex 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」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_StIndex プロパティ
- edmProp_ConceptTextIndex プロパティ
- edmProp_ConceptStIndex 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」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_StIndex プロパティ
- edmProp_ConceptTextIndex プロパティ
- edmProp_ConceptStIndex 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><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_StIndex プロパティ
- edmProp_ConceptStIndex 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)
```

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 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 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，Java クラスライブラリ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_ConceptTextIndex
- edmProp_ConceptStIndex

< 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」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_ConceptStIndex,
                   ? 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
- edmProp_ConceptStIndex 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。

5.9.3 変換関数（edmSQL 関数）

< 変換関数 > は，オブジェクト型の値を，Java 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。Java クラスライブラリでは，基本単位が VariableArray 型以外の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 サーバで 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 >（基本単位が VariableArray 型以外の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。

5.10 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
```

5.10.1 <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。
- < ソートキー > の < 選択項目 > として指定できるプロパティは，ソート可能（Orderable）なプロパティです。これは，プロパティ定義に次のように指定されたプロパティのことです。

  ```
  dmaProp_IsOrderable = bool = 1
  ```

- 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。

5.11 edmSQL の指定例

この節では，edmSQL の指定例について説明します。

5.11.1 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 5-39 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 5-40 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 5-41 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 5-42 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※ 基本単位が 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) 基本単位が VariableArray 型のプロパティを指定した属性検索

ここでは，基本単位が VariableArray 型のプロパティの検索方法について説明します。

(a) 基本単位が VariableArray 型のプロパティを取得する検索

ここでは，myPaper クラスの基本単位が 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) 基本単位が 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 クラスの基本単位が 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'
```

5.11.2 全文検索（HiRDB Text Search Plug-in を利用した検索）

ここでは，HiRDB Text Search Plug-in 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」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 5-43 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
edmProp_ConceptStIndex	全文検索インデクス（概念検索および構造検索ができる全文検索インデクス）

全文検索の対象になる文書の作成方法については，「4.5 全文検索の対象になる文書の作成」を参照して

ください。

また，Document X クラスは，構造を持った XML 文書の基になる DMA クラスです。構造『文章』の下位に，構造『名前』，構造『キーワード』および構造『本文』が定義されています。

Document X クラスを基に作成した XML 文書の例を次に示します。

Document X クラスを基に作成した XML 文書の例

```
<?xml version="1.0" encoding="Shift_JIS"?>
・・・
<文章>
  <名前>
   日立太郎
  </名前>
  <キーワード>
   edmSQL・・・
  </キーワード>
  <本文>
   edmSQLとは，SQLに準拠した・・・。
  </本文>
</文章>
```

(2) contains 関数を使用した全文検索

ここでは，contains 関数を使用した全文検索の指定方法について説明します。

Document X クラスの文書のうち，最上位の構造『文章』の下にある構造『本文』に「コンピュータ」を含む文書の OIID を取得します。ただし，「コンピュータ」は，同義語辞書「myDic」を使用して，同義語展開して検索します。

指定例

```
SELECT dmaProp_OIID
FROM   "Document X"
WHERE  contains(edmProp_ConceptStIndex,
       '文章[本文{SYNONYM(myDic,"コンピュータ")}]') IS TRUE
```

(3) extracts 関数を使用した全文検索

ここでは，extracts 関数を使用した全文検索の指定方法について説明します。

Document X クラスの文書のうち，最上位の構造『文章』の下にある構造『本文』に「Computer」を含む文書のコンテンツを検索して，「Computer」に <STRONG> タグを付けたテキストを出力します。

指定例

```
SELECT extracts(edmProp_ConceptStIndex
 '文章.本文','文章[本文{"COMPUTER"}]', 'STRONG')
FROM  "Document X"
WHERE contains(edmProp_ConceptStIndex,
       '文章[本文{"COMPUTER"}]') IS TRUE
```

(4) score 関数と contains_with_score 関数を使用した全文検索

ここでは，score 関数と contains_with_score 関数を使用した全文検索の指定方法について説明します。

Document X クラスの文書のうち，最上位の構造『文章』の下にある構造『本文』に「コンピュータ」を含む文書の OIID を取得します。ただし，「コンピュータ」は，同義語辞書「myDic」を使用して，同義語展開して検索します。

また，検索結果として，Title プロパティ，Author プロパティとともに，検索条件に対する文書のスコア

（score）を取得します。score は，contains_with_score 関数によって算出されます。したがって，スコア値を取得する検索を実行する場合は，このように score 関数と contains_with_score 関数を組み合わせて指定します。

さらに，この例では，取得したスコア値を基に，検索結果を昇順にソートします。

指定例

```
SELECT dmaProp_OIID,Title,Author,score(edmProp_ConceptStIndex)
FROM   "Document X"
WHERE contains_with_score(edmProp_ConceptStIndex,
        '文章[本文{SYNONYM(myDic,"コンピュータ")}]') IS TRUE
ORDER BY 3 ASC
```

（5） score_concept 関数と concept_with_score 関数を使用した概念検索

ここでは，score_concept 関数と concept_with_score 関数を使用した概念検索の指定方法について説明します。

Document X クラスの文書のうち，種文章と近い概念を含んでいる文書を検索します。種文章は，? パラメタとして指定します。種文章は 5 メガバイトまで指定できます。

この例では，種文章から抽出された検索用特徴タームを，同義語辞書 myDic を使用して同義語展開して検索します。

指定例

```
SELECT dmaProp_OIID,
      score_concept(edmProp_ConceptStIndex)
FROM  "Document X"
WHERE concept_with_score
       (edmProp_ConceptStIndex, ? AS BINARY (5m));
```

5.12 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。

5.12.1 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。

また，「dmaProp_」および「edmProp_」で始まるプロパティについては，検索できるプロパティと検索できないプロパティがあります。検索できるプロパティについては，「4.4 検索対象になる文書空間オブジェクト」を参照してください。

5.12.2 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

- 副問い合わせの SELECT 句に指定できるプロパティは一つです。複数指定した場合は，edmSQL 構文解析エラーになります。
  また，ここで指定できるプロパティのデータ型も，副問い合わせで指定している < 述語 > の規則に従います。不正なデータ型のプロパティを指定した場合は，edmSQL 構文解析エラーになります。
- SELECT 句に文字列型のプロパティを指定して重複排除をする場合，SELECT 句には 255 バイト以内で定義している文字列型のプロパティを指定してください。定義が 255 バイトを超える文字列型のプロパティを指定した場合，重複排除は実行できません。実行した場合，データベースエラーになります。
- 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
- 基本単位が VariableArray 型のプロパティを指定して，重複排除を指定すると，edmSQL 構文解析エラーになります。

● 基本単位が 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，データベースエラーになります。

(3) WHERE 句に関する制限事項

● WHERE 句に文字列定数を使用した条件を指定する場合，255 バイト以内で指定してください。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。

● WHERE 句に文字列型のプロパティを指定する場合は，255 バイト以内で定義されたプロパティを指定してください。制限を超えたプロパティを指定した場合は，データベースエラーになります。

● 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) ORDER BY 句に関する制限事項

● ORDER BY 句で指定できる SELECT 句でのプロパティのインデクス，またはプロパティは，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。

● ORDER BY 句に文字列型のプロパティを指定する場合は，255 バイト以内で定義されたプロパティを指定してください。制限を超えたプロパティを指定した場合は，データベースエラーになります。

(5) 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，「4.4　検索対象になる文書空間オブジェクト」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，「5.4　字句規則」を参照してください。

(6) 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。

(7) ? 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。

(8) 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。

5.12.3　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 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」の，表の結合に関する規則の説明を参照してください。

5.12.4 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。

6 Java クラスライブラリの機能

この章では，Java クラスライブラリの機能について説明します。また，Java 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。なお，この章で説明するクラス，インターフェースまたはメソッドの詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」を参照してください。

6.1 パッケージとクラス

この節では，DocumentBroker Development Kit および DocumentBroker Runtime が Java 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。

6.1.1 パッケージ名

DocumentBroker Development Kit および DocumentBroker Runtime が提供する Java 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
```

6.1.2 クラスの分類

Java 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

6.1.3 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。

6.1.4 パラメタクラス

パラメタクラスは，Java 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。

- 文書空間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
- XML プロパティマッピング結果情報
- 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

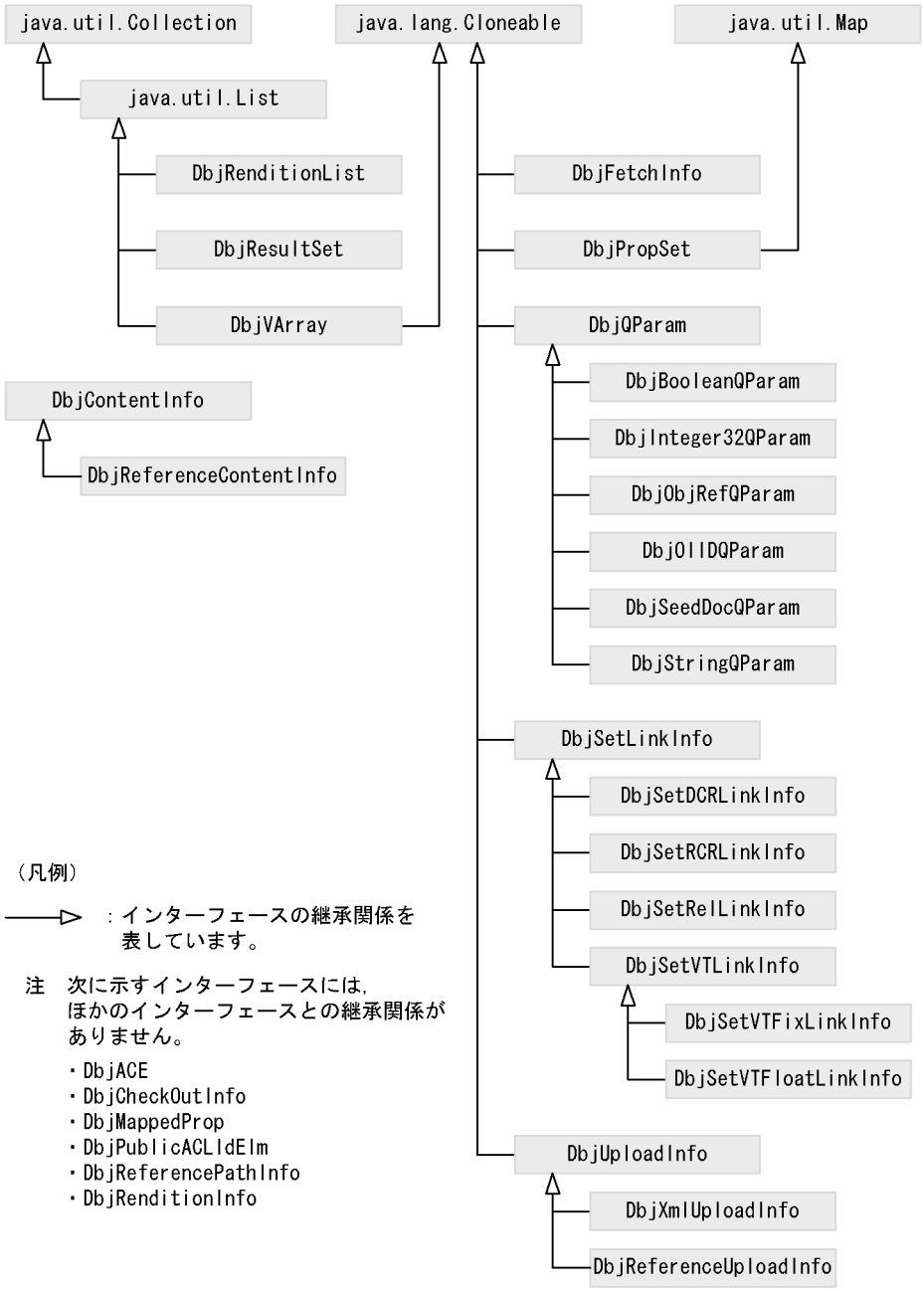
パラメタクラスのインターフェースをデータ型とするこれらの情報は，文書管理クラスのメソッドの引数に指定したり，メソッドの戻り値として返却されたりします。また，これらの情報を参照したり設定したりするメソッドは，パラメタクラスのインターフェースごとに定義されています。

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 6-1　パラメタクラスのインターフェースの継承関係



6.1.5　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

● 文書空間オブジェクトを作成する機能

● 文書空間オブジェクトを検索する機能

● 文書空間オブジェクトを操作する機能

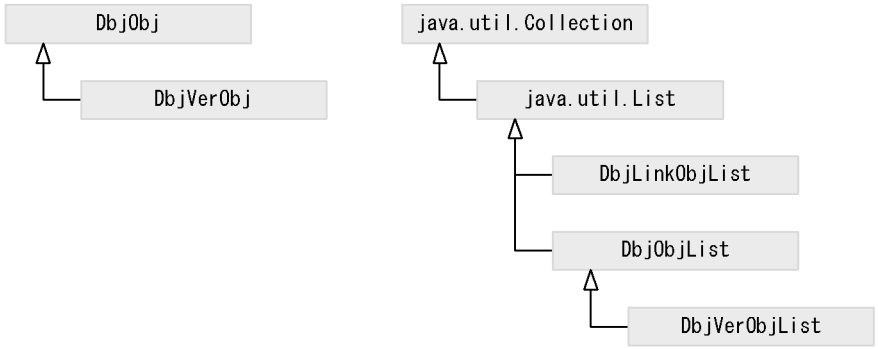
● 複数の文書空間オブジェクトをまとめて操作する機能

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，「3. 文書管理モデル」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 6-2 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
- DbjXmlTranslator

6.1.6 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 サーバで管理されている，DMA クラスや文書空間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

● DMA 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

● 文書空間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。

6.1.7 定数定義クラス

定数定義クラスとは，Java 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。

Java 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，次のような定数が定義されています。

● 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

● 文書空間オブジェクトのプロパティのデータ型を表す定数

● 真偽値を表す定数

● 文書空間オブジェクトのオブジェクト種別を表す定数

● 全文検索インデクスの作成方法を表す定数

● コンテンツのリファレンス種別を表す定数

● コンテンツのパス操作モードを表す定数

● リンク種別を表す定数

● ロック種別を表す定数

● データのソート種別を表す定数

● パーミッションを表す定数

● ユーザの特権を表す定数

● リレーション種別を表す定数

● レンディションの変換フラグおよび状態フラグを表す定数

● サブジェクト種別を表す定数

● システムサブジェクトを表す定数

● XML プロパティマッピング機能の構文解析レベルを表す定数

● そのほかの定数

DbjTraceDef クラスでは，次のような定数が定義されています。

● 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

● 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，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」を参照してください。

6.1.8 例外クラス

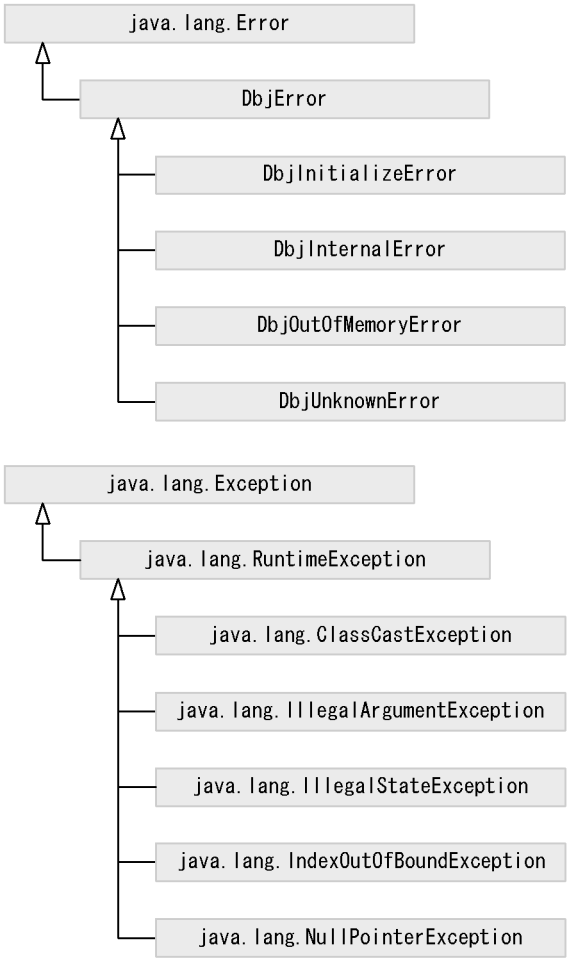
例外クラスは，Java クラスライブラリで発生する例外のうち，Java クラスライブラリ固有の例外を扱うクラスの総称です。

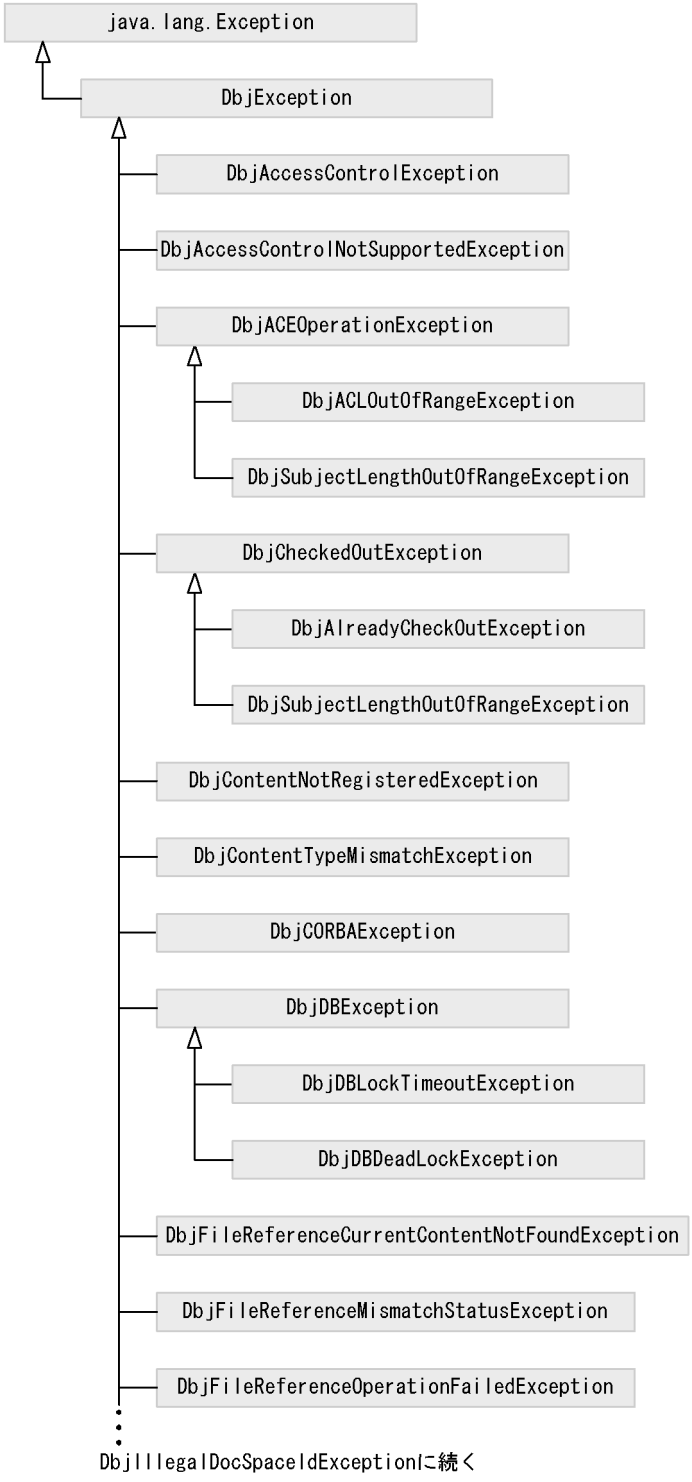
ユーザアプリケーションプログラムの処理中に，Java クラスライブラリ固有のエラーが発生すると，例外クラスのオブジェクトがスローされます。

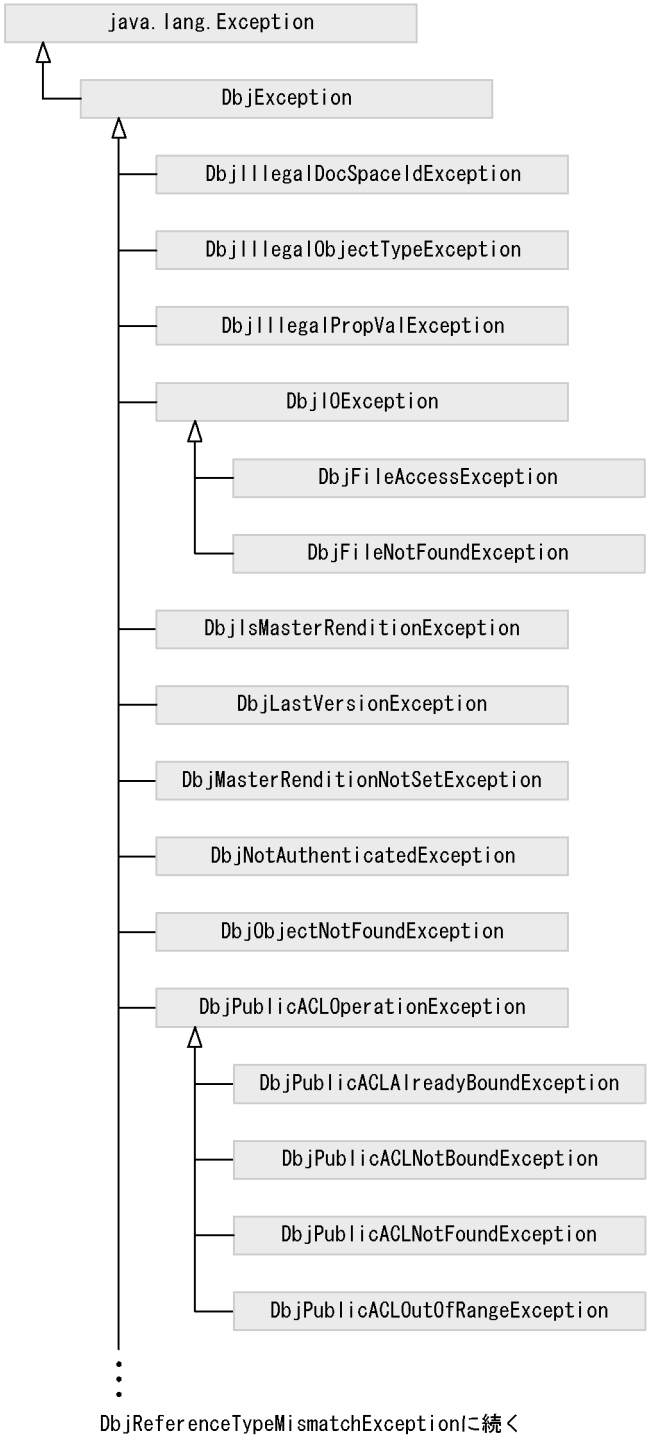
例外クラスの各クラスは，java.lang.Exception クラスまたは java.lang.Error クラスを継承しています。

例外クラスのクラスの継承関係を，次の図に示します。

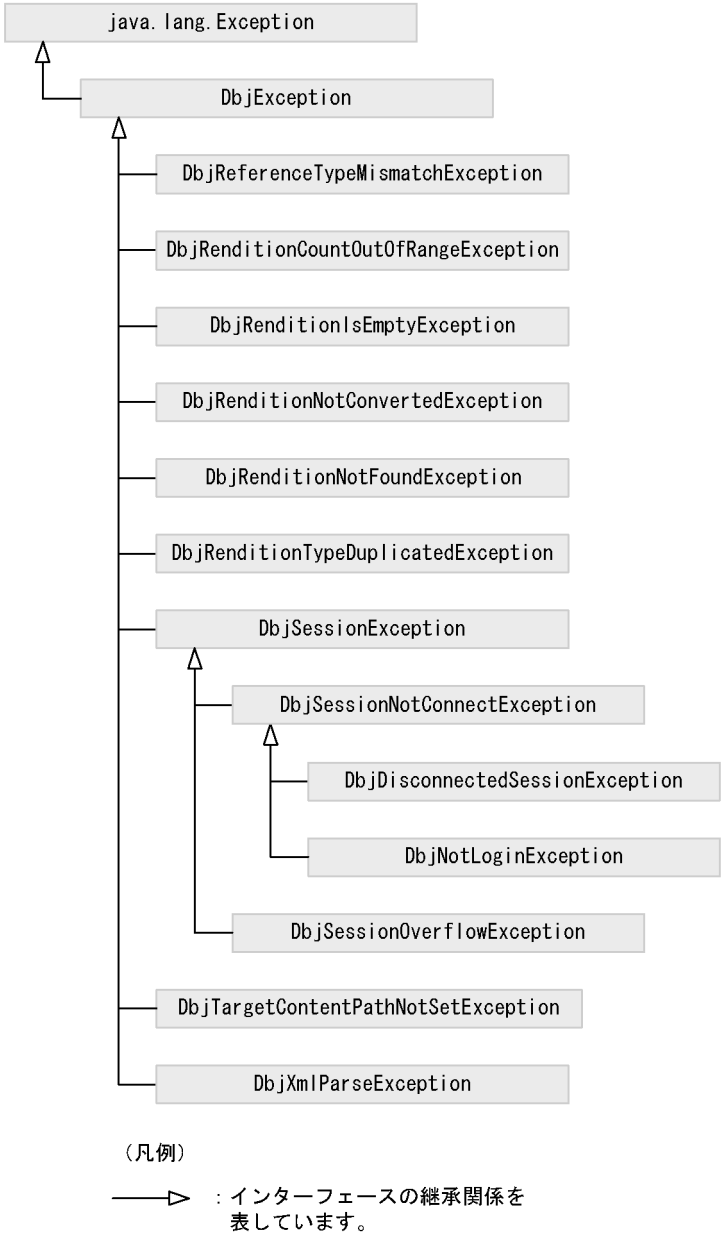
図 6-3 例外クラスのクラスの継承関係




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



6.1.9 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 Development Kit および DocumentBroker Runtime 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。

6.1.10 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，Java 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。

6.2 Java クラスライブラリで扱うデータ

この節では，Java クラスライブラリで扱うデータについて説明します。

6.2.1 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，Java 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 6-1 Java 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。

6.2.2 プロパティのデータ型と Java クラスライブラリ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 Java 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 6-2 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。

Java クラスライブラリで扱うプロパティのデータ型と，DocumentBroker サーバで扱うデータ型の対応については，「2.11.2 文書空間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 6-2 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。

6.2.3 定数

Java 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，マニュアル「DocumentBroker Version 3 クラスライブラリ Java リファレンス」を参照してください。

6.3 プログラムの流れ

この節では，Java 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。

6.3.1 インターフェースの取得

Java 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。

Java 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。

6.3.2 文書空間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

Java クラスライブラリでは，文書空間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。

文書空間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。

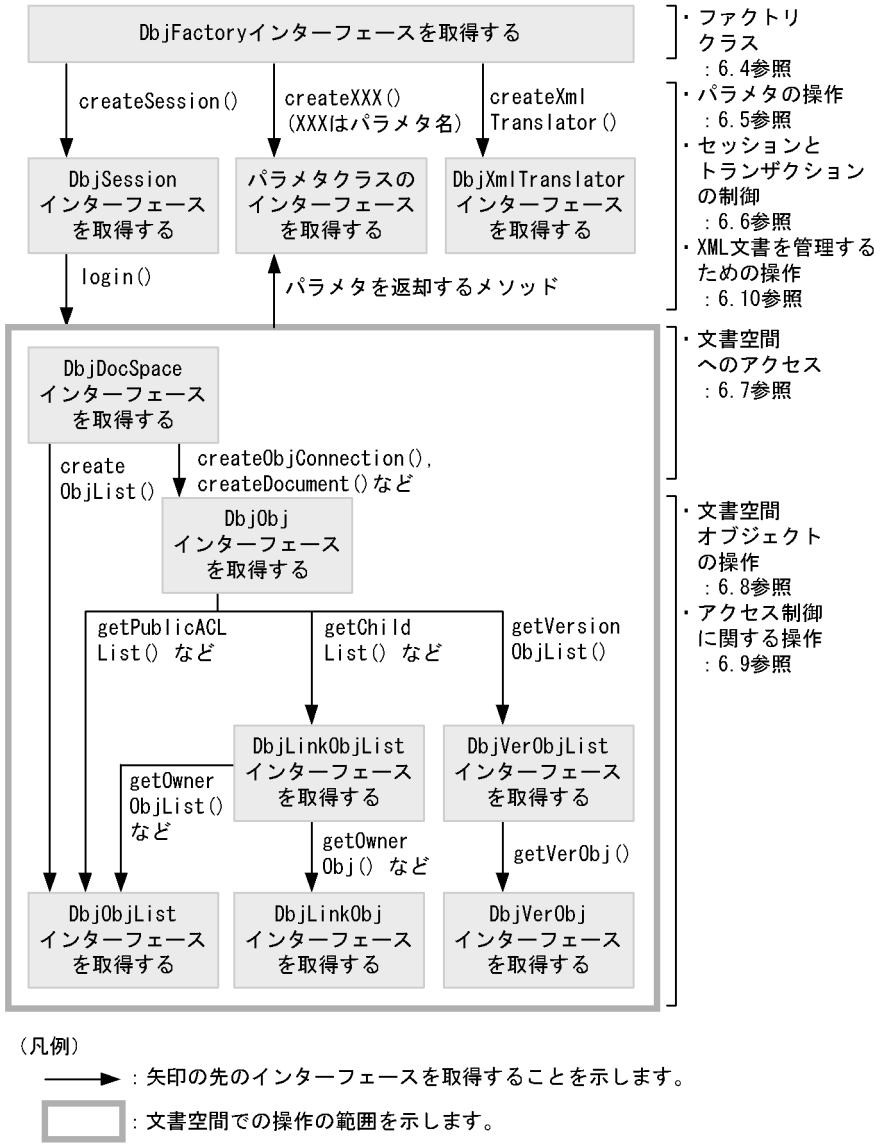
DbjFactory インターフェースを取得したあとの，文書管理機能を使用するユーザアプリケーションプログラムの処理手順は，次のようになります。

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，個々の文書空間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，文書空間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 6-4　文書空間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



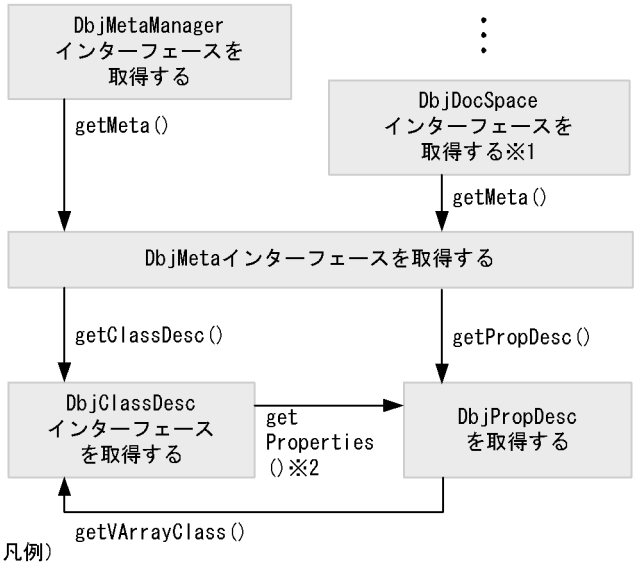
6.3.3　メタ情報を取得する場合のインターフェースの取得の流れ

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 6-5　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6-6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。

6.4 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

6.4.1 DbjFactory0200 クラスの機能

Java 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。

6.4.2 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
- 文書管理クラスの DbjXmlTranslator 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，Java 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，文書空間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，「6.5 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。文書空間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,「6.6　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,「7.6.3　動作環境定義ファイル」を参照してください。

(3) 文書管理クラスの DbjXmlTranslator インターフェース

DbjXmlTranslator インターフェースは，XML プロパティマッピング機能を提供するインターフェースです。DbjFactory#createXmlTranslator メソッドを実行すると，DbjXmlTranslator インターフェースが取得できます。

なお，AIX の場合，DbjXmlTranslator インターフェースは使用できません。

XML プロパティマッピング機能を使用する方法については,「6.10　XML 文書を管理するための操作」を参照してください。

6.5 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。

6.5.1 パラメタクラスのインターフェースの機能

Java クラスライブラリで扱うデータのうち，Java 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。

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，「6.1.4(2) インターフェースの継承関係」を参照してください。

6.5.2 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 6-3 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
DbjFetchInfo	検索結果取得情報
DbjMappedProp	XML プロパティマッピング結果情報（XML プロパティマッピング機能を実行して作成される情報）

インターフェース	扱う情報
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
DbjSetVTFixLinkInfo	構成管理型（構成管理モードは FIX モード）リンク設定情報
DbjSetVTFloatLinkInfo	構成管理型（構成管理モードは FLOATING モード）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）
DbjXmlUploadInfo	XML 文書のアップロード情報

（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，文書空間オブジェクトの VARRAY 型のプロパティの要素です。このため，ほかの文書空間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，「6.9.2　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，「6.8.9　文書のコンテンツの操作」を参照してください。

（4） 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，「6.7.4　文書空間オブジェクトの検索」を参照してください。

（5） DbjMappedProp インターフェース

XML プロパティマッピング結果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。

● プロパティマッピングされた DMA クラス名のリスト（classList）

● マッピングするプロパティ値集合（propSet）

なお，バージョン付きオブジェクトの場合，classList の要素が二つになり，一つ目の要素がバージョニングオブジェクトを作成するための DMA クラス，二つ目の要素がバージョンオブジェクトを作成するための DMA クラスになります。

このインターフェースの使用方法については，「6.10.2　XML 文書管理機能を使用する操作」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，文書空間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，要素のキーを文書空間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，「6.8.7　文書空間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（文書空間オブジェクトの dbrProp_ACLIdElem プロパティ）の要素です。このため，ほかの文書空間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，「6.9.3　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 Java 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，「6.7.4　文書空間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，「6.8.12　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）
- 登録するファイルのパスまたはダウンロード先のファイルのパス（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，「6.8.12　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，「6.8.12　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，「6.8.11　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，「6.8.11　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，文書空間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，文書空間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，「6.7.4　文書空間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，「6.8.13　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。

(d) DbjSetVTFixLinkInfo インターフェース

構成管理型リンク（構成管理モードは FIX モード）を表すリンク設定情報を扱うインターフェースです。

(e) DbjSetVTFloatLinkInfo インターフェース

構成管理型リンク（構成管理モードは FLOATING モード）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，「6.8.9　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，「6.8.7　文書空間オブジェクトのプロパティの操作」を参照してください。

(18) DbjXmlUploadInfo インターフェース

XML 文書のアップロード情報を扱うインターフェースです。DbjUploadInfo インターフェースを継承しています。

なお，AIX の場合，DbjXmlUploadInfo インターフェースは使用できません。

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）
このプロパティは常に null になります。

● 構文解析レベル（parseLevel）

● マッピング定義名（mappingId）

● インデクス種別（indexType）

● フィルタリング定義ファイル（filterFilePath）

● 構文解析メッセージ（parseMessage）

このインターフェースの使用方法については，「6.10.2　XML 文書管理機能を使用する操作」を参照してください。

6.6 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。

6.6.1 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 サーバと接続して，文書空間にログインすることです。文書空間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。

文書空間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 サーバ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 Java 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，文書空間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，文書空間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，Java 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。

6.6.2 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

6.6.3 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。

例えば，セッションの切断（ログアウト処理）を明示的に実行していない場合でも，DocumentBroker サーバで設定されているタイムアウトによって，自動的にセッションが切断されている場合があります。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
        ......
  }
}
```

6.6.4 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**）。エラーが発生した場合，明示的に DbjSession#rollback メソッドを実行し，ロールバックしてください。

明示的に終了しない場合でも，DbjSession#logout メソッドまたは DocumentBroker サーバのタイムアウト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 propdef = new HashSet();
  propdef.add( "Name" );
```

```
  // メソッド単位のトランザクションの例
  // 文書空間オブジェクトから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();

  // 文書空間オブジェクトの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();

  // 文書空間オブジェクトに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。

6.6.5 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 propdef = new HashSet();
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();
```

6.7 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。

6.7.1 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。

6.7.2 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。

- 文書空間オブジェクトの作成
- 文書空間オブジェクトの検索
- 文書空間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，文書空間オブジェクトを検索して，検索条件に合致する文書空間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，特定の文書空間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，特定の文書空間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。

6.7.3 文書空間オブジェクトの作成

ここでは，文書空間オブジェクトの作成について説明します。

(1) 文書空間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，作成する文書空間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。

作成できる文書空間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 6-4 文書空間オブジェクトの種別と作成に使用するメソッド

文書空間オブジェクト	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
バージョン付きフォルダ	createVrFolder

文書空間オブジェクト	メソッド
独立データ	createIndependentData
パブリック ACL	createPublicACL

なお，文書空間オブジェクトの作成が成功した場合，作成した文書空間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，「6.8　文書空間オブジェクトの操作」を参照してください。

また，文書空間オブジェクト作成時には，次の情報を指定できます。

● 文書空間オブジェクトのトップオブジェクトクラス

● 文書空間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。

文書空間オブジェクトのトップオブジェクトクラス

文書空間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 DMA クラス（トップオブジェクトクラス）を指定できます。各文書空間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，「2.2.2　文書空間オブジェクトクラスと DMA クラスの対応」を参照してください。

DocumentBroker では，DMA 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 DMA クラスを文書空間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 3　システム導入・運用ガイド」を参照してください。

文書空間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，それらの値を文書空間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，既存の文書空間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。

作成する文書空間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，参照型リンクおよび構成管理型リンク※
- 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

バージョン付きフォルダを作成する場合

- フォルダとの直接型リンク，参照型リンクおよび構成管理型リンク※

注※　構成管理モードは，FIX モード，FLOATING モードのどちらも設定できます。

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空間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，「6.8.7(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 uploadlist = new ArrayList();
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 linklist = new ArrayList();
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，「4.5.2　全文検索機能付き文書クラスの作成」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。

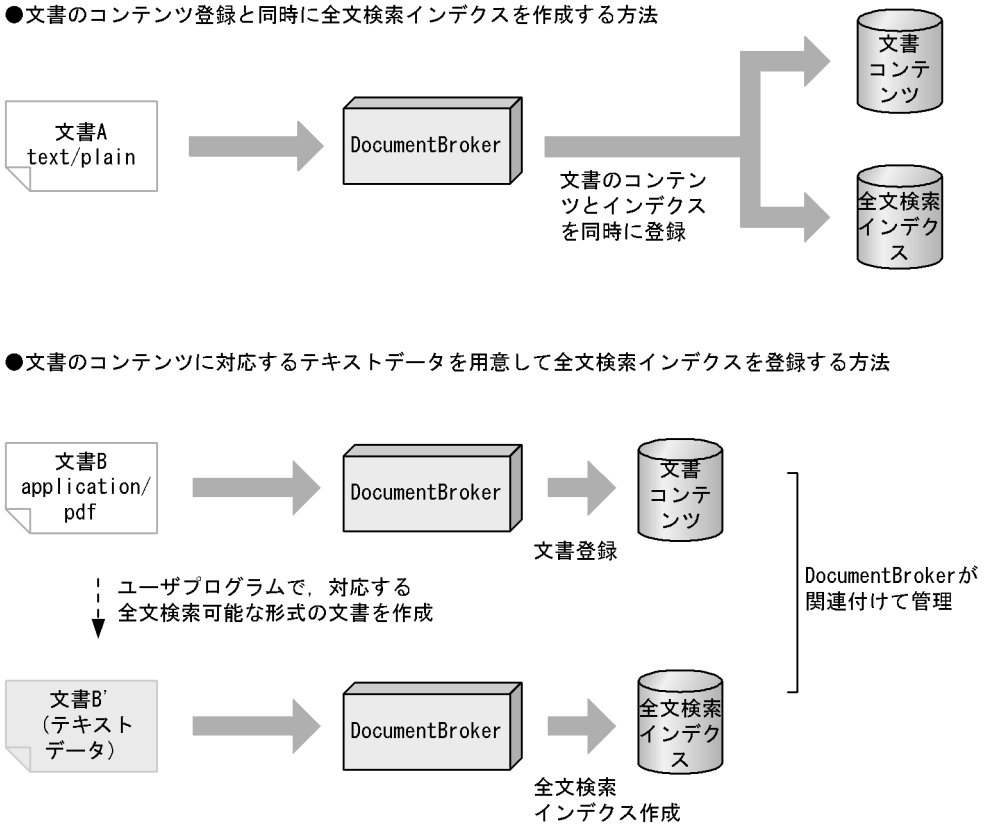
● 文書のコンテンツに対応するテキストデータを用意して全文検索インデクスを作成する方法

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 6-6　全文検索インデクスの作成方法



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 uploadlist = new ArrayList();
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
```

6.7.4 文書空間オブジェクトの検索

ここでは，文書空間オブジェクトの検索について説明します。

文書空間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，「4. 検索機能」および「5. edmSQL の文法」を参照してください。

文書空間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 サーバ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 qparams = new ArrayList();
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空間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，「4.3.3　アクセス制御機能付き検索」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，「4.3.4　名前付き検索結果を取得する検索」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 selectItems = new ArrayList();
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 selectItems = new ArrayList();
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 propNames = props.keySet();
```

検索結果集合に複数の DMA クラスの列（文書空間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（DMA クラス名）が指定できる形式を使用して，特定の DMA 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 DMA クラスのプロパティである dmaProp_OIID だけが取得できます。DMA 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 DMA 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，「4.3.5　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 qparamlst = new ArrayList();
    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，文書空間オブジェクトの一覧を取得する場合にも使用できます。

6.7.5 検索条件に合致した文書空間オブジェクトの削除

ここでは，検索条件に合致する文書空間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空間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空間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
```

6.7.6 既存の文書空間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，既存の文書空間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，文書空間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，「6.8　文書空間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。

// 文書空間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);
```

6.7.7 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，「6.11　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，
// 文書空間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");
```

6.8 文書空間オブジェクトの操作

この節では，文書空間オブジェクトの操作について説明します。

6.8.1 文書空間オブジェクトと Proxy オブジェクト

ここでは，文書空間オブジェクトと Proxy オブジェクトの関係について説明します。文書空間オブジェクトは，データベース上に存在する DMA 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。文書空間オブジェクトについての詳細は，「2.2　文書空間オブジェクト」を参照してください。

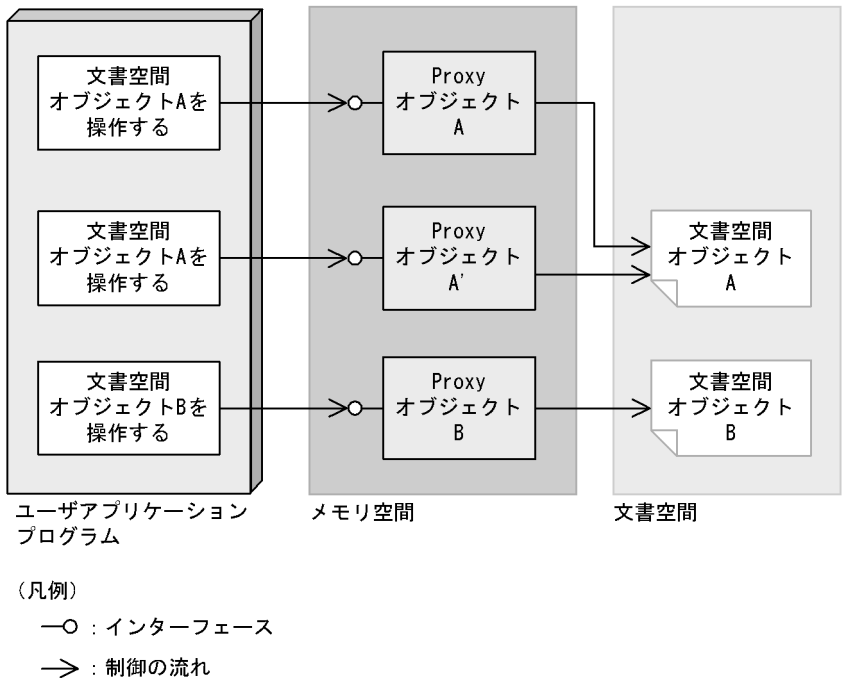
Proxy **オブジェクト**は，文書空間オブジェクトの概念的な**代理オブジェクト**です。Proxy オブジェクトは，メモリ空間上に存在します。

Java 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空間オブジェクトを操作する場合，文書空間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，文書空間オブジェクトを操作するためには，文書空間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空間オブジェクトの対応は，n:1（n は 1 以上の整数）です。一つの文書空間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，文書空間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空間オブジェクトを，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空間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 6-7　Proxy オブジェクトを通して文書空間オブジェクトを操作する例

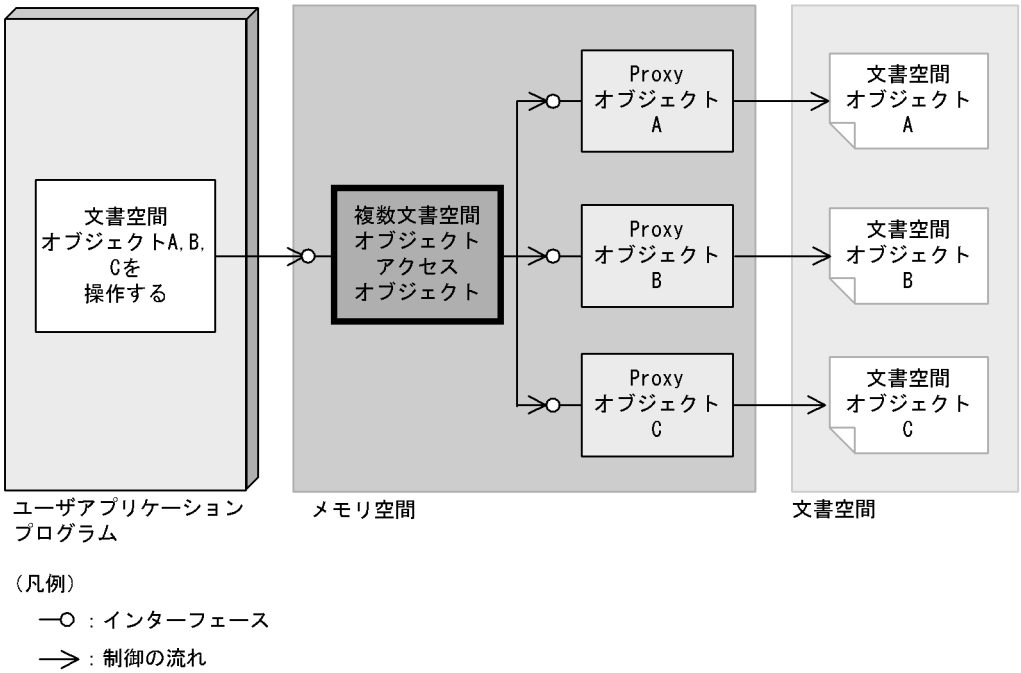


また，Java クラスライブラリでは，複数の文書空間オブジェクトをまとめて操作することもできます。複

数の文書空間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，**複数文書空間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。

複数の文書空間オブジェクトを操作する例を，次の図に示します。

図 6-8　複数の文書空間オブジェクトを操作する例



複数文書空間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，文書空間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，それ以外の文書空間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。

6.8.2　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，文書空間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。

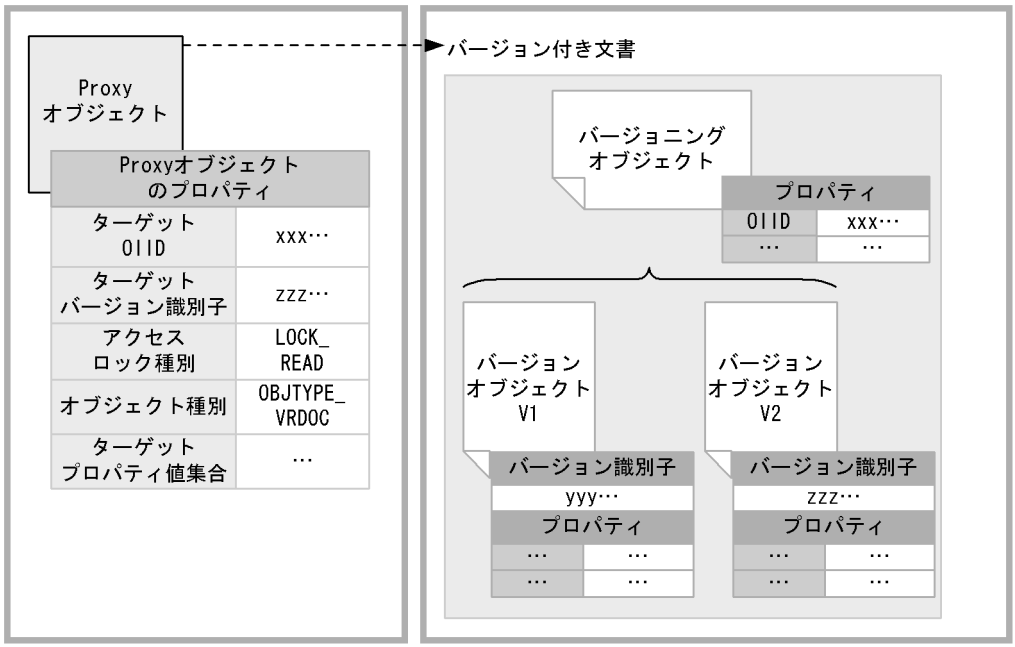
Proxy オブジェクトのプロパティを次に示します。

- 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空間オブジェクトの関係を，次の図に示します。

図 6-9 Proxy オブジェクトのプロパティと文書空間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 6-9 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
- バージョン付きフォルダ
- 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空間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。

文書空間オブジェクトのプロパティを参照する場合，まず，文書空間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，文書空間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，文書空間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。

文書空間オブジェクトのプロパティの操作についての詳細は，「6.8.7　文書空間オブジェクトのプロパティの操作」を参照してください。

6.8.3　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。

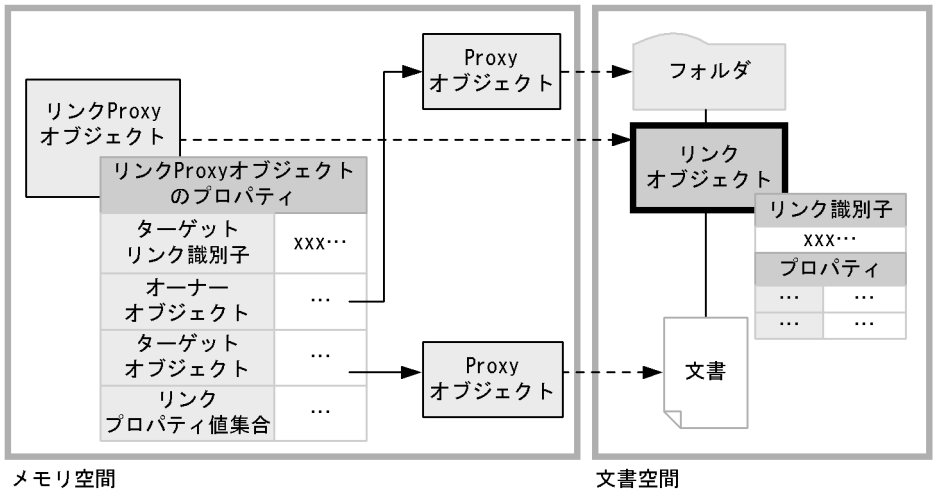
リンク Proxy オブジェクトはプロパティを持っています。このプロパティは，文書空間に存在するリンクオブジェクトのプロパティとは異なりますので，注意してください。

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 6-10　リンク Proxy オブジェクトのプロパティと文書空間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空間オブジェクトのうち，リンク元である文書空間オブジェクトの 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空間オブジェクトのうち，リンク先である文書空間オブジェクトの 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。

6.8.4 文書空間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，文書空間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) 文書空間オブジェクトを操作するインターフェースの種類

特定の文書空間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。Java クラスライブラリでは，文書空間オブジェクトを，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，文書空間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，どの文書空間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，操作する文書空間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。

文書空間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
個々の文書空間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，文書空間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，個々の文書空間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
複数の文書空間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
複数の文書空間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) 文書空間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，文書空間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。

文書空間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 6-5 文書空間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
文書空間オブジェクトの情報を取得する	OIID やオブジェクト種別，文書空間オブジェクトを構成する DMA クラスの情報を取得します。	6.8.5
文書空間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，文書空間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。	6.8.6
文書空間オブジェクトのプロパティを操作する	文書空間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。	6.8.7
文書空間オブジェクトを削除する	文書空間オブジェクトを削除します。	6.8.8
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。	6.8.9
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。	6.8.10
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。	6.8.11
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。	6.8.12
文書空間オブジェクトを関連付けて管理する	文書空間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空間オブジェクトを取得します。	6.8.13
複数の文書空間オブジェクトを一括して操作する	複数の文書空間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，複数の文書空間オブジェクトを一括して移動したり，削除したりします。	6.8.14
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 を作成して文書空間オブジェクトからバインドしたりします。	6.9
XML 文書管理機能を使用する	DbjXmlUploadInfo インターフェースおよび DbjXmlTranslator インターフェースを使用して XML 文書管理機能を使用します。	6.10

(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，実行できる文書空間オブ

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の詳細は，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」の各メソッドの説明を参照してください。

表 6-6　メソッドを実行できる文書空間オブジェクトのオブジェクト種別

メソッド	D	VD	F	VF	IP	PA
DbjObj インターフェース						
addRendition	○	○	×	×	×	×
bindPublicACL	○	○	○	○	○	×
cancelCheckOut	×	○	×	○	×	×
changeMasterRendition	○	○	×	×	×	×
changeToVTFix	×	×	×	○	×	×
changeToVTFloat	×	×	×	○	×	×
checkIn	×	○	×	○	×	×
checkOut	×	○	×	○	×	×
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	×	○	×	○	×	×

メソッド	D	VD	F	VF	IP	PA
unbindPublicACL	○	○	○	○	○	×
unlink	○	○	○	○	×	×
unlinkByLinkId	○	○	○	○	×	×
uploadContents	○	○	×	×	×	×
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またはバージョン付きフォルダのバージョンオブジェクト
VF：バージョン付きフォルダ
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空間オブジェクトで実行できるメソッドです。

6.8.5 文書空間オブジェクトの情報の取得

ここでは，文書空間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。

文書空間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
- DMA 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。文書空間オブジェクトの 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）

● バージョン付きフォルダ（DbjDef.OBJTYPE_VRFOLDER）

● 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） DMA クラス名の取得

文書空間オブジェクトを構成している DMA オブジェクトの DMA 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，または edmClass_ContainerVersion クラスもしくはそのサブクラス）の二つの DMA クラス名が取得できます。それ以外の文書空間オブジェクトの場合は，一つの DMA クラス名が取得できます。

ターゲットオブジェクトの DMA クラス名を取得する例を示します。

// DMAクラス名の取得例

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

6.8.6 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

文書空間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 Java クラスライブラリのオブジェクトまたは文書空間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 Java クラスライブラリのオブジェクトまたは文書空間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，文書空間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。

文書空間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，同じ文書空間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空間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，文書空間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。

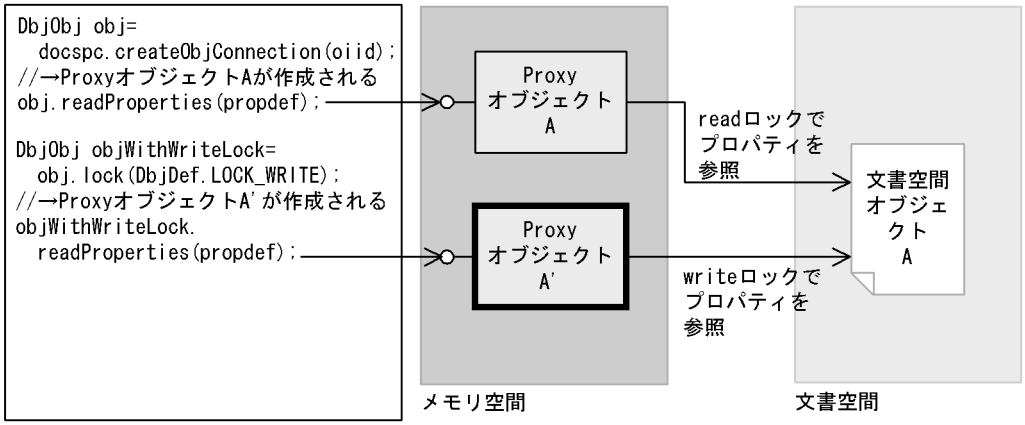
// 異なるロック種別で文書空間オブジェクトにアクセスする例

```
  // 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，文書空間オブジェクトに read ロックが設定されます。(b) では，文書空間オブジェクトに 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 6-11　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 6-11 の obj と objWithWriteLock は，同じ文書空間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。

6.8.7 文書空間オブジェクトのプロパティの操作

ここでは，文書空間オブジェクトのプロパティの操作について説明します。

文書空間オブジェクトのプロパティとは，文書空間オブジェクトが持っているプロパティです。DMA オブジェクト上に値が存在するものと，DMA 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。文書空間オブジェクトのプロパティの詳細については，「2.11　文書空間オブジェクトのプロパティ」を参照してください。

文書空間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，「3.9　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

文書空間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。

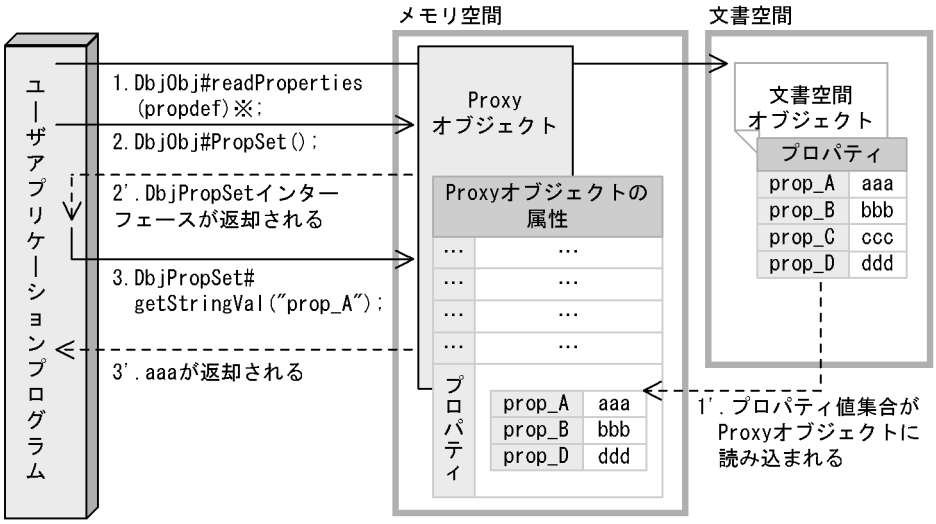
1. **文書空間オブジェクトのプロパティの値を，Proxy オブジェクトのターゲットプロパティ値集合プロパティにロードします。**
   ロードには，DbjObj#readProperties メソッドを使用します。
2. **プロパティ値集合を扱うインターフェース（DbjPropSet インターフェース）を取得します。**
   DbjPropSet インターフェースは，ターゲットプロパティ値集合プロパティにロードした Proxy オブジェクトのプロパティ値集合を扱うインターフェースです。DbjPropSet インターフェースは，DbjObj#propSet メソッドを実行して取得します。
   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，文書空間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 6-12　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 propdef = new HashSet();
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

文書空間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空間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空間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，一部だけを文書空間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空間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）の値を文書空間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 );

// 文書空間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，文書空間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 propdef = new HashSet();
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 );

// 文書空間オブジェクトに
// 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空間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 propdef = new HashSet();
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);

// 文書空間オブジェクトに
// Titleプロパティの値だけをフラッシュする
Set updateprop = new HashSet();
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 DMA 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。
- バージョン付きフォルダの場合は，edmClass_ContainerVersion 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。

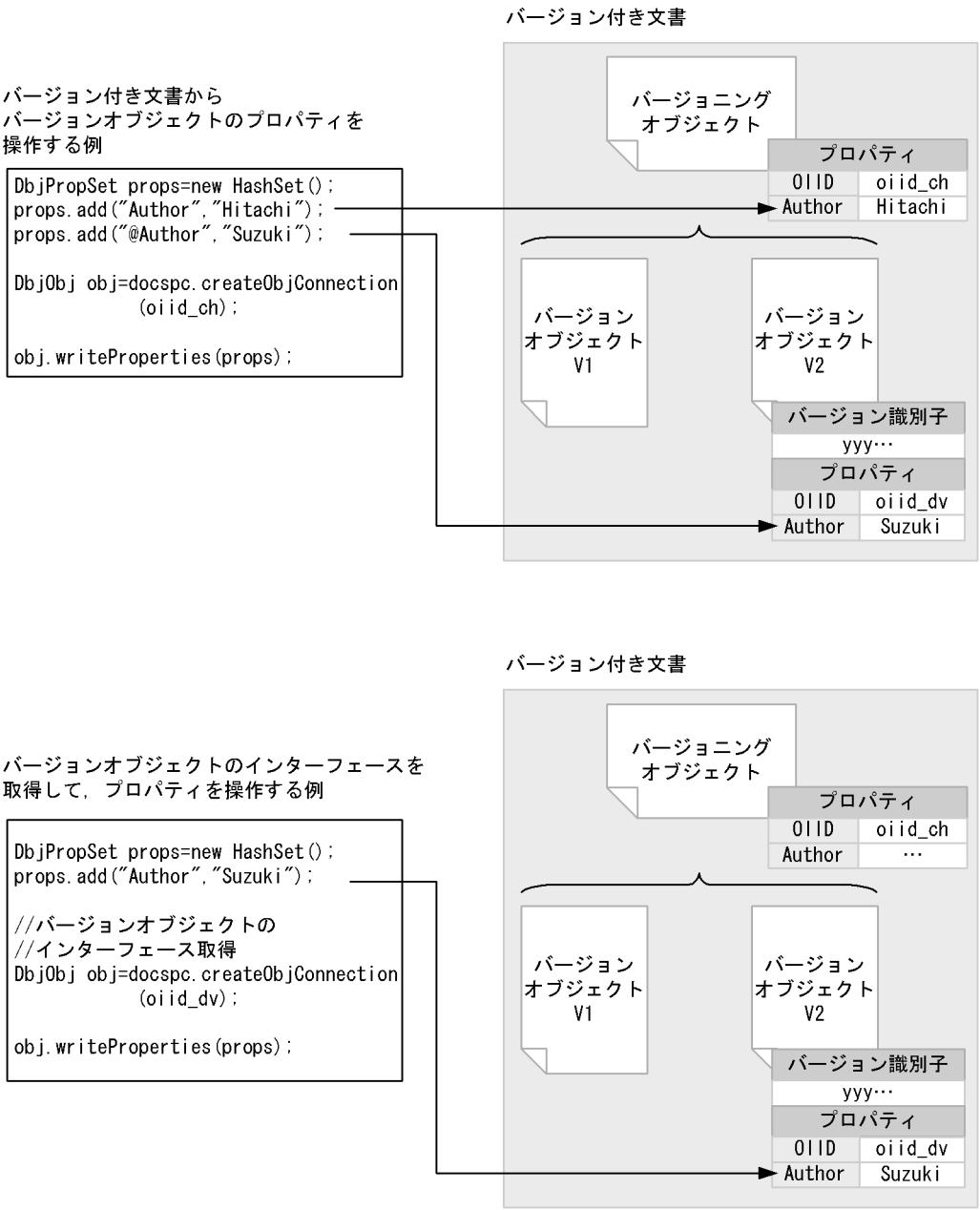
バージョニングオブジェクトのプロパティとバージョンオブジェクトのプロパティは，バージョンオブジェクトのプロパティ名に **@ プレフィックス**を付けて指定することで，明示的に区別して指定します。

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 6-13　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 propdef = new HashSet();
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。

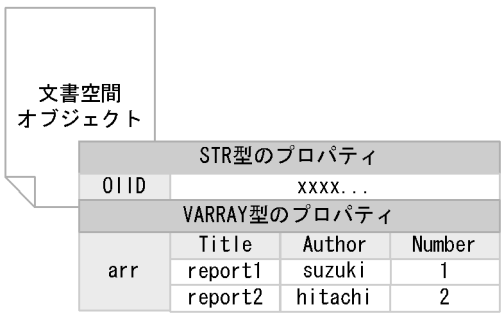
- Proxy オブジェクトに設定されているプロパティ値集合から DbjPropSet インターフェースを使用して取得する
- 検索結果集合から DbjResultSet インターフェースを使用して取得する

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空間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，次の図に示す文書空間オブジェクトの VARRAY 型のプロパティを対象にします。

図 6-14　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，文書空間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 6-14 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 propdef = new HashSet();
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 );

// 文書空間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，「6.8.11　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，「6.8.13　リンクの操作」を参照してください。

6.8.8 文書空間オブジェクトの削除

ここでは，文書空間オブジェクトの削除について説明します。

文書空間オブジェクトは，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空間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空間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。

文書空間オブジェクトを削除する例を示します。

// 文書空間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 );

// 文書空間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.");
}
```

6.8.9 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，「3.2　レンディション管理モデル」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空間オブジェクトの 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。文書空間オブジェクトの 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。また，サブレンディションのコンテンツを，「test.pdf」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" );
File dwlFile2 = new File( parentdir, "test.pdf" );

// 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() );

// サブレンディションのコンテンツを
// レンディションタイプを指定して，
// dwlFilePath2で指定したパスにダウンロードする
DbjContentInfo  continfo2 = obj.downloadContents(
                            "application/pdf",
                            dwlFile2.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 インターフェースまたは DbjXmlUploadInfo 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。

   DbjXmlUploadInfo インターフェースを使用する場合の指定については，「6.10.2(2) XML 文書の更新」を参照してください。
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，「4.5.3　全文検索インデクスの作成」を参照してください。また，DbjXmlUploadInfo インターフェースを使用して XML 形式のファイルをコンテンツとしてアップロードする場合には，アップロードと同時に構造指定検索用全文検索インデクスを作成・登録できます。「6.10.2(2) XML 文書の更新」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。DbjXmlUploadInfo インターフェースを使用した XML インデクスデータ作成機能も，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だけ有効です。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。また，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 DbjXmlUploadInfo インターフェースを指定したときは，DbjUploadInfo インターフェースとして扱われます。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，「6.8.11　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，マスタレンディションに対して，「tmp.html」というファイルを登録します。文書空間オブジェクトの 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
```

6.8.10 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，「3.1　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。

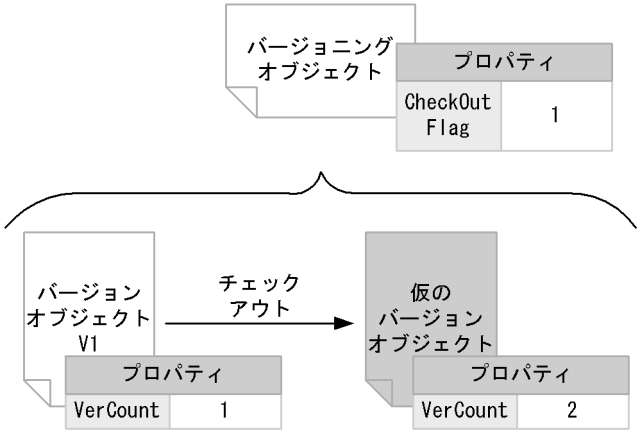
● 仮のバージョンオブジェクトの VerCount プロパティに，バージョン番号を示す値「チェックアウト前のバージョン番号 +1」を設定する。

// 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 propdef = new HashSet();
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 6-15　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。

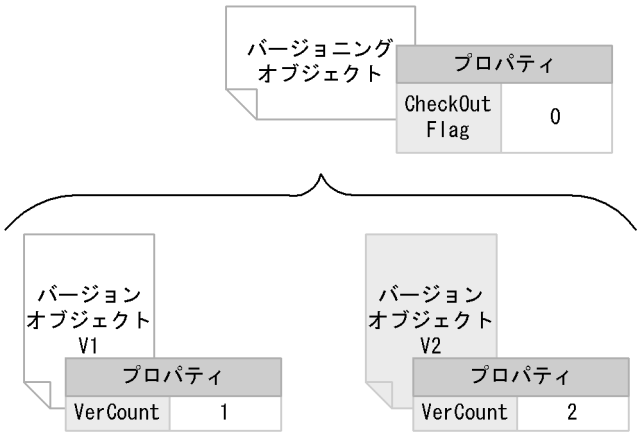
チェックインの例を示します。この例では，(1) でチェックアウトしたバージョン付きオブジェクトをチェックインします。同時に，バージョニングオブジェクトの CheckOutFlag プロパティの値を，チェックアウト中ではないことを示す値「0」に更新します。

// 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 6-16　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 propdef = new HashSet();
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空間オブジェクトに
// 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 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 dels = new ArrayList();
dels.add( versionId );
versioningObj.deleteVersion( dels );
```

6.8.11 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，「3.2　レンディション管理モデル」を参照してください。また，DocumentBroker Rendering Option と連携している場合は，サブレンディションのコンテンツを自動的に作成できます。DocumentBroker Rendering Option で実行できるレンディション変換については，マニュアル「DocumentBroker Rendering Option Version 3」または「DocumentBroker Rendering Option Version 3（活文 PDFstaff 編）」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。
- XML 文書管理機能を使用する場合，XML 文書のアップロード情報（DbjXmlUploadInfo インターフェース）は，リストの先頭に指定してください。リストの二つ目以降に指定した場合は，DbjUploadInfo インターフェースとして扱われるため，XML プロパティマッピング機能や XML インデクスデータ作成機能は使用できません。XML 文書を管理するための操作の詳細は，「6.10　XML 文書を管理するための操作」を参照してください。

また，DocumentBroker Rendering Option と連携している場合，マスタレンディションのコンテンツから，サブレンディションのコンテンツを自動作成することができます。自動作成する場合は，サブレンディションの文書のアップロード情報で，登録するファイルパスとして null を指定してください。

マルチレンディション文書を作成する例を示します。この例では，サブレンディションのコンテンツは DocumentBroker Rendering Option を使用して自動作成します。

// 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, // レンディション変換で作成
                      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 uplist = new ArrayList();

// 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。
- XML プロパティマッピング機能は使用できません。文書のアップロード情報として XML 文書のアップロード情報（DbjXmlUploadInfo インターフェース）を指定した場合，DbjUploadInfo インターフェースとして扱われます。

サブレンディションを追加する例を示します。この例では，追加するレンディションのコンテンツ（pdf 形式）は DocumentBroker Rendering Option を使用して自動的に作成します。

// 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, // レンディション変換で作成する
                      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 uplist = new ArrayList();

// 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 renlist = new ArrayList();
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。また，コンテンツ更新時に，DbjXmlUploadInfo インターフェースによって XML プロパティマッピング機能を実行することはできません。

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。また，レンディションタイプ変更時に XML プロパティマッピング機能を使用することはできません。

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 propdef = new HashSet();
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 propdef = new HashSet();
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 );
```

6.8.12 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メソッドの詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」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，「3.3　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，「6.8.10　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 uploadlist = new ArrayList();
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 linklist = new ArrayList();
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 );
```

6.8.13 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，文書空間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空間オブジェクトとは，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。

Java クラスライブラリで設定できるリンクは，次の 5 種類です。

- 直接型リンク
- 参照型リンク
- 構成管理モードが FIX モードの構成管理型リンク
- 構成管理モードが FLOATING モードの構成管理型リンク
- 文書間リンク

これらのリンクを使用した管理モデルについては，「3.5　リンクモデル」「3.6　バージョンなしフォルダによる管理モデル」および「3.7　バージョン付きフォルダによる管理モデル」を参照してください。

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 6-7　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 バージョン付きフォルダ	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 バージョン付きフォルダ
参照型リンク	バージョンなしフォルダ バージョン付きフォルダ	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 バージョン付きフォルダ
構成管理型リンク （構成管理モードは FIX モード， FLOATING モードどちらでも可）	バージョンなしフォルダ※1 バージョン付きフォルダ	バージョン付き文書※2 バージョン付きフォルダ
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

注※1　トップオブジェクトクラスが，edmClass_ContainerVersion クラスまたはそのサブクラス以外（dmaClass_Container クラスなど）の場合，構成管理型リンクは設定できません。
注※2　トップオブジェクトクラスが edmClass_VersionTracedDocVersion クラスまたはそのサブクラス以外（dmaClass_DocVersion クラスなど）の場合，構成管理の対象にはなりません。

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空間オブジェクトの参照

● 構成管理モードの変更（構成管理型リンクの場合）

● 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，文書空間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，リンク元の文書空間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) 文書空間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。

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

● 既存の文書空間オブジェクト同士をリンクで関連付ける方法

なお，文書空間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 は作成する文書空間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，文書空間オブジェクトの作成と同時に既存の文書空間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。文書空間オブジェクトの作成については，「6.7.3　文書空間オブジェクトの作成」を参照してください。

文書空間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。

// 文書空間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 linklist = new ArrayList();
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) 既存の文書空間オブジェクト間のリンクの設定

ここでは，既存の文書空間オブジェクトからほかの文書空間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 linklist = new ArrayList();
```

```
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，複数の文書空間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 childlist = new ArrayList();
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空間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空間オブジェクトは，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空間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空間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 6-8 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空間オブジェクト	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 構成管理型リンク※
getChildList	フォルダ	直接型リンク 参照型リンク 構成管理型リンク※
getRelList	文書	文書間リンク

注※ 構成管理モードは FIX モード，FLOATING モードどちらでも使用できます。ただし，フォルダの構成要素である DMA オブジェクトが，ContainerVersion オブジェクト（edmClass_ContainerVersion クラスまたはそのサブクラスを基に作成したオブジェクト）でない場合，構成管理型リンクは使用できません。

ここでは，直接型リンク，参照型リンクまたは構成管理型リンクをたどる参照と，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，参照型リンクまたは構成管理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，参照型リンクまたは構成管理型リンクをたどる参照について説明します。

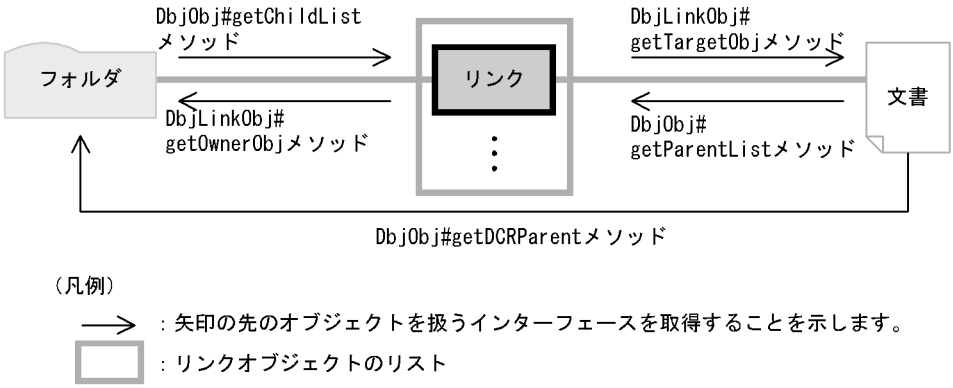
上位オブジェクトから下位オブジェクトを取得する場合，DbjObj#getChildList メソッドを使用します。このメソッドでは，上位オブジェクトから下位オブジェクトを関連付けているリンクオブジェクトのリストを扱うインターフェース（DbjLinkObjList インターフェース）を取得できます。メソッド実行時には，「直接型リンクだけを取得したい」，「直接型リンクと参照型リンクを取得したい」，「FLOATING モードの構成管理型リンクを取得したい」など，リンクの種別も指定できます。次に，DbjLinkObjList インターフェースの要素である DbjLinkObj インターフェースを取得して，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 6-17　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，「6.8.14　複数の文書空間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() );
}
```

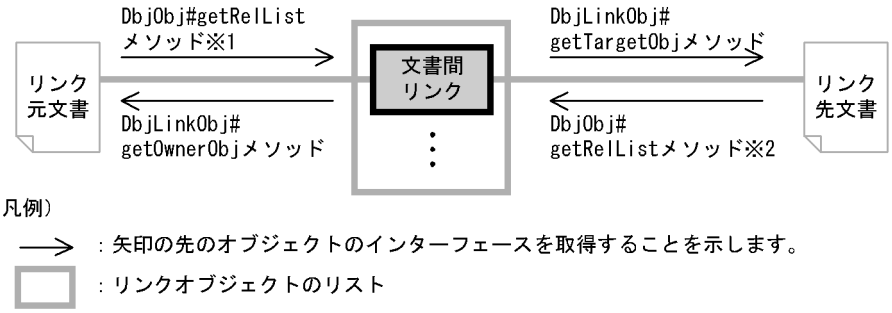
(b) 文書間リンクをたどる参照

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 6-18　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) 構成管理モードの変更

構成管理型リンクの構成管理モードは，次のどれかのインターフェースのメソッドで変更できます。

● バージョン付きフォルダ（DbjObj インターフェース）

● リンクオブジェクト（DbjLinkObj インターフェース）

● リンクオブジェクトのリスト（DbjLinkObjList インターフェース）

これらのインターフェースで，次のメソッドが提供されています。

● changeToVTFix メソッド
構成管理モードを FIX モードに変更するメソッドです。

● changeToVTFloat メソッド
構成管理モードを FLOATING モードに変更するメソッドです。

バージョン付きフォルダ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，構成管理モードを変更するリンクオブジェクトは，リンク識別子で特定します。リンク識別子のリストを指定することで，複数のリンクの構成管理モードを一度に変更できます。また，そのバージョン付きフォルダから FIX モードで関連付けているすべてのリンクをまとめて FLOATING モードに変更することもできます。

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，ターゲットリンクオブジェクトの構成管理モードを変更できます。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）を使用すると，リストに含まれる DbjLinkObj インターフェースのターゲットリンクオブジェクトの構成管理モードが，一括して変更できます。

構成管理モードを変更する例を示します。

// 構成管理モードを変更する例

```
  // docspc：DbjDocSpaceインターフェース

// バージョン付きフォルダのすべての構成管理型リンクを
// FIXモードに変更する（DbjObjインターフェースを使用する例）

DbjObj obj = docspc.createObjConnection( oiid );
obj.changeToVTFix();

// バージョン付きフォルダのすべての構成管理型リンクを
// FLOATINGモードに変更する
//  （DbjLinkObjListインターフェースを使用する例）

DbjLinkObjList linklist = obj.getChildList(
                            null,
                            null,
                            DbjDef.LINK_VCR,
                            DbjDef.OBJTYPE_ANY,
                            null );
linklist.changeToVTFloat();

// 一つの構成管理型リンクの構成管理モードを
// FIXモードに変更する（DbjLinkObjインターフェースを使用する例）
```

```
linklist.getLinkObj(0).changeToVTFix();
```

(6) 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，ほかの文書空間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，文書空間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
- DbjDocSpace#createVrFolder 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 );
```

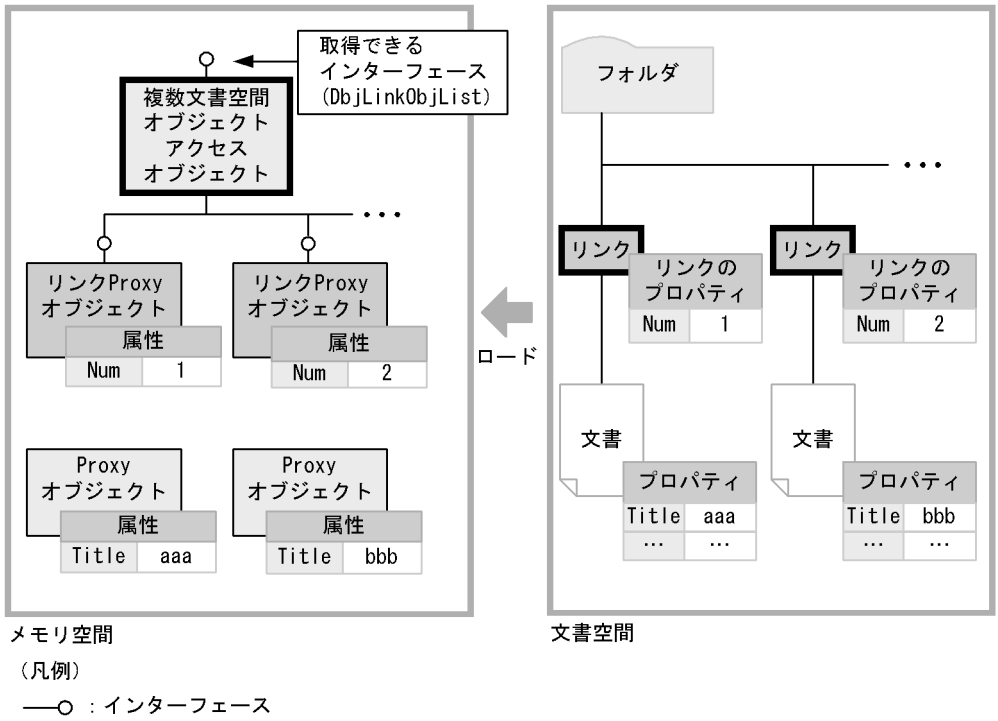
リンクのプロパティと同時に，下位オブジェクトのプロパティを参照する例を示します。

// 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 propdef = new HashSet();
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 linkpropdef = new HashSet();
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 6-19 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



(7) 文書空間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空間オブジェクトを，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。文書空間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 );
```

(8) 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 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();
```

6.8.14 複数の文書空間オブジェクトの一括操作

複数の文書空間オブジェクトの一括操作について説明します。これは，複数の文書空間オブジェクトに，1トランザクションでアクセスする機能です。

複数の文書空間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。

- 複数の文書空間オブジェクトのプロパティ一括操作
- 複数の文書空間オブジェクトに同じ種別のロックを設定した操作
- 複数の文書空間オブジェクトの一括削除
- 複数の文書空間オブジェクトの一括移動

なお，操作する文書空間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) 複数の文書空間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，複数の文書空間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，文書空間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，複数の文書空間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，「6.8.7　文書空間オブジェクトのプロパティの操作」を参照してください。

複数の文書空間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空間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。

// 複数の文書空間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 propdef = new HashSet();
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 );
}

// 文書空間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) 複数の文書空間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空間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。

複数の文書空間オブジェクトに 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空間オブジェクトに，write ロックを設定して操作します。

// 複数の文書空間オブジェクトに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) 複数の文書空間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，複数文書空間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。

複数の文書空間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。

// 複数文書空間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) 複数の文書空間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空間オブジェクト（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。複数の文書空間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 );
```

6.9 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，「3.10　アクセス制御モデル」を参照してください。

6.9.1 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，文書空間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，文書空間オブジェクトの 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，「6.8.7　文書空間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，文書空間オブジェクトの 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，可変長配列を文書空間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，独立した文書空間オブジェクトです。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。

文書空間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，文書空間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，文書空間オブジェクトの 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 を設定した文書空間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

6.9.2 ローカル ACL と ACE の操作

ローカル ACL は，文書空間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，「6.8.7　文書空間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 を文書空間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，文書空間オブジェクトにフラッシュします。**

文書空間オブジェクトにローカル ACL を設定する例を示します。

// 文書空間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空間オブジェクトにフラッシュする
obj.writeProperties( props );
```

6.9.3 パブリック ACL の操作

パブリック ACL は，独立した文書空間オブジェクトです。ここでは，パブリック ACL の作成と文書空間オブジェクトへのバインド，また，文書空間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，文書空間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空間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，ほかの文書空間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，同時に文書空間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 bindobjlist = new ArrayList();
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，文書空間オブジェクトから，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。

文書空間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。

// 文書空間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
```

```
  //   （文書空間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，文書空間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
  //   （文書空間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，文書空間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。

文書空間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。

// 文書空間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
  //   （文書空間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 propdef = new HashSet();
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空間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空間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空間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空間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空間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空間オブジェクトの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。

文書空間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。

// 文書空間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
  //    （文書空間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 unbindlist = new ArrayList();
unbindlist.add( docspc.createObjConnection( pacloiid ) );

// アンバインドを実行する
obj.unbindPublicACL( unbindlist );
```

6.10 XML 文書を管理するための操作

この節では，XML 文書を管理するための操作について説明します。

XML 文書の管理モデルについては，「3.4 XML 文書の管理モデル」を参照してください。

6.10.1 XML 文書を管理するためのインターフェースの機能

XML 形式のコンテンツを管理する場合，XML 文書管理機能が使用できます。XML 文書管理機能は，DbjXmlTranslator インターフェースと DbjXmlUploadInfo インターフェースの機能として提供されています。

DbjXmlTranslator **インターフェース**は，DbjMappedProp インターフェースを取得する getMappedProperties メソッドを提供しています。このメソッドを実行すると，マッピング定義ファイルと XML ファイルを基に，XML 文書作成時に指定する DMA クラスのリストが作成できます。また，XML 文書のコンテンツ更新時に再度プロパティマッピングをするためのプロパティ値集合が作成できます。

DbjXmlUploadInfo **インターフェース**は，パラメタクラスの DbjUploadInfo インターフェースのサブインターフェースです。XML 文書のコンテンツの新規登録時に，文書のアップロード情報として指定します。このインターフェースを使用すると，指定したコンテンツからマッピングするプロパティ値集合と全文検索インデクス用データが作成されます。

なお，XML 文書管理機能は，XML 文書の新規作成時と，XML 文書のコンテンツの更新時または XML 文書のプロパティの更新時では，使用方法が異なります。

XML 文書の新規作成時

XML 文書の新規作成時は，DbjXmlUploadInfo インターフェースで，マッピングするプロパティ値集合と構造指定検索用インデクスデータが作成できます。使用する DMA クラスがすでに決まっている場合は，DbjXmlTranslator インターフェースを使用しなくても，XML プロパティマッピング機能と XML インデクスデータ作成機能が実行できます。ただし，マッピング定義ファイルから DMA クラスのリストを取得したい場合は，DbjXmlTranslator インターフェースを使用します。
なお，マルチレンディション文書を作成する場合に XML 文書管理機能を使用したい場合は，XML 形式のファイルをマスタレンディションとして登録する必要があります。文書の作成時に指定する文書のアップロード情報のリストの，先頭の要素に XML 文書のアップロード情報（DbjXmlUploadInfo インターフェース）を指定します。

XML 文書のコンテンツ更新時

既存の XML 文書のコンテンツ更新時に XML プロパティマッピング機能および XML インデクスデータ作成機能を使用する場合は，DbjXmlTranslator インターフェースと DbjXmlUploadInfo インターフェースの両方を使用します。
XML プロパティマッピング機能には，DbjXmlTranslator インターフェースを使用します。DbjXmlTranslator#getMappedProperties メソッドを実行して，指定した XML 形式のファイルからプロパティ値集合を作成します。作成したプロパティ値集合は，DbjObj#writeProperties メソッドなどで文書空間オブジェクトに設定します。
XML インデクスデータ作成機能には，DbjXmlUploadInfo インターフェースを使用します。DbjObj#uploadContents メソッドなどの引数に，XML 形式のファイルをコンテンツとして指定した DbjXmlUploadInfo インターフェースを指定します。これによって，コンテンツから抽出された XML インデクスデータを基に，文書の全文検索インデクスが更新されます。

なお，XML 文書のコンテンツ更新時に XML インデクスデータ作成機能が使用できるのは，更新するコンテンツが XML 形式の場合に，XML 文書であるマスタレンディションを DbjXmlUploadInfo インターフェースを使用して更新するときだけです。

6.10.2 XML 文書管理機能を使用する操作

ここでは，XML 文書管理機能を使用する操作について説明します。

(1) XML 文書の新規作成

ここでは，XML 文書の新規作成について説明します。

まず，DbjXmlUploadInfo インターフェースだけを使用する例を示します。文書の作成に必要な DMA クラスの情報は抽出しません。

また，XML プロパティマッピング機能で XML 形式のファイルからマッピングするプロパティ以外にも，プロパティを設定します。作成した XML 文書には，プロパティ値集合に指定したプロパティとプロパティマッピング機能でマッピングされたプロパティがマージされて設定されます。なお，プロパティ値集合に指定したプロパティと同じプロパティが XML プロパティマッピング機能でマッピングされた場合は，XML プロパティマッピング機能でマッピングされた値が XML 文書のプロパティの初期値として設定されます。

// XML文書を登録する例1

```
  // 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();
// Authorプロパティの値を設定する
props.setPropVal("Author","hitachi");

// 文書のアップロード情報のリストを作成する
File uplFile = new File( parentdir, "test.xml" );

List uploadlist = new ArrayList();
uploadlist.add( factory.createXmlUploadInfo(
                 uplFile.getCanonicalPath(),
                 "test.xml",
                 "text/xml",
                 null,
                 DbjDef.XMLPARSE_NO_EXTERNAL_ENTITIES,
                 mappingId,
                 DbjDef.INDEXTYPE_STRUCTURED,
                 null ) );

// リンク設定情報のリストを作成する
List linklist = new ArrayList();
linklist.add( factory.createSetDCRLinkInfo(
                docspc.createObjConnection(parentoiid),
                null ) );

// XML文書を作成する
DbjObj obj = docspc.createDocument(
                 "mdmClass_Document",
                 props,
                 uploadlist,
                 linklist );
```

次に，マッピング定義ファイルから DMA クラスも抽出して，XML 文書を作成する例を示します。

// XML文書を登録する例2

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

// XMLプロパティマッピング機能を実行する
File uplFile = new File( parentdir, "test.xml" );

DbjXmlTranslator xmltranslator = factory.createXmlTranslator(
                    docspaceid,
                    null );
DbjMappedProp mappedProps = xmltranslator.getMappedProperties(
                    uplFile.getCanonicalPath(),
                    DbjDef.XMLPARSE_NO_EXTERNAL_ENTITIES,
                    mappingId );

// XML文書のアップロード情報を作成する
List uploadlist = new ArrayList();
uploadlist.add( factory.createXmlUploadInfo(
                uplFile.getCanonicalPath(),
                "test.xml",
                "text/xml",
                null,
                DbjDef.XMLPARSE_NO_EXTERNAL_ENTITIES,
                null,   // XMLプロパティマッピング機能は使用しない
                DbjDef.INDEXTYPE_STRUCTURED,
                null ) );

// XML文書を作成する
DbjObj obj = docspc.createVrDocument(
                mappedProps.getClassList.get(0),
                mappedProps.getClassList.get(1),
                mappedProps.propSet(),
                uploadlist,
                null );
```

(2) XML 文書の更新

ここでは，XML 文書のコンテンツ更新時に，更新する XML ファイルからプロパティをマッピングする方法について説明します。文書作成時以外には，XML 文書のアップロード情報による XML プロパティマッピング機能は使用できません。この場合は，DbjXmlTranslator インターフェースを使用して，コンテンツの更新とは別に，マッピングするプロパティ値集合を作成して，プロパティを更新します。

XML 文書作成時以外に XML プロパティマッピング機能を使用する例を示します。

// XML文書作成時以外にXMLプロパティマッピング機能を使用する例

```
  // factory：DbjFactoryインターフェース

// DbjXmlTranslatorインターフェースを取得する
DbjXmlTranslator xmlTran = factory.createXmlTranslator(
           docspaceid,
           null );

// XMLプロパティマッピング機能を実行して，
// DbjMappedPropインターフェースを取得する
File uplFile = new File( parentdir, "test.xml" );

DbjMappedProp mappedProps = xmlTran.getMappedProperties(
           upFile.getCanonicalPath(),
           DbjDef.XMLPARSE_NO_EXTERNAL_ENTITIES,
           mappingId );

// プロパティ値集合を作成して，文書空間オブジェクトに
// 設定する
DbjPropSet props = factory.createPropSet();
props.setPropVal(" Author", "hitachi" );
```

```
props.putAll( mappedProps.propSet() );

obj.writeProperties( props );
```

また，XML 文書のコンテンツ更新時に XML インデクスデータ作成機能を使用する場合は，XML 文書のアップロード情報を使用します。ただし，XML インデクスデータ作成機能が使用できるのは，コンテンツを更新する対象がマスタレンディションの場合だけです。

マスタレンディションである XML 文書のコンテンツを更新する例を示します。

// XML文書のコンテンツを更新する例

```
  // 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

// XML文書のアップロード情報を作成する
File uplFile = new File( parentdir, "test.xml" );

DbjXmlUploadInfo xmluploadinfo =
  factory.createXmlUploadInfo(
                 uplFile.getCanonicalPath(),
                 "test.xml",
                 "text/xml",
                 null,
                 DbjDef.XMLPARSE_NO_EXTERNAL_ENTITIES,
                 null,  // XMLプロパティマッピング機能は無効
                 DbjDef.INDEXTYPE_STRUCTURED,
                 null );

// XML文書のコンテンツを更新する
//   （XMLインデクスデータ機能が実行されて，
//     全文検索インデクスが更新される）
obj.uploadContents( null,
                    xmluploadinfo );
```

6.11 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。

6.11.1 メタ情報を取得するインターフェースの機能

DocumentBroker サーバ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 DMA クラスおよび文書空間オブジェクトのプロパティの定義情報です。

DMA 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，DMA クラスごとに定義されています。文書空間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。

6.11.2 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

● 文書空間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。

● DMA クラス名

● DMA 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，DMA 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 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 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() );
}
```

6.12 例外処理

ここでは，例外処理について説明します。

6.12.1 例外の種類

Java 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。

Java 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

● Java 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

Java 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，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」を参照してください。

6.12.2 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，文書空間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。詳細なエラー情報はメッセージとして syslog ファイル（UNIX の場合）またはイベントログ（Windows の場合）に出力されます。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，「8. 障害対策」を参照してください。

(3) Java 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，TPBroker でエラーが発生した場合など，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 クラスまたは DbjCORBAException 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，Java 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。

6.13 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 Development Kit および DocumentBroker Runtime 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() );
```

6.14 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，Java 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。

6.14.1 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。

表 6-9 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
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

6.14.2 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 6-10 トレース情報の出力先

出力先	Windows の場合	UNIX の場合
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	○	○
イベントログファイル	○	－
syslog 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。ト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 6-11 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	• Java 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 • Java 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※2	• Java 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 • Java 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 • Java 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※2	• Java 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 • Java 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
イベントログファイルまたは syslog ファイル	• Java 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 • Java 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	• msg	SYSTEM

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。

注※2

ファイルの出力先ディレクトリは，動作環境定義ファイルの APTracePath で指定します。動作環境定義ファイルについては，「7.6.3 動作環境定義ファイル」を参照してください。

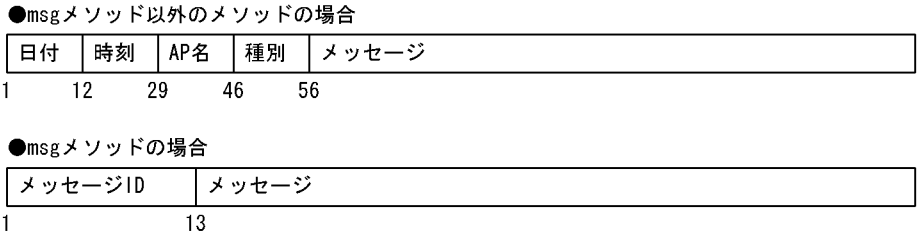
6.14.3 トレース情報の出力形式

トレース情報は，次に示す形式で出力されます。

(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 6-20 コマンドプロンプトの出力形式



注

- 数字は先頭からのバイト数です。
- 出力情報が 900 バイトを超える場合，901 バイト以降の情報は次の行に出力されます。

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 6-12　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 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。
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。

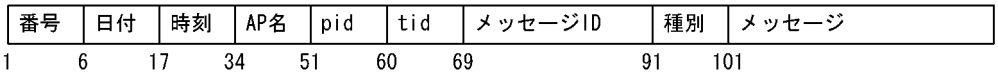
表 6-13　実行されたメソッドと種別に出力される文字列の対応

実行されたメソッド	種別に出力される文字列
enter	enter
exit	exit
call	call
returned	return
error	error
hint	hint
arg	arg
msg	msg

(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 6-21　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



注

- 数字は先頭からのバイト数です。
- 出力情報が 900 バイトを超える場合，901 バイト以降の情報は次の行に出力されます。

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。

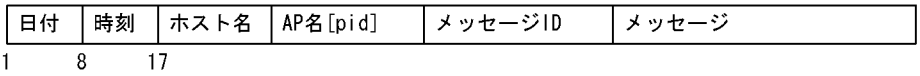
表 6-14 アプリケーショントレースファイルまたはアプリケーションエラーログファイルの出力項目

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。
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。

(3) イベントログファイルまたは syslog ファイルの出力形式

syslog ファイルの出力形式（UNIX の場合）を次の図に示します。イベントログファイルの出力形式（Windows の場合）は，イベントログのフォーマットに従います。

図 6-22 syslog ファイルの出力形式（UNIX の場合）



注

- 数字は先頭からのバイト数です。
- ホスト名が可変長のため，ホスト名以降の項目の開始バイト数は決まっていません。
- 出力情報が 900 バイトを超える場合，901 バイト以降の情報は次の行に出力されます。

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 6-15 syslog ファイルの出力項目（UNIX の場合）

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 の形式で出力されます。 hh:mm:ss 形式の hh には時（2 けた），mm には分（2 けた），ss には秒（2 けた）が出力されます。
ホスト名	ホスト名が出力されます。

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。
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。
メッセージ	英語のメッセージが出力されます。

6.14.4 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 6-16 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 ～ 9	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 ～ 19	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 ～ 29	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 APTraceLevel の値です。動作環境定義ファイルについては，「7.6.3　動作環境定義ファイル」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。

6.15 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。

7 環境の設定

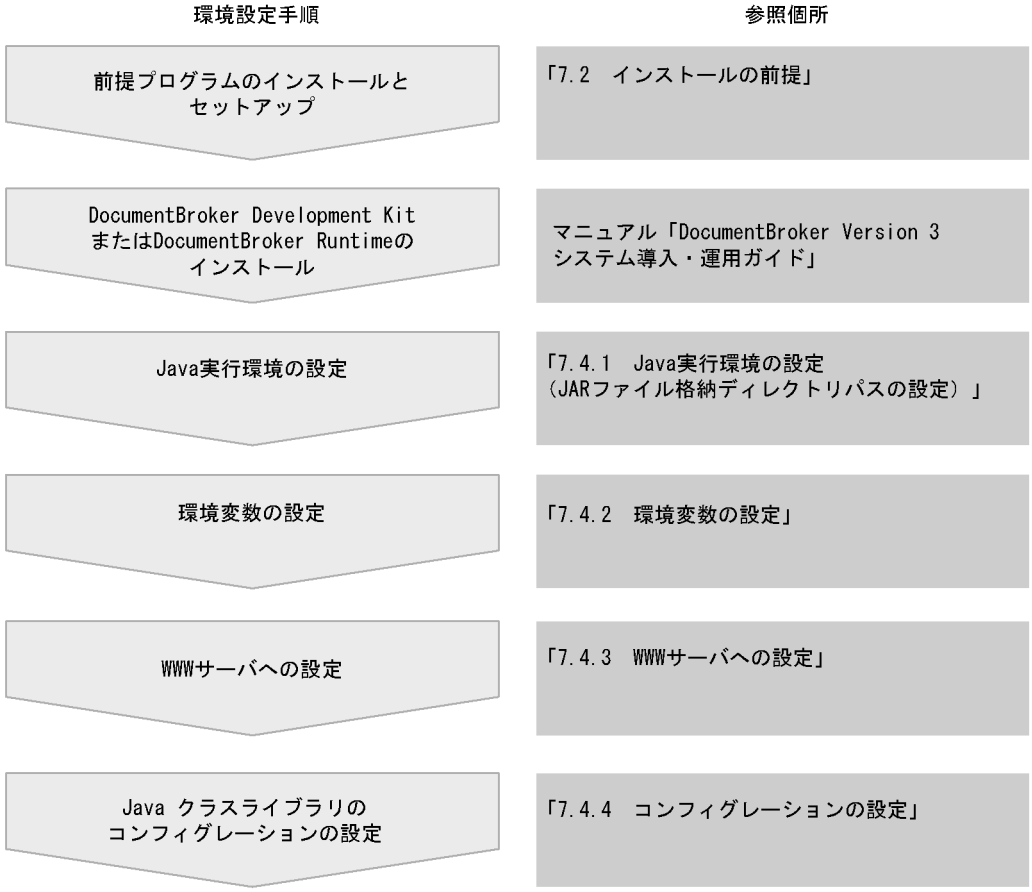
この章では，Java クラスライブラリを使用する環境の設定方法について説明します。

7.1　環境を設定する手順

この節では，DocumentBroker Development Kit および DocumentBroker Runtime の Java クラスライブラリを利用する環境の設定手順について説明します。

Java クラスライブラリを利用する環境の設定手順を次の図に示します。

図 7-1　環境設定の手順



注　このほか，DocumentBrokerクライアントとDocumentBrokerサーバを別マシンにインストールして運用する場合は，ファイル転送機能を開始する必要があります。

なお，サンプル Web アプリケーションを使用する場合は，このほかにサンプル Web アプリケーションの環境設定が必要です。詳細は，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　サンプル Web アプリケーション」を参照してください。

7.2 インストールの前提

この節では，DocumentBroker Development Kit または DocumentBroker Runtime をインストールする前に必要な前提の設定について説明します。

7.2.1 前提となる Java 実行環境のセットアップ

Java の実行環境をセットアップしておきます。

DocumentBroker Development Kit および DocumentBroker Runtime で提供する Java 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，実行するためには，Java 2 SDK. Standard Edition が必要です。開発したユーザアプリケーションプログラムを実行するためには，Java 2 Runtime Environment, Standard Edition が必要です。

また，JSP および Servlet が動作する実行環境もセットアップしておいてください。

Java の実行環境には Cosminexus を使用してください。

7.2.2 DocumentBroker サーバの実行環境のセットアップ

DocumentBroker サーバの実行環境をセットアップしておきます。また，データベースシステム（HiRDB），および必要な前提プログラムや連携プログラム（TPBroker，ディレクトリサービス，HiRDB Adapter for XML，Preprocessing Library for Text Search，DocumentBroker Rendering Option など）も使用できる状態にしておきます。

DocumentBroker サーバの設定方法の詳細については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

なお，DocumentBroker Development Kit（または DocumentBroker Runtime）を DocumentBroker サーバとは別のマシンにインストールして使用する場合はファイル転送機能を使用します。ファイル転送機能を使用するための環境設定については，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

7.2.3 WWW 環境の構築

Java クラスライブラリで開発したユーザアプリケーションプログラムは，WWW 環境から DocumentBroker にアクセスする業務アプリケーションとして使用されます。このため，WWW サーバと WWW ブラウザがイントラネットまたはインターネットで使用できる環境を準備しておきます。

また，DocumentBroker Development Kit および DocumentBroker Runtime で提供する Java クラスライブラリの API は，Enterprise JavaBeans のビジネスロジックを表すコードから呼び出したり，JavaBeans の中から呼び出したりして使用できます。これらを使用するための環境は，あらかじめセットアップしておきます。

Java クラスライブラリで開発したユーザアプリケーションプログラムを実行できる WWW ブラウザは Internet Explorer です。なお，WWW ブラウザ使用時には，次のことに注意してください。

WWW ブラウザ使用時の注意事項

- WWW ブラウザの Cookie についての設定を，Cookie を常に受け付ける設定にして使用してください。

- WWW ブラウザのキャッシュについての設定を，起動ページを表示するごとに文書を確認する設定にして使用してください。
- WWW ブラウザから文字を入力する時は，半角かたかな文字は使用しないでください。入力文字列の文字コードが正確に解析できなくなり，不正な動作をするおそれがあります。
- ボタンやリンクを選択する時は，ダブルクリックしないでください。連続して 2 回の処理が要求されて，エラーになる場合があります。

7.3　インストール

DocumentBroker Development Kit または DocumentBroker Runtime をインストールします。インストール方法の詳細は，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

7.4 インストール後の基本的な環境設定

この節では，DocumentBroker Development Kit または DocumentBroker Runtime をインストールしたあとに必要な基本的な環境設定について説明します。

7.4.1 Java 実行環境の設定（JAR ファイル格納ディレクトリのパスの設定）

Java クラスライブラリのパッケージの JAR ファイルは，次のディレクトリに格納されます。

UNIX の場合

TPBroker V3 を使用している場合

```
/opt/HiEDMS/client/java/lib
```

TPBroker V5 を使用している場合

```
/opt/HiEDMS/client/java/lib_tp5
```

Windows の場合

```
<インストールディレクトリ>¥java¥lib
```

この JAR ファイルのパスを，Java 実行環境に設定します。使用する実行環境に合わせてパスを追加してください。

例えば，Java 2 SDK, Standard Edition の場合は，付属している Java インタプリタ（java コマンド）を実行する時の -classpath オプションに，JAR ファイルのパスを追加してください。

Cosminexus での設定について説明します。なお，サーブレットエンジンモードで使用する場合と，J2EE サーバモードで使用する場合とで，設定する内容が異なります。

(1) サーブレットエンジンモードの場合

● Cosminexus Component Container の usrconf.cfg ファイルに，次のパスを追加設定します。

UNIX の場合

TPBroker V3 を使用している場合

```
web.add.class.path=/opt/HiEDMS/client/java/lib/Djlib.jar
add.library.path=/opt/HiEDMS/client/java/lib
```

TPBroker V5 を使用している場合

```
web.add.class.path=/opt/HiEDMS/client/java/lib_tp5/Djlib.jar
add.library.path=/opt/HiEDMS/client/java/lib_tp5
```

Windows の場合

```
web.add.class.path=<インストールディレクトリ>¥java¥lib¥Djlib.jar
add.library.path=<インストールディレクトリ>¥java¥lib
```

（例）

```
web.add.class.path=C:¥Program
Files¥HITACHI¥DocBroker¥DevKit¥java¥lib¥Djlib.jar
add.library.path=C:¥Program Files¥HITACHI¥DocBroker¥DevKit¥java¥lib
```

● Cosminexus Component Container の usrconf.properties ファイルに，次のキー値を追加設定します。

```
ejbserver.application.InitTermProcessClasses=jp.co.Hitachi.soft.docbroker.cli
ent.DbjInitTermCompoContainerEx
```

● Cosminexus Component Container の web.policy ファイルに，次のキー値を追加設定します。

UNIX の場合

TPBroker V3 を使用している場合

```
grant codeBase "file:///opt/HiEDMS/client/java/lib/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
grant codeBase "file://${webserver.install.root}/containers/
${webserver.serverName}/webapps/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
```

TPBroker V5 を使用している場合

```
grant codeBase "file:///opt/HiEDMS/client/java/lib_tp5/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
grant codeBase "file://${webserver.install.root}/containers/
${webserver.serverName}/webapps/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
```

Windows の場合

```
grant codeBase "file:///<インストールディレクトリ>/java/lib/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
grant codeBase "file://${webserver.install.root}/containers/
${webserver.serverName}/webapps/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
```

（例）

```
grant codeBase "file:///C:/Program Files/HITACHI/DocBroker/DevKit/java/lib/-"
{
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
grant codeBase "file://${webserver.install.root}/containers/
${webserver.serverName}/webapps/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
```

（2） J2EE サーバモードの場合

● Cosminexus Component Container の usrconf.cfg ファイルに，次のパスを追加設定します。

UNIX の場合

TPBroker V3 を使用している場合

```
add.class.path=/opt/HiEDMS/client/java/lib/Djlib.jar
add.library.path=/opt/HiEDMS/client/java/lib
```

TPBroker V5 を使用している場合

```
add.class.path=/opt/HiEDMS/client/java/lib_tp5/Djlib.jar
add.library.path=/opt/HiEDMS/client/java/lib_tp5
```

Windows の場合

```
add.class.path=<インストールディレクトリ>¥java¥lib¥Djlib.jar
add.library.path=<インストールディレクトリ>¥java¥lib
```

（例）

```
add.class.path=C:¥Program Files¥HITACHI¥DocBroker¥DevKit¥java¥lib¥Djlib.jar
add.library.path=C:¥Program Files¥HITACHI¥DocBroker¥DevKit¥java¥lib
```

● Cosminexus Component Container の usrconf.properties ファイルに，次のキー値を追加設定します。

```
ejbserver.application.InitTermProcessClasses=jp.co.Hitachi.soft.docbroker.cli
ent.DbjInitTermCompoContainerEx
```

● Cosminexus Component Container の server.policy ファイルに，次のキー値を追加設定します。

```
grant codeBase "file:${ejbserver.http.root}/web/${ejbserver.serverName}/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read, write";
};
```

7.4.2 環境変数の設定

環境変数の設定について説明します。

なお，Java 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・実行するためには，DocumentBroker Development Kit または DocumentBroker Runtime が動作するための環境変数が設定されている必要があります。

ここでは，DocumentBroker Development Kit または DocumentBroker Runtime の環境変数について説明します。そのほかの前提プログラムおよび関連プログラムで設定が必要な環境変数については，それぞれの前提プログラムおよび関連プログラムのマニュアルを参照してください。

また，DocumentBroker クライアントと DocumentBroker サーバを別マシンにインストールして使用する場合，下記の環境変数のほかに，ファイル転送機能を使用するための環境変数の設定が必要になります。ファイル転送機能については，「7.4.5　ファイル転送機能を使用するための環境設定」を参照してください。

(1) AIX の場合

ここでは，AIX の場合に設定が必要な環境変数について説明します。

次の環境変数を設定してください。

● LANG

環境変数「LANG」には，使用する文字コードセットを設定します。DocumentBroker の文書空間で使用する文字コード種別に合わせて値を設定してください。

文書空間で使用する文字コード種別が Shift-JIS の場合

設定する値は「Ja_JP」です。

文書空間で使用する文字コード種別が UTF-8 の場合

使用する言語に合わせて適切な値を設定してください。

ただし，次のときは「C」を設定してください。

- 文書空間で使用する文字コード種別に関係なく，ASCII コードのデータだけを扱う英語環境の場合
- 一つのアプリケーションから文字コード種別の異なる複数の文書空間に接続する場合に，接続先に文字コード種別が UTF-8 の文書空間が含まれるとき

なお，この環境変数に設定する値によって，メッセージの言語種別が変わります。「Ja_JP」を設定した場合は，日本語のメッセージが出力されます。「Ja_JP」以外を設定した場合は，英語のメッセージが出力されます。

● TZ

環境変数「TZ」には，マシン時間のタイムゾーンを設定します。設定する値は「JST-9」です。

● LIBPATH

環境変数「LIBPATH」には，次に示すライブラリが格納されているディレクトリを設定します。

- DocumentBroker Development Kit または DocumentBroker Runtime のライブラリ
- DocumentBroker Development Kit または DocumentBroker Runtime の前提プログラムのライブラリ

次の値を追加してください。

TPBroker V3 を使用している場合

```
:/opt/HiEDMS/client/lib
:/opt/HiEDMS/client/java/lib
:/opt/HiEDMS/ACLibrary/lib
```

```
:/opt/hitachi/common/lib
:/usr/vacpp/lib※
:/opt/TPBroker/lib
```

TPBroker V5 を使用している場合

```
:/opt/HiEDMS/client/lib_tp5
:/opt/HiEDMS/client/java/lib_tp5
:/opt/HiEDMS/ACLibrary/lib_tp5
:/opt/hitachi/common/lib
:/usr/vacpp/lib※
:TPBroker V5インストールディレクトリ/lib
```

注※　VisualAge C++ Professional for AIX または IBM XL C/C++ Enterprise Edition for AIX のライブラリの格納ディレクトリを設定します。ここでは，デフォルトのインストールディレクトリにインストールされている場合の値を設定しています。

● DBJ_CONF_PATH

環境変数「DBJ_CONF_PATH」には，動作環境定義ファイルが格納されているディレクトリを，ファイル名を含む絶対パスで設定します。文書空間で使用する文字コード種別が UTF-8 の場合は，印刷可能な ASCII コードで記述してください。次の値を設定します。

```
/opt/HiEDMS/client/java/etc/conf.properties
```

● XDK_HOME

環境変数「XDK_HOME」には，SystemManager オブジェクトのレジストリファイルのロケーションを設定します。設定する値は「/opt/HiEDMS/client/etc」です。

(2) Windows の場合

ここでは，Windows の場合に設定が必要な環境変数について説明します。

なお，環境変数「PATH」と「DBJ_CONF_PATH」は，DocumentBroker Development Kit または DocumentBroker Runtime のインストール時にインストーラが値を設定します。

次の環境変数を設定してください。

● TZ

環境変数「TZ」には，マシン時間のタイムゾーンを設定します。設定する値は「JST-9」です。

● PATH

環境変数「PATH」には，次に示すライブラリが格納されているディレクトリを設定します。

- DocumentBroker Development Kit または DocumentBroker Runtime のライブラリ
- DocumentBroker Development Kit または DocumentBroker Runtime の前提プログラムのライブラリ

次の値を追加します。

```
<インストールディレクトリ>¥lib
<インストールディレクトリ>¥java¥lib
```

● DBJ_CONF_PATH

環境変数「DBJ_CONF_PATH」には，動作環境定義ファイルが格納されているディレクトリを，ファイル名を含む絶対パスで設定します。文書空間で使用する文字コード種別が UTF-8 の場合は，印刷可能な ASCII コードで記述してください。次の値を設定します。

```
<インストールディレクトリ>¥java¥etc¥conf.properties
```

なお，メッセージの言語種別はコントロールパネルの「地域と言語のオプション」で設定してください。

7.4.3 WWW サーバへの設定

Java クラスライブラリで開発したユーザアプリケーションプログラムを，Web アプリケーションとして実行するための設定をします。作成したユーザアプリケーションプログラムの形態や実行環境に応じて，Servlet の設定など，必要な設定をしてください。

7.4.4 コンフィグレーションの設定

Java クラスライブラリが動作するための定義を設定します。Java クラスライブラリの動作は，動作環境定義ファイルに定義します。

また，動作環境定義ファイルには，Java クラスライブラリで使用する定義ファイルのパスも指定します。定義ファイルは，インストールされたサンプルを基に，必要に応じて編集して使用します。

動作環境定義ファイルの定義方法および定義ファイルの編集方法については，「7.6 定義ファイルの作成と編集」を参照してください。

7.4.5 ファイル転送機能を使用するための環境設定

DocumentBroker サーバに接続するクライアントが異なるマシン上に存在する場合，サーバとクライアントの間のファイル転送には，ファイル転送機能を使用する必要があります。例えば，DocumentBroker で管理している文書のファイルを異なるクライアントマシンに取得したり，異なるクライアントマシンに存在するファイルを DocumentBroker の文書として登録したりする場合に，ファイル転送が発生します。

ファイル転送機能は，ファイル転送サービスが提供しています。ファイル転送サービスを使用するための環境設定については，マニュアル「DocumentBroker Version 3 システム導入・運用ガイド」のファイル転送機能の使用についての説明を参照してください。

7.5 プログラミング時の注意と設定

この節では，Java 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。

7.5.1 使用できる文字コード種別

Java 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空間オブジェクトを操作したりする場合，Java 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）

Java 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，動作環境定義ファイルのコンフィグレーションキー「DocSpaceCharacterSet」の内容に合わせる必要があります。一つのアプリケーションから文字コード種別が異なる複数の文書空間に接続する場合は，「DocSpaceCharacterSet」に「FROMSERVER」を設定すれば，Shift-JIS および UTF-8 のどちらも使用できます。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。

なお，文書空間で使用する文字コード種別が UTF-8 の場合，Java クラスライブラリの XML 文書管理機能は使用できません。文書空間で使用する文字コード種別が UTF-8 の場合に使用できるクラス，インターフェースおよびメソッドについては，マニュアル「DocumentBroker Version 3　クラスライブラリ Java　リファレンス」を参照してください。

7.6 定義ファイルの作成と編集

この節では，Java クラスライブラリで使用する定義ファイルの編集方法について説明します。

Java クラスライブラリで使用する定義ファイルは，次の 3 種類です。

- クラス定義情報ファイル
- レンディション定義ファイル
- 動作環境定義ファイル

なお，レンディション定義ファイルおよび動作環境定義ファイルは，インストール実行時に，サンプルファイルが次のディレクトリに格納されます。必要に応じて編集して使用してください。

UNIX の場合

```
/opt/HiEDMS/client/java/etc/sample
```

Windows の場合

```
<インストールディレクトリ>¥java¥etc¥sample
```

7.6.1 クラス定義情報ファイル

クラス定義情報ファイルは，DMA クラスやプロパティの GUID の値と名前とを対応させて，DMA クラスやプロパティに名前でアクセスするために Java クラスライブラリが参照するファイルです。

(1) クラス定義情報ファイルの作成方法

DocumentBroker サーバが提供するクラス定義情報ファイル作成コマンド（EDMCrtSimMeta コマンド）を実行して作成してください。コマンドについての詳細は，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

このファイルは，文書空間に一つ必要です。ユーザアプリケーションプログラムを実行する前に，必ずこのファイルを作成してください。

(2) クラス定義情報ファイルの格納先

クラス定義情報ファイルのファイル名は，次の名称にしてください。

```
<文書空間識別子>.ini
```

なお，< 文書空間識別子 > の形式は，16 進数で，8 けた -4 けた -4 けた -4 けた -12 けたの GUID 値の形式です。

クラス定義情報ファイルは，次のディレクトリに格納してください。

UNIX の場合

```
/opt/HiEDMS/client/java/etc
```

Windows の場合

```
<インストールディレクトリ>¥java¥etc
```

なお，Windows では，クラス定義情報ファイルが存在しない場合，セッションの確立時（ログイン時）に DbjInitializeError がスローされます。

7.6.2 レンディション定義ファイル

ここでは，レンディション定義ファイルについて説明します。

レンディション定義ファイルは，文書のコンテンツを登録する時に，登録するファイルの拡張子からレンディションタイプ（MIME 形式）を自動的に設定するために使用するファイルです。

(1) レンディション定義ファイルの名称と格納先

レンディション定義ファイルの名称は，次のとおりです。

```
mime.properties
```

レンディション定義ファイルは，次のディレクトリに格納してください。

UNIX の場合

```
/opt/HiEDMS/client/java/etc
```

Windows の場合

```
<インストールディレクトリ>¥java¥etc
```

(2) レンディション定義ファイルの記述形式

レンディション定義ファイルは，必要に応じてカスタマイズして使用できます。

レンディション定義ファイルの記述形式は，Java のプロパティファイルの記述形式に従います。文書空間で使用する文字コード種別が UTF-8 の場合は，印刷可能な ASCII コードで記述してください。

形式は，次のとおりです。

```
拡張子 = レンディションタイプ
```

各行のキーに拡張子を指定して，キーの値に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は，MIME 形式で指定します。ただし，先頭の「MIME::」省略して指定してください。

複数の拡張子に同じレンディションタイプを割り当てる場合も，拡張子ごとに 1 列ずつ，同じレンディションタイプを指定してください。

記述例を示します。

（例）

```
txt = text/plain
htm = text/html
html = text/html
```

(3) ファイルの内容の有効期間

レンディション定義ファイルの内容は，Java クラスライブラリを初期化するときに読み込まれます。一度読み込まれたあとは，プロセスが終了するまで再読み込みはされません。

7.6.3 動作環境定義ファイル

Java クラスライブラリの動作については，動作環境定義ファイルに定義します。動作環境定義ファイルには，デフォルトの文書空間識別子およびトレースファイルに関する情報が定義できます。

(1) 動作環境定義ファイルの名称と格納先

動作環境定義ファイルの名称は，次のとおりです。

```
conf.properties
```

動作環境定義ファイルは，次のディレクトリに格納してください。

UNIX の場合

```
/opt/HiEDMS/client/java/etc
```

Windows の場合

```
<インストールディレクトリ>¥java¥etc
```

(2) 動作環境定義ファイルの記述形式

動作環境定義ファイルは，環境に応じて編集してください。

動作環境定義ファイルの記述形式は，Java のプロパティファイルの記述形式に従います。文書空間で使用する文字コード種別が UTF-8 の場合は，印刷可能な ASCII コードで記述してください。

各行のキーにコンフィグレーションキーを指定して，キー値にその設定値を指定します。

形式は，次のとおりです。

```
コンフィグレーションキー = 設定値
```

(3) コンフィグレーションキーの詳細

動作環境定義ファイルに設定するコンフィグレーションキーと，設定する値の詳細を，次の表に示します。なお，指定した値が，使用できる値の範囲を満たさない場合は，省略時の値が仮定されます。

表 7-1 コンフィグレーションキー

コンフィグレーションキー	設定値	省略値	説明
APTracePath	文字列（AIX の場合：1,022 バイト以内，Windows の場合：260 バイト以内）	• **AIX の場合** /opt/HiEDMS/client/java/log • **Windows の場合** <インストールディレクトリ>¥java¥log	次に示すファイルを出力するディレクトリをフルパスで設定します。 • アプリケーショントレースファイル「DbjAPTrace プロセス ID（0x で始まる 16 進数）_ トレースファイル番号 .log」 • アプリケーションエラーログファイル「DbjAPTraceErr プロセス ID（0x で始まる 16 進数）_ エラーログファイル番号 .log」 なお，パスの区切り文字は，AIX の場合は「/」，Windows の場合は「¥¥」を使用してください。 指定例：C:¥¥DocumentBroker¥¥log

コンフィグレーションキー	設定値	省略値	説明
APTraceLevel	数値（0 ～ 30）	10	トレースレベルを設定します。 この設定は，次に示す出力先にトレース情報を出力する場合に有効になります。 • コマンドプロンプト（Windows の場合） • 標準出力・標準エラー出力（AIX の場合） • アプリケーショントレースファイル
APTraceSize	数値（4,096 ～ 2,147,483,647）	1,000,000	アプリケーショントレースファイルのファイルサイズの上限を設定します（単位：バイト）。
APTraceNumber	数値（0 ～ 16）	2	アプリケーショントレースファイルのファイルサイズの上限を超えた場合に，切り替えるファイルの数を設定します。 切り替えるファイルの数の上限を超えると，最初のファイルに戻って出力されます。このとき，ファイルは上書きされます。 0 を設定した場合は，アプリケーショントレースファイルを出力しません。
APErrorLogSize	数値（4,096 ～ 2,147,483,647）	1,000,000	アプリケーションエラーログファイルのファイルサイズの上限を設定します（単位：バイト）。
APErrorLogNumber	数値（1 ～ 16）	2	アプリケーションエラーログファイルのファイルサイズの上限を超えた場合に，切り替えるファイルの数を設定します。 切り替えるファイルの数の上限を超えると，最初のファイルに戻って出力されます。このとき，ファイルは上書きされます。
DefaultDocSpaceId ※1	文字列（8-4-4-4-12 形式の GUID 文字列）	DbjDef.DEFAULT_DOCSPCID	デフォルトの文書空間識別子を設定します。
EnableFunctionFlag ※1※2	数値（0x00000000 ～ 0xffffffff）	0x00000000	オブジェクトを操作したときの動作を bit フラグで指定します。ユーザは動作環境定義ファイルに記載した値を変更しないでください。なお，新規インストール時に記載されている値は 0x00000003 です。
PromptOutput	STDOUT または STDERR	STDOUT	トレース情報を標準出力に出力するか，標準エラー出力に出力するかを次の文字列で設定します。文字列は，すべて大文字で設定します。 • STDOUT：標準出力 • STDERR：標準エラー出力
TracePath	文字列（AIX の場合：1,022 バイト以内，Windows の場合：260 バイト以内）	• **AIX の場合** /opt/HiEDMS/client/java/log • **Windows の場合** <インストールディレクトリ>¥java¥log	クライアント共用トレースファイル※3 を出力するディレクトリをフルパスで設定します。 なお，パスの区切り文字は，AIX の場合は「/」，Windows の場合は「¥¥」を使用してください。 指定例：C:¥¥DocumentBroker¥¥log
TraceSize	数値（4,096 ～ 2,147,483,647）	1,000,000	クライアント共用トレースファイルのファイルサイズの上限を設定します（単位：バイト）。

コンフィグレーションキー	設定値	省略値	説明
TraceNumber	数値（1 ～ 16）	2	クライアント共用トレースファイルのファイルサイズの上限を超えた場合に，切り替えるファイルの数を設定します。切り替えるファイルの数の上限を超えると，最初のファイルに戻って出力されます。このとき，ファイルは上書きされます。
DocSpaceCharacterSet	FROMSERVER，SJIS，または UTF-8	FROMSERVER	文書空間で使用する文字コード種別を，次の文字列で設定します。文字列は，すべて大文字で設定します。 • FROMSERVER： 文書空間へのログイン時に，文書空間で使用する文字コード種別の設定を DocumentBroker Server から受け取り，その設定に合わせた文字コード種別を使用します。 この文字列は，文字コード種別が異なる複数の文書空間に接続する場合に設定します。 なお，文書空間へのログイン前は無条件で Shift-JIS を使用するため，メソッドはログイン後に実行してください。 • SJIS： Shift-JIS を使用します。 この文字列は，ログインする文書空間の文字コード種別がすべて Shift-JIS の場合に設定します。ログインする文書空間の文字コード種別と異なる文字コード種別を設定した場合の動作は保証しません。 • UTF-8： UTF-8 を使用します。 この文字列は，ログインする文書空間の文字コード種別がすべて UTF-8 の場合に設定します。ログインする文書空間の文字コード種別と異なる文字コード種別を設定した場合の動作は保証しません。

注※ 1

同一文書空間へ複数のクライアント環境から接続する構成の場合，コンフィグレーションキーの設定値は各クライアント環境の動作環境定義ファイルで同じ値を指定してください。

注※ 2

Java クラスライブラリを使用したアプリケーションを移設する場合，コンフィグレーションキーの設定値は移設前の環境の動作環境定義ファイルと同じ値を指定してください。

注※ 3

クライアント共用トレースファイル
「DBJComTrace プロセス ID（0x で始まる 16 進数）_ トレースファイル番号 .log」

(4) ファイルの内容の有効期間

動作環境定義ファイルの内容は，Java クラスライブラリを初期化するときに読み込まれます。一度読み込まれたあとは，プロセスが終了するまで再読み込みはされません。

8 障害対策

この章では，Java クラスライブラリで障害が起きた場合に原因を追求して対処するために使用するトレース情報とメッセージ情報について説明します。

8.1 トレース情報

この節では，Java クラスライブラリが出力するトレース情報について説明します。

8.1.1 トレース情報の出力先

Java クラスライブラリのトレース情報は，**クライアント共用トレースファイル**に出力されます。このファイルには，C++ クラスライブラリのトレース情報が合わせて出力されます。

なお，トレースファイルは次のファイル名で出力されます。

出力ファイル名

DBJComTrace0x"PID"_"NO".log

注意：「PID」はプロセス ID（16 進数）です。「NO」は通番です。

クライアント共用トレースファイルの出力先は，動作環境定義ファイルで設定できます。設定方法については，「7.6.3　動作環境定義ファイル」を参照してください。

8.1.2 トレース情報の出力内容

クライアント共用トレースファイルには，次の内容が出力されます。

- Java クラスライブラリのメソッドの開始と終了※
- 発生したすべてのエラーメッセージ
- キャッチした例外
- ログインメソッド（DbjSession#login メソッド）で指定したユーザ識別子

注※

パラメタクラスのメソッドを除きます。

8.2　メッセージ情報

この節では，Java クラスライブラリが出力するメッセージ情報について説明します。出力されるメッセージの詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」を参照してください。なお，DocumentBroker サーバの運用中に出力されたメッセージ情報については，マニュアル「DocumentBroker Version 3　メッセージ」を参照してください。

8.2.1　メッセージ情報の出力先

Java クラスライブラリのメソッド実行時に発生したエラー，警告およびインフォメーションを表すメッセージは，次のファイルに出力されます。

- クライアント共用トレースファイル※

注※
　エラーメッセージだけが出力されます。

8.2.2　メッセージを出力する言語

ここでは，メッセージを出力する言語について説明します。DocumentBroker のメッセージは，日本語または英語で出力できます。出力する言語についての設定方法は，OS ごとに異なります。

UNIX の場合は，環境変数「LANG」に設定します。環境変数の設定については，「7.4.2　環境変数の設定」を参照してください。ただし，syslog ファイルには，環境変数の設定に関係なく，常に英語のメッセージが出力されます。

Windows の場合は，「コントロールパネル」の「地域と言語のオプション」で設定します。

付録

付録 A　文書空間オブジェクトのプロパティ一覧

ここでは，Java クラスライブラリで扱うことができる文書空間オブジェクトのプロパティについて説明します。文書空間オブジェクトのプロパティには，文書空間オブジェクトクラスごとに定義されたプロパティと，DMA クラスごとに定義されたプロパティがあります。

付録 A.1　文書空間オブジェクトクラスのプロパティ一覧

ここでは，文書空間オブジェクトクラスごとに設定されているプロパティ（dbrProp_ で始まるプロパティ名を持つプロパティ）の一覧を示します。

次に示す項目を説明しています。

プロパティ一覧

文書空間オブジェクトクラスが保持するプロパティの特性を次のような一覧表で示しています。

プロパティ名	データ型	編集可能
dbrProp_X	STR 型	○

プロパティの一覧表の各項目について説明します。

プロパティ名

プロパティの名前です。

データ型

プロパティ値のデータ型について記述しています。

編集可能

編集できるプロパティかまたは読み取り専用のプロパティかを記述しています。

○：プロパティの値を更新するメソッドなどで編集できるプロパティであることを示します。

－：読み取り専用のプロパティであることを示します。

(1) バージョンなし文書クラスのプロパティ

バージョンなし文書クラスのプロパティの一覧を次の表に示します。

表 A-1　バージョンなし文書クラスのプロパティ

プロパティ名	データ型	編集可能
dbrProp_ContainersCount	INT 型	－
dbrProp_ContainersCountVT	INT 型	－
dbrProp_HeadRelationsCount	INT 型	－
dbrProp_ParentCount	INT 型	－
dbrProp_RenditionStatus	INT 型	－
dbrProp_RenditionType	STR 型	－
dbrProp_RetrievalName	STR 型	○
dbrProp_TailRelationsCount	INT 型	－

アクセス制御機能を使用している場合は，表 A-1 のプロパティに加えて，アクセス制御情報を表すプロパティも定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラス

ライブラリ固有のプロパティ」を参照してください。

(2) バージョン付き文書クラスのプロパティ

バージョニングオブジェクトのプロパティの一覧を次の表に示します。

表 A-2　バージョン付き文書クラスのプロパティ（バージョニングオブジェクトのプロパティ）

プロパティ名	データ型	編集可能
dbrProp_ContainersCount	INT 型	−
dbrProp_ContainersCountVT	INT 型	−
dbrProp_CurrentVersion	STR 型	−
dbrProp_HeadRelationsCount	INT 型	−
dbrProp_ParentCount	INT 型	−
dbrProp_RenditionType	STR 型	−
dbrProp_RetrievalName	STR 型	○
dbrProp_TailRelationsCount	INT 型	−
dbrProp_VersionsCount	INT 型	−

アクセス制御機能を使用している場合は，表 A-2 のプロパティに加えて，アクセス制御情報を表すプロパティも定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ」を参照してください。

バージョンオブジェクトのプロパティの一覧を次の表に示します。

表 A-3　バージョン付き文書クラスのプロパティ（バージョンオブジェクトのプロパティ）

プロパティ名	データ型	編集可能
dbrProp_ContainersCount	INT 型	−
dbrProp_ContainersCountVT	INT 型	−
dbrProp_HeadRelationsCount	INT 型	−
dbrProp_ParentCount	INT 型	−
dbrProp_RenditionStatus	INT 型	−
dbrProp_RenditionType	STR 型	−
dbrProp_RetrievalName	STR 型	○
dbrProp_TailRelationsCount	INT 型	−

アクセス制御機能を使用している場合は，表 A-3 のプロパティに加えて，アクセス制御情報を表すプロパティも定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ」を参照してください。

(3) バージョンなしフォルダクラスのプロパティ

バージョンなしフォルダのプロパティを次の表に示します。

表 A-4　バージョンなしフォルダクラスのプロパティ

プロパティ名	データ型	編集可能
dbrProp_ChildrenCount	INT 型	－
dbrProp_ContaineesCount	INT 型	－
dbrProp_ContaineesCountVT	INT 型	－
dbrProp_ContainersCount	INT 型	－
dbrProp_ContainersCountVT	INT 型	－
dbrProp_ParentCount	INT 型	－

アクセス制御機能を使用している場合は，表 A-4 のプロパティに加えて，アクセス制御情報を表すプロパティも定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ」を参照してください。

また，トップオブジェクトクラスが，dmaClass_Container クラスなど，edmClass_ContainerVersion クラスまたはそのサブクラス以外の場合の，バージョンなしフォルダクラスのプロパティの一覧を次の表に示します。

表 A-5　バージョンなしフォルダクラスのプロパティ（トップオブジェクトクラスが edmClass_ContainerVersion クラスまたはそのサブクラス以外の場合）

プロパティ名	データ型	編集可能
dbrProp_ChildrenCount	INT 型	－
dbrProp_ContainersCount	INT 型	－
dbrProp_ContaineesCount	INT 型	－
dbrProp_ParentCount	INT 型	－

アクセス制御機能を使用している場合は，表 A-5 のプロパティに加えて，アクセス制御情報を表すプロパティも定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ」を参照してください。

(4) バージョン付きフォルダクラスのプロパティ

バージョニングオブジェクトのプロパティの一覧を次の表に示します。

表 A-6　バージョン付きフォルダクラスのプロパティ（バージョニングオブジェクトのプロパティ）

プロパティ名	データ型	編集可能
dbrProp_ChildrenCount	INT 型	－
dbrProp_ContaineesCount	INT 型	－
dbrProp_ContaineesCountVT	INT 型	－
dbrProp_ContainersCount	INT 型	－
dbrProp_ContainersCountVT	INT 型	－
dbrProp_CurrentVersion	STR 型	－
dbrProp_ParentCount	INT 型	－
dbrProp_VersionsCount	INT 型	－

アクセス制御機能を使用している場合は，表 A-6 のプロパティに加えて，アクセス制御情報を表すプロパティも定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ」を参照してください。

バージョンオブジェクトのプロパティの一覧を次の表に示します。

表 A-7　バージョン付きフォルダクラスのプロパティ（バージョンオブジェクトのプロパティ）

プロパティ名	データ型	編集可能
dbrProp_ChildrenCount	INT 型	－
dbrProp_ContaineesCount	INT 型	－
dbrProp_ContaineesCountVT	INT 型	－
dbrProp_ContainersCount	INT 型	－
dbrProp_ContainersCountVT	INT 型	－
dbrProp_ParentCount	INT 型	－

アクセス制御機能を使用している場合は，表 A-7 のプロパティに加えて，アクセス制御情報を表すプロパティも定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ」を参照してください。

(5) 独立データクラスのプロパティ

アクセス制御機能を使用していない場合，独立データクラスのプロパティはありません。

アクセス制御機能を使用している場合は，アクセス制御情報を表すプロパティが定義されます。詳細については，「(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ」を参照してください。

(6) パブリック ACL クラスのプロパティ

パブリック ACL クラスのプロパティの一覧を次の表に示します。

表 A-8　パブリック ACL クラスのプロパティ

プロパティ名	データ型	編集可能
dbrProp_ACL	VARRAY 型	○※1
dbrProp_BindObjectCount	INT 型	－
dbrProp_OwnerId	STR 型	○※2
dbrProp_SACL	VARRAY 型	○※2
dbrProp_UserPermission	INT 型	－

注※1　編集は，そのオブジェクトの所有者，セキュリティ ACL でアクセス制御情報変更権を与えられたユーザおよびセキュリティ管理者だけが実行できます。
注※2　編集は，そのオブジェクトの所有者およびセキュリティ管理者だけが実行できます。

(7) アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ

アクセス制御機能を使用する場合に，パブリック ACL 以外のすべての文書空間オブジェクトに追加されるプロパティの一覧を次の表に示します。

表 A-9　アクセス制御機能を使用する場合に追加される Java クラスライブラリ固有のプロパティ

プロパティ名	データ型	編集可能
dbrProp_ACL	VARRAY 型	○※1
dbrProp_EveryonePermission	INT 型	○※1
dbrProp_OwnerId	STR 型	○※2
dbrProp_OwnerPermission	INT 型	○※1
dbrProp_PrimaryGroupId	STR 型	○※1
dbrProp_PrimaryGroupPermission	INT 型	○※1
dbrProp_PublicACLCount	INT 型	－
dbrProp_PublicACLIds	VARRAY 型	○※1
dbrProp_SACL	VARRAY 型	○※2
dbrProp_UserPermission	INT 型	－

注※1　編集は，そのオブジェクトの所有者，セキュリティ ACL でアクセス制御情報変更権を与えられたユーザおよびセキュリティ管理者だけが実行できます。
注※2　編集は，そのオブジェクトの所有者およびセキュリティ管理者だけが実行できます。

付録 A.2　DMA のクラスのプロパティ一覧

ここでは，プロパティの定義元になる DMA のクラスのプロパティのうち，Java クラスライブラリで使用できる主なプロパティの一覧を示します。

クラスごとに，次に示す項目を説明しています。

対応する文書空間オブジェクトクラス
説明する DMA のクラスと対応する文書空間オブジェクトクラスについて説明しています。

クラスの説明
クラスの特徴と機能，注意事項などについて説明しています。

スーパークラス
クラスの直上のスーパークラスを記述しています。

プロパティ一覧
プロパティの一覧表の各項目について説明します。

プロパティ名
プロパティの名前です。

データ型
プロパティ値のデータ型について記述しています。
なお，「オブジェクトリファレンス」は，DMA オブジェクトへのリファレンスを表すデータ型のプロパティです。このプロパティは，edmSQL 文の中だけで使用できます。Java クラスライブラリのメソッドで扱うことはできません。

なお，DMA クラスに定義された dmaProp_ または edmProp_ で始まるプロパティには，Java クラスライブラリのメソッドで直接編集するプロパティはありません。

(1) dmaClass_ConfigurationHistory クラスのプロパティ

dmaClass_ConfigurationHistory クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

- バージョン付きフォルダクラス
- バージョン付き文書クラス

クラスの説明

dmaClass_ConfigurationHistory クラスは，バージョン管理する文書のバージョン構成を保持するための DMA クラスです。文書をバージョン管理するときの最上位に位置します。

スーパークラス

dmaClass_Containable クラス

dmaClass_ConfigurationHistory クラスのプロパティ一覧を，次の表に示します。

表 A-10　dmaClass_ConfigurationHistory クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Parent	オブジェクトリファレンス
dmaProp_ParentContainer	オブジェクトリファレンス
dmaProp_VersionedDmaObjectClass	オブジェクトリファレンス
dmaProp_PrimaryVersionSeries	オブジェクトリファレンス

(2) dmaClass_Container クラスのプロパティ

dmaClass_Container クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

バージョンなしフォルダクラス

クラスの説明

dmaClass_Container クラスは，ほかのオブジェクトを DirectContainmentRelationship オブジェクトまたは ReferentialContainmentRelationship オブジェクトを介して包含するための DMA クラスです。Container オブジェクトは直接コンテンツを持ちません。

スーパークラス

dmaClass_Containable クラス

dmaClass_Container クラスのプロパティ一覧を，次の表に示します。

表 A-11　dmaClass_Container クラスのプロパティ一覧

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Parent	オブジェクトリファレンス

プロパティ名	データ型
dmaProp_ParentContainer	オブジェクトリファレンス

(3) dmaClass_DirectContainmentRelationship クラスのプロパティ

dmaClass_DirectContainmentRelationship クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

リンクオブジェクトクラス

クラスの説明

dmaClass_DirectContainmentRelationship クラスは，一つの文書空間内でのオブジェクト間の直接型リンクを規定する DMA クラスで，dmaClass_ContainmentRelationship クラスのサブクラスです。

直接型リンクでは，Parent 対 Child は 1:n の関係にあるため，包含される Child に相当するオブジェクトは，dmaClass_DirectContainmentRelationship クラスを介さないで，dmaProp_ParentContainer プロパティから，自身を包含する Container オブジェクトを探索できます。

DocumentBroker では，この DMA クラスは検索できません。

スーパークラス

dmaClass_ContainmentRelationship クラス

dmaClass_DirectContainmentRelationship クラスのプロパティ一覧を，次の表に示します。

表 A-12　dmaClass_DirectContainmentRelationship クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Head	オブジェクトリファレンス
dmaProp_Tail	オブジェクトリファレンス

(4) dmaClass_DocVersion クラスのプロパティ

dmaClass_DocVersion クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

バージョンなし文書クラス

クラスの説明

dmaClass_DocVersion クラスは，文書を表現するオブジェクトのルートとなる DMA クラスです。dmaClass_Versionable クラスおよび dmaClass_Containable クラスのプロパティを継承します。したがって，文書空間では，dmaClass_DocVersion クラスまたはそのサブクラスのインスタンスをバージョン管理したり，ほかの文書と関連付けて管理したりできます。

スーパークラス

dmaClass_Versionable クラス

dmaClass_DocVersion クラスのプロパティ一覧を，次の表に示します。

表 A-13　dmaClass_DocVersion クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Parent	オブジェクトリファレンス
dmaProp_ParentContainer	オブジェクトリファレンス
dmaProp_CurrentOfSeriesCount	INT 型
edmProp_RenditionsCount	INT 型

(5) dmaClass_ReferentialContainmentRelationship クラスのプロパティ

dmaClass_ReferentialContainmentRelationship クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

リンクオブジェクトクラス

クラスの説明

dmaClass_ReferentialContainmentRelationship クラスは，文書空間内でのオブジェクト間の参照型リンクを規定する DMA クラスで，dmaClass_ContainmentRelationship クラスのサブクラスです。DocumentBroker では，このクラスは検索できません。

スーパークラス

dmaClass_ContainmentRelationship クラス

dmaClass_ReferentialContainmentRelationship クラスのプロパティ一覧を，次の表に示します。

表 A-14　dmaClass_ReferentialContainmentRelationship クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Head	オブジェクトリファレンス
dmaProp_Tail	オブジェクトリファレンス

(6) edmClass_ContainerVersion クラスのプロパティ

edmClass_ContainerVersion クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

バージョン付きフォルダクラス（バージョン付きフォルダのバージョン）

クラスの説明

edmClass_ContainerVersion クラスは，バージョン付きフォルダの機能を持ち，格納しているオブジェクトの集合全体としてのバージョンを管理するための DMA クラスです。また，edmProp_VTContainers プロパティを保持し，ほかのバージョン付きフォルダの機能を持つオブジェクトの構成要素にもなれます。

スーパークラス

dmaClass_Versionable クラス

edmClass_ContainerVersion クラスのプロパティ一覧を，次の表に示します。

表 A-15　edmClass_ContainerVersion クラス

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Parent	オブジェクトリファレンス
dmaProp_ParentContainer	オブジェクトリファレンス
dmaProp_CurrentOfSeriesCount	INT 型

（7）edmClass_ContentSearch クラスのプロパティ

edmClass_ContentSearch クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

なし。

クラスの説明

edmClass_ContentSearch クラスは，dmaClass_DocVersion クラスのサブクラスに仮想的に継承させて全文検索の対象にする文書を登録するための DMA クラスです。edmClass_ContentSearch クラスのプロパティを仮想的に継承した dmaClass_DocVersion クラスのサブクラスでは，edmClass_ContentSearch クラスの選択可能なプロパティを検索に使用するプロパティとして指定できます。また，検索可能なプロパティである edmProp_TextIndex プロパティ，edmProp_StIndex プロパティ，edmProp_ConceptTextIndex プロパティおよび edmProp_ConceptStIndex プロパティを指定して全文検索ができます。

スーパークラス

dmaClass_DMA クラス

edmClass_ContentSearch クラスのプロパティ一覧を，次の表に示します。

表 A-16　edmClass_ContentSearch クラスのプロパティの一覧

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
edmProp_TextIndex	STR 型
edmProp_StIndex	STR 型
edmProp_ConceptTextIndex	STR 型
edmProp_ConceptStIndex	STR 型
edmProp_Score	INT 型
edmProp_RawScore	INT 型
edmProp_DocLength	INT 型

プロパティ名	データ型
edmProp_ContentIndexStatus	INT 型

(8) edmClass_IndependentPersistence クラスのプロパティ

edmClass_IndependentPersistence クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

独立データクラス

クラスの説明

edmClass_IndependentPersistence クラスは，独立して存在するオブジェクトを新たに作成するためのクラスです。DocumentBroker では，このクラスを継承しているサブクラスを基にして作成したオブジェクトを DMA のオブジェクトとして扱います。

DocumentBroker で，この DMA クラスを基にサブクラスを定義する場合は，次の規則に従ってください。

クラス，プロパティの定義に関する規則

- クラスまたはプロパティの GUID は，DocumentBroker が予約している値以外の値で定義します。なお，複数のクラスに同じ GUID を定義することはできません。
- 複数のプロパティに同じ GUID を定義する場合は，プロパティ名およびプロパティの型を同じにします。
- edmClass_IndependentPersistence クラスで定義しているプロパティは，すべて継承します。削除はできません。

プロパティのデータ型に関する規則

定義できるプロパティのデータ型は次のとおりです。

BOOL 型

INT 型

STR 型

VARRAY 型

プロパティ値の設定および取得に関する規則

DocumentBroker またはユーザアプリケーションプログラムが定義する GUID かインデクスを使用して，プロパティ値の設定または取得ができます。

スーパークラス

dmaClass_DMA クラス

edmClass_IndependentPersistence クラスのプロパティ一覧を，次の表に示します。

表 A-17　edmClass_IndependentPersistence クラスのプロパティの一覧

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス

(9) edmClass_PublicACL クラスのプロパティ

edmClass_PublicACL クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

パブリック ACL クラス

クラスの説明

edmClass_PublicACL クラスは，パブリック ACL を表すオブジェクトの構成要素になるクラスです。

スーパークラス

dmaClass_DMA クラス

edmClass_PublicACL クラスのプロパティ一覧を，次の表に示します。

表 A-18　edmClass_PublicACL クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_This	オブジェクトリファレンス
edmProp_OwnerId	STR 型

（10）edmClass_Relationship クラスのプロパティ

edmClass_Relationship クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

リンクオブジェクトクラス

クラスの説明

edmClass_Relationship クラスは，文書間リンクによってオブジェクト間の関連付けを設定するための DMA クラスで，dmaClass_Relationship クラスのサブクラスです。

スーパークラス

dmaClass_Relationship クラス

edmClass_Relationship クラスのプロパティ一覧を，次の表に示します。

表 A-19　edmClass_Relationship クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Head	オブジェクトリファレンス
dmaProp_Tail	オブジェクトリファレンス
edmProp_RelationType	INT 型

（11）edmClass_VersionTraceableContainmentRelationship クラス

edmClass_VersionTraceableContainmentRelationship クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

リンクオブジェクトクラス

クラスの説明

edmClass_VersionTraceableContainmentRelationship クラスは，文書空間内でのオブジェクト間の構成管理型リンクを規定する DMA クラスで，dmaClass_ContainmentRelationship クラスのサブクラスです。フォルダの構成要素の最新バージョンを追跡してリンクを規定したり，任意のバージョンを固定してリンクを規定したりできるプロパティを保持します。
DocumentBroker では，この DMA クラスは検索できません。

スーパークラス

dmaClass_ContainmentRelationship クラス

edmClass_VersionTraceableContainmentRelationship クラスのプロパティ一覧を，次の表に示します。

表 A-20　edmClass_VersionTraceableContainmentRelationship クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Head	オブジェクトリファレンス
dmaProp_Tail	オブジェクトリファレンス
edmProp_VTMode	INT 型
edmProp_VTVersionSeries	オブジェクトリファレンス
edmProp_VTConfigurationHistory	オブジェクトリファレンス

(12) edmClass_VersionTracedDocVersion クラスのプロパティ

edmClass_VersionTracedDocVersion クラスのプロパティについて説明します。

対応する文書空間オブジェクトクラス

バージョンなし文書クラス

クラスの説明

edmClass_VersionTracedDocVersion クラスは，構成管理型リンクで包含される文書を表現するための DMA クラスで，dmaClass_DocVersion クラスのサブクラスです。構成管理型リンクで包含されるための edmProp_VTContainers プロパティを保持します。

スーパークラス

dmaClass_DocVersion クラス

edmClass_VersionTracedDocVersion クラスのプロパティ一覧を，次の表に示します。

表 A-21　edmClass_VersionTracedDocVersion クラスのプロパティ

プロパティ名	データ型
dmaProp_OIID	STR 型
dmaProp_ClassDescription	オブジェクトリファレンス
dmaProp_This	オブジェクトリファレンス
dmaProp_Parent	オブジェクトリファレンス
dmaProp_ParentContainer	オブジェクトリファレンス
dmaProp_CurrentOfSeriesCount	INT 型

プロパティ名	データ型
edmProp_RenditionsCount	INT 型

付録 A.3　各プロパティの説明

ここでは，それぞれのプロパティに設定される値について説明します。

（1） dbrProp_ACL プロパティ

ACL（アクセス制御リスト）を表す ACE（アクセス制御エレメント）のリストです。

（2） dbrProp_BindObjectCount プロパティ

このプロパティを持つパブリック ACL をバインドしているオブジェクトの数です。

（3） dbrProp_ChildrenCount プロパティ

直接型リンクで包含しているオブジェクトの個数です。

（4） dbrProp_ContaineesCount プロパティ

参照型リンクで包含しているオブジェクトの個数です。

（5） dbrProp_ContaineesCountVT プロパティ

構成管理型リンクで包含しているオブジェクトの個数です。

（6） dbrProp_ContainersCount プロパティ

参照型リンクでオブジェクトを包含しているフォルダの個数です。

（7） dbrProp_ContainersCountVT プロパティ

構成管理型リンクでオブジェクトを包含しているフォルダの個数です。

（8） dbrProp_CurrentVersion プロパティ

カレントバージョンのバージョン識別子です。

（9） dbrProp_EveryonePermission プロパティ

すべてのユーザに与えられるパーミッションを表す定数です。

（10） dbrProp_HeadRelationsCount プロパティ

文書間リンクでリンク付けているリンク先文書の個数です。

（11） dbrProp_OwnerId プロパティ

オブジェクトの所有者のユーザ識別子です。

（12） dbrProp_OwnerPermission プロパティ

オブジェクトの所有者に与えられるパーミッションを表す定数です。

（13）dbrProp_ParentCount プロパティ

直接型リンクでオブジェクトを包含しているフォルダの個数です。

（14）dbrProp_PrimaryGroupId プロパティ

プライマリグループのグループ識別子です。

（15）dbrProp_PrimaryGroupPermission プロパティ

プライマリグループに与えられるパーミッションを表す定数です。

（16）dbrProp_PublicACLCount プロパティ

バインドしているパブリック ACL の数です。

（17）dbrProp_PublicACLIds プロパティ

バインドしているパブリック ACL の OIID のリストです。

（18）dbrProp_RenditionStatus プロパティ

レンディションの状態を表す 16 進数です。

（19）dbrProp_RenditionType プロパティ

レンディションタイプを表す文字列です。

（20）dbrProp_RetrievalName プロパティ

登録されているファイルのファイル名を表す文字列です。

（21）dbrProp_TailRelationsCount プロパティ

文書間リンクでリンク付けられているリンク元文書の個数です。

（22）dbrProp_SACL プロパティ

セキュリティ ACL を表す ACE（アクセス制御エレメント）のリストです。

（23）dbrProp_VersionsCount プロパティ

オブジェクトが管理しているバージョンの個数です。

（24）dbrProp_UserPermission プロパティ

オブジェクトに設定されるログインユーザまたはログインユーザが所属するグループに与えられているパーミッションの論理和です。

（25）dmaProp_ClassDescription プロパティ

DMA オブジェクトの基になっているクラスについて定義する DMA オブジェクトである，ClassDescription オブジェクトへのオブジェクトリファレンスです。

（26）dmaProp_CurrentOfSeriesCount プロパティ

最新バージョンとしてチェックインしたバージョン付き文書のバージョンオブジェクト（VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクト）に対して与えられる番号で

す。

Java クラスライブラリでは，そのオブジェクトのバージョンの状態を明確にするために，次の値が設定されます。

表 A-22　Java クラスライブラリで設定される dmaProp_CurrentOfSeriesCount プロパティの値

VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクトの状態		dmaProp_CurrentOfSeriesCount プロパティに設定される値
バージョン付き文書のカレントバージョンに対応する VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクト		1
チェックアウト中の VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクト		-2
チェックイン後の VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクト	チェックインによって追加された VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクト	1
	チェックインによって旧バージョンになった VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクト	0
バージョン管理の対象でないバージョンなし文書に包含される VersionTracedDocVersion オブジェクトまたは DocVersion オブジェクト		-1

(27) dmaProp_Head プロパティ

関連付けられた二つのオブジェクトのうち，Head（リンク先）側のオブジェクトへのオブジェクトリファレンスです。例えば，リンクでは，包含される側のオブジェクトです。

このプロパティは，リンクのプロパティです。Java クラスライブラリでは，リンクが設定されているオブジェクトの dmaProp_This プロパティとこのプロパティを組み合わせて，検索実行時に検索対象クラスを結合する場合に利用できます。

(28) dmaProp_OIID プロパティ

オブジェクトを識別する識別子です。

(29) dmaProp_Parent プロパティ

このオブジェクトを直接型リンクで包含している Parent に相当する DMA オブジェクトを識別する DirectContainmentRelationship オブジェクトへのオブジェクトリファレンスです。

(30) dmaProp_ParentContainer プロパティ

DirectContainmentRelationship オブジェクトを介してこの DMA オブジェクトを包含している Container オブジェクトへのオブジェクトリファレンスです。

クライアントは，この dmaProp_ParentContainer プロパティによって，介在する DirectContainmentRelationship オブジェクトを参照することなく，直接 Parent に相当する Container オブジェクトを探索できます。

このプロパティを使用して探索を実行できます。

(31) dmaProp_PrimaryVersionSeries プロパティ

バージョン管理する文書に対する最初の VersionSeries オブジェクトへのオブジェクトリファレンスです。

ConfigurationHistory オブジェクトに，最初に VersionSeries オブジェクトがバインドされるときに値が設定されます。なお，VersionSeries オブジェクトとは，バージョン管理で使用する DMA オブジェクトです。

(32) dmaProp_RenditionType プロパティ

文書の表現形式を表す文字列です。

文字列の形式は次のようになります。

```
<rendition_type_space>::<typename>
```

DocumentBroker では，rendition_type_space に IANA で定義している MIME を使用します。表現形式がプレーンテキストの場合，次のように指定します。

```
MIME::text/plain
```

Java クラスライブラリでは，次の文書空間オブジェクトクラスのオブジェクトの dbrProp_RenditionType プロパティとして操作します。

- バージョンなし文書クラス
- バージョン付き文書クラス

(33) dmaProp_Tail プロパティ

関連付けられた二つのオブジェクトのうち，Tail（リンク元）側のオブジェクトへのオブジェクトリファレンスです。例えば，リンクでは，包含する側のオブジェクトです。

このプロパティは，リンクのプロパティです。Java クラスライブラリでは，リンクが設定されているオブジェクトの dmaProp_This プロパティとこのプロパティを組み合わせて，検索実行時に検索対象クラスを結合する場合に利用できます。

(34) dmaProp_This プロパティ

オブジェクト自身を表すオブジェクトリファレンスです。

問い合わせの中で，オブジェクト間の関連を表現して，問い合わせの結果の中から候補のオブジェクトを選択するために使用できます。

Java クラスライブラリでは，リンクの dmaProp_Head プロパティや dmaProp_Tail プロパティとこのプロパティを組み合わせて，検索実行時に検索対象クラスを結合する場合などに利用します。

(35) dmaProp_VersionedDmaObjectClass プロパティ

バージョン管理する文書のベースとなるクラスについて定義する ClassDescription オブジェクトへのオブジェクトリファレンスです。

バージョン管理する文書は，すべてこのプロパティの ClassDescription オブジェクトが定義するクラスを基に作成される必要があります。

DocumentBroker では，このプロパティに設定された ClassDescription オブジェクトで，バージョン管理

する文書のオブジェクトであるかどうかを確認しません。このプロパティの値は，あらかじめ DocumentBroker サーバのメタ情報ファイル（edms.ini）に設定してください。DocumentBroker サーバのメタ情報ファイルについては，マニュアル「DocumentBroker Version 3　システム導入・運用ガイド」を参照してください。

（36）edmProp_ConceptStIndex プロパティ

全文検索用に登録した文書の全文データです。

このプロパティは，全文検索エンジンである HiRDB Text Search Plug-in に対して，全文検索用の登録情報を参照することを示すために使用されるプロパティです。したがって，SELECT 句に検索結果として取得するプロパティには指定できません。

このプロパティでは，概念検索および構造指定検索を含む全文検索が実行できます。

（37）edmProp_ConceptTextIndex プロパティ

全文検索用に登録した文書の全文データです。

このプロパティは，全文検索エンジンである HiRDB Text Search Plug-in に対して，全文検索用の登録情報を参照することを示すために使用されるプロパティです。したがって，SELECT 句に検索結果として取得するプロパティには指定できません。

このプロパティでは，概念検索を含む全文検索が実行できます。

（38）edmProp_ContentIndexStatus プロパティ

全文検索インデクスの登録状態を示す値です。次の表に示す値が設定されます。

表 A-23　edmProp_ContentIndexStatus プロパティに設定される値

値	意　味
0	文書は未登録です。
1	文書は登録されていますが，全文検索インデクスは未登録です。
2	文書および全文検索インデクスが登録されています。
3	文書が更新されていますが，全文検索インデクスが更新されていないため，文書と全文検索インデクスの内容が一致しません。
100 以降	DocumentBroker Text Search Index Loader Version 3 が使用する値が設定されます。

（39）edmProp_DocLength プロパティ

検索した文書の長さです。バイト単位で示します。

（40）edmProp_OwnerId プロパティ

オブジェクトの所有者を表す文字列です。

Java クラスライブラリでは，dbrProp_OwnerId プロパティとして扱います。ただし，検索条件に指定する場合は，「edmProp_OwnerId」と指定します。

（41）edmProp_RawScore プロパティ

ランキング検索した場合のスコアです。

検索するときに算出されて値が設定されます。ランキング検索時のスコアを取得するための形式的なプロ

パティです。スコアは，検索結果集合内で正規化されないで返されます。

(42) edmProp_RelationType プロパティ

Relationship オブジェクトを使用した関連の種類を定義する値です。0 以下の値は設定できません。

(43) edmProp_RenditionsCount プロパティ

一つの文書中に登録されているレンディションの数を表すカウンタです。マスタレンディションだけが登録されている場合，値は常に 1 になります。サブレンディションが登録されると，マスタレンディションと登録されたサブレンディションの数を合わせた値になります。

(44) edmProp_Score プロパティ

ランキング検索した場合のスコアです。

検索するときに算出されて値が設定されます。ランキング検索時のスコアを取得するための形式的なプロパティです。スコアは，検索結果集合内で正規化されて返されます。

(45) edmProp_StIndex プロパティ

全文検索用に登録した文書の全文データです。

このプロパティは，全文検索エンジンである HiRDB Text Search Plug-in に対して，全文検索用の登録情報を参照することを示すために使用されるプロパティです。したがって，SELECT 句に検索結果として取得するプロパティには指定できません。

このプロパティでは，構造指定検索を含む全文検索が実行できます。

(46) edmProp_TextIndex プロパティ

全文検索用に登録した文書の全文データです。

このプロパティは，全文検索エンジンである HiRDB Text Search Plug-in に対して，全文検索用の登録情報を参照することを示すために使用されるプロパティです。したがって，SELECT 句に検索結果として取得するプロパティには指定できません。

このプロパティでは，プレーンテキストに対する全文検索が実行できます。

(47) edmProp_VTConfigurationHistory プロパティ

VersionTraceableContainmentRelationship オブジェクトが指している包含される側のオブジェクトのバージョンを統合している ConfigurationHistory オブジェクトへのオブジェクトリファレンスです。

(48) edmProp_VTMode プロパティ

包含するオブジェクトの最新バージョンを追跡するか（FLOATING モード），バージョンを固定するか（FIX モード）を示す値です。

(49) edmProp_VTVersionSeries プロパティ

VersionTraceableContainmentRelationship オブジェクトが指している包含される側のオブジェクトのバージョンを管理している VersionSeries オブジェクトへのオブジェクトリファレンスです。なお，VersionSeries オブジェクトとは，バージョン管理で使用する DMA オブジェクトです。

付録 B　用語解説

Java 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 Java クラスライブラリの用語については，マニュアル「DocumentBroker Version 3　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。
Java 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。

ContainerVersion オブジェクト

DMA クラスの edmClass_ContainerVersion クラスまたはそのサブクラスを基に作成された DMA オブジェクトです。

Container オブジェクト

DMA 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 DMA オブジェクトです。

CORBA（Common Object Request Broker Architecture）

OMG（Object Management Group）が提唱するオブジェクト間の通信メカニズムを提供する ORB（Object Request Broker）の標準アーキテクチャです。

DMA（Document Management Alliance）

文書管理インターフェースの標準化を図る団体 AIIM（Association for Information and Image Management International）によって定義される共通インターフェースです。

dmaClass_ConfigurationHistory クラス

バージョン管理に使用する DMA 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 DMA 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 DMA 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。

DMA オブジェクト

DMA クラスを基に作成されたオブジェクトです。

DMA クラス

DMA 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。
Java クラスライブラリの文書空間オブジェクトクラスのプロパティは，DMA クラスに定義します。

DocVersion オブジェクト

DMA クラスの dmaClass_DocVersion クラス，または edmClass_VersionTracedDocVersion クラス以外の dmaClass_DocVersion クラスのサブクラスを基に作成された DMA 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 DMA クラスです。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 DMA クラスです。また，構成管理型リンクを利用して構成要素であるバージョン付き文書の特定バージョンを固定してリンクしたり，常に最新バージョンをトレースしてリンクしたりもできます。
edmClass_ContainerVersion クラスまたはそのサブクラスは，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。また，バージョン付きフォルダ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。

edmClass_IndependentPersistence クラス

独立データを表す DMA 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 DMA 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_Relationship クラス

文書間リンクを表す DMA クラスです。単独では作成できない，文書に従属する Relationship オブジェクトの基となる DMA クラスです。

edmClass_VersionTracedDocVersion クラス

文書を表す DMA 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

文書空間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。DMA 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。

Java クラスライブラリ

DocumentBroker サーバ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 Development Kit および DocumentBroker Runtime が提供しているインターフェースです。

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』である文書空間オブジェクトか，作成者の所属が『日立製作所』である文書空間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

文書空間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。Java クラスライブラリでは，文書空間オブジェクトを操作するとき，文書空間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

文書空間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

DMA 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 DMA オブジェクトです。

W3C（World Wide Web Consortium）

HTML や XML など，WWW に関する技術の標準化を推進する非営利団体です。

XML インデクスデータ作成機能

XML 文書のコンテンツからインデクスデータを作成して，XML 文書のプレーンテキスト形式の全文検索インデクスまたは構造指定検索用の全文検索インデクスを登録する機能です。XML 文書の構文解析やタグ情報の除去もあわせて実行できます。

XML 構造指定検索

XML 文書の DTD で記述した論理構造（エレメント）やエレメントの属性をキーにして，XML 文書を全文検索する方法です。例えば，「タイトルに『コンピュータ』という単語が含まれ，章に『XML』という単語が含まれる文書を検索する」というような場合に使用します。また，「『document』というエレメントに設定されている属性『status』の属性値が『public』である文書を検索する」というような場合にも使用できます。

XML トランスレーターオブジェクト

DbjXmlTranslator インターフェースを取得したときにメモリ空間に作成される概念的なオブジェクトです。DbjXmlTranslator インターフェースの機能を実行できます。

XML プロパティマッピング機能

XML 文書を構文解析して，DocumentBroker サーバのプロパティマッピング定義ファイルの定義に従って XML 文書内のタグ間の文字列または属性値を XML 文書のプロパティに割り当てて設定する機能です。

XML プロパティマッピング結果情報

XML プロパティマッピング機能で作成される情報です。DbjMappedProp インターフェースで扱います。

XML 文書のアップロード情報

XML 文書作成またはコンテンツ更新時に使用する情報です。XML 文書作成時に，XML プロパティマッピング機能および XML インデクスデータ作成機能を実行するために使用できます。また，XML 文書のコンテンツ更新時に XML インデクスデータ作成機能を実行するために使用できます。DbjXmlUploadInfo インターフェースで扱います。

（ア行）

アクセス権

文書空間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空間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空間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空間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

文書空間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空間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）

文書空間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

Java 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。Java 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空間オブジェクトのうち，リンク元である文書空間オブジェクトの 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，文書空間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

文書空間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，文書空間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

文書空間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，バージョン付きフォルダ，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空間オブジェクトを，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空間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

DMA 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。Java 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。Java 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またはバージョン付きフォルダ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，文書空間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空間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空間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，文書空間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，文書空間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。

クラス定義情報ファイル

DocumentBroker サーバで定義されている DMA オブジェクトのクラスまたはそのサブクラスのクラス名を取得したり，プロパティ名から GUID，データ型などの情報を取得したりするために使用するファイルです。DocumentBroker サーバの EDMCrtSimMeta コマンドで作成できます。DMA クラスやプロパティに名前でアクセスする場合に DocumentBroker Development Kit および DocumentBroker Runtime が参照します。

クラスディスクリプション

DMA クラスに関するメタ情報です。DMA 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空間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。

構成管理型リンク
フォルダにバージョン付きオブジェクトを関連付けるときに，バージョン付きオブジェクトの特定のバージョンを固定して関連付けたり，常に最新バージョンをトレースして関連付けたりできるリンクの種別です。構成管理モードとして，FIX モードと FLOATING モードがあります。

構造指定検索
XML 文書を管理している場合に，文書の論理構造や構造の属性をキーとして検索する方法です。例えば，「タイトルに『コンピュータ』という単語が含まれ，章に『XML』という単語が含まれる文書を検索する」というような場合に使用します。また，「『document』というエレメントに設定されている属性『status』の属性値が『public』である文書を検索する」というような場合にも使用できます。

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。DMA で規定され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またはバージョン付き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 Development Kit および DocumentBroker Runtime 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。Java クラスライブラリ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，Java クラスライブラリのクラス，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
文書空間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，その文書空間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，その文書空間オブジェクトに対して実行できます。また，その文書空間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

文書空間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。

セキュリティ運用者

DocumentBroker のアクセス制御の運用情報の管理者です。セキュリティ定義ファイルを保守します。

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，すべての文書空間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空間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。

セキュリティ定義ファイル

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，および文書空間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。

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，Java 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_StIndex プロパティ，edmProp_ConceptTextIndex プロパティおよび edmProp_ConceptStIndex 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 DMA 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 DMA 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 DMA クラスを検索対象として FROM 句に指定します。

属性検索

文書空間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空間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空間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。文書空間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またはバージョン付きフォルダ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またはバージョン付きフォルダ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
一つのフォルダまたはバージョン付き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

Java 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 Sun Java System Directory Server 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。
LDAP 対応のディレクトリサービスとして使用できる製品の詳細については，マニュアル「DocumentBroker Version 3 システム導入・運用ガイド」を参照してください。

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空間オブジェクトです。
edmClass_IndependentPersistence クラスまたはそのサブクラスを基に作成した DMA オブジェクトをトップオブジェクトとする文書空間オブジェクトです。

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

文書空間オブジェクトを構成する DMA オブジェクトのうち，代表的な DMA 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 DMA クラスです。文書空間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

文書空間オブジェクトの処理単位です。トランザクション単位で，文書空間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

Java 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

Java 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空間オブジェクトです。DMA 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空間オブジェクトです。

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空間オブジェクトです。
バージョン付き文書の場合は，DMA 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空間オブジェクトです。バージョン付きフォルダの場合は，DMA オブジェクトの ContainerVersion オブジェクトをトップオブジェクトとする文書空間オブジェクトです。

バージョン管理

文書空間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空間オブジェクトの，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。

バージョン付きフォルダ

複数のバージョンを保持できるフォルダを表す文書空間オブジェクトです。

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空間オブジェクトです。

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空間オブジェクトです。

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空間オブジェクトです。

パーミッション

文書空間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

文書空間オブジェクトとして存在するアクセス制御情報です。複数の文書空間オブジェクトで共有できます。

パラメタクラス

Java 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，文書空間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。文書空間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，文書空間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 DMA クラス
Java クラスライブラリで扱う文書空間オブジェクトのプロパティが実際に定義されている DMA のクラスです。

プロパティディスクリプション
文書空間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空間オブジェクトです。コンテンツを保持できます。

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
DMA 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，バージョン付きフォルダ，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。DocumentBroker Rendering Option 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，DocumentBroker Rendering Option 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。Java 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

Java 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 サーバ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 サーバ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 Development Kit および DocumentBroker Runtime 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。

ランキング検索

全文検索条件に対する適応度をスコアとして算出して，スコアを基に検索結果一覧をソートして出力する検索です。

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 サーバ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 サーバ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
文書空間オブジェクトの関連付けに使用するオブジェクトです。要素は，dmaClass_Relationship クラスのサブクラスを基に作成された DMA 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空間オブジェクトです。

リンク識別子
文書空間オブジェクト間（フォルダとフォルダ間またはフォルダと文書間）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，構成管理モードを変更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，構成管理型リンク，文書間リンクがあります。なお，構成管理型リンクは，構成管理モードによって，さらに 2 種類の種別に分けられます。

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空間オブジェクトです。

例外クラス
Java クラスライブラリで発生する例外のうち，Java 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，HTML 形式のファイル，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

Development Kit および DocumentBroker Runtime 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

文書空間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

文書空間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空間オブジェクトに，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

R

S

T

V

W

X

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

な

は

ひ

ふ

へ

ま

め

も

ゆ

よ

ら

り

る

れ

ろ

わ