HITACHI
Inspire the Next

uCosminexus DocumentBroker Version 3

システム導入・運用ガイド

解説・手引書

3000-3-U71

■対象製品

R-1M95D-13※ uCosminexus DocumentBroker Server Version 3 03-67（適用 OS：AIX 5L V5.1，AIX 5L V5.2，AIX 5L V5.3）R-1L95D-13 uCosminexus DocumentBroker Server Version 3 03-11（適用 OS：Red Hat Enterprise Linux 6 (x86_64)）
R-1M95D-43※ uCosminexus DocumentBroker Development Kit Version 3 03-67（適用 OS：AIX 5L V5.1，AIX 5L V5.2，AIX 5L V5.3）
R-1L95D-43 uCosminexus DocumentBroker Development Kit Version 3 03-11（適用 OS：Red Hat Enterprise Linux 6 (x86_64)）
R-1M95D-53※ uCosminexus DocumentBroker Runtime Version 3 03-67（適用 OS：AIX 5L V5.1，AIX 5L V5.2，AIX 5L V5.3）
R-1L95D-53 uCosminexus DocumentBroker Runtime Version 3 03-11（適用 OS：Red Hat Enterprise Linux 6 (x86_64)）
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，外国為替および外国貿易法ならびに米国の輸出管理関連法規などの規制をご確認の上，必要な手続きをお取りください。
なお，ご不明な場合は，弊社担当営業にお問い合わせください。

■商標類

■マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

製品名称	略称	
Microsoft(R) Windows Server(R) 2003, Enterprise Edition 日本語版	Windows Server 2003	Windows
Microsoft(R) Windows Server(R) 2003, Standard Edition 日本語版		
Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition 日本語版	Windows Server 2003 R2	
Microsoft(R) Windows Server(R) 2003 R2, Standard Edition 日本語版		
Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition 日本語版	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		
Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit 日本語版	Windows Server 2008	
Microsoft(R) Windows Server(R) 2008 Standard 32-bit 日本語版		
Microsoft(R) Windows Server(R) 2008 R2 Enterprise 日本語版	Windows Server 2008 または Windows Server 2008 R2	
Microsoft(R) Windows Server(R) 2008 R2 Standard 日本語版		
Microsoft(R) Windows(R) XP Professional Operating System	Windows XP	
Microsoft(R) Windows Vista(R) Business 日本語版	Windows Vista	
Microsoft(R) Windows Vista(R) Enterprise 日本語版		
Microsoft(R) Windows Vista(R) Ultimate 日本語版		
Microsoft(R) Windows(R) 7 Professional 日本語版 (32 ビット版)	Windows 7	
Microsoft(R) Windows(R) 7 Enterprise 日本語版 (32 ビット版)		
Microsoft(R) Windows(R) 7 Ultimate 日本語版 (32 ビット版)		
Microsoft(R) Windows(R) 7 Professional 日本語版 (64 ビット版)		
Microsoft(R) Windows(R) 7 Enterprise 日本語版 (64 ビット版)		
Microsoft(R) Windows(R) 7 Ultimate 日本語版 (64 ビット版)		
Microsoft(R) Office Word	Word	
Microsoft(R) Word		

Windows Server 2003，Windows Server 2003 R2，Windows Server 2003 R2（x64），Windows Server 2008，Windows Server 2008 R2，Windows XP，Windows Vista，および Windows 7 を総称して Windows と表記することがあります。

■発行

2012 年 5 月 (第 1 版) 3000-3-U71

■著作権

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の環境設定および運用方法について説明したものです。

- R-1M95D-13　DocumentBroker Server Version 3
- R-1L95D-13　DocumentBroker Server Version 3
- R-1M95D-43　DocumentBroker Development Kit Version 3
- R-1L95D-43　DocumentBroker Development Kit Version 3
- R-1M95D-53　DocumentBroker Runtime Version 3
- R-1L95D-53　DocumentBroker Runtime Version 3

■対象読者

このマニュアルは，DocumentBroker の環境を管理および運用するシステム管理者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（AIX，および Linux）に関する知識
- HiRDB に関する知識
- XML に関する知識
- 分散オブジェクト技術に関する知識

■このマニュアルで使用する記号

このマニュアルで使用する記号を次に示します。

記号	意味
｜	横に並べられた複数の項目に対する項目間の区切りを示し，「または」の意味を表します。 （例） A ｜ B A または B を指定することを示します。
＿	括弧で囲まれた複数項目のうち 1 項目に対し使用され，括弧内のすべてを省略した場合にシステムが取る標準値を示します。 （例）〔A ｜ B〕 何も指定しない場合は A が仮定されます。
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを示します。項目が横に並べられ，記号｜で区切られている場合は，そのうちの一つを選択します。 （例） {A ｜ B ｜ C} A，B または C のどれかを指定することを示します。
〔 〕	この記号で囲まれている項目は省略してもよいことを意味します。複数の項目が横に並べて記述されている場合には，すべてを省略するか，記号 { } と同じくどれか一つを選択します。 （例 1）〔A〕 「何も指定しない」か「A を指定する」ことを示します。 （例 2）〔B ｜ C〕 「何も指定しない」か「B または C を指定する」ことを示します。
< >	この記号で囲まれている項目は，該当する要素を指定することを示します。 （例） <プロパティ> プロパティを記述します。
：：＝	この記号の左にあるものを右にあるもので定義することを示します。 （例） A：：＝ B 「A とは B である」と定義することを示します。
...	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ユーザ指定値の内容を表す記号です。

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

注　すべて半角文字を使用してください。

■このマニュアルの図中で使用する記号

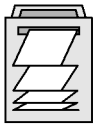
このマニュアルの図中で使用する記号を，次のように定義します。

●ワークステーション　端末

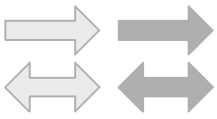
●サーバ

●入出力の動作

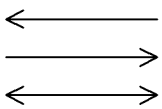
●プリンタ



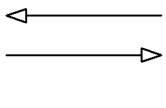
●データの流れ



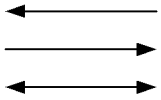
●制御の流れ



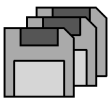
●クラスの継承関係



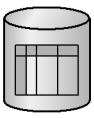
●その他の流れ



●フロッピーディスク　（記録媒体）



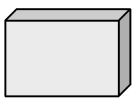
●データベース



●ファイル，システム

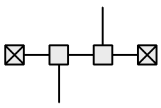
●文書，コンテント

●プログラム

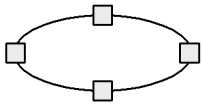


●ネットワーク

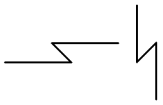
●ネットワーク　（ＬＡＮ）



●ネットワーク　（ＬＡＮ）



●通信回線



●プロセス

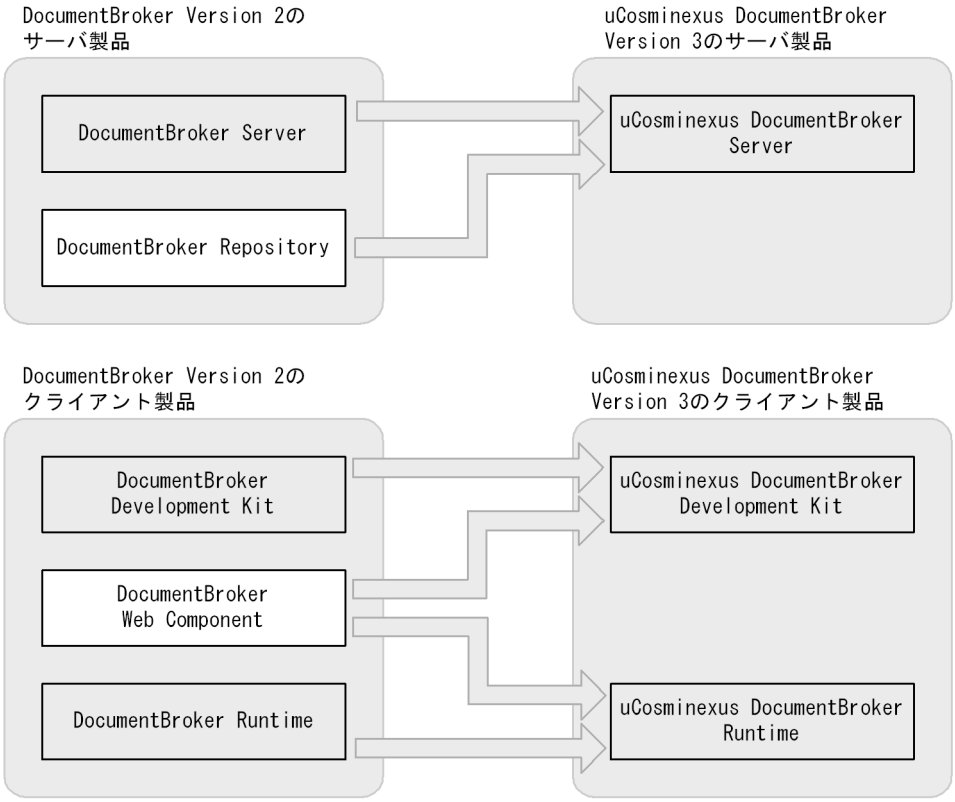
●磁気テープ

■ 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

Version 2 のマニュアル名称	Version 3 のマニュアル名称
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
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	DocumentBroker Rendering Option Version 3
	DocumentBroker Rendering Option Version 3 （活文 PDFstaff 編）
DocumentBroker Object Loader Version 2	uCosminexus DocumentBroker Object Loader Version 3

目次

1 概説 1

1.1 DocumentBroker の概要 2

1.1.1 DocumentBroker の目的 2

1.1.2 DocumentBroker の特長 4

1.2 DocumentBroker の機能 7

1.2.1 文書の登録機能とバージョン管理機能 8

1.2.2 マルチレンディション管理機能 9

1.2.3 マルチファイル管理機能 10

1.2.4 コンテナ管理機能 11

1.2.5 文書の構成管理機能 12

1.2.6 文書間リレーション管理機能 13

1.2.7 文書の属性情報の管理機能 14

1.2.8 リファレンスファイル管理機能 14

1.2.9 レンディションのコンテント種別変換機能 15

1.2.10 File Link 連携機能 18

1.2.11 XML 文書管理機能 19

1.2.12 独立データの管理機能 20

1.2.13 アクセス制御機能 20

1.2.14 アクセスログおよびトレースログの統計・解析機能 22

1.2.15 検索機能 22

1.2.16 オブジェクト一括登録機能 24

1.2.17 全文検索インデクス一括登録機能 24

1.2.18 レンディション変換機能 24

1.2.19 イメージ文書の登録機能 25

1.2.20 分散した文書の収集・検索機能 25

1.2.21 文書の分析機能 26

1.2.22 文書のライフサイクル管理機能 26

1.2.23 ファイル転送機能 26

1.2.24 複数の実行環境機能 26

1.2.25 システム導入支援機能 27

1.2.26 クライアントアプリケーション動作定義機能 27

1.3 DocumentBroker のシステム構成 28

1.3.1 DocumentBroker を使用したシステムの概略 28

1.3.2 DocumentBroker サーバを構成するプログラム 29

1.3.3 DocumentBroker Development Kit で開発したクライアントを構成するプログラム（C++ クラスライブラリの場合） 33

1.3.4 DocumentBroker Development Kit で開発したクライアントを構成するプログラム（Java クラスライブラリの場合） 37

1.3.5 DocumentBroker Standard GUI を使用する場合のクライアントを構成するプログラム 40

1.4 プロセス構成 43

1.4.1 DocumentBroker の基本プロセス構成 43
1.4.2 DocumentBroker のプロセスの関連 44

2 システム導入前の検討 47
2.1 適用業務の検討 48
2.2 文書管理に使用するオブジェクト 50
2.2.1 DocumentBroker のオブジェクト 50
2.2.2 このマニュアルで使用するオブジェクト名と機能名 51
2.2.3 バージョンなし文書の管理に必要なオブジェクト 52
2.2.4 バージョン付き文書の管理に必要なオブジェクト 53
2.2.5 マルチレンディション文書の管理に必要なオブジェクト 55
2.2.6 コンテナの管理に必要なオブジェクト 57
2.2.7 構成管理コンテナの管理に必要なオブジェクト 58
2.2.8 文書間リレーションの管理に必要なオブジェクト 61
2.2.9 XML 文書の管理に必要なオブジェクト 62
2.2.10 独立データの管理に必要なオブジェクト 64
2.2.11 全文検索機能付き文書クラスを作成する場合に必要なオブジェクト 65
2.2.12 オブジェクトとデータベースの関連 65
2.3 追加するクラスおよびプロパティの検討 67
2.3.1 クラスおよびプロパティの追加 67
2.3.2 サブクラスの追加 68
2.3.3 プロパティの追加 69
2.3.4 全文検索機能付き文書クラスの追加 70
2.3.5 クラスおよびプロパティの追加例 74
2.4 文書空間で使用する文字コード種別の検討 76
2.5 XML 文書管理機能を使用したシステム構築の検討 77
2.5.1 DocumentBroker で提供する XML 文書管理機能 77
2.5.2 XML プロパティマッピング機能 77
2.5.3 XML インデクスデータ作成機能 80
2.5.4 DocumentBroker が管理できる XML 文書 82
2.6 アクセス制御機能を使用したシステム構築の検討 85
2.6.1 アクセス制御情報の概要 85
2.6.2 文書管理モデルへのアクセス制御の適用 89
2.7 メモリ所要量とディスク占有量の見積もり 96
2.7.1 仮想メモリ所要量の見積もり 96
2.7.2 VariableArray 型のプロパティが使用するメモリ所要量の見積もり 97
2.7.3 ディスク占有量の見積もり 98
2.8 データベース容量の見積もり 99
2.8.1 データベース容量の見積もり方法 99
2.8.2 ユーザ用 RD エリアの容量の見積もり 99
2.8.3 ユーザ LOB 用 RD エリアの容量の見積もり 107
2.8.4 アクセス制御機能を使用する場合のデータベース容量の見積もり 110

2.9 リファレンスファイル管理機能を使用する場合のファイルシステムのディスク容量の見積もり 113
2.10 File Link 連携機能を使用する場合のファイルサーバ容量の見積もり 114
2.11 マルチレンディション管理機能を使用する場合の排他資源管理テーブルの見積り 115

3 環境設定 117
3.1 環境設定の流れ 118
3.2 環境設定の準備 120
3.3 インストールとアンインストール 121
3.3.1 インストール 121
3.3.2 アンインストール 124
3.3.3 インストール済みプログラムの一覧表示 125
3.4 オペレーティングシステムでの環境設定 127
3.4.1 システム管理者の登録 127
3.4.2 パラメタのカスタマイズ 127
3.5 TPBroker での環境設定 130
3.6 DocumentBroker の実行環境の作成 131
3.6.1 DocumentBroker 実行環境ディレクトリの作成 131
3.6.2 環境変数の設定（AIX の場合） 131
3.6.3 環境変数の設定（Linux の場合） 133
3.6.4 DocumentBroker の実行環境作成コマンドの実行 135
3.6.5 文書空間識別子の定義 135
3.7 ユーザ管理機能の設定 136
3.7.1 LDAP 対応のディレクトリサービスによるユーザ管理機能を使用する場合の設定 136
3.7.2 UNIX のパスワードファイルによるユーザ管理機能を使用する場合の設定 139
3.7.3 ユーザ作成のアクセスルーチンを使用する場合の設定 140
3.8 文書空間の定義 142
3.8.1 文書空間を定義する前の準備 142
3.8.2 ユーザ情報の取得方法 143
3.9 アクセス制御機能を使用する場合の設定 146
3.9.1 アクセス制御を使用するための管理者の設定 146
3.9.2 アクセス制御機能の設定 146
3.10 データベースシステムでの環境設定 148
3.10.1 HiRDB の環境設定の準備 148
3.10.2 HiRDB の環境設定 148
3.10.3 HiRDB のユーザ権限の設定 149
3.10.4 HiRDB Text Search Plug-in での環境設定 149
3.11 DocumentBroker でデータベースシステムを使用するための設定 150
3.11.1 データベース定義の名称の検討 150
3.11.2 データベースシステムの設定に必要なファイル 151
3.11.3 データベースシステムの設定手順 152
3.11.4 全文検索機能を使用する場合に出力されるデータベース定義文 154

3.12 システム導入支援機能での設定 162
3.13 ファイル転送機能を使用する場合の設定 164
3.13.1 ファイル転送機能の概要 164
3.13.2 ファイル転送機能のための DocumentBroker クライアントでの環境設定 166
3.13.3 ファイル転送機能のための DocumentBroker サーバでの環境設定 172
3.13.4 ファイル転送機能で使用するコマンド 173
3.13.5 ファイル転送サービスの開始と停止 175
3.14 サービスプロセスを使用する場合の設定 176
3.15 複数の実行環境を構築する場合の設定 177
3.16 リファレンスファイル管理機能を使用する場合の設定 179
3.16.1 コンテントを管理するための設定 179
3.16.2 ネットワーク上のマシンの共有ディスクでコンテントを管理する場合の設定 180
3.17 File Link 連携機能を使用する場合の設定 182
3.18 XML 文書管理機能を使用する場合の設定 183
3.18.1 定義ファイルの作成・生成の概要 183
3.18.2 XML 定義ファイルの登録 185
3.18.3 XML 文書管理機能の環境設定 186
3.18.4 XML 定義ファイルを作成する場合の注意事項 190

4 環境設定で必要なファイル 193
4.1 ファイルの種類 194
4.2 DocumentSpace 構成定義ファイル（docspace.ini） 196
4.2.1 DocumentSpace 構成定義ファイルの概要 196
4.2.2 DocumentSpace 構成定義ファイルの記述形式 200
4.2.3 DocumentSpace 構成定義ファイルの記述例 218
4.2.4 DocumentSpace 構成定義ファイルの注意事項 222
4.3 セキュリティ定義ファイル（docaccess.ini） 224
4.3.1 セキュリティ定義ファイルの概要 224
4.3.2 セキュリティ定義ファイルの記述形式 224
4.3.3 セキュリティ定義ファイルの記述例 227
4.4 ユーザ権限定義ファイル 228
4.4.1 ユーザ権限定義ファイルの概要 228
4.4.2 ユーザ権限定義ファイルの記述形式 228
4.4.3 ユーザ権限定義ファイルの記述例 229
4.5 RD エリア構成定義ファイル 230
4.5.1 RD エリア構成定義ファイルの概要 230
4.5.2 RD エリア構成定義ファイルの記述形式 231
4.5.3 RD エリア構成定義ファイルの記述例（全文検索用） 231
4.6 メタ情報ファイル 232
4.7 定義情報ファイル 233
4.7.1 定義情報ファイルの概要 233

4.7.2 サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則 233
4.7.3 定義情報ファイルの記述形式 234
4.7.4 定義情報ファイルの記述例 237
4.7.5 定義情報ファイルの記述例（VariableArray 型のプロパティを追加する場合） 245
4.7.6 定義情報ファイルの記述例（全文検索機能を使用する文書クラスを追加する場合） 250
4.7.7 定義情報ファイルの記述例（概念検索機能を使用する文書クラスを追加する場合） 251
4.7.8 定義情報ファイルの記述例（文字列型プロパティに対する全文検索機能を使用する場合） 252
4.8 RD エリア定義情報ファイル 257
4.8.1 RD エリア定義情報ファイルの概要 257
4.8.2 RD エリア定義情報ファイルの記述形式 257
4.8.3 RD エリア定義情報ファイルの記述例 263
4.8.4 RD エリア定義情報ファイルの注意事項 263
4.9 インデクス情報ファイル 265
4.9.1 インデクス情報ファイルの概要 265
4.9.2 インデクス名をデータベース定義の名称に使用する場合の規則 265
4.9.3 全文検索での差分インデクス機能を使用するための設定 265
4.9.4 インデクス情報ファイルの記述形式 266
4.9.5 インデクス情報ファイルの記述例 268
4.10 クラス定義情報ファイル 269
4.10.1 クラス定義情報ファイルの概要 269
4.10.2 クラス定義情報ファイルの出力形式 269
4.10.3 クラス定義情報ファイルの出力例 274
4.11 ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイル 277
4.11.1 ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの概要 277
4.11.2 ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力形式 277
4.11.3 ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力例 277
4.12 ファイル転送サービス環境定義ファイル（ftpsv.ini） 282
4.12.1 ファイル転送サービス環境定義ファイルの概要 282
4.12.2 ファイル転送サービス環境定義ファイルの記述形式 282
4.12.3 ファイル転送サービス環境定義ファイルの記述例 284
4.13 サービスプロセス定義ファイル 286
4.13.1 サービスプロセス定義ファイルの概要 286
4.13.2 サービスプロセス定義ファイルの記述形式 286
4.13.3 サービスプロセス定義ファイルの記述例 287
4.14 プロパティマッピング定義ファイル 289
4.14.1 プロパティマッピング定義ファイルの概要 289
4.14.2 プロパティマッピング定義ファイルの記述形式 289
4.14.3 プロパティマッピング定義ファイルの記述例 294
4.14.4 プロパティマッピング定義ファイルの注意事項 296
4.15 文書クラス定義ファイル 298
4.15.1 文書クラス定義ファイルの概要 298
4.15.2 追加できるクラス，およびプロパティの種類 298

4.15.3 文書クラス定義ファイルの記述形式 300
4.15.4 文書クラス定義ファイルの注意事項 309
4.16 文書空間情報ファイル 311
4.16.1 文書空間情報ファイルの概要 311
4.16.2 文書空間情報ファイルの記述形式 312
4.16.3 文書空間情報ファイルの記述例 319
4.17 見積もり基礎情報ファイル 320
4.17.1 見積もり基礎情報ファイルの概要 320
4.17.2 見積もり基礎情報ファイルの出力形式 320
4.17.3 見積もり基礎情報ファイルの出力例 322
4.18 クライアントアプリケーション動作定義ファイル（application.ini） 324
4.18.1 クライアントアプリケーション動作定義ファイルの概要 324
4.18.2 クライアントアプリケーション動作定義ファイルの記述形式 324
4.18.3 クライアントアプリケーション動作定義ファイルの記述例 326

5 DocumentBroker の起動と終了 329
5.1 DocumentBroker の起動方法 330
5.2 DocumentBroker の終了方法 331

6 運用と障害対策 333
6.1 データベースの運用 334
6.1.1 バックアップと回復 334
6.1.2 ジャーナルファイルの運用 337
6.1.3 レプリケーション 337
6.2 DocumentBroker のクラス定義の変更 339
6.2.1 クラスの定義を変更する場合について 339
6.2.2 クラスの追加 339
6.2.3 クラスの削除（オブジェクトが存在しない場合） 343
6.2.4 クラスの定義の変更（オブジェクトが存在しない場合） 344
6.2.5 クラスの削除（オブジェクトが存在する場合） 345
6.2.6 クラスの定義の変更（オブジェクトが存在する場合） 345
6.2.7 メタ情報の追加・削除コマンドの動作 347
6.2.8 追加するプロパティと列の並び 349
6.3 リファレンスファイル管理機能を使用する場合の運用 351
6.3.1 コンテントの格納先ディレクトリの運用方法 351
6.3.2 ファイルシステムのバックアップとリストア 351
6.4 障害対策 353
6.4.1 障害が発生した時に DocumentBroker が出力する情報 353
6.4.2 サーバマシンでの障害 354
6.4.3 データベースでの障害 355
6.4.4 セッション障害 356

6.4.5 定義情報の障害 356
6.4.6 ログの障害 357
6.4.7 DocumentBroker での障害 357
6.4.8 サーバ側で発生した障害情報の取得 358
6.4.9 クライアント側で発生した障害情報の取得 360
6.4.10 データベースで発生したデッドロック情報およびタイムアウト情報の取得 362
6.5 アクセスログに関する運用 363
6.5.1 アクセスログの取得 363
6.5.2 アクセスログの出力形式 363
6.5.3 出力ログ情報 364
6.6 エラーログに関する運用 368
6.6.1 エラーログの取得 368
6.6.2 エラーログの出力形式 368

7 コマンドリファレンス 369
7.1 コマンドの種類 370
7.1.1 システム運用コマンド 370
7.1.2 データベース運用コマンド 370
7.1.3 システム導入支援コマンド 371
7.1.4 トラブルシュートコマンド 372
7.2 コマンドの形式 373
7.3 コマンドの文法 375

付録 419
付録 A ディレクトリ構成（AIX の場合） 420
付録 A.1 DocumentBroker のインストールディレクトリの構成（AIX の場合） 420
付録 A.2 DocumentBroker の実行環境ディレクトリの構成（AIX の場合） 421
付録 B ディレクトリ構成（Linux の場合） 425
付録 B.1 DocumentBroker のインストールディレクトリの構成（Linux の場合） 425
付録 B.2 DocumentBroker の実行環境ディレクトリの構成（Linux の場合） 426
付録 C クラスタリングシステムでの運用 429
付録 C.1 HACMP によるクラスタリングシステムでの運用 429
付録 C.2 クラスタリングシステムでのコマンドの実行 448
付録 D 全文検索インデクスに UCS-4 の文字を使用する場合のデータベースの設定 450
付録 E データベース移行 453
付録 E.1 アクセス制御機能を使用するためのデータベース移行 453
付録 E.2 クラス名やプロパティ名などをデータベース定義の名称に使用するためのデータベース移行 455
付録 E.3 HiRDB の繰り返し列を使用するためのデータベース移行 458
付録 E.4 マルチファイル管理機能を使用するためのデータベース移行 460
付録 E.5 リファレンスファイル管理機能を使用するためのデータベース移行 462

付録 E.6　File Link 連携機能を使用するためのデータベース移行　464
付録 F　システムクラスおよびシステムプロパティの名称定義の規則　467
付録 F.1　システムクラスの名称定義の規則　467
付録 F.2　システムプロパティの名称定義の規則　467
付録 G　ユーザ作成のアクセスルーチンを使用するための関数　471
付録 G.1　dbrinitLibrary　472
付録 G.2　dbrinitSession　472
付録 G.3　dbrAuthenticateUser　473
付録 G.4　dbrGetUserInformation　474
付録 G.5　dbrFinalizeSession　475
付録 G.6　dbrFinalizeLibrary　476
付録 G.7　dbrSetDocSpaceCharacterSet　477
付録 H　メタ情報の記述内容　478
付録 H.1　ini ファイルのシンタクスの基本項目　478
付録 H.2　クラス定義　480
付録 H.3　プロパティ定義　481
付録 H.4　リストオブジェクト　490
付録 H.5　QueryOperator クラス定義　492
付録 H.6　QueryOperand クラス定義　494
付録 H.7　オブジェクト定義　495
付録 H.8　メタ情報管理用に確保する共用メモリセグメントサイズ　496
付録 I　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機能およびコマンド　497
付録 J　DocumentBroker Web Client を使用した DocumentBroker クライアントの開発　500
付録 K　このマニュアルの参考情報　503
付録 K.1　関連マニュアル　503
付録 K.2　このマニュアルでの表記　508
付録 K.3　uCosminexus DocumentBroker のマニュアルで使用する略語　511
付録 K.4　KB（キロバイト）などの単位表記について　512
付録 L　用語解説　514

索引　529

1 概説

この章では，DocumentBroker の概要について説明します。

1.1 DocumentBroker の概要

1.2 DocumentBroker の機能

1.3 DocumentBroker のシステム構成

1.4 プロセス構成

1.1 DocumentBroker の概要

この節では，DocumentBroker の目的と特長について説明します。

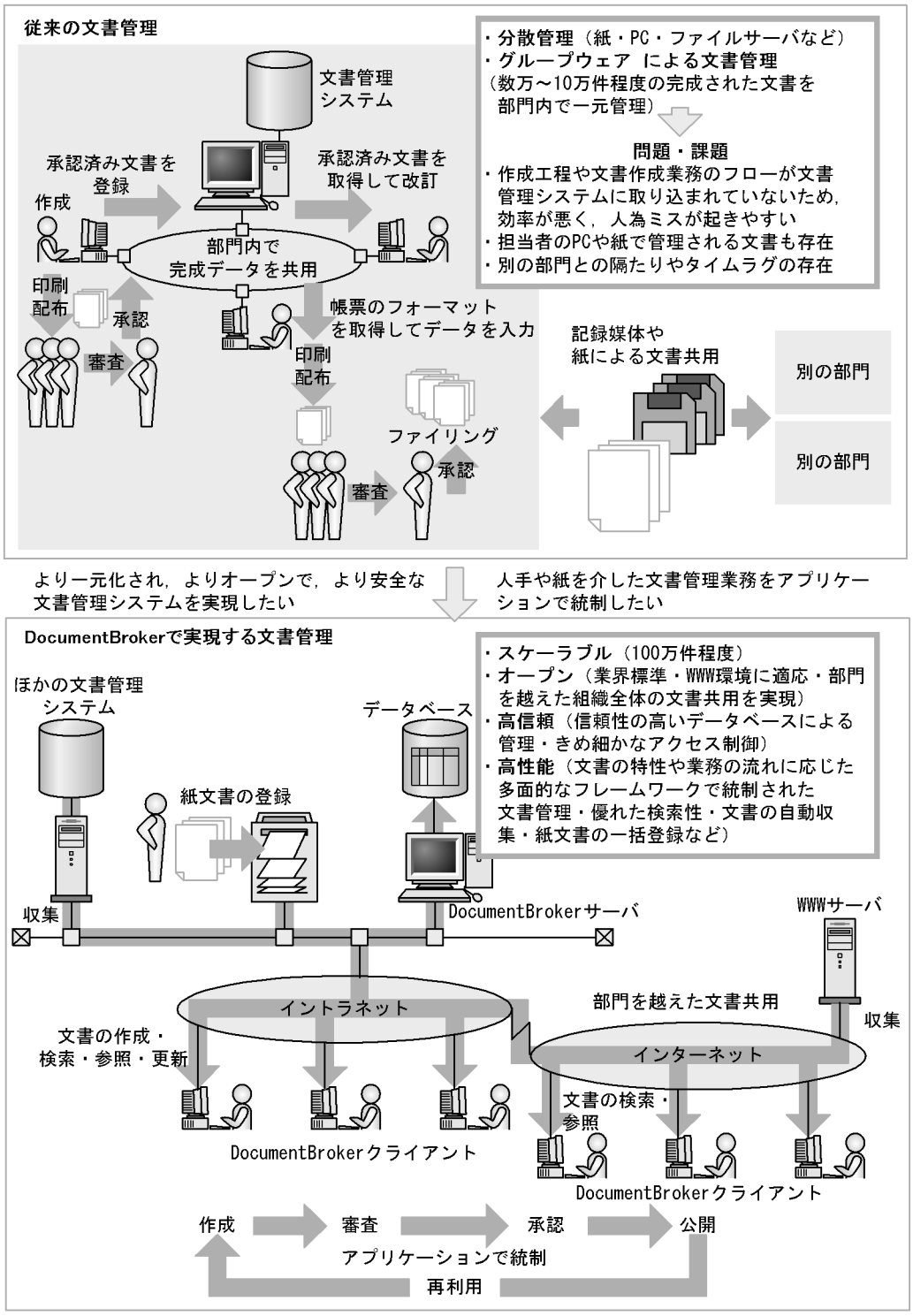
1.1.1 DocumentBroker の目的

DocumentBroker は，組織内に分散するさまざまな文書を，多面的に一元管理する文書管理基盤ソフトウェアであり，次の目的で開発されています。

- 業務に合わせて文書をわかりやすく整理できるさまざまなフレームワークを提供し，組織内に蓄積された膨大な量の文書の有効活用を促進する。
- 完成された文書を保管するだけでなく，文書の作成や収集から，公開，そして再利用や破棄までの流れをアプリケーションによって統制する。

DocumentBroker を利用した文書管理システムは，従来の文書管理システムでの問題や課題を解決し，業務の効率および生産性を向上させることができます。DocumentBroker は，文書を基幹業務として扱う企業や組織で，業務の形態に合わせた，スケーラブル，オープン，高信頼，そして高性能な文書管理システムの構築を支援します。DocumentBroker で実現する文書管理の概要を次の図に示します。

図 1-1 DocumentBroker で実現する文書管理システム



(1) 従来の文書管理

企業，官公庁，研究機関などの大規模な組織では，膨大な量の文書が作成されます。作成される文書の種類は，簡単なメモ，定型業務に利用される帳票や申請書類，会議の議事録，新聞記事，技術論文，マニュアルなど，業種や部門によってさまざまです。従来，紙で管理されていたこれらの文書は，パーソナルコンピュータなどの普及によって，電子化されて保存されることが多くなりました。

電子化された文書は，各担当者のパーソナルコンピュータで個別に管理され，フロッピーディスクなどの記録媒体を介して共用されたり，部門別のファイルサーバで共用されたりしてきました。その結果，組織内で日々作成される文書が，担当者のパーソナルコンピュータや部門別のファイルサーバに分散されてしまい，即時に組織全体で有用な文書として共用することが困難でした。また，最新の文書であるか，承認済みの文書であるか，などといった文書の状態については，担当者の認識に依存して管理されてきました。

そのあと，グループウェアの文書管理システムが発達して，電子化された文書はデータベースに格納されて一元管理されるようになり，文書の共用化が促進されてきました。グループウェアの文書管理システムは，完成された文書を保管し，組織内で共用することを目的としてきました。通常，数万～ 10 万件程度の文書の登録件数が想定され，完成された過去の文書を登録しておいて改訂時に流用したり，再利用できる書類のフォーマットを登録しておいて書類作成時に利用したりすることに主眼が置かれてきました。文書が完成されるまでの段階では，各担当者のパーソナルコンピュータで個別に文書が管理され，完成された時点で文書管理システムに登録されて共用されるという運用がほとんどで，最新の文書を正しく登録するかどうかについては，やはり担当者の認識に依存してきました。

このため，文書を作成する担当者の認識に依存しないで運用できる文書管理システムが望まれるようになってきました。また，容量の制限や安全性などの問題点から，従来は共用化されることのなかった組織の重要な基幹文書の管理も任せられる文書管理システムが必要とされています。基幹文書は，本来，作成／審査／承認といった業務の流れを経て完成されるものです。このため，単に「大規模，大容量で」,「安全に」,「完成された文書を保管できる」という文書管理システムだけでは不十分です。文書を作成する業務の流れに従って，作成から保管，破棄までの過程についても適切に管理できる文書管理システムが望まれています。

(2) DocumentBroker で実現する文書管理

DocumentBroker は，文書の変更履歴，作成中の文書の審査者による指摘，関連文書など，その文書に関する情報を一括管理して，組織全体で活用することを目的とします。文書の特性や作成方式，文書が作成される業務の流れなどを考慮して，文書をわかりやすく整理できるさまざまなフレームワークや連携機能を用意しているため，文書をさまざまな視点で分類したり，まとめたりして多面的な管理が実現できます。文書の内容による分類だけでなく，一つのファイルで構成される文書なのか，構造を持った複数ファイルで構成される文書なのか，担当者一人が作成する文書なのか，複数人で分担して作成する文書なのか，などといったさまざまな要素を考慮に入れて，文書管理のフレームワークを適用できます。

文書の登録件数は 100 万件程度で，きめ細かなアクセス制御や，文書の属性やキーワード，構造を意識した検索機能も備えています。

出版原稿や特許の申請書など，基幹業務で作成される文書の場合，業務を効率良く円滑に行うために最適な文書管理システムが必要です。また，保守マニュアルや設計図面など，業務を遂行するために必要な文書の場合，それらの情報を組織内で共用して，業務で効果的に活用できるようにするために，電子データとして厳密に管理する必要があります。このような場合，DocumentBroker が提供しているフレームワークを利用すると，文書の「特性」や「ライフサイクル」などを意識し，既存の業務システムと密接に連携した文書管理システムを構築できます。

DocumentBroker は，文書を基幹業務として扱う分野での，文書の共用による組織活動のスピードアップを実現するための文書管理システムの構築をサポートします。

1.1.2 DocumentBroker の特長

DocumentBroker の特長について説明します。

(1) システムの運用性，信頼性を確保

(a) すべての文書をデータベースシステムで保管

すべての文書および文書に付随する情報を，データベースシステムである HiRDB に格納します。小規模な基幹文書管理システムはもちろん，大規模，大容量の基幹文書管理システムを運用する上での運用性と，高い信頼性を確保しています。増え続けていくデータ量や業務の規模の拡大に合わせて段階的にシステムを拡張できます。

(b) ファイルシステムとの併用による高い運用性

文書の格納先として，データベースシステムだけでなく，ファイルシステムの任意のディレクトリを使用できます。これによって，文書の格納先を選択したり，文書を格納するディスクを移動したりできるため，柔軟に文書を管理できます。また，文書の格納先としてファイルシステムとデータベースシステムを併用することで，より大容量の文書を長期にわたって管理できるようになります。

(c) きめ細かなアクセス制御を実現

すべての文書に対して，担当業務単位，役割単位，ユーザ単位などに細分化したアクセス権を設定できます。これによって，文書に対する不正なアクセスによる情報の漏えいを防止できます。

(2) 必要な文書をすばやく確実に入手

文書中の単語をキーワードにした高速の全文検索，属性を利用した絞り込み検索はもちろん，XML の構造を指定した全文検索（構造指定検索），指定した文章と類似する概念を持つ文書の全文検索（概念検索）など，豊富な検索機能を提供しています。したがって，管理している膨大な量の文書の中から目的の文書をすばやく確実に入手できる検索システムを構築できます。

(3) 高度な文書管理を実現するさまざまなフレームワークを提供

複数の文書をフォルダにまとめて格納する従来の文書管理の基本的なフレームワークはもちろん，主に次のような文書管理のフレームワークを提供しています。

- 複数の視点で文書を分類するフォルダを利用したフレームワーク
- 文書の複数のバージョンをまとめて管理するフレームワーク
- 章ごとに複数に分割された文書を複数の担当者で分担して作成する場合に，分割された文書のバージョン構成を保ちながら一括管理するフレームワーク

これらのフレームワークを，業務の形態や扱う文書の特性に合わせて適用できます。

(4) これまで蓄積されてきた文書をむだなく登録

文書をすばやく入手できる検索システムや文書をわかりやすく整理するフレームワークが準備できても，文書が登録されていなければ，活用できません。DocumentBroker は，次のようなプログラムを用意して，有用で豊富な情報量を持つ文書管理システムを短期間で構築できるようサポートしています。

- 既存のデータベースのデータを DocumentBroker に一括して登録するプログラム
- 部門別のファイルサーバ，既存の文書管理システム，インターネット・イントラネットなどに分散している文書の中から有用な文書をシームレスに収集・登録し，検索するプログラム
- デジタル複合機を使用して紙で蓄積された文書や FAX で受信した文書を読み込むことでイメージファイルとして一括登録するプログラム

(5) さまざまな形式のデータの登録

日常利用しているワードプロセッサや表計算ソフトウェアで作成したデータ，イメージデータ，XML 形

式など，さまざまな形式のデータを，そのまま文書管理システムに格納して利用できます。同じ内容の文書を幾つかの異なるファイル形式で登録して用途によって使い分けたり，紙の文書をスキャナで読み込んでイメージデータとして登録したり，XML 形式の文書の構造を生かして登録・検索したりできます。

(6) オープンな環境でシステムを構築

DocumentBroker によって業界標準に対応したシステムを構築できます。DocumentBroker は，DMA に基づいたオブジェクトモデルと API を提供しています。DMA は文書管理インターフェースの標準化を図る団体 AIIM によって定義される共通インターフェースです。また，DocumentBroker によって構築した文書管理システムは，CORBA 仕様に従った ORB を利用した分散ネットワーク上で運用します。したがって，既存の業務システムと相互連携した付加価値の高いサービスを提供できます。

また，WWW 環境に対応して，さらにオープンな文書管理システムも構築できます。

(7) 文書のライフサイクルを管理

文書および文書に関する情報を，業務の流れに対応させて管理できます。

通常，文書は業務の流れに対応して作成，審査，承認，保管され，不要になった時点で破棄されていきます。このような，業務に合わせた文書の状態変化（作成／審査／承認／保管／破棄）を，**文書のライフサイクル**といいます。

1.2 DocumentBroker の機能

この節では，DocumentBroker の機能について説明します。

DocumentBroker の文書管理システムで提供する機能の一覧を次の表に示します。

表 1-1 DocumentBroker の文書管理システムで提供する機能の一覧

分類	機能	機能の実現に必要な関連製品 とユティリティ製品	参照先
文書管理	文書の登録機能	—	1.2.1
	バージョン管理機能	—	
	マルチレンディション管理機能	—	1.2.2
	マルチファイル管理機能	—	1.2.3
	コンテナ管理機能	—	1.2.4
	文書の構成管理機能	—	1.2.5
	文書間リレーション管理機能	—	1.2.6
	文書の属性情報の管理機能	—	1.2.7
	リファレンスファイル管理機能	—	1.2.8
	レンディションのコンテント種別変換機能	—	1.2.9
	File Link 連携機能	HiRDB File Link	1.2.10
構造化文書管理	XML 文書管理機能	• Preprocessing Library for Text Search • HiRDB Adapter for XML	1.2.11
独立データの管理	独立データの管理機能	—	1.2.12
アクセス制御	アクセス制御機能	LDAP 対応のディレクトリサービス	1.2.13
統計・解析	アクセスログおよびトレースログの統計・解析機能	—	1.2.14
検索	属性検索	—	1.2.15
	全文検索	• HiRDB Object Option ※1 • HiRDB Text Search Plug-in	
	構造指定検索	• HiRDB Object Option ※1 • HiRDB Text Search Plug-in • Preprocessing Library for Text Search ※2	
	概念検索	• HiRDB Object Option ※1 • HiRDB Text Search Plug-in • HiRDB Text Search Plug-in Conceptual Extension	

分類	機能	機能の実現に必要な関連製品 とユティリティ製品	参照先
	文字列型プロパティに対する全文検索	• HiRDB Object Option ※1 • HiRDB Text Search Plug-in	
変換・登録	オブジェクト一括登録機能	DocumentBroker Object Loader	1.2.16
	全文検索インデクス一括登録機能	DocumentBroker Text Search Index Loader	1.2.17
	レンディション変換機能	DocumentBroker Rendering Option	1.2.18
	イメージ文書の登録機能	DocumentBroker イメージ登録 for DocuCentre, DocumentBroker イメージ登録 for iR, DocumentBroker イメージ登録 for imagio, DocumentBroker イメージ登録 for Konica, または DocumentBroker イメージ登録文字認識オプション	1.2.19
収集・検索	分散した文書の収集機能	DocumentBroker Collector	1.2.20
	収集した文書の検索機能	DocumentBroker Integrated Search Suite for J，または DocumentBroker Integrated Search Suite	
分析	文書の分析機能	• DocumentBroker テキスト分析コンポーネント • DocumentBroker テキスト分析テンプレート	1.2.21
ライフサイクル管理	文書のライフサイクル管理機能	DocumentBroker Life Cycle Suite	1.2.22
環境構築	ファイル転送機能	—	1.2.23
	複数の実行環境機能	—	1.2.24
	システム導入支援機能	—	1.2.25
	クライアントアプリケーション動作定義機能	—	1.2.26

注※ 1

HiRDB Version 6 または HiRDB Version 7 を使用する場合に必要になります。

注※ 2

XML 文書の構造指定検索に必要です。

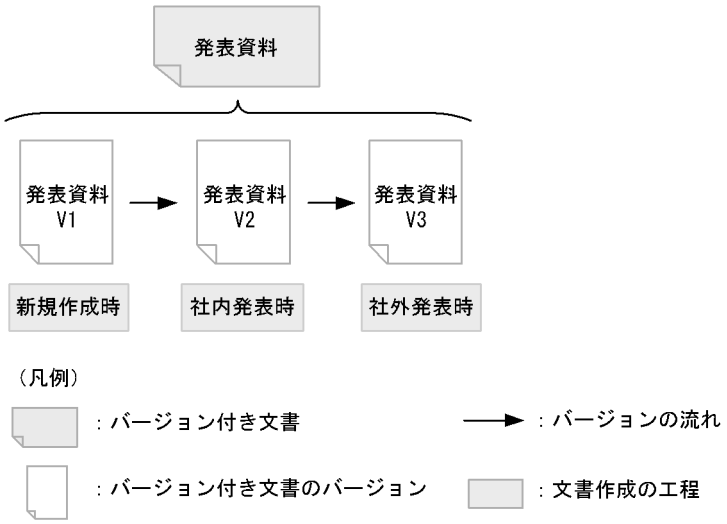
1.2.1 文書の登録機能とバージョン管理機能

文書をデータベースに登録して一元管理できます。また，文書を登録する際には，必要に応じて「Version1」や「Version2」などの版（**バージョン**）を付けて，文書の履歴を管理できます。文書のバージョンを管理することで，過去のある時点での文書を復元したり，その文書を流用して新たに文書を作成したりできます。一つのバージョンだけを持ち，バージョンを管理しない文書を，**バージョンなし文書**と

いいます。複数のバージョンを持ち，バージョンを管理する文書を，**バージョン付き文書**といいます。

バージョン付き文書の管理例を次の図に示します。

図 1-2　バージョン付き文書の管理例



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。

文書のバージョンを管理するときには，新しく文書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いきます。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。チェックアウト中はチェックアウトしたユーザ以外は文書を上書きできないため，ほかのユーザによる文書の二重更新を防止できます。

1.2.2　マルチレンディション管理機能

文書の実体であるコンテント（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，その形式を表す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**といいます。

バージョンなし文書とバージョン付き文書には，1 個または複数のレンディションを登録して管理できます。一つの文書に対して同じ内容を表す複数のレンディションを登録して管理する機能のことを，マルチレンディション管理機能といいます。また，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文書は，一つの文書の内容を，対応するアプリケーションごとの複数の形式に変換した場合などに使用できます。

マルチレンディション管理機能では，次の 2 種類のレンディションを区別して扱います。

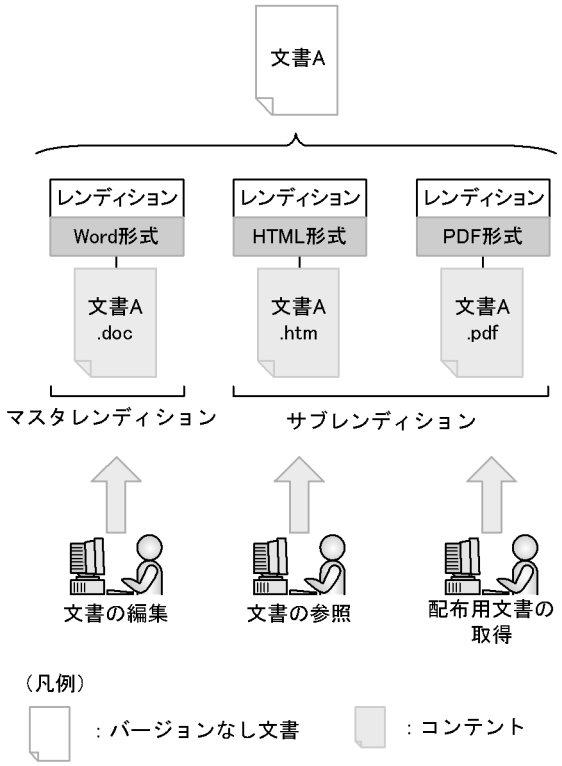
- **マスタレンディション**
  主要なレンディションです。レンディションを 1 個だけ登録した場合は，そのレンディションがマスタレンディションになります。
- **サブレンディション**
  マスタレンディション以外のレンディションです。

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，レンディションと呼びます。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。

マルチレンディション文書の管理例を次の図に示します。

図 1-3　マルチレンディション文書の管理例



この図では，マルチレンディション文書のコンテントとして，同じ内容を表す複数の形式のファイルを登録しています。文書を編集するユーザは Word 形式のファイル，Word がインストールされていないマシンで文書を参照するユーザは HTML 形式のファイル，配布用の文書を取得したいユーザは PDF 形式のファイルをそれぞれダウンロードできます。このように，マルチレンディション文書では，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，適切なレンディションを指定して文書を利用できます。

また，マルチレンディション管理機能とアクセス制御機能を組み合わせて，マルチレンディション文書を管理できます。例えば，Word 形式のファイルと PDF 形式のファイルを登録して，文書を更新できるユーザには Word 形式のファイルを参照させて，文書の参照だけできるユーザには PDF 形式のファイルを参照させるというように，ユーザに設定するアクセス権によって，ユーザが参照できるコンテントを限定するという運用が考えられます。アクセス制御機能については，「1.2.13　アクセス制御機能」を参照してください。

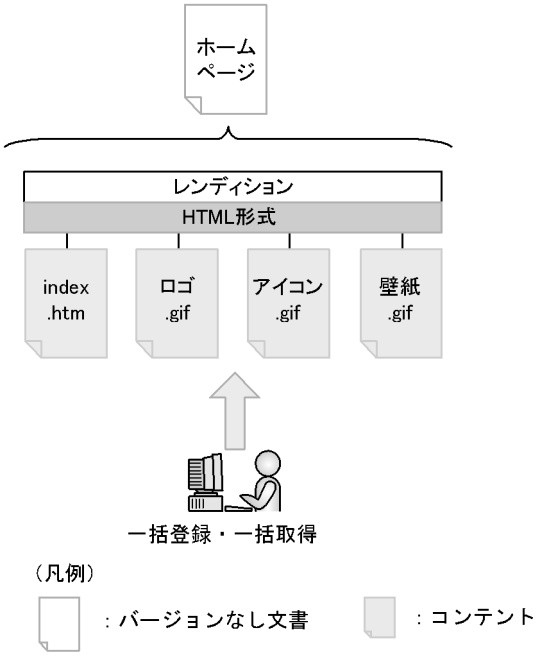
1.2.3　マルチファイル管理機能

マルチファイル管理機能では，一つの文書に複数のファイルを登録して管理します。一つの文書で複数のファイルを管理すると，文書を構成する複数のファイルを，一括して登録したり，一括して取得したりできます。

複数のファイルが登録された文書を，**マルチファイル文書**といいます。マルチファイル文書に対して，一つのファイルが登録された文書を，**シングルファイル文書**といいます。バージョンなし文書またはバージョン付き文書は，マルチファイル文書として管理できます。

マルチファイル文書の管理例を次の図に示します。

図 1-4　マルチファイル文書の管理例



この図では，ホームページのデータをマルチファイル文書として管理しています。このマルチファイル文書では，「index.htm」，「ロゴ .gif」，「アイコン .gif」および「壁紙 .gif」という四つのファイルを，HTML形式のレンディションのコンテントとして管理しています。

なお，マルチファイル管理機能を使用して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では，マルチレンディション管理機能を使用できません。

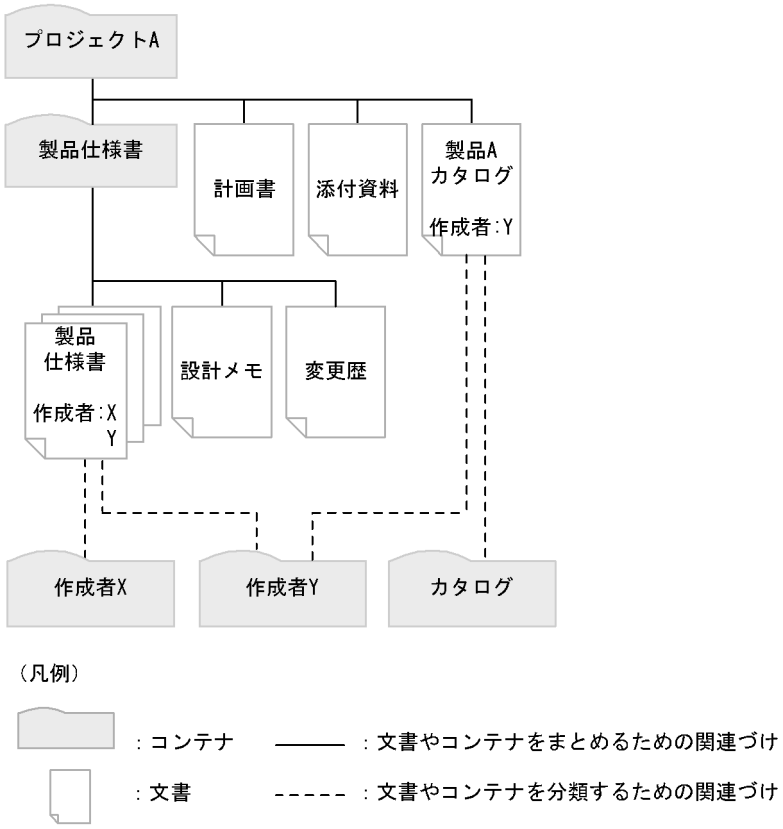
1.2.4　コンテナ管理機能

文書をまとめて格納するフォルダや，文書を分類するフォルダを利用して文書を管理できます。文書をまとめたり，分類したりするフォルダに相当するオブジェクトを，**コンテナ**といいます。

コンテナには，複数の文書またはコンテナを関連づけられます。コンテナを使用すると，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観点から分類して管理したりできます。また，コンテナとコンテナを関連づけることで，フォルダや分類に階層を持たせることもできます。

コンテナによる文書の管理例を次の図に示します。

図 1-5　コンテナによる文書の管理例



この図では，プロジェクト A というコンテナで，このプロジェクトに関連する文書をまとめて管理しています。「製品仕様書」は，製品仕様書，設計メモおよび変更歴で構成されます。したがって，「プロジェクト A」コンテナの下位に「製品仕様書」というコンテナを作成して，関連する文書をまとめて格納して管理しています。

また，「作成者 X」コンテナ，「作成者 Y」コンテナおよび「カタログ」コンテナは，文書を分類するためのコンテナです。「プロジェクト A に関連する文書」，「プロジェクト A の製品仕様書を構成する文書」というまとめ方とは別の視点で，文書をまとめています。したがって，製品 A カタログや製品仕様書がプロジェクト A に関連する文書であることを知らないユーザでも，文書を分類しているコンテナからたどることで，文書を見つけやすくなります。

1.2.5　文書の構成管理機能

実用書やマニュアルなどの出版物は，複数の章から構成されています。また，各章は，文字，図，表などから構成されています。出版物の章の構成が多くなって，各章を構成する文字，図，表などが多くなるほど，出版物の構成を保ちながら管理することが困難になっていきます。このような場合，DocumentBroker は，出版物の構成を保ったままコンテナで管理できます。

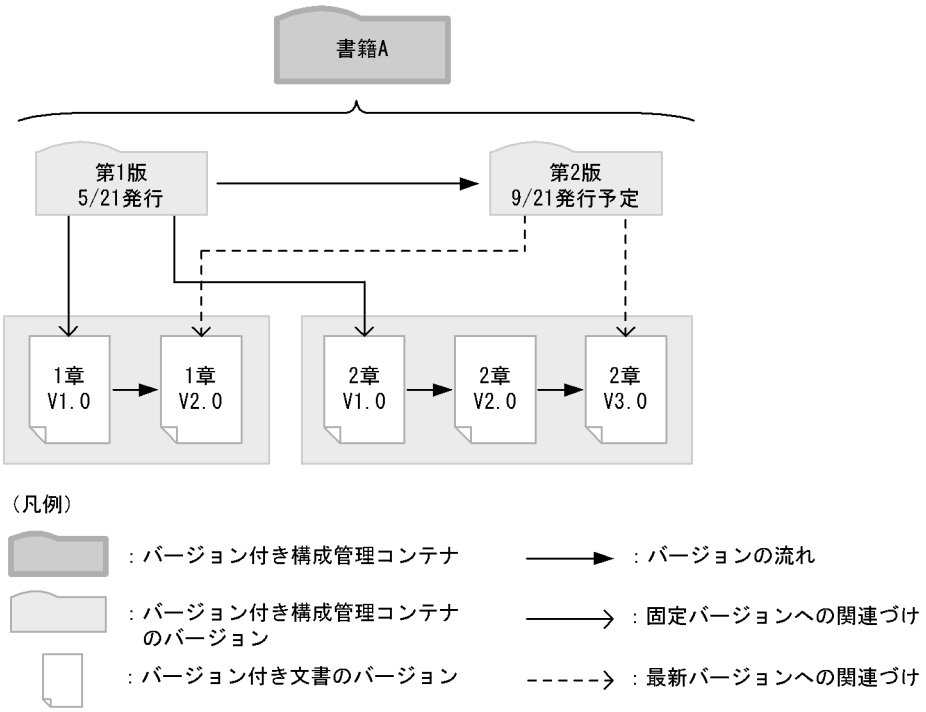
構成管理機能では，管理している文書やコンテナの，ある時点でのバージョン構成の状態を管理します。例えば，管理する複数の文書が，それぞれ異なるタイミングでバージョンアップする場合などに，構成管理機能を使用します。各文書のバージョン管理によって，常に最新の状態の文書を管理したり，執筆が完了した文書のバージョンを固定して管理したりできます。さらに，ある時点でのコンテナ内のすべての文書を固定しておきたい場合は，コンテナ自体のバージョンを上げることで，コンテナをバージョン管理できます。

構成を管理するコンテナを，**構成管理コンテナ**といいます。これに対して，構成を管理しないコンテナを，**バージョンなしコンテナ**といいます。

構成管理コンテナは，それ自身のバージョンを保持する**バージョン付き構成管理コンテナ**と，バージョンを保持しない**バージョンなし構成管理コンテナ**に分けられます。

バージョン付き構成管理コンテナによる文書の管理例を次の図に示します。

図 1-6　バージョン付き構成管理コンテナによる文書の管理例



この図では，二つの章で構成される文書の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「書籍 A」というバージョン付き構成管理コンテナにまとめて関連づけて，発行時期で管理しています。

バージョン付き構成管理コンテナ「第 1 版」では，5 月 21 日に発行した書籍 A の構成を管理しているため，1 章，2 章共に V1.0 の状態（5 月 21 日に発行した時点での構成）で固定して管理しています。一方，改訂作業のために，この書籍を構成する文書（1 章および 2 章）を編集する場合は，毎日の作業履歴として文書のバージョンを管理しながら，改訂作業が終了するまで常に最新バージョンを一括して管理する必要があります。したがって，バージョン付き構成管理コンテナをバージョンアップして（第 2 版），常に最新のバージョンを参照するように設定します。

なお，バージョンなし構成管理コンテナは，それ自身のバージョンを保持しないで，ある時点での文書全体の構成だけを管理します。したがって，構成管理コンテナで管理されている文書全体を版（バージョン）として管理しない場合は，バージョンなし構成管理コンテナを利用できます。

1.2.6　文書間リレーション管理機能

文書と文書を関連づけて管理する機能を，文書間リレーションといいます。

文書間リレーションは，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

DocumentBroker は，二つの文書を**文書間リレーション**という概念を利用して関連づけます。この場合，文書間リレーションを設定した文書を**リレーション元文書**，文書間リレーションを設定された文書を**リレーション先文書**といいます。

一つの文書間リレーションでは，一つのリレーション元文書と一つのリレーション先文書を関連づけられます。また，一つの文書に対して，複数の文書間リレーションを設定できます。一つの文書に対して複数の文書から文書間リレーションを設定することもできます。

1.2.7 文書の属性情報の管理機能

文書やコンテナなどのオブジェクト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。プロパティには，DocumentBroker によってあらかじめ定義されているプロパティ（システムプロパティ）と，ユーザが任意に追加定義するプロパティ（ユーザプロパティ）があります。例えば，文書を管理する場合，ユーザプロパティとして，「文書名」や「作成日時」などの標準的な属性情報だけでなく，「顧客名」や「競合他社名」などの業務に応じた属性情報も一緒に管理できます。

文書やコンテナにプロパティを定義すると，プロパティをキーにしてオブジェクトを検索したり，プロパティの値を参照してオブジェクトの状態を確認したりできます。

1.2.8 リファレンスファイル管理機能

リファレンスファイル管理機能では，文書の実体であるコンテントを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トの格納先の情報（**コンテントロケーション**）をデータベースに登録して管理します。

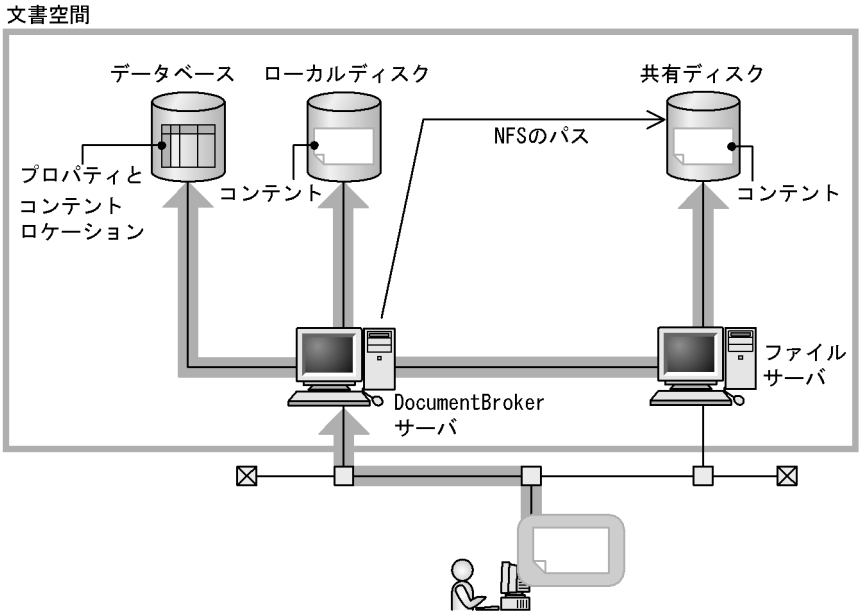
ファイルシステムには，DocumentBroker サーバが存在するマシン上のローカルディスク，またはネットワークファイルシステム（NFS）を利用してアクセス可能な共有ディスクを使用してください。

コンテントロケーションには，コンテントの格納先の基点となるディレクトリパス（**コンテント格納先ベースパス**）からの相対パスが登録されます。このため，コンテント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，コンテント格納先ベースパスを変更するだけで，コンテントの格納領域を変更できます。また，コンテント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。

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。バージョンなし文書およびバージョン付き文書は，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 1-7 リファレンスファイル文書の管理例



この図では，ファイルシステムとして，DocumentBroker サーバが存在するマシンのローカルディスクを利用してリファレンスファイル文書を管理する場合と，ファイルサーバの共有ディスクを利用してリファレンスファイル文書を管理する場合の例を示しています。ファイルシステムにコンテントを格納し，データベースにプロパティとコンテントロケーションを登録して管理しています。このように，リファレンスファイル管理機能では，データベースにコンテントを格納しないので，コンテントの格納先を選択したり，コンテントの格納先のディスクを移行したりと，柔軟にコンテントの格納先を管理できます。

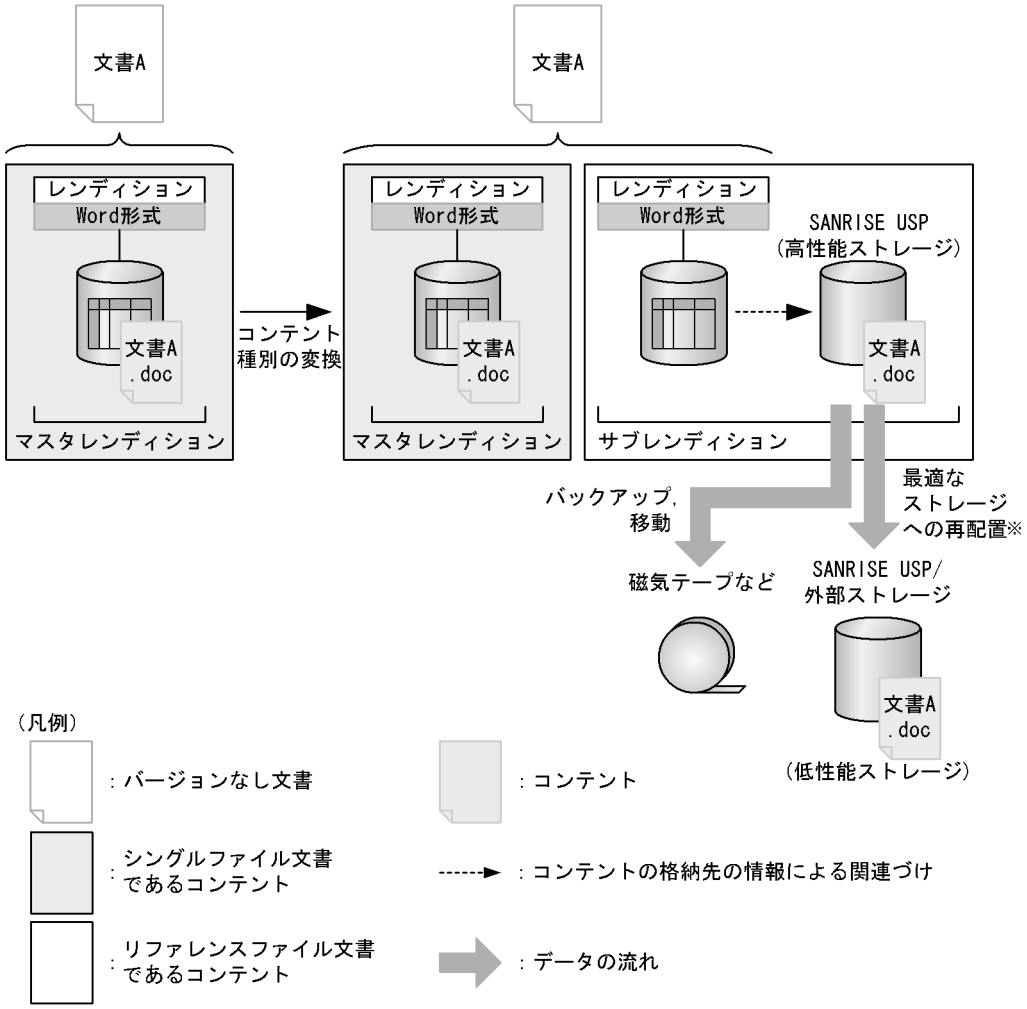
1.2.9 レンディションのコンテント種別変換機能

レンディションのコンテント種別変換機能は，レンディションのコンテントの格納先（コンテント種別）を，最適な格納先へ変換する機能です。

DocumentBroker では，文書の実体であるコンテントをデータベースや DocumentBroker サーバマシンが接続可能なファイルシステムの任意のディレクトリなどに格納して管理できます。これらの格納先で管理されるコンテントを，変化するデータの利用価値に応じて，最適な格納先へ変換できます。

コンテント種別を，シングルファイル文書（データベースに登録されている一つのファイルをコンテントとして持つ文書）から，リファレンスファイル文書（DocumentBroker サーバ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トとして持つ文書）に変換したコンテントは，JP1/HiCommand Tiered Storage Manager を利用した最適なストレージへの再配置（マイグレーション）や，磁気テープなどのストレージへのバックアップや移動など，コンテント管理の柔軟な運用を行うことができます。レンディションのコンテント種別変換機能を利用した運用例を次に示します。

図 1-8　レンディションのコンテント種別変換機能を利用した運用例



注※

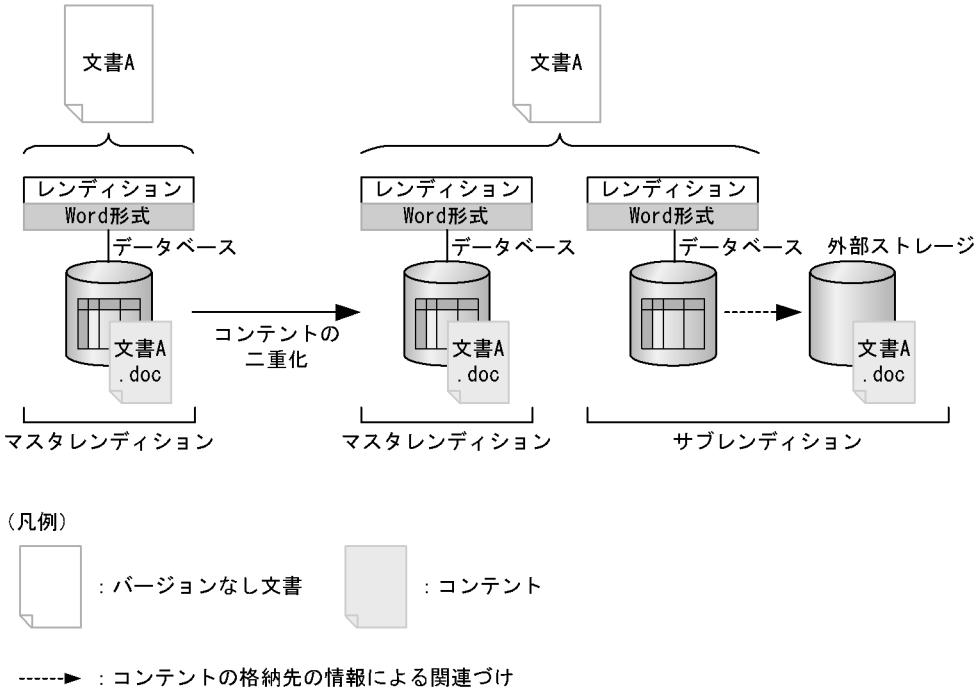
JP1/HiCommand Tiered Storage Managerを利用することもできます。
JP1/HiCommand Tiered Storage Managerは，コンテントを最適なストレージへ再配置するための製品です。

この例では，レンディションのコンテント種別変換機能によって，データベースで管理するコンテントをDocumentBroker サーバマシンが接続可能なファイルシステムの任意のディレクトリ上に変換して格納しています。

- 磁気テープなどへのバックアップや移動
  ユーザの操作によって，コンテントを磁気テープなどのストレージへバックアップや移動して管理します。
- 最適なストレージへの再配置
  JP1/HiCommand Tiered Storage Manager でコンテントの格納先のストレージを最適なストレージへ再配置します。JP1/HiCommand Tiered Storage Manager の使用方法については，マニュアル「JP1/HiCommand Tiered Storage Manager　ユーザーズガイド」を参照してください。

レンディションのコンテント種別変換機能を使用してコンテント種別を変換する例を次の図に示します。

図 1-9　レンディションのコンテント種別変換の例



コンテント種別の変換は，文書のマルチレンディション管理機能を使用して，変換後のコンテントをサブレンディションとして追加します。例では，シングルファイル文書のコンテントを，リファレンスファイル文書のコンテントに変換し，レンディションを追加します。これをコンテントの二重化と呼びます。コンテントを二重化する場合，追加するレンディションのレンディションタイプにコメントを付与して格納先を分類します。

なお，コンテントを二重化したあとにレンディションを操作する場合は，マルチレンディション管理機能を使用します。

(1) 変換できるコンテント種別

レンディションのコンテント種別変換機能で変換できるコンテント種別を次の表に示します。

表 1-2　レンディションのコンテント種別変換機能で変換できるコンテント種別

変換前のコンテント種別	変換後のコンテント種別
シングルファイル文書	リファレンスファイル文書
リファレンスファイル文書	シングルファイル文書

レンディションのコンテント種別は，レンディションのコンテント種別を表すプロパティを参照することで判別できます。プロパティの詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」を参照してください。

(2) 注意事項

コンテント種別を変換して，サブレンディションとして追加するときには，追加するレンディションに対してレンディションコメントを付与します。コンテントの変換前，または変換後のどちらかのレンディションに対してコメントを付与できますが，変換したコンテントであることが明確になるように変換後のコンテントに対して，コンテント種別を表すコメントを付けることをお勧めします。

ただし，DocumentBroker のユティリティ製品を使用した運用で，変換後のコンテントをマスタレンディションとする場合などでは，変換前のレンディションに対してコメントを付与できます。

レンディションのコンテント種別変換機能を使用する場合に，各 DocumentBroker のユティリティ製品を使用するときの注意事項を次に示します。

- DocumentBroker Object Loader
  レンディションタイプを完全一致文字列で扱います。このため，エクスポート機能を使用する場合，付与するコメントの規則性を明確にしてください。ただし，DocumentBroker Object Loader では，リファレンスファイル文書のエクスポートはできません。
- DocumentBroker Text Search Index Loader
  レンディションタイプを前方一致で扱います。レンディションのコンテント種別変換機能を使用した運用では，レンディション定義ファイル中で該当するレンディションタイプを前方一致で定義します。ただし，DocumentBroker Text Search Index Loader では，リファレンスファイル文書のテキスト抽出はできません。
- DocumentBroker Rendering Option
  レンディションタイプを完全一致文字列で扱います。このため，付与するコメントの規則性を明確にしてください。マスタレンディションのコンテント種別を変換して，変換後のレンディションをマスタレンディションとしてコメントを付与する場合は，レンディション変換プラグイン定義ファイルで，コメント付きのレンディションを定義します。ただし，DocumentBroker Rendering Option でのリファレンスファイル文書のレンディション変換はできません。レンディション変換プラグイン定義ファイルの詳細については，マニュアル「DocumentBroker Rendering Option」を参照してください。

1.2.10 File Link 連携機能

File Link 連携機能は，HiRDB File Link と連携して文書を管理する機能です。なお，次に示すどちらかの条件を満たす環境下では，File Link 連携機能を使用できません。

● Linux を使用している場合

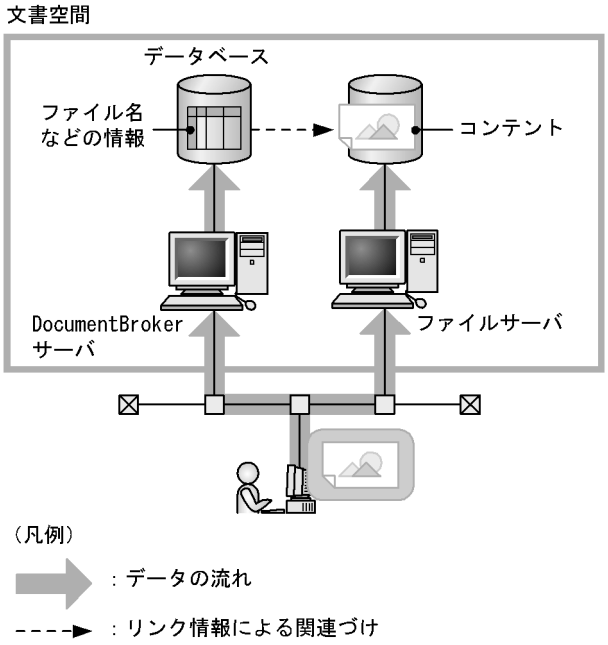
● TPBroker V5 を使用している場合

画像・音声などの容量が大きいコンテントをファイルサーバへ格納し，格納先の情報（ファイルの作成者，ファイル名などのリンク情報）をデータベースで管理する場合にこの機能を使用すると，データベースで管理している情報とファイルサーバ上のファイルとの整合性を確保できます。このため，クライアントアプリケーションで，データベースとファイルサーバの整合性を取るための処理をする必要はありません。

File Link 連携機能を使用して管理している文書を，**File Link 文書**といいます。バージョンなし文書またはバージョン付き文書は，File Link 文書として管理できます。

File Link 文書の管理例を次の図に示します。

図 1-10　File Link 文書の管理例



HiRDB File Link の詳細については，マニュアル「HiRDB File Link」を参照してください。

1.2.11　XML 文書管理機能

XML は，W3C によって標準化が進められている，文書の構造を定義するための言語です。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，Web 上のデータ交換フォーマットなどとして，多くの業務で適用されています。

DocumentBroker は，XML 形式で記述されたファイルを登録および操作できます。通常，XML を扱うシステムには，XML 形式で記述されたファイルを解析し，そのデータを処理するアプリケーションが必要となります。このアプリケーションは，ファイルの入力と解析や，抽出したデータのデータベースへの格納などの処理を実行します。なお，Linux の場合，XML 文書管理機能は使用できません。

DocumentBroker は，次のプログラムを利用して XML 文書管理機能を実現します。

- HiRDB Adapter for XML
- Preprocessing Library for Text Search
- HiRDB Text Search Plug-in

管理できる XML 文書，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次に説明します。

なお，このマニュアルでは，XML で記述されたファイルを，XML **ファイル**と呼びます。また，XML ファイルをコンテントとして DocumentBroker に登録した文書を，XML **文書**と呼びます。

(1) 管理できる XML 文書

一つのファイルの XML 文書を，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。また，XML から参照されるイメージファイルなど，複数ファイルから構成される文書群については，バージョンなしコンテナまたは構成管理コンテナを利用して管理できます。

なお，XML 文書管理機能で管理する XML 文書は，登録文書のままであり，文字コードの変換などは実行しません。

(2) 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は，XML 文書を登録するときに，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 文書のプロパティとしてマッピングする機能です。なお，XML プロパティマッピング機能を使用するためには，XML ファイルから抽出するタグの情報とマッピング先のプロパティとの対応を定義したファイルをユーザが作成する必要があります。

(3) 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は，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，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。なお，XML インデクスデータ作成機能を使用するためには，削除したい不要なタグやタグ間の文字列を定義したファイルをユーザが作成する必要があります。

1.2.12 独立データの管理機能

文書やコンテナなどのほかのオブジェクトの体系に関係しない，独立したデータを管理するオブジェクトのことを**独立データ**といいます。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。

独立データは，文書のようにまとめたり，コンテナに関連づけたり，バージョンを管理したり，コンテナのように上位・下位の階層構造で管理したりできません。

1.2.13 アクセス制御機能

DocumentBroker では，DocumentBroker サーバが管理するメモリ空間（文書空間）にユーザがログインするときに生成されるユーザ情報や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。文書空間にログインするときのユーザ認証，オブジェクトへのアクセス制御は，ユーザ管理システムとの連携によって実現します。

(1) ユーザ認証

DocumentBroker では，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と連携したユーザ認証機能を提供しています。これによって，システムへのユーザの不当なログインを制御できます。

文書空間へのログイン要求があった場合，DocumentBroker は，ユーザから提示されたユーザ ID とパスワードに対して，ユーザ管理システムの機能を利用してユーザを認証します。このとき，ログインしたユーザに関する情報を取得して，ユーザ情報を生成します。

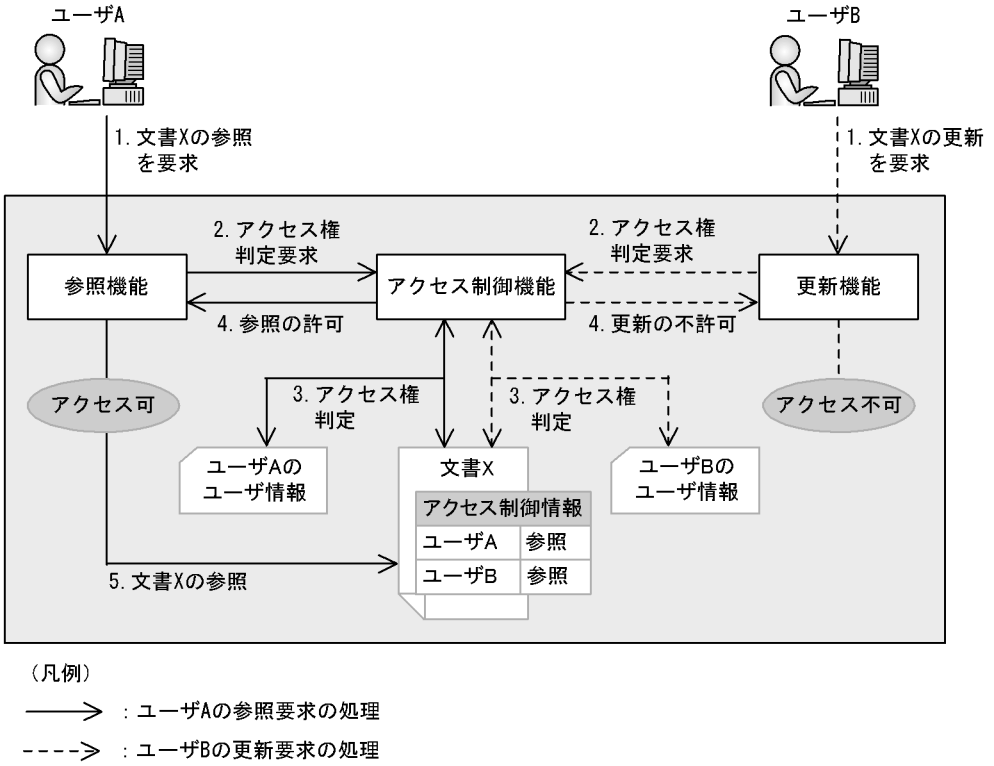
(2) オブジェクトに対するアクセス制御

DocumentBroker で管理しているオブジェクトに対して，アクセス権を設定できます。例えば，文書に対してアクセス権を設定すると，アクセスを許可されていないユーザが誤って文書を更新してしまうようなことがありません。また，特定の人やグループだけに文書の参照や編集を許可するような運用もできます。

DocumentBroker が提供するアクセス制御機能は，ユーザやグループに対してアクセス権を与えたり，アクセス権を変更したりできます。例えば，文書の作成，審査，承認といった作業行程に応じてアクセス権を設定できます。これによって，文書のライフサイクル全体を管理できます。

文書にアクセスする場合の，アクセス制御の概要を次の図に示します。

図 1-11　アクセス制御の概要



DocumentBroker のアクセス制御は，アクセス制御対象オブジェクトへアクセスする場合，次に示す二つの情報をアクセス制御機能によって判定し，アクセス制御を実行します。

● ログインしたユーザのユーザ情報

● アクセス制御対象オブジェクトのアクセス制御情報

文書の操作を要求した場合，操作要求の内容とユーザ情報を比較してアクセス権を判定します。その結果，アクセス権があるときは，要求された文書の操作を実行します。しかし，アクセス権がないときは，「アクセス権なし」というエラーが返却されます。図 1-11 では，ユーザ A が文書の参照を要求しています。DocumentBroker は，ユーザ A のログインによって作成されたユーザ情報と文書 X に設定されているアクセス制御情報とを比較します。その結果，ユーザ A に参照権があると判定されて，ユーザ A は文書 X を参照できます。しかし，ユーザ B が文書 X の更新を要求した場合，DocumentBroker は，アクセス権なしのエラーを返却しています。これは，ユーザ B には，文書 X を参照する権利だけが与えられているためです。

なお，LDAP 対応のディレクトリサービスと連携してアクセス制御機能を使用する場合，アクセス制御機能で使用するユーザ情報は，LDAP 対応のディレクトリサービスに管理されている情報を基に生成されます。ただし，運用中のユーザ管理システムが DocumentBroker のアクセス制御機能に適応していない場合，ユーザが作成したアクセスルーチンを組み込むことによって，そのユーザ管理システムを DocumentBroker のアクセス制御機能として使用できます。

(3) アクセスログ機能

アクセスログ機能では，オブジェクトに対するユーザの操作履歴をログとしてファイルに出力します。アクセスログファイルには，アクセスした日時，アクセスしたユーザのユーザ ID，操作内容，アクセス対象のオブジェクトの情報などが出力されます。

1.2.14 アクセスログおよびトレースログの統計・解析機能

DocumentBroker が出力したアクセスログファイルおよびトレースファイルの内容を解析・分類・集計して，レポートを出力する機能です。このレポートを参照することで，DocumentBroker の文書管理システムへのユーザのアクセス状況や DocumentBroker サーバおよびクライアントアプリケーションごとの処理内容を把握できます。この機能を利用する場合は，DocumentBroker サーバの一機能である統計解析ツールを使用します。統計解析ツールの詳細については，マニュアル「DocumentBroker Version 3　統計解析ツール」を参照してください。

1.2.15 検索機能

DocumentBroker で使用できる検索機能について説明します。

(1) 属性検索機能

オブジェクトに設定されているプロパティの値を条件にした検索を実行できる，属性検索機能を提供します。プロパティの値を基に，文書，フォルダ，インデクスに相当するオブジェクトなどが検索できます。例えば，文書名と著者がプロパティとして設定されている場合に，「文書名が『報告書』であり，著者が『日立太郎』である文書を検索する」というような検索ができます。

(2) 文書に対する全文検索機能

文書の内容に含まれるキーワード（**検索ターム**）を条件にした検索を実行できる，全文検索機能を提供します。

DocumentBroker は，次のプログラムを利用して全文検索機能を実現します。

● HiRDB Text Search Plug-in

さらに，次のプログラムを利用することによって，HiRDB Text Search Plug-in の機能を拡張した検索機能を実現します。

● Preprocessing Library for Text Search（構造指定検索の場合）

● HiRDB Text Search Plug-in Conceptual Extension（概念検索の場合）

全文検索機能では，文書中の任意の検索タームを条件にできます。例えば，「テキスト内に『文書』と『XML』という単語が含まれる文書を検索する」という場合に使用できます。

次に，全文検索機能の基本機能と拡張機能について説明します。なお，このマニュアルでは，これらの検索機能を総称して，全文検索といいます。それぞれの検索に固有の説明をする場合は，それぞれの機能名で説明します。

基本機能

HiRDB Text Search Plug-in の機能を使用した検索機能です。

- **同義語・異表記展開検索**
  検索タームとして「コンピュータ」を指定した場合，「PC」や「パソコン」といった検索タームの同義語に相当する単語を含む文書も検索できます（**同義語展開検索**）。
  アルファベットの大文字と小文字，かたかなの異表記および半角と全角の文字列を展開して，検索できます。例えば，「ski」を検索タームに指定する場合，「ski」，「SKI」，「Ski」などで表記されている文書も検索できます（**異表記展開検索**）。
- **近傍条件検索**
  二つの検索タームについて，それらの距離や出現順序関係を指定した検索ができます。例えば，「『最新』と『情報』の間の文字数が 20 文字ちょうどの文書を検索する。ただし，『最新』と『情

報』はどちらが先に出現してもよい」という指定ができます。

- **ランキング検索**
  指定する検索タームに重み（重要度）を付けてスコアを算出し，このスコアを基に検索結果をソートできます。

拡張機能

ほかのプログラムの機能によって HiRDB Text Search Plug-in の機能を拡張した検索機能です。

- **構造指定検索**
  XML 文書に対する構造指定全文検索ができます。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できます。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。
  なお，XML 文書の全文検索を実行するためには，Preprocessing Library for Text Search を利用して生成したインデクスデータを，XML 文書の全文検索インデクスとして登録する必要があります。
  なお，Linux の場合，XML 文書に対する構造指定検索機能を使用できません。
- **概念検索**
  全文検索では，指定する検索タームに適した具体的な単語が思いつかない場合に，検索したい文書に類似した概念を持つ文章（種文章）を検索条件とした検索もできます。これを，概念検索といいます。概念検索では，検索条件として指定した文書からその文書を特徴づける単語（特徴ターム）が抽出され，それを検索タームとした検索が実行されます。例えば，「『…リサイクルは，資源の有効活用に貢献するだけでなく，ゴミの減量化にも役立ちます…』という文章に近い概念の文書を検索する」という場合に使用できます。概念検索機能を使用するためには，HiRDB Text Search Plug-in の機能のほかに，HiRDB Text Search Plug-in Conceptual Extension が必要です。

（3）文字列型プロパティに対する全文検索機能

オブジェクトに設定されている文字列型プロパティの値に含まれるキーワード（**検索ターム**）を条件にした検索を実行できる，全文検索機能を提供します。全文検索機能を使用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。

なお，データベース定義の名称定義の方法が GUID 値を変換した値を使用する方法の場合，文字列型プロパティに対する全文検索機能を使用できません。データベース定義の名称については，「3.11.1　データベース定義の名称の検討」を参照してください。

DocumentBroker は，次のプログラムを利用して文字列型プロパティに対する全文検索機能を実現します。

● HiRDB Text Search Plug-in

HiRDB Text Search Plug-in を利用すると，次に示す機能を使用できます。

- **同義語・異表記展開検索**
  検索タームとして「コンピュータ」を指定した場合，「PC」や「パソコン」といった検索タームの同義語に相当する単語をプロパティの値に含む文書も検索できます（**同義語展開検索**）。
  アルファベットの大文字と小文字，かたかなの異表記および半角と全角の文字列を展開して，検索できます。例えば，「ski」を検索タームに指定する場合，「ski」，「SKI」，「Ski」などで表記されている文字列をプロパティの値に含む文書も検索できます（**異表記展開検索**）。
- **近傍条件検索**
  二つの検索タームについて，それらの距離や出現順序関係を指定した検索ができます。例えば，「『最新』と『情報』の間の文字数が 20 文字ちょうどの値をプロパティに含む文書を検索する。ただし，『最新』と『情報』はどちらが先に出現してもよい」という指定ができます。

文字列型プロパティに対する全文検索機能に対応した C++ クラスライブラリのクラ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　C++　解説」を，Java クラスライブラリの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　解説」を参照してください。

(4) 検索条件の論理演算機能

検索条件同士の論理積や論理和などを求める機能も提供します。

AND 検索

検索条件同士の論理積を求められます。例えば，「著者が『日立太郎』で，文書中に『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。

OR 検索

検索条件同士の論理和を求めることができます。例えば，「作成者が『日立太郎』である文書，または作成者の所属が『日立製作所』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。

NOT 検索

検索条件との不一致を求めることができます。例えば，「作成者が『日立太郎』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。

1.2.16 オブジェクト一括登録機能

大量のオブジェクトを DocumentBroker に一括して登録する機能です。この機能を利用する場合は，DocumentBroker Object Loader を使用します。DocumentBroker Object Loader では，DocumentBroker からオブジェクトを取得してほかの DocumentBroker へ一括して登録する場合に，DocumentBroker Object Loader の実行に必要な入力データファイルを生成する機能も提供しています。さらに，DocumentBroker Object Loader High-end Option を使用すると，複数の DocumentBroker Object Loader を同時に実行して，オブジェクトの登録時間を短縮できます。

DocumentBroker Object Loader および DocumentBroker Object Loader High-end Option の詳細については，マニュアル「DocumentBroker Object Loader Version 3」を参照してください。

1.2.17 全文検索インデクス一括登録機能

DocumentBroker に登録されている文書からテキスト情報を抽出し，全文検索インデクスとして一括登録する機能です。この機能を利用する場合は，DocumentBroker Text Search Index Loader を使用します。DocumentBroker Text Search Index Loader の詳細については，マニュアル「DocumentBroker Text Search Index Loader Version 3」を参照してください。

なお，マルチファイル文書，リファレンスファイル文書および File Link 文書の場合，全文検索インデクス一括登録機能は使用できません。

1.2.18 レンディション変換機能

マルチレンディション管理機能を利用して文書を管理する場合に，バッチ処理でレンディション変換を実行する機能です。次のような場合に，登録済みのマスタレンディションのファイルを基に，別のレンディションタイプのファイルを作成して，サブレンディションのコンテントとして登録すると便利です。

- マスタレンディションだけにファイルが登録されている場合
- サブレンディションのファイルの登録後にマスタレンディションのファイルが更新された場合

この機能を利用する場合は，DocumentBroker Rendering Option を使用します。DocumentBroker Rendering Option にレンディション変換を要求する機能のことを，**レンディション変換要求機能**といいます。レンディション変換要求機能を利用すると，複数のレンディションのコンテントの同期が取りやすくなります。

DocumentBroker Rendering Option の詳細については，マニュアル「DocumentBroker Rendering Option」を参照してください。

1.2.19 イメージ文書の登録機能

デジタル複合機でスキャンした紙の文書や受信した FAX を，DocumentBroker のイメージ文書として登録する機能です。この機能を利用する場合，次のプログラムの中から，ご使用のデジタル複合機に応じたプログラムを使用してください。

- DocumentBroker イメージ登録 for DocuCentre （DocuCentre 対応）
- DocumentBroker イメージ登録 for iR （imageRUNNER 対応）
- DocumentBroker イメージ登録 for imagio and IPSiO （imagio および IPSiO 対応）
- DocumentBroker イメージ登録 for Konica （Sitios 対応）

さらに，次のオプションプログラムを使用することで，DocumentBroker にイメージ文書を登録する際に，文書に記載された情報を文字認識して，文書のプロパティとして登録したり帳票ごとに分割して文書を登録したりできます。

- DocumentBroker イメージ登録 文字認識オプション

イメージ文書の登録機能の詳細については，マニュアル「DocumentBroker イメージ登録 for DocuCentre」，マニュアル「DocumentBroker イメージ登録 for iR」，マニュアル「DocumentBroker イメージ登録 for imagio and IPSiO」，またはマニュアル「DocumentBroker イメージ登録 for Konica」を参照してください。

1.2.20 分散した文書の収集・検索機能

ネットワーク上に散在した文書を収集して DocumentBroker に登録し，WWW ブラウザから文書を検索できる機能です。この機能を利用する場合は，DocumentBroker Collector と，DocumentBroker Integrated Search Suite for J または DocumentBroker Integrated Search Suite を使用します。これらの製品を使用した検索では，DocumentBroker の検索機能を使用できます。

DocumentBroker Collector は，インターネット，イントラネット，ファイルシステムやグループウェアなどに分散しているさまざまなフォーマットの文書から属性を収集し，XML 形式に変換して DocumentBroker に登録して管理できます。DocumentBroker Collector の詳細については，マニュアル「DocumentBroker Collector」を参照してください。

WWW ブラウザから DocumentBroker Integrated Search Suite for J，または DocumentBroker Integrated Search Suite が提供する検索テンプレートを使用することで，DocumentBroker Collector で収集および登録した文書を検索できます。DocumentBroker Integrated Search Suite for J の詳細については，マニュアル「DocumentBroker Integrated Search Suite for J Version 2」を参照してください。DocumentBroker Integrated Search Suite の詳細については，マニュアル「DocumentBroker Integrated Search Suite」を参照してください。

1.2.21 文書の分析機能

DocumentBroker で管理されている文書のテキスト情報と属性情報を分析する機能です。膨大な量の文書を分析して，その傾向や特徴を発見することで，情報の活用，新たな価値ある情報の創造を支援します。

分析対象となるデータを生成するためには，DocumentBroker テキスト分析コンポーネントを使用します。DocumentBroker テキスト分析コンポーネントの詳細については，マニュアル「DocumentBroker テキスト分析コンポーネント」を参照してください。

WWW ブラウザから DocumentBroker テキスト分析テンプレートが提供する分析テンプレートを使用することで，DocumentBroker テキスト分析コンポーネントで生成したデータを分析できます。DocumentBroker テキスト分析テンプレートの詳細については，マニュアル「DocumentBroker テキスト分析テンプレート」を参照してください。

1.2.22 文書のライフサイクル管理機能

文書管理システムには，文書の保管機能や検索機能に加えて，作成から破棄までの文書のライフサイクル管理に対応した機能が必要な場合があります。例えば，作成／審査／承認／公開／廃棄といった文書のライフサイクルと業務の流れが密接に関連する場合の文書管理システムには，文書の状態に応じて異なる情報を作業者に示す必要があります。

業務の流れと文書のライフサイクルを関連させて管理する文書管理システムを構築する場合，DocumentBroker Life Cycle Suite を使用します。DocumentBroker Life Cycle Suite は，業務の流れと文書のライフサイクルを関連させて業務の要件に適した基幹システムを構築できるよう，ワーク管理システムである WorkCoordinator と連携するための機能を提供しています。

また，DocumentBroker Life Cycle Suite が提供する文書ライフサイクル管理システムの構築を支援する各種のアプリケーションフレームワークを使用すると，システム構築期間の短縮，および開発効率の向上を実現できます。なお，DocumentBroker Life Cycle Suite の詳細については，マニュアル「DocumentBroker Life Cycle Suite Version 2」を参照してください。

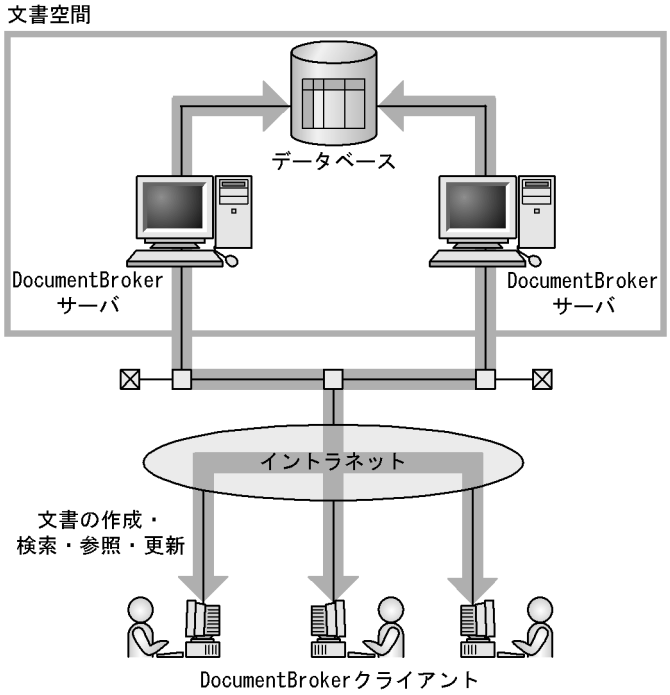
1.2.23 ファイル転送機能

DocumentBroker サーバと DocumentBroker クライアントを別のマシンで運用する場合，サーバとクライアント間のデータ転送には，ファイル転送機能を使用します。例えば，DocumentBroker で管理している文書のファイルをクライアントのマシンに取得したり，クライアントのマシン上にあるファイルを DocumentBroker の文書として登録したりする場合に，ファイル転送が必要になります。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。

1.2.24 複数の実行環境機能

使用ユーザ数や単位時間当たりのトランザクション数の増加などによるシステム負荷を軽減するために，一つのデータベースに複数の DocumentBroker サーバの実行環境を配置したシステム構成を構築できます。これによって，システム負荷が複数のサーバに分散されるため，システム全体の処理能力が向上します。複数の DocumentBroker サーバの実行環境を配置したシステム構成を次の図に示します。

図 1-12　複数の DocumentBroker サーバの実行環境を配置したシステム構成



実行環境を識別する**実行環境識別子**を使用して，文書を管理できます。これによって，複数の実行環境から，同じ文書空間にアクセスする運用形態を構築できます。

1.2.25　システム導入支援機能

システム導入支援機能は，新規に文書空間を構築するための設定を支援します。

システム導入支援機能を使用すると，システム導入支援機能を使用しないで文書空間を構築する場合に比べて，DocumentBroker でデータベースシステムを使用するための設定手順を簡易化できます。

システム導入支援機能を使用する場合の，データベースシステムを使用するための設定の詳細は，「3.12 システム導入支援機能での設定」を参照してください。

1.2.26　クライアントアプリケーション動作定義機能

クライアントアプリケーション動作定義機能を使用すると，あらかじめ定義しておいた動作環境でクライアントアプリケーションを動作させることができます。

クライアントアプリケーションの動作環境は，クライアントアプリケーション動作定義ファイルに定義します。クライアントアプリケーション動作定義ファイルの詳細については，「4.18　クライアントアプリケーション動作定義ファイル（application.ini）」を参照してください。

1.3 DocumentBroker のシステム構成

この節では，DocumentBroker のシステム構成について説明します。

1.3.1 DocumentBroker を使用したシステムの概略

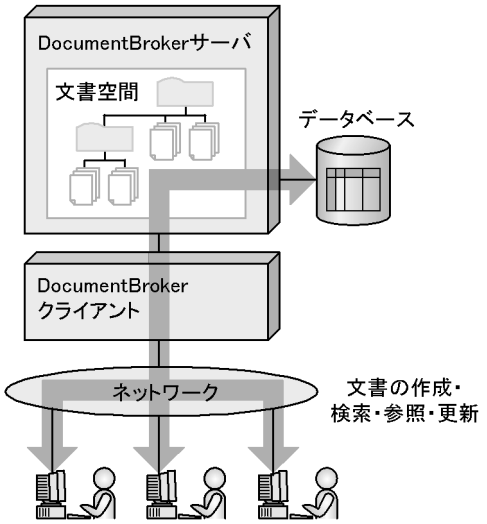
ここでは，DocumentBroker を使用したシステムの概略について説明します。

DocumentBroker は，分散オブジェクト環境での文書管理基盤フレームワークとして，文書管理に関するさまざまな機能を提供します。これによって，用途に合わせた各種業務システムを構築できます。また，ORB による既存システムと連携した業務システムを構築できます。

(1) DocumentBroker のシステムの構成

DocumentBroker で構築する文書管理システムは，DocumentBroker サーバと DocumentBroker クライアントで構成されます。DocumentBroker のシステムの構成を次の図に示します。

図 1-13 DocumentBroker のシステムの構成



DocumentBroker サーバは，文書の管理機能の実現に必要なオブジェクトと，DocumentBroker のメモリ空間である文書空間を提供します。DocumentBroker クライアントは，クライアントアプリケーションの開発環境と，作成したクライアントアプリケーションの実行環境を提供します。DocumentBroker クライアントでは，業務に対応したクライアントアプリケーションを構築し，それを実行することで DocumentBroker サーバの機能を利用します。クライアントアプリケーションを実行し，DocumentBroker サーバの文書空間上でデータベースに格納されているオブジェクトを操作することで，文書の参照や更新などの文書管理が実現します。DocumentBroker サーバと DocumentBroker クライアント間のオブジェクトのやり取りは CORBA 通信によって実現します。また，システムの運用形態に応じて使用する各種の関連プログラムも提供しています。

(2) DocumentBroker クライアントのインターフェース

クライアントアプリケーションの開発には，DocumentBroker Development Kit を使用します。

DocumentBroker Development Kit は，C++ 言語と Java 言語に対応したクライアントアプリケーション開発用のクラスライブラリ，およびクライアントの実行環境を提供しています。

C++ 言語対応のクラスライブラリ（C++ **クラスライブラリ**）では，C++ 環境で動作するクライアントアプリケーションを開発・実行できます。DocumentBroker で提供する機能のうち，マルチファイル管理機能，FileLink 連携機能は，C++ クラスライブラリで開発したクライアントアプリケーションでしか使用できません。

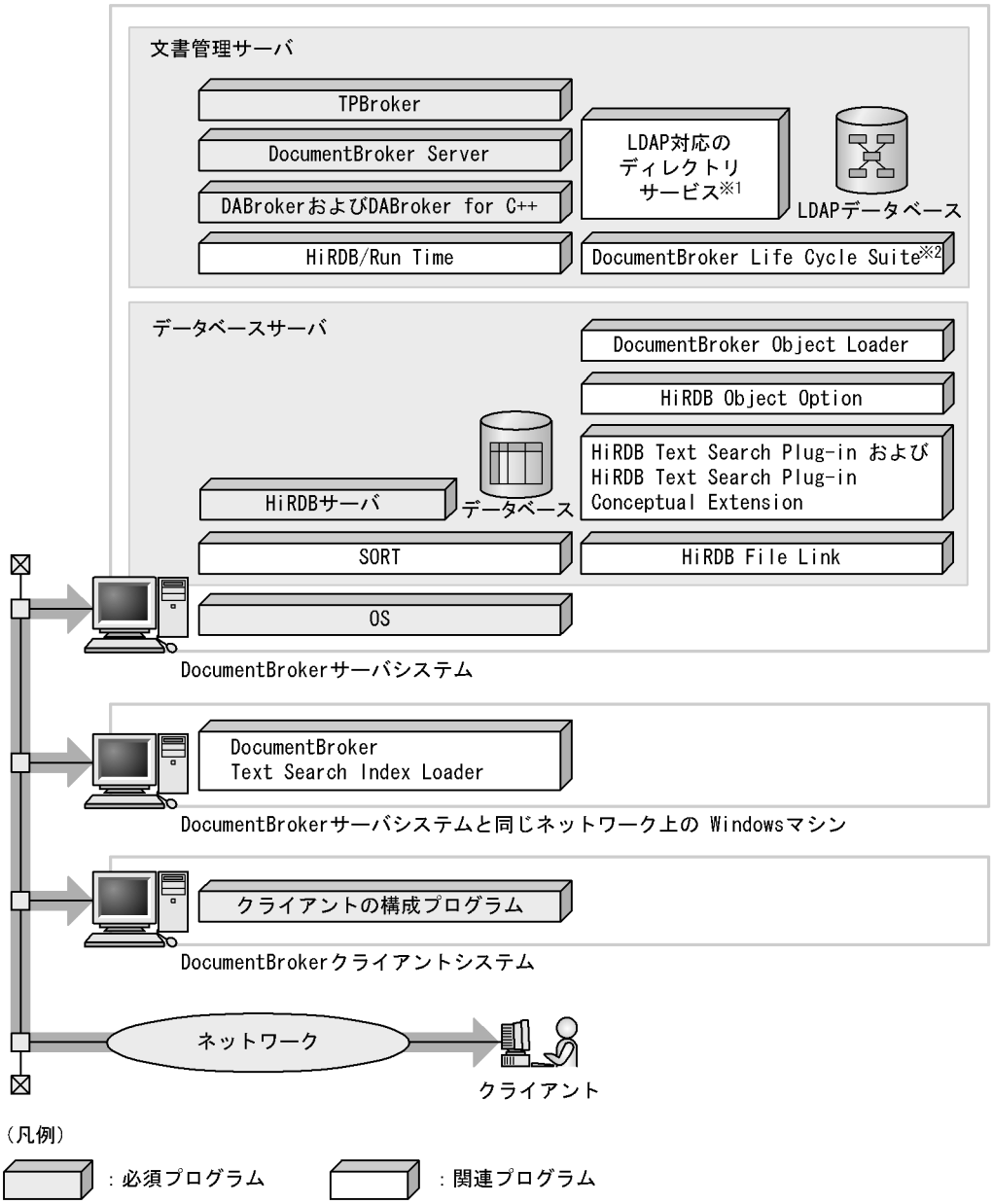
Java 言語対応のクラスライブラリ（Java **クラスライブラリ**）では，Java 環境で動作するクライアントアプリケーションを開発・実行できます。また，Java クラスライブラリを使用して開発したサンプルの Web アプリケーションも提供しているため，このサンプルを参考にして，業務に適合したクライアントアプリケーションを開発できます。

次に，DocumentBroker を使用したシステムでの，DocumentBroker サーバを構成するプログラムと，DocumentBroker クライアントを構成するプログラムについて説明します。

1.3.2 DocumentBroker サーバを構成するプログラム

DocumentBroker サーバを構成するプログラムを必須プログラムと関連プログラムに分けて説明します。DocumentBroker サーバを構成するプログラムのシステム構成を次の図に示します。

図 1-14　DocumentBroker サーバを構成するプログラムのシステム構成



注※1　ユーザ管理機能を使用するためには，LDAP対応のディレクトリサービス，UOCのアクセスルーチン対応のユーザ管理システム，またはUNIXのパスワードファイルシステムによるユーザ管理システムが必要です。

注※2　DocumentBroker Life Cycle Suiteを使用して文書のライフサイクル管理をするためには，DocumentBroker Development KitまたはDocumentBroker Runtime，およびWorkCoordinatorが必要です。

この図では，DocumentBroker サーバを配置したサーバを文書管理サーバ，HiRDB サーバを配置したサーバをデータベースサーバと呼んでいます。

(1) 必須プログラム

TPBroker

CORBA を利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。TPBroker は，ORB 機能を利用して，クライアントアプリケーションとの間の通信基盤として使用します。

VisiBroker で実現できる ORB 機能に，OTS 機能，運用支援機能などを備えています。
なお，Linux の場合は TPBroker V3 は使用できません。TPBroker V5 を使用してください。

DocumentBroker Server

DocumentBroker のサーバ機能を提供するプログラムです。文書管理に使用する文書空間を提供します。また，このほかに，アクセス制御機能，サーバ監視プロセス，および運用コマンドを提供します。

DABroker および DABroker for C++

DABroker は，分散オブジェクト環境でのシームレスなデータベースアクセスを提供するプログラムです。DABroker for C++ は，DABroker の機能を C++ クラスライブラリとして提供するプログラムです。

HiRDB サーバ

DocumentBroker で管理する文書や属性を格納するデータベースシステムです。HiRDB/Parallel Server または HiRDB/Single Server が必要です。

SORT

HiRDB サーバの前提製品で，データのソート時に必要な製品です。HiRDB Version 6 または HiRDB Version 7 を使用する場合に必要になります。HiRDB Version 8 以降を使用する場合，SORT は必要ありません。

OS

オペレーティングシステムは AIX または Linux を使用します。

(2) DocumentBroker サーバと HiRDB サーバを別マシンで利用する場合に必要なプログラム

HiRDB/Run Time

DocumentBroker サーバと HiRDB サーバは，別のマシンに配置できます。DocumentBroker サーバと HiRDB サーバを別のマシンに配置する場合は，DocumentBroker を配置する文書管理サーバ側に HiRDB/Run Time をインストールしてください。

(3) 全文検索機能を使用する場合に必要なプログラム

全文検索機能または概念検索機能使用するためには，次のプログラムが必要です。

文書に対する全文検索機能の場合

- HiRDB Object Option ※
- HiRDB Text Search Plug-in

概念検索機能の場合

- HiRDB Object Option ※
- HiRDB Text Search Plug-in
- HiRDB Text Search Plug-in Conceptual Extension

文字列型プロパティに対する全文検索機能の場合

- HiRDB Object Option ※
- HiRDB Text Search Plug-in

注※

HiRDB Version 6 または HiRDB Version 7 を使用する場合に必要になります。

各プログラムについて説明します。

HiRDB Object Option

RDB である HiRDB を ORDB に拡張するオプション製品です。HiRDB に HiRDB Object Option を組み込むことで，数値や文字データだけでなく，文書や画像などを格納できるようになります。

HiRDB Text Search Plug-in

検索タームを指定して全文検索する場合に必要なプログラムです。HiRDB Text Search Plug-in 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。

HiRDB Text Search Plug-in Conceptual Extension

文章または文字列をキーに概念検索する場合に必要なプログラムです。HiRDB Text Search Plug-in Conceptual Extension 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。

(4) File Link 連携機能を使用する場合に必要なプログラム

HiRDB File Link

コンテント自体は任意のファイルサーバへ格納し，格納先の情報（ファイル名などのリンク情報）をデータベース（HiRDB）に登録するという保管システムを構築する製品です。HiRDB File Link は，HiRDB サーバを配置したサーバ，ファイルサーバ，およびコンテンツアクセス用アプリケーションの実行環境（クライアント）すべてへインストールする必要があります。HiRDB File Link の詳細については，マニュアル「HiRDB File Link」を参照してください。

なお，Linux の場合および TPBroker V5 を使用している場合，File Link 連携機能は使用できません。

(5) アクセス制御機能を使用する場合に必要なプログラム

LDAP 対応のディレクトリサービス

DocumentBroker が提供するアクセス制御機能を使用する場合，DocumentBroker では，LDAP 対応のディレクトリサービスとして，次の製品との連携をサポートしています。

- IBM Directory Server（AIX の場合）
- IBM Tivoli Directory Server（AIX の場合）
- SecureWay Directory（AIX の場合）
- Oracle Directory Server（Linux の場合）
- Active Directory Server（Linux の場合）

なお，ユーザ管理システムとして UNIX のパスワードファイルを使用している場合，運用中の LDAP 対応のディレクトリサービスが，そのままでは DocumentBroker のアクセス制御機能に適応できないことがあります。このような場合，ユーザが作成したアクセスルーチン（UOC）を組み込むことによって，そのユーザ管理システムを DocumentBroker のアクセス制御機能として使用できます。

(6) オブジェクト一括登録機能を使用する場合に必要なプログラム

DocumentBroker Object Loader

大量のオブジェクトを DocumentBroker に一括して登録する場合は，DocumentBroker Object Loader が必要です。DocumentBroker Object Loader では，DocumentBroker からオブジェクトを取得してほかの DocumentBroker へ一括して登録する場合に，DocumentBroker Object Loader の実行に必要な入力データファイルを生成する機能も提供しています。

さらに，DocumentBroker Object Loader High-end Option を使用すると，複数の DocumentBroker Object Loader を同時に実行して，オブジェクトの登録時間を短縮できます。

DocumentBroker Object Loader, および DocumentBroker Object Loader High-end Option の詳細については，マニュアル「DocumentBroker Object Loader Version 3」を参照してください。

(7) 全文検索インデクス一括登録機能を使用する場合に必要なプログラム

DocumentBroker Text Search Index Loader

DocumentBroker に登録されている文書からテキスト情報を抽出して，全文検索インデクスとして一括登録する場合は，DocumentBroker Text Search Index Loader が必要です。DocumentBroker Text Search Index Loader は，DocumentBroker サーバと同じネットワーク上の Windows マシンにインストールしてください。

DocumentBroker Text Search Index Loader の詳細については，マニュアル「DocumentBroker Text Search Index Loader Version 3」を参照してください。

(8) 文書のライフサイクル管理機能を使用する場合に必要なプログラム

DocumentBroker Life Cycle Suite

文書のライフサイクル管理をする場合は，DocumentBroker Life Cycle Suite が必要です。

DocumentBroker から WorkCoordinator を操作するための C++ クラスライブラリおよび分散オブジェクト群を提供しています。また，デフォルトオブジェクトで対応できない複雑なロジックを持つルールを設定したい場合のために，アプリケーションフレームワークを提供しています。

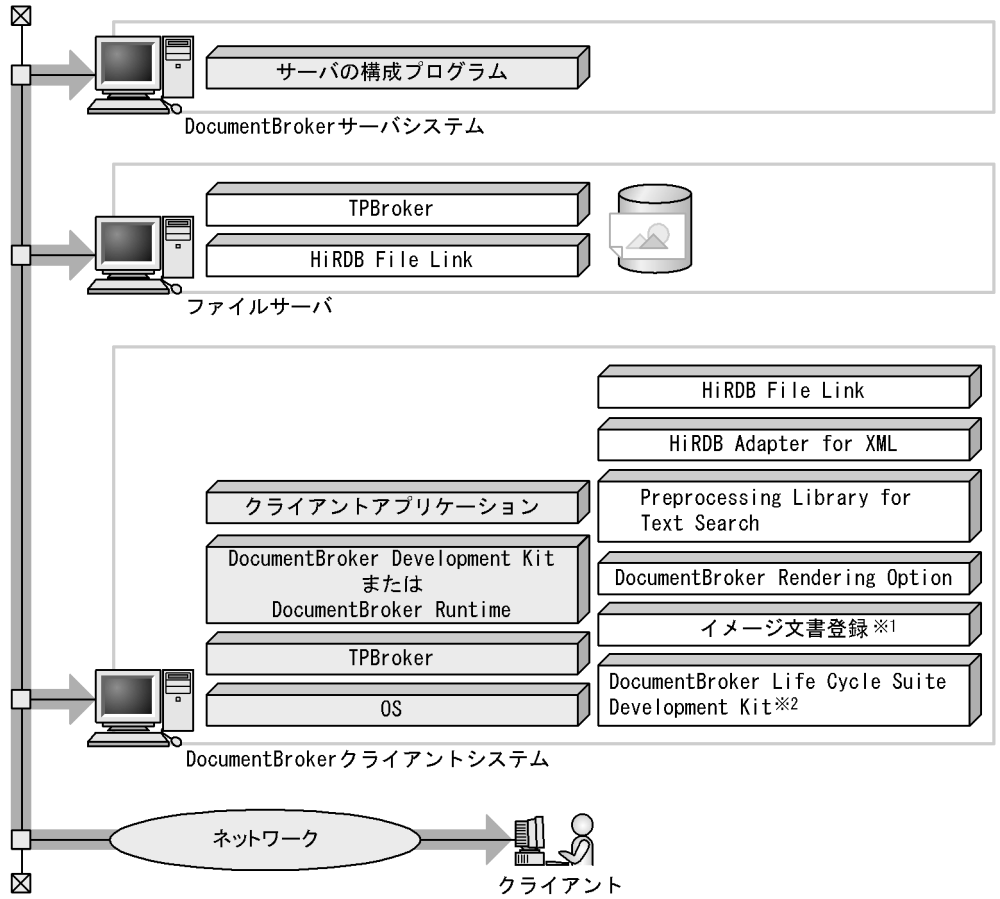
文書のライフサイクル管理をする場合は，DocumentBroker サーバを配置する文書管理サーバ上に，DocumentBroker Life Cycle Suite のほかに DocumentBroker Development Kit または DocumentBroker Runtime，および WorkCoordinator が必要です。

DocumentBroker Life Cycle Suite の詳細については，マニュアル「DocumentBroker Life Cycle Suite Version 2」を参照してください。

1.3.3 DocumentBroker Development Kit で開発したクライアントを構成するプログラム（C++ クラスライブラリの場合）

ここでは，DocumentBroker Development Kit が提供する C++ クラスライブラリで開発したクライアントを構成するプログラムを，必須プログラムと関連プログラムに分けて説明します。C++ クラスライブラリで開発したクライアントを構成するプログラムのシステム構成を次の図に示します。

図 1-15　C++ クラスライブラリで開発したクライアントを構成するプログラムのシステム構成



注※1　イメージ文書登録機能を使用する場合は，ご使用のプリンタに対応したプログラムをご使用ください。

注※2　DocumentBroker Life Cycle Suiteを使用して文書のライフサイクル管理をするためには，DocumentBroker Life Cycle Suite Development KitまたはDocumentBroker Life Cycle Suite Runtime，およびWorkCoordinator Definerが必要です。

(1) 必須プログラム

クライアントアプリケーション

C++ クラスライブラリを使用してユーザが作成するアプリケーションプログラムです。

C++ クラスライブラリを使用してクライアントアプリケーションを作成する方法については，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」を参照してください。

DocumentBroker Development Kit（クライアント開発環境）または DocumentBroker Runtime（クライアント実行環境）

DocumentBroker Development Kit は，C++ 言語対応のクライアントアプリケーション開発用のクラスライブラリ，およびクライアントの実行環境を提供します。

クライアントの実行環境だけが必要な場合は，DocumentBroker Runtime を使用します。

DocumentBroker Runtime は，DocumentBroker Development Kit で開発したクライアントアプリケーションを実行するためのライブラリを提供しています。

C++ クラスライブラリの詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　リファレンス　基本機能編」を参照してください。
また，クライアントアプリケーションを円滑に開発するために必要な C++ クラスライブラリの機能を，コマンド群（オブジェクト操作ツール）として提供している DocumentBroker オブジェクト操作ツールを使用する場合には，マニュアル「DocumentBroker Version 3　オブジェクト操作ツール」を参照してください。

TPBroker

DocumentBroker Development Kit の前提プログラムとして，CORBA 仕様に基づく分散アプリケーションを開発するためのプログラムです。

OS

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。
- **Linux の場合：**
  Red Hat Enterprise Linux 6 (x86_64) 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 2008 の場合：**
  Windows Server 2008，Enterprise Edition または Windows Server 2008，Standard Edition を使用してください。
- **Windows Server 2008 R2 の場合：**
  Windows Server 2008 R2，Enterprise Edition または Windows Server 2008 R2，Standard Edition 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 Business，Windows Vista Enterprise または Windows Vista Ultimate を使用してください。
- **Windows 7 の場合：**
  Windows 7 Professional，Windows 7 Enterprise または Windows 7 Ultimate を使用してください。

(2) File Link 連携機能を使用する場合に必要なプログラム

HiRDB File Link

コンテント自体は任意のファイルサーバへ格納し，格納先の情報（ファイル名などのリンク情報）をデータベース（HiRDB）に登録するという保管システムを構築する製品です。HiRDB File Link は，HiRDB サーバを配置したサーバ，ファイルサーバ，およびコンテンツアクセス用アプリケーションの実行環境（クライアント）すべてへインストールする必要があります。HiRDB File Link の詳細については，マニュアル「HiRDB File Link」を参照してください。
なお，Linux の場合および TPBroker V5 を使用している場合，File Link 連携機能は使用できません。

(3) XML 文書管理機能を使用する場合に必要なプログラム

HiRDB Adapter for XML

XML 文書を解析して，XML のタグを RDB の表や列に対応づけ，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 文書管理機能を利用して 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。

HiRDB Adapter for XML の詳細については，マニュアル「HiRDB Adapter for XML　ユーザーズガイド」を参照してください。

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利用する場合は，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を全文検索インデクスとして登録した XML 文書は，XML の構造（タグ間の文字列，タグの属性）をキーにした全文検索の対象にできます。

Preprocessing Library for Text Search の詳細については，マニュアル「Preprocessing Library for Text Search Version 2」を参照してください。

なお，Linux の場合，XML 文書管理機能は使用できません。

(4) 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

レンディション変換要求機能を利用する場合に，レンディション変換を実行するクライアントの，クライアントアプリケーションとして使用するプログラムです。DocumentBroker に登録された文書のマスタレンディションを，サブレンディションで指定されたレンディションに変換してサブレンディションのコンテントとしてサーバに登録する機能を持ちます。レンディション変換を実行するクライアントには，マスタレンディションのコンテントを扱うアプリケーションプログラム，および指定されたレンディションの形式へのコンテントの変換を実行するアプリケーションプログラムも必要です。例えば，Word 形式から PDF 形式にレンディション変換する場合は，Office および Adobe Acrobat が必要です。

DocumentBroker Rendering Option の詳細については，マニュアル「DocumentBroker Rendering Option」を参照してください。

(5) イメージ文書登録機能を使用する場合に必要なプログラム

イメージ文書登録機能を使用する場合，ご使用のデジタル複合機に対応したプログラムを使用してください。これらのプログラムは，デジタル複合機でスキャンした紙の文書や受信した FAX を DocumentBroker にイメージ文書として登録するクライアントの，クライアントアプリケーションとして使用します。

DocumentBroker **イメージ登録** for DocuCentre

DocuCentre でスキャンした紙の文書や受信した FAX を DocumentBroker にイメージ文書として登録するプログラムです。イメージ文書を登録するクライアントには，CentreWare スキャンサービスおよび CentreWare スキャンサービス カスタム連携ツールも必要です。

DocumentBroker **イメージ登録** for iR

imageRUNNER でスキャンした紙の文書や受信した FAX を，DocumentBroker にイメージ文書として登録するプログラムです。

DocumentBroker **イメージ登録** for imagio and IPSiO

imagio または IPSiO でスキャンした紙の文書や受信した FAX を，DocumentBroker にイメージ文書として登録するプログラムです。

DocumentBroker **イメージ登録** for Konica

Sitios でスキャンした紙の文書や受信した FAX を，DocumentBroker にイメージ文書として登録するプログラムです。

DocumentBroker **イメージ登録 文字認識オプション**

DocumentBroker にイメージ文書を登録するときに，文書に記載された情報を文字認識して，文書のプロパティとして登録したり帳票ごとに分割して文書を登録したりできます。

イメージ文書登録機能の詳細については，マニュアル「DocumentBroker イメージ登録 for DocuCentre」，マニュアル「DocumentBroker イメージ登録 for iR」，マニュアル「DocumentBroker イメージ登録 for imagio and IPSiO」，またはマニュアル「DocumentBroker イメージ登録 for Konica」を参照してください。

(6) 文書のライフサイクル管理機能を使用する場合に必要なプログラム

DocumentBroker Life Cycle Suite Development Kit**（クライアント開発環境）または** DocumentBroker Life Cycle Suite Runtime**（クライアント実行環境）**

文書のライフサイクル管理の機能を利用するクライアントアプリケーションを開発する場合，DocumentBroker Life Cycle Suite Development Kit が必要です。なお，その場合は，WorkCoordinator のビジネスプロセスを定義する WorkCoordinator Definer も必要です。

DocumentBroker Life Cycle Suite Development Kit は，文書のライフサイクル管理機能を利用したクライアントアプリケーション開発用のクラスライブラリおよびクライアント実行環境を提供します。

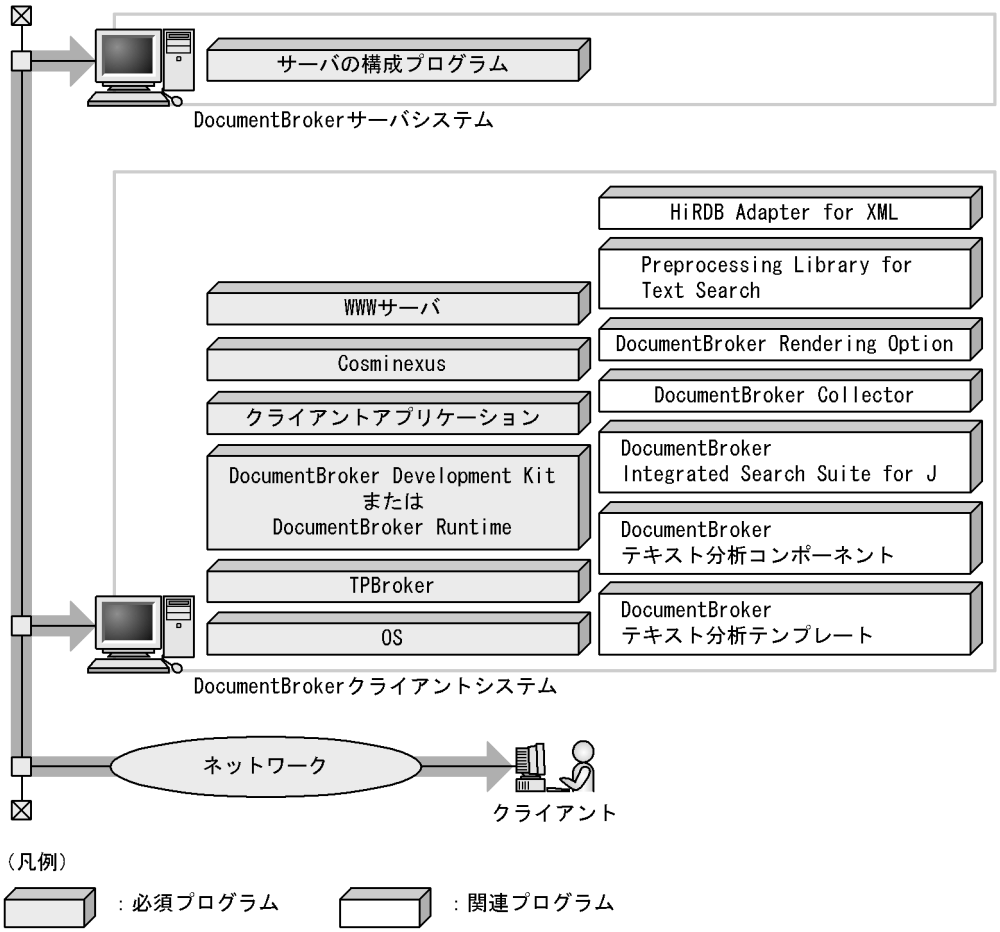
文書のライフサイクル管理の機能を利用するクライアントの実行環境だけが必要な場合は，DocumentBroker Life Cycle Suite Runtime を使用します。DocumentBroker Life Cycle Suite Runtime は，DocumentBroker Life Cycle Suite Development Kit で開発したクライアントアプリケーションを実行するためのライブラリを提供しています。

DocumentBroker Life Cycle Suite の詳細については，マニュアル「DocumentBroker Life Cycle Suite Version 2」を参照してください。

1.3.4 DocumentBroker Development Kit で開発したクライアントを構成するプログラム（Java クラスライブラリの場合）

ここでは，DocumentBroker Development Kit が提供する Java クラスライブラリで開発したクライアントを構成するプログラムを，必須プログラムと関連プログラムに分けて説明します。Java クラスライブラリで開発したクライアントを構成するプログラムのシステム構成を次の図に示します。

図 1-16　Java クラスライブラリで開発したクライアントを構成するプログラムのシステム構成



（1）必須プログラム

WWW サーバ

Java クラスライブラリは，WWW サーバシステムが構築されている環境で使用します。

Cosminexus

Cosminexus は，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 クラスライブラリに必要な環境を提供しています。

クライアントアプリケーション

Java クラスライブラリを使用してユーザが作成す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クライアントアプリケーションを作成する方法については，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　解説」を参照してください。

DocumentBroker Development Kit（クライアント開発環境）または DocumentBroker Runtime（クライアント実行環境）

DocumentBroker Development Kit は，Java 言語対応のクライアントアプリケーション開発用のクラスライブラリ，およびクライアントの実行環境を提供します。

クライアントの実行環境だけが必要な場合は，DocumentBroker Runtime を使用します。

DocumentBroker Runtime は，DocumentBroker Development Kit で開発したクライアントアプリケーションを実行するためのライブラリを提供しています。

また，Java クラスライブラリを使用して開発したサンプルの Web アプリケーションも提供している

ため，このサンプルを参考にして，業務に適合したクライアントアプリケーションを開発できます。Java クラスライブラリの詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　解説」，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　リファレンス」，およびマニュアル「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。

TPBroker

CORBA 仕様に基づく分散アプリケーションを開発するためのプログラムです。

OS

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。
- **Linux の場合：**
  Red Hat Enterprise Linux 6 (x86_64) 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 2008 の場合：**
  Windows Server 2008，Enterprise Edition または Windows Server 2008，Standard Edition を使用してください。
- **Windows Server 2008 R2 の場合：**
  Windows Server 2008 R2，Enterprise Edition または Windows Server 2008 R2，Standard Edition 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 Business，Windows Vista Enterprise または Windows Vista Ultimate を使用してください。
- **Windows 7 の場合：**
  Windows 7 Professional，Windows 7 Enterprise または Windows 7 Ultimate を使用してください。

(2) XML 文書管理機能を使用する場合に必要なプログラム

XML 文書管理機能を使用する場合は，次のプログラムが必要です。

- HiRDB Adapter for XML
- Preprocessing Library for Text Search

これらのプログラムの説明については，「1.3.3(3)　XML 文書管理機能を使用する場合に必要なプログラム」を参照してください。

なお，Linux の場合，XML 文書管理機能は使用できません。

(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

レンディション変換要求機能を使用する場合は，次のプログラムが必要です。

- DocumentBroker Rendering Option

このプログラムの説明については，「1.3.3(4) 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」を参照してください。

(4) 分散した文書の収集・検索機能を使用する場合に必要なプログラム

DocumentBroker Collector

ネットワーク上に散在した文書を収集して DocumentBroker に登録する場合に必要なプログラムです。インターネット，イントラネット，ファイルシステムやグループウェアなどに分散しているさまざまなフォーマットの文書から属性を収集し，XML 形式に変換して DocumentBroker に登録して管理するプログラムです。DocumentBroker Collector の詳細については，マニュアル「DocumentBroker Collector」を参照してください。

DocumentBroker Integrated Search Suite for J

DocumentBroker Collector によって収集および登録された文書を検索する場合に必要なプログラムです。DocumentBroker Integrated Search Suite for J が提供する検索テンプレートを使用して WWW ブラウザ上に表示した検索画面から文書を検索できます。また，Cosminexus Portal Framework と連携すると，検索画面を Cosminexus Portal Framework のポートレットとして表示できます。DocumentBroker Integrated Search Suite for J の詳細については，マニュアル「DocumentBroker Integrated Search Suite for J Version 2」を参照してください。

(5) 文書の分析機能を使用する場合に必要なプログラム

DocumentBroker テキスト分析コンポーネント

DocumentBroker で管理されている文書のテキスト情報と属性情報を分析する場合に必要なプログラムです。分析対象となるデータを生成するためのコマンドやクラスライブラリを提供しています。DocumentBroker テキスト分析コンポーネントの詳細については，マニュアル「DocumentBroker テキスト分析コンポーネント」を参照してください。

DocumentBroker テキスト分析テンプレート

DocumentBroker テキスト分析コンポーネントで生成したデータを分析する場合に必要なプログラムです。DocumentBroker テキスト分析テンプレートが提供する分析テンプレートを使用して WWW ブラウザ上に表示した分析画面から文書を分析できます。DocumentBroker テキスト分析テンプレートの詳細については，マニュアル「DocumentBroker テキスト分析テンプレート」を参照してください。

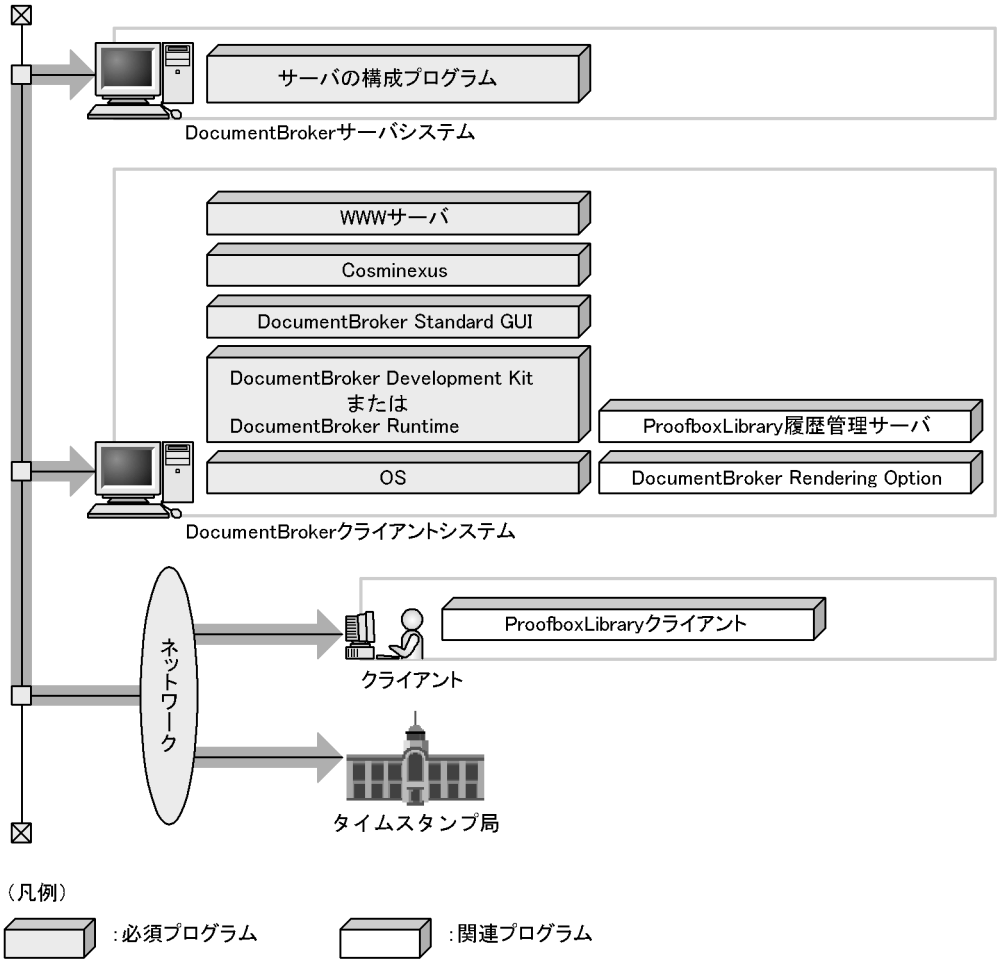
1.3.5 DocumentBroker Standard GUI を使用する場合のクライアントを構成するプログラム

DocumentBroker Standard GUI は，業種・業務に依存しない標準的な企業内文書管理システムを提供する ECM パッケージ製品です。

DocumentBroker Standard GUI は，文書を審査・承認後に公開する仕組み，公開する文書に電子署名や承認日時を付与する機能などを文書管理システムに取り込めるため，コンプライアンスや情報セキュリティへの対応が容易になります。

ここでは，DocumentBroker Standard GUI を使用する場合のクライアントを構成するプログラムを，必須プログラムと関連プログラムに分けて説明します。DocumentBroker Standard GUI を使用する場合のクライアントを構成するプログラムのシステム構成を次の図に示します。

図 1-17　DocumentBroker Standard GUI を使用する場合のクライアントを構成するプログラムのシステム構成



（1）必須プログラム

WWW サーバ

DocumentBroker Standard GUI は，WWW サーバシステムが構築されている環境で使用します。DocumentBroker Standard GUI で使用できる WWW サーバの種類と WWW サーバのバージョンについては，マニュアル「DocumentBroker Version 3 Standard GUI　システム導入・運用ガイド」を参照してください。

Cosminexus

Cosminexus は，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，DocumentBroker Standard GUI に必要な環境を提供しています。DocumentBroker Standard GUI で使用できる Cosminexus の種類と Cosminexus のバージョンについては，マニュアル「DocumentBroker Version 3 Standard GUI　システム導入・運用ガイド」を参照してください。

DocumentBroker Standard GUI

DocumentBroker Standard GUI は，Cosminexus で構築した WWW 環境で動作する DocumentBroker の標準 GUI です。DocumentBroker Standard GUI の詳細については，マニュアル「DocumentBroker Version 3 Standard GUI　システム導入・運用ガイド」およびマニュアル「DocumentBroker Version 3 Standard GUI　操作ガイド」を参照してください。

DocumentBroker Development Kit または DocumentBroker Runtime

DocumentBroker Development Kit および DocumentBroker Runtime は，DocumentBroker Standard GUI が DocumentBroker サーバに対して処理を要求するための API を持つプログラムです。DocumentBroker Standard GUI では，DocumentBroker Development Kit または DocumentBroker Runtime が提供する Java クラスライブラリを使用して DocumentBroker サーバに処理を要求して，標準 GUI の機能を実現します。

OS

DocumentBroker Standard GUI で使用できる OS については，マニュアル「DocumentBroker Version 3 Standard GUI　システム導入・運用ガイド」を参照してください。

(2) 電子署名機能およびタイムスタンプ機能を使用する場合に必要なプログラム

電子署名機能およびタイムスタンプ機能を使用するためには，次のプログラムが必要です。

- ProofboxLibrary 履歴管理サーバ
- ProofboxLibrary クライアント
  ProofboxLibrary クライアントは，クライアントに必要なプログラムです。

DocumentBroker Standard GUI で使用できるこれらのプログラムのバージョンについては，マニュアル「DocumentBroker Version 3 Standard GUI　システム導入・運用ガイド」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

レンディション変換要求機能を使用する場合は，次のプログラムが必要です。

- DocumentBroker Rendering Option

このプログラムの説明については，「1.3.3(4)　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」を参照してください。

1.4 プロセス構成

この節では，DocumentBroker のプロセス構成について説明します。

1.4.1 DocumentBroker の基本プロセス構成

DocumentBroker の基本プロセス構成を次の表に示します。

表 1-3 DocumentBroker の基本プロセス構成

プロセス	機能概要
起動プロセス（EDMStart）	DocumentBroker サーバを起動する。
停止プロセス（EDMStop）	DocumentBroker サーバを停止する。
サーバ監視プロセス（EDMDaemon）	一つの DocumentBroker システムを監視する。
サービスプロセス監視プロセス（EDMSrvMgr）	サービスプロセスの動作を監視する。
サービスプロセス（EDMService）	クライアントへ文書空間のサービスを供給する。

各プロセスについて説明します。

(1) 起動プロセス（EDMStart）

DocumentBroker サーバを起動するプロセスです。システム管理者が DocumentBroker サーバを起動した時点で（EDMStart コマンドの実行）生成されます。起動プロセスは，サーバ監視プロセスを起動します。

(2) 停止プロセス（EDMStop）

DocumentBroker サーバを終了するプロセスです。システム管理者が DocumentBroker サーバを終了した時点で（EDMStop コマンドの実行）生成されます。停止プロセスは，サーバ監視プロセスに対して終了要求を実行し，終了要求に対するサーバ監視プロセスの応答を確認してから終了します。

(3) サーバ監視プロセス（EDMDaemon）

DocumentBroker を監視するプロセスです。サービスプロセス監視プロセスの動作状況を監視して，サービスプロセス監視プロセスが終了した場合は，このプロセスが監視しているサービスプロセスをすべて強制終了させてから，サービスプロセス監視プロセスの再起動を実行します。このとき，強制終了したサービスプロセスに接続していたクライアントにはエラーが返却されて，サービスの提供を受けられなくなります。サーバ監視プロセスは，起動プロセスによって生成され，停止プロセスからの終了要求によってサービスプロセス監視プロセスへ停止要求を通知します。

(4) サービスプロセス監視プロセス（EDMSrvMgr）

サービスプロセスを定義されている数だけ生成して，サービスプロセスの状態を管理するプロセスです。サービスプロセスの動作状況も監視して，サービスプロセスが終了した場合は，サービスプロセスを再起動します。サービスプロセス監視プロセスは，サーバ監視プロセスによって生成され，サーバ監視プロセスからの停止要求によってサービスプロセスへ停止要求を通知します。

(5) サービスプロセス（EDMService）

クライアントへ文書空間のサービスを供給するプロセスです。サービスプロセスは，DocumentSpace 構成定義ファイルに指定されている数だけサービスプロセス監視プロセスによって生成されます。システム

管理者が DocumentBroker を終了した場合（EDMStop コマンドの実行）は，サービスプロセス監視プロセスからの停止要求によって，すべてのサービスプロセスは終了します。また，障害などによってサービスプロセスが終了した場合は，サービスプロセス監視プロセスによって再起動されます。

サービスプロセスは，1 プロセス当たり複数クライアントに対してサービスを供給できます。ただし，サービスプロセスがダウンした場合，ダウンしたプロセスに接続していたクライアントにはエラーが返却されて，接続できなくなります。したがって，クライアントからの再接続が必要となります。

参考　サービスプロセスのリフレッシュ

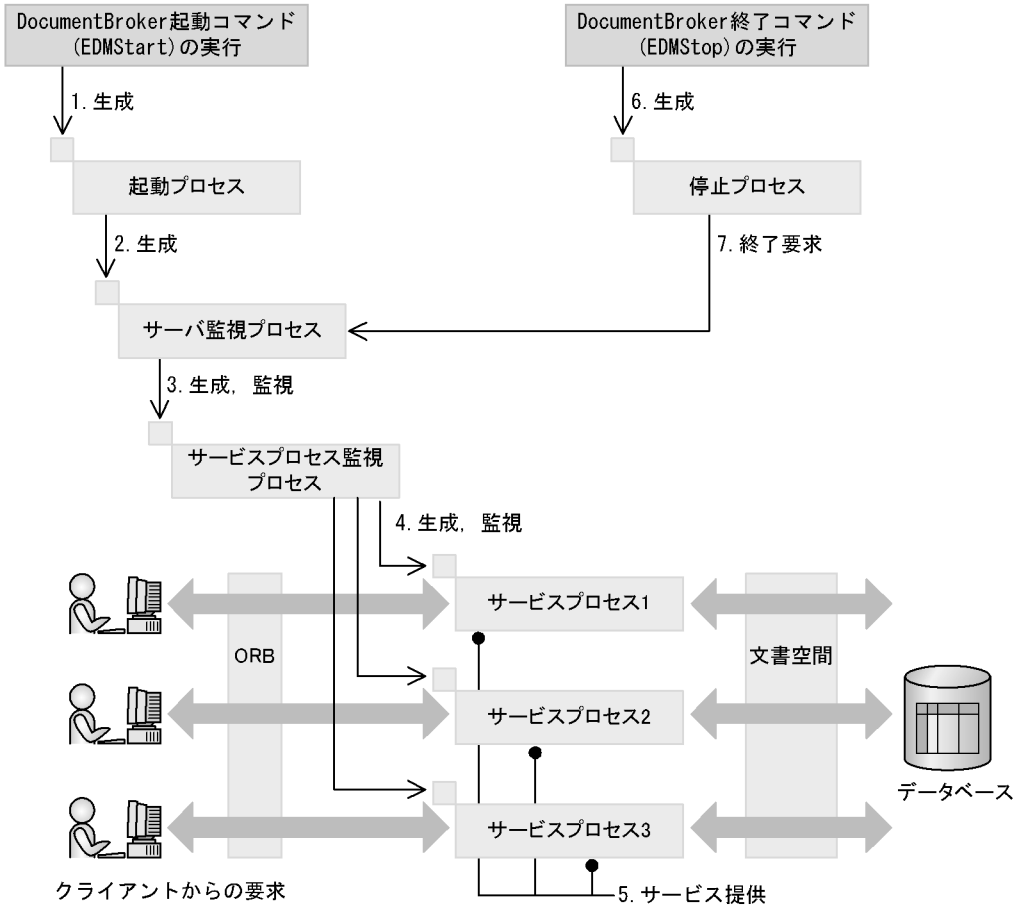
DocumentBroker サーバを長時間連続して運転すると，サービスプロセスのサイズが大きくなり，メモリ不足による障害などが起きる可能性があります。このため，DocumentBroker では，サービスプロセスをリフレッシュするコマンド（EDMRefresher）やサービスプロセスのメモリサイズを監視してリフレッシュする機能を提供しています。リフレッシュとは，DocumentBroker サーバを停止することなく，クライアントからの要求に対して文書空間へのサービスを供給しながら，サービスプロセスを順次再起動することです。OS の cron デーモンなどを利用して，EDMRefresher コマンドを定期的に実行したり，DocumentSpace 構成定義ファイルにリフレッシュする契機となるサービスプロセスのメモリサイズを指定することで，DocumentBroker サーバを連続して運転できます。

EDMRefresher コマンドの詳細については，「7.3　コマンドの文法」の「EDMRefresher（サービスプロセスのリフレッシュ）」を参照してください。

1.4.2　DocumentBroker のプロセスの関連

DocumentBroker のプロセスの関連について説明します。

図 1-18　基本プロセスの関連



1. DocumentBroker の起動コマンドの実行によって，起動プロセスが生成されます。
2. 起動プロセスは，サーバ監視プロセスを生成します。
3. サーバ監視プロセスは，サービスプロセス監視プロセスを生成して，その状態を監視します。
4. サービスプロセス監視プロセスは，サービスプロセスを生成して，その状態を監視します。
5. サービスプロセスは，クライアントからの要求に対して，文書空間へのサービスを供給します。
6. DocumentBroker の終了コマンドの実行によって，停止プロセスが生成されます。
7. 停止プロセスはサーバ監視プロセスに対して終了要求を実行し，終了要求に対するサーバ監視プロセスの応答を確認してから終了します。

2 システム導入前の検討

この章では，DocumentBroker を利用した文書管理システムの導入前に必要な検討項目について説明します。

2.1 適用業務の検討

2.2 文書管理に使用するオブジェクト

2.3 追加するクラスおよびプロパティの検討

2.4 文書空間で使用する文字コード種別の検討

2.5 XML 文書管理機能を使用したシステム構築の検討

2.6 アクセス制御機能を使用したシステム構築の検討

2.7 メモリ所要量とディスク占有量の見積もり

2.8 データベース容量の見積もり

2.9 リファレンスファイル管理機能を使用する場合のファイルシステムのディスク容量の見積もり

2.10 File Link 連携機能を使用する場合のファイルサーバ容量の見積もり

2.11 マルチレンディション管理機能を使用する場合の排他資源管理テーブルの見積り

2.1 適用業務の検討

DocumentBroker は，基幹業務での文書管理システムの構築を支援します。この文書管理システムによって膨大な量の文書として蓄積されているデータを扱うことができるとともに，文書管理を基幹業務プロセスの一環として組み込むことで業務の効率を高めます。この節では，DocumentBroker を使用した文書管理システムの導入時に検討していただきたいことについて説明します。

適用業務の分析

文書管理システムを適用する基幹業務の性質についての分析が必要です。検索機能を充実させたシステムが必要な業務か，手書きの書類を大量にデータとして登録する業務か，文書のライフサイクル管理が必要な業務か，異なる業務間の連携が必要な業務か，などといった基幹業務の性質について分析してください。

導入するプログラムの検討

適用業務の分析結果を基に，何を目的とした文書管理システムを構築するか，どの機能を使用するかを明確にして，その機能の実現に必要な前提プログラムおよび関連プログラムの導入について検討してください。

DocumentBroker の機能については，「1.2　DocumentBroker の機能」を参照してください。また，XML 文書管理機能とアクセス制御機能の詳細については，「2.5　XML 文書管理機能を使用したシステム構築の検討」，または「2.6　アクセス制御機能を使用したシステム構築の検討」を参照してください。

機能の実現に必要なプログラムとシステム構成については，「1.3　DocumentBroker のシステム構成」を参照してください。

動作環境や機能のサポート範囲などは製品ごとに異なります。各製品のマニュアルを参照してこれらの内容を十分に確認した上で，必要なプログラムを導入して，業務内容に最も適した機能を持つ文書管理システムを構築してください。

業務で扱う文書の分析

文書管理システムの導入時には，企業や組織の基幹的な業務で扱っている文書についての分析が必要です。これらの文書の種類は非常に多く，組織や部門ごとに異なった文書を扱っています。このため，現行の業務内容でそれぞれの文書が果たしている役割について，よく分析する必要があります。分析する場合の要点を次に示します。

- **文書の特性についての分析**
  管理する対象となる文書の作成と業務との関係について分析します。例えば，「文書の作成方法」については，「一つの文書は複数人で作成する」，「XML で作成する」，「文書をバージョン管理する」などといったことを分析します。この場合，扱っている文書の目的，作成者と作成方法，文書の利用方法，文書と業務との関係，文書同士の関係などを分析する必要があります。
- **文書のライフサイクルについての分析**
  基幹業務で扱っている文書のライフサイクルを分析します。この場合，「作成中／審査中／承認済／公開中／更新中」といった業務での文書の状態や，状態ごとの「作成／レビュー／審査／承認／公開／改訂／破棄」のようなイベントの実施形式などについてそれぞれ分析する必要があります。

文書に付加する情報の検討

対象の文書の分析結果を基にして，管理する文書に付加する情報を検討します。この作業では，管理する文書を種類ごとに分類して，それぞれに必要な付加情報を検討します。例えば，製品の設計ドキュメントを管理する場合は，「設計者名」，「開発完了予定日」，「特許情報」などの情報をあわせて管理することを検討します。なお，DocumentBroker では，「文書」は「クラス」に相当し，「付加情報」は「プロパティ」に相当します。したがって，ここでは，管理する文書の種類ごとにサブクラスを作成するとともに，文書の種類ごとに必要な付加情報については，各サブクラスにプロパティを追加していきます。

文書の管理方法の検討

DocumentBroker では，文書の管理方法によって，使用するオブジェクトが異なります。例えば，文書のバージョンを管理する場合はバージョン付き文書を，コンテナ機能を使用して文書をまとめて管理する場合はコンテナを，構成管理機能を利用して文書のバージョン構成を管理する場合は，構成管理コンテナを使用します。また，例えば，コンテナ機能を使用する場合は，コンテナに具体的でわかりやすい名称を付けるためのプロパティを追加することが考えられます。関連づけの種類ごとに，サブクラスを追加して，そのサブクラスを基にコンテナを作成していくような運用も考えられます。このように，文書の管理方法に合わせて，作成するオブジェクトについて検討し，管理方法に合わせたクラスおよびプロパティを効果的に定義する必要があります。

文書管理に使用するオブジェクトとオブジェクトの構成については，「2.2　文書管理に使用するオブジェクト」を参照してください。

また，クラスおよびプロパティの追加については，「2.3　追加するクラスおよびプロパティの検討」を参照してください。

文書空間で使用する文字コード種別の分析

文書管理システムで使用する文字コード種別を分析し，文書空間で使用する文字コード種別を選択します。

文字コード種別は，文書の付加情報やファイル名に使用する文字コード種別に応じて，Shift-JIS または UTF-8 を選択してください。

文書空間で使用する文字コード種別の検討については，「2.4　文書空間で使用する文字コード種別の検討」を参照してください。

2.2 文書管理に使用するオブジェクト

この節では，DocumentBroker で文書を管理するために知っておく必要がある，DocumentBroker のオブジェクトの概要，オブジェクトの構成，オブジェクトとデータベースとの関係について説明します。また，このマニュアルで使用するオブジェクト名と機能名についても説明します。

この節の内容を参照して，どのオブジェクトを使用して文書を管理するかを検討してください。また，「2.8　データベース容量の見積もり」で DocumentBroker が使用するデータベースの容量を見積もる際には，オブジェクトの構成，DocumentBroker のオブジェクトとデータベースの関係を参考にしてください。

2.2.1 DocumentBroker のオブジェクト

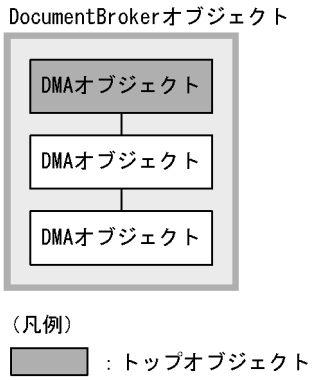
ここでは，DocumentBroker のオブジェクト，クラスおよびプロパティの関係について説明します。

（1）DocumentBroker オブジェクトと DMA オブジェクトの関係

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ます。DocumentBroker の文書空間上に存在する文書やコンテナなどのオブジェクト（**DocumentBroker オブジェクト**）は，データベース上に実体が存在するオブジェクトです。

DocumentBroker では，DMA のオブジェクトモデルに基づいたオブジェクト（**DMA オブジェクト**）を使用して文書を管理しており，DocumentBroker 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ます。

図 2-1　DocumentBroker オブジェクトと DMA オブジェクトの関係



このマニュアルでは，DocumentBroker オブジェクトを構成する DMA オブジェクトのうち，最上位に当たるオブジェクトを**トップオブジェクト**といいます。

DMA オブジェクトには，文書として表現するためのオブジェクト，文書をまとめるコンテナ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。DocumentBroker では，文書管理の用途に応じて，複数の DMA オブジェクトを，文書やコンテナなどのまとまった一つのオブジェクトとして扱います。

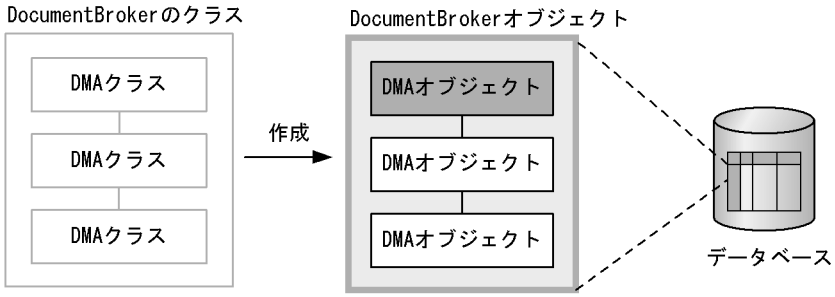
（2）DocumentBroker のクラスと DMA クラスの関係

DMA オブジェクトに対応するクラスが DMA クラスです。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。DMA クラスには，文書を表すオブジェクトのクラス，コンテナを表すオブ

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（クラス名が dmaClass_ で始まるクラス）に加えて，拡張した機能を持つ DMA クラス（クラス名が edmClass_ で始まるクラス）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。

DocumentBroker では，文書管理に必要な複数の DMA クラスをまとめたクラスを提供しています。また，DocumentBroker Development Kit では，文書管理に必要な操作をメソッドとして提供しています。DocumentBroker のクラスを基に DocumentBroker オブジェクトを作成することで，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，メソッドを実行して DocumentBroker オブジェクトを操作することで，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 DocumentBroker 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，文書のコンテントデータを管理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。

図 2-2　DocumentBroker でのオブジェクトの操作



2.2.2　このマニュアルで使用するオブジェクト名と機能名

ここでは，このマニュアルで使用するオブジェクト名と機能名について説明します。

このマニュアルでは，DocumentBroker の機能について説明するときに，C++ クラスライブラリに対応するマニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」で使用している用語を使用します。

なお，オブジェクト名や機能名などの用語は，DocumentBroker Development Kit が提供する C++ クラスライブラリと Java クラスライブラリで異なる場合があります。それぞれのオブジェクト名と機能名の対応を次の表に示します。

表 2-1　オブジェクト名と機能名の対応

分類	C++ クラスライブラリの場合	Java クラスライブラリの場合
DocumentBroker オブジェクト名	バージョンなし文書	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
	バージョンなしコンテナ	バージョンなしフォルダ
	バージョンなし構成管理コンテナ	―
	バージョン付き構成管理コンテナ	バージョン付きフォルダ
	独立データ	独立データ
	パブリック ACL	パブリック ACL
機能名	文書間リレーション	文書間リンク

分類	C++ クラスライブラリの場合	Java クラスライブラリの場合
	コンテインメント	リンク

（凡例）
－：サポートしていません。

なお，このマニュアルでは，それぞれの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。

コンテナ
「バージョンなしコンテナ」,「バージョンなし構成管理コンテナ」および「バージョン付き構成管理コンテナ」の総称として使用します。

構成管理コンテナ
「バージョンなし構成管理コンテナ」と「バージョン付き構成管理コンテナ」の総称として使用します。

次に，DocumentBroker Development Kit の C++ クラスライブラリに対応するマニュアルで使用しているオブジェクト名と機能名を使用して，DocumentBroker オブジェクトの構成や機能について説明します。なお，クライアントアプリケーションを構築するインターフェースごとにオブジェクトの構成や機能が異なる場合があります。オブジェクトや機能の名称，オブジェクトの構成およびオブジェクトの操作方法については，使用するクライアントアプリケーションを構築するインターフェースに応じて，次のマニュアルのどちらかを参照してください。

- 「DocumentBroker Version 3　クラスライブラリ　C++　解説」
- 「DocumentBroker Version 3　クラスライブラリ　Java　解説」

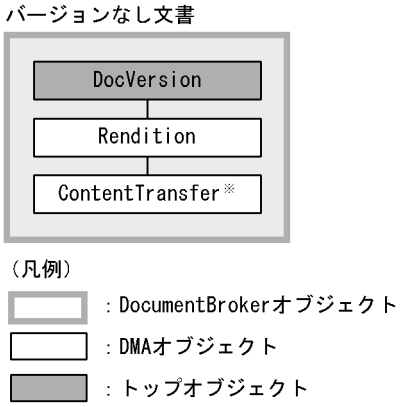
2.2.3 バージョンなし文書の管理に必要なオブジェクト

ここでは，バージョンなし文書のオブジェクト構成について説明します。

バージョンなし文書は，一つのバージョンだけを持ち，バージョンを管理しない文書です。バージョンなし文書は，文書の実体であるコンテント（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），プロパティ，およびレンディションについての情報を保持するオブジェクトによって構成されています。

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 2-3 バージョンなし文書を構成する DMA オブジェクト



注※ マルチファイル管理機能を使用する場合はContentTransfersオブジェクト，
リファレンスファイル管理機能を使用する場合はContentReferenceオブジェクト，
File Link連携機能を使用する場合はContentFileLinkオブジェクトになります。

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。なお，マルチファイル管理機能，リファレンスファイル管理機能または File Link 連携機能を使用する場合，DMA オブジェクトの構成が一部異なります。

- DocVersion **オブジェクト**
  バージョンなし文書で，文書を表現するオブジェクトです。dmaClass_DocVersion クラスまたはそのサブクラスのオブジェクトです。
- Rendition **オブジェクト**
  文書を構成するファイルの形式を管理するオブジェクトです。
- ContentTransfer **オブジェクト**，ContentTransfers **オブジェクト**，ContentReference **オブジェクト，または** ContentFileLink **オブジェクト**
  文書の実体であるコンテントデータを管理するオブジェクトです。マルチファイル管理機能，リファレンスファイル管理機能および File Link 連携機能を使用しない場合，ContentTransfer オブジェクトを使用します。マルチファイル管理機能を使用する場合，ContentTransfers オブジェクトを使用します。リファレンスファイル管理機能を使用する場合，ContentReference オブジェクトを使用します。File Link 連携機能を使用する場合，ContentFileLink オブジェクトを使用します。

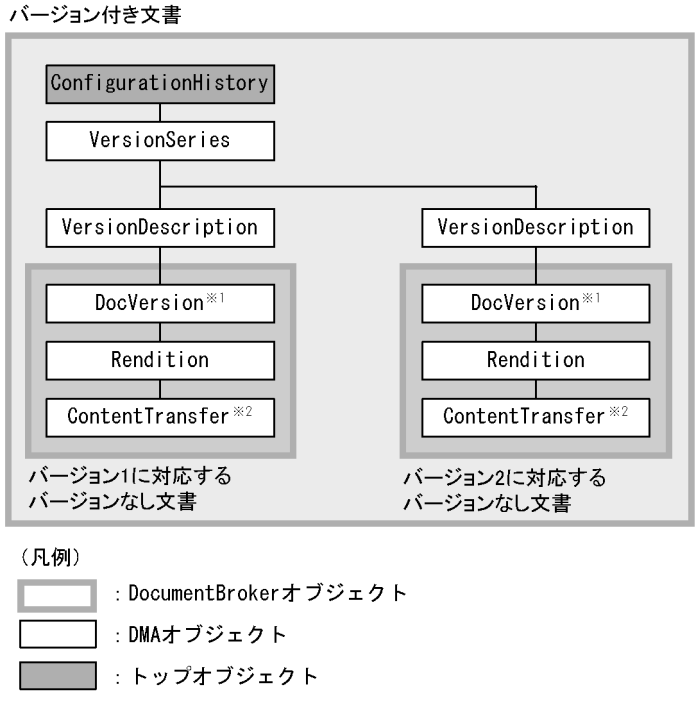
2.2.4 バージョン付き文書の管理に必要なオブジェクト

ここでは，バージョン付き文書のオブジェクト構成について説明します。

バージョン付き文書は，複数のバージョンを持ち，バージョンを管理する文書です。例えば，新規に作成した文書を編集する場合に，編集前の文書は残しておいて，編集後の文書を最新のバージョンとして管理するような運用が考えらる場合に，バージョン付き文書を使用します。

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 2-4 バージョン付き文書を構成する DMA オブジェクト



注※1 構成管理の対象となる場合はVersionTracedDocVersionオブジェクトになります。
注※2 マルチファイル管理機能を使用する場合はContentTransfersオブジェクト，
リファレンスファイル管理機能を使用する場合はContentReferenceオブジェクト，
File Link連携機能を使用する場合はContentFileLinkオブジェクトになります。

バージョン付き文書の個々のバージョンは，バージョンなし文書に対応しています。

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。なお，マルチファイル管理機能，リファレンスファイル管理機能または File Link 連携機能を使用する場合，DMA オブジェクトの構成が一部異なります。

- ConfigurationHistory **オブジェクト**
  文書の一連のバージョンを統括するオブジェクトです。dmaClass_ConfigurationHistory クラスまたはそのクラスのサブクラスのオブジェクトです。
- VersionSeries **オブジェクト**
  バージョンの構成を管理するオブジェクトです。
- VersionDescription **オブジェクト**
  VersionSeries オブジェクトと DocVersion オブジェクトを結び付け，DocVersion オブジェクトがどのバージョンに対応するかを管理するオブジェクトです。
- DocVersion **オブジェクトまたは** VersionTracedDocVersion **オブジェクト**
  バージョン付き文書での DocVersion オブジェクトは，文書の特定のバージョンに相当するオブジェクトです。dmaClass_DocVersion クラスまたはそのサブクラス（edmClass_VersionTracedDocVersion クラスを除く）のオブジェクトです。DocVersion オブジェクトで構成されるバージョン付き文書は，構成管理コンテナによる構成管理の対象になりません。
  VersionTracedDocVersion オブジェクトは，文書の特定のバージョンに相当するオブジェクトです。dmaClass_DocVersion クラスのサブクラスである，edmClass_VersionTracedDocVersion クラスまたはそのサブクラスのオブジェクトです。VersionTracedDocVersion オブジェクトで構成されるバージョン付き文書は，構成管理コンテナによる構成管理の対象になります。
- Rendition **オブジェクト**

文書を構成するファイルの形式を管理するオブジェクトです。

- ContentTransfer **オブジェクト，**ContentTransfers **オブジェクト，**ContentReference **オブジェクト，または** ContentFileLink **オブジェクト**
  文書の実体であるコンテントデータを管理するオブジェクトです。マルチファイル管理機能，リファレンスファイル管理機能および File Link 連携機能を使用しない場合，ContentTransfer オブジェクトを使用します。マルチファイル管理機能を使用する場合，ContentTransfers オブジェクトを使用します。リファレンスファイル管理機能を使用する場合，ContentReference オブジェクトを使用します。File Link 連携機能を使用する場合，ContentFileLink オブジェクトを使用します。
- Reservation **オブジェクト**
  チェックアウトとチェックインをすることで文書をバージョンアップする時点で，文書が予約されていることを示す一時的な DMA オブジェクトです。対象の文書をチェックアウトして予約すると，VersionSeries オブジェクトに Reservation オブジェクトが作成されます。バージョンを確定してチェックインをすると，Reservation オブジェクトは削除されて，VersionDescription オブジェクトに置き換えられます。

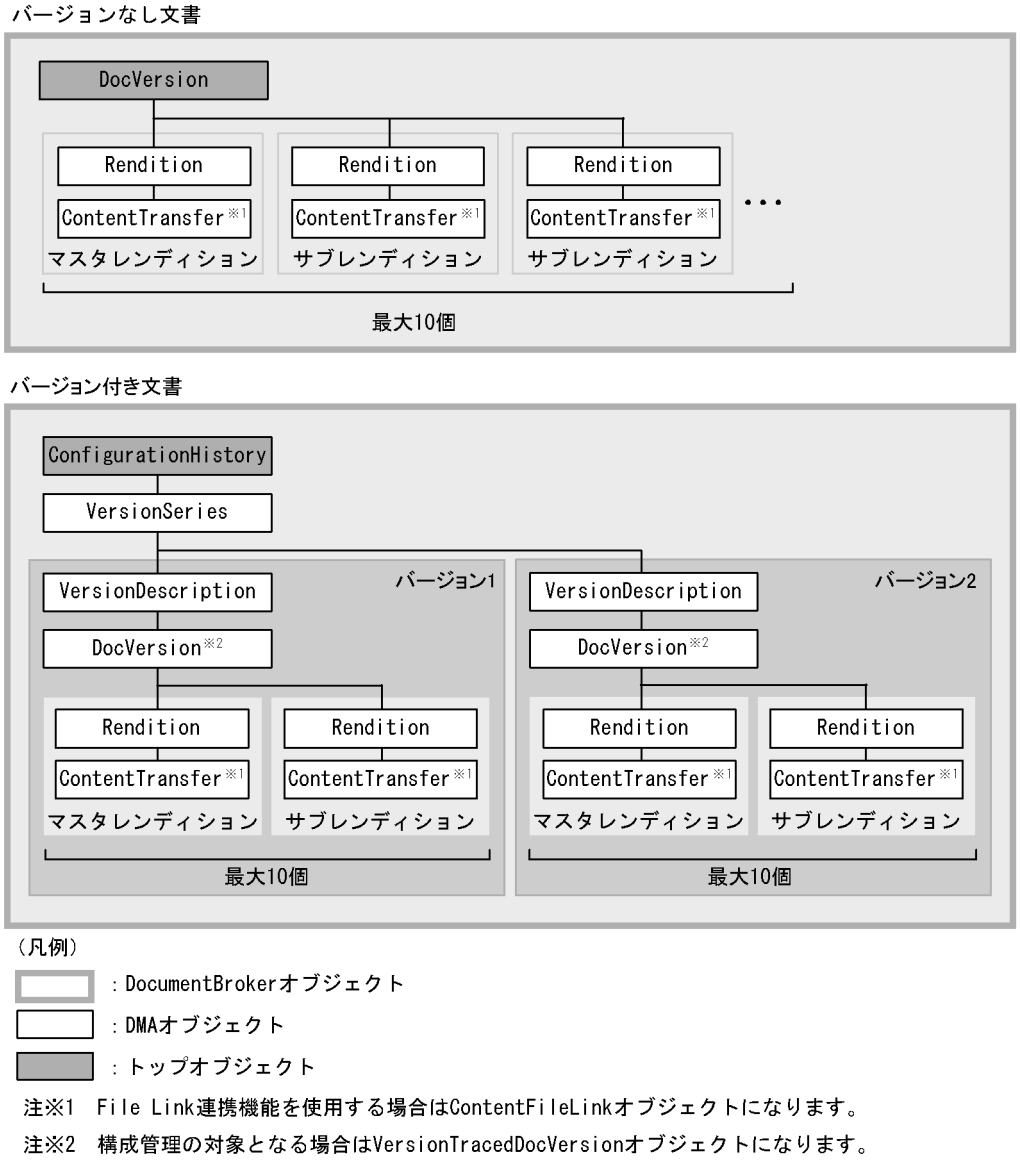
2.2.5 マルチレンディション文書の管理に必要なオブジェクト

ここでは，マルチレンディション文書のオブジェクト構成について説明します。

DocumentBroker 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できます。マルチレンディション文書では，一つの文書に 1 個のマスタレンディションと 9 個のサブレンディションを登録できます。また，マルチレンディションを使用する場合は，ファイルの形式に合わせてレンディションタイプを指定する必要があります。なお，同じレンディションタイプを重複して登録することはできません。

マルチレンディション文書を構成する DMA オブジェクトを次の図に示します。

図 2-5 マルチレンディション文書を構成する DMA オブジェクト



マルチレンディション文書を構成する DMA オブジェクトについて説明します。なお，File Link 連携機能を使用する場合，DMA オブジェクトの構成が一部異なります。

- **DocVersion オブジェクト**
  バージョンなし文書で，文書を表現するオブジェクトです。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 オブジェクトまたは VersionTracedDocVersion オブジェクトに複数のレンディションを登録できます。
- **DocVersion オブジェクトまたは VersionTracedDocVersion オブジェクト**
  バージョン付き文書で，文書の特定のバージョンに相当するオブジェクトです。
  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 DocVersion オブジェクトまたは VersionTracedDocVersion オブジェクトに複数のレンディションを登録できます。
- **Rendition オブジェクト**
  文書のレンディションタイプを管理するオブジェクトです。マスタレンディションを含めて最大 10 個登録できます。

- **ContentTransfer オブジェクトまたは ContentFileLink オブジェクト**
  文書のコンテントである文書ファイルを格納するためのオブジェクトです。Rendition オブジェクトに対応するコンテントを指定して登録します。
  File Link 連携機能を使用しない場合，ContentTransfer オブジェクトを使用します。File Link 連携機能を使用する場合，ContentFileLink オブジェクトを使用します。

2.2.6 コンテナの管理に必要なオブジェクト

ここでは，コンテナを利用した文書管理と，バージョンなしコンテナのオブジェクト構成について説明します。なお，構成管理コンテナについては，「2.2.7 構成管理コンテナの管理に必要なオブジェクト」で説明します。

(1) コンテナによる文書管理

コンテナを利用してオブジェクトを管理することを，**コンテインメント**といいます。コンテインメントは，次の 3 種類に区別できます。

- **直接型のコンテインメント**
  コンテナをフォルダのように利用して文書を管理する方法です。
- **参照型のコンテインメント**
  コンテナをインデクスのように利用して，コンテナと文書を関連づけて管理するような方法です。
- **構成管理型のコンテインメント**
  構成管理コンテナを利用してオブジェクトのバージョンを管理する方法です。

ここでは，直接型のコンテインメントと参照型のコンテインメントについて説明します。構成管理型のコンテインメントについては，「2.2.7 構成管理コンテナの管理に必要なオブジェクト」で説明します。

コンテナを利用してオブジェクトを管理する場合は，コンテナの用途によって次に示す点を考慮する必要があります。

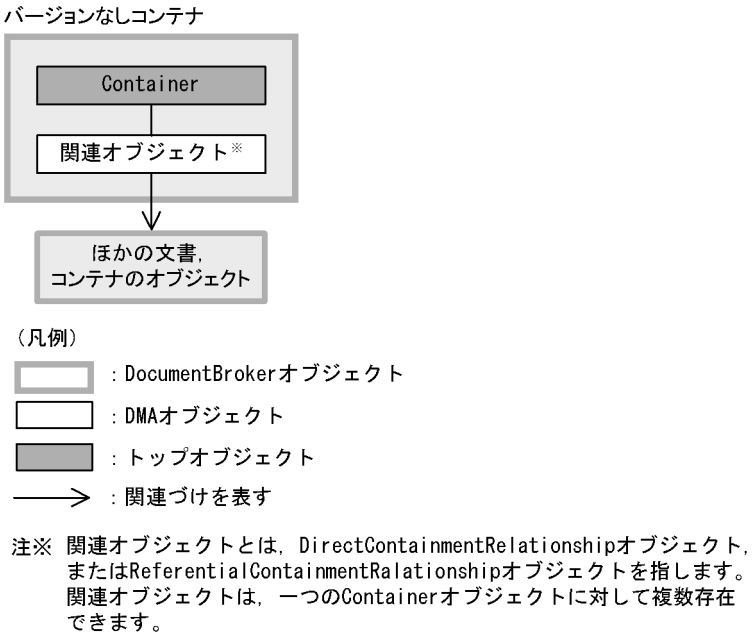
● 直接型のコンテインメントの場合
文書の管理体系に沿ったコンテナを設計したり，必要に応じてコンテナの階層構造を設計したりする必要があります。例えば，プロジェクト名をキーにしてコンテナを作成すると，そのプロジェクトに関連する文書をまとめて管理できます。プロジェクト内で作成，管理する文書はさまざまな種類があるので，種類ごとにコンテナの下にコンテナを作成することもできます。また，書籍やマニュアルのように複数の章から構成される文書は章単位で分割して，一つのコンテナでまとめて管理できます。

● 参照型のコンテインメントの場合
管理する文書の内容や種類を考慮した管理体系を設計する必要があります。例えば，書籍を管理するようなシステムを構築する場合を考えます。書籍を分類していくと，実用書，文学作品，哲学，史学，政治学などのように，幾つかのカテゴリによって整理できます。また，その書籍が書かれた時代，地域，著者ごとに分類することもできます。このように管理する文書を分類して，その分類結果に合わせた管理体系をコンテナの階層化によって表現することで，ある特定のカテゴリに属する文書を，内容にかかわらないでまとめて管理できます。このようなコンテナによる分類体系を作成すると，コンテナを閲覧することで，あるカテゴリに関連する文書の集合を参照できます。

(2) バージョンなしコンテナを構成する DMA オブジェクト

バージョンなしコンテナを構成する DMA オブジェクトを次の図に示します。

図 2-6 バージョンなしコンテナを構成する DMA オブジェクト



バージョンなしコンテナを構成する DMA オブジェクトについて説明します。

- Container **オブジェクト**
  バージョンなしコンテナを表すオブジェクトです。
- DirectContainmentRelationship **オブジェクト**
  直接型のコンテインメントを表すオブジェクトです。
- ReferentialContainmentRelationship **オブジェクト**
  参照型のコンテインメントを表すオブジェクトです。

2.2.7 構成管理コンテナの管理に必要なオブジェクト

ここでは，構成管理コンテナによる文書管理と，構成管理コンテナのオブジェクト構成について説明します。

(1) 構成管理コンテナによる文書管理

実用書やマニュアルなどの出版物は，複数の章から構成されます。このような出版物の章に相当する部分を文書として扱い，全体の構成を管理する場合は，**構成管理コンテナ**を使用します。構成管理コンテナは，「2.2.6 コンテナの管理に必要なオブジェクト」で説明したバージョンなしコンテナと同様に，複数の文書をまとめたり，分類したりするために使用します。さらに，構成管理コンテナは，包含されるオブジェクトのバージョンを固定して管理したり，常に最新のバージョンを追跡して管理したりすることで，包含されるオブジェクト全体の構成を管理できます。このように，構成管理コンテナを利用してオブジェクトのバージョンを管理することを，**構成管理型のコンテインメント**といいます。

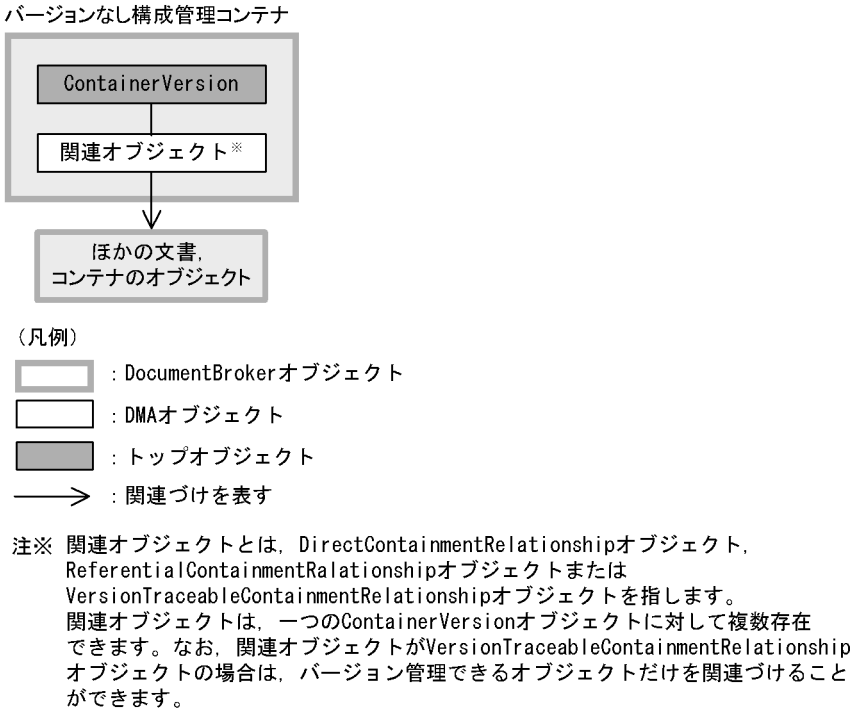
構成管理コンテナは，それ自身がバージョンを保持できるバージョン付き構成管理コンテナと，バージョンを保持しないバージョンなし構成管理コンテナに分けられます。バージョン付き構成管理コンテナを使用する場合，包含されるオブジェクトの全体の構成を，バージョン付き構成管理コンテナのバージョンごとにまとめて管理できます。バージョンなし構成管理コンテナを使用する場合，包含されるオブジェクトの全体の構成だけを管理します。

(2) バージョンなし構成管理コンテナを構成する DMA オブジェクト

バージョンなし構成管理コンテナのオブジェクト構成について説明します。

バージョンなし構成管理コンテナを構成する DMA オブジェクトを次の図に示します。

図 2-7 バージョンなし構成管理コンテナを構成する DMA オブジェクト



バージョンなし構成管理コンテナを構成する DMA オブジェクトについて説明します。

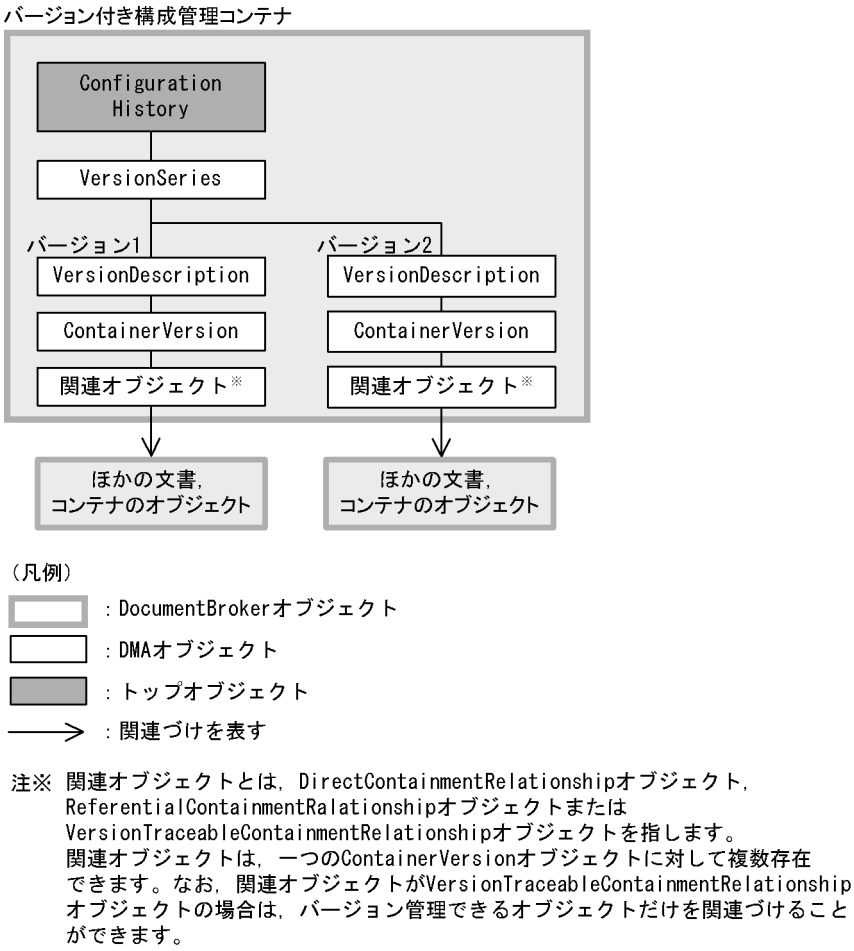
- ContainerVersion **オブジェクト**
  バージョンなし構成管理コンテナを表すオブジェクトです。
- DirectContainmentRelationship **オブジェクト**
  直接型のコンテインメントを表すオブジェクトです。
- ReferentialContainmentRelationship **オブジェクト**
  参照型のコンテインメントを表すオブジェクトです。
- VersionTraceableContainmentRelationship **オブジェクト**
  構成管理型のコンテインメントを表す場合に，コンテナ（VTContainer）とコンテナに包含されるオブジェクト（VTContainee）を関連づけるオブジェクトです。

(3) バージョン付き構成管理コンテナを構成する DMA オブジェクト

バージョン付き構成管理コンテナのオブジェクト構成について説明します。

バージョン付き構成管理コンテナを構成する DMA オブジェクトを次の図に示します。

図 2-8 バージョン付き構成管理コンテナを構成する DMA オブジェクト



バージョン付き構成管理コンテナを構成する DMA オブジェクトについて説明します。

- ConfigurationHistory **オブジェクト**
  構成管理コンテナの一連のバージョンを統括するオブジェクトです。
- VersionSeries **オブジェクト**
  バージョン構成を管理するオブジェクトです。
- VersionDescription **オブジェクト**
  VersionSeries オブジェクトと ContainerVersion オブジェクトを結び付け，ContainerVersion オブジェクトがどのバージョンに対応するかを管理するオブジェクトです。
- ContainerVersion **オブジェクト**
  バージョン付き構成管理コンテナの 1 バージョンに相当するオブジェクトです。
- DirectContainmentRelationship **オブジェクト**
  直接型のコンテインメントを表すオブジェクトです。
- ReferentialContainmentRelationship **オブジェクト**
  参照型のコンテインメントを表すオブジェクトです。
- VersionTraceableContainmentRelationship **オブジェクト**
  構成管理型のコンテインメントを表す場合に，コンテナ（VTContainer）とコンテナに包含されるオブジェクト（VTContainee）を関連づけるオブジェクトです。

2.2.8 文書間リレーションの管理に必要なオブジェクト

ここでは，文書間リレーションを表す DMA オブジェクトについて説明します。

文書間リレーションは，文書を表す次のオブジェクト間の関連づけに使用できます。

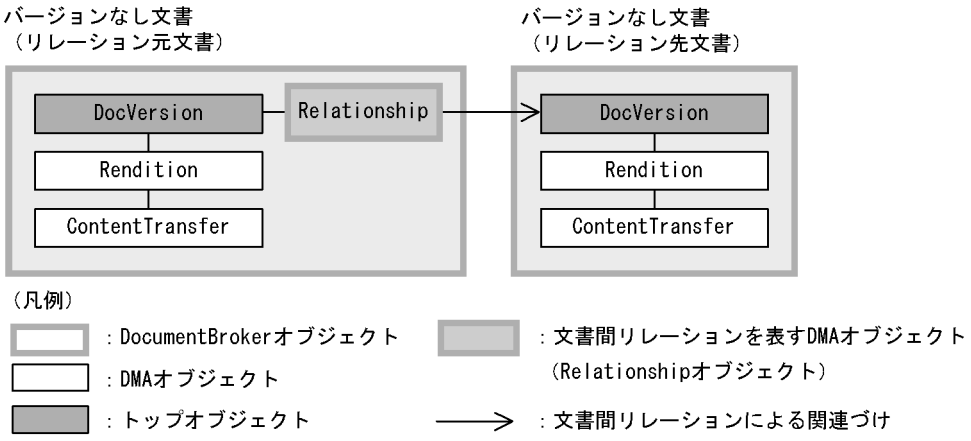
- バージョンなし文書
- バージョン付き文書

文書間リレーションを表す DMA オブジェクトについて説明します。

- **Relationship オブジェクト**
  文書間リレーションを表すオブジェクトです。Relationship オブジェクトは，リレーション元文書に従属するオブジェクトです。
  文書間リレーションは，リレーション元文書から操作します。リレーション元文書から文書間リレーションを設定した時点でこのオブジェクトが作成されて，リレーション元文書とリレーション先文書のトップオブジェクトを関連づけます。

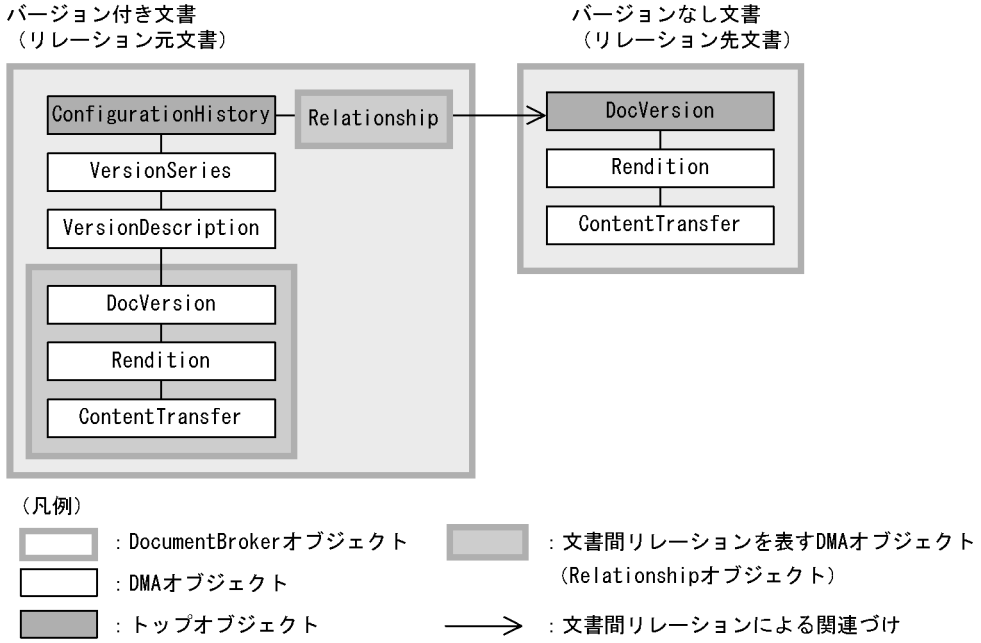
バージョンなし文書間に文書間リレーションを設定した場合のオブジェクトの構成を次の図に示します。

図 2-9 バージョンなし文書間に文書間リレーションを設定した場合のオブジェクトの構成



バージョン付き文書とバージョンなし文書間に文書間リレーションを設定した場合のオブジェクトの構成を次の図に示します。

図 2-10 バージョン付き文書とバージョンなし文書間に文書間リレーションを設定した場合のオブジェクトの構成



2.2.9 XML 文書の管理に必要なオブジェクト

XML 文書は，バージョンなし文書またはバージョン付き文書として作成，管理します。バージョンなし文書については，「2.2.3 バージョンなし文書の管理に必要なオブジェクト」を参照してください。バージョン付き文書については，「2.2.4 バージョン付き文書の管理に必要なオブジェクト」を参照してください。

ここでは，XML 文書を管理する場合の注意点と，XML 文書管理機能によって設定される情報について説明します。

XML 文書を管理する場合は，利用目的に応じて次に示す点に注意する必要があります。

● XML プロパティマッピング機能を使用する場合

マッピング先となる XML 文書のプロパティを追加する必要があります。
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ください。詳細については，「2.5 XML 文書管理機能を使用したシステム構築の検討」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する場合

バージョンなし文書またはバージョン付き文書の構成要素として，全文検索機能付き文書クラスを指定する必要があります。また，全文検索インデクス用プロパティを追加する必要があります。詳細については，「2.2.11 全文検索機能付き文書クラスを作成する場合に必要なオブジェクト」，および「2.5 XML 文書管理機能を使用したシステム構築の検討」を参照してください。

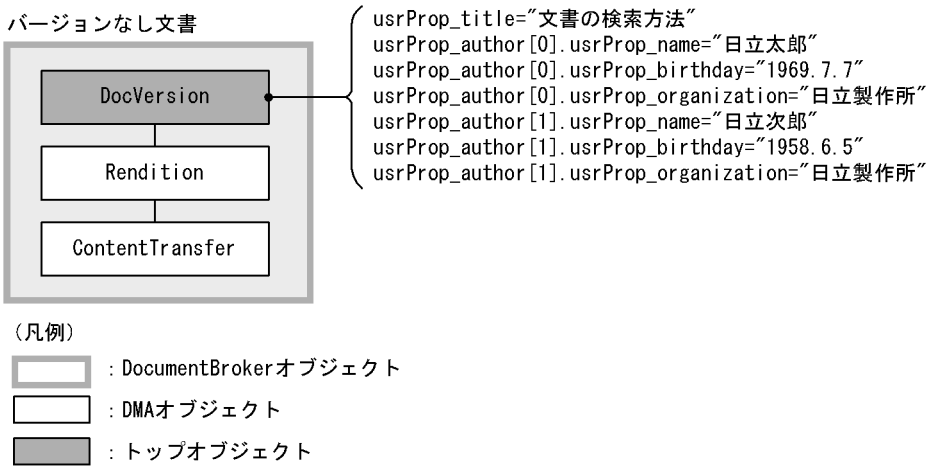
それぞれのオブジェクトに対して，XML 文書管理機能によって設定される情報について説明します。

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，DMA オブジェクトの 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 XML プロパティマッピングの例を次の図に示します。

図 2-11 バージョンなし文書に対する XML プロパティマッピングの例



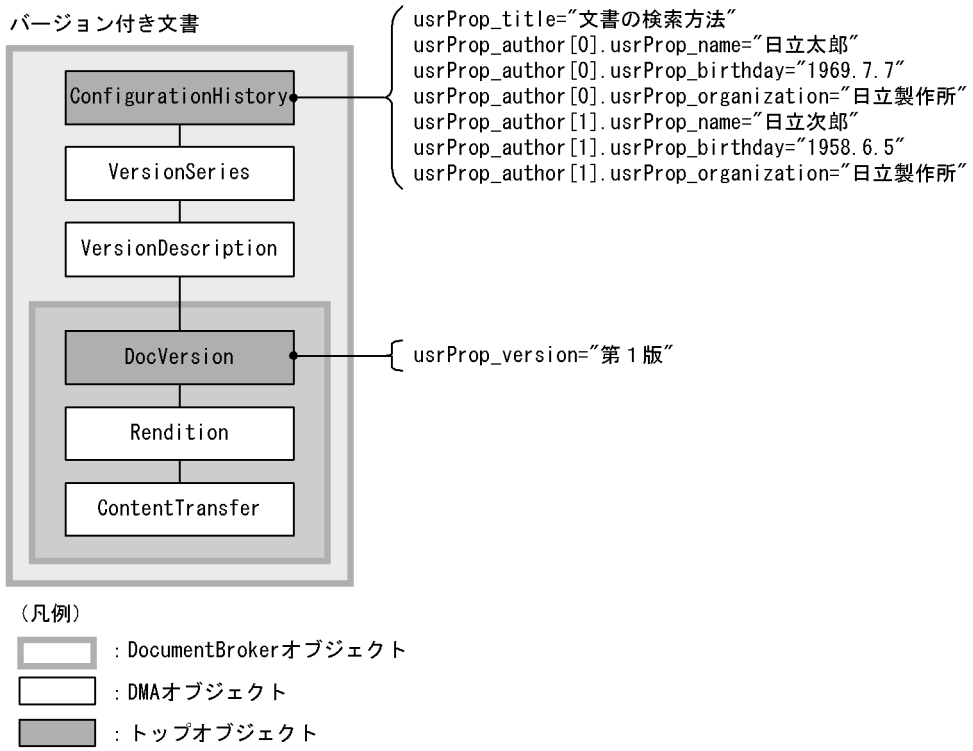
また，XML インデクスデータ作成機能で作成したインデクスデータを基に，DocVersion オブジェクトの全文検索インデクス用プロパティが設定できます。

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，DMA オブジェクトの ConfigurationHistory オブジェクトと DocVersion オブジェクト（または VersionTracedDocVersion 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，ConfigurationHistory 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，DocVersion オブジェクトのプロパティにマッピングできます。バージョン付き文書に対する XML プロパティマッピングの例を次の図に示します。

図 2-12 バージョン付き文書に対する XML プロパティマッピングの例



また，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，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。

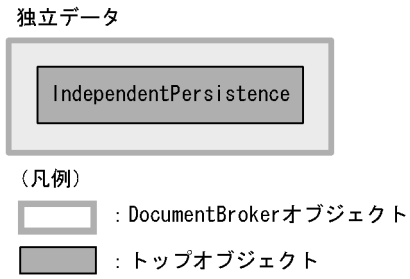
2.2.10 独立データの管理に必要なオブジェクト

ここでは，独立データの概要と，独立データのオブジェクト構成について説明します。

独立データは，文書やコンテナなどのほかのオブジェクトの体系に関係しない，独立したデータを扱うオブジェクトです。

独立データを表す DMA オブジェクトを次の図に示します。

図 2-13 独立データを表す DMA オブジェクト



- **IndependentPersistence オブジェクト**
  独立データを表すオブジェクトです。edmClass_IndependentPersistence クラスまたはそのサブクラスのオブジェクトです。

2.2.11 全文検索機能付き文書クラスを作成する場合に必要なオブジェクト

ここでは，全文検索機能付き文書クラスの作成に必要な DMA オブジェクトについて説明します。

全文検索は，全文検索機能を持ったクラスを基に作成された文書に対して実行できます。全文検索に必要な機能を持ったプロパティを追加したサブクラスのことを，**全文検索機能付き文書クラス**といいます。全文検索機能付き文書クラスは，dmaClass_DocVersion クラスのサブクラスとしてユーザが作成する必要があります。

なお，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に，概念検索用の機能を持ったプロパティも追加します。

全文検索の対象になるオブジェクトを作成する場合は，全文検索機能付き文書クラスを，次に示す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

文書の種類ごとにサブクラスを追加するような場合，そのサブクラスを全文検索機能付き文書クラスとするかどうかについても検討する必要があります。なお，全文検索機能付き文書クラスの追加については，「2.3.4　全文検索機能付き文書クラスの追加」を参照してください。

2.2.12 オブジェクトとデータベースの関連

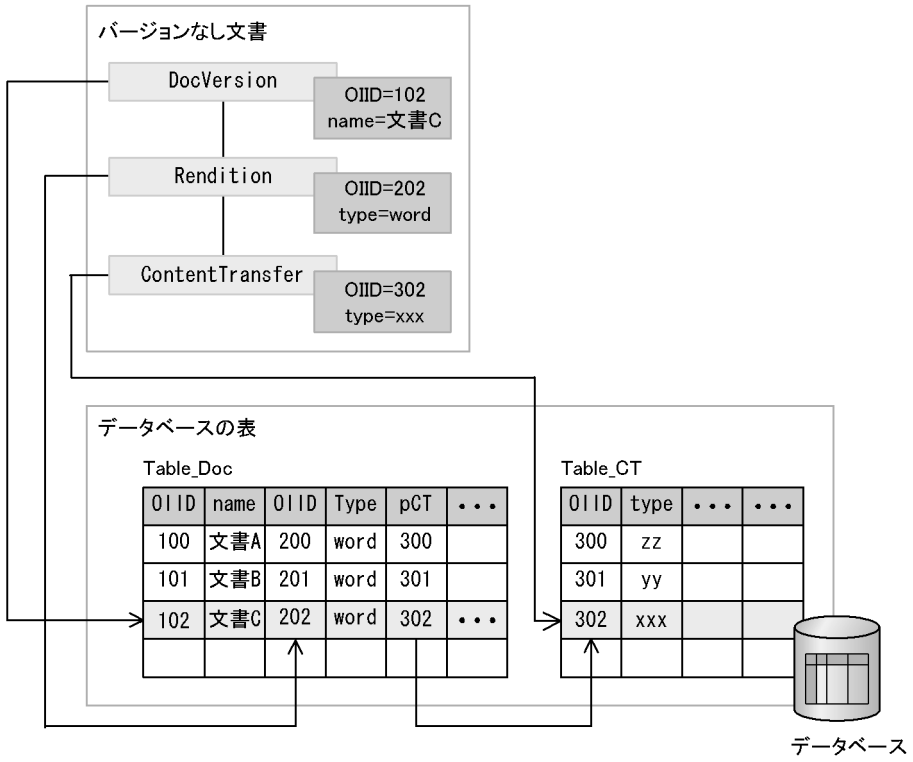
ここでは，DocumentBroker オブジェクトとデータベースの関係について説明します。

DocumentBroker オブジェクトは，複数の DMA オブジェクトを包含した形で表現されるメモリ上のオブジェクトです。データベースのスキーマは，DMA のオブジェクトモデルと対応して構成されています。なお，DMA オブジェクトは，各 DMA クラスに対応する表の 1 レコードになります。通常，DocumentBroker オブジェクトに対してメソッドを発行すると，構成している複数の DMA オブジェクトへのメソッドとして配置され，該当する表が操作されます。

例えば，dmaClass_DocVersion クラスには，一つの表が対応します。この場合，DocVersion オブジェクトは，その表のレコードとして格納されます。また，dmaClass_DocVersion クラスのサブクラスには，（dmaClass_DocVersion クラスに対応する表と異なる）別の表が対応します。したがって，このサブクラスのオブジェクトは，サブクラスに対応する表のレコードとして格納されます。

DocumentBroker オブジェクトおよび DMA オブジェクトとデータベースの対応を次の図に示します。

図 2-14　DocumentBroker オブジェクトおよび DMA オブジェクトとデータベースの対応



（凡例）

Table_Doc ： DocVersionオブジェクトを格納する表。dmaClass_DocVersionクラスに対応する。
Table_CT ： ContentTransferオブジェクトを格納する表。dmaClass_ContentTransferクラスに対応する。

この図では，バージョンなし文書を構成する DMA オブジェクトとデータベースとの関係を示しています。バージョンなし文書は，次に示す DMA オブジェクトを包含したオブジェクトです。

- DocVersion オブジェクト（dmaClass_DocVersion クラスのインスタンス）
- Rendition オブジェクト（dmaClass_Rendition クラスのインスタンス）
  ただし，ここでの Rendition オブジェクトは，マスタレンディションだけを表します。
- ContentTransfer オブジェクト（dmaClass_ContentTransfer クラスのインスタンス）

データベースの表は，各 DMA クラスに対応しており，DMA オブジェクトは，各 DMA クラスに対応する表の 1 レコードに相当します。ただし，文書中の一つ目の Rendition オブジェクトのレコードは，dmaClass_DocVersion クラスに対応する表と同じ表に格納されます。二つ目の Rendition オブジェクトのレコードは，dmaClass_Rendition クラスに対応する表に格納されます。この例では，レンディションが一つだけ存在するので，Rendition オブジェクトのレコードが格納されているのは，dmaClass_DocVersion クラスに対応する表と同じ表です。

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 によって表現します。この場合，DocVersion オブジェクトは，ContentTransfer オブジェクトの OIID の値（この例では「302」です。実際はこれと異なる値が格納されます）を保持することで，参照先の ContentTransfer オブジェクトを特定します。

2.3 追加するクラスおよびプロパティの検討

この節では，クラスとプロパティの追加の概要，サブクラスとプロパティを追加できるクラス，全文検索機能を使用するために作成する全文検索機能付き文書クラスの追加について説明します。また，具体的な例を示してクラスとプロパティの追加について説明します。

DocumentBroker では，文書の管理方法に合わせてサブクラスを作成したり，クラスにプロパティを追加したりできます。この節の内容を参照して，サブクラスの作成とプロパティの追加について検討してください。なお，実際にサブクラスを作成して，クラスにプロパティを追加するためには，定義情報ファイルを作成する必要があります。定義情報ファイルの作成については，「3.11.2　データベースシステムの設定に必要なファイル」，および「4.7　定義情報ファイル」を参照してください。

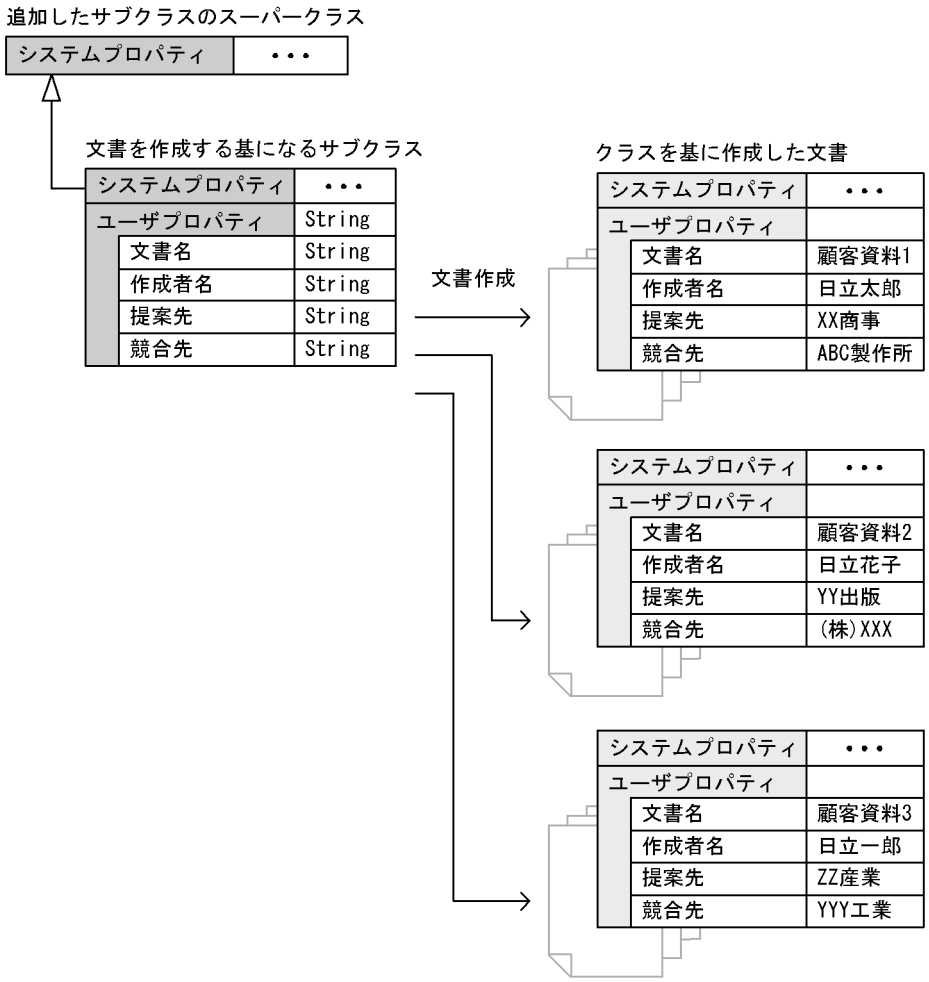
2.3.1 クラスおよびプロパティの追加

DocumentBroker で扱うオブジェクトは，オブジェクト指向に基づいて作成されるため，あるクラスを基に作成されるオブジェクトは，そのクラスの**インスタンス**として扱われ，クラスからすべてのプロパティを継承します。すなわち，あるクラスを基に作成したオブジェクトは，同じ種類のオブジェクトであり，同じプロパティが付加されることになります。

幾つかのクラスは，文書の種類や管理方法に合わせてサブクラスを追加できます。DocumentBroker が提供するクラスやユーザが追加したサブクラスには，必要に応じてプロパティを追加できます。プロパティには，文書に共通的に追加すると便利な情報やその文書に固有の情報など，ユーザがオブジェクトの管理情報として保持したい情報を設定できます。また，追加したプロパティをキーにしてオブジェクトを検索できます。オブジェクトをどのように管理するか，どのように検索するかを考慮して，必要なサブクラスを作成し，管理情報として利用できるプロパティや検索のキーになるようなプロパティを追加して，プロパティの値を設定する必要があります。

DocumentBroker で管理する文書はその種類ごとに分類できます。例えば，ある製品の購入を勧めるためのプレゼンテーション資料を管理する場合，プレゼンテーション資料作成用のサブクラスを定義して，そのサブクラスに対して，必要なプロパティを追加するような運用が考えられます。プレゼンテーション資料作成用に定義したクラスと文書の作成例を次の図に示します。

図 2-15 プレゼンテーション資料作成用に定義したクラスと文書の作成例



この図では，プレゼンテーション資料作成用に追加定義したサブクラスに，次に示すプロパティを追加しています。

- **文書管理に必要な一般的な管理情報**
  「文書名」と「作成者名」という String 型のプロパティを追加しています。
- **プレゼンテーション資料としての付加情報**
  プレゼンテーション先の顧客名を管理する「提案先」というプロパティと競合する会社名を管理する「競合先」というプロパティを追加しています。

このクラスを基に作成した文書（プレゼンテーション資料）は，共通のプロパティを持ちます。プロパティに値を入れることによって，それぞれの文書の内容をプロパティから判別したり，文書の検索性を向上させたりできます。

2.3.2 サブクラスの追加

DocumentBroker が提供しているクラスに対して，必要に応じてサブクラスを追加できます。ここでは，

サブクラスを追加できるクラスとサブクラスを追加する場合の注意事項について説明します。

(1) サブクラスを追加できるクラス

サブクラスを追加できるクラスは次のとおりです。

- dmaClass_ConfigurationHistory クラス
- dmaClass_Container クラス
- dmaClass_DocVersion クラス
- edmClass_ComponentDocVersion クラス
- edmClass_ContainerVersion クラス
- edmClass_IndependentPersistence クラス
- edmClass_Struct クラス
- edmClass_VersionTraceableContainer クラス
- edmClass_VersionTracedDocVersion クラス
- edmClass_VersionTracedComponentDocVersion クラス

(2) サブクラスを追加する場合の注意事項

- 追加するサブクラスの名称には「dmaClass」および「edmClass」で始まる文字列は使用できません。
- サブクラスの名称は重複しないようにしてください。
- 追加したサブクラスはスーパークラスのすべてのプロパティを継承します。
- dmaClass_DocVersion クラスのサブクラスとして追加したサブクラスは，問い合わせ中に参照される検索可能なクラスとなります。

2.3.3 プロパティの追加

DocumentBroker が提供するクラスとユーザが定義したクラスに対して，プロパティを追加できます。例えば，dmaClass_DocVersion クラスに「文書名」のプロパティを追加するような場合です。ここでは，プロパティを追加できるクラス，プロパティのデータ型，プロパティの基本単位，およびプロパティを追加する場合の注意事項について説明します。

(1) プロパティを追加できるクラス

プロパティを追加できるクラスは次のとおりです。

- dmaClass_DirectContainmentRelationship クラス
- dmaClass_ReferentialContainmentRelationship クラス
- edmClass_VersionTraceableContainmentRelationship クラス
- edmClass_PublicACL クラス
- edmClass_Relationship クラス
- ユーザが追加したサブクラス

(2) データ型の対応

追加できるプロパティのデータ型と HiRDB で対応するデータ型について次の表に示します。

表 2-2 追加できるプロパティのデータ型と HiRDB で対応するデータ型

追加できるプロパティのデータ型	HiRDB で対応するデータ型
String	• MVARCHAR • FREEWORD ※
Integer32	INTEGER

追加できるプロパティのデータ型	HiRDB で対応するデータ型
Boolean	

注※

全文検索機能付き文字列型プロパティは，HiRDB Text Search Plug-in の FREEWORD 型を使用します。

(3) プロパティの基本単位

プロパティは，データ型に従った値を 1 個持つか複数個持つかが決められています。これを**基本単位**といいます。基本単位には，次の種類があります。

値を 1 個持つプロパティ

データ型に従った値を一つだけ持つプロパティです。この基本単位は Scalar **型**です。

値を複数個持つプロパティ

データ型に従った複数の値を一つの値として持つプロパティです。この基本単位は List 型，Enumeration 型および VariableArray 型です。

List **型**

すべてのデータ型の値をリスト化できます。ただし，異なる種類のデータ型は混在できません。したがって，List 型のプロパティは，ListOfInteger32 オブジェクト，ListOfString オブジェクトのように，データ型ごとに値をリスト化するオブジェクトへのリファレンスを値として持ちます。

Enumeration **型**

すべてのデータ型の値を列挙します。ただし，異なる種類のデータ型は混在できません。また，DocumentBroker は Object 型の値だけを列挙できます。したがって，Enumeration 型のプロパティは，EnumerationOfObject オブジェクトのようにデータ型に従って値を列挙するオブジェクトへのリファレンスを値として持ちます。

VariableArray **型**

すべてのデータ型の値の一次元配列です。この一次元配列は可変長です。List 型，Enumeration 型と同様に，異なる種類のデータ型は混在できません。また DocumentBroker は Object 型の値だけを配列要素にできます。したがって，VariableArray 型のプロパティは，VariableArrayOfObject オブジェクトのようにデータ型に従った値を一次元配列として持つオブジェクトへのリファレンスを値として持ちます。

なお，追加できるプロパティは，基本単位が Scalar 型と VariableArray 型であるプロパティだけです。

(4) プロパティを追加する場合の注意事項

- 追加するプロパティの名称には「dmaProp」および「edmProp」で始まる文字列は使用できません。
- 全文検索機能付き文字列型プロパティを追加する場合には，メタ情報の追加コマンド（EDMAddMeta）を使用してください。文書空間の定義コマンド（EDMCDefDocSpace）および文書空間の構築コマンド（EDMCBuildDocSpace）の使用では，全文検索機能付き文字列型プロパティを追加できません。

2.3.4 全文検索機能付き文書クラスの追加

ここでは，全文検索機能を利用するための全文検索機能付き文書クラスの追加について説明します。

構造を持たないテキスト形式の文書（**プレーンテキスト**）および構造を持つ文書に対して全文検索を実行するためには，全文検索の対象となるサブクラスを追加する必要があります。全文検索の対象となるサブ

クラスのことを，全文検索機能付き文書クラスといいます。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のことです。

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に，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。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のことを特に，**概念検索機能付き文書クラス**といいます。

全文検索の対象になるオブジェクトを作成する場合は，全文検索機能付き文書クラスを，次に示す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

全文検索機能付き文書クラスに追加するプロパティは，使用する検索機能によって異なります。ここでは，プレーンテキストおよび構造を持つ文書を対象として，次に示す文書クラスの定義について説明していきます。

- 全文検索機能付き文書クラス
- 概念検索機能付き文書クラス

なお，構造を指定した検索機能を使用する文書クラスの定義については，それぞれの文書クラスの定義に含めて説明します。

(1) 全文検索機能付き文書クラスの定義

ここでは，検索タームを指定する全文検索機能を使用する文書クラスの追加定義に必要なプロパティについて説明します。

(a) 追加するプロパティ

全文検索機能を使用する文書クラスを作成する場合，dmaClass_DocVersion クラスのサブクラスとして，次に示すプロパティを追加した文書クラスを作成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

全文検索インデクス用プロパティを次の表に示します。

表 2-3 全文検索インデクス用プロパティ

プロパティ	全文検索	構造指定検索
edmProp_TextIndex	○	×
edmProp_StIndex	○	○
edmProp_ConceptTextIndex	○	×
edmProp_ConceptStIndex	○	○
edmProp_Content（Version 1 互換用）	○	○

（凡例）
○：実行できる。
×：実行できない。

(b) 必要に応じて追加するプロパティ

全文検索機能を使用する文書クラスの定義には，次に示すプロパティのうち，必要なプロパティを追加してください。

1. edmProp_TextIndex プロパティ
   検索エンジンに対して，全文検索用の登録情報を参照することを示すためのプロパティです。プレーンテキストの登録情報を参照する場合は，このプロパティを追加します。
2. edmProp_StIndex プロパティ
   検索エンジンに対して，文書の構造を指定した全文検索用の登録情報を参照することを示すためのプロパティです。XML 文書の登録情報を参照する場合は，このプロパティを追加します。
3. edmProp_ConceptTextIndex プロパティ
   検索エンジンに対して，全文検索用または概念検索用の登録情報を参照することを示すためのプロパティです。プレーンテキストの登録情報を参照する場合は，このプロパティを追加します。
4. edmProp_ConceptStIndex プロパティ
   検索エンジンに対して，文書の構造を指定した全文検索用または概念検索用の登録情報を参照することを示すためのプロパティです。XML 文書の登録情報を参照する場合は，このプロパティを追加します。
5. edmProp_Content プロパティ
   全文検索エンジンに対して，全文検索用および文書の構造を指定した全文検索用の登録情報を参照することを示すためのプロパティです。プレーンテキスト，XML 文書のプロパティとして追加できます。
   なお，このプロパティは，Version 1 との互換用のプロパティです。
6. edmProp_DocLength プロパティ
   検索した文書の長さを示すプロパティです。バイト単位で示します。
7. edmProp_ContentIndexStatus プロパティ
   全文検索機能を使用する文書クラスに，全文検索インデクスの登録状態を示します。
8. edmProp_Score プロパティ
   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。全文検索の結果のスコアを取得する場合に必要となります。算出されたスコアは検索結果集合内で正規化されて，値として設定されます。
9. edmProp_RawScore プロパティ
   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。全文検索の結果のスコアを取得する場合に必要となります。算出されたスコアは検索結果集合内で正規化されないで，値として設定されます。

文書の管理方法に合わせて，プロパティを追加してください。

全文検索機能を使用する文書クラスを定義する場合

1.，2.，3.，4.，および 5. のプロパティのうち，どれか一つを必ず追加してください。また，6. および 7. のプロパティを必ず追加してください。
なお，5. のプロパティは，Version 1 との互換用として，1.，2.，3.，および 4. のプロパティの代わりに追加できます。

全文検索結果のスコアを取得する場合

8. および 9. のプロパティを必ず追加してください。

(2) 概念検索機能付き文書クラスの定義

ここでは，概念検索機能を使用する文書クラスの追加定義に必要なプロパティについて説明します。

(a) 追加するプロパティ

概念検索機能を使用する文書クラスを作成する場合，dmaClass_DocVersion クラスのサブクラスとして，

次に示すプロパティを追加した文書クラスを作成する必要があります。

- 全文検索インデクス用プロパティ（概念検索機能用）
- edmProp_DocLength プロパティ
- edmProp_ContentIndexStatus プロパティ
- edmProp_Score プロパティ
- edmProp_RawScore プロパティ
- edmProp_ScoreConcept プロパティ

全文検索インデクス用プロパティ（概念検索機能用）を次の表に示します。

表 2-4　全文検索インデクス用プロパティ（概念検索機能用）

プロパティ	概念検索	全文検索	構造指定検索
edmProp_ConceptTextIndex	○	○	×
edmProp_ConceptStIndex	○	○	○

（凡例）
○：実行できる。
×：実行できない。

(b) 必要に応じて追加するプロパティ

概念検索機能を使用する文書クラスの定義には，次に示すプロパティのうち，必要なプロパティを追加してください。

1. edmProp_ConceptTextIndex プロパティ
   検索エンジンに対して，概念検索用の登録情報を参照することを示すためのプロパティです。プレーンテキストの登録情報を参照する場合は，このプロパティを追加します。
2. edmProp_ConceptStIndex プロパティ
   検索エンジンに対して，文書の構造を指定した概念検索用の登録情報を参照することを示すためのプロパティです。XML 文書の登録情報を参照する場合は，このプロパティを追加します。
3. edmProp_DocLength プロパティ
   検索した文書の長さを示すプロパティです。バイト単位で示します。
4. edmProp_ContentIndexStatus プロパティ
   概念検索機能を使用する文書クラスに，概念検索インデクスの登録状態を示します。
5. edmProp_Score プロパティ
   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。全文検索の結果のスコアを取得する場合に必要となります。算出されたスコアは検索結果集合内で正規化されて，値として設定されます。
6. edmProp_RawScore プロパティ
   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。全文検索の結果のスコアを取得する場合に必要となります。算出されたスコアは検索結果集合内で正規化されないで，値として設定されます。
7. edmProp_ScoreConcept プロパティ
   概念検索結果のスコアを取得するための形式的なプロパティです。

文書の管理方法に合わせて，プロパティを追加してください。

概念検索機能を使用する文書クラスを追加する場合

1. または 2. のプロパティを必ず追加してください。また，3. および 4. のプロパティを必ず追加してください。

全文検索結果のスコアを取得する場合

5. および 6. のプロパティを必ず追加してください。

概念検索結果のスコアを取得する場合

7. のプロパティを必ず追加してください。

2.3.5 クラスおよびプロパティの追加例

ここでは，具体的な例を示して，クラスとプロパティの追加について説明します。

文書を作成，管理するために追加するクラス（dmaClass_DocVersion クラスおよび edmClass_VersionTracedDocVersion クラスのサブクラス）とプロパティについて，例に沿って説明します。

（例）

「問題点管理ドキュメント」と「設計ドキュメント」という 2 種類の文書を管理するためにクラスを追加します。この例で扱う文書の詳細情報を次の表に示します。

表 2-5 文書の詳細情報

文書の種類	概要	付加情報	管理方法
問題点管理ドキュメント	製品の開発時の問題点を管理するための文書	• 問題点分類 • 問題点発生日時 • 検討結果 • 解決日時 • 対策日	直接型のコンテインメントを使用して管理する。
設計ドキュメント	製品開発の基となる設計書	• 文書名 • 執筆者 • 特許情報 • 執筆日時	構成管理型のコンテインメントを使用して，複数の設計書の構成を管理する。

この 2 種類の文書を管理するためのクラスを次のように設計します。追加するサブクラスとスーパークラスの対応を次の表に示します。

表 2-6 追加するサブクラスとスーパークラスの対応

管理する文書	追加するサブクラス	スーパークラス
問題点管理ドキュメント	usrClass_ProbInfoDoc	dmaClass_DocVersion
設計ドキュメント	usrClass_DesignDoc	edmClass_VersionTracedDocVersion

問題点管理ドキュメントは，直接型のコンテインメントを使用して管理します。したがって，dmaClass_DocVersion クラスのサブクラスとして usrClass_ProbInfoDoc クラスを追加して管理します。

設計ドキュメントは，バージョン付き構成管理コンテナを使用して構成を管理する文書として作成します。したがって，edmClass_VersionTracedDocVersion クラスのサブクラスとして usrClass_DesignDoc クラスを追加して管理します。

次に，文書に対して付加する情報を各クラスのプロパティとして追加します。文書の付加情報とプロパティの対応を次の表に示します。

表 2-7 付加情報とプロパティの対応

クラス名	付加情報	プロパティ名	データ型
usrClass_ProbInfoDoc	問題点分類	usrProp_Category	String 型

クラス名	付加情報	プロパティ名	データ型
	問題点発生日時	usrProp_DateOfProb	Integer32 型
	検討結果	usrProp_Status	Boolean 型
	解決日時	usrProp_DateOfResolve	Integer32 型
usrClass_DesignDoc	文書名	usrProp_DocName	String 型
	執筆者	usrProp_DocWriter	String 型
	特許情報	usrProp_PatentInfo	String 型
	執筆日時	usrProp_DateOfWrite	Integer32 型

なお，追加するサブクラスは，全文検索機能付き文書クラスとして追加します。したがって，この表に示したプロパティのほかに，全文検索用のプロパティを追加する必要があります。全文検索機能付き文書クラスに追加するプロパティについては，「2.3.4　全文検索機能付き文書クラスの追加」を参照してください。

このように，管理する文書の種類や付加情報などから必要なクラスとプロパティを決定してから，その設計情報を基に必要なクラスおよびプロパティを追加してください。

2.4 文書空間で使用する文字コード種別の検討

この節では，文書空間で使用する文字コード種別について説明します。

DocumentBroker の文書空間では，次のどちらかの文字コード種別を使用できます。一つの文書空間では複数の文字コード種別を使用できません。

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

DocumentBroker クライアント，DocumentBroker サーバ，およびデータベースで使用する文字コード種別は，必ず一致させてください。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。

また，文書空間で使用する文字コード種別が UTF-8 の場合，DocumentBroker で使用できる機能やコマン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機能やコマンドの詳細は，「付録 I　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機能およびコマンド」を参照してください。

(1) データベースの文字コード種別の設定

データベース（HiRDB）の文字コード種別は，データベースの環境構築時に HiRDB の動作環境の設定コマンド（pdntenv）で設定します。HiRDB の動作環境の設定コマンドの詳細は，マニュアル「HiRDB コマンドリファレンス」を参照してください。

(2) 文書空間の文字コード種別の設定

文書空間の文字コード種別は，DocumentBroker の環境構築時に次のどちらかで設定します。

- メタ情報の初期設定コマンド（EDMInitMeta）の -C オプション
- 文書空間情報ファイルの DocSpaceCharacterSet エントリ（文書空間の定義コマンド（EDMCDefDocSpace）を使用するとき）

メタ情報の初期設定コマンド（EDMInitMeta）については，「7.3　コマンドの文法」の「EDMInitMeta（メタ情報の初期設定）」を参照してください。文書空間情報ファイルについては，「4.16　文書空間情報ファイル」を参照してください。

2.5 XML 文書管理機能を使用したシステム構築の検討

この節では，XML を使用した文書管理について説明します。XML 文書管理機能を使用したシステムを構築する場合に参照してください。なお，Linux の場合，XML 文書管理機能は使用できません。

2.5.1 DocumentBroker で提供する XML 文書管理機能

DocumentBroker 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トとして管理できます。

また，DocumentBroker は，XML 文書を管理するために次のような機能を提供しています。

● 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

上記の機能を使用する場合に，前提プログラムとして必要なプログラムを次に示します。

XML プロパティマッピング機能を使用する場合

HiRDB Adapter for XML

XML インデクスデータ作成機能を使用する場合

- HiRDB Adapter for XML
- Preprocessing Library for Text Search

なお，XML 文書を全文検索する場合には，HiRDB Text Search Plug-in も必要です。

2.5.2 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について説明します。

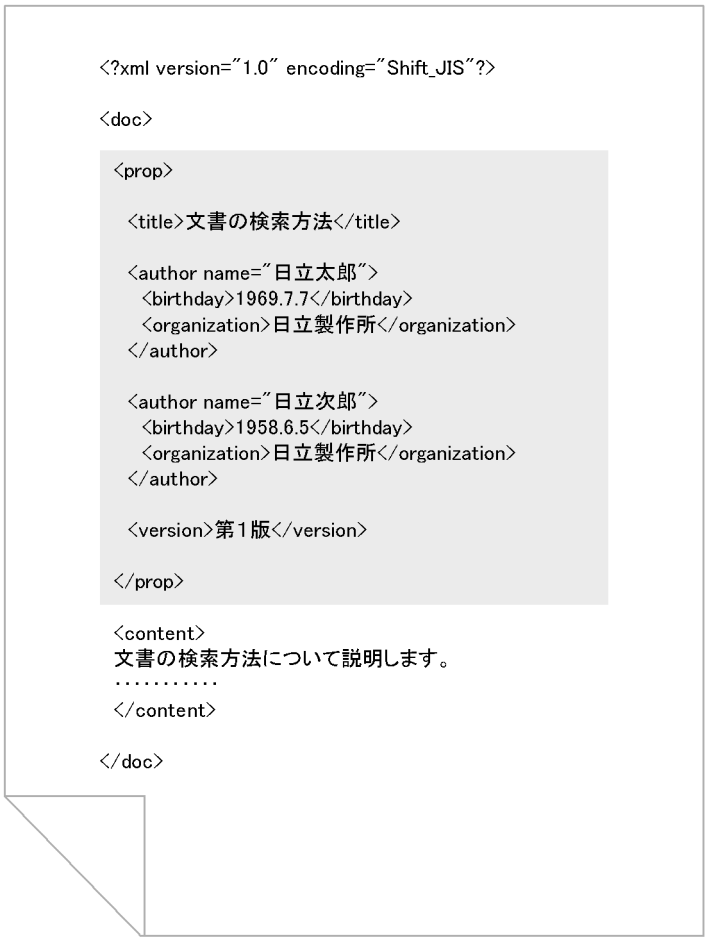
(1) XML プロパティマッピング機能とは

XML プロパティマッピング機能とは，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設定するための情報を作成する機能です。この情報を 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，次の検索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場合のキーにする。

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 2-16　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



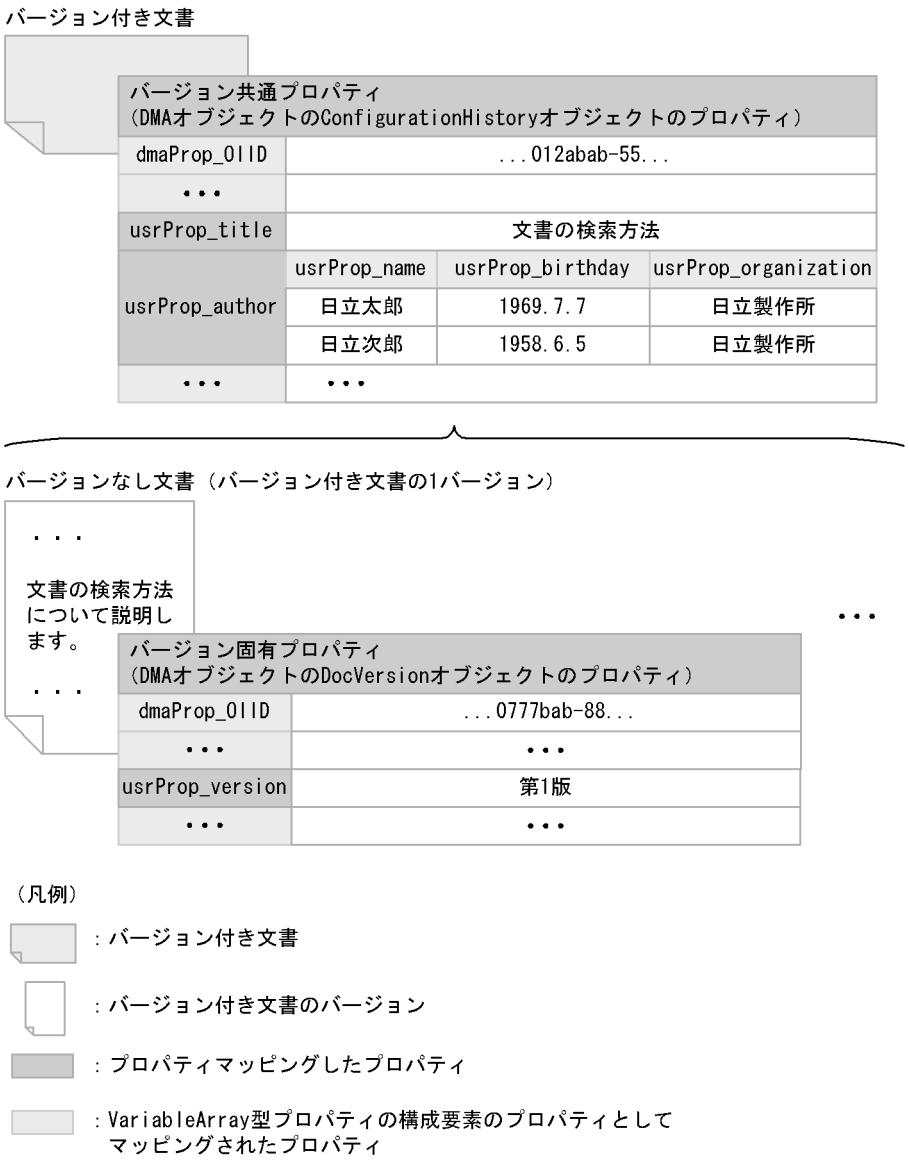
```
<?xml version="1.0" encoding="Shift_JIS"?>

<doc>

 <prop>

  <title>文書の検索方法</title>

  <author name="日立太郎">
    <birthday>1969.7.7</birthday>
    <organization>日立製作所</organization>
  </author>

  <author name="日立次郎">
    <birthday>1958.6.5</birthday>
    <organization>日立製作所</organization>
  </author>

  <version>第1版</version>

 </prop>

 <content>
 文書の検索方法について説明します。
 ・・・・・・・・・・
 </content>

</doc>
```

<prop> タグで囲まれた内容には，タイトル，著者名，著者の所属する組織名，版についての情報などが，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ト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ン付き文書のプロパティ（DMA オブジェクトの ConfigurationHistory オブジェクトのプロパティ）として，バージョン情報はバージョンなし文書（バージョン付き文書の一つのバージョン）のプロパティ（DMA オブジェクトの DocVersion オブジェクトのプロパティ）としてマッピングすることにします。

図 2-16 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 2-17　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> タグ間の情報は，バージョン付き文書の usrProp_title プロパティにマッピングしています。<author> タグ間の情報は，複数の情報を表すタグを含むので，VariableArray 型のプロパティである usrProp_author プロパティにマッピングしています。<author> タグの属性である，name の値，<author> タグ間に含まれる <birthday> タグおよび <organization> タグ間の情報は，VariableArray 型のプロパティの構成要素を表す usrProp_name プロパティ，usrProp_birthday プロパティおよび usrProp_organization プロパティにそれぞれマッピングしています。

(2) XML プロパティマッピング機能の処理内容

XML プロパティマッピング機能は，ユーザの定義に従って XML 文書のタグ間の文字列，またはタグの属性値から必要な情報を取得し，これを DocumentBroker のプロパティリストに変換します。DocumentBroker は，プロパティリストを生成するライブラリを提供します。XML 文書管理機能を実現するためには，登録する XML 文書の文書構造を表現するマッピング元 XML タグ定義，および XML 文書中のタグ名とその内容をマッピングするクラス名，プロパティ名の対応を定義したプロパティマッピング

定義をユーザが作成する必要があります。ユーザが作成したプロパティマッピング定義で，HiRDB Adapter for XML が提供する，機能およびマッピング定義情報に変換するコマンドを利用して XML 文書を解析し，DocumentBroker のプロパティリストを生成します。このライブラリを利用して生成されたプロパティリストを C++ クラスライブラリの引数に指定することで，プロパティマッピングができます。

(3) XML プロパティマッピング機能を使用するための設定

XML プロパティマッピング機能を使用する場合は，登録する XML ファイルの論理構造を明確にして，マッピング先となる XML 文書にプロパティを追加する必要があります。また，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，次のファイルがあります。

- マッピングセット定義ファイル（XMS）※1
- マッピング定義ファイル（XMP）※1
- マッピング元 XML タグ定義ファイル（DCD）※2
- プロパティマッピング定義ファイル（DPM）※2

注※1

DocumentBroker のコマンド，または HiRDB Adapter for XML によって生成される定義ファイルです。

注※2

ユーザが作成する定義ファイルです。

XML 文書管理機能を使用するための定義ファイルの作成および生成については，「3.18　XML 文書管理機能を使用する場合の設定」を参照してください。

(4) XML プロパティマッピング機能の処理対象となる文字コード

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
- EUC
- UTF-8
- UTF-16

ただし，プロパティリストの文字列は，AIX の場合，すべて Shift-JIS に変換されます。

2.5.3　XML インデクスデータ作成機能

ここでは，XML インデクスデータ作成機能について説明します。

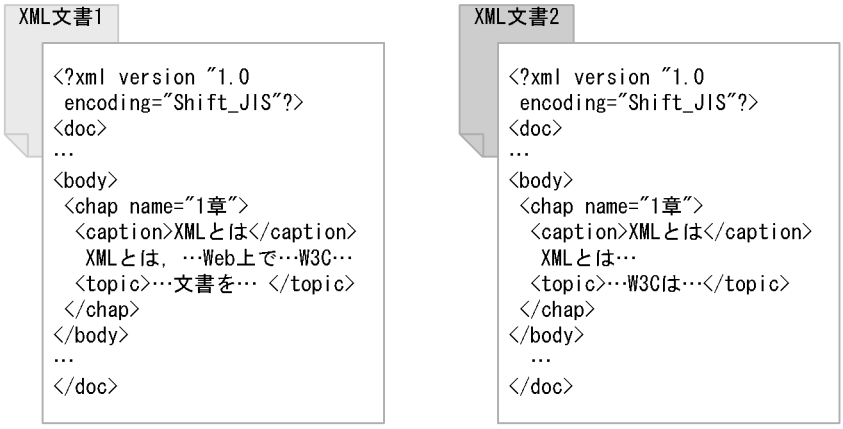
(1) XML インデクスデータ作成機能とは

XML インデクスデータ作成機能は，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，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。

論理構造を持つ文書の構造を指定して実行する全文検索を，**構造指定検索**といいます。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作成した XML 文書は，構造指定検索の対象になります。

構造指定検索で使用する XML 文書の例を次の図に示します。

図 2-18　構造指定検索で使用する XML 文書の例



構造指定検索では，この図の 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」といった検索ができます。この例の場合は，XML 文書 2 が検索結果として取得できます。

(2) XML インデクスデータ作成機能の処理内容

XML インデクスデータ作成機能では，Preprocessing Library for Text Search を利用して XML ファイルの構文解析を実行し，全文検索用のインデクスデータ（ESIS-B 形式のデータ）を生成します。このインデクスデータを XML 文書の全文検索インデクスとして登録することで，XML 文書の構造を指定した全文検索ができるようになります。

また，インデクスデータを作成するときには，Preprocessing Library for Text Search のフィルタリング機能を使用して，XML ファイルの内容から不要なタグを表す情報を削除したり，不要なタグおよびそのタグに囲まれた文字列全体を削除したりできます。なお，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。

(3) XML インデクスデータ作成機能を使用するための設定

XML インデクスデータ作成機能を使用する場合は，フィルタリング定義ファイル（TFD）を作成して，削除するタグについて定義しておく必要があります。フィルタリング定義ファイルの作成については，「3.18　XML 文書管理機能を使用する場合の設定」を参照してください。

また，構造指定検索の対象になる XML 文書は，次の条件に従って作成する必要があります。

- バージョンなし文書またはバージョン付き文書の構成要素として，全文検索を実行するためのプロパティを追加した dmaClass_DocVersion クラスまたはそのサブクラス（全文検索機能付き文書クラス）が指定されていること。
- 全文検索機能付き文書クラスに追加するプロパティのうち，全文検索インデクス用プロパティとして，次のプロパティのうち，どれか一つが追加されていること。
  - edmProp_StIndex プロパティ（構造指定検索用全文検索インデクス用）
  - edmProp_ConceptStIndex プロパティ（構造指定検索用概念検索インデクス用）
  - edmProp_Content プロパティ（Version 1 互換用全文検索インデクス用）

全文検索機能付き文書クラスについては，「2.3.4　全文検索機能付き文書クラスの追加」を参照してください。

(4) XML インデクスデータ作成機能の処理対象となる文字コード

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
- EUC
- UTF-8
- UTF-16

ただし，全文検索インデクスは，AIX の場合，すべて Shift-JIS で登録されます。このため，XML 文書に対して構造指定検索を実行するための検索条件は，AIX の場合，すべて Shift-JIS で入力してください。

2.5.4 DocumentBroker が管理できる XML 文書

ここでは，DocumentBroker が XML 文書のコンテントとして管理できる XML ファイルについて説明します。

(1) XML ファイルの形式

DocumentBroker は，次のような XML ファイルを XML 文書のコンテント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ことが必要です。

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，次の条件を満たす必要があります。

● 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 回だけ出現する（VariableArray 型のプロパティにマッピングするタグは除く）。

● VariableArray 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 2-19 VariableArray 型のプロパティにマッピングできるタグの構造

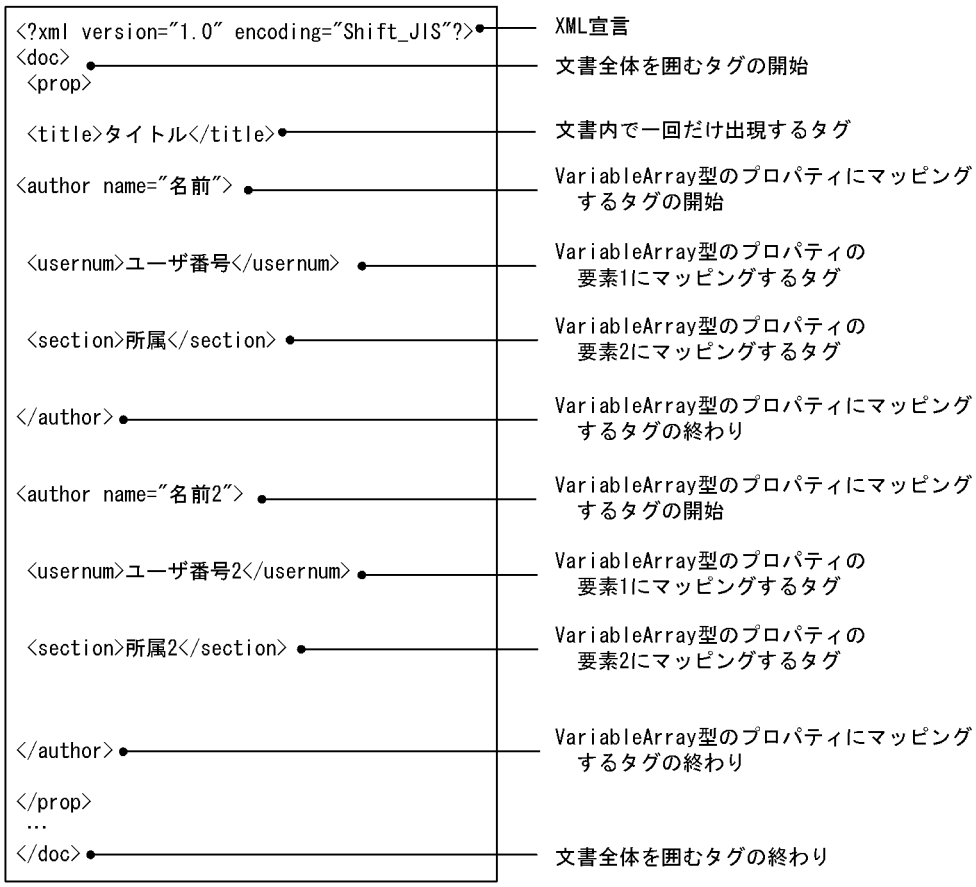
```
〈上位タグ〉
 〈VariableArray型のプロパティのタグ〉
  〈要素1のタグ〉要素1の内容〈/要素1のタグ〉
  〈要素2のタグ〉要素2の内容〈/要素2のタグ〉
   …
  〈/VariableArray型のプロパティのタグ〉
 〈VariableArray型のプロパティのタグ〉
  〈要素1のタグ〉要素1の内容〈/要素1のタグ〉
  〈要素2のタグ〉要素2の内容〈/要素2のタグ〉
   …
 〈/VariableArray型のプロパティのタグ〉
〈上位タグ〉
```

〈VariableArray型のプロパティのタグ〉
　VariableArray型のプロパティに対応する情報を囲んだタグ
〈要素1のタグ〉および〈要素2のタグ〉
　VariableArray型のプロパティの要素に対応する情報を囲んだタグ

(2) XML プロパティマッピング機能が使用できる 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 2-20 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が使用できない 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，次の図のような文書です。

図 2-21 マッピングできない例 1：プロパティに割り当てるタグが複数回出現する XML ファイルの例

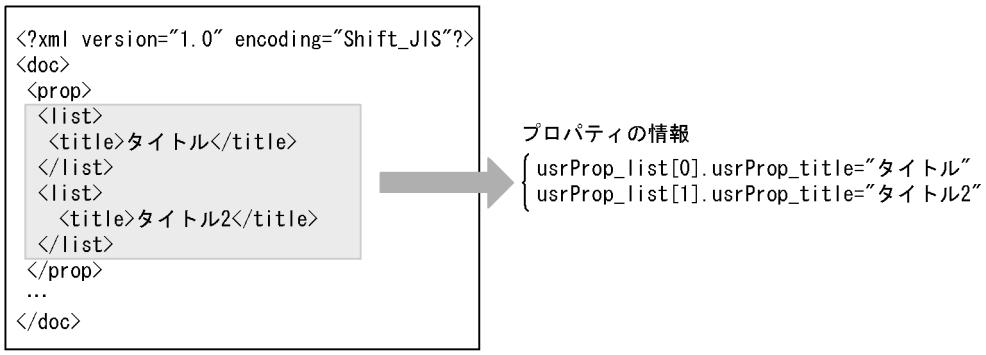
```
<?xml version="1.0" encoding="Shift_JIS"?>
<doc>
 <prop>
 <title>タイトル</title>
 <title>タイトル2</title>
 </prop>
 …
</doc>
```

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は，VariableArray 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VariableArray 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。

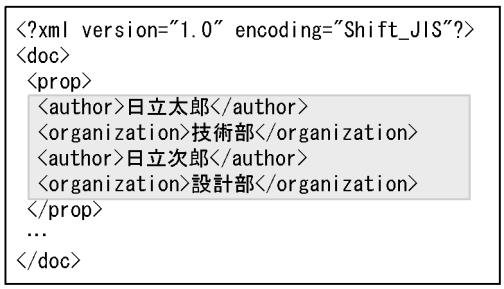
図 2-22　例 1 を登録できる構造に変更した例



使用できない例 2：要素が繰り返し出現して VariableArray 型のプロパティを表すタグがない XML ファイル

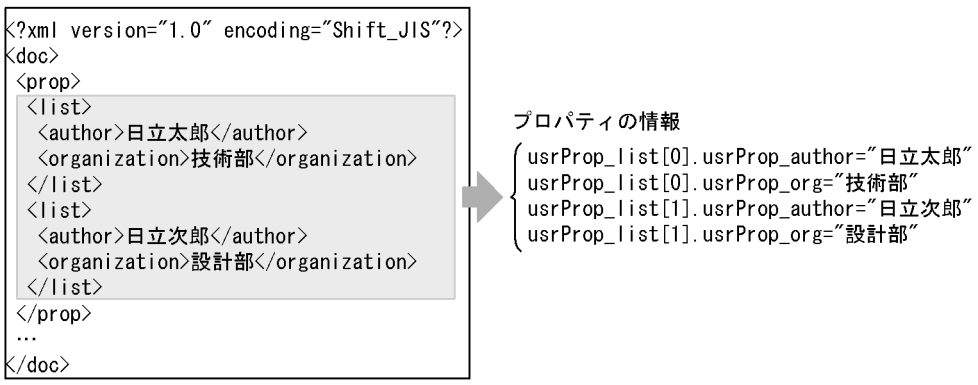
例えば，次の図のような XML ファイルです。

図 2-23　マッピングできない例 2：要素が繰り返し出現して VariableArray 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，VariableArray 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 VariableArray 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，VariableArray 型のプロパティとして登録できます。

図 2-24　例 2 を登録できる構造に変更した例



2.6　アクセス制御機能を使用したシステム構築の検討

この節では，DocumentBroker のアクセス制御機能の概要およびアクセス制御機能を使用した文書管理について説明します。アクセス制御機能を使用したシステムを構築する場合に参照してください。

また，アクセス制御機能の詳細については，使用するクライアントアプリケーションを構築するインターフェースに応じて，次のマニュアルのどちらかを参照してください。

- 「DocumentBroker Version 3　クラスライブラリ　C++　解説」
- 「DocumentBroker Version 3　クラスライブラリ　Java　解説」

2.6.1　アクセス制御情報の概要

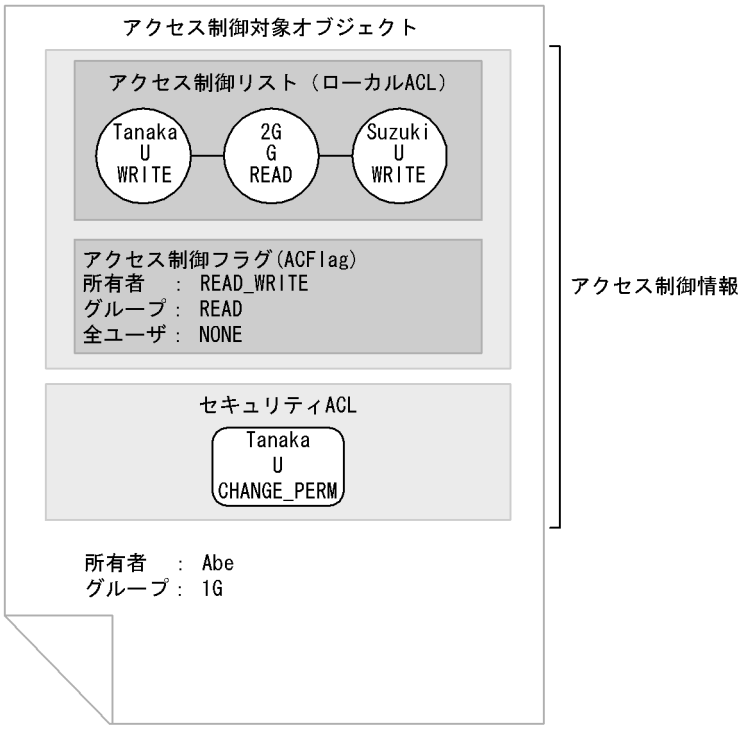
DocumentBroker のアクセス制御は，ユーザが文書空間にログインしたときに生成されるユーザ情報と，アクセス制御の対象である DocumentBroker オブジェクトに付与されたアクセス制御情報を使用してアクセス制御を実行します。

アクセス制御対象オブジェクトに設定されたアクセス制御情報には，次に示す 3 種類の情報があります。

- アクセス制御フラグ（ACFlag）
- アクセス制御リスト（ACL）
- セキュリティ ACL

これらの情報は，組み合わせて使用できます。DocumentBroker のアクセス制御情報の設定例を次の図に示します。

図 2-25　DocumentBroker のアクセス制御情報の設定例



(1) アクセス制御フラグ（ACFlag）

アクセス制御フラグ（以降，ACFlag と呼びます）は，アクセス制御対象オブジェクトのプロパティとして設定します。ACFlag を使用することで次のユーザに対してアクセス制御対象オブジェクトのアクセス権を設定できます。

- ユーザ（アクセス制御対象のオブジェクトの所有者）
- グループ
- すべてのユーザ

ACFlag を使用すると，グループに対してアクセス権を設定できます。図 2-25 では，「文書に対して，文書の所有者は参照と更新が可能であり，グループは参照が可能である」というアクセス権を設定しています。このアクセス権の場合，全ユーザに対してはアクセス権を与えていないので，グループに所属していないユーザは文書を参照できません。

(2) アクセス制御リスト（ACL）

アクセス制御リスト（以降，ACL と呼びます）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。アクセス制御対象オブジェクトに付与されるアクセス制御情報の一部として使用されます。ACL を使用すると，ユーザ単位，およびグループ単位に異なるアクセス権を設定できます。図 2-25 では，「文書に対して，ユーザ『Tanaka』は文書の参照と更新が可能であり，グループ『2G』は参照が可能で，ユーザ『Suzuki』は参照と更新が可能である」というアクセス権を設定しています。

ACL は，設定されるオブジェクトや用途によって，次の 3 種類に分けられます。

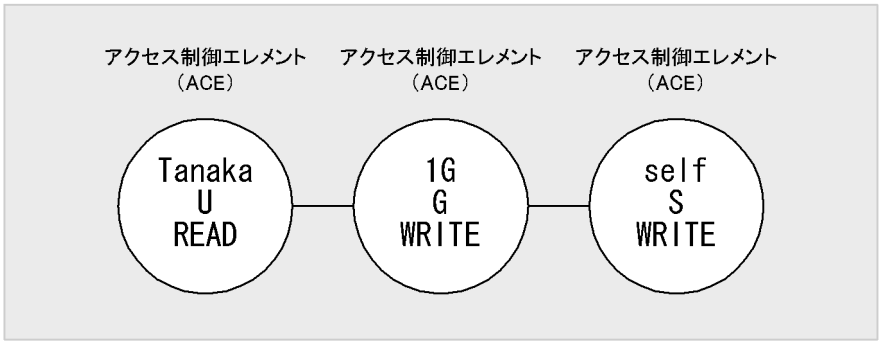
- ローカル ACL
- パブリック ACL
- セキュリティ ACL

ここでは，ACL に設定する ACE と，ローカル ACL およびパブリック ACL について説明します。セキュリティ ACL については，「(3) セキュリティ ACL」で説明します。

(a) アクセス制御エレメント（ACE）

ACL は，アクセス制御エレメント（以降，ACE と呼びます）のリストです。ACL と ACE の関係を次の図に示します。なお，一つの ACL が保持できる ACE の個数は，64 個までです。

図 2-26 ACL と ACE の関係



ACE には，ユーザ単位，およびグループ単位に許可する操作（例えば，参照や更新など）を設定します。ACE に設定する項目を説明します。

● サブジェクト
アクセスを許可するユーザ，またはグループの識別子を設定します。図 2-26 では，「Tanaka」，「1G」および「self」がサブジェクトです。

● サブジェクトタイプ
サブジェクトの種別を設定します。「ユーザサブジェクト（U)」，「グループサブジェクト（G)」，および「システムサブジェクト（S)」があります。

● パーミッション
オブジェクトの作成，オブジェクトのプロパティ参照，オブジェクトのコンテント更新などの実行可能な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。図 2-26 では，「READ」，「WRITE」がパーミッションです。

(b) ローカル ACL

アクセス制御対象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。ローカル ACL は，アクセス制御対象オブジェクトのプロパティとして，アクセス制御対象オブジェクトに一つだけ設定します。

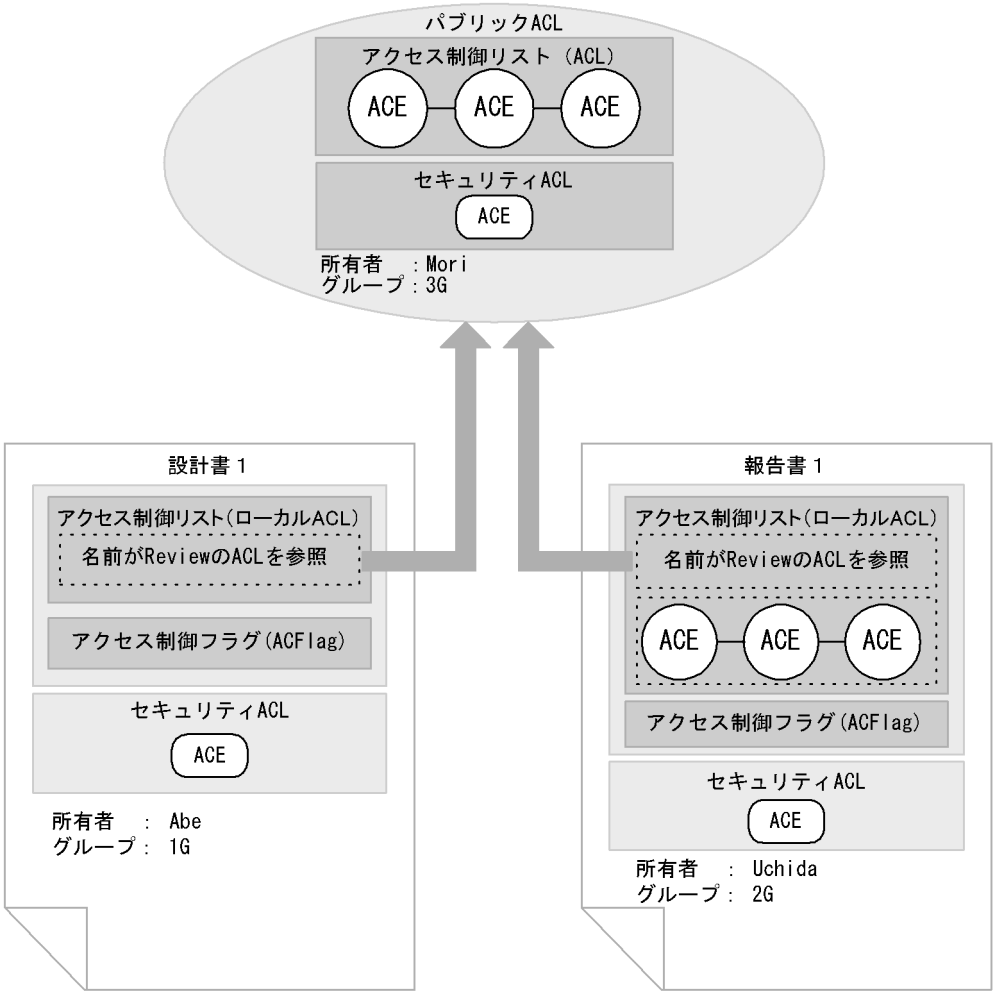
(c) パブリック ACL

複数のオブジェクトに対して同じアクセス制御情報を設定したい場合に使用する ACL です。パブリック ACL を使用すると，複数の文書やコンテナの間で，同じアクセス制御情報を共有できます。

パブリック ACL は，独立したオブジェクトとして文書空間に作成しておきます。このオブジェクトに共有したい ACL を設定し，複数のオブジェクトから参照することで，ACL を共有できます。

パブリック ACL の設定例を次の図に示します。

図 2-27　パブリック ACL の設定例



この図では，「設計書 1」および「報告書 1」が，パブリック ACL を参照して同じアクセス制御情報を共有しています。また，「報告書 1」は，ローカル ACL も参照しています。このように，アクセス制御対象オブジェクトの ACL とパブリック ACL を混在させて使用することもできます。

パブリック ACL は一つのアクセス制御対象オブジェクトに対して 10 個まで設定できます。また，パブリック ACL は，パブリック ACL 自身の所有者を持ち，パブリック ACL のプロパティに値を設定できます。なお，文書やコンテナを表す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 は，使用する場合に応じて文書に設定できます。例えば，文書の審査，承認の各段階で担当者が決まっている場合などでは，各担当者が持つ権利（参照権や更新権など）を記載したパブリック ACL をあらかじめ作成しておくことができます。したがって，パブリック ACL の運用に当たっては，その用途を検討して，あらかじめ設計しておくことをお勧めします。また，パブリック ACL は複数の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。

(3) セキュリティ ACL

DocumentBroker のアクセス制御機能は，ACFlag や ACL に設定したアクセス制御情報を使用してアクセス制御を実行します。オブジェクトに対してアクセス権を持たないユーザでも，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。アクセス制御情報変更権は，各オブジェクトのプロパティに，ACL として設定されています。この ACL を**セキュリティ** ACL といいます。

一つのセキュリティ ACL が保持できる ACE の個数は，64 個までです。また，セキュリティ ACL に設定できるパーミッションは，「CHANGE_PERM（アクセス制御情報変更権）」です。

図 2-25 で，アクセス制御対象オブジェクトの所有者である「Abe」は，セキュリティ ACL で「Tanaka」に対して「CHANGE_PERM」を与えています。これによって，「Tanaka」は，アクセス制御対象オブジェクトのアクセス制御情報（ACL，ACFlag）を変更できます。

2.6.2 文書管理モデルへのアクセス制御の適用

ここでは，具体的な例を示して，DocumentBroker が想定する文書管理モデルへのアクセス制御機能の適用について説明します。

（1）文書のライフサイクルに合わせた文書管理

DocumentBroker は，基幹業務で作成および使用される文書を対象に文書のライフサイクルに合わせた文書管理を実現します。文書のライフサイクルに合わせた文書管理とは，文書が作成され，審査および承認を経て，公開され破棄される過程を管理することです。ここで，DocumentBroker による文書管理を，文書のライフサイクルから，次に示す二つの段階に分けて説明します。

- 作成フェーズ（文書の公開前の段階）
- 公開フェーズ（文書の公開後の段階）

作成フェーズでの，業務担当者による文書の共有を目的とする文書管理システムを「作成フェーズの文書管理システム」と呼びます。また，公開フェーズでの文書登録，文書の参照，および文書の再利用を目的とする文書管理システムを「公開フェーズの文書管理システム」と呼びます。

（a）作成フェーズの目的

作成フェーズの文書管理システムの主な目的は，共通の業務目的を達成するための文書の作成，審査，および承認の遂行と，それらの作業での文書の共有です。作成フェーズの文書の登録者，文書の利用者，および登録後の文書について次に説明します。

- **文書の登録者**
  文書の登録者は，その文書の作成者です。
- **文書の利用者**
  文書の利用者は，その文書の審査担当者や文書を承認する承認者です。また，すでに登録されている文書を参照して新しい文書を作成する作成者も文書の利用者になります。
- **登録後の文書**
  登録された文書は，頻繁に更新されます。

（b）公開フェーズの目的

公開フェーズの文書管理システムの主な目的は，利用者が必要とする文書を迅速かつ正確に検索して，参照または公開することです。公開フェーズの文書の登録，文書の利用者，および公開後の文書について次に説明します。

- **文書の登録**
  文書の登録は，文書の作成者または文書の管理者です。

- **文書の利用者**
  文書の利用者は，組織内部および組織外部に存在します。
- **公開後の文書**
  公開された文書は，長期間，参照や再利用されます。

(2) 各フェーズの文書管理システムのアクセス制御

ここでは，作成フェーズの文書管理システム，および公開フェーズの文書管理システムでのアクセス制御について説明します。

(a) 作成フェーズの文書管理システムのアクセス制御

作成フェーズの文書管理システムのアクセス制御には，ローカル ACL，ACFlag，およびパブリック ACL を使用します。

作成フェーズの文書管理システムは，文書の作成，審査，および承認の段階で，文書の参照者が変化します。この場合，文書のライフサイクルを管理する管理者などが，あらかじめ作成フェーズの各段階に合わせたパブリック ACL を作成しておく必要があります。例えば，「作成用パブリック ACL」，「審査用パブリック ACL」，「承認用パブリック ACL」を作成しておきます。

文書に付与するパブリック ACL を切り替えることで，作成，審査，および承認の各段階に応じたアクセス制御を実行できます。次に，作成，審査，および承認段階でのアクセス制御の実行例を示します。

1. **作成段階のアクセス権の設定**
   文書の作成者が，作成中の文書に「作成用パブリック ACL」を付与します。作成の初期段階には，「作成用パブリック ACL」ではなく，ACFlag やローカル ACL を使用してアクセス権を設定することもできます。
   また，審査・承認を依頼するときに，担当者自身がアクセス権を付与できるようにしておく必要があります。文書の作成者は，審査担当者と承認担当者に対してセキュリティ ACL の「アクセス制御情報変更権」を付与しておきます。
2. **審査段階のアクセス権の設定**
   文書の作成者が，審査を依頼する文書に「審査用パブリック ACL」を付与します。「審査用パブリック ACL」には，審査担当者が参照できるアクセス権が記載されています。審査担当者は，審査が完了した文書の承認を，承認者に依頼します。なお，文書の作成者は，文書を審査担当者に渡す前に，文書に付与されている「作成用パブリック ACL」を解除しておく必要があります。
3. **承認段階のアクセス権の設定**
   文書の審査担当者が承認を依頼する文書に「承認用パブリック ACL」を付与します。承認用パブリック ACL には，承認担当者が参照できるアクセス権が記載されています。なお，文書の審査担当者は，文書を承認担当者に渡す前に，文書に付与されている「審査用パブリック ACL」を解除しておく必要があります。承認者は，承認が完了した文書に対して公開のための作業をして，公開します。

(b) 公開フェーズの文書管理システムのアクセス制御

公開フェーズの文書管理システムのアクセス制御には，パブリック ACL を使用します。

公開フェーズの文書管理システムは，文書管理者が，あらかじめ公開用のパブリック ACL を作成します。また，公開用パブリック ACL を複数作成することで，公開する文書によって異なるアクセス権を設定できます。例えば，「グループ限定の公開用パブリック ACL」，「社内の公開用パブリック ACL」，「社外への公開用パブリック ACL」などを作成しておきます。

公開用の文書を登録する場合，文書管理者または作成者が，文書に公開用のパブリック ACL を付与します。この場合，付与するパブリック ACL を変更することで，文書を公開する範囲を限定できます。また，公開する文書の所有者や，アクセス制御フラグなどを必要に応じて変更します。例えば，文書の所有者を

作成者から文書管理者に変更することで，文書の作成者が，承認後の文書に対して自由にアクセス権を設定できないようにすることができます。ただし，文書の所有者の変更は，所有者だけ実行できます。文書の所有者を変更する場合は，変更前の所有者に変更を依頼する必要があります。

文書のアクセス権は，付与するパブリック ACL を変えるだけで変更できます。同じパブリック ACL を使用しているすべての文書のアクセス権を一括して変更する場合，そのパブリック ACL の内容を変更するだけで，すべての文書のアクセス権を変更できます。

また，組織外の利用者に対しては，文書などのアクセス制御対象オブジェクトを作成できないようにすることもできます。

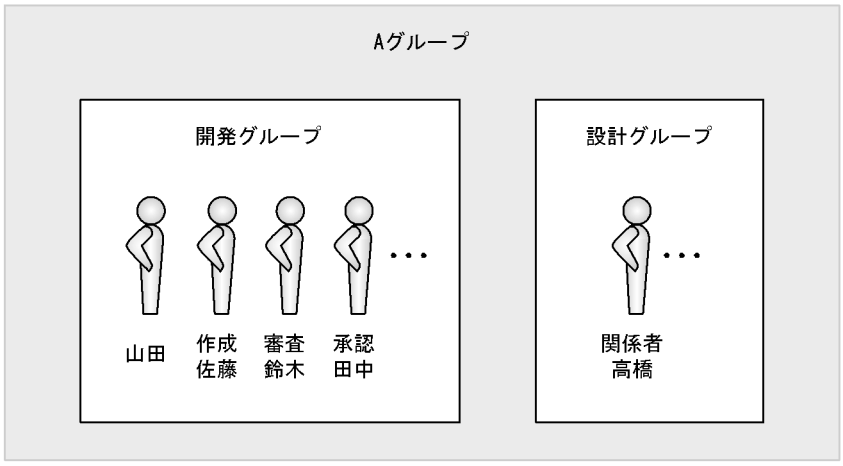
(3) 各フェーズの文書管理へのアクセス制御の適用例

ここでは，作成フェーズの文書管理，および公開フェーズの文書管理へのアクセス制御の適用例を説明します。

(a) アクセス制御を運用するグループ

アクセス制御の適用例を説明する前提として，次の図に示すグループを想定します。このグループは，開発グループ，および設計グループによって構成されています。開発グループの「鈴木」は審査担当者で，「田中」は承認担当者です。また，設計グループの「高橋」は，開発グループの関係者です。

図 2-28 グループの概要



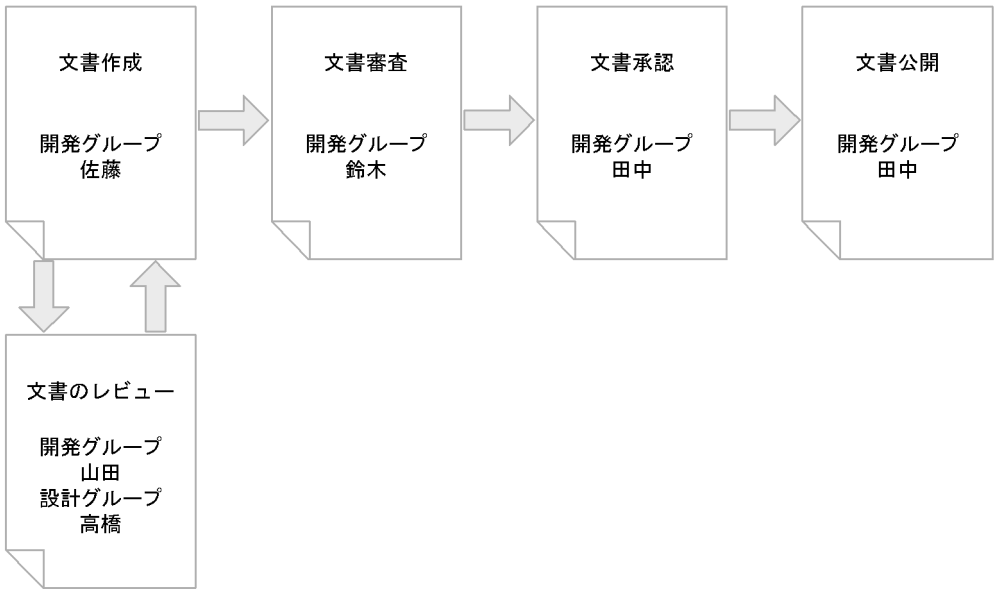
なお，このグループで作成された文書は，審査，および承認を経て公開されます。

ここでは，次のようなアクセス制御の運用について説明します。

- 開発グループの「佐藤」による文書の作成
- 開発グループの「山田」および設計グループの「高橋」による文書のレビュー
- 開発グループの「鈴木」による文書の審査
- 開発グループの「田中」による承認および公開

文書の作成から，公開までの流れを次の図に示します。

図 2-29 文書の作成から公開までの流れ

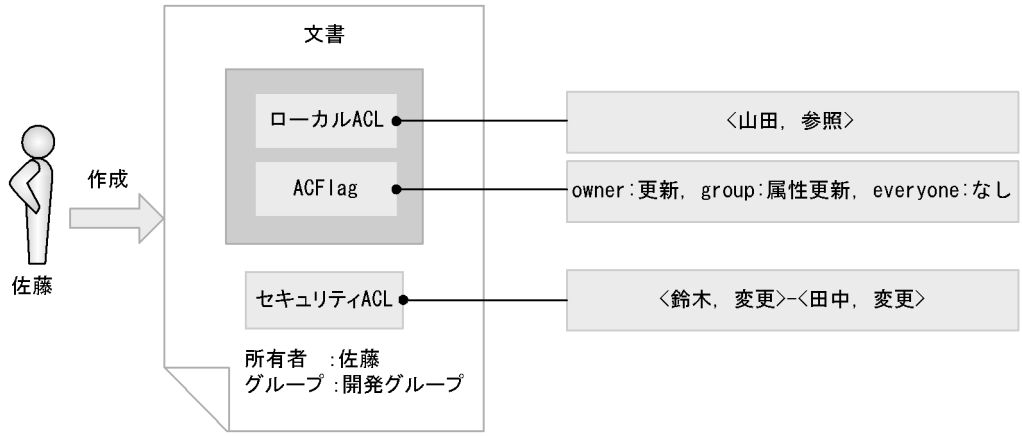


(b) 作成フェーズ

● 文書の作成

開発グループの「佐藤」が，文書を作成します。「佐藤」は，文書を作成した場合，次の図に示すように文書にアクセス制御情報を付与します。

図 2-30 文書の作成

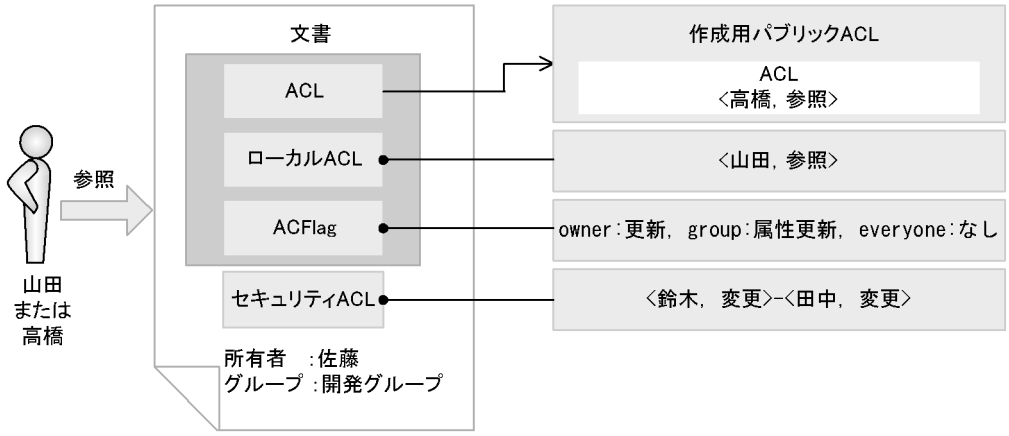


この場合，「佐藤」と同じグループに所属する「山田」に対して，ローカル ACL に参照権を記載します。また，セキュリティ ACL で開発グループの「鈴木」および「田中」にアクセス制御情報変更権を与えます。これは，文書の審査および承認を依頼するときに，「鈴木」および「田中」がアクセス制御情報を変更できるようにするためです。

● 関係者に対する文書のレビューの依頼

文書の作成者である「佐藤」が，同じ開発グループの「山田」および設計グループ内の関係者である「高橋」にレビューを依頼します。この場合，作成用パブリック ACL に，設計グループの「高橋」に対して参照権を記載して，文書に付与します。次の図に，「山田」および「高橋」が参照する場合の文書のアクセス制御情報を示します。

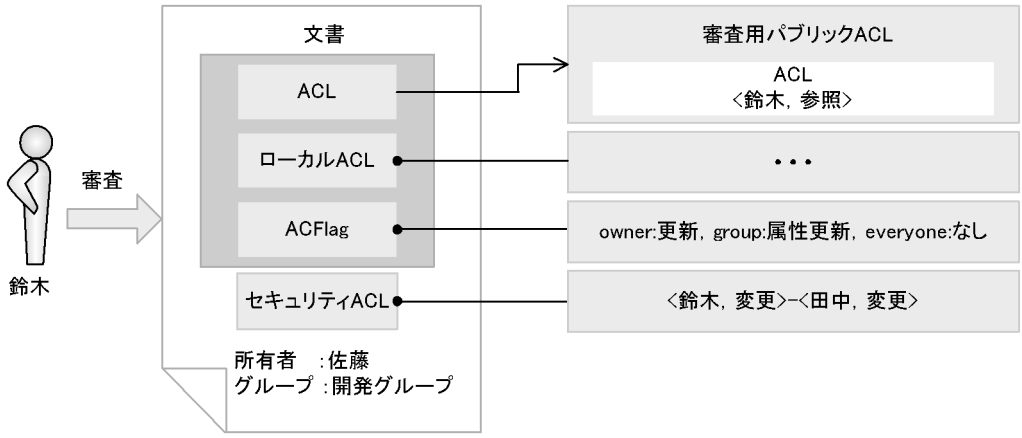
図 2-31 文書の参照



● 文書の審査の依頼

文書作成者である「佐藤」は，作成した文書の審査を依頼します。「佐藤」は，作成した文書に審査用パブリック ACL を付与して，開発グループの審査担当者である「鈴木」に文書を渡します。審査用パブリック ACL には，審査担当者が文書を参照できるアクセス権が記載されています。なお，「佐藤」は，文書を「鈴木」に渡す前に，ローカル ACL に記載されている「山田」に対する参照権を削除し，作成用パブリック ACL を解除しておきます。次の図に，審査担当者の「鈴木」が審査する場合の文書のアクセス制御情報を示します。

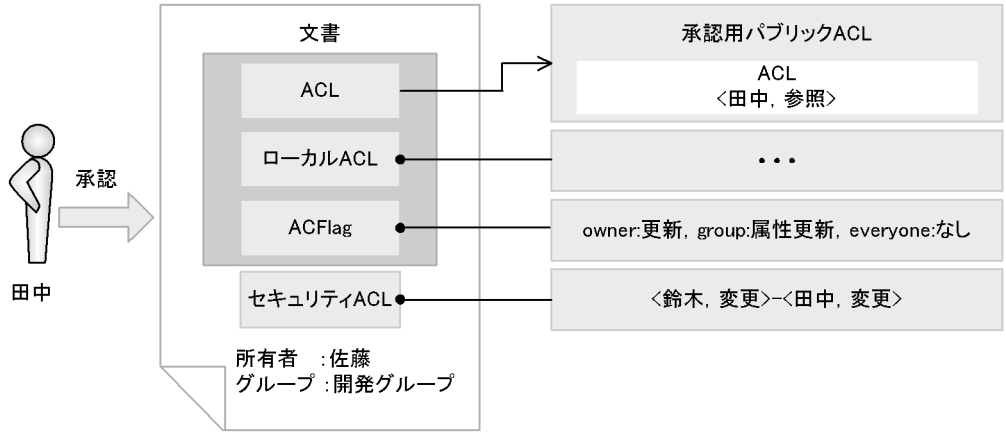
図 2-32 文書の審査



● 文書の承認の依頼

審査担当者である「鈴木」が，文書の承認を依頼します。この場合，「鈴木」は，審査した文書に承認用パブリック ACL を付与して，開発グループの承認担当者である「田中」に文書を渡します。なお，「鈴木」は，文書を渡す前に，審査用パブリック ACL を解除しておきます。承認用のパブリック ACL には，承認者が文書を参照できるアクセス権が記載されています。次の図に，「田中」が承認する場合の文書のアクセス制御情報を示します。

図 2-33 文書の承認

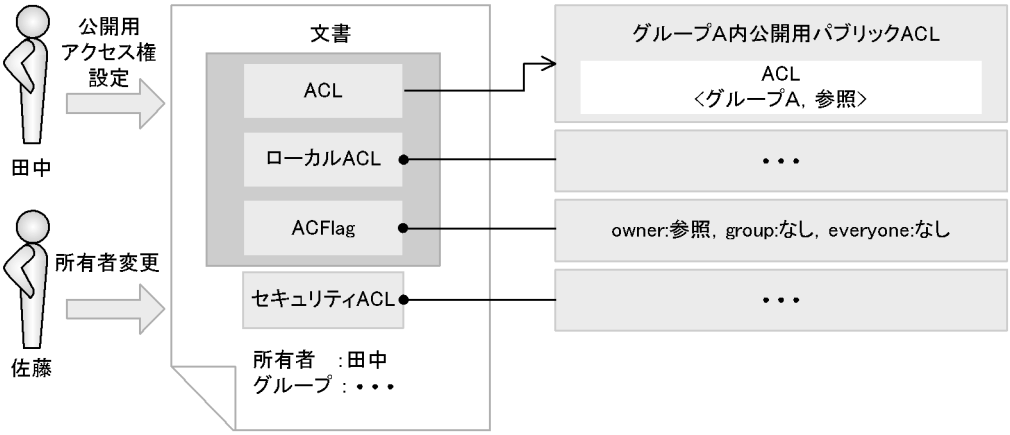


承認担当者である「田中」は，承認が完了した文書に対して，文書の公開のための作業を実行してから，文書を公開します。

(c) 公開フェーズ

文書を公開する場合，承認担当者である「田中」は，文書の作成者である「佐藤」に文書の所有者を「田中」へ変更するように依頼します。「佐藤」は，文書の所有者を「田中」に変更すると，文書に対してアクセス権を設定できなくなります。文書の所有者になった「田中」は，文書を公開するための作業として，セキュリティ ACL と ACFlag を変更して，公開用のパブリック ACL を付与します。次の図に，承認者の「田中」が文書を公開するときの文書のアクセス制御情報を示します。

図 2-34 文書の公開



また，公開用のパブリック ACL を変更することで，文書を公開する対象者によって，異なるアクセス権を設定できます。次の図に公開用のパブリック ACL の例を示します。

図 2-35　公開用のパブリック ACL

2.7 メモリ所要量とディスク占有量の見積もり

この節では，DocumentBroker のメモリ所要量とディスク占有量の見積もりについて説明します。

2.7.1 仮想メモリ所要量の見積もり

仮想メモリ所要量は，次に示す計算式で算出してください。なお，仮想メモリが不足すると，動作が不安定になり異常終了することがありますので必要量を確保してください。

計算式（単位：メガバイト）

AIX **の場合**

- TPBroker V3 **を使用するとき**

  26 ＋ r ＋（7 ＋ r）× p ＋（0.02 ＋ a ＋ b）× u ＋ d

- TPBroker V5 **を使用するとき**

  33 ＋ r ＋（10 ＋ r）× p ＋（0.02 ＋ a ＋ b）× u ＋ d

Linux **の場合**

192 ＋ r ＋（124 ＋ r）× p ＋（0.02 ＋ a ＋ b）× u ＋ d

ただし，d ＝ e × s × c

仮想メモリの計算式に使用する変数のうち，d 以外の変数に設定する値を次の表に示します。

表 2-8 仮想メモリの計算式の変数に設定する値

変数	設定する値
r	メタ情報管理用に確保する共用メモリセグメントサイズ。 この値は，DocumentSpace 構成定義ファイルの [Entry0001] セクションの XdkShmemSize エントリの説明で記述されている計算式を基に算出してください。DocumentSpace 構成定義ファイルについては，「4.2 DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。
p	サービスプロセス数
a	**アクセスログを取得する場合** 2 **アクセスログを取得しない場合** 0（ゼロ）
b	VariableArray 型プロパティが使用するメモリ所要量。 算出方法の詳細については「2.7.2 VariableArray 型のプロパティが使用するメモリ所要量の見積もり」を参照してください。
u	同時接続セッション数
e	**ファイル転送機能を使用しない場合** 2 **ファイル転送機能を使用する場合** 4

変数	設定する値
s	次の 1. と 2. で算出した値の中での最大値（単位：メガバイト） 1. **マルチファイル文書の場合** DocumentSpace 構成定義ファイルの BlobSubstrMode エントリに指定する値によって，算出値が異なります。 ALL の場合：一つの文書に格納するファイルの合計サイズの最大サイズ＋ 2.5 ELEMENT の場合：文書空間で管理する最大ファイルサイズ＋ 2.5 THRESHOLD の場合：BlobSubstrThreshold エントリ値＋ 2.5 2. **上記以外の文書の場合** 文書空間で管理する最大文書サイズ DocumentSpace 構成定義ファイルについては，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。
c	文書空間で管理する同時作業文書数

2.7.2　VariableArray 型のプロパティが使用するメモリ所要量の見積もり

VariableArray 型のプロパティが使用するメモリ所要量の算出について説明します。DocumentBroker は，VariableArray 型のプロパティの要素を格納する方法として，次の方法を提供しています。

- HiRDB の繰り返し列に格納する方法
- 別表に要素を格納する方法

ここでは，VariableArray 型のプロパティ「A」を例にして，それぞれの方法についてメモリ所要量の算出方法を説明します。メモリ所要量を算出する基となる VariableArray 型のプロパティ「A」のデータ構造を次の図に示します。

図 2-36　VariableArray 型のプロパティ「A」のデータ構造

	第1 複合データ	第2 複合データ	・・・	第N 複合データ
第1要素	値$V_{1,1}$	値$V_{1,2}$	・・・	値$V_{1,N}$
第2要素	値$V_{2,1}$	値$V_{2,2}$	・・・	値$V_{2,N}$
⋮	⋮	⋮	⋮	⋮
第L要素	値$V_{L,1}$	値$V_{L,2}$	・・・	値$V_{L,N}$
⋮	⋮	⋮	⋮	⋮
第M要素	－	－	・・・	－

この図では，L は実際に登録される要素数，M はデータベースに定義した最大要素数を表します。それぞれの値は，複合データに設定した値を表します。**複合データ**とは，複数の異なる型によって表されるデータです。VariableArray 型のプロパティの要素の値を格納するために使用します。

値 $V_{X,X}$ のデータサイズ（単位はバイト）を $d_{X,X}$，値 $V_{X,X}$ が格納される表の列に定義されているデータサイズ（単位はバイト）を D_X とすると，1 セッション当たり VariableArray 型のプロパティ「A」を取得する場合に占有するメモリ量は，次に示す算出式で求めることができます。

ただし，次に示す算出式は BatchSizeHint エントリが 1 の場合の式です。BatchSizeHint エントリが 2 以上の場合，算出式の（a）および（b）は BatchSizeHint エントリに指定した値を掛けた値になります。なお，BatchSizeHint エントリは DocumentSpace 構成定義ファイル（docspace.ini）で指定できます。

● HiRDB の繰り返し列に要素を格納する VariableArray 型のプロパティの算出式

VariableArray 型のプロパティ「A」を取得するために占有するメモリ量 =（a）クライアント・サーバ間通信用領域＋（b）DABroker が確保する領域＋（c）HiRDB のクライアントが確保する領域

- （a）：
  クライアント・サーバ間通信用領域 ＝ 8 ＋（ 8 ＋ 8 × N ＋ $\Sigma_{j=1,L}\Sigma_{i=1,N}$ d_{ij} ）× L
- （b）：
  DABroker が確保する領域 ＝（（$\Sigma_{i=1,N}$ (16 ＋ D_i)) ＋ 2 ×（ 1 ＋ N ））× M
- （c）：
  HiRDB のクライアントが確保する領域 ＝（（$\Sigma_{i=1,N}$(16 ＋ D_i)) ＋ 2 ×（ 1 ＋ N ））× M

● 別表に要素を格納する VariableArray 型プロパティの算出式

VariableArray 型のプロパティ「A」を取得するために占有するメモリ量 =（a）クライアント・サーバ間通信用領域 ＋（b）DABroker が確保する領域＋（c）HiRDB のクライアントが確保する領域

- （a）：
  クライアント・サーバ間通信用領域 ＝ 8 ＋（ 8 ＋ 8 × N ＋ $\Sigma_{j=1,L}\Sigma_{i=1,N}$ d_{ij} ）× L
- （b）：
  DABroker が確保する領域 ＝ 136 ＋（$\Sigma_{i=1,N}D_i$) ＋ 2 ×（ 4 ＋ N ）
- （c）：
  HiRDB のクライアントが確保する領域 ＝ 136 ＋（$\Sigma_{i=1,N}D_i$) ＋ 2 ×（ 4 ＋ N ）

2.7.3 ディスク占有量の見積もり

DocumentBroker では，すべての文書および文書に付随する情報を，データベースシステムである HiRDB に格納します。ディスク占有量として，HiRDB でデータベースシステムを構築するために必要なデータベースの容量と，プログラムファイルの容量を合計した量を確保してください。

また，BLOB データの取得先を "$DOCBROKERDIR/tmp" 下に一時ファイルとする場合は，BLOB データ用の一時ファイルの容量も見積もってディスク占有量に加えてください。BLOB データを一時ファイルとして取得するか，メモリを使用して取得するかは，DocumentSpace 構成定義ファイル（docspace.ini）の BlobGettingMethod エントリに設定します。詳細については，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

プログラムファイルの容量

各プログラムファイルのディスク占有量を確保してください。

データベースの容量

「2.8　データベース容量の見積もり」に従って値を算出してください。

BLOB データ用の一時ファイルの容量

＝DocumentBrokerに同時に接続できるクライアントの最大数×1回の接続で取得するBLOBデータの大きさ

DocumentBroker に同時に接続できるクライアントの最大数は，DocumentSpace 構成定義ファイル（docspace.ini）の［Entry0001］セクションの SessionMax エントリに指定する値です。
なお，作成された一時ファイルは，クラスのオブジェクトを破棄（デストラクタを実行）した時点で削除されます。

2.8 データベース容量の見積もり

ここでは，DocumentBroker のデータベース容量の見積もりについて説明します。

2.8.1 データベース容量の見積もり方法

HiRDB でデータベースシステムを構築するために必要なデータベース容量として，次の容量を確保してください。

データベース容量 = RD エリアの容量 + システムファイルの容量

RD エリアは，HiRDB が管理する表およびインデクスを格納する論理的なエリアのことです。DocumentBroker で使用する RD エリアを次の表に示します。

表 2-9 DocumentBroker で使用する RD エリア

分類	RD エリアの種類	説明
システム用 RD エリア	マスタディレクトリ用 RD エリア	データベースシステムの情報を格納します。
	データディレクトリ用 RD エリア	
	データディクショナリ用 RD エリア	
システム LOB 用 RD エリア	データディクショナリ LOB 用 RD エリア	
レジストリ用 RD エリア	レジストリ用 RD エリア	HiRDB Text Search Plug-in を利用した全文検索機能を使用する場合に，HiRDB Text Search Plug-in のレジストリ情報を格納します。
	レジストリ LOB 用 RD エリア	
ユーザ用 RD エリア	ユーザ表用 RD エリア	DocumentBroker で扱う文書やコンテナのデータを格納します。
	ユーザインデクス用 RD エリア	
ユーザ LOB 用 RD エリア	コンテント格納用 RD エリア	文書のコンテントを格納します。
	SGMLTEXT データ格納用 RD エリア	HiRDB Text Search Plug-in を利用した全文検索機能を使用する場合に，コンテントのテキストデータ，テキストデータのインデクスを格納します。
	n-gram インデクス情報格納用 RD エリア	
	全文検索機能付き文字列型プロパティのインデクス格納用 RD エリア	HiRDB Text Search Plug-in を利用した文字列型プロパティに対する全文検索機能を使用する場合に，全文検索機能付き文字列型プロパティのデータのインデクスを格納します。

このマニュアルでは，DocumentBroker で扱う文書やコンテナのデータを格納する，次の RD エリアの容量の算出方法について説明します。

- ユーザ用 RD エリア
- ユーザ LOB 用 RD エリア

ほかの RD エリアの容量およびシステムファイルの容量の算出方法については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。なお，レジストリ用 RD エリアの容量の算出方法については，マニュアル「HiRDB Text Search Plug-in」も参照してください。

2.8.2 ユーザ用 RD エリアの容量の見積もり

ここでは，DocumentBroker で使用するユーザ用 RD エリアの容量の見積もりについて説明します。なお，アクセス制御機能で使用する容量も加算して，RD エリアの容量を見積もる必要があります。アクセ

ス制御機能を使用する場合の容量の算出方法については，「2.8.4　アクセス制御機能を使用する場合のデータベース容量の見積もり」を参照してください。

(1) DocumentBroker で使用するユーザ用 RD エリア

DocumentBroker で管理する文書やコンテナは，DocumentBroker オブジェクトとして，HiRDB に格納されて管理されます。DocumentBroker オブジェクトは，複数の DMA オブジェクトを包含した形で表現されています。したがって，ユーザ用 RD エリアの容量を見積もる場合，DocumentBroker オブジェクトに包含される DMA オブジェクトの構成を考慮して，容量を算出しておく必要があります。

DocumentBroker で使用するユーザ用 RD エリアを次の表に示します。

表 2-10　DocumentBroker で使用するユーザ用 RD エリア

ユーザ用 RD エリアの種類	説明
ユーザ表用 RD エリア	次の情報を格納します。 • DMA クラスに対応する表 • DocumentBroker のメタ情報
ユーザインデクス用 RD エリア	次の情報を格納します。 • DMA クラスに対応する表のインデクス • DocumentBroker のメタ情報のインデクス

ユーザ用 RD エリアには，表とそのインデクス，および DocumentBroker のメタ情報とそのインデクスを格納します。これらの容量は，文書やコンテナなどの数，定義されているプロパティなどから算出します。また，バージョン付き文書の場合は管理するバージョンの数，マルチレンディション文書の場合はレンディションの数など，文書の構成を考慮する必要があります。

(2) ユーザ用 RD エリアが使用するデータベースリソースの所要量

ここでは，ユーザ用 RD エリアの容量を見積もるための目安として，DocumentBroker のクラス（表）ごとに，各レコードのデータサイズ，および各表に格納するレコード数の見積もり方法を説明した一覧を示します。この一覧を参照して，どのように RD エリアの容量を見積もるかについては，「(3)　ユーザ用 RD エリアの容量の見積もり方法」で説明します。

ユーザ用 RD エリア分のデータベースリソースの所要量を次の表に示します。なお，DocumentBroker オブジェクトと DMA オブジェクト，DMA クラスの対応については，「2.2　文書管理に使用するオブジェクト」を参照してください。

表 2-11　ユーザ用 RD エリア分のデータベースリソースの所要量

種別	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
文書	DocVersion※1	11	573	バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2	－

種別	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
	Rendition	6	380	（バージョンごとのRendition 数－ 1）×（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2）	Rendition が一つの場合は，DocVersion の表に含まれます。
	ContentReference	8	4949	バージョンごとのRendition 数×（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2）	リファレンスファイル管理機能を使用する場合にだけ必要なクラス。
	ContentTransfer	6	605	バージョンごとのRendition 数×（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2）	－
	ContentTransfers	12	629 + 543 × n	バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2	マルチファイル管理機能を使用する場合にだけ必要なクラス。n は一つの文書に格納するファイルの最大数であり，EDMInitMeta コマンドの -c オプションで指定します。
	ContentFileLink	6	774	バージョンごとのRendition 数×（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2）	File Link 連携機能を使用する場合にだけ必要なクラス。
	VersionTracedDocVersion※1	11	573	バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2	－
コンテナ	Container※1	5	176	バージョンなしコンテナ数	－
	ContainerVersion※1	6	180	バージョン付き構成管理コンテナごとの平均バージョン数×バージョン付き構成管理コンテナ数＋バージョンなし構成管理コンテナ数	－

種別	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
	VersionTraceableContainer ※1	5	176	バージョンなし構成管理コンテナ数	―
バージョン	ConfigurationHistory ※1	6	228	バージョン付き文書数＋バージョン付き構成管理コンテナ数	―
	VersionSeries	11	656	バージョン付き文書ごとのVersionSeries数×（バージョン付き文書数＋バージョン付き構成管理コンテナ数）	バージョン付き文書ごとのVersionSeries数は一つです。
	VersionDescription	6	180	バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョン付き構成管理コンテナごとの平均バージョン数×バージョン付き構成管理コンテナ数	―
コンテインメント	DirectContainmentRelationship	4	172	DirectContainmentRelationshipオブジェクトを利用した関連するオブジェクトの個数	―
	ReferentialContainmentRelationship	4	172	ReferentialContainmentRelationshipオブジェクトを利用した関連するオブジェクトの個数	―
	VersionTraceableContainmentRelationship	7	280	VersionTraceableContainmentRelationshipオブジェクトを利用した関連するオブジェクトの個数	―
文書間リレーション	Relationship	5	176	文書間リレーションに使用しているRelationshipオブジェクトの個数	―
	VTRelationship	13	496	文書間リレーションに使用しているVTRelationshipオブジェクトの個数	構成管理の対象となる文書間リレーションを使用する場合のクラス。
独立データ	IndependentPersistence ※1	3	72	独立データオブジェクト数	―
全文検索	全文検索機能付き文書クラス	14	606	バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※2	全文検索用，および構造指定検索用のデータを格納する列を追加した文書クラス。
OIID	OIID	3	20	実行環境の数	最新のOIIDです。

種別	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
VariableArray	Struct	3	84	VariableArray 型プロパティの配列に格納する平均要素数×VariableArray 型プロパティを定義するオブジェクト数	基本単位が VariableArray 型のプロパティの配列の要素を定義する場合のクラス。
メタ情報	EDMS_METAMETA	3	33	1	―
	EDMS_METAINI	3	158	2,200 ＋ a × 7 ＋ b × 5 ＋ c	a：定義するユーザクラスの数 b：定義するユーザプロパティの数 c：各クラスに定義する VariableArray 型のユーザプロパティの合計数
	EDMSMETAREGENVID	5	554	実行環境の数	リファレンスファイル管理機能を使用する場合は，列数を 6，データサイズを 558 バイトとして見積もってください。
	EDMS_META_edms	5	527	700 ＋ a × 23 ＋ b × 7 ＋ c ＋ d × 6	a：定義するユーザクラスの数 b：各クラスに定義する VariableArray 型のユーザプロパティの合計数 c：各クラスに定義するクラスに追加するユーザプロパティの合計数 d：全文検索機能付き文書クラスの数
	EDMS_META_dmaclass	5	527	1,800	―
	EDMS_META_dmaprop	5	527	3,200	―
	EDMS_META_dmaproto	5	527	500	―

種別	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
	EDMS_META_dsclass	5	527	1,500 ＋ a × 48 ＋ b ＋ c × 6	a：「dmaClass_」で始まるクラスのサブクラスとして定義するユーザクラスの数 b：a の各ユーザクラス，dmaClass_DirectContainmentRelationship クラス，および dmaClass_ReferentialContainmentRelationship クラスに定義するユーザプロパティの合計数 c：a のクラスのうち全文検索機能付き文書クラスの数
	EDMS_META_dsprop	5	527	1,500 ＋ a × 43	a：「dmaClass_」で始まるクラスのサブクラスに定義するユーザプロパティの合計数 b：a のユーザプロパティに定義するオペレータの平均数
	EDMS_META_dsqop	5	527	1,500	—
	EDMS_META_edmclass	5	527	1,100 ＋ a × 50 ＋ b ＋ c × 6	a：「edmClass_」で始まるクラスのサブクラスとして定義するユーザクラスの数 b：a の各ユーザクラス，edmClass_VersionTraceableContainmentRelationship クラス，edmClass_PublicACL クラス，および edmClass_Relationship クラスに定義するユーザプロパティの合計数 c：a のクラスのうち全文検索機能付き文書クラスの数
	EDMS_META_edmprop	5	527	2,000 ＋ a × 43	a：「edmClass_」で始まるクラスおよびサブクラスに定義するユーザプロパティの合計数 b：a のユーザプロパティに定義するオペレータの平均数
	EDMS_META_edmqop	5	527	600	—
	EDMS_META_edmsys	5	527	100	—

種別	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
	EDMS_META_edmsysclass	5	527	400	―
	EDMS_META_edmsysprop	5	527	200	―
	EDMS_META_ssysobj	5	527	50	―
	EDMS_META_edmnmclass	5	527	100 ＋ a	a：定義するユーザクラスの数
	EDMS_META_edmnmprop	5	527	200 ＋ a	a：定義するユーザプロパティの数
	EDMSMETAdocinfo	2	260	a × 2 ＋ b ＋ 2	文書空間構築コマンド（EDMCBuildDocSpace）実行時だけ a：文書空間情報ファイルに指定するセクション数 b：文書空間情報ファイルに指定するエントリの合計数
	EDMSMETAclassdef	2	260	a × 16 ＋ b × 10 ＋ c × 5 ＋ d × 9 ＋ 2	文書空間構築コマンド（EDMCBuildDocSpace）実行時だけ a：定義するユーザクラスの数 b：各クラスに定義するユーザプロパティの合計数 c：全文検索機能付き文書クラスの数 d：定義するユーザインデクスの数

注※ 1

サブクラスごとに算出する必要があります。

注※ 2

バージョンなし文書は，バージョン付き文書の個々のバージョンに対応しない文書を指します。

(3) ユーザ用 RD エリアの容量の見積もり方法

ここでは，ユーザ用 RD エリアの容量を見積もるための方法について説明します。ここで算出した値を基にして，実際に使用するディスク占有量を算出してください。ディスク占有量の算出方法については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

「(2)　ユーザ用 RD エリアが使用するデータベースリソースの所要量」の表 2-11 の内容を参考にして，次の値を算出してください。

(a) 各表に格納するレコードの総数

表 2-11 のデータサイズおよびレコード数の見積もり方法を参考にして，文書数やコンテナの数などから，各表に格納するレコードの総数を算出してください。

(b) 各表に定義する列の総数

表 2-11 には，DocumentBroker のシステムプロパティが使用する列数が示してあります。サブクラスを定義してユーザプロパティを追加した場合は，この表に示すスーパークラスの列数の値に，ユーザプロパティ数を加算して算出してください。

(c) 各列のデータサイズ

表 2-11 には，DocumentBroker のシステムプロパティが使用するデータサイズが示してあります。サブクラスを定義してユーザプロパティを追加した場合は，この表に示すスーパークラスのデータサイズの値に，ユーザプロパティのデータサイズを加算してください。ユーザプロパティのデータサイズについては，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。対応する HiRDB のデータ型を基に，データサイズを算出してください。

(d) インデクス

オブジェクトのプロパティに定義されるインデクスの算出については，データベース定義文出力コマンド（EDMCrtSql）によって出力されるデータベース定義文を参考にしてください。

EDMS_META_ で始まるメタ情報の表には，次に示す複数列インデクスが一つ定義されます。

- UNIQUE 指定あり
- キー長：136

EDMSMETAREGENVID のメタ情報の表には，次に示す単一列インデクスが一つ定義されます。

- UNIQUE 指定あり
- キー長：2

EDMSMETAdocinfo および EDMSMETAclassdef のメタ情報の表には，次に示す単一列インデクスが一つ定義されます。

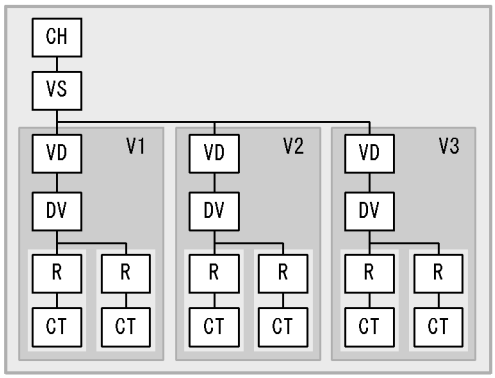
- UNIQUE 指定あり
- キー長：4

(4) 見積もりの例

ここでは，文書を管理する表の見積もりの例として，バージョン付き文書を管理する表の見積もりの例を示します。バージョン付き文書を管理する表の見積もりの例を次の図に示します。

図 2-37 バージョン付き文書を管理する表の見積もりの例

バージョン付き文書



対応するDMAクラス

クラス名（表名）	データサイズ
文書に一つあるクラス	
ConfigurationHistory	228
VersionSeries	656
バージョン数分あるクラス	
VersionDescription	180
DocVersion	573
レンディション数分あるクラス	
Rendition	380
ContentTransfer	605

（凡例）

CH ：ConfigurationHistoryオブジェクト
VS ：VersionSeriesオブジェクト
VD ：VersionDescriptionオブジェクト
DV ：DocVersionオブジェクト
R ：Renditionオブジェクト
CT ：ContentTransferオブジェクト

この例では，バージョン付き文書で，三つのバージョンと二つのレンディションを管理しています。ユーザプロパティの追加はありません。例えば，このバージョン付き文書の文書数が 1,000 の場合，バージョン付き文書を管理する表の容量は次のように算出できます。

文書数×（文書情報＋（バージョン数×（バージョン情報＋（レンディション数－ 1）× Rendition オブジェクト＋（レンディション数× ContentTransfer オブジェクト））））

＝ 1,000 ×（884 ＋（3 ×（753 ＋（2 － 1）× 380 ＋（2 × 605））））

＝約 8.0 メガバイト

文書情報

＝ ConfigurationHistory のデータサイズ＋ VersionSeries のデータサイズ

＝ 228 バイト＋ 656 バイト

＝ 884 バイト

バージョン情報

＝ VersionDescription のデータサイズ＋ DocVersion のデータサイズ

＝ 180 バイト＋ 573 バイト

＝ 753 バイト

2.8.3 ユーザ LOB 用 RD エリアの容量の見積もり

ここでは，DocumentBroker で使用するユーザ LOB 用 RD エリアの容量の見積もりについて説明します。

（1） DocumentBroker で使用するユーザ LOB 用 RD エリア

DocumentBroker で使用するユーザ LOB 用 RD エリアを次の表に示します。

表 2-12 DocumentBroker で使用するユーザ LOB 用 RD エリア

ユーザ LOB 用 RD エリアの種類	説明
コンテント格納用 RD エリア	文書の実体であるコンテントを格納します。 この RD エリアは必ず作成してください。

ユーザ LOB 用 RD エリアの種類	説明
SGMLTEXT データ格納用 RD エリア	コンテントのテキストデータ（SGMLTEXT）を格納します。 HiRDB Text Search Plug-in を利用した全文検索機能を使用する場合には，この RD エリアを作成してください。
n-gram インデクス情報格納用 RD エリア	コンテントのテキストデータのインデクス（n-gram インデクス）を格納します。 HiRDB Text Search Plug-in を利用した全文検索機能を使用する場合には，この RD エリアを作成してください。
全文検索機能付き文字列型プロパティのインデクス格納用 RD エリア	全文検索機能付き文字列型プロパティのデータのインデクスを格納します。 HiRDB Text Search Plug-in を利用した文字列型プロパティに対する全文検索機能を使用する場合には，この RD エリアを作成してください。

ユーザ LOB 用 RD エリアに格納するデータについて説明します。

コンテント

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バイナリ形式のファイルです。コンテントは，dmaClass_ContentTransfer クラス（マルチファイル文書の場合は，edmClass_ContentTransfers クラス）に対応する表に格納されます。

テキストデータ（SGMLTEXT）

コンテントから抽出されたテキスト形式のデータです。テキストデータ（SGMLTEXT）は，dmaClass_DocVersion クラスに対応する表に格納されます。

なお，SGML TEXT は，HiRDB Text Search Plug-in でプレーンテキストを扱う抽象データ型（SGMLTEXT 型）のデータのことです。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。

n-gram インデクス

テキストデータ（SGMLTEXT）の列に対するインデクスデータです。n-gram インデクスは，dmaClass_DocVersion クラスに対応する表に格納されます。

なお，n-gram インデクスは，文書中の n 文字（n-gram）の出現位置を登録するインデクスのことで，HiRDB Text Search Plug-in で全文検索を実行するときに使用されます。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。

全文検索機能付き文字列型プロパティのデータのインデクス

全文検索機能付き文字列型プロパティのデータの列に対するインデクスデータです。全文検索機能付き文字列型プロパティのインデクスデータは，全文検索機能付き文字列型プロパティを追加するサブクラスに対応する表に格納されます。

(2) ユーザ LOB 用 RD エリアが使用するデータベースリソースの所要量

ここでは，ユーザ LOB 用 RD エリアの容量を見積もるための目安として，DocumentBroker のクラス（表）ごとのデータサイズの見積もり方法を説明した一覧を示します。この一覧を参照して，どのように RD エリアの容量を見積もるかについては，「(3) ユーザ LOB 用 RD エリアの容量の見積もり方法」で説明します。

ユーザ LOB 用 RD エリア分のデータベースリソースの所要量を次の表に示します。なお，DocumentBroker オブジェクトと DMA オブジェクト，DMA クラスの対応については，「2.2 文書管理に使用するオブジェクト」を参照してください。

表 2-13 ユーザ LOB 用 RD エリア分のデータベースリソースの所要量

分類	クラス	データサイズ（バイト）	備考
コンテント格納用	ContentTransfer	バージョンごとの Rendition 数×（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※1）×1 文書に格納するファイルのファイルサイズの平均値	文書の実体であるコンテントデータを管理するクラス。 なお，マルチファイル管理機能を使用する場合は，ContentTransfers クラスを使用します。
	ContentTransfers	（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※1）×1 文書に格納するファイルのファイルサイズの平均値	
全文検索データ格納用	全文検索機能付き文書クラス	登録する文書のコンテントのサイズに従って算出※2	－
全文検索機能付き文字列型プロパティのインデクス格納用	全文検索機能付き文字列型プロパティを追加するサブクラス	オブジェクトに設定する全文検索機能付き文字列型プロパティのデータのサイズに従って算出※2	－

注※ 1

バージョンなし文書は，バージョン付き文書の個々のバージョンに対応しない文書を指します。

注※ 2

登録する文書のコンテントのサイズ，またはオブジェクトに設定するプロパティのデータのサイズを求めて，マニュアル「HiRDB Text Search Plug-in」に従って算出する必要があります。

(3) ユーザ LOB 用 RD エリアの容量の見積もり方法

ここでは，ユーザ LOB 用 RD エリアの容量を見積もるための方法について説明します。ここで算出した値を基にして，実際に使用するディスク占有量を算出してください。ディスク占有量の算出方法については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

ユーザ LOB 用 RD エリアに格納するデータの容量は，登録するコンテントのデータサイズや登録する文書の数などを考慮して，的確に見積もってください。

(a) コンテントの容量

「(2)　ユーザ LOB 用 RD エリアが使用するデータベースリソースの所要量」の表 2-13 の内容を参考にして，コンテント格納用 RD エリアに格納するコンテントの容量を算出してください。レンディションの数，ファイルの数，文書の論理構造など，文書の構成を考慮して，登録する文書の総数と文書のコンテントの平均サイズから容量を算出してください。

(b) 全文検索機能で使用する容量

全文検索機能を使用するためには，全文検索機能付き文書クラスごとに，SGMLTEXT データ格納用 RD エリアと n-gram インデクス情報格納用 RD エリアをそれぞれ作成する必要があります。全文検索機能を使用する場合の RD エリアについては，「4.5.1　RD エリア構成定義ファイルの概要」を参照してください。また，全文検索機能を使用するための RD エリアの容量の見積もりについては，マニュアル「HiRDB Text Search Plug-in」を参照してください。

なお，HiRDB Text Search Plug-in に登録する文書サイズは，次に示す値（単位：バイト）を使用して算出してください。

(c) 文字列型プロパティに対する全文検索機能で使用する容量

文字列型プロパティに対する全文検索機能を使用するためには，オブジェクトに設定する全文検索機能付き文字列型プロパティごとに，プロパティのデータのインデクスを格納するための RD エリアを作成します。文字列型プロパティに対する全文検索機能を使用する場合の RD エリアについては，「4.5.1　RD エリア構成定義ファイルの概要」を参照してください。また，文字列型プロパティに対する全文検索機能を使用するための RD エリアの容量の見積もりについては，マニュアル「HiRDB Text Search Plug-in」を参照してください。

なお，HiRDB Text Search Plug-in に登録するプロパティのデータのサイズは，オブジェクトに設定する全文検索機能付き文字列型プロパティのデータのサイズと同じです。

2.8.4　アクセス制御機能を使用する場合のデータベース容量の見積もり

アクセス制御機能を使用する場合のデータベース容量の見積もり方法を説明します。なお，アクセス制御を使用しないため，アクセス制御対応のデータベーススキーマを定義しない場合は，この見積もりは不要です。

ここでは，アクセス制御機能を使用する場合の RD エリアの容量を見積もるための目安として，DocumentBroker のクラス（表）ごとに，各表に定義する列数，各列のデータサイズ，および各表に格納するレコード数の見積もり方法を説明した一覧を示します。アクセス制御機能を使用する場合は，ここで算出した値を，「2.8.2　ユーザ用 RD エリアの容量の見積もり」で算出した値に加算して，データベースの容量を見積もってください。

アクセス制御機能が使用するデータベースリソースの所要量を次の表に示します。

表 2-14　アクセス制御機能が使用するデータベースリソースの所要量

分類	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
アクセス制御対象オブジェクト	DocumentBroker オブジェクトのトップオブジェクトに該当する表	7	24 ＋ u ＋ g ＋ 17 × n	すべての DocumentBroker オブジェクトのトップオブジェクト数	• u：ユーザ識別子の最大長※1 • g：グループ識別子の最大長※1 • n：DocumentBroker オブジェクトにバインドしたパブリック ACL の平均の個数
ACL	edmClass_ACL	6	84 ＋（MAX※2（u，g）＋ 8）×（m ＋ n）	すべての DocumentBroker オブジェクトのトップオブジェクト数	• u：ユーザ識別子の最大長※1 • g：グループ識別子の最大長※1 • m：DocumentBroker オブジェクトのローカル ACL に設定した ACE の平均の個数 • n：DocumentBroker オブジェクトのセキュリティ ACL に設定した ACE の平均の個数

分類	クラス（表名）	列数	データサイズ（バイト）	レコード数見積もり方法	備考
パブリックACL	edmClass_PublicACL	8※3	88＋u＋（MAX※2（u, g）＋8）×（m＋n）※4	パブリックACLの個数	• u：ユーザ識別子の最大長※1 • g：グループ識別子の最大長※1 • m：パブリックACLのローカルACLに設定したACEの平均の個数 • n：パブリックACLのセキュリティACLに設定したACEの平均の個数
バインド	edmClass_BindRelationship	2	68	すべてのDocumentBrokerオブジェクトのトップオブジェクト数×DocumentBrokerオブジェクトごとのバインドするパブリックACLの平均の個数	なし

注※1

ユーザ識別子，グループ識別子の最大長を拡張しない場合，ユーザ識別子，グループ識別子の最大長はそれぞれ254バイトです。

ユーザ識別子の最大長を拡張する場合，ユーザ識別子の最大長は次のどちらかに指定する値です。

- メタ情報の初期設定コマンド（EDMInitMeta）の -n オプション
- 文書空間情報ファイルの UserIDMaxLength エントリ（文書空間の定義コマンド（EDMCDefDocSpace）を使用するとき）

グループ識別子の最大長を拡張する場合，グループ識別子の最大長は次のどちらかに指定する値です。

- メタ情報の初期設定コマンド（EDMInitMeta）の -g オプション
- 文書空間情報ファイルの GroupIDMaxLength エントリ（文書空間の定義コマンド（EDMCDefDocSpace）を使用するとき）

メタ情報の初期設定コマンド（EDMInitMeta）については，「7.3　コマンドの文法」の「EDMInitMeta（メタ情報の初期設定）」を参照してください。文書空間情報ファイルについては，「4.16　文書空間情報ファイル」を参照してください。

注※2

MAXは計算結果の最も大きい値を選ぶことを示しています。例えば，MAX（u，g）でuが254，gが512の場合，MAX（u，g）は512となります。

注※3

パブリックACLにユーザプロパティを追加した場合，追加したプロパティ数を加えて算出してください。

注※4

パブリックACLにユーザプロパティを追加した場合，追加したプロパティのデータサイズを加えて

算出してください。

なお，表 2-14 の「レコード数見積もり方法」に記述されている「すべての DocumentBroker オブジェクトのトップオブジェクト数」とは，作成した文書やコンテナなどに含まれるオブジェクトの個数を表します。このオブジェクトを次に示します。

- ComponentDocVersion オブジェクト
- ConfigurationHistory オブジェクト
- Container オブジェクト
- ContainerVersion オブジェクト
- DocVersion オブジェクト
- IndependentPersistence オブジェクト
- VersionTraceableContainer オブジェクト
- VersionTracedComponentDocVersion オブジェクト
- VersionTracedDocVersion オブジェクト

2.9 リファレンスファイル管理機能を使用する場合のファイルシステムのディスク容量の見積もり

ここでは，リファレンスファイル管理機能を使用する場合の，ファイルシステムのディスク容量の見積もりについて説明します。なお，リファレンスファイル管理機能を使用しない場合，この見積もりは不要です。

ファイルシステムのディスク容量は，コンテント格納先ベースパスごとに見積もってください。コンテント格納先ベースパスごとに格納する文書の最大サイズを見積もり，見積もった値より大きいディスク容量を確保する必要があります。コンテント格納先ベースパスに対して格納する文書の最大サイズの見積もり式を次に示します。

表 2-15 コンテント格納先ベースパスに対するデータサイズの見積もり

クラス	データサイズ（バイト）	備考
ContentReference	バージョンごとの Rendition 数 ×（バージョン付き文書ごとの平均バージョン数 × バージョン付き文書数 + バージョンなし文書数※）× 1 文書に格納するファイルのファイルサイズの平均値	リファレンスファイル管理機能を使用する場合にだけ必要なクラス。

注※

バージョンなし文書は，バージョン付き文書の個々のバージョンに対応しない文書を指します。

2.10 File Link 連携機能を使用する場合のファイルサーバ容量の見積もり

ここでは，File Link 連携機能を使用する場合の，ファイルサーバの容量の見積もりについて説明します。File Link 連携機能を使用しない場合，この見積もりは不要です。なお，Linux の場合および TPBroker V5 を使用している場合，File Link 連携機能は使用できません。

DocumentBroker が使用するファイルサーバのリソースの所要量を次の表に示します。

表 2-16 ファイルサーバリソースの所要量

クラス	データサイズ（バイト）	備考
ContentFileLink	バージョンごとの Rendition 数×（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※ 1）×１文書に格納するファイルのファイルサイズの平均値※ 2	File Link 連携機能を使用する場合に必要なクラス。

注※ 1

バージョンなし文書は，バージョン付き文書の個々のバージョンに対応しない文書を指します。

注※ 2

フォルダを登録の対象とする場合は，フォルダ下に格納されたファイルのファイルサイズの総和の平均値になります。

2.11 マルチレンディション管理機能を使用する場合の排他資源管理テーブルの見積り

ここでは，マルチレンディション管理機能を使用する場合の，排他資源管理テーブルの見積もりについて説明します。なお，マルチレンディション管理機能を使用しない場合，この見積もりは不要です。

HiRDB の SQL 最適化オプションに「DETER_AND_INDEXES」が指定されていない場合，ファイルの更新およびレンディションタイプの変更の際に，排他制御のための HiRDB の排他資源管理テーブル不足エラーが発生することがあります。このため，サーバ当たりの排他制御用プールサイズに，次に示す計算式が成立する値を指定してください。SQL 最適化オプションについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。サーバ当たりの排他制御用プールサイズについては，マニュアル「HiRDB システム定義」を参照してください。

計算式

t ＞ r

ただし，t = p × c

排他資源管理テーブルの計算式に使用する変数に設定する値を次の表に示します。

表 2-17 排他資源管理テーブルの計算式の変数に設定する値

変数	設定する値
t	排他資源管理テーブル数
p	サーバ当たりの排他制御用プールサイズ。この値は HiRDB のサーバ定義の pd_lck_pool_size に指定する値です。（単位：キロバイト）
c	32bit モードの HiRDB ／シングルサーバまたはパラレルサーバの場合：6 64bit モードの HiRDB ／シングルサーバまたはパラレルサーバの場合：4
r	（バージョンごとの Rendition 数− 1）×（バージョン付き文書ごとの平均バージョン数×バージョン付き文書数＋バージョンなし文書数※ 1）×安全係数※ 2

注※ 1

バージョンなし文書は，バージョン付き文書の個々のバージョンに対応しない文書を指します。

注※ 2

安全係数として，1.3 倍程度の余裕を見ておくことをお勧めします。

3 環境設定

この章では，DocumentBroker の環境設定について説明します。

3.1 環境設定の流れ

3.2 環境設定の準備

3.3 インストールとアンインストール

3.4 オペレーティングシステムでの環境設定

3.5 TPBroker での環境設定

3.6 DocumentBroker の実行環境の作成

3.7 ユーザ管理機能の設定

3.8 文書空間の定義

3.9 アクセス制御機能を使用する場合の設定

3.10 データベースシステムでの環境設定

3.11 DocumentBroker でデータベースシステムを使用するための設定

3.12 システム導入支援機能での設定

3.13 ファイル転送機能を使用する場合の設定

3.14 サービスプロセスを使用する場合の設定

3.15 複数の実行環境を構築する場合の設定

3.16 リファレンスファイル管理機能を使用する場合の設定

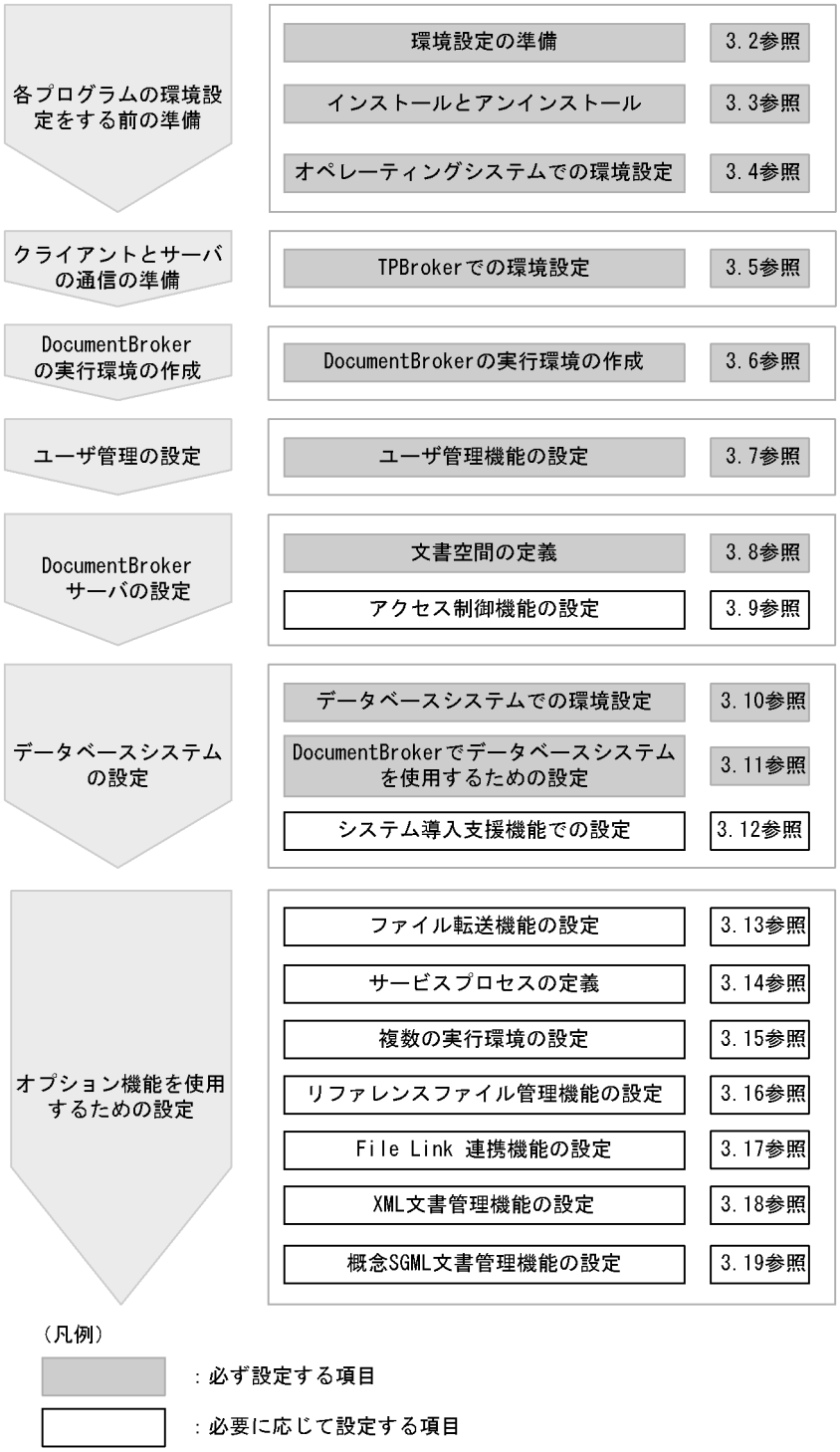
3.17 File Link 連携機能を使用する場合の設定

3.18 XML 文書管理機能を使用する場合の設定

3.1 環境設定の流れ

この節では，DocumentBroker の環境設定の流れについて説明します。DocumentBroker の環境を設定する順序を，次の図に示します。

図 3-1 DocumentBroker の環境設定の流れ



● 環境設定の概要

1. **各プログラムの環境設定をする前の準備**
   各プログラムの環境設定をする前の準備として，前提プログラムのインストールやオペレーティングシステムにシステム管理者の登録などをします。
2. **クライアントとサーバの通信の準備**
   クライアントとサーバ間の通信機能を使用するために必要な，TPBroker の環境設定をします。
3. DocumentBroker **の実行環境の作成**
   DocumentBroker の実行環境を作成します。
4. **ユーザ管理の設定**
   DocumentBroker には，幾つかのユーザ管理方法が用意されています。ここでは，利用環境に合わせたユーザ管理機能を設定します。なお，ここで設定したユーザ管理機能の内容は，次の文書空間の設定で必要になります。
5. DocumentBroker **サーバの設定**
   DocumentBroker の文書空間に関連する内容を設定します。なお，ここで設定した内容は，次のデータベースシステムの環境設定で必要になります。
6. **データベースシステムの設定**
   データベースシステムの環境設定と，DocumentBroker でデータベースシステムを利用するために必要な環境設定をします。
7. **オプション機能を利用するための設定**
   DocumentBroker では，オプション機能として幾つかの機能が用意されています。ここでは，使用するオプション機能に応じて，必要な環境設定をします。

なお，DocumentBroker をクラスタリングシステムで運用する場合の環境設定については，「付録 C　クラスタリングシステムでの運用」を参照してください。

3.2 環境設定の準備

この節では，DocumentBroker の環境を設定する場合の準備について説明します。

(1) システム管理者の決定

DocumentBroker を運用，管理するシステム管理者を決定してください。システム管理者の設定は，スーパーユーザが実行してください。システム管理者は，DocumentBroker のシステムディレクトリおよびシステムファイルの所有者として設定されます。また，運用コマンドはシステム管理者だけが実行できます。

また，DocumentBroker のアクセス制御機能を使用する場合，セキュリティに関する運用管理のために，セキュリティ運用者を指定する必要があります。セキュリティ運用者については，「3.9.1(1) セキュリティ運用者」を参照してください。

(2) 利用ユーザおよびグループの決定

DocumentBroker を利用するユーザとユーザが所属するグループを決定してください。また，ここで決定した利用ユーザおよびグループは，ユーザ管理システムに登録してください。ユーザ管理システムについては，「3.7 ユーザ管理機能の設定」を参照してください。

DocumentBroker のアクセス制御機能を使用する場合，DocumentBroker に登録されたオブジェクトや，そのアクセス権を保守するために，セキュリティ管理者を必ず指定してください。セキュリティ管理者については，「3.9.1(2) セキュリティ管理者」を参照してください。

(3) DocumentBroker の実行環境を作成するディレクトリの決定

DocumentBroker の実行環境を作成するディレクトリを決定してください。

(4) DocumentBroker クライアントと DocumentBroker サーバの接続時の注意

DocumentBroker クライアントと DocumentBroker サーバのバージョンが異なる場合の接続の整合性について説明します。

- DocumentBroker サーバが DocumentBroker クライアントよりも新しい場合，接続を保証します。ただし，DocumentBroker サーバで使用する文書空間の文字コード種別と DocumentBroker クライアントで使用する文字コード種別が異なる場合，DocumentBroker クライアントからは接続できません。
- DocumentBroker クライアントが DocumentBroker サーバのバージョンよりも古い場合，その DocumentBroker クライアントの機能セットの範囲で使用できます。ただし，DocumentBroker クライアント 02-10 以前のファイル転送機能は使用できません。
- DocumentBroker クライアントが DocumentBroker サーバのバージョンよりも新しい場合，接続は保証しません。ただし，Linux 版の DocumentBroker サーバ 03-11 の場合は，Windows 版の DocumentBroker クライアント 03-68 以降と接続が可能です。

3.3 インストールとアンインストール

この節では，次に示すプログラムのインストールおよびアンインストールについて，説明します。

● DocumentBroker Server

● DocumentBroker Development Kit または DocumentBroker Runtime

また，前提プログラムについてもインストールしてください。インストール方法については，各製品のマニュアルを参照してください。

3.3.1 インストール

DocumentBroker のインストールディレクトリは，/opt/HiEDMS です。なお，デバイススペシャルファイル名や CD-ROM のマウントディレクトリは，使用する環境によって異なります。詳細については，オペレーティングシステムのマニュアルを参照してください。また，インストールの手順は，プログラムの提供媒体によって異なります。

(1) インストールする場合の注意

● DocumentBroker Server のインストール環境

DocumentBroker Server は，次の製品がインストールされているマシンにはインストールできません。

- DocumentBroker Repository Version 2
- DocumentBroker Server Version 2

すでに該当する製品がインストールされている場合，次のどちらかの方法でインストールしてください。

- 該当する製品をアンインストールしてからインストールする
- 別のマシンにインストールする

● DocumentBroker Development Kit のインストール環境

DocumentBroker Development Kit は，次の製品がインストールされているマシンにはインストールできません。

- DocumentBroker Development Kit Version 2
- DocumentBroker Runtime Version 2
- DocumentBroker Runtime Version 3
- DocumentBroker Web Component Version 2

すでに該当する製品がインストールされている場合，次のどちらかの方法でインストールしてください。

- 該当する製品をアンインストールしてからインストールする
- 別のマシンにインストールする

● DocumentBroker Runtime のインストール環境

DocumentBroker Runtime は，次の製品がインストールされているマシンにはインストールできません。

- DocumentBroker Development Kit Version 2
- DocumentBroker Development Kit Version 3
- DocumentBroker Runtime Version 2
- DocumentBroker Web Component Version 2

すでに該当する製品がインストールされている場合，次のどちらかの方法でインストールしてください。

- 該当する製品をアンインストールしてからインストールする
- 別のマシンにインストールする

● DocumentBroker Runtime と DocumentBroker Development Kit を入れ換えてインストールする場合すでにインストールされているプログラムのアンインストール後に，インストールしてください。なお，アンインストールの手順については，「3.3.2　アンインストール」を参照してください。

（2）CD-ROM からのインストール

プログラムが CD-ROM で提供されている場合のインストールの手順を次に示します。

＜操　作＞

1. **スーパーユーザ（id:root）としてオペレーティングシステムにログインする。**
2. **「**`mkdir/cdrom`**」を実行して** CD-ROM **をマウントする「**`/cdrom`**」ディレクトリを作成する。**
   「`/cdrom`」の部分には，CD-ROM をファイルシステムとしてマウントするディレクトリ名を指定してください。
3. DocumentBroker **の** CD-ROM **を** CD-ROM **ドライブにセットする。**
4. **「**`mount -r -v cdfs /dev/dsk/c0t2d0 /cdrom`**」を実行して** CD-ROM **をマウントする。**
   「`/dev/dsk/c0t2d0`」および「`/cdrom`」の部分は使用する環境によって異なります。使用するデバイススペシャルファイル名および CD-ROM ファイルシステムを指定してください。
5. CD-ROM **のセットアッププログラム「**`/cdrom/aix/setup /cdrom`**」を実行して日立** PP **インストーラを起動する。**
   初期画面が表示されます。なお，CD-ROM のディレクトリ名や，ファイル名は，ls コマンドで確認して，表示されたファイル名を入力してください。
6. **「**`i`**」または「**`I`**」を入力して「**`I）Install Software`**」を選択する。**
   インストールできるプログラムの一覧が表示されます。
7. **インストールするプログラムにカーソルを移動させて，スペースキーで選択する。**
   選択したプログラムの左側に「`<@>`」が付きます。なお，複数のプログラムを選択できます。
8. **「**`i`**」または「**`I`**」を入力して「**`I）Install`**」を選択する。**
   プログラムをインストールするかどうかについてのメッセージが最下行に表示されます。
9. **最下行に表示されるメッセージに対して「**`y`**」または「**`Y`**」を入力する。**
   インストールが始まります。ただし，「`n`」または「`N`」を入力すると，インストールが中止されて，手順 7. で表示されたインストールできるプログラムの一覧が表示されます。
10. **プログラムのインストールが終了したら，「**`q`**」または「**`Q`**」を入力して「**`Q)Quit`**」を選択して初期画面に戻る。**
    インストールを終了する場合，初期画面で「`q`」または「`Q`」を入力して「`Q)Quit`」を選択して終了します。

（3）DAT からのインストール

プログラムが DAT で提供されている場合のインストールの手順を次に示します。

＜操　作＞

1. **スーパーユーザ（id:root）としてオペレーティングシステムにログインする。**
2. DocumentBroker **のテープを** DAT **ドライブにセットする。**

3. **「tar xf /dev/rmt/0m」を実行して日立 PP インストーラを取り出す。**
   日立 PP インストーラ「/etc/hitachi_setup」を取り出します。なお，使用環境によって，デバイススペシャルファイル名（「/dev/rmt/0m」の部分）が変わります。使用環境に合わせてファイル名を変更してください。

4. **「/etc/hitachi_setup -i /dev/rmt/0mnb」を実行して日立 PP インストーラを起動する。**
   初期画面が表示されます。なお，使用する環境によってデバイススペシャルファイル名（「/dev/rmt/0mnb」の部分）が変わります。使用環境に合わせてファイル名を変更してください。

5. **「i」または「I」を入力して「I）Install Software」を選択する。**
   インストールできるプログラムの一覧が表示されます。

6. **インストールするプログラムにカーソルを移動させて，スペースキーで選択する。**
   選択したプログラムの左側に「<@>」が付きます。なお，複数のプログラムを選択できます。

7. **「i」または「I」を入力して「I）Install」を選択する。**
   プログラムをインストールするかどうかについてのメッセージが最下行に表示されます。

8. **最下行に表示されるメッセージに対して「y」または「Y」を入力する。**
   インストールが始まります。ただし，「n」または「N」を入力すると，インストールが中止されて，手順 6. で表示されたインストールできるプログラムの一覧が表示されます。

9. **プログラムのインストールが終了したら，「q」または「Q」を入力して「Q）Quit」を選択して初期画面に戻る。**
   インストールを終了する場合，初期画面で「q」または「Q」を入力して「Q)Quit」を選択して終了します。

（4） バージョンアップする場合のインストールの方法

DocumentBroker を最新バージョンにバージョンアップするための上書きインストールの方法を次に示します。

＜操　作＞

1. **起動しているプログラムを停止する。**
   停止するプログラムについては，「3.3.2(2)　アンインストールする前に停止しておくプログラム」を参照してください。

2. **最新バージョンのプログラムをインストールする。**
   上書きインストールが実行されます。インストールの手順については，「3.3.1　インストール」を参照してください。

3. **使用しなくなったファイルおよびディレクトリを削除する。**
   削除するファイルおよびディレクトリを次に示します。
   - /opt/HiEDMS/manual/< **バージョン名** >※1/< **製品名** >※2 **以下のファイルおよびディレクトリ**
   - /opt/HiEDMS/manual/< **バージョン名** >※1/< **製品名** >※2
   - /opt/HiEDMS/manual/< **バージョン名** >※1

   注※1　バージョン名
   すでにインストールされていたプログラムのバージョン名が，表示されます。例えば，すでにインストールされているプログラムのバージョンが 03-00 の場合，「0300」と表示されます。

   注※2　製品名
   インストールされていたプログラムが，DocumentBroker Server の場合は，” server ” と表示されます。また，インストールされていたプログラムが，DocumentBroker Development Kit の場合

は，”devkit ”と表示されます。

3.3.2 アンインストール

プログラムを再セットアップする場合，すでにインストールされているプログラムをアンインストールする必要があります。アンインストールする時の注意および手順を，次に説明します。

(1) アンインストールする場合の注意

プロパティマッピング機能を使用している場合，DocumentBroker Development Kit または DocumentBroker Runtime のアンインストールによって，DocumentBroker クライアント環境下の「/opt/HiEDMS/client/etc/xml_files」ディレクトリに格納されている XML 定義ファイルが，ディレクトリごと削除されます。

DocumentBroker Development Kit または DocumentBroker Runtime の再セットアップによって XML 定義ファイルを取得する場合は，DocumentBroker サーバの環境から再度コピーするか，またはバックアップから回復してください。なお，プロパティマッピング機能については，「3.18　XML 文書管理機能を使用する場合の設定」を参照してください。

(2) アンインストールする前に停止しておくプログラム

アンインストールする前に停止または終了しておくプログラムを次に示します。

DocumentBroker Server をアンインストールする場合

DocumentBroker 終了コマンド（EDMStop）の実行によって，DocumentBroker を終了してください。

DocumentBroker Development Kit および DocumentBroker Runtime をアンインストールする場合

DocumentBroker オブジェクト操作ツールおよびライブラリを使用しているプログラムを停止してください。また，ファイル転送サービスを開始している場合は，ファイル転送サービスを停止してください。

(3) アンインストール

アンインストールの手順を次に示します。

＜操　作＞

1. **スーパーユーザ（id:root）としてオペレーティングシステムにログインする。**
2. **起動しているプログラムを停止する。**
   停止するプログラムについては，「3.3.2(2)　アンインストールする前に停止しておくプログラム」を参照してください。
3. **「**`/etc/hitachi_setup`**」を実行して日立 PP インストーラを起動する。**
4. **「**`d`**」または「**`D`**」を入力して「**`D) Delete Software`**」を選択する。**
   アンインストールできるプログラムの一覧が表示されます。
5. **アンインストールするプログラムにカーソルを移動させて，スペースキーで選択する。**
   選択したプログラムの左側に「<@>」が付きます。なお，複数のプログラムを選択できます。
6. **「**`d`**」または「**`D`**」を入力して「**`D) Delete`**」を選択する。**
   プログラムをアンインストールするかどうかについてのメッセージが最下行に表示されます。
7. **最下行に表示されるメッセージに対して「**`y`**」または「**`Y`**」を入力する。**

アンインストールが始まります。ただし，「n」または「N」を入力すると，アンインストールが中止されて，手順 5. で表示されたアンインストールできるプログラムの一覧が表示されます。

8. **プログラムのアンインストールが終了したら，「q」または「Q」を入力して「Q) Quit」を選択して初期画面に戻る。**
   アンインストールを終了する場合，初期画面で「q」または「Q」を入力して「Q) Quit」を選択して終了します。

(4) 上書きインストールした場合のアンインストールの方法

上書きインストールした場合のアンインストールの方法について説明します。なお，上書きインストールについては，「3.3.1(4) バージョンアップする場合のインストールの方法」を参照してください。

上書きインストールした DocumentBroker をアンインストールする場合，「3.3.2 アンインストール」で示している手順の実行だけでは削除されないファイルおよびディレクトリがあります。これらのファイルおよびディレクトリを削除する場合は，手動によって削除してください。このとき，ユーザはスーパーユーザでログインして，オペレーティングシステムのファイル削除コマンドおよびディレクトリ削除コマンドを使用して不要なファイルおよびディレクトリを削除してください。

手動で削除するファイルおよびディレクトリを次に示します。

- /opt/HiEDMS/manual/< **バージョン名** >※1/< **製品名** >※2 **以下のファイルおよびディレクトリ**
- /opt/HiEDMS/manual/< **バージョン名** >※1/< **製品名** >※2
- /opt/HiEDMS/manual/< **バージョン名** >※1
- /opt/HiEDMS/manual
- /opt/HiEDMS

注※1 バージョン名
すでにインストールされていたプログラムのバージョン名が，表示されます。例えば，すでにインストールされているプログラムのバージョンが 03-00 の場合，「0300」と表示されます。

注※2 製品名
インストールされていたプログラムが，DocumentBroker Server の場合は，” server ” と表示されます。また，インストールされていたプログラムが，DocumentBroker Development Kit の場合は，” devkit ” と表示されます。

3.3.3 インストール済みプログラムの一覧表示

日立 PP インストーラで，すでにマシンにインストールされているプログラムの一覧を表示できます。プログラムの一覧を表示する手順を次に示します。

1. **日立 PP インストーラを起動する。**
   初期画面が表示されます。なお，日立 PP インストーラの起動については，「3.3.1(2) CD-ROM からのインストール」の手順 5.，「3.3.1(3) DAT からのインストール」の手順 4.，または「3.3.2(3) アンインストール」の手順 3. を参照してください。

2. **「l」または「L」を入力して「L) List Installed Software」を選択する。**
   PP 一覧表示画面が表示されます。この画面には，すでにマシンにインストールされているプログラムの一覧が表示されます。

3. **PP 一覧表示画面で「q」または「Q」を入力して「Q)Quit」を選択する。**
   初期画面に戻ります。なお，PP 一覧表示画面で「p」または「P」を入力して「P) Print to」を選択すると，すでにインストールされているプログラムの一覧が「/tmp/hitachi_PPLIST」に出力され

ます。

3.4 オペレーティングシステムでの環境設定

この節では，オペレーティングシステムでの環境設定について説明します。

3.4.1 システム管理者の登録

オペレーティングシステムに，システム管理者を登録します。次の場合は，スーパーユーザ権限のあるユーザとして登録してください。

● AIX の場合で，パスワードファイル（etc/passwd）を使用してユーザ管理をするとき

● Linux の場合で，シャドウパスワードファイル（etc/shadow）を使用してユーザ管理をするとき

なお，シャドウパスワードについては，オペレーティングシステムのマニュアルを参照してください。システム管理者の登録に必要な設定項目について，次の表に示します。

表 3-1 システム管理者の登録に必要な設定項目

設定項目		説明
ユーザ登録	ユーザ名	/etc/passwd ファイルに任意の名称を登録する。
	ユーザ識別子	/etc/passwd ファイルに任意のユーザ識別子を登録する。
	グループ識別子	/etc/group ファイルに登録済みのグループ識別子を /etc/passwd ファイルに登録する。
	ホームディレクトリ	/etc/passwd ファイルに既存のディレクトリをホームディレクトリとして登録する。
	パスワード	passwd コマンドを使用して任意のパスワードを設定する。
グループ登録	グループ名	/etc/group ファイルに任意の名称を登録する。
	グループ識別子	/etc/group ファイルに任意の識別子を登録する。ここで登録したグループ識別子と同じ識別子をユーザ登録の項目「グループ識別子」に登録する。

3.4.2 パラメタのカスタマイズ

ここでは，DocumentBroker を使用する場合に，オペレーティングシステムで設定するパラメタの推奨値について説明します。

(1) AIX の場合

AIX の場合のパラメタの変更手順を次に示します。

1. **スーパーユーザとしてオペレーティングシステムにログインする。**
2. **パラメタに値を設定する。**
   パラメタに設定する値は，表 3-2 に示す推奨値よりも大きい値に設定してください。ただし，data_hard および nofiles には，表 3-3 に示す値を設定してください。値を変更する場合については，オペレーティングシステムのシステム管理インタフェース・ツール（SMIT）やオペレーティングシステムの提供するほかのコマンドを利用できます。SMIT の使用方法およびパラメタの詳細については，オペレーティングシステムのマニュアルを参照してください。

表 3-2 オペレーティングシステムで設定するパラメタと推奨値（AIX の場合）

パラメタ	推奨値
maxuproc	512

パラメタ	推奨値
ライセンス数	64

表 3-3 オペレーティングシステムで設定するパラメタと設定値（AIX の場合）

パラメタ	設定値
data_hard	134217728（0x08000000）以上を設定する。
nofiles	DocumentSpace 構成定義ファイルでの設定値に依存する。

nofiles に設定する値について

nofiles に設定する値は，DocumentSpace 構成定義ファイルに指定する Process エントリおよび SessionMax エントリの値を基に，次の計算式 a を使用して算出してください。

a：nofiles >= (SessionMax **設定値** / Process **設定値**) × 8 ＋ 40

ただし，次に示す計算式 b で得られる値が計算式 a で得られる値を超える場合，計算式 b で得られる値を設定してください。

b：nofiles >= SessionMax **設定値**＋ 50

また，多数のクライアントから同時接続したり，長大な文書にアクセスするようなユーザ環境では，DocumentBroker Server が無応答になったりメモリ不足になったりする場合があります。このような場合，次に示すパラメタについて見直す必要があります。

- オペレーティングシステムのシステムパラメタ
  stack_hard
- HiRDB の pdsys ファイルのパラメタ
  pd_max_users

(2) Linux の場合

Linux の場合のパラメタの変更手順を次に示します。

1. **スーパーユーザとしてオペレーティングシステムにログインする。**
2. **パラメタに値を設定する。**
   パラメタに設定する値は，表 3-4 に示す推奨値よりも大きな値に設定してください。ただし，nofile には，表 3-5 に示す値を設定してください。

表 3-4 オペレーティングシステムで設定するパラメタと推奨値（Linux の場合）

パラメタ	推奨値
nproc	512

表 3-5 オペレーティングシステムで設定するパラメタと設定値（Linux の場合）

パラメタ	設定値
nofile	DocumentSpace 構成定義ファイルでの設定値に依存する。

nofile に設定する値について

nofile に設定する値は，DocumentSpace 構成定義ファイルに指定する Process エントリおよび SessionMax エントリの値を基に，次の計算式 a を使用して算出してください。

a：nofile >= (SessionMax **設定値** / Process **設定値**) × 8 ＋ 40

ただし，次に示す計算式 b で得られる値が計算式 a で得られる値を超える場合，計算式 b で得られる値を設定してください。
b：nofile >= SessionMax **設定値**＋ 50

また，多数のクライアントから同時接続したり，長大な文書にアクセスするようなユーザ環境では，DocumentBroker Server が無応答になったりメモリ不足になったりする場合があります。このような場合，次に示すパラメタについて見直す必要があります。

- オペレーティングシステムのシステムパラメタ
  stack_size
- HiRDB の pdsys ファイルのパラメタ
  pd_max_users

3.5 TPBroker での環境設定

この節では，DocumentBroker の前提プログラムである TPBroker で必要な環境設定について説明します。ただし，すでに TPBroker のシステムを構築している場合は，ここで説明する設定は不要です。

(1) TPBroker システムの構築

DocumentBroker は CORBA 仕様に基づいた ORB である TPBroker を利用した CORBA 通信を利用します。したがって，DocumentBroker の環境を設定する前に，TPBroker のシステムを構築しておいてください。TPBroker システムの構築については，マニュアル「TPBroker　ユーザーズガイド」を参照してください。

(2) スマートエージェントの設定

スマートエージェントは，ネットワーク上で使用できるオブジェクトを管理し，クライアントからサーバオブジェクトへのアクセスを実現します。DocumentBroker を使用するためには，ネットワーク内に少なくとも一つのホスト上でスマートエージェントを開始しておく必要があります。スマートエージェントに関する設定や開始方法については，マニュアル「VisiBroker for C++ プログラマーズガイド」（TPBroker V3 を使用する場合）または「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R) デベロッパーズガイド」（TPBroker V5 を使用する場合）を参照してください。

3.6 DocumentBroker の実行環境の作成

この節では，DocumentBroker の実行環境の作成について説明します。

3.6.1 DocumentBroker 実行環境ディレクトリの作成

DocumentBroker の実行環境を格納するためのディレクトリを作成してください。ここで作成したディレクトリの下に DocumentBroker の各種ディレクトリおよびファイルが格納されます。ディレクトリの名称は任意ですが，次に示す情報を設定してください。

ディレクトリ所有者
システム管理者

ディレクトリへのアクセスパーミッション
0755

3.6.2 環境変数の設定（AIX の場合）

DocumentBroker の動作に必要な環境変数の設定について説明します。環境変数は，各サーバマシンのログインシェル環境に合わせて次のファイルに設定してください。なお，$HOME は，ログインユーザのホームディレクトリを示します。

Bourne シェル環境の場合
$HOME/.profile

C シェル環境の場合
$HOME/.cshrc **または** $HOME/.login

（1） LANG の設定

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。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。AIX の場合，次に示す値を指定します。

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，文書空間で使用する文字コード種別に関係なく，ASCII コードだけを扱う英語環境の場合は「C」を設定します。

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。

（2） TZ の設定

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。指定する値は JST-9 です。

（3） DOCBROKERDIR の設定

環境変数「DOCBROKERDIR」には，DocumentBroker の実行環境を作成するディレクトリを指定してください。

（4） PSALLOC の設定

環境変数「PSALLOC」には，メモリ領域の確保をメモリ使用時でなく，メモリ領域要求時に行う，早期ページスペース割り当てを実行するための設定をします。設定する値は「early」です。

（5） NODISCLAIM の設定

環境変数「NODISCLAIM」には，メモリ領域の解放を抑止するための設定をします。設定する値は「true」です。

（6） EXTSHM の設定

環境変数「EXTSHM」には，プロセス内で使用できる共用メモリ数を拡張するための設定をします。設定する値は「ON」です。

（7） LIBPATH

環境変数「LIBPATH」には，関連プログラムおよび DocumentBroker のライブラリのディレクトリを設定します。次の値を追加してください。

TPBroker V3 を使用する場合

```
:HiRDBインストールディレクトリ/client/lib
:/opt/hitachi/common/lib
:/usr/vacpp/lib※1
:/opt/TPBroker/lib
:/opt/DABroker/lib※2
:/opt/HiEDMS/lib
:/usr/lib※3
```

TPBroker V5 を使用する場合

```
:HiRDBインストールディレクトリ/client/lib
:/opt/hitachi/common/lib
:/usr/vacpp/lib※1
:TPBroker V5インストールディレクトリ/lib
:/opt/DABroker/lib※2
:/opt/HiEDMS/lib_tp5
:/usr/lib※3
```

注※ 1

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。これは，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です。

注※ 2

DABroker が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です。

注※ 3

この値は，AIX 5L V5.1 の SecureWay Directory，AIX 5L V5.2 の IBM Directory Server，または AIX 5L V5.3 の IBM Tivoli Directory Server に付属している LDAP クライアントライブラリを使用してユーザ認証を実行する場合に指定します。LDAP クライアントのファイルセットがインストールされていることを確認してください。なお，デフォルトではインストールされません。詳細については，

AIX 5L V5.1，AIX 5L V5.2，または AIX 5L V5.3 のマニュアルを参照してください。

（8） XDK_HOME の設定

環境変数「XDK_HOME」には，smgrreg.ini を格納しているディレクトリの絶対パスを設定します。指定する値は，/opt/HiEDMS/etc です。smgrreg.ini には，レジストリ情報が記述されています。
smgrreg.ini の内容を次の図に示します。

図 3-2　smgrreg.ini の内容（AIX の場合）

```
[07a17522-a626-11d0-b11f-0020af27a837]
ServiceObjectID=text=07a17522-a626-11d0-b11f-0020af27a837
ServiceObjectTypeID=text=236b6b10-a096-11d0-88fe-00a024e8a766
Profile=text=ssysobj.ini
ModuleTypeID=text=352b5360-752c-11d5-8e8b-0000e237083a
ModuleLocation=text=../../../../../lib/libxdmasys1A.a
CharSetEncodingID=text=17
```

（9） XDK_SHMEM_SIZE の設定

環境変数「XDK_SHMEM_SIZE」には，メタ情報管理用に確保する共用メモリセグメントサイズを設定します。単位はバイトです。3,000,000 ～ 1,073,741,824 の間で指定してください。省略した場合，3,000,000 が仮定されます。なお，この共用メモリセグメントサイズは，DocumentSpace 構成定義ファイルの XdkShmemSize エントリにも指定できます。ただし，環境変数「XDK_SHMEM_SIZE」および XdkShmemSize エントリの両方に値を指定している場合は，XdkShmemSize エントリに指定した値が優先されます。したがって，メタ情報管理用に確保する共用メモリセグメントサイズは，XdkShmemSize エントリに指定することを推奨します。メタ情報管理用に確保する共用メモリセグメントの詳細については，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

（10） 注意事項

クライアントアプリケーションと DocumentBroker サーバを同一マシンで実行する場合，同一の実行環境（同一の環境変数を持つ環境）で実行しないでください。

3.6.3　環境変数の設定（Linux の場合）

DocumentBroker の動作に必要な環境変数の設定について説明します。環境変数は，各サーバマシンのログインシェル環境に合わせて次のファイルに設定してください。なお，$HOME は，ログインユーザのホームディレクトリを示します。

Bourne シェル環境の場合
　$HOME/.profile

C シェル環境の場合
　$HOME/.cshrc **または** $HOME/.login

（1） LANG の設定

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。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.SJIS」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，文書空間で使用する文字コード種別に関係なく，ASCII コードだけを扱う英語環境の場合は「C」を設定します。

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.UTF-8」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.UTF-8」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。

(2) TZ の設定

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。指定する値は JST-9 です。

(3) DOCBROKERDIR の設定

環境変数「DOCBROKERDIR」には，DocumentBroker の実行環境を作成するディレクトリを指定してください。

(4) LD_LIBRARY_PATH の設定

環境変数「LD_LIBRARY_PATH」に，関連プログラムおよび DocumentBroker のライブラリのディレクトリを設定します。次の値を追加してください。

:HiRDB インストールディレクトリ /client/lib

:/opt/hitachi/common/lib

:/TPBroker V5 インストールディレクトリ /lib

:/opt/DABroker/lib ※

:/opt/HiEDMS/lib

注※

DABroker が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です。

(5) XDK_HOME の設定

環境変数「XDK_HOME」には，smgrreg.ini を格納しているディレクトリの絶対パスを設定します。指定する値は，/opt/HiEDMS/etc です。smgrreg.ini には，レジストリ情報が記述されています。
smgrreg.ini の内容を次の図に示します。

図 3-3　smgrreg.ini の内容（Linux の場合）

```
[07a17522-a626-11d0-b11f-0020af27a837]
ServiceObjectID=text=07a17522-a626-11d0-b11f-0020af27a837
ServiceObjectTypeID=text=236b6b10-a096-11d0-88fe-00a024e8a766
Profile=text=ssysobj.ini
ModuleTypeID=text=352b5360-752c-11d5-8e8b-0000e237083a
ModuleLocation=text=../../../../../lib/libxdmasys1A.a
CharSetEncodingID=text=17
```

(6) XDK_SHMEM_SIZE の設定

環境変数「XDK_SHMEM_SIZE」には，メタ情報管理用に確保する共用メモリセグメントサイズを設定します。単位はバイトです。3,000,000 ～ 1,073,741,824 の間で指定してください。省略した場合，3,000,000 が仮定されます。なお，この共用メモリセグメントサイズは，DocumentSpace 構成定義ファイルの XdkShmemSize エントリにも指定できます。ただし，環境変数「XDK_SHMEM_SIZE」および XdkShmemSize エントリの両方に値を指定している場合は，XdkShmemSize エントリに指定した値が優先されます。したがって，メタ情報管理用に確保する共用メモリセグメントサイズは，XdkShmemSize エントリに指定することを推奨します。メタ情報管理用に確保する共用メモリセグメントの詳細については，「4.2　DocumentSpace 構成定義ファイル（docspace.ini)」を参照してください。

(7) 注意事項

クライアントアプリケーションと DocumentBroker サーバを同一マシンで実行する場合，同一の実行環境（同一の環境変数を持つ環境）で実行しないでください。

3.6.4　DocumentBroker の実行環境作成コマンドの実行

DocumentBroker の実行環境作成コマンドを実行してください。

AIX の場合

TPBroker V3 を使用する場合

実行環境作成コマンド「/opt/HiEDMS/bin/EDMSetup」を実行してください。

TPBroker V5 を使用する場合

実行環境作成コマンド「/opt/HiEDMS/bin_tp5/EDMSetup」を実行してください。

Linux の場合

実行環境作成コマンド「/opt/HiEDMS/bin/EDMSetup」を実行してください。

DocumentBroker の実行環境作成コマンドの文法と注意事項などの詳細については，「7.3　コマンドの文法」を参照してください。

3.6.5　文書空間識別子の定義

実行環境ディレクトリ /etc/slocalreg.ini の ServiceObjectID エントリには，文書空間の GUID を設定してください。ただし，クライアントで指定する文書空間の GUID（クライアント側の clocalreg.ini で設定できます）と同じ GUID を設定してください。

3.7 ユーザ管理機能の設定

この節では，DocumentBroker のユーザ管理機能について説明します。

DocumentBroker は，LDAP 対応のディレクトリサービス，および UNIX のパスワードファイルと連携したユーザ管理機能を提供します。なお，DocumentBroker が提供するアクセス制御機能を使用する場合，ユーザ管理システムとして，次の製品と連携しています。

- IBM Directory Server（AIX の場合）
- IBM Tivoli Directory Server（AIX の場合）
- SecureWay Directory（AIX の場合）
- Oracle Directory Server（Linux の場合）
- Active Directory Server（Linux の場合）

DocumentBroker の利用環境に合わせたユーザ管理機能を使用してください。また，ここで設定したユーザ管理機能は，DocumentSpace 構成定義ファイルの UserAuthentication エントリで指定する必要があります。DocumentSpace 構成定義ファイルについては，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

3.7.1 LDAP 対応のディレクトリサービスによるユーザ管理機能を使用する場合の設定

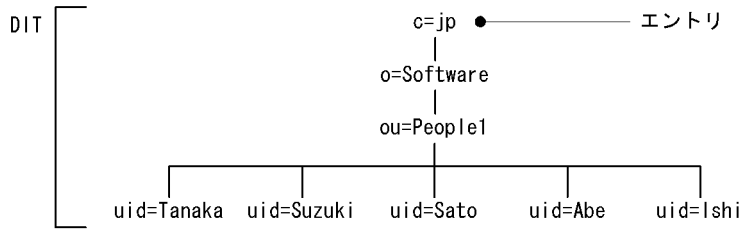
DocumentBroker のユーザ管理機能として LDAP 対応のディレクトリサービスを使用する場合の設定について説明します。

LDAP 対応のディレクトリサービスによるユーザ管理機能を使用する場合，DocumentSpace 構成定義ファイルでの設定が必要です。DocumentSpace 構成定義ファイルについては，「3.8　文書空間の定義」を参照してください。

(1) ディレクトリサービスの情報管理の概念

ここでは，ディレクトリサービスでの情報の管理方法について，図 3-4 を基に説明します。

図 3-4　ディレクトリサービスで管理する情報の構造の例



(a) DIT

ディレクトリサービスが管理する情報は，図 3-4 の形式の構造図によって示されます。この構造を DIT と呼びます。

(b) エントリ

エントリまたはディレクトリエントリは，DIT を構成する情報で，図 3-4 の DIT の各節に該当します（以降，エントリと呼びます）。

図 3-4 の「c=jp」,「o=Software」,「ou=People1」,「uid=Tanaka」などがエントリです。エントリは，左辺と右辺を「=」（等号）でつないでいます。左辺は属性で右辺は属性値を表します。また，エントリは一つ以上の属性によって構成されています。ディレクトリサービスでは，データをエントリおよび属性によって管理します。

(c) DN

DN は，DIT の各エントリを一意に識別するための情報名です。ファイルシステム内のファイルパスのように扱われます。DN は，「,」（コンマ）で区切られたエントリの集まりで，各エントリはディレクトリ内の位置を表す相対識別名（RDN）で構成されます。相対識別名は，「属性 = 属性値」の形式で表現されます。例えば，uid=Tanaka や ou=People1 などが相対識別名に当たります。DN は，ファイルシステム内のファイルパスのように扱われますが，相対識別名の並びがファイルシステムとは逆の順番になります。つまり，ファイルシステムでは，ファイルの名称をいちばん右に指定して，左から右へパスを指定していきますが，DN では，いちばん左の位置に DIT の最下層のエントリを指定し，いちばん右側の位置に DIT の最上位のエントリを指定していきます。DN の記述例を次に示します。

```
uid=Tanaka,ou=People1,o=Software,c=jp
```

この記述例は，図 3-4 に示した「jp」というディレクトリ内の，「Software」というサブディレクトリ内にある「People1」というサブディレクトリの「Tanaka」というエントリの DN を示します。

(2) LDAP 対応のディレクトリサービスへのバインド

DocumentBroker で LDAP 対応のディレクトリサービスへのバインド時に，認証を行うサーババインドを使用するか，認証を行わない匿名バインドを使用するかを選択できます。使用する LDAP 対応のディレクトリサービスへのバインド方法に合わせて設定してください。

DocumentBroker から LDAP 対応のディレクトリサービスへのバインド方法は，DocumentSpace 構成定義ファイル（docspace.ini）の次に示すエントリに設定します。

- LdapBindUserDN エントリ
- LdapBindPassword エントリ

DocumentSpace 構成定義ファイル（docspace.ini）の設定の詳細は，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

(3) LDAP 対応のディレクトリサービスのユーザ認証

ユーザ管理システムとして LDAP 対応のディレクトリサービスを使用する場合は，DN とパスワードによってユーザを認証します。パスワードのチェックは，ユーザクラスの userpassword 属性の属性値と，ログインするときに入力したパスワードが一致しているかどうかによって行われます。

DocumentBroker が LDAP 対応のディレクトリサービスに対して，DN を指定する方法は次に示す 3 種類です。運用方法に合わせて，環境を定義してください。

(a) DN を直接指定する方法

セッションを確立する時に，ユーザ名として DN を直接指定する方法です。

(b) ユーザが指定した DN の一部を基に DocumentBroker が作成する DN を指定する方法

セッションを確立する時に，DN のうち，最後の相対識別名の属性値だけを指定する方法です。ただし，DocumentSpace 構成定義ファイルで，ユーザ識別子として使用する属性と DN を構成するために DocumentBroker が補う相対識別名を定義しておく必要があります。ユーザ名の指定によるセッションの

確立から，ユーザ認証用の DN の作成までの流れを次に示します。

1. ユーザ認証に使用する相対識別名の属性値（ユーザ名）を指定する。
2. DocumentSpace 構成定義ファイルの定義に従って，1. で指定した属性値に対して DocumentBroker が属性を付与して相対識別名を作成する。
3. DocumentSpace 構成定義ファイルの定義に従って，2. の相対識別名に DocumentBroker が相対識別名を補って DN を作成する。

次に，DocumentBroker がユーザ認証用の DN を作成するまでの流れを説明します。

定義内容

- ユーザ識別子として使用する属性：uid 属性
- DocumentBroker が補う相対識別名：ou=People1,o=Software,c=jp

DocumentBroker がユーザ認証用の DN を作成する流れ

1. ユーザ名として「Tanaka」を指定する。
2. DocumentBroker が属性値（ユーザ名：Tanaka）に対して属性を付与して「uid=Tanaka」として，相対識別名を作成する。
3. DocumentBroker は，作成した相対識別名（uid=Tanaka）に「ou=People1,o=Software,c=jp」を補い，「uid=Tanaka,ou=People1,o=Software,c=jp」という DN を作成する。

この方法は，ユーザがあるディレクトリにフラットに存在する場合にだけ有効です。また，ディレクトリ内に存在するユーザが一意に識別できるようになっている必要があります。

DocumentSpace 構成定義ファイルについては，「3.8　文書空間の定義」を参照してください。

(c) DIT から検索した DN を指定する方法

セッションを確立する時に，ユーザ属性の任意の一つの値を指定する方法です。セッションを確立する時に，ユーザが指定したユーザ名を基に，DocumentSpace 構成定義ファイルで指定した DIT を検索してユーザエントリを特定し，その DN を指定します。この方法を利用すると，指定するユーザ名をわかりやすい名前にできます。また，ディレクトリ構成を意識しないでユーザを特定できます。ただし，ディレクトリ内に存在するユーザが一意に識別できるようになっている必要があります。

DocumentSpace 構成定義ファイルについては，「3.8　文書空間の定義」を参照してください。

(4) LDAP 対応のディレクトリサービスでのユーザ情報取得

DocumentBroker のアクセス制御機能を使用して，アクセス制御されている文書に接続する場合，DocumentBroker は LDAP 対応のディレクトリサービスに登録されているユーザに関する情報を使用して，目的の文書に対するアクセスが許可されているかどうか判断します。DocumentBroker がアクセス制御機能で使用するユーザに関する情報は，**ユーザ識別子**および**グループ識別子**です。ユーザ識別子およびグループ識別子について，次に説明します。

(a) ユーザ識別子

ユーザを一意に識別するための情報です。LDAP 対応のディレクトリサービスを使用する場合，ユーザ識別子として使用する属性は，DocumentSpace 構成定義ファイルに指定する必要があります。
DocumentSpace 構成定義ファイルについては，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

(b) グループ識別子

グループを一意に識別する情報です。LDAP 対応のディレクトリサービスでは，「**組織**」と「**グループ**」

をグループとして扱えます。組織とグループについて次に説明します。

組織

オブジェクトクラス「organizationalUnit」またはそのクラスから派生したクラスのオブジェクトです。DocumentBroker では，ユーザの属性として格納される組織情報（例えば，ou 属性や departmentnumber 属性など）およびユーザを識別する DN に記述される組織情報をそれぞれ抽出して利用できます。

グループ

オブジェクトクラス「groupOfUniqueNames」またはそのクラスから派生したクラスのオブジェクトです。

(5) LDAP 対応のディレクトリサービスとの連携時の注意

- 組織構成の変更などによって，ディレクトリ（LDAP）上のグループ識別子が変更されることがあります。このような場合，オブジェクトに付与しているアクセス制御情報で表されるアクセス権を保持するには，ACL に記載されているグループ識別子を変更する必要があります。グループ識別子の変更に伴うオブジェクトのアクセス権の設定の変更については，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」またはマニュアル「DocumentBroker Version 3　クラスライブラリ　Java　解説」を参照してください。

3.7.2 UNIX のパスワードファイルによるユーザ管理機能を使用する場合の設定

UNIX のパスワードファイル（/etc/passwd）を使用してユーザ管理する場合，DocumentBroker を利用するユーザを /etc/passwd ファイルに登録してください。また，ユーザの所属するグループを /etc/group ファイルに登録してください。ただし，NIS を使用した UNIX システムのユーザ管理機能には対応していません。

なお，DocumentBroker のアクセス制御機能を使用してアクセス制御されている文書に接続する場合，DocumentBroker はユーザの属性についてのユーザ情報を取得します。そして，取得したユーザ情報を基に，ユーザの接続したい文書に対してアクセスを許可されているかどうかを判断します。しかし，DocumentBroker は UNIX のパスワードファイルによるユーザ認証機能を使用する場合，アクセスが許可されているかどうか判断するためのユーザ情報を生成できません。この場合は，ユーザが作成したアクセスルーチンを組み込んで，ユーザを一意に識別できるユーザ管理システムを設定してください。

利用ユーザの登録に必要な設定項目について，次の表に示します。

表 3-6　利用ユーザの登録に必要な設定項目

設定項目		説明
ユーザ登録	ユーザ名	/etc/passwd ファイルに任意の名称を登録する。
	ユーザ識別子	/etc/passwd ファイルにユーザ識別子を登録する。
	グループ識別子	/etc/group ファイルに登録済みのグループ識別子を /etc/passwd ファイルに登録する。
	ホームディレクトリ	/etc/passwd ファイルに既存のディレクトリをホームディレクトリとして登録する。
	パスワード	passwd コマンドを使用して任意のパスワードを設定する。
グループ登録	グループ名	/etc/group ファイルに任意の名称を登録する。

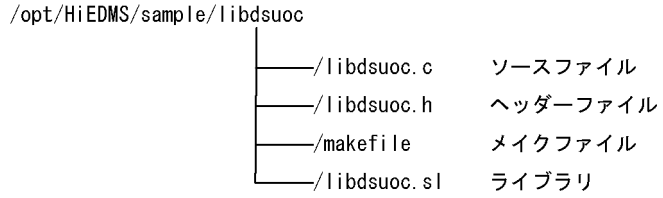
設定項目		説明
	グループ識別子	/etc/group ファイルに任意の識別子を登録する。ここで登録したグループ識別子と同じ識別子をユーザ登録の項目「グループ識別子」に登録する。

3.7.3 ユーザ作成のアクセスルーチンを使用する場合の設定

DocumentBroker は，DocumentSpace 構成定義ファイルの定義によって，使用するユーザ管理システムのアクセスルーチンに，ユーザが作成したアクセスルーチンを組み込む機能を提供しています。これは，ユーザ管理システムに LDAP 対応のディレクトリサービス以外を使用する場合や，すでに使用している LDAP 対応のディレクトリサービスの運用形態が DocumentBroker のサポートしていない形態である場合に利用する機能です。なお，ユーザが作成したアクセスルーチン（コーディング部分）を UOC と呼びます。

(1) アクセスルーチンの作成方法

DocumentBroker に組み込むユーザ管理システムのアクセスルーチンは，決められた形式および仕様に基づいて作成する必要があります。DocumentBroker は，サンプルとして UNIX のユーザ管理ファイル（/etc/passwd および /etc/group）にアクセスするルーチン（libdsuoc.sl）を提供しています。
DocumentBroker で提供する関数を使用して，運用中のユーザ管理システムにアクセスできるように，アクセスルーチンを作成してください。DocumentBroker で提供する関数については，「付録 G　ユーザ作成のアクセスルーチンを使用するための関数」を参照してください。DocumentBroker がライブラリとして提供している UNIX ユーザ管理アクセスのためのサンプルであるユーザ用アクセスルーチンの格納先（AIX の場合）を次に示します。なお，Linux の場合のライブラリの格納先は，「/libdsuoc.sl」が「/libdsuoc.so」となります。



また，作成したアクセスルーチンを組み込む場合は，DocumentSpace 構成定義ファイル（docspace.ini）での設定が必要です。DocumentSpace 構成定義ファイルの UserAuthentication エントリに UOC を指定し，UOCLibrary エントリに必要な値を指定してください。なお，DocumentSpace 構成定義ファイルについては，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

(2) ユーザ作成のアクセスルーチンを使用する場合のユーザ情報の取得

DocumentBroker のアクセス制御機能を使用して，アクセス制御されている文書に接続する場合，DocumentBroker はユーザの属性についてのユーザ情報を取得します。そして，取得したユーザ情報を基に，ユーザの接続したい文書に対してアクセスが許可されているかどうかを判断します。UOC によるユーザ管理機能を使用してユーザ情報を取得する場合に，取得できるユーザ情報を次に示します。

(a) ユーザ識別子

関数 dbrGetUserInformation() から返却されたユーザ識別子文字列を使用します。関数の詳細については，「付録 G　ユーザ作成のアクセスルーチンを使用するための関数」を参照してください。

(b) グループ識別子

関数 dbrGetUserInformation() から返却されたプライマリグループ識別子およびグループ識別子文字列を使用します。関数の詳細については,「付録 G　ユーザ作成のアクセスルーチンを使用するための関数」を参照してください。

3.8 文書空間の定義

この節では，DocumentBroker の文書空間の構成を定義する DocumentSpace 構成定義ファイル（実行環境ディレクトリ /etc/docspace.ini）について説明します。

3.8.1 文書空間を定義する前の準備

DocumentSpace 構成定義ファイルを作成する前に，次に示す点について決定してください。

(1) サービスプロセス数の決定

DocumentSpace 構成定義ファイルには，サービスプロセス監視プロセスが作成するサービスプロセスの数を指定します。

サービスプロセスは，1 プロセス当たり複数クライアントに対してサービスを供給できます。ただし，サービスプロセスがダウンした場合，ダウンしたプロセスに接続していたすべてのユーザがサービスを受けられなくなります。したがって，メモリの使用効率と障害が発生した場合の影響を考慮してサービスプロセス数を決定してください。

(2) DB コネクションプールの管理方法の決定

DocumentBroker は，データベースのメモリのオーバーヘッド削減および接続時のオーバーヘッド削減のために DB コネクションプール機能を提供しています。

(a) DB コネクションプール機能の概要

DocumentBroker を起動した時，データベースに対する複数のコネクションをサービスプロセス単位にプールします。クライアントからの接続要求時やトランザクションの開始要求時に，プールされている DB コネクションを動的に割り当てます。

(b) DB コネクションのプール管理方法について

DB コネクションのプール管理方法について説明します。

1. DocumentBroker を起動した時，サービスプロセス単位に DocumentSpace 構成定義ファイルで指定した DB コネクションプール数だけデータベースとコネクションを確立して，そのコネクションをプールします。
2. クライアントからの接続要求時やトランザクションの開始要求時に，DB コネクションプールからコネクションを取得します。プールされているコネクションがすべて使われている場合は，一時コネクションを動的に作成します。一時コネクション数が上限に達している場合は，コネクション待ち合わせキューにキューイングされます。
   - 一時コネクションの割り当て
     一時コネクションは，プール数とは別に定義した最大値まで，動的にデータベースに接続します。トランザクション終了時に，コネクションの空きを待っているユーザがいる場合に割り当てます。コネクションの空きを待っているユーザがいなければ，データベースとのコネクションを切断します。
   - DB コネクション割り当て待ち時間の監視
     プールされた DB コネクションがすべて使用されている場合に，一時的に DB コネクションの最大要求数を超えたトランザクションの要求が発生したときの DocumentBroker サーバの動作として，「空きコネクションが発生したときに割り当てる」動作を設定しているとき，「割り当て待ち時間」のタイマ監視機能を使用できます。このとき，設定した待ち時間を超えたタイムアウトを検知すると，エラーメッセージを出力して DB コネクション割り当て待ちリストから要求を削除するとともに，クライアントにエラーの戻り値を返却します。

3. クライアントからの解放要求時やトランザクションの終了要求時に，コネクションを DB コネクションプールに戻します。コネクション待ち合わせキューにコネクション待ちのユーザがいれば，このコネクションを割り当てます。

これらの動作は，DocumentSpace 構成定義ファイルの [Entry0001] セクションを構成する次のエントリの指定方法に依存します。

● DBConnectionPoolCount エントリ

● DBConnectionPoolDynamic エントリ

● DBConnectionPoolTiming エントリ

● DBConnectionPoolOver エントリ

● DBConnectionScope エントリ

● DBConnectionPoolWaitTimeOut エントリ

DB コネクションプールの管理方法に従って，最適な設定をしてください。各エントリの指定方法については，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

3.8.2　ユーザ情報の取得方法

ここでは，ログインするときに生成されて値が設定されるユーザ情報の取得方法について説明します。アクセス制御機能を使用する場合，DocumentBroker が連携しているユーザ管理システムによって，ユーザ情報として使用される情報は異なります。また，LDAP 対応のディレクトリサービスを使用したユーザ管理に適応できない場合について，ユーザが作成したアクセスルーチンをユーザ管理システムに組み込む機能を提供します。

（1）ログインするときに生成されるユーザ情報（LDAP 対応のディレクトリサービスを使用する場合）

ユーザ情報は，DocumentBroker オブジェクトに対するアクセス権を判定する場合に使用されます。ユーザ情報には，「ユーザ識別子」および「グループ識別子」があります。これらの情報は，ログインした時に生成されます。

（a）ユーザ識別子

ユーザを識別する情報です。ユーザ識別子は，1 ～ 254 バイト※（ただし，「¥0」を含まない）の ASCII 半角英数字，「-」，「.」，「@」，「_」で構成されます。ユーザ識別子は，一つだけ作成されます。

注※

ユーザ識別子の最大長は，512 バイトまで拡張することもできます。最大長を拡張するには，次のどちらかの方法でユーザ識別子の最大長を定義します。

- メタ情報の初期設定コマンド（EDMInitMeta）の -n オプションに指定する
- 文書空間情報ファイルの UserIDMaxLength エントリに指定する（文書空間の定義コマンド（EDMCDefDocSpace）を使用する場合）

メタ情報の初期設定コマンドの詳細については，「7.3　コマンドの文法」の「EDMInitMeta（メタ情報の初期設定）」を参照してください。文書空間情報ファイルについては，「4.16　文書空間情報ファイル」を参照してください。

ユーザ識別子として取得される情報は，DocumentSpace 構成定義ファイルで任意に指定できます。ユーザ識別子を取得する場合に DocumentSpace 構成定義ファイルで指定するエントリを次に示します。

● LdapUserId エントリ

● LdapUserCase エントリ

これらのエントリの値は，ユーザ識別子として取得する情報および取得方法に従って指定してください。各エントリの指定方法については，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

(b) グループ識別子

ユーザが所属するグループを識別する情報です。ユーザが複数のグループに所属する場合があるため，グループ識別子は，一人のユーザに対して複数存在できます。一つのグループ識別子は，1 ～ 254 バイト※（ただし，「¥0」を含まない）の ASCII 半角英数字，「-」，「.」，「@」，「_」で構成されます。

注※

グループ識別子の最大長は，512 バイトまで拡張することもできます。最大長を拡張するには，次のどちらかの方法でグループ識別子の最大長を定義します。

- メタ情報の初期設定コマンド（EDMInitMeta）の -g オプションに指定する
- 文書空間情報ファイルの GroupIDMaxLength エントリに指定する（文書空間の定義コマンド（EDMCDefDocSpace）を使用する場合）

メタ情報の初期設定コマンドの詳細については，「7.3　コマンドの文法」の「EDMInitMeta（メタ情報の初期設定）」を参照してください。文書空間情報ファイルについては，「4.16　文書空間情報ファイル」を参照してください。

グループとして，プライマリグループも設定できます。プライマリグループは，文書作成時に ACFlag にデフォルトで設定されるグループで，一人のユーザに対して一つのグループが存在することがあります。すなわち，プライマリグループは，必ず存在しなくてもよいグループです。なお，LDAP 対応のディレクトリサービスを使用している場合，プライマリグループは存在しません。

グループ識別子の取得方法は，DocumentSpace 構成定義ファイルで定義します。

グループ情報の取得方法について，次に説明します。なお，DocumentBroker にログインする場合のログイン名を属性として登録してあるユーザエントリのことを，ログインユーザエントリと呼ぶことにします。

グループエントリの属性から取得する方法

日立ディレクトリサービスでは，hdsGroupOfUniqueNames を継承したエントリを使用して，グループ管理する機能を提供しています。このグループの取得方法は，hdsGroupOfUniqueNames のようなグループ管理用のエントリの属性に，ユーザが所属するグループ名称が登録されている場合に使用する方法です。なお，グループ管理については，マニュアル「日立ディレクトリサービス　導入編」および「日立ディレクトリサービス　システム管理編」を参照してください。

ログインユーザエントリの属性から取得する方法

ログインユーザエントリの属性に，そのユーザが所属する組織名称が登録されている場合に使用する方法です。

組織エントリの属性から取得する方法

ログインユーザエントリの属性に，そのユーザが所属する組織エントリの DN が登録されている場合に使用する方法です。

ログインユーザエントリの DN から取得する方法

ログインユーザエントリの DN の構成要素（RDN）の属性値を，グループ識別子として使用する方法です。

これらのグループ識別子を取得する方法は，選択して使用したり，組み合わせて使用したりできます。ただし，「**ログインユーザエントリの属性から取得する方法**」と「**組織エントリの属性から取得する方法**」を組み合わせることはできません。DocumentSpace 構成定義ファイルで指定した方法に従って，グループ識別子の取得方法ごとに，指定する必要があるエントリを次の表に示します。

表 3-7　グループ識別子の取得方法ごとに指定するエントリの一覧

グループ情報の取得方法	指定するエントリ
グループエントリの属性から取得する方法	• LdapGroup=Yes（必須） • LdapGroupRoot=<DN>（必須） • LdapGroupScope エントリ • LdapGroupTimeout エントリ • LdapGroupClass エントリ • LdapGroupId エントリ • LdapGroupFilterLeft エントリ • LdapGroupFilterRight エントリ
ログインユーザエントリの属性から取得する方法	• LdapGroupFromUserAttr=Yes（必須） • LdapGroupIdFromUserAttr エントリ • LdapGroupCase エントリ
組織エントリの属性から取得する方法	• LdapGroupFromUserAttr=Yes（必須） • LdapGroupIsDnFromUserAttr=Yes（必須） • LdapGroupIdFromUserAttr エントリ • LdapGroupIdAttrFromUserAttr エントリ • LdapGroupCase エントリ
ログインユーザエントリの DN から取得する方法	• LdapGroupFromUserDn=Yes（必須） • LdapGroupIdFromUserDn エントリ • LdapGroupCase エントリ

各エントリの指定方法については，「4.2　DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

(2) ログインするときに生成されるユーザ情報（ユーザ作成のアクセスルーチンを使用する場合）

ユーザ管理機能として LDAP 対応のディレクトリサービスに連携していない場合や運用中の LDAP 対応のディレクトリサービスが DocumentBroker の運用形態に合わない場合，DocumentBroker は，ユーザが作成したアクセスルーチンを組み込む機能を提供しています。

ユーザ管理システムへのアクセスルーチンとして，ユーザが作成したアクセスルーチンを使用する場合は，DocumentSpace 構成定義ファイルの UserAuthentication エントリで「UOC」を指定する必要があります。この場合，UOCLibrary エントリの値として，ユーザが作成したアクセスルーチンのライブラリを絶対パスで指定します。UserAuthentication エントリで「UOC」を指定した場合に生成されるユーザ情報について，次に説明します。

(a) ユーザ識別子

関数 dbrGetUserInformation() から返却されたユーザ識別子の文字列を使用します。関数の詳細については，「付録 G　ユーザ作成のアクセスルーチンを使用するための関数」を参照してください。

(b) グループ識別子

関数 dbrGetUserInformation() から返却されたグループ識別子の文字列を使用します。関数の詳細については，「付録 G　ユーザ作成のアクセスルーチンを使用するための関数」を参照してください。

3.9 アクセス制御機能を使用する場合の設定

この節では，アクセス制御機能を使用した文書管理の概略について説明します。

3.9.1 アクセス制御を使用するための管理者の設定

(1) セキュリティ運用者

セキュリティ運用者は，セキュリティに関する運用および管理するユーザです。アクセス制御機能の運用情報を定義する**セキュリティ定義ファイル**を更新することで，セキュリティ管理者を定義したり，ユーザの権限を定義したり変更したりします。セキュリティ運用者は，DocumentBroker が動作するオペレーティングシステム上のユーザから選択します。

DocumentBroker をインストールした直後は，システム管理者がセキュリティ運用者の役割も持ちます。システム管理者が持っているセキュリティ定義ファイルの更新権を，セキュリティ運用者に所有者を変更することで，セキュリティに関する運用を明確に分けることができます。

(2) セキュリティ管理者

セキュリティ管理者は，DocumentBroker に登録されたオブジェクトやオブジェクトのアクセス権を保守するユーザです。セキュリティ管理者は，文書空間のすべてのアクセス制御対象オブジェクトに対してフルコントロールのアクセス権を持ちます。セキュリティ管理者は，連携するユーザ管理システムに登録されているユーザから選択してセキュリティ定義ファイルに定義します。

セキュリティ管理者は必ず指定してください。なお，セキュリティ管理者の定義については，「3.9.2(2) セキュリティ定義ファイルの定義」を参照してください。

3.9.2 アクセス制御機能の設定

ここでは，アクセス制御機能を使用するために必要な設定について説明します。

(1) データベーススキーマの作成

アクセス制御機能対応のデータベーススキーマを構築する必要があります。具体的には，メタ情報の初期設定コマンド（EDMInitMeta）を実行するとき，コマンドオプションにアクセス制御機能を利用するための -A オプションを指定します。なお，メタ情報の初期設定コマンド（EDMInitMeta）は，「3.11.3　データベースシステムの設定手順」で実行します。

メタ情報の初期設定コマンド（EDMInitMeta）で -A オプションが指定されるとアクセス制御情報を含んだメタ情報がデータベースに登録されます。そのあと，メタ情報の追加コマンド（EDMAddMeta），および DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）を実行すると，アクセス制御対応のメタ情報およびデータベース定義文が出力されます。出力されたデータベース定義文およびメタ情報を使用することで，アクセス制御機能に対応したデータベーススキーマを構築できます。メタ情報の初期設定コマンド（EDMInitMeta）の詳細については，「7.3　コマンドの文法」を参照してください。なお，アクセス制御機能対応のデータベースのスキーマを構築すると，スキーマは変更できません。

メタ情報の初期設定コマンド（EDMInitMeta）で -A オプションが指定されなかった場合，アクセス制御情報なしのメタ情報が登録されます。そのあと，メタ情報の追加コマンド（EDMAddMeta），および DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）を実行すると，アクセス制御非対応のメタ情報およびデータベース定義文が出力されます。出力されたデータベース定義文およびメタ情報を使用することで，アクセス制御機能に非対応のデータベーススキーマを構築できます。

(2) セキュリティ定義ファイルの定義

セキュリティ定義ファイルは，次に示すアクセス制御機能の運用に関する情報を定義するファイルです。

- セキュリティ管理者
  DocumentBroker に登録されたオブジェクトやオブジェクトのアクセス権を保守するユーザです。
- ユーザ権限定義ファイル名
  文書空間にオブジェクトを作成する権限や文書空間内のオブジェクトに対する操作の範囲を定義するために作成するファイルの名称です。
- デフォルトで設定されるパーミッション
  新規にオブジェクトを作成した場合に，ACFlag に設定するパーミッションです。

セキュリティ定義ファイルの定義方法については，「4.3　セキュリティ定義ファイル（docaccess.ini）」を参照してください。

(3) ユーザ権限の定義

セキュリティ定義ファイルに指定したユーザ権限定義ファイルに，**ユーザ権限**を定義します。ユーザ権限とは，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。セキュリティ運用者は，このユーザ権限定義ファイルを更新することでユーザ権限を変更できます。

ユーザ権限定義ファイルの定義方法については，「4.4　ユーザ権限定義ファイル」を参照してください。

3.10 データベースシステムでの環境設定

この節では，DocumentBroker の前提プログラムであるデータベースシステムで必要な環境設定について説明します。なお，全文検索および概念検索といった全文検索機能を利用する場合は，HiRDB のプラグインプログラムである HiRDB Text Search Plug-in での環境設定も必要です。

3.10.1 HiRDB の環境設定の準備

DocumentBroker で管理するデータはデータベースに格納します。DocumentBroker で使用できるデータベースは HiRDB だけです。したがって，DocumentBroker の環境を設定する前に，HiRDB のシステムを構築しておく必要があります。HiRDB の環境設定の準備として，HiRDB のインストールまでは終わらせておいてください。このとき，DocumentBroker の文書空間で使用する文字コード種別に合わせて，HiRDB で使用する文字コード種別を設定してください。HiRDB のシステム構築の流れおよびインストール方法については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

また，HiRDB の環境設定を始める前に，DocumentBroker で利用する RD エリアについては，格納する文書数や追加するプロパティの数を考慮して見積もってください。DocumentBroker で利用する RD エリアの見積もりについては，「2.8　データベース容量の見積もり」を参照してください。

3.10.2 HiRDB の環境設定

HiRDB の環境設定について，次に示します。

● RD エリアの作成
HiRDB の環境設定では，システムファイルの作成や DocumentBroker を運用するために必要な RD エリアなどを作成します。RD エリアの作成は，create rdarea 文を使用した RD エリア構成定義ファイルで定義します。HiRDB の環境設定については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。なお，ユーザ用 RD エリアおよびユーザ LOB 用 RD エリアについては，「2.8 データベース容量の見積もり」で算出した値に従って作成してください。
また，全文検索および概念検索といった全文検索機能を使用するためには，全文検索用の表や管理情報を格納する RD エリアを HiRDB に確保しておく必要があります。全文検索用に使用する RD エリアは，次の HiRDB コマンドを使用してあらかじめ確保しておきます。

pdmod -a RD **エリア構成定義ファイル名**

使用するデータベースが HiRDB パラレルサーバの場合，全文検索に使用する RD エリアは，すべて HiRDB の BES に作成します。また，使用するデータベースが HiRDB シングルサーバの場合，全文検索用に使用する RD エリアはシングルサーバに作成します。作成する RD エリアのサイズや初期値などについては，「4.5　RD エリア構成定義ファイル」を参照してください。また，HiRDB のデータベース構成変更ユティリティ（pdmod）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。なお，RD エリアを定義する前提として，HiRDB ファイル領域および HiRDB ファイルを定義しておく必要があります。

● インデクスキー値無排他の適用
HiRDB のインデクスキー値無排他の適用基準については，マニュアル「HiRDB　UAP 開発ガイド」を参照してください。

3.10.3 HiRDB のユーザ権限の設定

DocumentBroker で使用するデータベースを定義する場合，HiRDB に対して接続する必要があります。したがって，データベースを作成するユーザに対して，CONNECT 権限を付与してください。また，スキーマを定義する必要もあるので，スキーマ定義権限も付与してください。なお，ここで CONNECT 権限やスキーマ定義権限を付与するユーザは，DocumentSpace 構成定義ファイルの PdUser エントリで指定したユーザにしてください。DocumentSpace 構成定義ファイルについては，「4.2　DocumentSpace 構成定義ファイル（docspace.ini)」を参照してください。

HiRDB のユーザ権限は，定義系 SQL の GRANT 文を使用して与えます。CONNECT 権限を与える GRANT 文の例を次に示します。

（例）

ユーザ（ユーザ名：USER01　パスワード：PASS01）に CONNECT 権限を与えます。

＜記述例＞

```
GRANT CONNECT TO USER01 IDENTIFIED BY PASS01
```

HiRDB のユーザ権限の付与については，マニュアル「HiRDB　システム運用ガイド」を参照してください。また，GRANT 文の文法および詳細な注意事項などについては，マニュアル「HiRDB　SQL リファレンス」を参照してください。

3.10.4 HiRDB Text Search Plug-in での環境設定

全文検索および概念検索といった全文検索機能を利用する場合には，HiRDB のプラグインプログラムである HiRDB Text Search Plug-in が必要です。HiRDB Text Search Plug-in を使用する場合，HiRDB へのセットアップや RD エリアの追加などが必要です。HiRDB Text Search Plug-in の環境設定については，マニュアル「HiRDB Text Search Plug-in」を参照してください。また，プラグインプログラムの環境設定については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

3.11 DocumentBroker でデータベースシステムを使用するための設定

この節では，DocumentBroker でデータベースシステムを使用するために必要な設定について説明します。

3.11.1 データベース定義の名称の検討

メタ情報の初期設定コマンド（EDMInitMeta）を実行した場合の -u オプションで指定する値によって，クラス名やプロパティ名を表すデータベース定義の名称の定義方法が異なります。

(1) データベース定義の名称定義の方法

DocumentBroker で提供するデータベース定義の名称定義の方法を次に示します。

● クラス名やプロパティ名を使用する方法

● GUID 値を変換した値を使用する方法

新規に環境を構築する場合は，クラス名やプロパティ名をデータベース定義の名称として使用することをお勧めします。

クラス名やプロパティ名をデータベース定義の名称として使用する場合，クラス名は表識別子の制約を受け，プロパティ名は列名の制約を受けます。なお，GUID 値を変換した値をデータベース定義の名称定義として使用する場合は，DocumentBroker によって任意のデータベース定義の名称が設定されます。

(2) データベース定義の名称定義の規則

定義情報ファイルの dmaProp_DisplayName プロパティで，サブクラス名やプロパティ名をデータベース定義の名称定義として使用する場合の規則について説明します。ここでは，表識別子（表の名称として一意に識別できる値），列名，インデクス名（表を検索するためのキーとして列に付けた索引の名称），およびエリア名を定義する場合の規則を説明します。

クラス名やプロパティ名をデータベース定義の名称とする場合と，GUID 値を変換した識別子をデータベース定義の名称とする場合の名称定義の規則を次の表に示します。

表 3-8 データベース定義の名称定義の環境ごとの規則

データベース定義の名称定義の環境	クラス名やプロパティ名を，データベース定義の名称とする環境	GUID 値を変換した識別子を，データベース定義の名称とする環境
表識別子	サブクラス名	'T'+guid 変換文字列
列名	プロパティ名	'P'+guid 変換文字列
列名（二つのクラスを一つの表に格納する場合）	プロパティ名 プロパティ名 +'_1'	'P'+guid 変換文字列 'A'+guid 変換文字列
列名（三つのクラスを一つの表に格納する場合）	プロパティ名 プロパティ名 +'_1' プロパティ名 +'_2'	'P'+guid 変換文字列 'A'+guid 変換文字列 'B'+guid 変換文字列
VariableArray 用のクラス（外づけの表）	VariableArray プロパティ名	'T'+VariableArray 型プロパティ guid 変換文字列
VariableArray 型のプロパティ（外づけの表）	要素プロパティ名	'P'+ 要素プロパティ guid 変換文字列

データベース定義の名称定義の環境	クラス名やプロパティ名を，データベース定義の名称とする環境	GUID 値を変換した識別子を，データベース定義の名称とする環境
VariableArray 型のプロパティ（HiRDB の繰り返し列）	VariableArray プロパティ名 +'_'+ 要素プロパティ名	'P'+ 要素プロパティ guid 変換文字列 + プロパティ順序の変換文字列
インデクス名（キーが一つ，または指定省略時）	サブクラス名 +< クラスで一意文字列（改）>※	'I'+ クラスの guid 変換文字列 + プロパティ順序の変換文字列
インデクス名（キーが複数，または指定省略時）	サブクラス名 +< クラスで一意文字列（改）>※	'I'+ クラスの guid 変換文字列 +< クラスで一意文字列 >※
サブクラス名で表識別子の制約	あり	なし
プロパティ名で列名の制約	あり	なし

注※

< クラスで一意文字列 >=G ～ Z，GG ～ ZG，GH ～ ZH，GI ～ ZZ

< クラスで一意文字列（改）>=01 ～ VV（32 進文字列）

なお，システムクラスおよびシステムプロパティのデータベース定義の名称一覧は，「付録 F　システムクラスおよびシステムプロパティの名称定義の規則」を参照してください。また，サブクラス名に対応する表識別子とプロパティ名に対応する列名の詳細な規則については，「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」を参照してください。インデクス名を定義する場合の詳細な規則については，「4.9.2　インデクス名をデータベース定義の名称に使用する場合の規則」を参照してください。エリア名については，データベース定義の制約に従って定義してください。

3.11.2　データベースシステムの設定に必要なファイル

この節では，DocumentBroker でデータベースシステムを使用するための設定に必要なファイルについて説明します。データベースの設定を始める前に，次のファイルを準備，作成してください。

- メタ情報ファイル
- 定義情報ファイル
- RD エリア定義情報ファイル
- インデクス情報ファイル

(1) メタ情報ファイル

メタ情報ファイルは，DocumentBroker で使用するクラス（DMA で規定されているクラス）や検索に使用するオペレータなどについての定義情報が記述された，初期設定用のファイルです。メタ情報ファイルの詳細については，「4.6　メタ情報ファイル」を参照してください。

(2) 定義情報ファイル

定義情報ファイルは，ユーザの環境に合わせてサブクラスやプロパティを追加する場合に作成する，オブジェクトを定義するファイルです。例えば，全文検索機能を使用する場合は，dmaClass_DocVersion クラスのサブクラスの作成と全文検索用のプロパティを追加するために必要な情報を定義情報ファイルに記述します。

定義情報ファイルの作成については「4.7　定義情報ファイル」を参照してください。なお，このファイルに定義した情報は，メタ情報として追加されます。

(3) RD エリア定義情報ファイル

DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびメタ情報の追加コマンド

（EDMAddMeta）で出力されるデータベース定義文中の RD エリア名をユーザの指定した RD エリアで出力させるために，RD エリア定義情報ファイルを作成します。この場合，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびメタ情報の追加コマンド（EDMAddMeta）の -r オプションに RD エリア定義情報ファイル名を指定します。このファイルで指定された内容が，-o オプションで指定したデータベース定義文格納ファイル中の RD エリア名に反映されます。

これによって，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびメタ情報の追加コマンド（EDMAddMeta）で出力されるデータベース定義文中の RD エリアの手動による変更が不要になります。RD エリア定義情報ファイルの作成については，「4.8　RD エリア定義情報ファイル」を参照してください。

（4） インデクス情報ファイル

インデクスの作成をデータベース定義に追加する場合，インデクス情報ファイルを作成します。DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）およびメタ情報の追加コマンド（EDMAddMeta）を実行する場合に，インデクス情報ファイルを基にデータベース定義文にインデクス作成を定義します。インデクス情報ファイルで定義できる内容を次に示します。

- ユーザが定義するプロパティに対するインデクスの作成
- 全文検索インデクス用プロパティに対する差分インデクスの作成

インデクス情報ファイルの作成については，「4.9　インデクス情報ファイル」を参照してください。

3.11.3　データベースシステムの設定手順

ここでは，DocumentBroker でデータベースシステムを使用するための設定手順について説明します。なお，データベースシステムの設定は，システム管理者が実行してください。また，データベースを設定するユーザに対しては，あらかじめ CONNECT 権限およびスキーマ定義権限を付与しておいてください。

DocumentBroker で使用できるデータベースは HiRDB です。したがって，ここで説明するデータベースシステムの設定手順は，HiRDB を使用する場合の設定手順です。また，データベースシステムを設定する前に，HiRDB の環境設定とデータベースシステムを設定するために必要なファイルを作成しておいてください。HiRDB の環境設定については「3.10.2　HiRDB の環境設定」を，必要なファイルについては「3.11.2　データベースシステムの設定に必要なファイル」を参照してください。

DocumentBroker で使用するデータベースシステムの設定手順を次に示します。設定中は，HiRDB を起動させておいてください。HiRDB の起動方法については，マニュアル「HiRDB　システム運用ガイド」を参照してください。また，設定手順にある「(2)　SGML 定義情報を登録します」，「(7)　データベース定義文を修正します」および「(9)　ユーザ定義識別子ファイルの作成コマンド（EDMCreateIds）を実行します」については，必要に応じて設定してください。

（1） スキーマを定義します

スキーマを定義するためには，HiRDB のデータベース定義ユティリティ（pddef）を実行して，定義系 SQL の CREATE SCHEMA を使用します。スキーマを定義する場合は，スキーマ定義権限が必要です。HiRDB のデータベース定義ユティリティについては，マニュアル「HiRDB　コマンドリファレンス」を参照してください。CREATE SCHEMA の文法については，マニュアル「HiRDB　SQL リファレンス」を参照してください。

(2) SGML 定義情報を登録します

XML のような構造を持った文書を検索する場合，HiRDB Text Search Plug-in で次に示す SGML 定義情報を登録する必要があります。

(a) DTD の登録

DocumentBroker が提供する「EDM_DEFAULT.DTD」というファイルを「EDM_DEFAULT.DTD」という登録名称で DTD としてレジストリに登録する必要があります。次に示すコマンドを HiRDB 管理者が実行します。

```
phssgmlreg DTD EDM_DEFAULT.DTD /opt/HiEDMS/sample/EDM_DEFAULT.DTD
```

(b) 正規化パラメタファイルの登録

正規化パラメタファイルとは，XML 文書中のタグを制御するパラメタを記述するファイルです。XML 文書中のタグの制御が必要な場合または特定文字データ (SDATA) を使用する場合に登録してください。次に示すコマンドを HiRDB 管理者が実行します。

```
phssgmlreg NORparm 登録名称 正規化パラメタファイルのパス名
```

なお，SGML 定義情報の登録の詳細については，マニュアル「HiRDB Text Search Plug-in」を参照してください。

(3) メタ情報の初期設定コマンド（EDMInitMeta）を実行します

メタ情報をデータベースに登録するために，メタ情報の初期設定コマンド（EDMInitMeta）を実行します。メタ情報の初期設定コマンドの使用方法ついては，「7.3　コマンドの文法」を参照してください。動作環境メタ情報ファイルは，この時点で作成されます。

メタ情報の初期設定コマンドを実行する場合，データベースの表識別子と列名の名称定義の方法を -u オプションで指定します。また，アクセス制御機能を使用する場合は，-A オプションを指定してください。アクセス制御情報を含んだメタ情報がデータベースに登録されます。詳細については，「3.9.2(1)　データベーススキーマの作成」を参照してください。

-f オプションを指定した場合，実行環境識別子に 0 が割り当てられます。

複数の実行環境から同一の文書空間にアクセスするための実行環境の構築方法については，「3.15　複数の実行環境を構築する場合の設定」を参照してください。

(4) メタ情報の追加コマンド（EDMAddMeta）を実行します

作成した定義情報ファイルを入力ファイルとして，メタ情報の追加コマンド（EDMAddMeta）を実行することで，サブクラスおよびプロパティを追加します。定義情報ファイルについては，「4.7　定義情報ファイル」を参照してください。

メタ情報の追加コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。この時点で，動作環境メタ情報ファイルも更新されます。

(5) DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）を実行します

RD エリア定義情報ファイルを入力ファイルとして，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）を実行し，データベース定義文を出力します。さらに，ユーザが追加するプロパティ

にインデクスを定義する場合は，インデクス情報ファイルも入力ファイルとして指定してください。RD エリア定義情報ファイルについては「4.8　RD エリア定義情報ファイル」を，インデクス情報ファイルについては「4.9　インデクス情報ファイル」を参照してください。DocumentBroker 用データベース定義文の作成コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。また，全文検索機能を利用する場合のデータベース定義文の内容については，「3.11.4　全文検索機能を使用する場合に出力されるデータベース定義文」を参照してください。

(6) クラス定義情報ファイルの作成コマンド（EDMCrtSimMeta）を実行します

クラス定義情報ファイルの作成コマンド（EDMCrtSimMeta）を実行して，クラス定義情報ファイルを出力します。次の場合に，クラス定義情報ファイルを作成してください。

- Java クラスライブラリを使用する場合
- DocumentBroker オブジェクト操作ツールを使用する場合

このコマンドを実行する場合，メタ情報の追加コマンド（EDMAddMeta）によってサブクラスおよびプロパティの追加を完了させておく必要があります。

クラス定義情報ファイルについては，「4.10　クラス定義情報ファイル」を参照してください。クラス定義情報ファイルの作成コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。

(7) データベース定義文を修正します

出力したデータベース定義文の RD エリア名を，HiRDB のデータベース構成変更ユティリティで追加した RD エリアの名称に変更します。必要に応じてインデクスなどのデータベース属性も追加してください。ただし，RD エリア定義情報ファイルを作成して，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびメタ情報の追加コマンド（EDMAddMeta）の -r オプションに RD エリア定義情報ファイル名を指定した場合は，RD エリア名の変更は不要です。

全文検索機能を使用する場合に必要なデータベース定義文の変更については，「3.11.4　全文検索機能を使用する場合に出力されるデータベース定義文」を参照してください。

(8) データベース定義ユティリティを実行します

データベース定義文を入力ファイルとして，HiRDB でデータベース定義ユティリティ（pddef）を実行して定義を追加します。HiRDB のデータベース定義ユティリティについては，マニュアル「HiRDB　コマンドリファレンス」を参照してください。

(9) ユーザ定義識別子ファイルの作成コマンド（EDMCreateIds）を実行します

DocumentBroker で提供されているクラスおよびプロパティ以外にユーザクラスおよびユーザプロパティを定義して，ユーザアプリケーションプログラムを開発する場合，ユーザ定義識別子ファイルを作成してください。

ユーザ定義識別子ファイルについては，「4.11　ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイル」を参照してください。ユーザ定義識別子ファイルの作成コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。

3.11.4　全文検索機能を使用する場合に出力されるデータベース定義文

ここでは，全文検索機能を使用する場合に出力されるデータベース定義文について説明します。

(1) 出力されるデータベース定義文

全文検索機能を使用する場合，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）を実行すると，次に示すデータベース定義文が出力されます。

- 全文検索機能を使用する文書クラスを定義したデータベース定義文
- 概念検索機能を使用する文書クラスを定義したデータベース定義文
- 差分インデクス作成を定義したデータベース定義文
- 文字列型プロパティに対する全文検索機能を使用するためのプロパティを定義したデータベース定義文

以降，それぞれのデータベース定義文について説明します。なお，構造を指定した検索機能の定義については，それぞれのデータベース定義に含めて説明します。

(2) 全文検索機能を使用する文書クラスを定義したデータベース定義文

全文検索機能を使用する文書クラスに定義される全文検索インデクス用プロパティ（検索タームを指定する全文検索機能用）ごとに出力されるデータベース定義文について説明します。ここでは，次に示す全文検索インデクス用プロパティを定義したクラスのデータベース定義文について説明します。

- edmProp_TextIndex プロパティ
- edmProp_StIndex プロパティ
- edmProp_Content プロパティ

(a) edmProp_TextIndex プロパティを定義した場合

プレーンテキストを対象とした全文検索機能を使用する文書クラスに，edmProp_TextIndex プロパティを定義します。これによって，次の図に示すデータベース定義文が出力されます。

図 3-5 edmProp_TextIndex プロパティを定義した場合に出力されるデータベース定義文

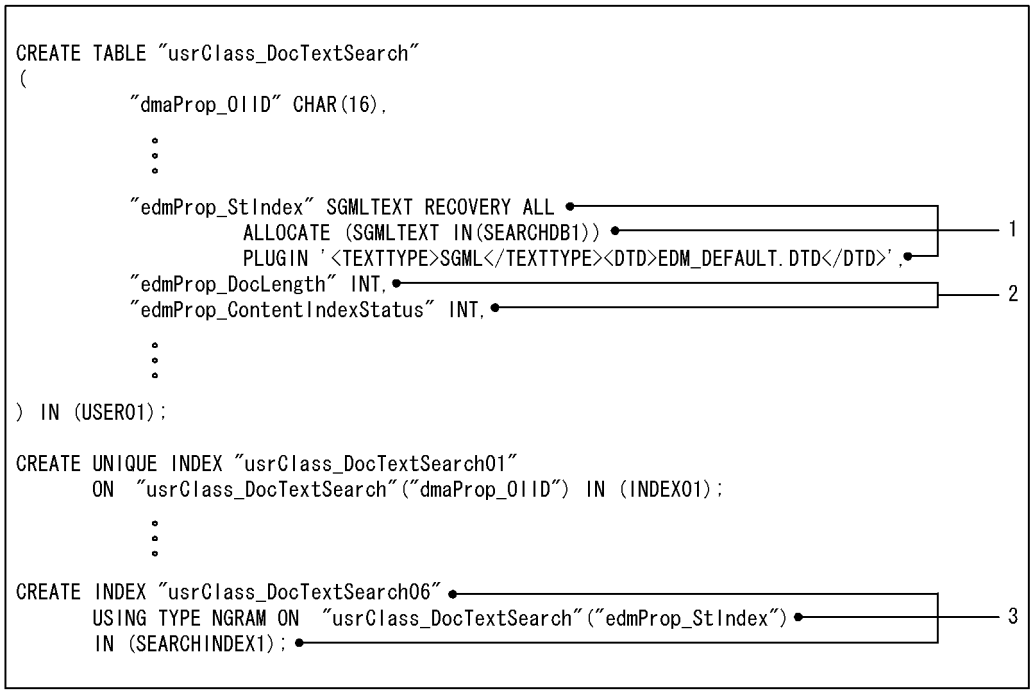
```
CREATE TABLE "usrClass_DocTextSearch"
(
          "dmaProp_OID" CHAR(16),
          "dmaProp_This" CHAR(52),
          "dmaProp_Parent" CHAR(52),
          "dmaProp_ParentContainer" CHAR(52),
          "dmaProp_CurrentOfSeriesCnt" INT,
          "edmProp_TextIndex" SGMLTEXT RECOVERY ALL
                    ALLOCATE (SGMLTEXT IN(SEARCHDB1))                    1
                    PLUGIN '<TEXTTYPE>PLAIN</TEXTTYPE>',
          "edmProp_DocLength" INT,                                       2
          "edmProp_ContentIndexStatus" INT,
            :
            :
            :
) IN (USER01);

CREATE UNIQUE INDEX "usrClass_DocTextSearch01"
       ON "usrClass_DocTextSearch"("dmaProp_OID") IN (INDEX01);
            :
            :
            :
CREATE INDEX "usrClass_DocTextSearch06"
       USING TYPE NGRAM ON  "usrClass_DocTextSearch"("edmProp_TextIndex")  3
       IN (SEARCHINDEX1);
```

(b) edmProp_StIndex プロパティを定義した場合

XML 文書のように構造を持つ文書を対象とした全文検索機能を使用する文書クラスに，edmProp_StIndex プロパティを定義します。これによって，次の図に示すデータベース定義文が出力され

ます。

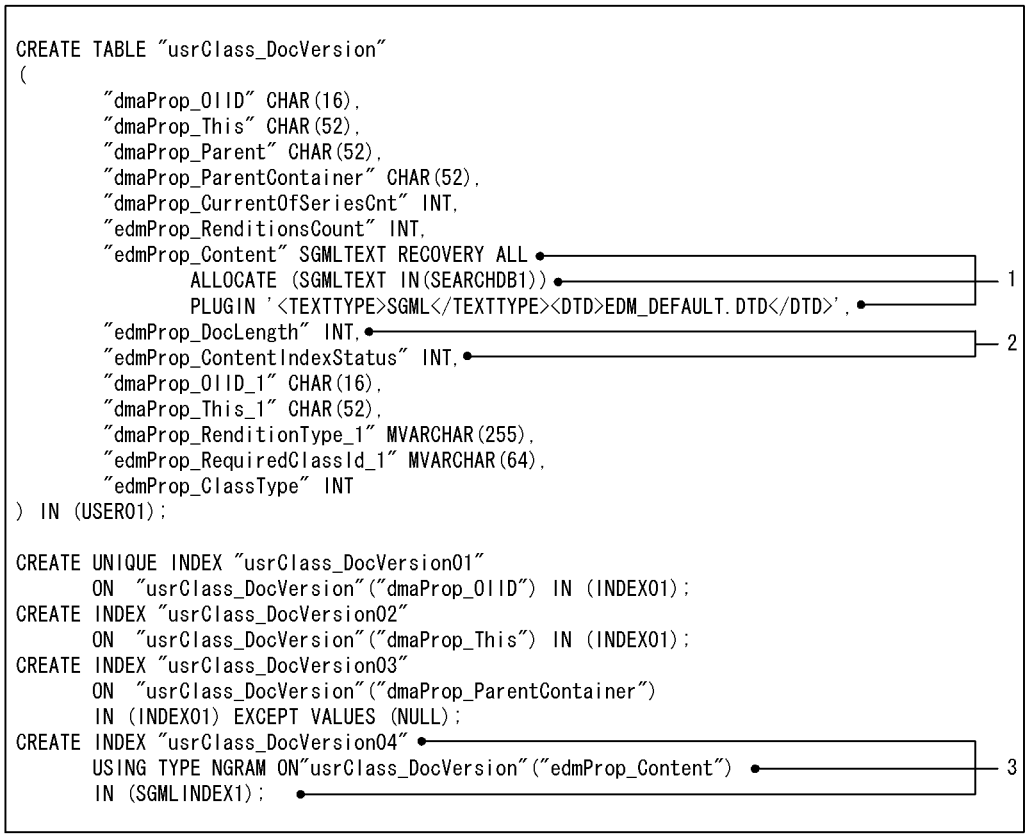
図 3-6　edmProp_StIndex プロパティを定義した場合に出力されるデータベース定義文



(c) edmProp_Content プロパティを定義した場合

プレーンテキストまたは構造を持つ文書（XML 文書）を対象とした全文検索機能を使用する文書クラスに，edmProp_Content プロパティを定義します。これによって，次の図に示すデータベース定義文が出力されます。なお，このプロパティは，Version 1 との互換用プロパティです。

図 3-7 edmProp_Content プロパティを定義した場合に出力されるデータベース定義文



(d) データベース定義文の定義内容の解説

図 3-5，図 3-6，および図 3-7 のデータベース定義文中の 1.，2.，および 3. の意味を次に示します。

1. 全文検索機能付き文書クラスのための列作成定義
   全文検索用の文書の内容を格納するためのユーザ LOB 用 RD エリアを使用します。例では，SEARCHDB1 というユーザ LOB 用 RD エリアを指定しています。
2. 全文検索機能付きクラスのための列作成定義
3. 全文検索機能付きクラスのためのインデクス作成定義
   全文検索用のユーザ LOB 用 RD エリアを使用します。例では，SEARCHINDEX1 または SGMLINDEX1 というユーザ LOB 用 RD エリアを指定しています。

(e) edmProp_Content プロパティを定義した場合のデータベース定義文の変更

全文検索機能を使用する文書クラスの全文検索インデクス用プロパティに edmProp_Content プロパティを追加している場合，必要に応じてデータベース定義文を次に示すように変更してください。

プレーンテキストを全文検索する場合のプラグイン宣言の変更

プレーンテキストの全文検索用クラスのデータベース定義文を変更する場合，図 3-7 のデータベース定義文中にある 1 のプラグイン宣言を，次のように変更します。

変更前

```
ALLOCATE (SGMLTEXT IN(SEARCHDB1))
PLUGIN '<TEXTTYPE>SGML</TEXTTYPE><DTD>EDM_DEFAULT.DTD</DTD>',
```

変更後

```
ALLOCATE (SGMLTEXT IN(SEARCHDB1))
```

```
    PLUGIN '<TEXTTYPE>PLAIN</TEXTTYPE>',
```

XML 文書を全文検索する場合の正規化パラメタファイル名の追加

XML 文書の全文検索用クラスのデータベース定義文で，文書中のタグを制御するか，特定文字データ（SDATA）を使用する場合は，<NORparm> に利用する正規化パラメタファイル名を指定します。この場合，図 3-7 のデータベース定義文中にある 1. の部分を次に示すように変更します。なお，DocumentBroker のデータベース定義では，必ず <DTD> に「EDM_DEFAULT.DTD」を設定します。

変更前

```
ALLOCATE (SGMLTEXT IN(SEARCHDB1))
PLUGIN '<TEXTTYPE>SGML</TEXTTYPE><DTD>EDM_DEFAULT.DTD</DTD>',
```

変更後

```
ALLOCATE (SGMLTEXT IN(SEARCHDB1))
PLUGIN '<TEXTTYPE>SGML</TEXTTYPE><DTD>EDM_DEFAULT.DTD</DTD>
        <NORparm>SAMPLE.NORM</NORparm>',
```

(3) 概念検索機能を使用する文書クラスを定義したデータベース定義文

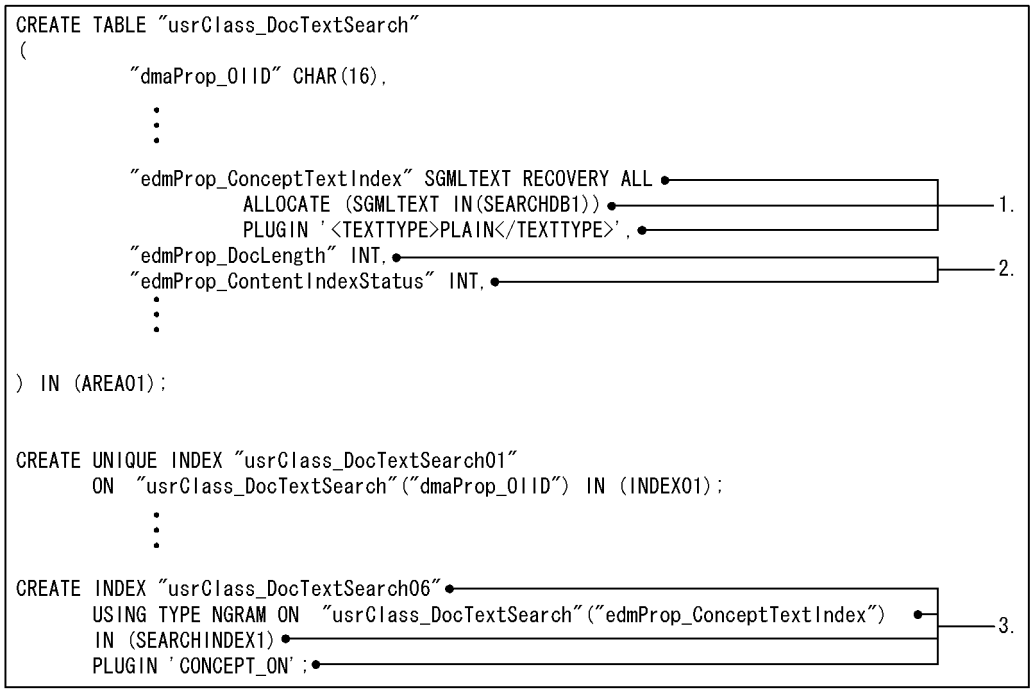
概念検索機能を使用する文書クラスに定義される全文検索インデクス用プロパティ（概念検索機能用）ごとに出力されるデータベース定義文について説明します。ここでは，次に示す全文検索インデクス用プロパティを定義したクラスのデータベース定義文について説明します。

- edmProp_ConceptTextIndex プロパティ
- edmProp_ConceptStIndex プロパティ

(a) edmProp_ConceptTextIndex プロパティを定義した場合

プレーンテキストを対象とする概念検索機能を使用する文書クラスに，edmProp_ConceptTextIndex プロパティを定義します。これによって，次の図に示すデータベース定義文が出力されます。

図 3-8 edmProp_ConceptTextIndex プロパティを定義した場合に出力されるデータベース定義文



(b) edmProp_ConceptStIndex プロパティを定義した場合

XML 文書のように構造を持つ文書を対象とした概念検索機能を使用する文書クラスに，edmProp_ConceptStIndex プロパティを定義します。これによって，次の図に示すデータベース定義文が出力されます。

図 3-9 edmProp_ConceptStIndex プロパティを定義した場合に出力されるデータベース定義文

```
CREATE TABLE "usrClass_DocTextSearch"
(
          "dmaProp_OIID" CHAR(16),
            :
          "edmProp_ConceptStIndex" SGMLTEXT RECOVERY ALL
                    ALLOCATE (SGMLTEXT IN(SEARCHDB1))                    1.
                    PLUGIN '<TEXTTYPE>SGML</TEXTTYPE><DTD>EDM_DEFAULT.DTD</DTD>,'
          "edmProp_DocLength" INT,                                       2.
          "edmProp_ContentIndexStatus" INT,
            :
) IN (AREA01);

CREATE UNIQUE INDEX "usrClass_DocTextSearch01"
       ON  "usrClass_DocTextSearch"("dmaProp_OIID") IN (INDEX01);
            :
CREATE INDEX "usrClass_DocTextSearch06"
       USING TYPE NGRAM ON  "usrClass_DocTextSearch"("edmProp_ConceptStIndex")   3.
       IN (SEARCHINDEX1)
       PLUGIN 'CONCEPT_ON';
```

(c) データベース定義文の定義内容の解説

図 3-8 および図 3-9 のデータベース定義文中の 1.，2.，および 3. の意味を次に示します。

1. 概念検索機能を使用する文書クラスのための列作成定義
   概念検索用の文書の内容を格納するためのユーザ LOB 用 RD エリアを使用します。例では，SEARCHDB1 というユーザ LOB 用 RD エリアを指定しています。
2. 概念検索機能を使用する文書クラスのための列作成定義
3. 概念検索機能を使用する文書クラスのためのインデクス作成定義
   概念検索用のユーザ LOB 用 RD エリアを使用します。例では，SEARCHINDEX1 というユーザ LOB 用 RD エリアを指定しています。

(4) 差分インデクス作成を定義したデータベース定義文

インデクス情報ファイルで，差分インデクスの作成を定義した場合のデータベース定義文を示します。ここでは，edmProp_TextIndex プロパティに差分インデクスの作成を定義した場合のデータベース定義文を次の図に示します。

図 3-10 edmProp_TextIndex プロパティに差分インデクスの作成を定義した場合に出力されるデータベース定義文

```
CREATE TABLE “usrClass_DocTextSearch”
(
          “dmaProp_OIID” CHAR(16),
                    :
                    :
) IN (USER01);
                    :
                    :
CREATE INDEX “usrClass_DocTextSearch06”
       USING TYPE NGRAM ON “usrClass_DocTextSearch”(“edmProp_TextIndex”)
       IN (SEARCHINDEX1)
       PLUGIN 'SUB_INDEX=51200';●──────────────────────────────────────1.
```

図 3-10 のデータベース定義文中の 1. の意味を次に示します。

1. 差分インデクスの作成および差分インデクスの容量の定義
   差分インデクスの作成，および作成する差分インデクスの容量を定義します。ここでは，51,200 キロバイトを定義しています。

なお，差分インデクス作成の定義は，インデクス情報ファイルに記述します。インデクス情報ファイルについては，「4.9 インデクス情報ファイル」を参照してください。

(5) 文字列型プロパティに対する全文検索機能を使用するためのプロパティを定義したデータベース定義文

ここでは，文字列型プロパティに対する全文検索機能を使用するためのプロパティを定義した場合に出力されるデータベース定義文について説明します。

usrClass_PropTextSearch クラスに対して，全文検索機能付き文字列型プロパティである usrProp_DocSummary プロパティを定義した場合に出力されるデータベース定義文を次の図に示します。

図 3-11 全文検索機能付き文字列型プロパティを定義した場合に出力されるデータベース定義文

```
CREATE TABLE "usrClass_PropTextSearch"
(
          "dmaProp_OIID" CHAR(16),
                    :
                    :
          "usrProp_DocSummary" FREEWORD●──────────────────────────────────1.
) IN (USERAREA);

CREATE UNIQUE INDEX "usrClass_PropTextSearch01"
               ON  "usrClass_PropTextSearch"("dmaProp_OIID") IN (INDEXAREA);
                    :
                    :
CREATE INDEX "usrClass_PropTextSearch02"●──────────────────────────┐
          USING TYPE IXFREEWORD ON "usrClass_PropTextSearch"("usrProp_DocSummary")●─ 2.
          IN (SEARCHINDEX1);●──────────────────────────────────────┘
```

図 3-11 のデータベース定義文中の 1. および 2. の意味を次に示します。

1. 全文検索機能付き文字列型プロパティのための列作成定義
2. 全文検索機能付き文字列型プロパティのためのインデクス作成定義
   全文検索機能付き文字列型プロパティ用のユーザ LOB 用 RD エリアを使用します。例では，

SEARCHINDEX1 というユーザ LOB 用 RD エリアを指定しています。

(6) データベース定義文中の RD エリア名についての注意事項

全文検索機能を使用する場合に出力されるデータベース定義文に定義されている RD エリア名について説明します。

これらの RD エリア名は,「3.10.2　HiRDB の環境設定」で作成した RD エリア名に変更する必要があります。ただし，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）の -r オプションで RD エリア定義情報ファイル名を指定している場合，この作業は不要です。RD エリア定義情報ファイルについては,「4.8　RD エリア定義情報ファイル」を参照してください。

3.12 システム導入支援機能での設定

システム導入支援機能は，新規に文書空間を構築するための設定を支援する機能です。システム導入支援機能を使用すると，DocumentBroker でデータベースシステムを使用するための設定が自動でできるため，システム導入支援機能を使用しないで文書空間を構築する場合に比べて，文書空間の構築を簡易化できます。

「3.11.3　データベースシステムの設定手順」で説明している次の手順の代わりに，「(1) システム導入支援機能を使用する場合のデータベースシステムの設定」を実行します。

- メタ情報の初期設定コマンド（EDMInitMeta）の実行（参照先：3.11.3(3)）
- メタ情報の追加コマンド（EDMAddMeta）の実行（参照先：3.11.3(4)）
- DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）の実行（参照先：3.11.3(5)）
- クラス定義情報ファイルの作成コマンド（EDMCrtSimMeta）の実行（参照先：3.11.3(6)）

また，上書きインストールして，システム導入支援コマンドを使用する場合の設定手順についても説明します。

(1) システム導入支援機能を使用する場合のデータベースシステムの設定

システム導入支援機能を使用する場合のデータベースシステムの設定手順について説明します。

「3.11.3(2) SGML 定義情報を登録します」までを実行したあと，次に示す手順 1. ～ 3. を実行します。手順 1. ～ 3. が終わったら，「3.11.3(7) データベース定義文を修正します」以降に説明する手順を実行してください。

1. **文書空間定義コマンドの実行に必要な文書クラス定義ファイル，および文書空間情報ファイルを作成します。**
   文書クラス定義ファイルについては，「4.15　文書クラス定義ファイル」を参照してください。文書空間情報ファイルについては，「4.16　文書空間情報ファイル」を参照してください。
2. **文書空間定義コマンド（EDMCDefDocSpace）を実行します。**
   文書空間定義コマンド（EDMCDefDocSpace）については，「7.3　コマンドの文法」の「EDMCDefDocSpace（文書空間の定義）」を参照してください。
3. **文書空間構築コマンド（EDMCBuildDocSpace）を実行します。**
   文書空間が新規に構築されます。文書空間構築コマンド（EDMCBuildDocSpace）については，「7.3 コマンドの文法」の「EDMCBuildDocSpace（文書空間の構築）」を参照してください。

(2) 上書きインストールして，システム導入支援コマンドを使用する場合の設定

上書きインストールして，システム導入支援コマンドで文書空間を新規に構築する場合，上書きインストール後に次の手順 1.，2. で文書空間を定義します。また，上書きインストールして，システム導入支援コマンドで上書きインストール前の文書空間の定義で文書空間を新規に構築する場合，上書きインストール後に次の手順 1. ～ 3. で文書空間を再定義します。

1. **文書空間の定義をバックアップします。**
   「実行環境ディレクトリ /etc」下のファイルを任意のディレクトリへコピーします。
2. **メタ情報ファイルを更新します。**
   「実行環境ディレクトリ /adm/etc_org」の下位にあるすべてのファイルを「実行環境ディレクトリ /etc」下にコピーします。
3. **文書空間を定義します。**

コピー先の docspace.ini，docaccess.ini，userperm.ini，edmrpt.ini，propex.ini，process.ini，getrascustom.ini を上書きインストール前の実行環境で変更している場合は，1. でバックアップしたファイルを基に，コピーしたファイルを再設定してください。

3.13 ファイル転送機能を使用する場合の設定

この節では，DocumentBroker クライアントで動作させるファイル転送機能の設定方法について説明します。

3.13.1 ファイル転送機能の概要

DocumentBroker サーバに接続する DocumentBroker クライアントが異なるマシン上に存在する場合，DocumentBroker サーバと DocumentBroker 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，DocumentBroker クライアントで開始したファイル転送サービスが提供しています。ただし，ファイル転送サービスを使用するには，DocumentBroker クライアントでの環境設定が必要です。

(1) ファイル転送サービス開始モード

ファイル転送サービスには，次の二つの開始モードがあります。

- 静的モード
- 動的モード

(a) 静的モード

クライアントアプリケーションがファイル転送サービスを使用する前に，ファイル転送サービスをコマンドによって明示的に開始するモードです。

(b) 動的モード

クライアントアプリケーションで最初にファイル転送サービスを使用する時点で，ファイル転送サービスが自動的に開始されるモードです。

このモードは，クライアントアプリケーションを実行するときにプロセスが起動されるため，クライアントアプリケーションの処理時間が多く掛かります。

(2) ファイル転送サービスのプロセス構成

ファイル転送サービスのプロセス構成について，次の表に示します。なお，「(静的モードの場合)」とあるプロセスは，開始モードが静的モードの場合だけ使用するプロセスです。これらのプロセス間の通信には，メッセージキューを利用しています。

表 3-9 ファイル転送サービスのプロセス構成

プロセス名	機能概要
ファイル転送サービス開始	ファイル転送サービスを開始する（静的モードの場合）。
ファイル転送サービス監視	ファイル転送サービスを監視する（静的モードの場合）。
ファイル転送サービス	ファイル転送サービスを提供する。
ファイル転送サービス停止	ファイル転送サービスを停止する（静的モードの場合）。

(a) ファイル転送サービス開始プロセス（静的モードの場合）

ファイル転送サービスを，静的モードで開始するプロセスです。

このプロセスは，静的モードのファイル転送サービス開始コマンド（FtpSvStart）を実行したときに生成され，ファイル転送サービス監視プロセスを起動します。そして，ファイル転送サービス監視プロセスから起動完了メッセージの応答を確認すると終了します。

(b) ファイル転送サービス監視プロセス（静的モードの場合）

静的モードで開始されたファイル転送サービスプロセスを管理するプロセスです。

静的モードのファイル転送サービス開始コマンド（FtpSvStart）で指定した数のファイル転送サービスプロセスを起動します。そのあと，ファイル転送サービスプロセスの動作状況を監視して，ファイル転送サービスプロセスが終了した場合は再起動します。さらに，ファイル転送サービスを使用するクライアントアプリケーションに対して，最適なファイル転送サービスプロセスを割り当てます。

このプロセスは，ファイル転送サービス開始プロセスによって生成されます。ファイル転送サービス開始制御プロセスからの終了要求メッセージを受信すると，ファイル転送サービスプロセスに終了要求メッセージを通知し，ファイル転送サービスプロセスの終了を確認すると停止します。

(c) ファイル転送サービスプロセス

ファイル転送サービスを提供するプロセスです。

このプロセスは，同時に複数実行できます。

開始モードが静的モードの場合

ファイル転送サービスプロセスは，静的モードのファイル転送サービス開始コマンド（FtpSvStart）を実行した時に，このコマンドで指定した数だけファイル転送サービス監視プロセスによって生成されます。このプロセスは，クライアントアプリケーションに対して文書空間への接続単位（セッション単位）に割り当てられます。クライアントアプリケーションでファイル転送を要求すると，ファイル転送サービスプロセスの中から，割り当て数が最も少ないプロセスが割り当てられます。また，静的モードのファイル転送サービス停止コマンド（FtpSvStop）を実行したときに，ファイル転送サービス監視プロセスから終了要求メッセージを受信し，すべてのファイル転送サービスプロセスが終了します。

静的モードで開始されたファイル転送サービスプロセスは，一つのプロセスで，クライアントマシン上で動作するすべてのクライアントアプリケーションプロセスの，すべてのセッションに対してサービスを提供できます。ただし，ファイル転送サービスプロセスがダウンした場合，ダウンしたプロセスに割り当てられていたセッションを使用しファイル転送を要求すると，その要求したセッションに対してエラーが返却されます。この場合，ファイル転送の失敗でエラーが発生した API を再度実行して，ファイル転送を要求すると新たなファイル転送サービスプロセスが割り当てられます。

開始モードが動的モードの場合

ファイル転送サービスプロセスは，クライアントアプリケーションプロセス内で最初にファイル転送サービスを利用するときに，クライアントアプリケーションプロセスによって生成されます。したがって，ファイル転送サービスを利用するクライアントアプリケーションプロセスの数だけファイル転送サービスプロセスが生成されます。また，起動したクライアントアプリケーションプロセスが終了すると，停止します。

動的モードで開始されたファイル転送サービスプロセスは，そのプロセスを起動したクライアントアプリケーションプロセス内で動作するセッションに対してだけ，サービスを提供できます。ただし，ファイル転送サービスプロセスがダウンした場合，ダウンしたプロセスを生成したクライアントアプリケーションプロセスがファイル転送を要求すると，エラーが返却されます。この場合，ファイル転送の失敗でエラーが発生した API を再度実行して，ファイル転送を要求すると新たなファイル転送

サービスプロセスが生成され，サービスが提供されます。

(d) ファイル転送サービス停止プロセス（静的モードの場合）

静的モードで開始されたファイル転送サービスを停止するプロセスです。

このプロセスは，静的モードのファイル転送サービス停止コマンド（FtpSvStop）を実行したときに生成され，ファイル転送サービス監視プロセスに対して終了要求メッセージを送信します。ファイル転送サービス監視プロセスからの終了完了メッセージの応答を確認すると終了します。

(3) 静的モードと動的モードの違い

静的モードと動的モードの違いを，次の表に示します。

表 3-10 静的モードと動的モードの違い

項目	静的モード	動的モード
開始方法および停止方法	手動（コマンド）でファイル転送サービスを開始または停止する。	クライアントアプリケーション実行時に自動的にファイル転送サービスが開始または停止される。
ファイル転送サービスプロセス割り当て単位	文書空間への接続単位	クライアントアプリケーションプロセス単位
DocumentBroker サーバのファイルにアクセスするときのアクセス権	ファイル転送サービス開始実行ユーザのアクセス権	クライアントアプリケーションプロセス実行ユーザのアクセス権
DocumentBroker サーバから転送されるファイルの所有者	ファイル転送サービス開始実行ユーザ	クライアントアプリケーションプロセス実行ユーザ

動的モード使用時の注意事項

動的モードでファイル転送機能を使用する場合，ファイル転送サービスプロセスはクライアントアプリケーションプロセスの数だけ作成されます。また，プロセスの起動ごとに，一時的に使用するメッセージキューを一つ作成します。このため，リソース不足が発生する可能性があります。なお，メッセージキューの設定については，「3.4.2 パラメタのカスタマイズ」を参照してください。

リソース不足が発生した場合は，次のどちらかの対処をしてください。

- 静的モードに変更してファイル転送サービスを開始する
- ファイル転送サービスを使用するクライアントアプリケーションプロセス数を減少させる

3.13.2 ファイル転送機能のための DocumentBroker クライアントでの環境設定

ここでは，ファイル転送機能を使用するための，DocumentBroker クライアントの環境設定について説明します。次に示す手順で環境設定を実行してください。

(1) ファイル転送サービス実行環境ディレクトリの作成

ファイル転送サービスの実行環境を作成するためのディレクトリを作成してください。ここで，作成したディレクトリの下に，ファイル転送サービスの各種ディレクトリおよびファイルが格納されます。ディレクトリ名称は任意ですが，次に示す情報を設定してください。

● ディレクトリ所有者
ファイル転送サービスセットアップコマンド（FtpSvSetup）実行者

● ディレクトリへのアクセスパーミッション
0755

(2) 環境変数の設定（AIX の場合）

ファイル転送サービスを使用するために，クライアントアプリケーション実行環境とファイル転送サービス実行環境に必要な環境変数を設定してください。設定が必要な環境変数について次に説明します。

(a) LANG の設定

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。接続先の DocumentBroker サーバ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
- ファイル転送サービスが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。

(b) TZ の設定

次の値を指定します。

JST-9

(c) _HIEDMS_FTPDIR の設定

ファイル転送サービスの実行環境ディレクトリを指定してください。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 コードで値を設定してください。

(d) DocumentBroker クライアントのライブラリの設定

DocumentBroker クライアントに必要なライブラリのディレクトリを次のように設定します。

LIBPATH の設定

- TPBroker V3 を使用する場合
  :/opt/HiEDMS/client/lib
- TPBroker V5 を使用する場合
  :/opt/HiEDMS/client/lib_tp5

(e) PATH の設定

次の値を追加します。

: **ファイル転送サービスの実行環境ディレクトリ** /bin

(f) _HIEDMS_FTPMODE の設定

使用するファイル転送サービスの開始モードを指定します。次のどれかを指定してください。なお，この環境変数は，クライアントアプリケーション実行環境だけに必要です。

STATIC
静的モードで開始されたファイル転送サービスを使用します。

DYNAMIC
動的モードで開始されたファイル転送サービスを使用します。

NONE
ファイル転送サービスを使用しません。

この環境変数の設定を省略した場合，「NONE」が仮定されます。

(g) _HIEDMS_FTPORBBOA_OPTION の設定

動的モードで開始されるファイル転送サービスプロセスに固有の ORB および BOA のオプションを指定します。この環境変数は，クライアントアプリケーション実行環境だけに必要です。

この環境変数の設定を省略した場合に仮定される値はありません。ORB および BOA のオプション以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。

動的モードで開始されるファイル転送サービスプロセスの ORB および BOA のオプションは，ファイル転送サービス環境定義ファイルの [FtpService] セクションの FtpProcessOrbBoaOption エントリの値と，この環境変数に設定した値を連結した値になります。

静的モードで開始する場合，この環境変数を設定しても無視されます。静的モードで開始する場合は，ファイル転送サービス環境定義ファイルの [FtpProcessXXXX] セクション（XXXX は 0001 ～ 0020）の OrbBoaOption エントリに設定してください。

なお，この環境変数は，TPBroker V5 と連携して動作する環境では指定できません。

TPBroker V5 環境で指定した場合は，指定が無視されます。TPBroker V5 環境では，VisiBroker プロパティとして _HIEDMS_FTPVB_PROPERTY に指定してください。

(h) _HIEDMS_FTPVB_PROPERTY の設定

動的モードで開始されるファイル転送サービスプロセスに固有の VisiBroker プロパティを指定します。この環境変数は，クライアントアプリケーション実行環境だけに必要です。

この環境変数の設定を省略した場合に仮定される値はありません。VisiBroker プロパティ以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。指定できるプロパティの詳細は，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。

動的モードで開始されるファイル転送サービスプロセスの VisiBroker プロパティは，ファイル転送サービス環境定義ファイルの [FtpService] セクションの FtpProcessVBProperty エントリの値と，この環境変数に設定した値を連結した値になります。

静的モードで開始する場合，この環境変数を設定しても無視されます。静的モードで開始する場合は，ファイル転送サービス環境定義ファイルの [FtpProcessXXXX] セクション（XXXX は 0001 ～ 0020）の VBProperty エントリに設定してください。

なお，この環境変数は，TPBroker V3 と連携して動作する環境では指定できません。

TPBroker V3 環境で指定した場合は，指定が無視されます。TPBroker V3 環境では，ORB ／ BOA オプションとして _HIEDMS_FTPORBBOA_OPTION に指定してください。

(i) PSALLOC の設定

メモリ領域の確保をメモリ使用時でなく，メモリ領域要求時に行う，早期ページスペース割り当てを実行するための設定をします。次の値を指定します。

early

(j) NODISCLAIM の設定

メモリ領域の解放を抑止するための設定をします。次の値を指定します。

true

(3) 環境変数の設定（Linux の場合）

ファイル転送サービスを使用するために，クライアントアプリケーション実行環境とファイル転送サービス実行環境に必要な環境変数を設定してください。設定が必要な環境変数について次に説明します。

(a) LANG の設定

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。接続先の DocumentBroker サーバ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.SJIS」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
- ファイル転送サービスが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.SJIS」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.SJIS」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。

(b) TZ の設定

次の値を指定します。

JST-9

(c) _HIEDMS_FTPDIR の設定

ファイル転送サービスの実行環境ディレクトリを指定してください。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 コードで値を設定してください。

(d) DocumentBroker クライアントのライブラリの設定

DocumentBroker クライアントに必要なライブラリのディレクトリを次のように設定します。

LD_LIBRARY_PATH の設定

:/opt/HiEDMS/client/lib

(e) PATH の設定

次の値を追加します。

: **ファイル転送サービスの実行環境ディレクトリ** /bin

(f) _HIEDMS_FTPMODE の設定

使用するファイル転送サービスの開始モードを指定します。次のどれかを指定してください。なお，この環境変数は，クライアントアプリケーション実行環境だけに必要です。

STATIC
　静的モードで開始されたファイル転送サービスを使用します。

DYNAMIC
　動的モードで開始されたファイル転送サービスを使用します。

NONE
　ファイル転送サービスを使用しません。

この環境変数の設定を省略した場合，「NONE」が仮定されます。

(g) _HIEDMS_FTPORBBOA_OPTION の設定

動的モードで開始されるファイル転送サービスプロセスに固有の ORB および BOA のオプションを指定します。この環境変数は，クライアントアプリケーション実行環境だけに必要です。

この環境変数の設定を省略した場合に仮定される値はありません。ORB および BOA のオプション以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。

動的モードで開始されるファイル転送サービスプロセスの ORB および BOA のオプションは，ファイル転送サービス環境定義ファイルの [FtpService] セクションの FtpProcessOrbBoaOption エントリの値と，この環境変数に設定した値を連結した値になります。

静的モードで開始する場合，この環境変数を設定しても無視されます。静的モードで開始する場合は，ファイル転送サービス環境定義ファイルの [FtpProcessXXXX] セクション（XXXX は 0001 ～ 0020）の OrbBoaOption エントリに設定してください。

なお，この環境変数は，TPBroker と連携して動作する環境では指定できません。

TPBroker 環境で指定した場合は，指定が無視されます。TPBroker 環境では，VisiBroker プロパティとして _HIEDMS_FTPVB_PROPERTY に指定してください。

(h) _HIEDMS_FTPVB_PROPERTY の設定

動的モードで開始されるファイル転送サービスプロセスに固有の VisiBroker プロパティを指定します。この環境変数は，クライアントアプリケーション実行環境だけに必要です。

この環境変数の設定を省略した場合に仮定される値はありません。VisiBroker プロパティ以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。指定できるプロパティの詳細は，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。

動的モードで開始されるファイル転送サービスプロセスの VisiBroker プロパティは，ファイル転送サービス環境定義ファイルの [FtpService] セクションの FtpProcessVBProperty エントリの値と，この環境変数に設定した値を連結した値になります。

静的モードで開始する場合，この環境変数を設定しても無視されます。静的モードで開始する場合は，ファイル転送サービス環境定義ファイルの [FtpProcessXXXX] セクション（XXXX は 0001 ～ 0020）の VBProperty エントリに設定してください。

(i) PSALLOC の設定

メモリ領域の確保をメモリ使用時でなく，メモリ領域要求時に行う，早期ページスペース割り当てを実行するための設定をします。次の値を指定します。

early

(j) NODISCLAIM の設定

メモリ領域の解放を抑止するための設定をします。次の値を指定します。

true

(4) ファイル転送サービス実行環境の作成

ファイル転送サービスセットアップコマンドを実行してください。

AIX の場合

TPBroker V3 を使用する場合

ファイル転送サービスセットアップコマンド（/opt/HiEDMS/client/bin/FtpSvSetup）を実行してください。

TPBroker V5 を使用する場合

ファイル転送サービスセットアップコマンド（/opt/HiEDMS/client/bin_tp5/FtpSvSetup）を実行してください。

Linux の場合

ファイル転送サービスセットアップコマンド（/opt/HiEDMS/client/bin/FtpSvSetup）を実行してください。

ファイル転送サービスセットアップコマンドの文法と注意事項などの詳細については，「3.13.4　ファイル転送機能で使用するコマンド」を参照してください。

(5) ファイル転送サービス環境定義ファイルの設定

ファイル転送機能を使用するためには，ファイル転送サービス環境定義ファイルの定義が必要になります。ファイル転送サービス環境定義ファイルについては，「4.12　ファイル転送サービス環境定義ファイル（ftpsv.ini）」を参照してください。

(6) ファイル転送サービスのメモリ所要量の見積もり

ファイル転送サービスを使用する場合のメモリ所要量を算出します。計算式を次に示します。

計算式（単位：メガバイト）

AIX の場合

- TPBroker V3 を使用するとき
  7 ＋（4 × p）＋（0.02 ＋ d）× u
- TPBroker V5 を使用するとき
  8 ＋（6 × p）＋（0.02 ＋ d）× u

Linux の場合

138 ＋（125 × p）＋（0.02 ＋ d）× u

ただし，d ＝（4 × s）

p，u，および s に設定する値の説明を次の表に示します。なお，s の単位は，メガバイトです。

表 3-11　ファイル転送サービスの所要メモリの計算式で使用する変数と設定する値

変数	指定する値
p	ファイル転送サービスプロセス数
u	同時にファイル転送するセッション数
s	次の 1. と 2. で算出した値の中での最大値 1. **マルチファイル文書の場合** DocumentSpace 構成定義ファイルの BlobSubstrMode エントリに指定する値によって，算出値が異なります。 ALL または ELEMENT の場合：文書空間で管理する最大ファイルサイズ（メガバイト） THRESHOLD の場合：BlobSubstrThreshold エントリ値 2. **上記以外の文書の場合** 文書空間で管理する最大文書サイズ（メガバイト）

(7) ファイル転送サービスの開始モードの設定

DocumentBroker クライアント側での環境設定の際に，ファイル転送サービスの開始モードとして，「静的モード」または「動的モード」のどちらかを設定する必要があります。開始モードは，環境変数「_HIEDMS_FTPMODE」で設定します。環境変数「_HIEDMS_FTPMODE」については，「(2)　環境変数の設定（AIX の場合）」または「(3)　環境変数の設定（Linux の場合）」を参照してください。

3.13.3 ファイル転送機能のための DocumentBroker サーバでの環境設定

ここでは，ファイル転送機能を使用するための，DocumentBroker サーバの環境設定について説明します。

(1) 環境変数の設定

ファイル転送機能を使用するために，DocumentBroker サーバ側で次の環境変数を設定します。

_HIEDMS_CON_TIMEOUT

ファイル転送サービスへの接続要求に対して，サービスプロセスで接続を確立するまでの待ち時間を，秒で設定します。設定できる値は，0 ～ 2,147,483,647 です。「0」を指定した場合，TCP/IP で固有の，サーバへの接続を確立するまでの待ち時間が設定されます。
範囲外の値を指定した場合，または環境変数を設定しない場合，「0」が仮定されます。

(2) ファイル分割転送機能の設定

指定したデータ転送サイズにファイルを分割して転送する，ファイル分割転送機能を使用する場合は，DocumentSpace 構成定義ファイルの FtpBufferSize エントリにデータ転送サイズを指定します。FtpBufferSize エントリの詳細については，「4.2.2　DocumentSpace 構成定義ファイルの記述形式」を参照してください。

ファイル分割転送機能の特長は，この機能を使用しない場合と比べて，DocumentBroker サーバおよび DocumentBroker クライアントのメモリ所要量を削減できることです。ファイル分割転送機能を使用しな

い場合は，転送するファイルの容量の大小に関係なく，1 回にファイルサイズ分のデータを転送します。一方，ファイル分割転送機能を使用する場合は，転送するファイルの容量が指定したデータ転送サイズを超えていれば，指定したデータ転送サイズにファイルを分割して転送します。このように，ファイル分割転送機能を使用すると，データ転送サイズよりも容量の大きいファイルは分割して転送するため，この機能を使用しない場合と比べて，ファイル転送時に使用するメモリ容量は少なくなります。ただし，ファイル転送回数の増加によって，処理性能が劣化する可能性があります。

DocumentBroker サーバおよび DocumentBroker クライアントのメモリ容量，ならびに処理性能のバランスを考慮して，データ転送サイズを決定してください。なお，ファイル分割転送機能を使用しない場合のクライアントのメモリ所要量については，「3.13.2(6)　ファイル転送サービスのメモリ所要量の見積もり」を参照してください。

3.13.4　ファイル転送機能で使用するコマンド

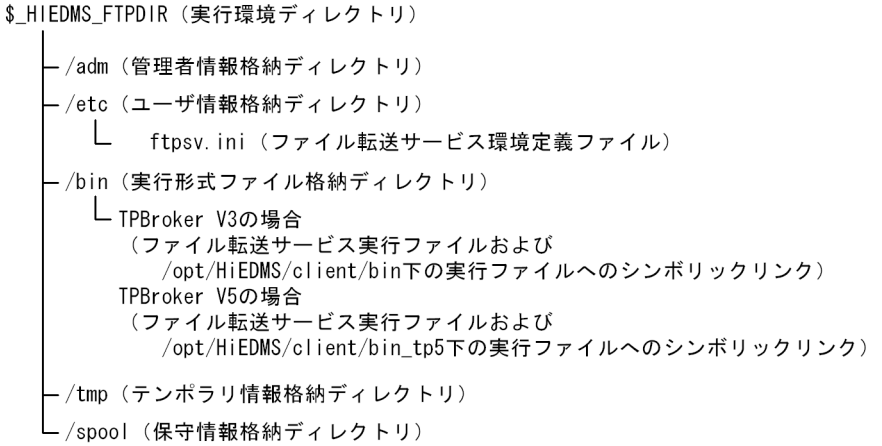
ここでは，ファイル転送サービスを使用するためのコマンドについて説明します。

(1) FtpSvSetup（ファイル転送サービスセットアップ）

機能

ファイル転送サービスを使用するための環境を作成します。
コマンドを実行すると，指定した＜実行環境ディレクトリパス＞下に，図 3-12 に示すディレクトリ構成で，ファイル転送サービスの実行環境が作成されます。なお，AIX を使用する場合で，TPBroker V3 を使用するときは，「/opt/HiEDMS/client/bin」ディレクトリに格納されている FtpSvSetup コマンドを実行してください。AIX を使用する場合で，TPBroker V5 を使用するときは，「/opt/HiEDMS/client/bin_tp5」ディレクトリに格納されている FtpSvSetup コマンドを実行してください。Linux を使用する場合は，「/opt/HiEDMS/client/bin」ディレクトリに格納されている FtpSvSetup コマンドを実行してください。

図 3-12　ファイル転送サービスセットアップコマンド（FtpSvSetup）で作成される実行環境のディレクトリ構成



このコマンドの終了後，ファイル転送サービス環境定義ファイル（実行環境ディレクトリ /etc/ftpsv.ini）を設定して，ファイル転送サービスを開始してください。
なお，このコマンドを 1 回実行すれば，同じマシンで実行するすべてのクライアントアプリケーションでファイル転送機能が使用できるようになります。

形式

FtpSvSetup　実行環境ディレクトリパス

コマンド引数

実行環境ディレクトリパス

ファイル転送サービスの実行環境ディレクトリへのパスを，絶対パスで指定します。

終了ステータス

0：正常終了

2：コマンドライン不正

3：エラー発生

注意事項

- 実行環境ディレクトリパスに，インストールディレクトリ（/opt/HiEDMS）以下のパスを指定することはできません。
- 実行環境ディレクトリとして指定したディレクトリが存在しない場合，または指定したディレクトリに対してコマンドを実行したユーザがアクセス権（読み取り・書き込み・実行（検索））を持っていない場合は，エラーになります。コマンドを実行する前に，指定するディレクトリについて確認してください。
- 実行環境作成時に，同名のファイルまたはディレクトリがすでに存在した場合は，新しくファイルまたはディレクトリは作成しません。
- ファイル転送サービスの実行環境を削除する場合は，OS のコマンドを実行して削除してください。

(2) FtpSvStart（静的モードのファイル転送サービス開始）

機能

静的モードでファイル転送サービスを開始します。

形式

FtpSvStart 〔-n　起動プロセス数〕

オプション

-n **起動プロセス数**

ファイル転送サービスプロセスの起動プロセス数を指定します。指定できる値は，1 ～ 20 です。ファイル転送サービスプロセスは，1 プロセス当たり複数のクライアントに対してサービスを提供できます。ただし，ファイル転送サービスプロセスがダウンした場合，ダウンしたプロセスが割り当てられていたすべてのユーザはファイル転送サービスが受けられなくなります。したがって，メモリの使用効率と障害が発生したときの影響を考慮してファイル転送サービスプロセスの数を決定してください。なお，ファイル転送サービス使用時のメモリ所要量の見積もりについては，「3.13.2(6)　ファイル転送サービスのメモリ所要量の見積もり」を参照してください。

このオプションを省略した場合は，「1」が仮定されます。

終了ステータス

0：正常終了

2：コマンドライン不正

3：エラー発生

注意事項

- このコマンドを実行できるのは，ファイル転送サービスセットアップコマンド（FtpSvSetup）を実行したユーザだけです。
- このコマンドは，静的モードで開始されたファイル転送サービスを使用するクライアントアプリケーション（環境変数「_HIEDMS_FTPMODE」に STATIC を指定しているクライアントアプリ

ケーション）を実行する前に実行してください。
- 同一のファイル転送サービス実行環境で，すでに静的モードでファイル転送サービスを開始している場合，このコマンドを再度実行することはできません。

(3) FtpSvStop（静的モードのファイル転送サービス停止）

機能

静的モードで開始されたファイル転送サービスを停止します。

形式

FtpSvStop 〔-l FORCE〕

オプション

-l FORCE

ファイル転送サービスを強制的に停止します。通常の停止処理がエラーになり，ファイル転送サービスが停止できない場合に，このオプションを指定してください。

終了ステータス

0：正常終了
2：コマンドライン不正
3：エラー発生

注意事項

- このコマンドを実行できるのは，ファイル転送サービスセットアップコマンド（FtpSvSetup）を実行したユーザだけです。
- このコマンドを複数同時に実行すると，タイムアウトが発生する場合があります。
- ファイル転送サービスを停止する前に，ファイル転送サービスを使用するアプリケーションが動作中でないことを確認してください。

3.13.5 ファイル転送サービスの開始と停止

ここではファイル転送機能の開始方法と停止方法について説明します。

(1) ファイル転送サービスの開始方法

環境変数に静的モードを指定した場合，クライアントアプリケーションを開始する前にファイル転送サービスを開始する必要があります。ファイル転送サービス開始コマンド（FtpSvStart）の指定方法については，「3.13.4 ファイル転送機能で使用するコマンド」を参照してください。

(2) ファイル転送サービスの停止方法

ファイル転送サービスを停止する前に，ファイル転送サービスを使用するアプリケーションが動作中でないことを確認してください。ファイル転送サービス停止コマンド（FtpSvStop）の指定方法については，「3.13.4 ファイル転送機能で使用するコマンド」を参照してください。

3.14 サービスプロセスを使用する場合の設定

この節では，サービスプロセスごとに動作を定義する方法およびサービスプロセス定義ファイルについて説明します。

サービスプロセスごとに動作を定義することで，ポート番号を指定して，サービスプロセスを複数起動できるようになります。サービスプロセス定義ファイルについては，「4.13　サービスプロセス定義ファイル」を参照してください。

3.15　複数の実行環境を構築する場合の設定

この節では，複数の実行環境から，同一の文書空間にアクセスする場合の実行環境の構築方法について説明します。DocumentBroker では，使用ユーザ数や単位時間あたりのトランザクション数の増加などのシステム負荷を軽減するために，一つのデータベースに複数の DocumentBroker サーバの実行環境を配置したシステム構成を構築できます。これによって，システム全体の負荷分散が実現できます。

(1) 新規に実行環境を構築する場合

複数の実行環境から同一文書空間にアクセスするために，新規に実行環境を構築する手順を次に示します。なお，次の手順では，接続する文書空間にアクセスする実行環境がすでに構築されていることが前提です。

1. **DocumentBroker の実行環境作成コマンド（EDMSetup -m create）を実行して，新規に DocumentBroker の実行環境を作成する。**
   なお，DocumentBroker の実行環境の作成については，「3.6　DocumentBroker の実行環境の作成」を参照してください。
2. **slocalreg.ini ファイル，DocumentSpace 構成定義ファイル，および実行環境ディレクトリ /etc/edms.ini を設定する。**
   このとき，同一文書空間として使用したい実行環境の DocumentSpace 構成定義ファイルの SerialId エントリと同じ値を，次のエントリに指定します。
   - ServiceObjectID エントリ（slocalreg.ini ファイル）
   - SerialId エントリ（DocumentSpace 構成定義ファイルの [Entry0001] セクション）
   - dmaProp_DocSpaceId エントリ（実行環境ディレクトリ /etc/edms.ini の [dmaClass_DocSpace] セクション）

   また，DocumentSpace 構成定義ファイルに定義されている次のエントリの値を変更してください。
   - ErrFlagOfObjectOperation エントリ
   - EnbFncFlagOfObjectOperation エントリ
     変更後の値が，メタ情報初期設定コマンド（EDMInitMeta）を実行した DocumentBroker 実行環境の DocumentSpace 構成定義ファイルで指定されている値になるようにします。

   DocumetSpace 構成定義ファイルについては，「4.2　DocumentSpace 構成定義ファイル（docspace.ini)」を参照してください。
3. **メタ情報ファイルの出力コマンド（EDMPrintMeta -F）を実行する。**
4. **DocumentBroker 実行環境の情報の登録コマンド（EDMRegEnvId -r）を実行する。**

これによって，新規に構築した実行環境から同一文書空間にアクセスできるようになります。

(2) 実行環境を変更する場合

別マシンへの実行環境の移動や，実行環境ディレクトリのパスの変更によって実行環境を変更する場合の手順を次に示します。

なお，変更する実行環境の実行環境識別子の値によって，手順が異なります。実行環境識別子は，DocumentBroker 実行環境の情報の登録コマンド（EDMRegEnvId -l）で確認してください。

1. **変更後の DocumentBroker の実行環境を作成する。**
2. **変更前の実行環境ディレクトリ /etc に格納されているファイルを，変更後の実行環境ディレクトリ /etc に移動する。**
3. **変更後の実行環境で，メタ情報ファイルの出力コマンド（EDMPrintMeta -F）を実行する。**

4. **実行環境識別子が** 1 **～** 254 **の場合，**DocumentBroker **実行環境の情報の登録コマンド（**EDMRegEnvId -u**）を実行する。**
   実行環境識別子が「0」の場合，この作業は必要ありません。
5. **変更前の実行環境で，実行環境の削除（**EDMSetup -m delete**）を実行する。**

（3） 実行環境を削除する場合

実行環境を削除する手順を次に示します。

1. DocumentBroker **実行環境の情報の登録コマンド（**EDMRegEnvId -d**）を実行する。**

ただし，削除しようとする実行環境で，すでに OIID が割り当てられているオブジェクトが作成されていた場合，実行環境は削除できません。

3.16　リファレンスファイル管理機能を使用する場合の設定

この節では，リファレンスファイル管理機能を使用するための設定について説明します。

3.16.1　コンテントを管理するための設定

ここでは，リファレンスファイル管理機能を使用して，ファイルシステム上の任意のディレクトリにコンテントを格納するための設定について説明します。

(1) コンテントを格納する領域の確保

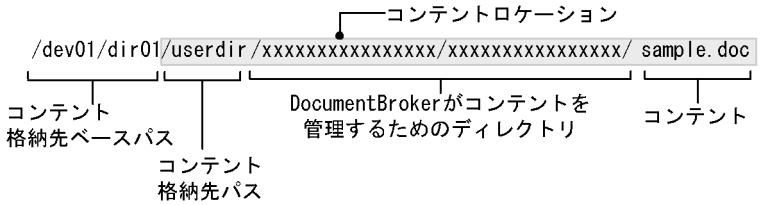
コンテントを格納するために，ファイルシステム上に領域を確保する必要があります。確保する領域のディスク容量の見積もり方法については「2.9　リファレンスファイル管理機能を使用する場合のファイルシステムのディスク容量の見積もり」を参照してください。見積もった値よりも大きいディスク容量を確保してください。

(2) コンテントを格納するディレクトリの作成

コンテントの格納先とするディレクトリを基点としたパスをコンテント格納先ベースパスといいます。コンテント格納先ベースパスに設定するディレクトリは，あらかじめファイルシステム上の任意の領域に作成しておく必要があります。

作成したディレクトリの下に，DocumentBroker によってコンテントが格納されます。コンテント格納先ベースパス下のコンテント格納先の例を次の図に示します。

図 3-13　コンテント格納先ベースパス下のコンテント格納先の例



この例は，コンテント格納先ベースパスにはディレクトリ「/dev01/dir01」を，コンテント格納先パスにはディレクトリ「userdir」を設定して，コンテント「sample.doc」が格納された場合を示しています。

コンテント格納先ベースパス

ファイルシステム上の任意の領域に，コンテントの格納先としてシステム管理者が作成するディレクトリです。このディレクトリは，あらかじめ作成しておく必要があります。作成したディレクトリをコンテント格納先ベースパスとして設定します。

文書空間で使用する文字コード種別が UTF-8 の場合，コンテント格納先ベースパスには，ASCII コードで表せるパスを指定してください。

コンテント格納先パス

コンテント格納先ベースパスからの相対パスで指定するディレクトリです。このディレクトリは，コンテント格納時に DocumentBroker によって作成されます。

文書空間で使用する文字コード種別が UTF-8 の場合，コンテント格納先パスには，ASCII コードで表せるパスを指定してください。

DocumentBroker がコンテントを管理するためのディレクトリ

DocumentBroker によって作成される，コンテントを管理するためのディレクトリです。このディレクトリの下にコンテントが格納されます。

コンテント
DocumentBroker によってコンテントが登録，管理されます。

コンテント格納先パス，DocumentBroker がコンテントを管理するためのディレクトリ，およびコンテントを合わせて，コンテントロケーションとしてデータベースで管理されます。

なお，コンテント格納先ベースパスの長さ，コンテント格納先パスの長さおよびファイル名の長さの合計が，次の値を超えないようにしてください。

コンテント格納先ベースパスの長さ（バイト）＋コンテント格納先パスの長さ（バイト）＋ファイル名の長さ（バイト）＜OSの最大ファイルパス長（バイト）－35（バイト）

(3) コンテントを格納するディレクトリおよびコンテントのアクセス権限の設定

コンテント格納先ベースパスに指定するディレクトリには，システム管理者およびシステム管理者グループに対してだけ読み取り権，書き込み権，実行権（770）のアクセス権限を設定してください。そのほかのグループまたはユーザに対しては，操作可能な権限を設定しないでください。

(4) システム構成に応じたコンテントの格納先の設定

次に示すシステム構成の場合，必要に応じてコンテントの格納先を設定してください。

● 複数の実行環境を構成する場合
DocumentBroker サーバの実行環境ごとに，コンテントの格納先ディレクトリを確保してください。このとき，コンテントの格納先とするファイルシステムは，DocumentBroker サーバが存在するマシンから接続可能なネットワーク上のマシンの共有ディスクを使用してください。ネットワーク上のマシンの共有ディスクでコンテントを管理するための設定については，「3.16.2　ネットワーク上のマシンの共有ディスクでコンテントを管理する場合の設定」を参照してください。

● 複数の異なる文書空間で，コンテントの格納先を同一のファイルシステムとする場合
各文書空間で，コンテント格納先ベースパスが重複しないようにしてください。複数の異なる文書空間で同一のコンテント格納先ベースパスを使用した場合，ファイル破壊の原因となります。

3.16.2　ネットワーク上のマシンの共有ディスクでコンテントを管理する場合の設定

ここでは，複数の実行環境を構成するシステムやクラスタリングシステムでリファレンスファイル管理機能を使用する場合に，ネットワーク上のマシンの共有ディスクにコンテントを格納するための設定について説明します。

NFS を使用するためには，DocumentBroker サーバを起動する前に次の設定を実施してください。

● NFS サーバの構成
コンテントの格納先となるサーバマシンで，NFS サーバを構成しておきます。

● NFS ファイルシステムのエクスポート
コンテントの格納先となるサーバマシンでコンテントの格納先ディレクトリを作成し，NFS ファイルシステムのエクスポートを行います。

● NFS クライアントの構成
DocumentBroker サーバが存在するマシンで，NFS クライアントの構成を定義します。

● NFS ファイルシステムのマウント

DocumentBroker サーバが存在するマシンで，NFS ファイルシステムをマウントします。

NFS の設定の詳細については，オペレーティングシステムのマニュアルを参照してください。

なお，コンテントの格納先となるサーバマシンと，DocumentBroker サーバが存在するマシンは同一のオペレーティングシステムを使用してください。また，パスワードファイルに設定するシステム管理者ユーザのユーザ ID，およびシステム管理者ユーザが所属するシステム管理者グループのグループ ID は，マシン間で同一の内容にしておく必要があります（同じパスワードファイルを使用することを推奨します）。

システム管理者ユーザのユーザ ID，およびシステム管理者ユーザが所属するシステム管理者グループのグループ ID が異なる場合，ファイルアクセスでのエラーの原因となります。

3.17 File Link 連携機能を使用する場合の設定

この節では，File Link 連携機能を使用する場合の設定について説明します。

File Link 連携機能を使用する場合，HiRDB のプラグインプログラムである HiRDB File Link を使用してファイルサーバを構築する必要があります。ファイルサーバの構築時，ファイルサーバ上でコンテントを管理する FAM の環境設定では，FAM コンフィグレーションファイルの SETUP セクションのコンテント更新方法（COPY_UPDATE エントリ）に，必ず「Y」を指定してください。ファイルサーバの構築については，マニュアル「HiRDB File Link」を参照してください。

また，プラグインプログラムの環境設定については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

なお，Linux の場合および TPBroker V5 を使用している場合，File Link 連携機能は使用できません。

3.18 XML 文書管理機能を使用する場合の設定

この節では，XML を利用して XML 文書管理機能を使用するための設定について説明します。なお，Linux の場合，XML 文書管理機能は使用できません。

XML 文書管理機能を利用する場合，次に示すプログラムをクライアント環境にインストールする必要があります。

● HiRDB Adapter for XML
クライアントの環境変数「XMLBRKDIR」に，HiRDB Adapter for XML のインストールディレクトリの絶対パスを設定する必要があります。プログラムのインストール方法については，マニュアル「HiRDB Adapter for XML ユーザーズガイド」を参照してください。

また，XML インデクスデータ作成機能を使用する場合，上記のプログラムに加えて次のプログラムをクライアント環境にインストールする必要があります。

● Preprocessing Library for Text Search

プログラムのインストール方法については，マニュアル「Preprocessing Library for Text Search Version 2」を参照してください。

3.18.1 定義ファイルの作成・生成の概要

XML 文書管理機能を利用するためには，HiRDB Adapter for XML に必要な定義ファイル，および Preprocessing Library for Text Search に必要な定義ファイルを作成して，実行時に指定する必要があります。

HiRDB Adapter for XML を動作させるための定義ファイルは，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）で生成します。生成した定義ファイルは DocumentBroker サーバに格納されます。なお，このコマンドに指定する入力ファイルは，ユーザが作成する必要があります。

DocumentBroker の XML 文書管理機能に必要な XML 定義ファイルについて説明します。

（1）XML 定義ファイルの関連

XML 文書管理機能で使用する定義ファイル（**XML 定義ファイル**と呼びます）の関連について説明します。特に，HiRDB Adapter for XML を動作させるためのファイルを **HAX 定義ファイル**と呼びます。

XML 定義ファイルには，次の表に示すファイルがあります。

表 3-12 XML 定義ファイルの一覧

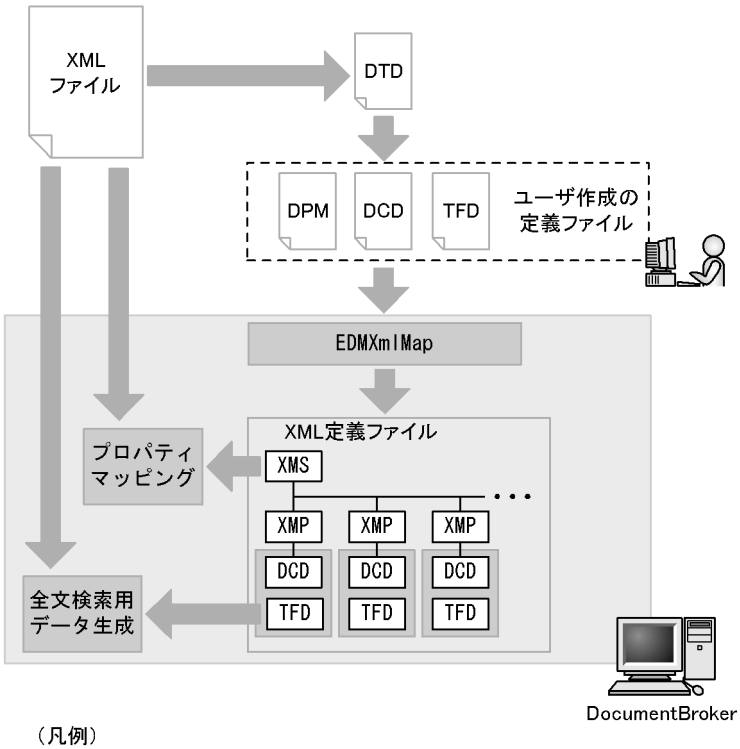
ファイル名	用途	作成方法	HiRDB Adapter for XML 使用時の総称
マッピングセット定義ファイル（XMS）	プロパティマッピングに使用するファイル	EDMXmlMap コマンドで作成	HAX 定義ファイル
マッピング定義ファイル（XMP）			
マッピング元 XML タグ定義ファイル（DCD）		ユーザが作成	
プロパティマッピング定義ファイル（DPM）			－

ファイル名	用途	作成方法	HiRDB Adapter for XML 使用時の総称
フィルタリング定義ファイル（TFD）	インデクスデータ作成に使用するファイル		―

（凡例）
―：総称なし。

これらの XML 定義ファイルの関連を次の図に示します。

図 3-14　XML 定義ファイルの関連



（凡例）
DPM：プロパティマッピング定義ファイル
DCD：マッピング元XMLタグ定義ファイル
TFD：フィルタリング定義ファイル
XMS：マッピングセット定義ファイル
XMP：マッピング定義ファイル
EDMXmlMap：XML定義ファイルの追加／更新／削除コマンド

この図について説明します。一つのマッピングセット定義ファイル（XMS）からは，複数のマッピング定義ファイル（XMP）を参照します。また，各マッピング定義ファイルは，一つのマッピング元 XML タグ定義ファイル（DCD）を参照します。

プロパティマッピング定義ファイルは，HAX 定義ファイルを生成するために必要なユーザ作成のファイルです。

フィルタリング定義ファイルは，Preprocessing Library for Text Search を動作させるためのファイルです。このファイルは，登録対象の XML 文書から全文検索用データを作成する際に利用するユーザ作成のファイルです。

(2) ユーザが作成する定義ファイル

ユーザが作成する必要のある定義ファイルを次に示します。

プロパティマッピング定義ファイル

XML 文書中のタグ名とその内容をマッピングするクラス名，プロパティ名の対応関係の定義（プロパティマッピング定義）を記述したファイルです。この定義ファイルを基に，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）がマッピング定義ファイルを生成します。
プロパティマッピング定義は，登録時にマッピング定義名を付けて登録します。プロパティマッピングの実行時には，このマッピング定義名を指定して使用するプロパティマッピング定義を選択します。
プロパティマッピング定義ファイルの作成については，「4.14　プロパティマッピング定義ファイル」を参照してください。

マッピング元 XML タグ定義ファイル

登録する XML 文書の構造を DCD と呼ばれる形式で記述したファイルです。この定義ファイルの文法は HiRDB Adapter for XML での規則に従ってください。詳細については，マニュアル「HiRDB Adapter for XML ユーザーズガイド」を参照してください。

フィルタリング定義ファイル

Preprocessing Library for Text Search のフィルタリング機能によって全文検索用データ（ESIS-B データ）から不要なタグを除去したり，不要なタグ間のテキストをタグごと除去したりする場合に必要です。
この定義ファイルの文法は，Preprocessing Library for Text Search での規則に従ってください。詳細については，マニュアル「Preprocessing Library for Text Search Version 2」を参照してください。

(3) コマンドで生成する定義ファイル

ユーザによって作成された定義ファイルを基に，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）によって生成する HAX 定義ファイルについて説明します。HAX 定義ファイルは，HiRDB Adapter for XML を動作させるための定義ファイルとして使用します。

HAX 定義ファイル

XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）の入力ファイルとして，プロパティマッピング定義ファイル，およびマッピング元 XML タグ定義ファイルを指定して出力されるファイルです。
出力される HAX 定義ファイルは，次の三つの定義ファイルから構成されます。

- マッピングセット定義ファイル（XMS）
- マッピング定義ファイル（XMP）
- マッピング元 XML タグ定義ファイル（DCD）

なお，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）の入力ファイルとしてフィルタリング定義ファイルを指定した場合は，DocumentBroker サーバへの登録用にフィルタリング定義ファイルのコピーを生成します。

HAX 定義ファイルおよびフィルタリング定義ファイルは，DocumentBroker サーバ上で管理します。

3.18.2　XML 定義ファイルの登録

XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）を実行すると，ユーザ作成の定義ファイルから XML 定義ファイルが生成されます。この XML 定義ファイルは，DocumentBroker サーバの XML 定義ファイル格納ディレクトリに格納・登録されます。XML 定義ファイル格納ディレクトリを次に

示します。

$DOCBROKERDIR/etc/xml_files

コマンド実行によって生成されるファイルを次に示します。

(1) マッピング定義ファイル（XMP）

HiRDB Adapter for XML が使用できる文法に変換したマッピング定義が記述されているファイルです。ユーザ作成のプロパティマッピング定義ファイル（DPM）とマッピング元 XML タグ定義ファイル（DCD）から生成します。

生成されるファイル名を次に示します。

'**文書空間識別子**'_'**マッピング定義名**'.xmp

(2) マッピング元 XML タグ定義ファイル（DCD）

XML 定義ファイル登録用にファイル名を変更した，ユーザ作成のマッピング元 XML タグ定義ファイルです。

生成されるファイル名を次に示します。

'**文書空間識別子**'_'**マッピング定義名**'.dcd

(3) フィルタリング定義ファイル（TFD）

XML 定義ファイル登録用にファイル名を変更した，ユーザ作成のフィルタリング定義ファイルです。

生成されるファイル名を次に示します。

'**文書空間識別子**'_'**マッピング定義名**'.tfd

(4) マッピングセット定義ファイル（XMS）

HiRDB Adapter for XML が使用する，登録されたマッピング定義の一覧を管理するファイルです。このファイルは，一つのファイルだけ存在します。

生成されるファイル名を次に示します。

'**文書空間識別子**'.xms

3.18.3 XML 文書管理機能の環境設定

XML 文書管理機能を利用するためには，各種の定義ファイルを作成し，実行時に指定する必要があります。ここでは，定義ファイルの管理および取得方法について説明します。なお，ここでは定義ファイルの管理および取得方法の説明に，C++ クラスライブラリのメソッドを使用します。

(1) DocumentBroker への XML 定義ファイルの登録

XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）を実行して，ユーザ作成の定義ファイルから生成するマッピング定義ファイルを，DocumentBroker サーバの実行環境の XML 定義ファイル格納ディレクトリに登録します。登録時には，マッピングセット定義ファイルに，登録したマッピング定義のエントリを追加します。

また，構造指定全文検索を実行する際に不要な構造などを除去するフィルタリング定義ファイルについても，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）に指定することによって HAX 定

義ファイルと関連づけて登録します。

(2) XML 文書管理機能を使用するための操作

XML 文書管理機能を使用するためには，DocumentBroker サーバに格納されている XML 定義ファイルを DocumentBroker クライアントの環境に転送する必要があります。転送では，ユーザが DocumentBroker サーバにある XML 定義ファイルを DocumentBroker クライアントの XML 定義ファイル格納ディレクトリにコピーします。

クライアント環境では次のファイルを同じディレクトリに格納します。

● HAX 定義ファイル

● フィルタリング定義ファイル

(3) XML 定義ファイルの DocumentBroker クライアント環境への転送

XML 文書管理機能を使用するためには，XML 定義ファイルを，DocumentBroker サーバ環境の XML 定義ファイル格納ディレクトリから，DocumentBroker クライアント環境の XML 定義ファイル格納ディレクトリに転送する必要があります。

XML 文書管理機能の前提プログラムで使用する定義ファイルを次に示します。これらのファイルは，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）によって生成された時点で，DocumentBroker サーバに格納されています。

HiRDB Adapter for XML **で使用する定義ファイル**

- マッピングセット定義ファイル（拡張子は，xms）
- マッピング定義ファイル（拡張子は，xmp）
- マッピング元 XML タグ定義ファイル（拡張子は，dcd）

Preprocessing Library for Text Search **で使用する定義ファイル**

- フィルタリング定義ファイル（拡張子は，tfd）

これらの定義ファイルは，DocumentBroker サーバの実行環境にある XML 定義ファイル格納ディレクトリに格納されています。格納ディレクトリを次に示します。

$DOCBROKERDIR/etc/xml_files

また，それぞれの定義ファイルの名称は次のようになります。

- マッピング定義ファイル（XMP）
  '**文書空間識別子**'_'**マッピング定義名**'.xmp
- マッピング元 XML タグ定義ファイル（DCD）
  '**文書空間識別子**'_'**マッピング定義名**'.dcd
- フィルタリング定義ファイル（TFD）
  '**文書空間識別子**'_'**マッピング定義名**'.tfd
- マッピングセット定義ファイル（XMS）
  '**文書空間識別子**'.xms

これらの定義ファイルを DocumentBroker クライアント環境下の XML 定義ファイル格納ディレクトリに転送します。転送先のディレクトリを次に示します。

AIX の DocumentBroker クライアントの環境を使用する場合
　/opt/HiEDMS/client/etc/xml_files

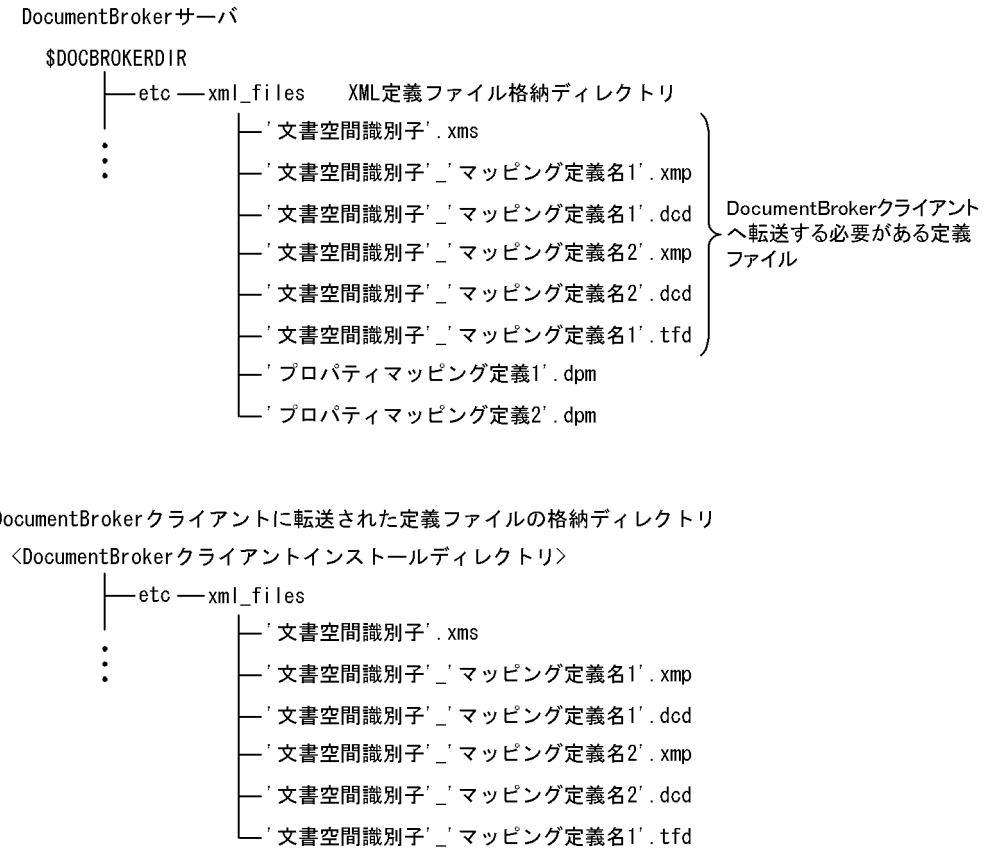
Windows，Windows XP，Windows Vista，および Windows 7 の DocumentBroker クライアントの環境を使用する場合

- DocumentBroker Development Kit を使用するとき
  <DocumentBroker **クライアントのインストールディレクトリ** >¥Devkit¥etc¥xml_files
- DocumentBroker Runtime を使用するとき
  <DocumentBroker **クライアントのインストールディレクトリ** >¥Runtime¥etc¥xml_files

XML 文書管理機能を使用するためには，これらのディレクトリに，DocumentBroker クライアントの接続先の DocumentBroker サーバに格納された定義ファイルをすべて転送します。

転送する必要がある定義ファイルを次の図に示します。

図 3-15　DocumentBroker サーバから DocumentBroker クライアントに転送する必要のある定義ファイルと格納ディレクトリ



この図に示した定義ファイルの格納状態で，文書空間識別子だけを指定した CdbrXmlTranslatorFactory::Initialize メソッドを実行することで，必要な定義ファイルを読み込み，XML 文書管理機能を実行できます。

また，フィルタリング定義ファイルについては，GetDmaInfoList メソッド，GetIndexData メソッド実行時に使用することで，不要なタグの除去などができます。

(4) 定義ファイルのカスタマイズ

DocumentBroker サーバの XML 定義ファイル格納ディレクトリには，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）によって生成されたすべての XML 定義ファイルが格納されています。し

かし，DocumentBroker クライアント環境では，これらの定義ファイルのすべてが必要ではない場合があります。「3.18.3(3) XML 定義ファイルの DocumentBroker クライアント環境への転送」で説明した手順でファイルを転送したあと，DocumentBroker クライアント環境に転送した定義ファイルのうちマッピングセット定義（XMS）ファイルの内容を編集することで，必要な定義ファイルだけを選択して使用できます。

このように，XMS ファイルから不要なマッピング定義（XMP）ファイルの定義を削除することで，CdbrXmlTranslatorFactory クラスの Initialize メソッドの実行時間を短縮できます。また，マッピング定義を保持するためのメモリ領域も削減できます。

また，編集した XMS ファイルを別のファイルとして保存できます。この場合は，CdbrXmlTranslatorFactory クラスの Initialize メソッドの実行時にファイル名を指定することで，カスタマイズした XMS ファイルを基にプロパティマッピング処理を実現できます。

以降，XML 定義ファイルのカスタマイズの手順を説明します。ただし，カスタマイズできるのは，XMS ファイルだけです。

XMS ファイルのサンプルを次に示します。マッピングセット定義の方法は，HiRDB Adapter for XML の規則に従ってください。詳細については，マニュアル「HiRDB Adapter for XML ユーザーズガイド」を参照してください。

```
<?xml version="1.0" encoding="Shift_JIS"?>

<mappingset>
<!--EDM name="Map1" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_Map1.xmp"
time="20000111184505" CH="U_CH1" DV="U_DV1" -->
<schema name="Map1" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_Map1.xmp"/>
<!--EDM name="Prop2" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_Prop2.xmp"
time="20000112120030" CH="ch2" DV="dv2" -->
<schema name="Prop2" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_Prop2.xmp"/>
<!--EDM name="SAMPLE2" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_SAMPLE2.xmp"
time="20000116084500" CH="UsrClass_CH" DV="UsrProp_DV" -->
<schema name="SAMPLE2" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_SAMPLE2.xmp"/>
</mappingset>
```

XMS ファイルの定義では，マッピング定義名とマッピング定義ファイルの対応を記述し，それぞれの定義にコメントが付加されています。このファイルを編集する場合，コメントを参照して編集対象の定義を特定します。

XMS ファイルの編集内容を次に示します。

1. 不要な schema 定義の削除
2. XMP ファイルのパス変更

「1. 不要な schema 定義の削除」について説明します。不要な schema 定義は，該当行を削除して，読み込まれないようにします。

XMS ファイルのサンプルの 2 番目の定義（下線の付いた部分）が不要な場合，この部分を削除します。削除した定義を次に示します。

```
<?xml version="1.0" encoding="Shift_JIS"?>

<mappingset>
<!--EDM name="Map1" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_Map1.xmp"
time="20000111184505" CH="U_CH1" DV="U_DV1" -->
<schema name="Map1" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_Map1.xmp"/>
```

```
<!--EDM name="SAMPLE2" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_SAMPLE2.xmp"
time="20000116084500" CH="UsrClass_CH" DV="UsrProp_DV" -->
<schema name="SAMPLE2" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_SAMPLE2.xmp"/>
</mappingset>
```

編集済みの XMS ファイルを編集元の XMS ファイルと同じディレクトリ（XML 定義ファイル格納ディレクトリ）に格納することで，編集済みの定義を利用できます。

編集済みの XMS ファイルを別ディレクトリに格納する場合は，参照される XMP ファイルおよびマッピング元 XML タグ定義（DCD）ファイルを編集済みの XMS ファイルと同じディレクトリに格納してください。これは，schema 定義で指定する XMP ファイルの指定が相対パスだからです。XMS ファイルだけを別ディレクトリに格納した場合は，XMP ファイルのパスの変更が必要です。

次に，「2. XMP ファイルのパス変更」について説明します。

「1. 不要な schema 定義の削除」の手順で作成した XMS ファイルを別ディレクトリに格納して，XML 定義格納ディレクトリに存在する XMP ファイルおよび DCD ファイルを使用する場合，XMS ファイルの schema 定義で指定されている XMP ファイルのパスを XML 定義ファイル格納ディレクトリに格納されているファイルを指すように修正する必要があります。

ファイルパスは，絶対パスまたは XMS ファイルからの相対パスを指定できます。ここでは絶対パスへの変更例を示します。相対パスへの変更も同じ手順で行ってください。

修正した定義を次に示します。

```
<?xml version="1.0" encoding="Shift_JIS"?>

<mappingset>
<!--EDM name="Map1" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_Map1.xmp"
time="20000111184505" CH="U_CH1" DV="U_DV1" -->
<schema name="Map1" map="/opt/HiEDMS/client/etc/xml_files/
673d2be0-d1fd-11d0-ab59-08002be29e1d_Map1.xmp"/>
<!--EDM name="SAMPLE2" map="673d2be0-d1fd-11d0-ab59-08002be29e1d_SAMPLE2.xmp"
time="20000116084500" CH="UsrClass_CH" DV="UsrProp_DV" -->
<schema name="SAMPLE2" map="/opt/HiEDMS/client/etc/xml_files/
673d2be0-d1fd-11d0-ab59-08002be29e1d_SAMPLE2.xmp"/>
</mappingset>
```

XMP ファイルのパスを絶対パスに変更することで，XML 定義ファイル格納ディレクトリに格納されたファイルを参照する定義に変更できます。

この XMS ファイルを任意のディレクトリに格納して，CdbrXmlTranslatorFactory::Initialize メソッドで指定することで，カスタマイズされた XMS ファイルによるマッピング処理ができます。

3.18.4 XML 定義ファイルを作成する場合の注意事項

XML 定義ファイルを作成する場合の注意事項を説明します。なお，ここでは説明に，C++ クラスライブラリのメソッドを使用します。

（1）Boolean 型のプロパティにマッピングする場合の値

Boolean 型のプロパティにマッピングする場合，マッピング元のタグ（属性）の値が「0」または「1」以外の値では，CdbrXmlTranslator::GetDmaInfoList メソッドでエラーになります。

Boolean 型のプロパティにマッピングするタグ（属性）の値が「0」または「1」以外の値にならないようにしてください。

(2) マッピング元 XML タグ定義ファイル（DCD）の記述エラーの検知

マッピング元 XML タグ定義ファイル（DCD）に記述エラーがあると，
CdbrXmlTranslatorFactory::Initialize メソッドでエラーになります。

なお，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）は，この記述エラーを検知しません。

(3) フィルタリング定義ファイル（TFD）の記述エラーの検知

フィルタリング定義ファイル（TFD）に記述エラーがあると，
CdbrXmlTranslatorFactory::GetDmaInfoList メソッド，または
CdbrXmlTranslatorFactory::GetIndexData メソッドでエラーになります。

なお，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）は，この記述エラーを検知しません。

4 環境設定で必要なファイル

この章では，DocumentBroker の環境設定で必要なファイルについて説明します。

4.1 **ファイルの種類**

4.2 DocumentSpace **構成定義ファイル**（docspace.ini）

4.3 **セキュリティ定義ファイル**（docaccess.ini）

4.4 **ユーザ権限定義ファイル**

4.5 RD **エリア構成定義ファイル**

4.6 **メタ情報ファイル**

4.7 **定義情報ファイル**

4.8 RD **エリア定義情報ファイル**

4.9 **インデクス情報ファイル**

4.10 **クラス定義情報ファイル**

4.11 **ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイル**

4.12 **ファイル転送サービス環境定義ファイル**（ftpsv.ini）

4.13 **サービスプロセス定義ファイル**

4.14 **プロパティマッピング定義ファイル**

4.15 **文書クラス定義ファイル**

4.16 **文書空間情報ファイル**

4.17 **見積もり基礎情報ファイル**

4.18 **クライアントアプリケーション動作定義ファイル**（application.ini）

4.1 ファイルの種類

DocumentBroker の環境設定で必要なファイルの一覧を次に示します。

表 4-1 環境設定で必要なファイル

名称	内容	作成時期	参照先
DocumentSpace 構成定義ファイル（docspace.ini）※1	DocumentBroker の文書空間の構成を定義する。	DocumentBroker の文書空間を定義するとき	4.2
セキュリティ定義ファイル（docaccess.ini）※1	文書空間のセキュリティについて定義する。	アクセス制御を設定するとき	4.3
ユーザ権限定義ファイル※1	文書空間のすべてのオブジェクトに対しての権限をユーザ，またはグループ単位で定義する。		4.4
RD エリア構成定義ファイル	HiRDB に確保する RD エリアの構成を定義する。	HiRDB の環境を設定するとき	4.5
メタ情報ファイル※1	DocumentBroker が利用する DMA のクラス，プロパティ，検索オペレータなどの詳細情報が定義されている。	実行環境を作成したときに格納されるファイルを使用	4.6
定義情報ファイル	サブクラスおよびプロパティを追加するときに，追加するオブジェクトの定義情報を定義する。	DocumentBroker でデータベースシステムを使用するための設定を始める前	4.7
RD エリア定義情報ファイル	DocumentBroker が使用するユーザ表およびインデクスを，どの RD エリアに格納するか定義する。		4.8
インデクス情報ファイル	追加するプロパティのインデクス情報を定義する。		4.9
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，そのサブクラスのクラス，およびプロパティから，GUID，データ型，基本単位などの情報が出力される。	DocumentBroker でデータベースシステムを使用するための設定をするとき	4.10
ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイル	ユーザ定義のクラス，プロパティのそれぞれに対応するユーザ定義識別子，およびユーザ定義識別子を設定した変数の宣言が出力される。		4.11
ファイル転送サービス環境定義ファイル（ftpsv.ini）	ファイル転送機能で使用する環境を定義する。	ファイル転送機能を使用するための設定をするとき	4.12
サービスプロセス定義ファイル※1	サービスプロセスごとに動作を定義する。	サービスプロセスごとに動作を定義するとき	4.13
プロパティマッピング定義ファイル	XML 文書中のタグ名とその内容をマッピングするクラス名，プロパティ名の対応関係を定義する。	XML 文書管理を設定するとき	4.14
文書クラス定義ファイル※2	サブクラスおよびプロパティを追加するときに，追加するオブジェクトの定義を文書クラスとして定義する。	システム導入支援機能を使用して，DocumentBroker でデータベースシステムを使用するための設定をするとき	4.15
文書空間情報ファイル※2	DocumentBroker の文書空間の定義 / 構築で使用する情報を定義する。		4.16

名称	内容	作成時期	参照先
見積もり基礎情報ファイル	文書空間を構築するためのデータベース容量を見積もるための基礎情報が出力される。		4.17
クライアントアプリケーション動作定義ファイル（application.ini）※3	Linux の場合に，クライアントアプリケーションの動作環境を定義する。		4.18

注※ 1

これらのファイルは，次の場所に提供されています。

実行環境ディレクトリ /etc/

注※ 2

これらのファイルは，次の場所にサンプルファイルが提供されています。

/opt/HiEDMS/sample/

注※ 3

これらのファイルは，次の場所にサンプルファイルが提供されています。

DocumentBroker Development Kit をインストールした場合

インストールディレクトリ・DevKit・sample

DocumentBroker Runtime をインストールした場合

インストールディレクトリ・Runtime・sample

4.2 DocumentSpace 構成定義ファイル（docspace.ini）

この節では，DocumentSpace 構成定義ファイルについて説明します。

DocumentSpace 構成定義ファイルは，" 実行環境ディレクトリ /etc/docspace.ini" に提供されています。

4.2.1 DocumentSpace 構成定義ファイルの概要

DocumentSpace 構成定義ファイルは，DocumentBroker の文書空間の構成を定義するファイルです。" 実行環境ディレクトリ /etc" に格納されている「docspace.ini」をシステム管理者がテキストエディタなどを使用して，編集してください。

以降，DocumentSpace 構成定義ファイルで指定するエントリを示します。各エントリの詳細については，「4.2.2 DocumentSpace 構成定義ファイルの記述形式」を参照してください。

（1）[DocSpace] セクション

文書空間の共通定義を指定します。[DocSpace] セクションを構成する各エントリを次の表に示します。

表 4-2 [DocSpace] セクションのエントリー覧

種類	設定項目	エントリ名	指定内容	省略の可否
文書空間の共通定義	文書空間の数	Count	DocumentBroker が提供する文書空間の数	×
	ORB および BOA のオプション	DocSpaceOrbBoaOption	サービスプロセス監視プロセスに対する ORB および BOA オプション	○
	VisiBroker プロパティ	DocSpaceVBProperty	サービスプロセス監視プロセスに対する VisiBroker プロパティ	○
	エラーログ	ErrLogFileCount	エラーログを取得するファイル数	○
		ErrLogFileSize	エラーログファイルのサイズ	○
	デッドロックおよびタイムアウト監視コマンド	DbLockWatcher	デッドロックおよびタイムアウト監視コマンド（EDMLckWatcher）の実行方法	○

（凡例）
○：指定を省略できます。
×：指定を省略できません。

（2）[Entry0001] セクション

文書空間の詳細を定義します。[Entry0001] セクションを構成する各エントリを次の表に示します。

表 4-3 [Entry0001] セクションのエントリー覧

種類	設定項目	エントリ名	指定内容	省略の可否
サービス	文書空間	SerialId	文書空間の識別子	×
	プロセス設定	Process	サービスプロセス数	×

種類	設定項目	エントリ名	指定内容	省略の可否
		SessionMax	DocumentBroker に同時に接続できるクライアントの最大数	○
		SessionTimeOut	セッションアイドル時間の最大値	○
		ProcessOrbBoaOption	全サービスプロセス共通の ORB オプションおよび BOA オプション	○
		ProcessVBProperty	全サービスプロセス共通の VisiBroker プロパティ	○
ユーザ認証	ユーザ認証	UserAuthentication	ユーザ認証方式	○
	LDAP 設定（LDAP または LDAPEX 指定時）	LdapClientLib	LDAP クライアントライブラリの種別	○
		LdapHost	ホスト名または IP アドレス	○
		LdapPort	ポート番号	○
	UOC ライブラリの設定（UOC 指定時）	UOCLibrary	UOC ライブラリのフルパス	△※1
ユーザ情報の検索（LDAP 指定時）	検索条件	LdapPrefixDn	最初の相対識別名に DocumentBroker が付与する DN のパス	○
		LdapUserId	ユーザクラスの属性	△
	検索オプション	LdapUserTimeout	ユーザ検索時の最大待ち時間	○
		LdapUserCase	ユーザ識別子の文字種の制限	○
ユーザ情報の検索（LDAPEX 指定時）	検索条件	LdapUserRoot	ノードの DN	○
		LdapUserScope	ユーザ検索時の検索範囲	○
		LdapUserClass	ユーザとして定義するクラスの名称	△
		LdapUserId	ユーザクラスの属性	△
		LdapUserFilterLeft	検索フィルタの「(」（左括弧）および検索フィルタの内容	○
		LdapUserFilterRight	検索フィルタの「)」（右括弧）および検索フィルタの内容	○
	検索オプション	LdapUserTimeout	ユーザ検索時の最大待ち時間	○
		LdapUserCase	ユーザ識別子の文字種の制限	○
グループ情報の検索（LDAP または LDAPEX 指定時）	ディレクトリエントリからの検索	LdapGroup	グループ識別子のオブジェクトからの検索方法	○
		LdapGroupRoot	ノードの DN	○
		LdapGroupScope	グループ検索時の検索範囲	○
		LdapGroupClass	グループ識別子にするオブジェクトのクラス	△

種類	設定項目	エントリ名	指定内容	省略の可否
		LdapGroupId	グループ識別子として利用する属性	△
		LdapGroupFilterLeft	検索フィルタの「(」（左括弧）および検索フィルタの内容	○
		LdapGroupFilterRight	検索フィルタの「)」（右括弧）および検索フィルタの内容	○
		LdapGroupMember	グループに所属しているユーザを記述する属性	△
	ユーザの属性値からの検索	LdapGroupFromUserAttr	ユーザの属性値からのディレクトリのノード情報の検索方法	○
		LdapGroupIdFromUserAttr	グループ情報が格納されているユーザの属性	○
		LdapGroupIsDnFromUserAttr	ユーザの属性に格納されているグループ情報	○
		LdapGroupIdAttrFromUserAttr	グループ識別子として利用する属性	○
	ユーザの DN からの検索	LdapGroupFromUserDn	グループ識別子の検索方法	○
		LdapGroupIdFromUserDn	グループ識別子として使用する属性名	○
	検索オプション	LdapGroupTimeout	グループ検索時の最大待ち時間	○
		LdapGroupCase	グループ識別子の文字種の制限	○
ディレクトリサービスに対するバインド方法（LDAP または LDAPEX 指定時）	ディレクトリサービスに対するバインド方法	LdapBindUserDN	ディレクトリサービスへのサーババインドに使用する DN	○
		LdapBindPassword	ディレクトリサービスへのサーババインドに使用するパスワード	○
データベース	接続するデータベースの設定	DbType	データベースの種別	×
HiRDB	接続設定	PdHost	サーバのホスト名	×
		PdNamePort	サーバのポート番号	×
		PdUser	データベースにアクセスするための DocumentBroker 用のユーザ名とパスワード	×
	コネクションプールの設定	DBConnectionPoolCount	サービスプロセス当たりの DB コネクションプール数	○
		DBConnectionPoolDynamic	サービスプロセス当たりの一時コネクションの最大数	○
		DBConnectionPoolTiming	プールする DB コネクション作成のタイミング	○
		DBConnectionPoolOver	プールしている DB コネクションがすべて使用中，かつ一時コネクションが最大数に達している場合のアクション	○
		DBConnectionScope	DB コネクション割り当て期間	○

種類	設定項目	エントリ名	指定内容	省略の可否
		DBConnectionPoolWaitTimeOut	DB コネクション割り当て待ち時間	○
	検索時の設定	BatchSizeHint	メモリにバッファリングする検索結果行の行数	○
	HiRDB Text Search Plug-in の設定	PdTSPluginOwner	HiRDB Text Search Plug-in を登録したユーザの認可識別子	△※2
アクセスログ	アクセスログの設定	AcLogUse	アクセスログの取得の有無	○
		AcLogLevel	アクセスログに出力するアクセスレベル	○
		AcLogFileCount	AcLogFileSize エントリに指定した上限値を超えた場合，切り替えるファイル数	○
		AcLogFileSize	アクセスログファイルのサイズ	○
メタ情報	共有メモリの設定	XdkShmemManage	メタ情報管理用の共用メモリの確保，解放の方法	○
		XdkShmemSize	メタ情報管理用に確保する共用メモリセグメントサイズ	○
ファイル分割転送機能	データ転送サイズの設定	FtpBufferSize	データ転送サイズ	○
複数の実行環境	サーバの選択	SelectServerInMultiServer	接続する DocumentBroker サーバ	○
プロセスリフレッシュ機能	メモリサイズ	SRefreshLimit	サービスプロセスのメモリサイズ	○
	猶予時間の設定	SRefreshGraceTime	リフレッシュ時のプロセス終了までの猶予時間	○
	プロセス終了契機	RefreshTiming	プロセス終了契機	○
オブジェクト操作	VariableArray 型のプロパティでの要素のチェック	VArrayElementCheck	プロパティの要素に指定した値のチェックの有無	○
	オブジェクト操作時のエラーチェックのフラグ	ErrChkFlagOfObjectOperation	オブジェクト操作時に実行するエラーチェック	○
	オブジェクト操作時の動作のフラグ	EnbFncFlagOfObjectOperation	オブジェクト操作時の動作	○
BLOB データ	BLOB データの設定	BlobSubstrMode	マルチファイル管理機能を使用する場合の BLOB データの取得，格納方法	○
	データサイズの設定	BlobSubstrThreshold	マルチファイル管理機能を使用する場合に，1 回のデータベースへのアクセスで取得，格納するデータサイズ	○
	BLOB データの取得先	BlobGettingMethod	BLOB データの取得先	○
DocumentBroker Life Cycle Suite	WorkCoordinator 連携の設定	LifeCycleSuiteConnection	WorkCoordinator との連携	○

（凡例）

○：指定を省略できます。
△：使用環境の条件によって，指定の省略の可否が変わります。
×：指定を省略できません。

注※ 1
UserAuthentication エントリで UOC を指定した場合，省略できません。

注※ 2
HiRDB Text Search Plug-in の登録時に，pdplgrgst コマンドの -u オプション（プラグイン登録スキーマ指定）を使用した場合，省略できません。

4.2.2 DocumentSpace 構成定義ファイルの記述形式

ここでは，DocumentSpace 構成定義ファイルの記述形式について説明します。

DocumentSpace 構成定義ファイルは，次に示す二つのセクションと，各セクションに指定するエントリによって構成されます。

- [DocSpace] セクション
- [Entry0001] セクション

セクションとエントリの記述規則を次に示します。

- セクション名は，[　]（角括弧）で囲んで指定します。一つのセクションは，セクション名を指定してから，次のセクション名を指定するまで，またはファイルの終端までの範囲です。
- エントリは，「エントリ名＝指定値」の形式で指定します。
- 「;」（セミコロン）で始まる行はコメント行として扱われます。
- 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 コードで記述します。

記述形式を次に示します。

[< セクション名 >]
< エントリ名 >=< 値 >

以降，DocumentSpace 構成定義ファイルを構成する各セクションと，セクションごとに指定するエントリについて説明します。

（1） [DocSpace] セクション

文書空間の共通定義を指定します。[DocSpace] セクションを構成する各エントリは次のとおりです。

Count エントリ

DocumentBroker が提供する文書空間の数を指定します。「1」を指定してください。なお，このエントリの指定は省略できません。

DocSpaceOrbBoaOption エントリ

文書空間のサービスプロセスの状態を監視するサービスプロセス監視プロセスに対して ORB および BOA のオプションを指定します。指定を省略した場合，「-OAthreadMax 64 -OAlocalipc 0」が仮定されます。なお，指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。
なお，このエントリは，TPBroker V3 環境の場合に指定するエントリです。TPBroker V5 と連携して動作する環境では指定できません。TPBroker V5 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V5 環境では，VisiBroker プロパティとして DocSpaceVBProperty エ

ントリに指定してください。

DocSpaceVBProperty エントリ

文書空間のサービスプロセスの状態を監視するサービスプロセス監視プロセスに対して VisiBroker プロパティを指定します。指定を省略した場合，
「-Dvbroker.se.iiop_tp.scm.iiop_tp.manager.type=Socket
-Dvbroker.se.iiop_tp.scm.iiop_tp.dispatcher.threadMax=64」が仮定されます。なお，指定できるプロパティについては，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。
なお，このエントリは，TPBroker V5 環境の場合に指定するエントリです。TPBroker V3 と連携して動作する環境では指定できません。TPBroker V3 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V3 環境では，ORB ／ BOA オプションとして DocSpaceOrbBoaOption エントリに指定してください。

ErrLogFileCount エントリ

エラーログを取得するファイル数を 2 ～ 16 で指定します。出力ファイル名は EDMError_%d.log で，「%d」は出力ファイル通番を示します。出力ファイル通番とは，1 ～ ErrLogFileCount エントリに指定した数（出力ファイル数）です。
エラーログは，出力ファイル通番が「1」のファイルから順番に出力されます。あるファイルにエラーログを出力する場合に，ファイルサイズが ErrLogFileSize エントリに指定した出力ファイルサイズよりも大きくなるとき，出力ファイル通番が一つ大きいファイルに出力します。例えば，出力ファイル通番が「1」であるファイルにエラーログを出力する場合に，このファイルのサイズの最大量を超えてしまうときは，ファイル出力通番が「2」のファイルにログを出力します。
ErrLogFileCount エントリに指定した最大のファイル通番のファイルにエラーログを出力しようとして，そのファイルのファイルサイズが ErrLogFileSize エントリに指定した出力ファイルサイズよりも大きくなってしまう場合，出力ファイル通番が「1」のファイルを初期化してログの出力を継続します。
指定を省略した場合および範囲外の値を指定した場合，「2」が仮定されます。

ErrLogFileSize エントリ

エラーログファイルのサイズを 4,096 ～ 2,147,483,647（バイト）で指定します。エラーログの出力が指定されたサイズを超える場合，次の通番のファイルへ出力を切り替えます。また，カレントファイル出力中に異常（入出力エラーなど）が発生した場合も，次のファイルへ出力を切り替えます。ただし，切り替えは 1 回だけ実行します。
指定を省略した場合および範囲外の値を指定した場合，1,048,576（1 メガバイト）が仮定されます。
エラーログとして得られる情報の詳細については，「6.6　エラーログに関する運用」を参照してください。

DbLockWatcher エントリ

DocumentBroker サーバの起動時（EDMStart コマンドの実行時）に，デッドロックおよびタイムアウト監視コマンド（EDMLckWatcher）を自動で実行するかどうかを指定します。なお，デッドロックおよびタイムアウト監視コマンド（EDMLckWatcher）を手動で実行する場合，このエントリの指定は不要です。

- None
  DocumentBroker サーバの起動時に実行しません。
- Accept
  DocumentBroker サーバの起動時に自動で実行します。

指定を省略した場合，「None」が仮定されます。また，上記以外の不正な値を指定した場合にはエ

ラーとなり，DocumentBroker サーバの起動に失敗します。
デッドロックおよびタイムアウト監視コマンド（EDMLckWatcher）の詳細は，「7.3　コマンドの文法」の「EDMLckWatcher（データベースのデッドロックおよびタイムアウトの監視）」を参照してください。

（2）[Entry0001] セクション

文書空間の詳細を定義します。[Entry0001] セクションを構成する各エントリは次のとおりです。

SerialId エントリ

文書空間の識別子として，GUID を指定します。" 実行環境ディレクトリ /etc/slocalreg.ini" の ServiceObjectID エントリの値（文書空間の GUID）を指定してください。なお，このエントリの指定は省略できません。
また，" 実行環境ディレクトリ /etc/edms.ini" の [dmaClass_DocSpace] セクションの [dmaProp_DocSpaceId] エントリの値を同じ値に変更したあと，メタ情報を初期設定する必要があります。

Process エントリ

文書空間が提供するサービスプロセスの数を指定します。1 ～ 20 の間で指定してください。なお，このエントリの指定は省略できません。

SessionMax エントリ

DocumentBroker に同時に接続できるクライアントの最大数を指定します。
1 ～ 1,024 の間で指定してください。指定を省略した場合，「64」が仮定されます。

SessionTimeOut エントリ

セッションアイドル時間の最大値を分単位で指定します。0 ～ 120（分）の間で指定してください。0 を指定した場合，セッション監視機能を使用しません。指定を省略した場合，「0（分）」が仮定されます。

ProcessOrbBoaOption エントリ

文書空間のサービスを供給する全プロセスに共通する ORB オプションおよび BOA オプションを指定します。指定を省略した場合，「-OAthreadMax 64 -OAlocalipc 0」が仮定されます。指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。
なお，このエントリは，TPBroker V3 環境の場合に指定するエントリです。TPBroker V5 と連携して動作する環境では指定できません。TPBroker V5 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V5 環境では，VisiBroker プロパティとして ProcessVBProperty エントリに指定してください。

ProcessVBProperty エントリ

文書空間のサービスを供給する全プロセスに共通する VisiBroker プロパティを指定します。省略を省略した場合，「-Dvbroker.se.iiop_tp.scm.iiop_tp.manager.type=Socket
-Dvbroker.se.iiop_tp.scm.iiop_tp.dispatcher.threadMax=64」が仮定されます。指定できるプロパティの詳細は，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。
なお，このエントリは，TPBroker V5 環境の場合に指定するエントリです。TPBroker V3 と連携して動作する環境では指定できません。TPBroker V3 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V3 環境では，ORB ／ BOA オプションとして ProcessOrbBoaOption エントリに指定してください。

UserAuthentication エントリ

文書空間で使用するユーザ認証方式を指定します。

- BASIC
  オペレーティングシステムの標準のパスワードファイル /etc/passwd/ を使用します。
- LDAP
  LDAP 対応のディレクトリサービスと連携してユーザ認証を実行します。この値を指定した場合は，LdapPrefixDn エントリの値として指定した相対識別名，LdapUserId に指定した属性およびユーザが入力するログイン名からユーザ認証に使用する DN を生成します。ただし，LdapPrefixDn エントリの指定が省略されている場合は，ユーザが入力したログイン名がユーザ認証に使用されます。
  また，この値を指定する場合は LdapPrefixDn エントリで指定した相対識別名の直下に，ユーザを一意に識別する情報がフラットな状態で構成されている必要があります。
- LDAPEX
  LDAP 対応のディレクトリサービスと連携してユーザ認証を実行します。使用するログイン名は，LdapUserRoot エントリ，LdapUserClass エントリ，LdapUserId エントリ，LdapUserFilterLeft エントリ，LdapUserFilterRight エントリなどの設定に依存します。
  また，ユーザを識別する情報は一意である必要がありますが，DIT の構成を意識する必要はありません。
- UOC
  ユーザ管理システムへのアクセスルーチンとして，ユーザが作成したアクセスルーチンを使用する場合に指定します。

指定を省略した場合，LDAP が仮定されます。

LdapClientLib エントリ

UserAuthentication エントリで，LDAP または LDAPEX を指定した場合に有効になります。
DocumentBroker の LDAP 認証で必要になる LDAP クライアントライブラリの種別を指定します。

- SecureWay
  AIX の SecureWay Directory，IBM Directory Server，または IBM Tivoli Directory Server の LDAP クライアントライブラリを使用する場合に指定します。なお，この値は，AIX の場合だけ指定できます。
- OpenLDAP
  Linux の OpenLDAP の LDAP クライアントライブラリを使用する場合に指定します。なお、この値は，Linux の場合だけ指定できます。

指定を省略した場合，次の値が仮定されます。

AIX の場合

SecureWay

Linux の場合

OpenLDAP

上記以外の不正な値を指定した場合はエラーとなり，DocumentBroker の起動は失敗します。この時，KMBR03005-E のメッセージが出力されます。また，上記の値を指定した場合でも DocumentBroker が動作するオペレーティングシステムによっては，LDAP クライアントライブラリがサポートされていないことがあります。この場合についてもエラーとなり，DocumentBroker の起動は失敗します。この時，KMBR03332-E のメッセージが出力されます。なお，メッセージの詳細については，マニュアル「DocumentBroker Version 3　メッセージ」を参照してください。次にオペレーティングシステムごとの LDAP クライアントライブラリの対応関係を次の表に示します。

表 4-4 オペレーティングシステムごとの LDAP クライアントライブラリの対応関係

オペレーティングシステム	エントリの指定値に対する LDAP クライアントライブラリ	
	SecureWay	OpenLDAP
AIX	libldap.a （SecureWayDirectory Version3.2.1，SecureWayDirectory Version3.2.2，または IBMDirectory Server Version4.1，同梱）	―
	libibmldap.a （IBM Tivoli Directory Server Version 5.2 同梱）	
Linux	―	libldap_r.so(RHEL 6.1 以降同梱の OpenLDAP)

（凡例）

―：対応していません。

LdapHost エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。LDAP 対応のディレクトリサービスのホスト名または IP アドレスを，255 バイト以内で指定します。指定を省略した場合，自ホストが仮定されます。

LdapPort エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。LDAP 対応のディレクトリサービスのポート番号を指定します。指定を省略した場合，標準のポート番号（389）が仮定されます。

UOCLibrary エントリ

UserAuthentication エントリで UOC を指定した場合には，必ず UOC ライブラリへの絶対パスを指定します。

パスは印刷可能な ASCII コードで指定してください。

LdapPrefixDn エントリ

UserAuthentication エントリで LDAP を指定した場合に有効になります。ユーザ認証に使用する DN を生成するために，最初の相対識別名に DocumentBroker が付与する DN のパスを，128 バイト以内で指定します。指定を省略した場合，ログイン名として指定した文字列を，そのまま認証に使用します。すなわち，UserAuthentication エントリで LDAP を指定している場合で，このエントリの指定を省略したときはログイン名として DN をすべて指定することになります。

LdapUserId エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。ユーザ識別子として利用するユーザクラスの属性を，64 バイト以内で指定します。ただし，指定する属性は，ユーザの一意性を保証している必要があります。指定を省略した場合，「uid」が仮定されます。

なお，UserAuthentication エントリで LDAPEX を指定した場合で，LDAP 対応のディレクトリサービスとして Active Directory を使用するときは，「sAMAccountName」または「userPrincipalName」を指定してください。

ただし，「userPrincipalName」を指定した場合，ログイン名として「ユーザ名 @ ドメイン名」の形式で指定する必要があります。

LdapUserTimeout エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。ディレク

トリエントリからユーザを検索するときの最大待ち時間を指定します。0 ～ 180（秒）の範囲で指定します。0 を指定した場合，待ち時間が無制限になります。指定を省略した場合，「60（秒）」が仮定されます。

LdapUserCase エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。
DocumentBroker で扱うユーザ識別子の文字種についての制限を指定します。

- Upper
  ユーザ識別子の文字列を，すべて大文字に変換する場合に指定します。LDAP 対応のディレクトリサービスから取得したユーザ識別子を，DocumentBroker 内で大文字に変換します。
- Lower
  ユーザ識別子の文字列を，すべて小文字に変換する場合に指定します。LDAP 対応のディレクトリサービスから取得したユーザ識別子を，DocumentBroker 内で小文字に変換します。
- Default
  ユーザ識別子の文字列を変換しません。

指定を省略した場合，「Default」が仮定されます。

LdapUserRoot エントリ

UserAuthentication エントリで LDAPEX を指定した場合に有効になります。DIT 上でユーザを検索するベースとなるノードの DN を，128 バイト以内で指定します。指定を省略した場合，「c=JP」が仮定されます。

LdapUserScope エントリ

UserAuthentication エントリで LDAPEX を指定した場合に有効になります。ディレクトリエントリからユーザを検索するときの検索範囲を指定します。

- Onelevel
  検索開始点の一つ下のレベルにあるすべてのエントリを検索します。
- Subtree
  検索開始点と，その下のすべてのレベルにあるすべてのエントリを検索します。

指定を省略した場合，「Subtree」が仮定されます。

LdapUserClass エントリ

UserAuthentication エントリで LDAPEX を指定した場合に有効になります。ユーザとして定義しているクラスの名称を，64 バイト以内で指定します。指定を省略した場合，「inetOrgPerson」が仮定されます。
なお，LDAP 対応のディレクトリサービスとして Active Directory を使用する場合，「user」または user のサブクラスを指定してください。

LdapUserFilterLeft エントリ

UserAuthentication エントリで LDAPEX を指定した場合に有効になります。ユーザを検索する場合に指定する検索フィルタの「(」（左括弧）および追加指定する検索フィルタの内容を，128 バイト以内で指定します。指定を省略した場合，「(」が仮定されます。

LdapUserFilterRight エントリ

UserAuthentication エントリで LDAPEX を指定した場合に有効になります。ユーザを検索する場合に指定する検索フィルタの「)」（右括弧）および追加指定する検索フィルタの内容を，128 バイト以内で指定します。指定を省略した場合，「)」が仮定されます。

LdapGroup エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループ識別子を，メンバ（ユーザ）をリストとしたオブジェクトから検索することを指定します。

- Yes
  メンバ（ユーザ）をリストとしたオブジェクトから検索します。
- No
  メンバ（ユーザ）をリストとしたオブジェクトから検索しません。

指定を省略した場合，「Yes」が仮定されます。

LdapGroupRoot エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。DIT 上でグループを検索する基となるノードの DN を，128 バイト以内で指定します。指定を省略した場合，「c=JP」が仮定されます。

LdapGroupScope エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。ディレクトリエントリでグループを検索する場合の検索範囲を指定します。

- Onelevel
  検索開始点の一つ下のレベルにあるすべてのエントリを検索します。
- Subtree
  検索開始点とその下のすべてのレベルにあるすべてのエントリを検索します。

指定を省略した場合，「Subtree」が仮定されます。

LdapGroupClass エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループ識別子としてメンバ（ユーザ）をリストとしたオブジェクトのクラスを，64 バイト以内で指定します。指定を省略した場合，次の値が仮定されます。

- hdsgroupOfUniqueNames

なお，LDAP 対応のディレクトリサービスとして Active Directory を使用する場合，「group」または group のサブクラスを指定してください。

LdapGroupId エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループ識別子として利用する属性を，64 バイト以内で指定します。指定を省略した場合，次の値が仮定されます。

- groupOfUniqueNamesId

なお，LDAP 対応のディレクトリサービスとして Active Directory を使用する場合，「sAMAccountName」を指定してください。

LdapGroupFilterLeft エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループを検索するときに指定する検索フィルタの「(」（左括弧）および追加指定する検索フィルタの内容を，128 バイト以内で指定します。指定を省略した場合，「(」が仮定されます。

LdapGroupFilterRight エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループを検索する場合に LDAP に指定する検索フィルタの「)」（右括弧）および追加指定する検索フィルタの内容を，128 バイト以内で指定します。指定を省略した場合，「)」が仮定されます。

LdapGroupMember エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定し，LdapGroup エントリに Yes を指定したときに有効になります。グループに所属しているユーザを記述する属性を指定します。指定を省略した場合，「uniqueMember」が仮定されます。

なお，LDAP 対応のディレクトリサービスとして Active Directory を使用する場合，「member」を指定してください。

LdapGroupFromUserAttr エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループ識別子としてユーザの属性値からディレクトリのノード情報を検索するか，検索しないかを指定します。

- Yes
  ユーザの属性値からディレクトリのノード情報を検索します。
- No
  ユーザの属性値からディレクトリのノード情報を検索しません。

指定を省略した場合，「No」が仮定されます。

LdapGroupIdFromUserAttr エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループ情報が格納されているユーザの属性を 64 バイト以内で指定します。指定を省略した場合，「ou」が仮定されます。

LdapGroupIsDnFromUserAttr エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。ユーザの属性に格納されているグループの情報が DN かどうかを指定します。

- Yes
  ユーザの属性に格納されているグループの情報が DN の場合，指定します。この場合，DocumentBroker は，LdapGroupIdAttrFromUserAttr エントリで指定されたグループの属性の情報を取得します。
- No
  ユーザの属性に格納されているグループの情報が DN でない場合，指定します。この場合，格納された情報を，そのままグループ識別子とします。

指定を省略した場合，「No」が仮定されます。

LdapGroupIdAttrFromUserAttr エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。ユーザの属性から取得した情報で，グループのグループ識別子として利用する属性を，64 バイト以内で指定します。指定を省略した場合，「dn」が仮定されます。

LdapGroupFromUserDn エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。グループ識別子をユーザエントリの DN の構成要素となる属性値から検索するか，検索しないかを指定します。

- Yes
  ユーザの DN からディレクトリのノード情報を検索します。
- No
  ユーザの DN からディレクトリのノード情報を検索しません。

指定を省略した場合，「No」が仮定されます。

LdapGroupIdFromUserDn エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。ユーザエントリの DN の構成要素の中で，グループ識別子として使用する属性名を，64 バイト以内で指定します。指定を省略した場合，「ou」が仮定されます。

LdapGroupTimeout エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。ディレクトリエントリからグループを検索するときの最大待ち時間を指定します。0 ～ 180（秒）の範囲で指定します。0 を指定した場合，待ち時間が無制限になります。指定を省略した場合，「60（秒）」が仮定されます。

LdapGroupCase エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。
DocumentBroker で扱うグループ識別子の文字種についての制限を指定します。

- Upper
  グループ識別子の文字列を，すべて大文字に変換する場合に指定します。LDAP 対応のディレクトリサービスから取得したグループ識別子を，DocumentBroker 内で大文字に変換します。
- Lower
  グループ識別子の文字列を，すべて小文字に変換する場合に指定します。LDAP 対応のディレクトリサービスから取得したグループ識別子を，DocumentBroker 内で小文字に変換します。
- Default
  グループ識別子の文字列を変換しません。

指定を省略した場合，「Default」が仮定されます。

LdapBindUserDN エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。
ディレクトリサービスからのデータ読み込み時に，ディレクトリサービスに対して認証を行ってバインドする場合は，認証に使用する DN を指定します。
指定を省略した場合，ディレクトリサービスからのデータ読み込み時に，ディレクトリサービスに対して匿名バインドが使用されます。
なお，ディレクトリサービスに対して認証を行ってバインドする場合は，必ず LdapBindUserDN エントリと LdapBindPassword エントリの両方を指定してください。どちらか一方でも指定を省略した場合は，ディレクトリサービスに対して匿名バインドが使用されます。

LdapBindPassword エントリ

UserAuthentication エントリで LDAP または LDAPEX を指定した場合に有効になります。
ディレクトリサービスからのデータ読み込み時に，ディレクトリサービスに対して認証を行ってバインドする場合は，認証に使用するパスワードを指定します。
なお，ディレクトリサービスに対して認証を行ってバインドする場合は，必ず LdapBindUserDN エントリと LdapBindPassword エントリの両方を指定してください。どちらか一方でも指定を省略した場合は，ディレクトリサービスに対して匿名バインドが使用されます。

DbType エントリ

文書空間が接続するデータベース種別を指定します。使用できるデータベースシステムは HiRDB です。したがって，このエントリには「HIRDB」と指定してください。なお，このエントリの指定は省略できません。

PdHost エントリ

データベースを構築しているサーバのホスト名を指定します。使用できるデータベースシステムは

HiRDB です。したがって，このエントリには HiRDB サーバのホスト名を指定してください。なお，このエントリの指定は省略できません。

PdNamePort エントリ

データベースを構築しているサーバのポート番号を指定します。使用できるデータベースシステムは HiRDB です。したがって，このエントリには HiRDB サーバのポート番号を指定してください。なお，このエントリの指定は省略できません。

PdUser エントリ

データベースにアクセスするための DocumentBroker 用のユーザ名とパスワードを指定します。使用できるデータベースシステムは HiRDB です。したがって，HiRDB のユーザ権限で CONNECT 権限およびスキーマ定義権限を付与したユーザを指定してください。指定方法を次に示します。

" ユーザ名 "/" パスワード "

HiRDB のユーザ権限については，「3.10.3　HiRDB のユーザ権限の設定」を参照してください。なお，このエントリの指定は省略できません。

DBConnectionPoolCount エントリ

サービスプロセス当たりの DB コネクションプール数を指定します。0 ～ 64 の間で指定してください。0 を指定した場合，プールされずに，要求ごとに DB コネクションを確立します。指定を省略した場合，「4」が仮定されます。

ただし，DBConnectionPoolCount エントリに指定した値と，次に説明する DBConnectionPoolDynamic エントリに指定した値を合わせて，1 以上になるように設定してください。

DBConnectionPoolDynamic エントリ

サービスプロセス当たりの一時コネクションの最大数を指定します。0 ～ 64 の間で指定してください。0 を指定した場合，一時コネクションは確立されません。指定を省略した場合，「4」が仮定されます。

ただし，上記で説明した DBConnectionPoolCount エントリに指定した値と，DBConnectionPoolDynamic エントリに指定した値を合わせて，1 以上になるように設定してください。

DBConnectionPoolTiming エントリ

プールする DB コネクション作成のタイミングを指定します。

- Static
  静的（DocumentBroker 起動時）にプールするすべてのコネクションを作成します。
- Dynamic
  動的（コネクション解放時）にプールします。

指定を省略した場合，「Static」が仮定されます。

DBConnectionPoolOver エントリ

プールしている DB コネクションがすべて使用中，かつ一時コネクションが最大数に達している場合のアクションを指定します。

- Error
  クライアントにエラーの戻り値を返却します。
- FIFO
  コネクション待ち行列に入れ，空きコネクションが発生したときに割り当てます。

指定を省略した場合，「Error」が仮定されます。

DBConnectionScope エントリ

DB コネクション割り当て期間を指定します。

- Connection
  文書空間への接続単位に割り当てます。文書空間への接続時に割り当て，文書空間への接続解除時に解放します。
- Transaction
  トランザクション単位に割り当てます。トランザクション開始時に割り当て，トランザクション終了時に解放します。

指定を省略した場合，「Transaction」が仮定されます。

DBConnectionPoolWaitTimeOut エントリ

DBConnectionPoolOver エントリに「FIFO」を指定している場合，DB コネクション割り当て待ち時間を 0 ～ 7,200,000（ミリ秒）の範囲で指定します。
0 を指定した場合，コネクション割り当て待ち時間が無制限になります。
このエントリの指定によってタイムアウトが発生した場合，エラーメッセージを出力して，DB コネクション割り当て待ちリストから削除します。この時，DocumentBroker クライアントには，戻り値として DMARC_LOST_CONNECTION を返却します。指定を省略した場合，「0（ミリ秒）」が仮定されます。

BatchSizeHint エントリ

BatchSizeHint エントリの値には，データベース上の検索結果集合から，検索結果を取得する際にメモリにバッファリングする検索結果行の行数を指定できます。最適値は「1」です。なお，指定を省略した場合，「1」が仮定されます。よって，このエントリの指定は不要です。

PdTSPluginOwner エントリ

全文検索および構造指定検索を実行する場合に，HiRDB Text Search Plug-in を登録したユーザの認可識別子を指定します。
なお，HiRDB Text Search Plug-in はデフォルト（MASTER）のスキーマに格納されるため，このエントリの指定は不要です。HiRDB Text Search Plug-in の登録時に，pdplgrgst コマンドの -u オプション（プラグイン登録スキーマ指定）を使用した場合だけ，このエントリの指定が必要です。

AcLogUse エントリ

アクセスログの取得の有無を指定します。

- Yes
  アクセスログを取得します。
- No
  アクセスログを取得しません。

指定を省略した場合，「No」が仮定されます。

AcLogLevel エントリ

アクセスログに出力するアクセスレベルを指定します。指定した出力レベルによってアクセスログに出力する出力ログ情報を調節します。表 4-5 に出力レベルによる出力ログ情報を示します。
アクセスログレベルおよび取得される情報の詳細については，「6.5　アクセスログに関する運用」を参照してください。なお，指定を省略した場合，「Write」が仮定されます。

表 4-5 アクセスログの出力レベルと出力ログ情報

出力レベル	出力ログ情報
Write	• セッションの確立 • セッションの切断 • オブジェクトの作成 • オブジェクトの削除 • オブジェクト間の関連づけの設定 • オブジェクト間の関連づけの解除 • ACL のバインド • ACL のバインドの解除 • 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 • チェックアウトの取り消し • バージョンの削除 • バージョンの固定 • バージョンの固定の解除 • リレーション情報の作成 • リレーション情報の削除 • レンディションの作成 • レンディションの削除 • マスタレンディションの変更 • プロパティの設定 • ファイルのアップロード • 全文検索インデクスの作成 • 全文検索インデクスの削除
Read	• 出力レベルが「Write」の場合に出力される情報 • オブジェクトとの接続 • オブジェクトの接続の解除 • オブジェクト一覧の取得 • オブジェクトを包含するコンテナ一覧の取得 • コンテナに包含されるオブジェクト一覧の取得 • ACL 一覧の取得 • バージョン一覧の取得 • バージョン管理情報一覧の取得 • リザベーションの取得 • リレーション情報の一覧取得 • レンディションの一覧取得 • プロパティの取得 • ファイルのダウンロード • edmSQL の構文チェック • 問い合わせの実行 • 問い合わせの実行と問い合わせ結果の取得 • 問い合わせ結果の削除
Error	• 出力レベルが「Read」の場合に出力される情報 • エラー

AcLogFileCount エントリ

アクセスログを取得しているファイルのサイズが，AcLogFileSize エントリに指定した上限値を超えた場合に，切り替えるファイル数を 2 ～ 16 で指定します。出力ファイル名は EDMAccess_NO.log で，「NO」は出力ファイル通番を示します。出力ファイル通番とは，1 ～ AcLogFileCount エントリに指定した数（出力ファイル数）です。

アクセスログは，出力ファイル通番が「1」のファイルから順番に出力されます。あるファイルにアクセスログを出力する場合に，ファイルサイズが AcLogFileSize エントリに指定した出力ファイルサイズよりも大きくなるとき，出力ファイル通番が一つ大きいファイルに出力します。例えば，出力ファイル通番が「1」であるファイルにアクセスログを出力する場合に，このファイルのサイズの最大量を超えてしまうときは，ファイル出力通番が「2」のファイルにログを出力します。

AcLogFileCount エントリに指定した最大のファイル通番のファイルにアクセスログを出力しようと

して，そのファイルのファイルサイズが AcLogFileSize エントリに指定した出力ファイルサイズよりも大きくなってしまう場合，出力ファイル通番が「1」のファイルを初期化してログの出力を継続します。
指定を省略した場合，「2」が仮定されます。

AcLogFileSize エントリ

アクセスログファイルのサイズを，4,096 ～ 2,147,483,647（バイト）で指定します。アクセスログの出力が指定されたサイズを超える場合，次の通番のファイルへ出力を切り替えます。また，カレントファイル出力中に異常（入出力エラーなど）が発生した場合も，次の通番のファイルへ出力を切り替えます。ただし，切り替えは 1 回だけ実行します。
指定を省略した場合，「1,048,576（1 メガバイト）」が仮定されます。

XdkShmemManage エントリ

メタ情報管理用の共用メモリの確保，解放の方法を指定します。指定する値によって，メタ情報管理に使用する共用メモリの確保および解放のタイミングを変更できます。

- NORMAL
  共用メモリを再利用しない場合に指定します。通常の運用の場合，NORMAL を指定してください。メタ情報管理用の共用メモリは，DocumentBroker サーバの起動時に確保され，終了時に解放されます。
- MEMHOLD
  共用メモリを再利用する場合に指定します。共用メモリ領域のフラグメントの発生によって，DocumentBroker サーバの起動時に，メタ情報の管理に必要な共用メモリが確保できないことがあります。このような場合に，MEMHOLD を指定してください。
  MEMHOLD を指定した場合，メタ情報管理用の共用メモリは，DocumentBroker サーバの起動時に確保され，通常の終了時には解放されません。ただし，強制終了時には，解放されます。メタ情報管理用の共用メモリの確保方法は，要求サイズ（環境変数「XDK_SHMEM_SIZE」の値，または XdkShmemSize エントリに指定する値）と確保済みの共用メモリサイズによって異なります。
  要求サイズ（環境変数「XDK_SHMEM_SIZE」の値，または XdkShmemSize エントリに指定する値）よりも確保済みの共用メモリサイズが大きい場合，DocumentBroker サーバの起動時にはメタ情報を確保済みの共用メモリに読み込みます。要求サイズ（環境変数「XDK_SHMEM_SIZE」の値，または XdkShmemSize エントリに指定する値）よりも，確保済みの共用メモリサイズが小さい場合，確保済みの共用メモリを解放して要求サイズのメモリを確保してからメタ情報を読み込みます。

XdkShmemSize エントリ

メタ情報管理用に確保する共用メモリセグメントサイズを，3,000,000 ～ 1,073,741,824（バイト）で指定します。指定を省略した場合，「3,000,000（バイト）」が仮定されます。なお，この共用メモリセグメントサイズは，環境変数「XDK_SHMEM_SIZE」にも指定できます。ただし，XdkShmemSize エントリおよび環境変数「XDK_SHMEM_SIZE」の両方に値を指定している場合は，
XdkShmemSize エントリの値が優先されます。したがって，メタ情報管理用に確保する共用メモリセグメントサイズは，XdkShmemSize エントリに指定することを推奨します。メタ情報管理用に確保する共用メモリセグメントサイズに設定される値を次の表に示します。

表 4-6　メタ情報管理用に確保する共用メモリセグメントサイズの値

XDK_SHMEM_SIZE	XdkShmemSize エントリ	共用メモリセグメントサイズの値
○	○	XdkShmemSize エントリの指定値
○	－	XDK_SHMEM_SIZE の指定値

XDK_SHMEM_SIZE	XdkShmemSize エントリ	共用メモリセグメントサイズの値
－	○	XdkShmemSize エントリの指定値
－	－	3,000,000 バイト

（凡例）
○：指定あり。
－：指定なし。

なお，オペレーティングシステムのページングファイルサイズの値には，このメモリマップトファイルが確保できる十分なサイズを指定してください。
AIX の場合，次の点に注意してください。

- 環境変数「EXTSHM」の値には「ON」を指定してください。

共用メモリセグメントを確保できない場合は，DocumentBroker サーバが停止します。この時，KMBR02004-E のメッセージが出力されます。メッセージの詳細については，マニュアル「DocumentBroker Version 3　メッセージ」を参照してください。
XdkShmemSize エントリおよび環境変数「XDK_SHMEM_SIZE」に指定する，メタ情報管理用に確保する共用メモリセグメントサイズの見積もり式を次に示します。

メタ情報管理用に確保する共用メモリセグメントサイズ（バイト）
＝ 3,000,000 ＋ 10,640 × A ＋ 12,584 × B ＋ 752 × C

（凡例）

- A：ユーザが定義するクラス数を表します。
- B：ユーザが定義するプロパティ数を表します。
- C：ユーザが定義するクラスに定義するユーザ定義のプロパティの総数を表します。この値はメタ情報の追加コマンド（EDMAddMeta）に指定する定義情報ファイル内の [AddProperty/クラス名] の数です。

FtpBufferSize エントリ

ファイル分割転送機能を使用してファイル転送する場合の，データ転送サイズを指定します。指定できる値の範囲は，4,096 ～ 2,147,483,647（バイト）です。なお，4,096 ～ 65,000（バイト）の範囲で指定することを推奨します。
このエントリの記述を省略した場合は，ファイル分割転送機能を使用しないでファイル転送を実行します。なお，ファイル分割転送機能については，「3.13.3(2)　ファイル分割転送機能の設定」を参照してください。

SelectServerInMultiServer エントリ

複数の実行環境から，同じ文書空間にアクセスする運用形態の場合に，クライアント側で接続する DocumentBroker サーバを選択するかどうかを指定します。

- Yes
  クライアント側で接続する DocumentBroker サーバを選択します。
- No
  クライアント側で接続する DocumentBroker サーバを選択しません。

「Yes」を指定した場合，クライアント側で，接続する DocumentBroker サーバの実行環境識別子を指定する必要があります。指定方法については，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」を参照してください。上記以外の値を指定した場合，DocumentBroker サーバの起動時にエラーが発生します。指定を省略した場合，「No」が仮定されます。
なお，同じ文書空間にアクセスするすべての実行環境で SelectServerInMultiServer エントリの値を同じにしてください。また，DocumentBroker サーバのバージョンが 02-40 より前のメタ情報ファイ

ルを使用している場合，このエントリを省略するか，「No」を指定してください。「Yes」を指定した場合，DocumentBroker サーバ起動時にエラーが発生します。

SRefreshLimit エントリ

DocumentBroker サーバがサービスプロセスをリフレッシュ（サービスプロセスを再起動）する契機となるサービスプロセスのメモリサイズを指定します。サービスプロセスが指定されたメモリサイズを超過すると，DocumentBroker サーバはサービスプロセスをリフレッシュします。リフレッシュすると，メモリ所要量等のリソースを DocumentBroker 起動直後の状態にすることができます。
メモリサイズ監視によるサービスプロセスのリフレッシュは，1 プロセスずつ行います。複数のサービスプロセスが，指定したメモリサイズを超過している場合，1 プロセスのリフレッシュ完了後，順次 1 プロセスずつリフレッシュします。
メモリサイズ監視によるリフレッシュ実行中に，サービスプロセスのリフレッシュコマンド（EDMRefresher）が実行された場合，サービスプロセスのリフレッシュコマンド（EDMRefresher）はメモリサイズ監視によるリフレッシュの終了を待ってから実行されます。複数のサービスプロセスが指定したメモリサイズを超過しており，そのうち 1 つのサービスプロセスのリフレッシュを実行中に，サービスプロセスのリフレッシュコマンド（EDMRefresher）が実行された場合，実行中のサービスプロセスのリフレッシュ終了後，サービスプロセスのリフレッシュコマンド（EDMRefresher）によるリフレッシュが全サービスプロセスに対して順次実行されます。そのため，メモリサイズが本エントリの指定値を超過している残りのサービスプロセスは，サービスプロセスのリフレッシュコマンド（EDMRefresher）によってリフレッシュされます。

指定内容と指定できる値は次のとおりです。

起動時以降に 1 サービスプロセスで増加するヒープメモリサイズの上限値を指定します。増加量が指定した値を超過した時，メモリサイズ監視によるサービスプロセスのリフレッシュが動作します。指定できる値は，64 ～ 2048 です。単位はメガバイトで指定してください。指定を省略した場合，サービスプロセスのメモリサイズ監視は行いません。

指定値は，アプリケーションが扱う最大文書サイズや同時作業文書数等を考慮して設定してください。指定値が小さ過ぎると，頻繁にサービスプロセスのリフレッシュが実行され性能低下の原因となります。
また，サービスプロセスをリフレッシュすると，アプリケーションプロセスとサービスプロセスの通信で使用するポート番号がリフレッシュ前後で変更される可能性があります。アプリケーションプロセスがサービスプロセスに接続後，メモリサイズ監視によるリフレッシュが動作し通信ポート番号が変更され，アプリケーションプロセスとリフレッシュ前のサービスプロセスの通信で使用していたポート番号がほかのプログラムに使用されると，以後アプリケーションプロセスとサービスプロセスとの通信で無応答が発生する原因となります。本現象を回避するために，アプリケーションプロセスとサービスプロセスとの通信で使用するポート番号を固定してください。アプリケーションプロセスとサービスプロセスとの通信で使用する固定ポート番号は，サービスプロセス定義ファイルに指定することができます。固定ポート番号の指定については，「4.13　サービスプロセス定義ファイル」と，マニュアル「VisiBroker for C++ プログラマーズガイド」またはマニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照願います。

SRefreshGraceTime エントリ

メモリサイズ監視によってサービスプロセスをリフレッシュする時に，ユーザがログアウトまたはトランザクションを決着するための猶予時間を秒単位で指定します。RefreshTiming エントリに「Connection」を指定した場合は，ユーザがログアウトするための猶予時間になり，「Transaction」を指定した場合は，トランザクションを決着する，またはユーザがログアウトするための猶予時間になります。接続中のユーザが存在する，または未決着のトランザクションが存在する場合，少なくともこの猶予時間はリフレッシュされません。指定した猶予時間が経過した場合，接続中のユーザは強制的にサーバとの接続を切断されます。この時，各 API で次のエラーが返却される場合があります。

major_code = ERR_DBR, minor_code = ERR_SESSION_NOT_CONNECT

接続中のユーザが存在しない，または未決着のトランザクションが存在しない場合は直ちにメモリサイズ監視によるリフレッシュが実行されます。また，猶予時間内に接続中のユーザがログアウトした場合，または未決着のトランザクションが決着された場合は，本エントリに指定した値まで猶予時間が経過するのを待たずにメモリサイズ監視によるリフレッシュが実行されます。
メモリサイズ監視によるリフレッシュ実行中に，ユーザが文書空間に接続した時，
CdbrSession::Connect メソッドで，次のエラーが返却される場合があります。

- major_code = ERR_DBR, minor_cde = ERR_NO_SERVICE
- major_code = ERR_DMA, minor_code = DMARC_NETWORK_UNAVAILABLE

指定できる値は，0 ～ 7200 です。0 を指定した場合，接続中ユーザの有無または未決着トランザクションの有無にかかわらず直ちにメモリサイズ監視によるリフレッシュが実行されます。
指定を省略した場合，「600」が仮定されます。

RefreshTiming エントリ

サービスプロセスのリフレッシュコマンド（EDMRefresher）やメモリサイズ監視によるサービスプロセスのリフレッシュを猶予する場合，猶予時間内であってもリフレッシュが実行される契機となる事象を指定します。このエントリに指定した状態になるか，猶予時間が経過するまでプロセスの終了が猶予されます。

- Connection
  リフレッシュ対象サービスプロセスに接続したすべてのユーザがログアウトするまで猶予されます。この時サーバへの接続要求が実行された場合，猶予中のサービスプロセスには接続できません。猶予中のサービスプロセスに接続済みのユーザは，新たにトランザクションを開始することができます。
- Transaction
  リフレッシュ対象サービスプロセスで実行中のすべてのトランザクションが決着する，またはリフレッシュ対象サービスプロセスに接続したすべてのユーザがログアウトするまで猶予されます。この時サーバへの接続要求が実行された場合，猶予中のサービスプロセスには接続できません。猶予中のサービスプロセスに接続済みのユーザは，新たにトランザクションを開始することができません。トランザクションの開始を実行すると，次のエラーが返却されます。

  major_code = ERR_DBR，minor_code = ERR_SESSION_NOT_CONNECT

指定を省略した場合，「Connection」が仮定されます。

VArrayElementCheck エントリ

VariableArray 型プロパティの要素に指定した値をチェックするか，チェックしないかを指定します。

- Yes
  VariableArray 型プロパティの要素の値をチェックします。
- No
  VariableArray 型プロパティの要素の値をチェックしません。

次に，VariableArray 型のプロパティに指定した値のデータ型と実行されるチェックの内容について次の表に示します。なお，指定を省略した場合，「No」が仮定されます。

表 4-7 VariableArray 型プロパティのデータ型と実行されるチェックの内容

プロパティに指定した値のデータ型	チェックの内容
DMA_DATATYPE_BOOLEAN	DMA_TRUE，DMA_FALSE，または DMA_UNKNOWN 以外は DMARC_BAD_VALUE のエラーになる。
DMA_DATATYPE_INTEGER32	プロパティの最小値，および最大値をメタ情報から取得し，この範囲外であれば DMARC_BAD_VALUE のエラーになる。※1
DMA_DATATYPE_STRING	プロパティの最大文字列長をメタ情報から取得し，これを超える場合は DMARC_BAD_VALUE のエラーになる。※2

注※ 1

プロパティの最小値および最大値は，「− 2,147,483,648 ～ 2,147,483,647」です。最小値よりも小さい値または最大値よりも大きい値の場合，エラーになります。

注※ 2

プロパティの最大文字列長は，ユーザが VariableArray 型プロパティを定義する場合に設定した最大文字列長です。最大文字列長よりも長い文字列の場合，エラーになります。

ErrChkFlagOfObjectOperation エントリ

DocumentBroker のオブジェクトを操作するときに実行するエラーチェックを bit フラグで指定します。なお，DocumentSpace 構成定義ファイルで定義されている値を変更できるのは，次の場合だけです。

- ユーザのアプリケーションプログラムが動作していた実行環境を移行する場合
  アプリケーションが動作していた DocumentBroker 実行環境の DocumentSpace 構成定義ファイルで指定されている値に変更してください。
- 複数の実行環境から同じ文書空間にアクセスする実行環境を構築する場合
  メタ情報初期設定コマンド（EDMInitMeta）を実行した DocumentBroker 実行環境の DocumentSpace 構成定義ファイルで指定されている値に変更してください。

EnbFncFlagOfObjectOperation エントリ

DocumentBroker の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。なお，DocumentSpace 構成定義ファイルで定義されている値を変更できるのは，次の場合だけです。

- ユーザのアプリケーションプログラムが動作していた実行環境を移行する場合
  アプリケーションが動作していた DocumentBroker 実行環境の DocumentSpace 構成定義ファイルで指定されている値に変更してください。
- 複数の実行環境から同じ文書空間にアクセスする実行環境を構築する場合
  メタ情報初期設定コマンド（EDMInitMeta）を実行した DocumentBroker 実行環境の DocumentSpace 構成定義ファイルで指定されている値に変更してください。

BlobSubstrMode エントリ

マルチファイル管理機能を使用する場合の，BLOB データの取得・格納方法を指定します。

- ALL
  文書からコンテントを取得する場合，BLOB に格納されたファイルを 1 回のデータベースへのアクセスで取得して，ファイル単位に出力します。
  文書にコンテントを格納する場合（文書を作成する場合または文書のコンテントを更新する場合），1 回のデータベースへのアクセスで，複数のファイルを一つの BLOB に格納します。
  この方式では，データベースへのアクセスの回数が削減されます。ただし，BLOB データの取得先には，一つの文書に格納しているファイルの合計サイズ分のメモリまたはディスク容量が必要です。
- ELEMENT
  文書からコンテントを取得する場合，BLOB に格納されたファイルをファイル単位のデータベース

へのアクセスで取得して，ファイル単位に出力します。
文書にコンテントを格納する場合（文書を作成する場合または文書のコンテントを更新する場合），ファイル単位のデータベースへのアクセスで，複数のファイルを一つの BLOB に格納します。
この方式では，データベースへのアクセス回数は増加します。ただし，BLOB データの取得先に必要なメモリまたはディスク容量が，一つのファイルサイズ分だけになります。

- THRESHOLD
  文書からコンテントを取得する場合，BLOB に格納されたファイルを BlobSubstrThreshold エントリに指定するサイズ単位のデータベースへのアクセスで取得して，ファイル単位に出力します。
  文書にコンテントを格納する場合（文書を作成する場合または文書のコンテントを更新する場合），BlobSubstrThreshold エントリに指定するサイズ単位のデータベースへのアクセスで，複数のファイルを一つの BLOB に格納します。
  この方式では，データベースへのアクセスの回数は，すべてのファイルの合計サイズを BlobSubstrThreshold エントリに指定するサイズで割った商 +1 の値になります。また，BLOB データの取得先には，BlobSubstrThreshold エントリに指定するサイズ分のメモリまたはディスク容量が必要です。

指定を省略した場合，「ELEMENT」が仮定されます。

BlobSubstrThreshold エントリ

マルチファイル管理機能を使用する場合で，BlobSubstrMode エントリの値に THRESHOLD を指定しているときに，1 回のデータベースへのアクセスで取得・格納するデータのサイズを指定します。
指定する値の単位はバイトです。4,096 ～ 5,242,880（バイト）の間で指定してください。
なお，BlobSubstrMode エントリの値に ALL または ELEMENT を指定している場合，BlobSubstrThreshold エントリの設定値は無視されます。
指定を省略した場合，「1,048,576（バイト）」が仮定されます。

BlobGettingMethod エントリ

データベースから BLOB データを取得するときの取得先を指定します。

- Mem
  メモリを使用して BLOB データを取得します。
  「Mem」を指定する場合は，メモリ所要量を確保してください。容量の見積もりについては，「2.7.1 仮想メモリ所要量の見積もり」を参照してください。
- File
  BLOB データを "$DOCBROKERDIR/tmp" 下に一時ファイルとして取得します。
  「File」が指定できるのは，DocumentBroker サーバと HiRDB サーバが同一マシンの場合だけです。DocumentBroker サーバと HiRDB サーバが異なるマシンの場合は，データベースから BLOB データを取得しようとしたときにエラーになります。
  「File」を指定する場合は，一時ファイルの容量を確保してください。容量の見積もりについては，「2.7.3　ディスク占有量の見積もり」を参照してください。

指定を省略した場合，「Mem」が仮定されます。

LifeCycleSuiteConnection エントリ

DocumentBroker Life Cycle Suite を使用して WorkCoordinator と連携するか，連携しないかを指定します。

- Yes
  DocumentBroker Life Cycle Suite を使用して WorkCoordinator と連携します。
- No
  DocumentBroker Life Cycle Suite を使用して WorkCoordinator と連携しません。

指定を省略した場合，「No」が仮定されます。
なお，このエントリは，TPBroker V3 環境の場合に指定するエントリです。TPBroker V5 と連携して動作する環境では指定できません。TPBroker V5 環境で指定した場合は，指定が無視され省略値も有効になりません。

4.2.3 DocumentSpace 構成定義ファイルの記述例

DocumentSpace 構成定義ファイルの記述例を，次に示します。

(1) 前提条件

定義例は次の前提条件に従っていることとします。

● LDAP 対応のディレクトリサービスを使用してユーザ管理をしている。

● LDAP 対応のディレクトリサービスで構築されている DIT は図 4-1 に示す。

● 各エントリに対応するディレクトリ情報は表 4-8 に示す。

図 4-1 DIT の例

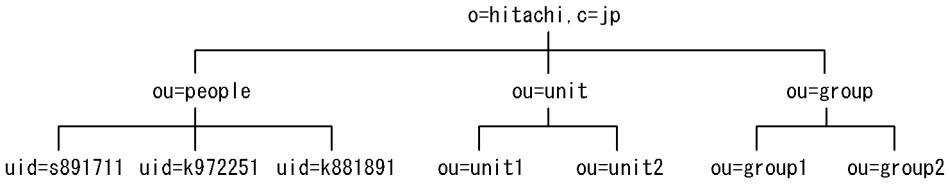


表 4-8 エントリに対応するディレクトリ情報の例

エントリ	各エントリに対応するディレクトリ情報
o: hitachi,c=jp	dn: o=hitachi,c=jp objectclass: top objectclass: organization o: hitachi,c=jp
ou=people	dn:ou=people,o=hitachi,c=jp objectclass: top objectclass: person objectclass: organizationalperson objectclass: inetOrgPerson sn: person cn: personal
uid: s891711	dn: uid=s891711,ou=people,o=hitachi,c=jp objectclass: top objectclass: person objectclass: organizationalperson objectclass: inetOrgPerson cn: Taro Hitachi cn;lang-ja: 日立太郎 uid: s891711 sn: hitachi ou: unit1 ou: unit userpassword: s891711

エントリ	各エントリに対応するディレクトリ情報
uid: k972251	dn: uid=k972251,ou=people,o=hitachi,c=jp objectclass: top objectclass: person objectclass: organizationalperson objectclass: inetOrgPerson cn: Ichiro Yamada cn;lang-ja: 山田 一郎 uid: k972251 sn: yamada ou: unit2 ou: unit userpassword: k972251
uid:k881891	dn: uid=k881891,ou=people,o=hitachi,c=jp objectclass: top objectclass: person objectclass: organizationalperson objectclass: inetOrgPerson cn: Jiro Suzuki cn;lang-ja: 鈴木 二郎 uid: k881891 sn: suzuki ou: unit2 ou: unit userpassword: k881891
ou: unit	dn: ou=unit,o=hitachi,c=jp objectclass: top objectclass: organizationalUnit ou: unit ou;lang-ja: ユニット
ou: unit1	dn: ou=unit1,ou=unit,o=hitachi,c=jp objectclass: top objectclass: organizationalUnit ou: unit1 ou;lang-ja: ユニット 1
ou: unit2	dn: ou=unit2,ou=unit,o=hitachi,c=jp objectclass: top objectclass: organizationalUnit ou: unit2 ou;lang-ja: ユニット 2
ou: group	dn: ou=group,o=hitachi,c=jp objectclass: top objectclass: groupOfUniqueNames objectclass: hdsgroupofuniquenames ou: group ou;lang-ja: グループ cn:grp0 groupOfUniqueNamesId: g0
ou: group1	dn: ou=group1,ou=group,o=hitachi,c=jp objectclass: top objectclass: groupOfUniqueNames objectclass: hdsgroupofuniquenames cn: grp1 ou: group1 ou;lang-ja: グループ 1 groupOfUniqueNamesId: g1 uniqueMember: uid=s891711,ou=people,o=hitachi,c=jp

エントリ	各エントリに対応するディレクトリ情報
ou: group2	dn: ou=group2,ou=group,o=hitachi,c=jp objectclass: top objectclass: groupOfUniqueNames objectclass: hdsgroupofuniquenames cn:grp2 ou: group2 ou;lang-ja: グループ 2 groupOfUniqueNamesId: g2 uniqueMember:uid=k972251,ou=people,o=hitachi,c=jp uniqueMember:uid=k881891,ou=people,o=hitachi,c=jp

(2) 定義例

前提条件に基づいた DocumentSpace 構成定義ファイルの定義例を次に示します。

```
[DocSpace]
Count = 1

[Entry0001]
Process = 1
SerialId = 673d2be0-d1fd-11d0-ab59-08002be29e1d
DbType = HIRDB
PdHost = d_ks5g
PdNamePort = 20570
PdUser = "hirdb"/"hirdb"
XdkShmemManage = NORMAL
SessionMax = 32
SessionTimeOut = 15
ACLogUse = No
ACLogLevel = Error
ErrChkFlagOfObjectOperation = 0x0000017f
EnbFncFlagOfObjectOperation = 0x0000007f

UserAuthentication = LDAPEX
LdapHost = 123.45.67.78
LdapPort = 389
LdapUserRoot = o = hitachi,c = jp
LdapGroup = Yes
LdapGroupFromUserAttr = Yes
LdapUserScope = Subtree
LdapUserTimeout = 60
LdapGroupTimeout = 60
LdapUserClass = inetOrgPerson
LdapUserId = uid
LdapUserFilterLeft = (
LdapUserFilterRight = )
LdapUserCase = Default
LdapGroupFromUserDn = Yes
LdapGroupClass = hdsgroupOfUniqueNames
LdapGroupId = groupOfUniqueNamesId
LdapGroupRoot = o = hitachi,c = jp
LdapGroupFilterLeft = (
LdapGroupFilterRight = )
LdapGroupIdFromUserAttr = ou
LdapGroupIsDnFromUserAttr = No
LdapGroupIdAttrFromUserAttr = ou
LdapGroupIdFromUserDn = ou
LdapGroupCase = Default
LdapBindUserDN = uid=Tanaka,ou=People,o=hitachi,c=jp
LdapBindPassword = password
```

(3) 認証，ユーザ情報の取得例

ここでは，例で示した DocumentSpace 構成定義ファイルを使用した場合のユーザ認証とユーザ情報の取得について説明します。なお，説明ではログイン名は「s891711」であると仮定します。

認証

UserAuthentication エントリが LDAPEX であるため，DocumentBroker は次のような検索条件に適合するユーザのエントリを検索します。

検索基点

LdapUserRoot エントリで指定されている「o=hitachi,c=jp」です。

検索範囲

LdapUserScope エントリに「Subtree」が指定されているので検索基点からのすべてのサブツリーが検索範囲となります。

検索フィルタによる検索条件

LdapUserFilterLeft エントリ，LdapUserClass エントリ，LdapUserId エントリおよび LdapUserFilterRight エントリの指定によって，「(&(objectclass=inetOrgPerson)(uid=s891711))」となります。

この検索によって見つかったエントリの userpassword 属性の値と入力されたパスワードを比較して，一致すれば認証成功になります。

ユーザ情報取得

LdapGroup エントリが Yes であるため，グループのエントリを検索します。この場合，次のような検索条件を用いて検索します。

検索基点

LdapGroupRoot エントリで指定されている「o=hitachi,c=jp」です。

検索範囲

LdapGroupScope エントリに「Subtree」が指定されているので検索基点からのすべてのサブツリーが検索範囲となります。

検索条件

LdapGroupFilterLeft エントリ，LdapGroupClass エントリおよび LdapGroupFilterRight エントリの指定によって，「(&(objectclass=hdsgroupOfUniqueNames)(uniqueMember=uid=s891711,ou=people,o=hitachi,c=jp))」となります。

この検索によって見つかったエントリの groupOfUniqueNamesId 属性（LdapGroupId エントリで指定されている属性）の値「g1」をグループ識別子として取得します。

LdapGroupFromUserAttr エントリが Yes であるため，ユーザの属性値からグループ情報を取得します。この場合，ユーザの ou 属性（LdapGroupIdFromUserAttr エントリで指定されている属性）の値は「unit1」および「unit」ですが，LdapGroupIsDnFromUserAttr エントリが No であるため，その値をそのままグループ識別子として取得します。

LdapGroupFromUserDn エントリが Yes であるため，ユーザの dn からグループ情報を取得します。この場合，ユーザの dn:uid=s891711,ou=people,o=hitachi,c=jp の ou（LdapGroupIdFromUserDn エントリで指定した DN のキー）の値「people」をグループ識別子として取得します。

このような検索結果から「g1」,「unit1」,「unit」および「people」をこのユーザのグループ識別子として取得できます。

4.2.4 DocumentSpace 構成定義ファイルの注意事項

DocumentSpace 構成定義ファイルで文書空間を定義する場合の注意事項について説明します。

（1） SessionMax エントリを指定する場合の注意事項

DocumentBroker は 1 セッション当たり 7 ～ 8 個のファイル記述子を利用します。オペレーティングシステムのデフォルトの設定では，1 プロセス当たり 2,000 個のファイル記述子を利用できるようになっています。しかし，DocumentBroker 起動時に利用可能なファイル記述子は 1,960 個です。すなわち，デフォルトの指定では 1 プロセス当たり 245 人までログインできます。したがって，Process エントリで指定したプロセス数で SessionMax エントリで指定したユーザにサービスを供給できるようにオペレーティングシステムのカーネルパラメタ（nofiles（AIX の場合）または nofile（Linux の場合））を変更してください。変更方法を次に示します。

nofiles または nofile ＞＝（SessionMax 指定値÷ Process 指定値）× 8 ＋ 40

（2） 検索フィルタへ検索条件を追加する場合の注意事項

アクセス制御されている文書に接続する場合，DocumentBroker は，ユーザの属性からのユーザ情報を取得して，アクセスしたい文書にアクセスできるかどうか判断します。ユーザ情報を取得するとき，検索フィルタに検索条件を指定してユーザを検索します。この検索フィルタには，デフォルトの検索フィルタとデフォルトの検索フィルタに検索条件を追加した検索フィルタがあります。

DocumentSpace 構成定義ファイルでデフォルトの検索フィルタに検索条件を追加するためには，LdapUserFilterLeft エントリおよび LdapUserFilterRight エントリを指定しておく必要があります。

LdapUserFilterLeft エントリおよび LdapUserFilterRight エントリの概要

LdapUserFilterLeft エントリおよび LdapUserFilterRight エントリは，ユーザ識別子を検索するときにデフォルトの検索フィルタに対して，検索条件を追加して指定します。したがって，デフォルトフィルタに情報を付与して，特定の情報を参照できるユーザを検索できます。

ユーザ識別子を検索する場合に DocumentBroker が指定するフィルタ

DocumentBroker はユーザ識別子を検索する場合，次に示すデフォルトフィルタを指定します。

&(objectclass=$LdapUserClass$)($LdapUserId$= ログインユーザ名)

ただし，x は，DocumentSpace 構成定義ファイルでエントリ x に指定された値です。

このデフォルトフィルタの意味は，(objectclass=$LdapUserClass$) および ($LdapUserId$= ログインユーザ名) が真となるユーザ識別子を検索条件として指定するということです。

このデフォルトフィルタに対して LdapUserFilterLeft エントリおよび LdapUserFilterRight エントリにユーザ識別子の検索情報を指定して，検索フィルタを設定します。

デフォルトフィルタへの検索条件の追加例

次の例を基に検索フィルタの設定方法を説明します。

（例）

ユーザが使用するアプリケーションプログラムを示す「ApplicationType=DocumentBroker」という属性および属性値を，デフォルトフィルタに追加する。

「ApplicationType という属性の属性値が DocumentBroker であるユーザ」をデフォルトフィルタに追加条件として付与する場合，LdapUserFilterLeft エントリおよび LdapUserFilterRight エントリには次のように指定します。

- LdapUserFilterLeft=(
- LdapUserFilterRight=(ApplicationType=DocumentBroker))

ただし，指定する際，「(」（左括弧）と「)」（右括弧）の個数をそろえる必要があります。これ

によって作成される検索フィルタを次に示します。

(&(objectclass=$LdapUserClass$)($LdapUserId$= **ログインユーザ名**)
(ApplicationType=DocumentBroker))
この検索フィルタは，(objectclass=$LdapUserClass$)，($LdapUserId$= ログインユーザ名)，および (ApplicationType=DocumentBroker)) が真となるユーザ識別子を検索することを意味します。

なお，グループ識別子の検索フィルタに検索情報を追加指定するためのエントリとして，LdapGroupFilterLeft エントリおよび LdapGroupFilterRight エントリがあります。これらのエントリの指定方法は，LdapUserFilterLeft エントリおよび LdapUserFilterRight エントリの指定方法を参考にしてください。

(3) 文書空間の構成を変更する場合の注意事項

DocumentBroker の運用を開始してから，セッションタイムアウト時間の変更やデータベースから取り出す検索結果行の行数の変更などによって，文書空間の構成を変更するような場合が考えられます。このような場合は，次の手順で文書空間の構成を変更してください。

1. DocumentBroker **を終了する。**
2. DocumentSpace **構成定義ファイルを変更する。**
3. DocumentBroker **を再起動する。**

なお，DocumentBroker の起動と終了については，「5.　DocumentBroker の起動と終了」を参照してください。

(4) アクセス制御機能を使用する場合の注意事項

アクセス制御機能を利用する環境では，UserAuthentication エントリの値として，LDAP，LDAPEX，または UOC を必ず指定してください。これらの値以外が指定されている場合や指定が省略されている場合は，DocumentBroker サーバが起動しません。

(5) ProcessOrbBoaOption **エントリを指定する場合の注意事項**

-OAthreadMax オプションの値を 129 以上としている場合に，129 以上のクライアントから CdbrSession::Connect メソッドを同時に実行すると，タイミングによっては，CdbrSession::Connect メソッドからエラーが返却されることがあります。

なお，CdbrSession::Connect 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　C++　解説」を参照してください。

4.3 セキュリティ定義ファイル（docaccess.ini）

この節では，セキュリティ定義ファイルについて説明します。

セキュリティ定義ファイルは，" 実行環境ディレクトリ /etc/docaccess.ini" に提供されています。

4.3.1 セキュリティ定義ファイルの概要

セキュリティ定義ファイルは，次に示すアクセス制御機能の運用に関する情報を定義するファイルです。

● **セキュリティ管理者**
DocumentBroker に登録されたオブジェクトやオブジェクトのアクセス権を保守するユーザです。

● **ユーザ権限定義ファイル名**
文書空間にオブジェクトを作成する権限や文書空間内のオブジェクトに対する操作の範囲を定義するために作成するファイルの名称です。ユーザ権限定義ファイルについては，「4.4　ユーザ権限定義ファイル」を参照してください。

● **デフォルトで設定されるパーミッション**
新規にオブジェクトを作成した場合に，ACFlag に設定するパーミッションです。

4.3.2 セキュリティ定義ファイルの記述形式

ここでは，セキュリティ定義ファイルの記述形式について説明します。

セキュリティ定義ファイルは，次に示す二つのセクションと，各セクションに指定するエントリによって構成されます。

● [Security] セクション

● [Entry0001] セクション

セクションとエントリの記述規則を次に示します。

● セクション名は，[　] （角括弧）で囲んで指定します。一つのセクションは，セクション名を指定してから，次のセクション名を指定するまで，またはファイルの終端までの範囲です。

● エントリは，「エントリ名 = 指定値」の形式で指定します。

●「;」（セミコロン）で始まる行はコメント行として扱われます。

● 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 コードで記述します。

記述形式を次に示します。

[< セクション名 >]
　　< エントリ名 >=< 値 >

以降，セキュリティ定義ファイルを構成する各セクションと，セクションごとに指定するエントリについて説明します。なお，エントリは，1 行 2,048 バイト以内で指定してください。

（1） [Security] セクション

[Security] セクションを構成するエントリは次のとおりです。

SecurityAdmin **エントリ**

DocumentBroker に登録されたオブジェクトやオブジェクトのアクセス権を保守するユーザ（**セキュリティ管理者**）を指定します。このエントリは，アクセス制御機能に対応するデータベーススキーマが構築されている場合，有効になります。セキュリティ管理者名には，ユーザ識別子を指定してください。ユーザ識別子は，「"」（引用符）で囲んで指定します。セキュリティ管理者を複数指定する場合は，「"」（引用符）で囲んだユーザ識別子を「,」（コンマ）で区切って指定します。セキュリティ管理者の人数に制限はありませんが，エントリの定義に従って 1 行 2,048 バイト以内で指定してください。セキュリティ管理者の指定例を次に示します。

セキュリティ管理者の指定例

SecurityAdmin=" **ユーザ識別子** 1"," **ユーザ識別子** 2"

ユーザ識別子については，「3.8.2　ユーザ情報の取得方法」を参照してください。なお，このエントリの指定は省略できません。

（2） [Entry0001] セクション

[Entry0001] セクションには，ユーザ権限定義ファイル名およびオブジェクトを新規に作成した場合に，そのオブジェクトに設定する ACFlag のデフォルト値を定義します。[Entry0001] セクションを構成する各エントリは次のとおりです。

UserPermDefFile エントリ

文書空間のすべてのオブジェクトに対しての権限をユーザ，またはグループ単位で定義するユーザ権限定義ファイルの名称を指定します。このエントリは，アクセス制御機能に対応するデータベーススキーマが構築されている場合，有効になります。ファイル名は，実行環境ディレクトリ /etc からの相対パス指定として解釈されます。指定を省略した場合，すべてのユーザがオブジェクト作成権限を持ち，オブジェクト操作権限は持たないと仮定されます。
パスは印刷可能な ASCII コードで指定してください。

DefaultACFlagOwner エントリ

オブジェクトの作成時にデフォルトで付加される ACFlag を指定します。オブジェクトの所有者に対するパーミッションを指定します。ACFlag は，表 4-9 および表 4-10 に示されるパーミッション文字列を「|」（ストローク）で結合して指定します。指定を省略した場合，フルコントロールが仮定されます。

DefaultACFlagGroup エントリ

オブジェクトの作成時にデフォルトで付加される ACFlag を指定します。グループに対するパーミッションを指定します。ACFlag は，表 4-9 および表 4-10 に示されるパーミッション文字列を「|」（ストローク）で結合して指定します。指定を省略した場合，フルコントロールが仮定されます。

DefaultACFlagEveryone エントリ

オブジェクトの作成時にデフォルトで付加される ACFlag を指定します。すべてのユーザに対するパーミッションを指定します。ACFlag は，表 4-9 および表 4-10 に示されるパーミッション文字列を「|」（ストローク）で結合して指定します。指定を省略した場合，フルコントロールが仮定されます。

DefaultACFlagOwner エントリ，DefaultACFlagGroup エントリ，および DefaultACFlagEveryone エントリに設定できるパーミッションには，**基本パーミッション**と**組み合わせパーミッション**があります。どちらのパーミッションも指定できます。また，この 2 種類の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。したがって，各エントリの値には，次に示す表のパーミッション文字列を指定します。基本パーミッションと対応する文字列を表 4-9 に示します。また，組み合わせパーミッションと対応する文字列を表 4-10 に示します。

表 4-9 基本パーミッションと対応する文字列

パーミッション文字列	パーミッション	意味
PRIM_READ_PROPS	基本プロパティ参照権	プロパティを参照する権限
PRIM_WRITE_PROPS	基本プロパティ更新権	プロパティを更新する権限
PRIM_READ_CONTENTS	基本コンテント参照権	文書の内容を参照する権限
PRIM_WRITE_CONTENTS	基本コンテント更新権	文書の内容を更新する権限
PRIM_LINK	基本リンク権	オブジェクト同士を関連づける権限
PRIM_VERSION	基本バージョン管理権	オブジェクトのバージョンに対する操作をする権限
PRIM_DELETE	基本削除権	オブジェクトを削除する権限

基本プロパティ参照権は，そのほかの基本パーミッションにも含まれます。したがって，基本プロパティ参照権以外の基本パーミッションを指定すると，各基本パーミッションで許可される操作に加えて，オブジェクトのプロパティの参照が許可されます。なお，基本パーミッションは，アクセス制御の対象となるオブジェクトごとに許可される操作の内容が異なります。詳細については，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」またはマニュアル「DocumentBroker Version 3　クラスライブラリ　Java　解説」を参照してください。

表 4-10 組み合わせパーミッションと対応する文字列

パーミッション文字列	パーミッション	意味
READ_PROPS	プロパティ参照権	基本プロパティ参照権と同じ権限
READ	参照権	次の基本パーミッションを組み合わせた権限 • 基本プロパティ参照権 • 基本コンテント参照権
WRITE_PROPS	プロパティ更新権	次の基本パーミッションを組み合わせた権限 • 基本プロパティ参照権 • 基本プロパティ更新権
READ_WRITE	参照更新権	次の基本パーミッションを組み合わせた権限 • 基本プロパティ参照権 • 基本プロパティ更新権 • 基本コンテント参照権 • 基本コンテント更新権
DELETE	削除権	次の基本パーミッションを組み合わせた権限 • 基本プロパティ参照権 • 基本削除権
LINK	リンク権	次の基本パーミッションを組み合わせた権限 • 基本プロパティ参照権 • 基本リンク権
VERSION	バージョン権	次の基本パーミッションを組み合わせた権限 • 基本プロパティ参照権 • 基本プロパティ更新権 • 基本コンテント参照権 • 基本コンテント更新権 • 基本バージョン管理権
FULL_CONTROL	フルコントロール	すべての基本パーミッションを組み合わせた権限

なお，表 4-9 および表 4-10 に示したパーミッションのほかに，「NONE」が指定できます。NONE を指定すると，新規にオブジェクトを作成した場合に権限は設定されません。また「NONE」は，ほかのパーミッション文字列と一緒には指定できません。例えば，「DefaultACFlagEveryone=NONE ｜ LINK」といった指定はできません。

4.3.3 セキュリティ定義ファイルの記述例

セキュリティ定義ファイルの記述例を例題に沿って説明します。

(1) セキュリティ定義ファイルで使用する例題

例題として使用するセキュリティ定義ファイルの定義内容を次の表に示します。

表 4-11 セキュリティ定義ファイルの定義内容

指定項目	指定する値	説明
セキュリティ管理者	"admin01"	セキュリティ管理者としてユーザ ID「admin01」のユーザを指定します。
ユーザ権限定義ファイル	userperm.ini	使用するユーザ権限定義ファイルのファイル名「userperm.ini」を指定します。
文書フォルダの所有者の権限	FULL_CONTROL	文書やフォルダの所有者に必ず与える権限として，すべての操作を許可します。
文書やフォルダのプライマリグループの権限	READ_WRITE|LINK|VERSION	文書やフォルダのプライマリグループに必ず与える権限として次の操作を許可します。 • オブジェクトのプロパティの参照と更新 • オブジェクト同士の関連づけ • オブジェクトのバージョンに対する操作 • 文書の内容の参照と更新
すべてのユーザの権限	NONE	すべてのユーザに必ず与える権限としては，何も権限を与えません。

(2) セキュリティ定義ファイルの記述例

セキュリティ定義ファイルの記述例を次に示します。

```
[Security]
SecurityAdmin = "admin01"

[Entry0001]
UserPermDefFile = userperm.ini
DefaultACFlagOwner = FULL_CONTROL
DefaultACFlagGroup = READ_WRITE|LINK|VERSION
DefaultACFlagEveryone = NONE
```

4.4 ユーザ権限定義ファイル

この節では，ユーザ権限定義ファイルについて説明します。

ユーザ権限定義ファイルには，サンプルファイル（実行環境ディレクトリ /etc/userperm.ini）が提供されています。DocumentBroker の運用形態に合わせてサンプルファイルを編集してください。また，ファイル名は，セキュリティ定義ファイルで定義したファイル名に変更してください。

4.4.1 ユーザ権限定義ファイルの概要

ユーザ権限定義ファイルは，**ユーザ権限**を定義するためのファイルです。ユーザ権限とは，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。セキュリティ定義ファイルに指定したユーザ権限定義ファイルに，ユーザ権限を定義します。

セキュリティ運用者は，このユーザ権限定義ファイルを更新することでユーザ権限を変更できます。

4.4.2 ユーザ権限定義ファイルの記述形式

ここでは，ユーザ権限定義ファイルの記述形式について説明します。

ユーザ権限定義ファイルは，[UserPermList] セクションとエントリによって構成されます。

セクションとエントリの記述規則を次に示します。

● 印刷可能な ASCII コードで記述します。

● セクション名は，[　] （角括弧）で囲んで指定します。

● エントリは，「(対象)：(対象の種別)：(権限を表す文字列)」の形式で指定します。

●「;」（セミコロン）で始まる行はコメント行として扱われます。

記述形式を次に示します。

[UserPermList]

Subject:SubjectType: **パーミッション文字列**

Subject

権限を与える対象となるユーザ，グループ，またはシステムサブジェクトを指定します。

SubjectType

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サブジェクトなのかを識別するための情報です。次に示す識別子のどれかを指定します。

- U

  Subject に指定した権限を与える対象が，ユーザであることを示します。
- G

  Subject に指定した権限を与える対象が，グループであることを示します。
- S

  Subject に指定した権限を与える対象が，システムサブジェクトであることを示します。なお，この値を指定する場合，Subject には「everyone」だけ指定できます。

パーミッション文字列

Subject に設定するパーミッションに対応する文字列を指定します。オブジェクト作成権限を表

すパーミッション文字列「CREATE」および基本パーミッションを表すパーミッション文字列を指定します。基本パーミッションを表すパーミッション文字列については，「4.3.2(2) [Entry0001] セクション」の表 4-9 を参照してください。パーミッション文字列は，「|」（ストローク）で結合して複数指定できます。

なお，[UserPermList] セクションのエントリの指定で，次に示す場合については，複数のエントリに指定されたパーミッション文字列の論理和のユーザ権限が与えられます。

- 同一ユーザや同一グループを複数指定して，それぞれにパーミッション文字列を指定した場合
- すべてのユーザ（everyone）と特定のユーザや，特定のユーザとそのユーザが所属するグループなど，同一ユーザが該当するサブジェクトを複数指定して，それぞれにパーミッション文字列を指定した場合

ユーザ権限は，ユーザ権限定義ファイルの [UserPermList] セクションに，ユーザおよびグループ単位でパーミッションを指定します。ただし，セキュリティ定義ファイルの UserPermDefFile エントリにユーザ権限定義ファイル名を指定した場合，このセクションの指定は省略できません。なお，このセクションにエントリが指定されなかった場合，すべてのユーザにユーザ権限が与えられません。

4.4.3 ユーザ権限定義ファイルの記述例

ユーザ権限定義ファイルの記述例を例題に沿って説明します。

(1) ユーザ権限定義ファイルで使用する例題

例題として使用するユーザ権限定義ファイルの定義内容を次の表に示します。

表 4-12 ユーザ権限定義ファイルの定義内容

指定項目	指定する値	説明
DocumentBroker を利用するすべてのユーザ (everyone) の権限	CREATE|PRIM_READ_PROPS	DocumentBroker を利用するすべてのユーザ (everyone) に対して，次の操作を許可します。 • オブジェクトの作成 • すべてのオブジェクトのプロパティの参照
ユーザ「User1」の権限	PRIM_READ_CONTENTS	「User1」というユーザに対して，次の操作を許可します。 • 文書のプロパティの参照 • 文書の内容の参照
グループ「Group1」の権限	PRIM_DELETE	「Group1」というグループに所属するユーザに対して，すべてのオブジェクトの削除を許可します。

(2) ユーザ権限定義ファイルの記述例

ユーザ権限定義ファイルの記述例を次に示します。

```
[UserPermList]
everyone:S:CREATE|PRIM_READ_PROPS
User1:U:PRIM_READ_CONTENTS
Group1:G:PRIM_DELETE
```

4.5 RD エリア構成定義ファイル

この節では，RD エリア構成定義ファイルについて説明します。

4.5.1 RD エリア構成定義ファイルの概要

RD エリア構成定義ファイルは，HiRDB で使用する RD エリアの構成を定義するファイルです。ユーザ用 RD エリアとユーザ LOB 用 RD エリアには，DocumentBroker の運用に合わせた RD エリアを確保する必要があります。ユーザ用 RD エリアおよびユーザ LOB 用 RD エリアについては，「2.8　データベース容量の見積もり」で算出した値に従って作成してください。

また，次に示す全文検索機能を使用する場合は，全文検索用の表や管理情報を格納する RD エリアを HiRDB に確保します。

- 検索タームを指定する全文検索
- 構造指定検索
- 概念検索
- 文字列型プロパティに対する全文検索

ここでは，全文検索機能を使用するために作成する RD エリアのサイズや，初期値などを記述した RD エリア構成定義ファイルについて説明します。

全文検索機能を使用する場合に必要な RD エリアは次の RD エリアです。

(1) 全文検索用の文書内容を格納するための RD エリア

- **RD エリアの種類**：SGMLTEXT データ格納用 RD エリア
- **必要数**：全文検索機能付き文書クラス数

全文検索機能付き文書クラスごとに一つの RD エリアが必要です。ほかの用途で使用する RD エリアとは共用できません。ページサイズは 8,192 バイト，1 セグメントは 1 ページで見積もる必要があります。

(2) 全文検索用のインデクスを格納するための RD エリア

- **RD エリアの種類**：n-gram インデクス情報格納用 RD エリア
- **必要数**：全文検索機能付き文書クラス数

全文検索機能付き文書クラスごとに一つの RD エリアが必要です。ほかの用途で使用する RD エリアとは共用できません。ページサイズは 8,192 バイト，1 セグメントは 1 ページで見積もる必要があります。また，セグメント数は 5,000 以上見積もる必要があります。

(3) 文字列型プロパティに対する全文検索用のインデクスを格納するための RD エリア

- **RD エリアの種類**：全文検索機能付き文字列型プロパティのインデクス格納用 RD エリア
- **必要数**：全文検索機能付き文字列型プロパティの設定数

文書のサブクラスに追加する全文検索機能付き文字列型プロパティごとに一つの RD エリアが必要です。ほかの用途で使用する RD エリアとは共用できません。ページサイズは 8,192 バイト，1 セグメントは 1 ページで見積もる必要があります。また，セグメント数は 5,000 以上見積もる必要があります。

(4) 表を格納するための RD エリア

- **RD エリアの種類**：ユーザ表用 RD エリア
- **必要数**：一つ以上

ほかの表を格納する RD エリアと共用できます。文書数に応じて見積もる必要があります。

4.5.2 RD エリア構成定義ファイルの記述形式

RD エリア構成定義ファイルの記述形式を次に示します。

- RD エリア構成定義ファイルは，HiRDB の create rdarea 文で記述します。create rdarea 文には，ページサイズ，セグメントサイズ，ファイル名，セグメント数などを定義します。
  RD エリアの作成，追加方法および create rdarea 文の記述形式については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。
- 印刷可能な ASCII コードで記述します。

4.5.3 RD エリア構成定義ファイルの記述例（全文検索用）

全文検索用の表を二つ使用する場合の RD エリア構成定義ファイルの記述例を次に示します。

```
# 全文検索用文書内容格納用RDエリアの定義 #
create rdarea SEARCHDB1 for LOB used by public
    server name bes01
    page 8192 characters
    storage control segment 1 pages
    file name "/opt/HiEDMS/HiRDB_P/DB/pdfsForDB01/rdlob02"
    initial 1000 segments ;
# 全文検索インデクス格納用RDエリアの定義 #
create rdarea SGMLINDEX1 for LOB used by public
    server name bes01
    page 8192 characters
    storage control segment 1 pages
    file name "/opt/HiEDMS/HiRDB_P/DB/pdfsForDB01/rdlob03"
    initial 6000 segments ;
# 全文検索用文書内容格納用RDエリアの定義 #
create rdarea SEARCHDB2 for LOB used by public
    server name bes01
    page 8192 characters
    storage control segment 1 pages
    file name "/opt/HiEDMS/HiRDB_P/DB/pdfsForDB01/rdlob04"
    initial 1000 segments ;
# 全文検索インデクス格納用RDエリアの定義 #
create rdarea SGMLINDEX2 for LOB used by public
    server name bes01
    page 8192 characters
    storage control segment 1 pages
    file name "/opt/HiEDMS/HiRDB_P/DB/pdfsForDB01/rdlob05"
    initial 6000 segments ;
# テーブル格納用RDエリアの定義 #
create rdarea USER01 for user used by public
    server name bes01
    page 16384 characters
    storage control segment 10 pages
    file name "/opt/HiEDMS/HiRDB_P/DB/pdfsForDB01/rdusr02"
    initial 50 segments  ;
```

4.6　メタ情報ファイル

この節では，メタ情報ファイルについて説明します。

DocumentBroker で使用するクラス（DMA で規定されているクラス）や検索に使用するオペレータなどについての定義情報を**メタ情報**といいます。このメタ情報は，**メタ情報ファイル**という初期設定用のファイルに記述されています。メタ情報ファイルには，例えば，クラスの詳細情報を定義する「ClassDescription」やクラスのプロパティの詳細情報を定義する「PropertyDescription」などの情報が記述されています。DocumentBroker サーバは，このメタ情報を基にデータベース定義文を出力します。

メタ情報ファイルは，DocumentBroker の実行環境を作成したときに，実行環境ディレクトリ /etc に格納されます。データベースを初期設定する場合には，このメタ情報ファイルをデータベースに登録します。同時に，実行環境ディレクトリ /etc/meta_files には，DocumentBroker が起動するときに参照する「**動作環境メタ情報ファイル**」が出力されます。

メタ情報ファイルは，直接編集しないでください。メタ情報を追加する場合は，定義情報ファイルに必要なオブジェクトの定義を追加します。定義情報ファイルについては，「4.7　定義情報ファイル」を参照してください。

また，次のような場合には，メタ情報の記述内容の確認コマンド（EDMChkMeta）を実行して，メタ情報ファイルの内容が記述方法に一致しているか，記述に矛盾がないかを確認してください。

- メタ情報ファイルにクラス定義を追加する場合
  メタ情報の追加コマンド（EDMAddMeta）が定義する範囲外のクラスを定義するときに，このコマンドを実行します。例えば，DocumentBroker Development Kit で開発したクライアントアプリケーションで使用するクラスを追加して定義する場合に相当します。
- メタ情報ファイルのクラス定義を変更する場合
  ユーザの環境から DocumentBroker のメタ情報ファイルを編集したり，メタ情報の追加コマンド（EDMAddMeta）が定義する範囲外のクラスの定義を変更したりするときに，このコマンドを実行します。

メタ情報の記述内容確認コマンド（EDMChkMeta）については，「7.3　コマンドの文法」の「EDMChkMeta（メタ情報の記述内容の確認）」を参照してください。

4.7　定義情報ファイル

この節では，定義情報ファイルについて説明します。

4.7.1　定義情報ファイルの概要

定義情報ファイルは，ユーザの環境に合わせてサブクラスやプロパティを追加する場合に作成する，オブジェクトを定義するファイルです。このファイルに定義した情報は，メタ情報として追加されます。

DocumentBroker が提供するクラスおよびプロパティ以外を使用する場合は，ユーザが追加するクラスおよび各クラスのプロパティを記述した定義情報ファイルを作成します。

なお，文書に対する全文検索機能を使用する場合は，全文検索用のプロパティを追加した dmaClass_DocVersion クラスのサブクラスを作成するために必要な情報を，定義情報ファイルに記述します。

また，文字列型プロパティに対する全文検索機能を使用する場合は，全文検索機能付き文字列型プロパティを追加したクラスを作成するために必要な情報を，定義情報ファイルに記述します。

4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則

定義情報ファイルの dmaProp_DisplayName プロパティに記述するサブクラス名やプロパティ名を，データベース定義の名称定義として使用する場合の規則を説明します。ここでは，表識別子（表の名称として一意に識別できる値），列名を定義する場合の規則について説明します。

なお，システムクラスおよびシステムプロパティのデータベース定義の名称一覧は，「付録 F　システムクラスおよびシステムプロパティの名称定義の規則」を参照してください。

(1) サブクラス名に対応する表識別子

サブクラス名に対応する表識別子を定義する場合，英小文字，英大文字，数字，空白，「_」（下線文字），および「-」（マイナス記号）で指定します。サブクラスの追加によってサブクラス名に対応する表識別子を指定する場合，次に示す規則およびデータベース定義の制約に従ってください。

- サブクラス名には，「dmaClass」および「edmClass」で始まる文字列は使用できません。
- サブクラス名には，「SystemDefinition」を使用できません。
- サブクラスの名称は重複しないようにしてください。
- RD エリア定義ファイルには，空白を含むクラス名，プロパティ名，RD エリア名を定義できません。そのため，空白を含むサブクラス名は指定できません。
- 表識別子の先頭 1 バイトは，英小文字，または英大文字を指定します。
- 表識別子の末尾に，空白は設定できません。
- 表識別子は，1 ～ 28 バイトで指定します。
- 表識別子は，「dmaClass」および「edmClass」で始まる名称を指定できません。
- 表識別子は，重複しないように付ける必要があります。VariableArray 型プロパティの要素を別表へ格納するように定義している場合，VariableArray 型プロパティに対応する表識別子も重複しないように付ける必要があります。

(2) プロパティ名に対応する列名

プロパティ名に対応する列名を定義する場合，英小文字，英大文字，数字，空白，「_」（下線文字），およ

び「-」（マイナス記号）で指定します。プロパティの追加によってプロパティ名に対応する列名を指定する場合は，次に示す規則およびデータベース定義の制約に従ってください。

- 新規に追加するプロパティ名には，「dmaProp」および「edmProp」で始まる文字列は使用できません。
- RD エリア定義ファイルには，空白を含むクラス名，プロパティ名，RD エリア名を定義できません。そのため，空白を含むプロパティ名は指定できません。
- 列名の先頭 1 バイトは，英小文字か英大文字で指定します。
- 列名の末尾に，空白は設定できません。
- VariableArray 型のプロパティの列名は，1 ～ 14 バイトで指定します。
  VariableArray 型のプロパティは，dmaProp_DataType の値が DMA_DATATYPE_OBJECT であり，かつ dmaProp_Cardinality の値が 255 のプロパティである必要があります。
- VariableArray 型の要素のプロパティの列名は，1 ～ 15 バイトで指定します。
  VariableArray 型の要素のプロパティとは，edmClass_Struct クラスのサブクラスに追加したプロパティです。
- 列名には，「dmaProp」および「edmProp」で始まる名称を指定できません。
- 列名は重複しないように付ける必要があります。
  列名は，プリフィクスが「VariableArray 型プロパティの列名 +'_'」と重複しないように付ける必要があります。例えば列名「varray_elem」と VariableArray 型のプロパティの列名「varray」がある場合，両方ともプリフィクス「varray_」が重複するため，片方に別の名称を付ける必要があります。
- 上記以外の列名は 1 ～ 28 バイトで指定します。

4.7.3 定義情報ファイルの記述形式

ここでは，定義情報ファイルの記述形式について説明します。

（1） 記述規則

定義情報ファイルを記述する場合の規則を次に示します。

● 印刷可能な ASCII コードで記述します。

●「;」（セミコロン）および「#」（シャープ）で始まる行はコメント行として扱われます。

● 各行は 399 バイト以内で記述します。

● 行の終わりは，<EOF> または <LF> です。

● GUID 値はオペレーティングシステムの機能を使用して取得してください。

（2） 記述形式

定義情報ファイルは次の形式に従って記述します。

```
[Action]
 PropertyName=DataType=Value
```

各項目について説明します。

Action（アクション）

定義するオブジェクトごとにセクション名として [　] 内に記述します。指定できるアクションは次のとおりです。

Action::= "AddSubClass" | "AddProperty/ **クラス名** " | "AssumeProperty/ **データタイプ** "

- AddSubClass
  サブクラスを追加することを示します。
- AddProperty/ クラス名
  指定したクラスに対してプロパティを追加することを示します。
- AssumeProperty/ データタイプ
  AddProperty で記述を省略した場合の仮定値の記述をデータタイプごとに設定します。
  データタイプ ::= " DMA_DATATYPE_BOOLEAN "
  | "DMA_DATATYPE_INTEGER32"
  | "DMA_DATATYPE_STRING"
  | "DMA_DATATYPE_OBJECT_VARIABLE_ARRAY"
  プロパティは，データ型に従った値を 1 個持つか複数個持つかが決められています。これを基本単位といいます。プロパティの基本単位が VariableArray 型，かつデータ型が Object 型（DMA_DATATYPE_OBJECT）であるプロパティは AssumeProperty/ DMA_DATATYPE_OBJECT_VARIABLE_ARRAY を仮定値とします。プロパティの基本単位が VariableArray 型でなければ，AddProperty の dmaProp_DataType プロパティで指定するデータタイプと AssumeProperty/ データタイプのデータタイプが同じ場合に AssumeProperty/ データタイプを仮定値とします。

PropertyName（プロパティ名）

DMA で定義されているプロパティ名またはユーザが定義したプロパティ名を記述します。128 バイト以内で記述してください。

DataType（データ型）と Value（値）

プロパティのデータ型を記述します。データ型は次のどれかです。
DataType::= "bool" | "obj" | "text" | "guid" | "int"
各データ型と値の記述方法について説明します。なお，値は 256 バイト以内で記述してください。

bool

プロパティが Boolean 型の値を持つことを示します。

＜記述形式＞

```
PropertyName=bool=0|1
```

＜記述例＞

```
dmaProp_IsSelectable=bool=1
```

obj

プロパティが Object 型の値を持つことを示します。

＜記述形式＞

```
PropertyName=obj=SectionName|NULL
```

- SectionName
  同一ファイル，またはほかのメタ情報ファイル中のセクション名を指定します。ここに指定したセクションに，オブジェクトの初期化データに相当する内容を記述しておく必要があります。

＜記述例＞

```
dmaProp_RequiredClass=obj=usrClass_Struct
```

この例は，VariableArray 型のプロパティの要素に「usrClass_Struct」を選択することを示しています。

```
=obj=dsqop.ini@dmaQueryOperator_And
```

この例は，dmaProp_QueryOperatorDescriptions プロパティのリストの要素に選ぶ検索オペレータの記述です。メタ情報ファイル名とセクション名は「@」（単価記号）で区切ります。

text

プロパティが String 型の値を持つことを示します。

＜記述形式＞

```
PropertyName=text=dmaString|NULL
```

- dmaString
  任意の文字列を指定します。

＜記述例＞

```
dmaProp_DisplayName=text=Docspace Mumble
```

guid

プロパティが ID 型の値を持つことを示します。具体的には，プロパティの値として GUID を持つことを示します。GUID とは DMA のオブジェクトに与えるユニークな識別子です。

＜記述形式＞

```
PropertyName=guid=guid|dmaNameCorrespondingToGuid|空値
```

- guid
  オペレーティングシステムの機能を利用して取得した GUID の値を指定します。GUID は次の形式に従って，36 バイトで指定してください。
  GUID の形式：XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
  「X」は 16 進数値文字を示します。「-」はマイナス記号です。
- dmaNameCorrespondingToGuid
  GUID が定義されているプロパティ名を指定します。
- 空値
  何も指定しないことを意味します。空値を指定した場合は，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを必ず指定してください。メタ情報の追加コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。

＜記述例＞

```
dmaProp_DocSpaceId=guid=673D2BE0-D1FD-11d0-AB59-08002BE29E1D
```

int

プロパティが Integer32 型の値を持つことを示します。

＜記述形式＞

```
PropertyName=int=digitStringBase10
```

- digitStringBase10
  任意の数値（10 進数）を -2,147,483,648 ～ 2,147,483,647 で指定します。

＜記述例＞

```
dmaProp_OperandDataType=int=3
```

4.7.4　定義情報ファイルの記述例

サブクラスおよびプロパティを追加する場合の定義情報ファイルについて，例題に沿って説明します。

(1) 使用する例題

サブクラスおよびプロパティを追加する場合の定義情報ファイルを作成するための例題を次に示します。ここで使用する例題は，「2.3.5　クラスおよびプロパティの追加例」で示した例題と同じです。

(例)

「問題点管理ドキュメント」と「設計ドキュメント」という 2 種類の文書を管理するためにサブクラスを追加します。追加するサブクラスには，それぞれ文書の管理に必要なプロパティを追加します。管理する文書の詳細情報を次の表に示します。

表 4-13　管理する文書の詳細情報

文書の種類	概要	付加情報	管理方法
問題点管理ドキュメント	製品の開発時の問題点を管理するための文書	• 問題点分類 • 問題点発生日時 • 検討結果 • 解決日時 • 対策日	直接型のコンテインメントを使用して管理する。
設計ドキュメント	製品開発の基となる設計書	• 文書名 • 執筆者 • 特許情報 • 執筆日時	構成管理型のコンテインメントを使用して，複数の設計書の構成を管理する。

この 2 種類の文書を管理するためのサブクラスを次のように設計します。追加するサブクラスとそのスーパークラスを次の表に示します。

表 4-14　追加するサブクラスとそのスーパークラス

管理する文書	追加するサブクラス	スーパークラス
問題点管理ドキュメント	usrClass_ProbInfoDoc	dmaClass_DocVersion
設計ドキュメント	usrClass_DesignDoc	edmClass_VersionTracedDocVersion

問題点管理ドキュメントは，直接型のコンテインメントを使用して管理します。したがって，dmaClass_DocVersion クラスのサブクラスとして usrClass_ProbInfoDoc クラスを追加して管理します。

設計ドキュメントは，バージョン付きコンテナを使用して構成を管理する文書として作成します。したがって，edmClass_VersionTracedDocVersion クラスのサブクラスとして usrClass_DesignDoc クラスを追加して管理します。

次に，文書に対して付加する情報を各クラスのプロパティとして追加します。文書の付加情報とクラスに追加するプロパティの対応を，次の表に示します。

表 4-15　文書の付加情報とサブクラスに追加するプロパティの対応

サブクラス名	付加情報	プロパティ名	データ型
usrClass_ProbInfoDoc	問題点分類	usrProp_Category	String 型
	問題点発生日時	usrProp_DateOfProb	Integer32 型
	検討結果	usrProp_Status	Boolean 型
	解決日時	usrProp_DateOfResolve	Integer32 型

サブクラス名	付加情報	プロパティ名	データ型
usrClass_DesignDoc	文書名	usrProp_DocName	String 型
	執筆者	usrProp_DocWriter	String 型
	特許情報	usrProp_PatentInfo	String 型
	執筆日時	usrProp_DateOfWrite	Integer32 型

(2) 定義情報ファイルの記述例

ここでは，例題を基にサブクラスおよびプロパティを追加するための定義情報ファイルの記述例を示します。なお，定義情報ファイルの記述例を二つに分けていますが，実際は，一つのファイルとして作成します。

(a) 定義情報ファイルを記述する場合の注意事項

サブクラスおよびプロパティを追加するために，定義情報ファイルを記述する場合の注意事項について説明します。

- [AddProperty/ クラス名] の前に記述した [AssumeProperty/ データタイプ] が仮定値と解釈されます。
- 定義済みのプロパティをクラスに追加する場合は，[AddProperty/ クラス名] の dmaProp_DisplayName プロパティに定義済みのプロパティと同じ値を指定するか，または dmaProp_DisplayName プロパティと dmaProp_DataType プロパティに定義済みのプロパティと同じ値を指定してください。
- サブクラスに追加するプロパティをキーにして検索したい場合，プロパティにインデクスを定義する必要があります。プロパティにインデクスを定義する場合は，インデクス情報ファイルを作成してください。なお，インデクス情報ファイルでは，NULL 値の除外を指定できます。NULL 値の重複が多い場合は，NULL 値の除外を指定することでインデクス容量を削減できます。インデクス情報ファイルの詳細については，「4.9　インデクス情報ファイル」を参照してください。

(b) 追加するプロパティのデータ型に対応する仮定値の定義

サブクラスに追加するプロパティのデータ型ごとに，プロパティの詳細情報（プロパティディスクリプション）を定義する必要があります。ここでは，データ型ごとで共通に指定するプロパティの仮定値を記述します。

例題で追加するプロパティは，String 型，Integer32 型および Boolean 型なので，それぞれのデータ型に対応する仮定値を記述します。記述例を次に示します。

```
;
; DMA_DATATYPE_STRINGの仮定値
;
[AssumeProperty/DMA_DATATYPE_STRING]
dmaProp_QueryOperatorDescriptions
=obj=dsqop.ini@dmaQueryOperator_And
=obj=dsqop.ini@dmaQueryOperator_Or
=obj=dsqop.ini@dmaQueryOperator_Not
=obj=dsqop.ini@dmaQueryOperator_IsNull
=obj=dsqop.ini@dmaQueryOperator_Exists
=obj=dsqop.ini@dmaJoinOperator_Cross
=obj=dsqop.ini@dmaJoinOperator_Inner
=obj=dsqop.ini@dmaQueryOperator_EqualString
=obj=dsqop.ini@dmaQueryOperator_UnequalString
=obj=dsqop.ini@dmaQueryOperator_GreaterString
=obj=dsqop.ini@dmaQueryOperator_GreaterOrEqualString
=obj=dsqop.ini@dmaQueryOperator_LessString
=obj=dsqop.ini@dmaQueryOperator_LessOrEqualString
```

```
=obj=dsqop.ini@dmaQueryOperator_Like
=obj=dsqop.ini@dmaQueryOperator_InString
=obj=dsqop.ini@edmQueryOperator_Xlike

dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE
dmaProp_IsSelectable=bool=1
dmaProp_IsSearchable=bool=1
dmaProp_IsOrderable=bool=1
dmaProp_IsValueRequired=bool=0
dmaProp_PropertyDefaultString=text=NULL
dmaProp_MaximumLengthString=int=64

;
; DMA_DATATYPE_INTEGER32の仮定値
;
[AssumeProperty/DMA_DATATYPE_INTEGER32]
dmaProp_QueryOperatorDescriptions
=obj=dsqop.ini@dmaQueryOperator_And
=obj=dsqop.ini@dmaQueryOperator_Or
=obj=dsqop.ini@dmaQueryOperator_Not
=obj=dsqop.ini@dmaQueryOperator_IsNull
=obj=dsqop.ini@dmaQueryOperator_Exists
=obj=dsqop.ini@dmaJoinOperator_Cross
=obj=dsqop.ini@dmaJoinOperator_Inner
=obj=dsqop.ini@dmaQueryOperator_EqualInteger32
=obj=dsqop.ini@dmaQueryOperator_UnequalInteger32
=obj=dsqop.ini@dmaQueryOperator_GreaterInteger32
=obj=dsqop.ini@dmaQueryOperator_GreaterOrEqualInteger32
=obj=dsqop.ini@dmaQueryOperator_LessInteger32
=obj=dsqop.ini@dmaQueryOperator_LessOrEqualInteger32
=obj=dsqop.ini@dmaQueryOperator_AddInteger32
=obj=dsqop.ini@dmaQueryOperator_SubtractInteger32
=obj=dsqop.ini@dmaQueryOperator_NegateInteger32
=obj=dsqop.ini@dmaQueryOperator_AbsoluteValueInteger32
=obj=dsqop.ini@dmaQueryOperator_MultiplyInteger32
=obj=dsqop.ini@dmaQueryOperator_DivideInteger32
=obj=dsqop.ini@dmaQueryOperator_InInteger32

dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE
dmaProp_IsSelectable=bool=1
dmaProp_IsSearchable=bool=1
dmaProp_IsOrderable=bool=1
dmaProp_IsValueRequired=bool=0
dmaProp_PropertyDefaultInteger32=int=0

;
; DMA_DATATYPE_BOOLEANの仮定値
;
[AssumeProperty/DMA_DATATYPE_BOOLEAN]
dmaProp_QueryOperatorDescriptions
=obj=dsqop.ini@dmaQueryOperator_And
=obj=dsqop.ini@dmaQueryOperator_Or
=obj=dsqop.ini@dmaQueryOperator_Not
=obj=dsqop.ini@dmaQueryOperator_IsNull
=obj=dsqop.ini@dmaQueryOperator_Exists
=obj=dsqop.ini@dmaJoinOperator_Cross
=obj=dsqop.ini@dmaJoinOperator_Inner
=obj=dsqop.ini@dmaQueryOperator_InBoolean
=obj=dsqop.ini@dmaQueryOperator_EqualBoolean
=obj=dsqop.ini@dmaQueryOperator_UnequalBoolean

dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE
dmaProp_IsSelectable=bool=1
dmaProp_IsSearchable=bool=1
dmaProp_IsOrderable=bool=1
dmaProp_IsValueRequired=bool=0
```

```
dmaProp_PropertyDefaultBoolean=bool=0
```

記述例について説明します。

アクション名の説明

データ型に対応して，次の三つのアクション名が指定されています。アクション名に指定する「データタイプ」には，追加するプロパティのデータ型を示す文字列を指定します。

- [AssumeProperty/DMA_DATATYPE_STRING]
  String 型のプロパティを追加する場合は，「データタイプ」として「DMA_DATATYPE_STRING」と指定します。
- [AssumeProperty/DMA_DATATYPE_INTEGER32]
  Integer32 型のプロパティを追加する場合は，「データタイプ」として「DMA_DATATYPE_INTEGER32」と指定します。
- [AssumeProperty/DMA_DATATYPE_BOOLEAN]
  Boolean 型のプロパティを追加する場合は，「データタイプ」として「DMA_DATATYPE_BOOLEAN」と指定します。

プロパティの仮定値

各アクション名に続くエントリには，[AddProperty/ クラス名] で記述を省略するプロパティの仮定値を記述します。各プロパティの記述は省略することもできます。ただし，各プロパティが [AddProperty/ クラス名] に続くエントリに指定されている場合は，その値が優先されます。ここで仮定値を指定できるプロパティは次のとおりです。

- dmaProp_QueryOperatorDescriptions プロパティ
  追加するプロパティに対して使用することを許可する検索オペレータを，メタ情報ファイル「edms.ini」の [dmaClass_Scope/dmaProp_Operators] セクションに指定されているリストの要素の中から選択して指定します。指定を省略した場合，空のリストが仮定されます。
- dmaProp_Cardinality プロパティ
  追加するプロパティの基本単位を表す Integer32 型の値です。ユーザが追加できるプロパティは，基本単位が Scalar 型および VariableArray 型のプロパティです。追加するプロパティの基本単位が Scalar 型の場合は，「DMA_CARDINALITY_SINGLE」を指定します。追加するプロパティの基本単位が VariableArray 型の場合は「255」を指定します。VariableArray 型のプロパティを追加するための定義情報ファイルの作成方法については，「4.7.5　定義情報ファイルの記述例 (VariableArray 型のプロパティを追加する場合)」を参照してください。
- dmaProp_IsSelectable プロパティ
  追加するプロパティが検索結果集合の一項目として選択できるかどうかを指定する Boolean 型の値です。値として「1」を指定した場合は，このプロパティが検索結果集合の一項目として選択できるプロパティであることを示します。値として「0」を指定した場合は，このプロパティが検索結果集合の一項目として選択できないプロパティであることを示します。指定を省略した場合，「0」が仮定されます。
- dmaProp_IsSearchable プロパティ
  追加するプロパティが検索の条件として使用できるかどうかを指定する Boolean 型の値です。値として「1」を指定した場合は，このプロパティが検索の条件として使用できるプロパティであることを示します。値として「0」を指定した場合は，このプロパティが検索の条件として使用できないプロパティであることを示します。指定を省略した場合，「0」が仮定されます。
- dmaProp_IsOrderable プロパティ
  追加するプロパティが検索結果をソートする基準として使用できるプロパティであるかどうかを示す Boolean 型の値です。値として「1」を指定した場合は，検索結果をソートする基準として使用できるプロパティであることを示します。値として「0」を指定した場合は，検索結果をソートする

基準として使用できないプロパティであることを示します。指定を省略した場合，「0」が仮定されます。

- dmaProp_IsValueRequired プロパティ
  追加するプロパティが値を保持する必要があるかどうかを示す Boolean 型の値です。値として「1」を指定した場合は，追加するプロパティは値を保持する必要のあるプロパティであることを示します。値として「0」を指定した場合は，追加するプロパティは値を保持しなくてもよいプロパティであることを示します。定義するプロパティの基本単位が List 型である場合，このプロパティの値に「1」を指定すると，リストが必ず一つの要素を含むことを示します。指定を省略した場合，「0」が仮定されます。
- dmaProp_IsHidden プロパティ
  追加するプロパティが，システム管理者だけに表示するプロパティであるかどうかを示す Boolean 型の値です（ただし，この例題では指定を省略しています)。値として「1」を指定した場合は，システム管理者だけに公開されるプロパティとなります。値として「0」を指定した場合は，アクセスをシステム管理者だけに制限しないプロパティとなります。指定を省略した場合，「0」が仮定されます。なお，DocumentBroker は，このプロパティの値によるプロパティへのアクセスを制御しません。
- dmaProp_PropertyDefaultString プロパティ
  追加するプロパティにデフォルトで設定される String 型の値です。255 バイト以下の値を指定できます。このプロパティは，dmaProp_DataType プロパティが DMA_DATATYPE_STRING の場合だけ指定できます。指定を省略した場合，長さ 0（バイト）の文字列が仮定されます。
- dmaProp_PropertyDefaultInteger32 プロパティ
  追加するプロパティにデフォルトで設定される Integer32 型の値です。Integer32 型の値を指定できます。このプロパティは，dmaProp_DataType プロパティが DMA_DATATYPE_INTEGER32 の場合だけ指定できます。指定を省略した場合，「0」が仮定されます。
- dmaProp_PropertyDefaultBoolean プロパティ
  追加するプロパティにデフォルトで設定される Boolean 型の値です。「0」または「1」を指定できます。このプロパティは，dmaProp_DataType プロパティが DMA_DATATYPE_BOOLEAN の場合だけ指定できます。指定を省略した場合，「0」が仮定されます。
- dmaProp_MaximumLengthString プロパティ
  追加するプロパティのデータ型が String 型の場合，プロパティの値（NULL 文字以外）の最大長です。文字列の長さを 1 ～ 32,000 で表します。指定を省略した場合，「1」が仮定されます。なお，このプロパティの値は，追加するプロパティのデータ型が String 型（dmaProp_DataType プロパティの値が「DMA_DATATYPE_STRING」）である [AddProperty] の仮定として有効になります。

(c) サブクラスの追加とプロパティの追加に関する定義

追加するサブクラスの定義とそのサブクラスに追加するプロパティの定義について記述します。記述例を次に示します。

```
;
; 問題点管理ドキュメント用のサブクラスの定義
;
[AddSubClass]
dmaProp_DisplayName=text=dmaClass_DocVersion/usrClass_ProbInfoDoc

;
; 問題点分類（usrProp_Categoryプロパティの定義）
;
[AddProperty/usrClass_ProbInfoDoc]
dmaProp_DisplayName=text=usrProp_Category
dmaProp_DataType=int=DMA_DATATYPE_STRING
```

```
;
; 問題点発生日時（usrProp_DateOfProbプロパティの定義）
;
[AddProperty/usrClass_ProbInfoDoc]
dmaProp_DisplayName=text=usrProp_DateOfProb
dmaProp_DataType=int=DMA_DATATYPE_INTEGER32

;
; 検討結果（usrProp_Statusプロパティの定義）
;
[AddProperty/usrClass_ProbInfoDoc]
dmaProp_DisplayName=text=usrProp_Status
dmaProp_DataType=int=DMA_DATATYPE_BOOLEAN

;
; 解決日時（usrProp_DateOfResolveプロパティの定義）
;
[AddProperty/usrClass_ProbInfoDoc]
dmaProp_DisplayName=text=usrProp_DateOfResolve
dmaProp_DataType=int=DMA_DATATYPE_INTEGER32

########

;
; 設計ドキュメント用のサブクラスの定義
;
[AddSubClass]
dmaProp_DisplayName=text=edmClass_VersionTracedDocVersion/usrClass_DesignDoc

;
; 文書名（usrProp_DocNameプロパティの定義）
;
[AddProperty/usrClass_DesignDoc]
dmaProp_DisplayName=text=usrProp_DocName
dmaProp_DataType=int=DMA_DATATYPE_STRING

;
; 執筆者（usrProp_DocWriterプロパティの定義）
;
[AddProperty/usrClass_DesignDoc]
dmaProp_DisplayName=text=usrProp_DocWriter
dmaProp_DataType=int=DMA_DATATYPE_STRING

;
; 特許情報（usrProp_PatentInfoプロパティの定義）
;
[AddProperty/usrClass_DesignDoc]
dmaProp_DisplayName=text=usrProp_PatentInfo
dmaProp_DataType=int=DMA_DATATYPE_STRING

;
; 執筆日時（usrProp_DateOfWriteプロパティの定義）
;
[AddProperty/usrClass_DesignDoc]
dmaProp_DisplayName=text=usrProp_DateOfWrite
dmaProp_DataType=int=DMA_DATATYPE_INTEGER32
```

記述例について説明します。

サブクラスを追加するためのアクション名とプロパティの説明

サブクラスを追加するための定義情報には，アクション名として [AddSubClass] を指定します。アクション名に続くエントリには，次に示すプロパティとその値を指定します。

- dmaProp_DisplayName プロパティ
  追加するサブクラス名を値として指定します。サブクラスを追加するスーパークラス名のあとに

「/」（スラント）で区切ってサブクラス名を指定します。
この例題では，問題点管理ドキュメントを管理するためのサブクラスは，dmaClass_DocVersion クラスのサブクラスとして定義します。したがって，このプロパティの値には「dmaClass_DocVersion/usrClass_ProbInfoDoc」を指定しています。また，設計ドキュメントを管理するためのサブクラスは，edmClass_VersionTracedDocVersion クラスのサブクラスとして定義します。したがって，このプロパティの値には「edmClass_VersionTracedDocVersion/usrClass_DesignDoc」を指定しています。

- dmaProp_DescriptiveText プロパティ
  追加するサブクラスを簡単に説明する文字列を値として指定します。省略した場合，dmaProp_DisplayName プロパティに指定した値から「スーパークラス名 /」を除いたサブクラス名（ここでは，usrClass_ProbInfoDoc）が仮定されます。なお，この例題では指定を省略しています。
- dmaProp_Ids プロパティ
  追加するサブクラスに付与する GUID を値として指定します。指定を省略した場合，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを必ず指定してください。なお，この例題では指定を省略しています。

サブクラスにプロパティを追加するためのアクション名とプロパティの説明

プロパティを追加する場合は，アクション名 [AddProperty/ クラス名] とそれに続くエントリとしてプロパティ自身のプロパティを記述します。アクション名に続くエントリには，次に示すプロパティとその値を指定できます。

[AddProperty/ クラス名] に続くエントリとして指定するプロパティ

ほかのクラスに同じプロパティが定義されていない場合，次のプロパティは [AddProperty/ クラス名] に続くエントリとして指定してください。これらのプロパティは，[AssumeProperty/ データタイプ] には指定できません。指定するプロパティとその値について次に示します。

- dmaProp_DisplayName プロパティ
  追加するプロパティ名を値として指定します。この値の指定は省略できません。なお，この例題では表 4-15 に示したプロパティ名を指定しています。
- dmaProp_DescriptiveText プロパティ
  追加するプロパティを簡単に説明する文字列を値として指定します。指定を省略した場合，dmaProp_DisplayName プロパティに指定したプロパティ名が仮定されます。なお，この例題では指定を省略しています。
- dmaProp_Ids プロパティ
  追加するプロパティに付与する GUID を値として指定します。指定を省略した場合，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを必ず指定してください。なお，この例題では指定を省略しています。
- dmaProp_DataType プロパティ
  追加するプロパティのデータ型を示す文字列を指定します。なお，この文字列の指定は，省略できません。データ型とデータ型を示す文字列の対応を次の表に示します。

表 4-16　データ型とデータ型を示す文字列の対応

データ型	データ型を示す文字列
String 型	DMA_DATATYPE_STRING
Integer32 型	DMA_DATATYPE_INTEGER32
Boolean 型	DMA_DATATYPE_BOOLEAN

[AddProperty/ クラス名] に続くエントリとして指定するプロパティ（複数のクラスに同じ名前のプロパティを追加し，クラスごとにデフォルト値を定義する場合）

複数のクラスに同じ名前のプロパティを追加する場合は，プロパティのデフォルト値をクラスごとに設定できます。クラスごとにデフォルト値を定義するには，次に示すプロパティを [AddProperty/ クラス名] に続くエントリとして指定してください。

- dmaProp_PropertyDefaultString プロパティ
  追加するプロパティにデフォルトで設定される String 型の値です。255 バイト以下の値を指定できます。このプロパティは，dmaProp_DataType プロパティが DMA_DATATYPE_STRING の場合だけ指定できます。指定を省略した場合，長さ 0（バイト）の文字列が仮定されます。
- dmaProp_PropertyDefaultInteger32 プロパティ
  追加するプロパティにデフォルトで設定される Integer32 型の値です。Integer32 型の値を指定できます。このプロパティは，dmaProp_DataType プロパティが DMA_DATATYPE_INTEGER32 の場合だけ指定できます。指定を省略した場合，「0」が仮定されます。
- dmaProp_PropertyDefaultBoolean プロパティ
  追加するプロパティにデフォルトで設定される Boolean 型の値です。「0」または「1」を指定できます。このプロパティは，dmaProp_DataType プロパティが DMA_DATATYPE_BOOLEAN の場合だけ指定できます。指定を省略した場合，「0」が仮定されます。

クラスごとのプロパティのデフォルト値の設定を有効にするには，-D オプションを指定してメタ情報の追加コマンド（EDMAddMeta）を実行します。-D オプションを省略すると，データ型ごとのプロパティのデフォルト値の設定が有効になります。メタ情報の追加コマンド（EDMAddMeta）については，「7.3　コマンドの文法」の「EDMAddMeta（メタ情報の追加）」を参照してください。

[AssumeProperty/ データタイプ] にも指定できるプロパティ

次に示すプロパティは，[AssumeProperty/ データタイプ] に続くエントリとしても指定できます。ただし，[AddProperty/ クラス名] に続くエントリに同じプロパティが指定されている場合は，[AddProperty/ クラス名] に指定されている値を優先します。[AddProperty/ クラス名] の値を省略した場合で，かつ [AssumeProperty/ データタイプ] の値も省略した場合は，各プロパティでの省略値が仮定されます。

- dmaProp_Cardinality プロパティ
- dmaProp_IsSelectable プロパティ
- dmaProp_IsSearchable プロパティ
- dmaProp_IsOrderable プロパティ
- dmaProp_QueryOperatorDescriptions プロパティ
- dmaProp_IsValueRequired プロパティ
- dmaProp_IsHidden プロパティ
- dmaProp_MaximumLengthString プロパティ

なお，この例題ではこれらのプロパティとその値を [AssumeProperty/ データタイプ] のエントリとして指定しています。すなわち，すでに仮定値は指定されています。

4.7.5 定義情報ファイルの記述例（VariableArray 型のプロパティを追加する場合）

プロパティの基本単位が VariableArray 型であるプロパティを追加する場合の定義情報ファイルについて，例題に沿って説明します。

（1） 使用する例題

プロパティは，データ型に従った値を 1 個持つか複数個持つかが決められています。これを基本単位といいます。プロパティの基本単位が VariableArray 型のプロパティとは，そのプロパティが配列の要素を保持するプロパティです。

VariableArray 型のプロパティとして定義するプロパティは，edmClass_Struct クラスのインスタンスをリファレンスとして保持します。VariableArray 型のプロパティを定義するための例題を次に示します。

（例）

管理する文書として「文書リスト」を定義します。この文書は，書籍の著者に関する情報を管理します。この「文書リスト」について，サブクラスを追加して必要な付加情報を追加します。追加するサブクラスとそのスーパークラスを次の表に示します。

表 4-17 追加するサブクラスとそのスーパークラス

管理する情報	追加するサブクラス	スーパークラス
文書リスト	usrClass_DocList	edmClass_IndependentPersistence
著者の情報	usrClass_WriterInfo	edmClass_Struct

「文書リスト」は，書籍の著者に関する情報を管理する文書で，edmClass_IndependentPersistence クラスのサブクラスとして，「usrClass_DocList クラス」を定義します。「著者の情報」は，著者の情報を管理する文書であり，edmClass_Struct クラスのサブクラスとして「usrClass_WriterInfo クラス」を定義します。

「文書リスト」には付加情報としてプロパティを追加します。付加情報に対応するプロパティを次の表に示します。

表 4-18 usrClass_DocList クラスの構造

クラス名	付加情報	プロパティ名（VariableArray 型のプロパティ）	データ型
usrClass_DocList	文書名	uP_DocName	String 型
	著者の情報	uP_WriterInfo	Object 型

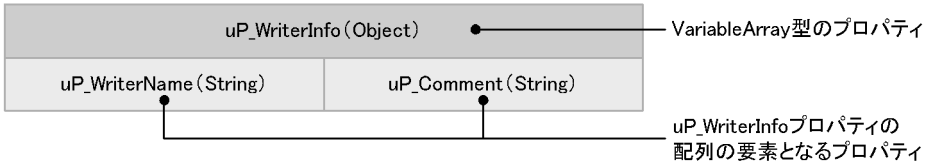
「uP_WriterInfo プロパティ」は，VariableArray 型のプロパティです。表 4-19 に，「uP_WriterInfo プロパティ」に定義するプロパティを示します。「uP_WriterInfo プロパティ」は，配列の要素としてプロパティを管理します。VariableArray 型の「uP_WriterInfo プロパティ」の要素を管理する「usrClass_WriterInfo クラス」に格納されているプロパティを説明します。

表 4-19 VariableArray 型の「uP_WriterInfo プロパティ」の要素となるクラスの構造

クラス名（VariableArray 型のプロパティの要素）	付加情報	プロパティ名（VariableArray 型のプロパティ要素の構成）	データ型
usrClass_WriterInfo	著者名	uP_WriterName	String 型
	著者の紹介	uP_Comment	String 型

VariableArray 型の「uP_WriterInfo プロパティ」の構造を次の図に示します。

図 4-2 VariableArray 型の「uP_WriterInfo プロパティ」の構造



配列の要素として管理されるこれらのプロパティは，edmClass_Struct クラスのプロパティとして追加します。

プロパティの基本単位が VariableArray 型であるプロパティを追加できるクラスは次のとおりです。

- edmClass_IndependentPersistence クラスのサブクラス
- dmaClass_ConfigurationHistory クラスのサブクラス
- dmaClass_Container クラスのサブクラス
- edmClass_ContainerVersion クラスのサブクラス
- edmClass_ComponentDocVersion クラスのサブクラス
- dmaClass_DocVersion クラスのサブクラス
- edmClass_VersionTracedComponentDocVersion クラスのサブクラス
- edmClass_VersionTracedDocVersion クラスのサブクラス

（2） VariableArray 型のプロパティの追加方法

VariableArray 型のプロパティを追加する場合，プロパティを追加するために必要な情報を定義情報ファイルに記述する必要があります。プロパティを追加するために必要な定義情報ファイルの記述方法については，「4.7.4 定義情報ファイルの記述例」を参照してください。ただし，定義情報ファイルに記述する内容に異なる部分がありますので注意してください。

VariableArray 型のプロパティを追加する場合，このプロパティを追加する前に edmClass_Struct クラスのサブクラスを追加してください。このサブクラスは，VariableArray 型のプロパティが参照するオブジェクトを作成するためのクラスです。また，VariableArray 型のプロパティの配列の要素となるプロパティは，このサブクラスのプロパティとして定義しておく必要があります。サブクラスを追加するための定義情報ファイルの記述方法については，「4.7.4 定義情報ファイルの記述例」を参照してください。

なお，edmClass_Struct クラスのサブクラスの追加に関する定義情報と VariableArray 型のプロパティの追加に関する定義情報を同じ定義情報ファイルに記述する場合，VariableArray 型で定義するプロパティを追加する前に，edmClass_Struct クラスのサブクラスの追加を記述してください。

定義した VariableArray 型プロパティの要素に値を指定する場合に，その値が定義内容と合っているかどうかについてチェックを実行するとき，DocumentSpace 構成定義ファイルの VArrayElementCheck エントリを設定してください。なお，VArrayElementCheck エントリの詳細については，「4.2.2

DocumentSpace 構成定義ファイルの記述形式」を参照してください。

次に，VariableArray 型のプロパティを追加する場合の定義情報ファイルの記述方法および記述順序について，「(1) 使用する例題」で設定した例題に沿って説明します。

(a) 追加するプロパティのデータ型に対応する仮定値の定義

サブクラスに追加するプロパティのデータ型ごとに，プロパティの詳細情報（プロパティディスクリプション）を定義する必要があります。ここでは，データ型ごとに共通に指定するプロパティの仮定値を記述します。

例題で追加するプロパティについて，それぞれのデータ型に対応する仮定値を記述します。記述例を次に示します。

```
;
; DMA_DATATYPE_OBJECT_VARIABLE_ARRAY型の仮定値
;
[AssumeProperty/DMA_DATATYPE_OBJECT_VARIABLE_ARRAY]
dmaProp_IsSelectable=bool=1
dmaProp_IsSearchable=bool=1

;
; DMA_DATATYPE_STRINGの仮定値
;
[AssumeProperty/DMA_DATATYPE_STRING]
dmaProp_QueryOperator_Description
=obj=dsqop.ini@dmaQueryOperator_EqualString
=obj=dsqop.ini@dmaQueryOperator_UnequalString
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE
dmaProp_IsSelectable=bool=1
dmaProp_IsSearchable=bool=1
dmaProp_PropertyDefaultString=text=NULL
dmaProp_MaximumLengthString=int=64
```

＜アクションの説明＞

ここでは，[AssumeProperty/ データタイプ] のエントリとして追加するプロパティ自身のプロパティを記述します。

[AssumeProperty/ **データタイプ**]

VariableArray 型のプロパティを追加する場合，このアクションの「データタイプ」には，「DMA_DATATYPE_OBJECT_VARIABLE_ARRAY」を指定します。[AssumeProperty/ データタイプ] には，[AddProperty/ クラス名] で記述を省略するプロパティの仮定値を記述します。

＜プロパティの説明＞

次に示すプロパティは [AssumeProperty/ データタイプ] および [AddProperty/ クラス名] に指定できます。ただし，[AddProperty/ クラス名] に同じプロパティが指定されている場合は，[AddProperty/ クラス名] に指定されている値を優先します。[AddProperty/ クラス名] の値を省略した場合で，かつ [AssumeProperty/ データタイプ] の値を省略した場合は，各プロパティでの省略値が仮定されます。

- dmaProp_IsSelectable プロパティ
  追加するプロパティが検索結果集合の一項目として選択できるかどうかを指定する Boolean 型の値です。値として「1」を指定した場合は，このプロパティが検索結果集合の一項目として選択できるプロパティであることを示します。値として「0」を指定した場合は，このプロパティが検索結果集合の一項目として選択できないプロパティであることを示します。指定を省略した場合，「0」が仮定されます。
- dmaProp_IsSearchable プロパティ

追加するプロパティが検索の条件に使用できるかどうかを指定する Boolean 型の値です。値として「1」を指定した場合は，このプロパティが検索の条件として使用できるプロパティであることを示します。値として「0」を指定した場合は，このプロパティが検索の条件として使用できないプロパティであることを示します。指定を省略した場合，「0」が仮定されます。

- dmaProp_IsHidden プロパティ
  追加するプロパティをシステム管理者だけに表示するかどうかを示す Boolean 型の値です。「0」を指定してください。指定を省略した場合，「0」が仮定されます。
- dmaProp_MaximumElements プロパティ
  VariableArrayOfObject オブジェクトの要素を HiRDB の繰り返し列へ格納する場合に指定します。dmaProp_MaximumElements プロパティの値には，HiRDB の繰り返し列の最大要素数（2 ～ 30,000 の符号なし整数値）を指定します。
  ただし，定義済みのプロパティをクラスに追加する場合は，定義済みのプロパティに設定された dmaProp_MaximumElements プロパティの指定が有効になります。
  dmaProp_MaximumElements プロパティを指定する場合，メタ情報の初期設定コマンド（EDMInitMeta）の -v オプションを指定しておく必要があります。メタ情報の初期設定コマンド（EDMInitMeta）の -v オプションで指定した値（HIRDB，Own，Both）と dmaProp_MaximumElements プロパティの指定について次に示します。

- HIRDB：dmaProp_MaximumElements プロパティの指定は必須
- Own：dmaProp_MaximumElements プロパティは指定できない
- Both：dmaProp_MaximumElements プロパティの指定は任意（省略可能）

なお，メタ情報の初期設定コマンド（EDMInitMeta）の -v オプションで Both を指定してメタ情報を登録した場合，dmaProp_MaximumElements プロパティの指定の有無によって，VariableArray 型プロパティの要素の格納先が異なります。また，-v オプションを省略した場合，HiRDB の繰り返し列を使わないで VariableArrayOfObject オブジェクトを格納します。

(b) サブクラスの追加とプロパティの追加に関する定義

追加するサブクラスの定義とそのサブクラスに追加するプロパティの定義について記述します。記述例を次に示します。なお，ここでは VariableArray 型プロパティの要素の格納先として別表を使用する場合，および HiRDB の繰り返し列に格納する場合の記述例を示します。

1. VariableArray 型プロパティ「uP_WriterInfo プロパティ」の要素となるクラス「usrClass_WriterInfo クラス」を追加するための定義情報ファイルの内容を次に示します。

```
; VariableArray型プロパティを追加するために使用する，
; edmClass_Structクラスのサブクラス（usrClass_WriterInfo）を定義する。
;
; 著者の情報を管理するクラスの定義
;
[AddSubClass]
dmaProp_DisplayName=text=edmClass_Struct/usrClass_WriterInfo

; 著者名の定義
[AddProperty/usrClass_WriterInfo]
dmaProp_DisplayName=text=uP_WriterName
dmaProp_DataType=int=DMA_DATATYPE_STRING

; 著者の紹介
[AddProperty/usrClass_WriterInfo]
dmaProp_DisplayName=text=uP_Comment
dmaProp_DataType=int=DMA_DATATYPE_STRING
dmaProp_MaximumLengthString=int=200
```

2. VariableArray 型プロパティ「uP_WriterInfo プロパティ」の要素を別表に格納する場合に，「usrClass_DocList クラス」を追加するための定義情報ファイルの内容を次に示します。
   なお，VariableArray 型プロパティ「up_WriterInfo プロパティ」の要素を HiRDB の繰り返し列に格納する場合に，「usrClass_DocList クラス」を追加するための定義情報ファイルの内容については，「手順 3.」に示す記述例を参照してください。

```
;サブクラスとしてusrClass_DocListクラスを定義し，プロパティとして
;VariableArray型プロパティであるuP_WriterInfoプロパティを定義する。
;

; 文書リスト
[AddSubClass]
dmaProp_DisplayName=text=edmClass_IndependentPersistence/usrClass_DocList

; 文書名
[AddProperty/usrClass_DocList]
dmaProp_DisplayName=text=uP_DocName
dmaProp_DataType=int=DMA_DATATYPE_STRING

; 著者の情報
[AddProperty/usrClass_DocList]
dmaProp_DisplayName=text=uP_WriterInfo
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_Cardinality=int=255
dmaProp_RequiredClass=obj=usrClass_WriterInfo
```

3. VariableArray 型プロパティ「up_WriterInfo プロパティ」の要素を HiRDB の繰り返し列に格納する場合に，「usrClass_DocList クラス」を追加するための定義情報ファイルの内容を次に示します。

```
;サブクラスとしてusrClass_DocListクラスを定義し，プロパティとして
;VariableArray型プロパティであるuP_WriterInfoプロパティを定義する。
;

; 文書リスト
[AddSubClass]
dmaProp_DisplayName=text=edmClass_IndependentPersistence/usrClass_DocList

; 文書名
[AddProperty/usrClass_DocList]
dmaProp_DisplayName=text=uP_DocName
dmaProp_DataType=int=DMA_DATATYPE_STRING

; 著者の情報
[AddProperty/usrClass_DocList]
dmaProp_DisplayName=text=uP_WriterInfo
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_Cardinality=int=255
dmaProp_RequiredClass=obj=usrClass_WriterInfo
dmaProp_MaximumElements=int=100
```

＜アクションの説明＞

ここでは，[AddProperty/ クラス名] のエントリとして追加するプロパティ自身のプロパティを記述します。

[AddProperty/ **クラス名**]

VariableArray 型のプロパティを追加する場合，このアクションの「クラス名」には，プロパティを追加するクラス名（ここでは，usrClass_WriterInfo および usrClass_DocList）を指定します。

＜プロパティの説明＞

ほかのクラスに同じプロパティが定義されていない場合，次のプロパティは [AddProperty/ クラス名] に指定してください。これらのプロパティは，[AssumeProperty/ データタイプ] には指定できません。指定するプロパティとその値について次に示します。

- dmaProp_DisplayName プロパティ
  追加するプロパティ名を値として指定します。なお，この値の指定は，省略できません。
- dmaProp_DescriptiveText プロパティ
  追加するプロパティを簡単に説明する文字列を値として指定します。指定を省略した場合，dmaProp_DisplayName プロパティに指定したプロパティ名が仮定されます。
- dmaProp_Ids プロパティ
  追加するプロパティに付与する GUID 値を指定します。指定を省略した場合，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを必ず指定してください。
- dmaProp_DataType プロパティ
  追加するプロパティのデータ型を指定します。必ず「DMA_DATATYPE_OBJECT」を指定してください。省略できません。
- dmaProp_Cardinality プロパティ
  追加するプロパティの基本単位を表す Integer32 型の値です。必ず「255」を指定してください。省略できません。
- dmaProp_RequiredClass プロパティ
  edmClass_Struct クラスのサブクラスのクラス名（ここでは，usrClass_WriterInfo）を指定します。省略できません。
- dmaProp_MaximumElements プロパティ
  VariableArrayOfObject オブジェクトの要素を HiRDB の繰り返し列へ格納する場合に指定します。dmaProp_MaximumElements プロパティの値には，HiRDB の繰り返し列の最大要素数（2 ～ 30,000 の符号無し整数値）を指定します。ただし，定義済みのプロパティをクラスに追加する場合は，定義済みのプロパティに設定された dmaProp_MaximumElements プロパティの指定が有効になります。
  dmaProp_MaximumElements プロパティを指定する場合，メタ情報の初期設定コマンド（EDMInitMeta）の -v オプションを指定しておく必要があります。メタ情報の初期設定コマンド（EDMInitMeta）の -v オプションで指定した値（HIRDB，Own，Both）と dmaProp_MaximumElements プロパティの指定について次に示します。

- HIRDB：dmaProp_MaximumElements プロパティの指定は必須
- Own：dmaProp_MaximumElements プロパティは指定できない
- Both：dmaProp_MaximumElements プロパティの指定は任意（省略可能）

なお，メタ情報の初期設定コマンド（EDMInitMeta）の -v オプションで Both を指定してメタ情報を登録した場合，dmaProp_MaximumElements プロパティの指定の有無によって，VariableArray 型プロパティの要素の格納先が異なります。また，-v オプションを省略した場合，HiRDB の繰り返し列を使わないで VariableArrayOfObject オブジェクトを格納します。

4.7.6 定義情報ファイルの記述例（全文検索機能を使用する文書クラスを追加する場合）

ここでは，全文検索機能を使用する文書クラスを追加するための定義情報ファイルの記述例を次に示します。

＜記述例＞

```
# dmaClass_DocVersionクラスのサブクラスusrClass_DocTextSearchクラスを生成する
```

```
  [AddSubClass]
  dmaProp_DisplayName=text=dmaClass_DocVersion/usrClass_DocTextSearch

# usrClass_DocTextSearchクラスに五つのプロパティを追加する

  [AddProperty/usrClass_DocTextSearch]
  dmaProp_DisplayName=text=edmProp_TextIndex

  [AddProperty/usrClass_DocTextSearch]
  dmaProp_DisplayName=text=edmProp_DocLength

  [AddProperty/usrClass_DocTextSearch]
  dmaProp_DisplayName=text=edmProp_ContentIndexStatus

  [AddProperty/usrClass_DocTextSearch]
  dmaProp_DisplayName=text=edmProp_Score

  [AddProperty/usrClass_DocTextSearch]
  dmaProp_DisplayName=text=edmProp_RawScore
```

＜説明＞

全文検索機能を使用する文書クラスとして，usrClass_DocTextSearch クラスを定義して，そのクラスに五つのプロパティを追加することを示しています。

サブクラスを追加するので，アクション名は [AddSubClass] とします。定義するプロパティとその値は次のとおりです。

- dmaProp_DisplayName プロパティ
  追加するサブクラス名を値として指定します。dmaClass_DocVersion クラスのサブクラスとして，usrClass_DocTextSearch クラスを作成するので「dmaClass_DocVersion/usrClass_DocTextSearch」と指定しています。
- dmaProp_DescriptiveText プロパティ
  追加するサブクラスを簡単に説明する文字列を値として指定します。ここでは，記述を省略しています。この場合，dmaProp_DisplayName プロパティに指定した値から「スーパークラス名 /」を除いたサブクラス名，すなわち，usrClass_DocTextSearch が仮定されます。
- dmaProp_Ids プロパティ
  追加するサブクラスに付与する GUID を値として指定します。ここでは，記述を省略しています。したがって，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを指定する必要があります。

usrClass_DocTextSearch クラスに追加する五つのプロパティは動作環境メタ情報ファイル「edmprop.ini」にすでに記述されています。したがって，アクション名 [AddProperty/usrClass_DocTextSearch] には，dmaProp_DisplayName プロパティだけを指定しています。

4.7.7 定義情報ファイルの記述例（概念検索機能を使用する文書クラスを追加する場合）

ここでは，概念検索機能を使用する文書クラスを追加するための定義情報ファイルの記述例を次に示します。

＜記述例＞

```
# dmaClass_DocVersionクラスのサブクラスusrClass_DocTextSearchクラスを生成する

  [AddSubClass]
  dmaProp_DisplayName=text=dmaClass_DocVersion/usrClass_DocTextSearch

# usrClass_DocTextSearchクラスに六つのプロパティを追加する
```

```
[AddProperty/usrClass_DocTextSearch]
dmaProp_DisplayName=text=edmProp_ConceptStIndex

[AddProperty/usrClass_DocTextSearch]
dmaProp_DisplayName=text=edmProp_DocLength

[AddProperty/usrClass_DocTextSearch]
dmaProp_DisplayName=text=edmProp_ContentIndexStatus

[AddProperty/usrClass_DocTextSearch]
dmaProp_DisplayName=text=edmProp_Score

[AddProperty/usrClass_DocTextSearch]
dmaProp_DisplayName=text=edmProp_RawScore

[AddProperty/usrClass_DocTextSearch]
dmaProp_DisplayName=text=edmProp_ScoreConcept
```

＜説明＞

概念検索機能を使用する文書クラスとして，usrClass_DocTextSearch クラスを定義して，そのクラスに対して六つのプロパティを追加することを示しています。
サブクラスを追加するので，アクション名は [AddSubClass] とします。定義するプロパティとその値は次のとおりです。

- dmaProp_DisplayName プロパティ
  追加するサブクラス名を値として指定します。dmaClass_DocVersion クラスのサブクラスとして，usrClass_DocTextSearch クラスを作成するので「dmaClass_DocVersion/usrClass_DocTextSearch」と指定しています。
- dmaProp_DescriptiveText プロパティ
  追加するサブクラスを簡単に説明する文字列を値として指定します。ここでは，記述自体を省略しています。この場合，dmaProp_DisplayName プロパティに指定した値から「スーパークラス名 /」を除いたサブクラス名，すなわち，usrClass_DocTextSearch クラスが仮定されます。
- dmaProp_Ids プロパティ
  追加するサブクラスに付与する GUID を値として指定します。ここでは，記述自体を省略しています。したがって，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを指定する必要があります。

usrClass_DocTextSearch クラスに追加する六つのプロパティは動作環境メタ情報ファイル「edmprop.ini」にすでに記述されています。したがって，アクション名 [AddProperty/usrClass_DocTextSearch] には，dmaProp_DisplayName プロパティだけを指定しています。

4.7.8 定義情報ファイルの記述例（文字列型プロパティに対する全文検索機能を使用する場合）

文字列型プロパティに対する全文検索機能を使用する場合の定義情報ファイルについて，例題に沿って説明します。

全文検索機能を使用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。文字列型プロパティに対して全文検索機能を使用する場合，全文検索機能付き文字列型プロパティを追加します。

なお，全文検索機能付き文字列型プロパティの追加方法は，基本的に String 型のプロパティの追加方法と同じです。そのため，ここでは String 型のプロパティの追加方法と異なる個所について説明します。
String 型のプロパティの追加方法については，「4.7.4　定義情報ファイルの記述例」を参照してください。

また，データベース定義の名称定義の方法が GUID 値を変換した値を使用する方法の場合，文字列型プロパティに対する全文検索機能を使用できません。データベース定義の名称については，「3.11.1 データベース定義の名称の検討」を参照してください。

(1) 使用する例題

全文検索機能付き文字列型プロパティを文書クラスに定義するための例題を次に示します。

(例)

管理する文書として「設計ドキュメント」を定義します。この文書は，製品の開発に関する情報を管理します。
「設計ドキュメント」を管理するためにサブクラスを追加します。追加するサブクラスには，全文検索機能付き文字列型プロパティを追加します。
追加するサブクラスとそのスーパークラスを次に示します。

表 4-20 追加するサブクラスとそのスーパークラス（文字列型プロパティに対する全文検索機能を使用する場合）

管理する文書	追加するサブクラス	スーパークラス
設計ドキュメント	usrClass_PropTextSearch	dmaClass_DocVersion

dmaClass_DocVersion クラスのサブクラスとして usrClass_PropTextSearch クラスを追加して，設計ドキュメントを管理します。
また，文書に付加する情報としてプロパティを追加します。設計ドキュメントの付加情報と，付加情報に対応するプロパティを次の表に示します。

表 4-21 usrClass_PropTextSearch クラスの構造

クラス名	付加情報	プロパティ名	データ型
usrClass_PropTextSearch	文書名	usrProp_DocName	String 型
	執筆者	usrProp_DocWriter	String 型
	文書の概要	usrProp_DocSummary（全文検索機能付き文字列型プロパティ）	String 型

usrProp_DocSummary プロパティは，全文検索機能付き文字列型プロパティです。

全文検索機能付き文字列型プロパティを追加できるクラスは次のとおりです。

- edmClass_IndependentPersistence クラスのサブクラス
- dmaClass_ConfigurationHistory クラスのサブクラス
- dmaClass_Container クラスのサブクラス
- edmClass_ContainerVersion クラスのサブクラス
- edmClass_ComponentDocVersion クラスのサブクラス
- dmaClass_DocVersion クラスのサブクラス
- edmClass_VersionTracedComponentDocVersion クラスのサブクラス
- edmClass_VersionTracedDocVersion クラスのサブクラス

(2) 全文検索機能付き文字列型プロパティの追加方法

全文検索機能付き文字列型プロパティを追加するには，プロパティを追加するために必要な情報を定義情報ファイルに記述します。

全文検索機能付き文字列型プロパティを追加する場合の定義情報ファイルの記述方法について，「(1) 使用する例題」で設定した例題に沿って説明します。

(a) 追加するプロパティのデータ型に対応する仮定値の定義

サブクラスに追加するプロパティのデータ型ごとに，プロパティの詳細情報（プロパティディスクリプション）を定義します。ここでは，データ型ごとで共通に指定するプロパティの仮定値を記述します。

なお，全文検索機能付き文字列型プロパティのデータ型は String 型です。そのため，[AssumeProperty/DMA_DATATYPE_STRING] で指定する仮定値は，データ型が String 型のプロパティと全文検索機能付き文字列型プロパティで共通の値になります。

例題で追加するプロパティはすべて String 型なので，String 型に対応する仮定値を記述します。

記述例を次に示します。

```
;
; DMA_DATATYPE_STRINGの仮定値
;
[AssumeProperty/DMA_DATATYPE_STRING]
dmaProp_QueryOperatorDescriptions
=obj=dsqop.ini@dmaQueryOperator_EqualString
=obj=dsqop.ini@dmaQueryOperator_UnequalString
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE
dmaProp_IsSelectable=bool=1
dmaProp_IsSearchable=bool=1
dmaProp_IsOrderable=bool=1
dmaProp_PropertyDefaultString=text=NULL
dmaProp_MaximumLengthString=int=64
```

記述例について説明します。

＜アクションの説明＞

ここでは，[AssumeProperty/ データタイプ] のエントリとして追加するプロパティ自身のプロパティを記述します。

[AssumeProperty/ **データタイプ**]

全文検索機能付き文字列型プロパティを追加する場合，「データタイプ」には「DMA_DATATYPE_STRING」を指定します。

[AssumeProperty/ データタイプ] に続くエントリには，[AddProperty/ クラス名] で記述を省略するプロパティの仮定値を記述します。

＜プロパティの説明＞

ここでは，[AssumeProperty/DMA_DATATYPE_STRING] のエントリとして追加するプロパティを記述します。

[AssumeProperty/DMA_DATATYPE_STRING] のエントリとして追加するプロパティのうち，仮定値の定義方法が String 型のプロパティを追加する場合と異なるプロパティを次に示します。そのほかのプロパティは，String 型のプロパティを追加する場合に記述するプロパティと同じです。

- dmaProp_QueryOperatorDescriptions プロパティ
  全文検索機能付き文字列型プロパティを追加する場合，このプロパティの指定は無視されます。
- dmaProp_IsSelectable プロパティ
  追加するプロパティが検索結果集合の一項目として選択できるかどうかを指定する Boolean 型の値です。値として「1」を指定した場合は，このプロパティが検索結果集合の一項目として選択できるプロパティであることを示します。値として「0」を指定した場合は，このプロパティが検索結果集合の一項目として選択できないプロパティであることを示します。指定を省略した場合，「0」が仮定されます。

[AssumeProperty/DMA_DATATYPE_STRING] での指定を省略する場合は，全文検索機能付き文字列型プロパティを追加する [AddProperty/ クラス名] で「1」を指定してください。

- dmaProp_IsSearchable プロパティ
  追加するプロパティが検索の条件として使用できるかどうかを指定する Boolean 型の値です。値として「1」を指定した場合は，このプロパティが検索の条件として使用できるプロパティであることを示します。値として「0」を指定した場合は，このプロパティが検索の条件として使用できないプロパティであることを示します。指定を省略した場合，「0」が仮定されます。
  [AssumeProperty/DMA_DATATYPE_STRING] での指定を省略する場合は，全文検索機能付き文字列型プロパティを追加する [AddProperty/ クラス名] で「1」を指定してください。
- dmaProp_IsOrderable プロパティ
  全文検索機能付き文字列型プロパティを追加する場合，このプロパティの指定は無視されます。全文検索機能付き文字列型プロパティは，検索結果をソートできません。

(b) サブクラスの追加とプロパティの追加に関する定義

全文検索機能付き文字列型プロパティを追加する場合に，追加するサブクラスの定義とそのサブクラスに追加するプロパティの定義について記述します。記述例を次に示します。

```
;
; 設計ドキュメント用のサブクラスの定義
;
[AddSubClass]
dmaProp_DisplayName=text=dmaClass_DocVersion/usrClass_PropTextSearch

;
; 文書名（usrProp_DocNameプロパティの定義）
;
[AddProperty/usrClass_PropTextSearch]
dmaProp_DisplayName=text=usrProp_DocName
dmaProp_DataType=int=DMA_DATATYPE_STRING

;
; 執筆者（usrProp_DocWriterプロパティの定義）
;
[AddProperty/usrClass_PropTextSearch]
dmaProp_DisplayName=text=usrProp_DocWriter
dmaProp_DataType=int=DMA_DATATYPE_STRING

;
; 文書の概要（usrProp_DocSummaryプロパティの定義）
;
[AddProperty/usrClass_PropTextSearch]
dmaProp_DisplayName=text=usrProp_DocSummary
dmaProp_DataType=int=DMA_DATATYPE_STRING
dmaProp_MaximumLengthString=int=255
edmProp_SearchExtention=text=FreeWordIndex
```

記述例について説明します。

＜サブクラスを追加するためのアクションの説明＞

サブクラスを追加するためには，定義情報として [AddSubClass] を指定します。アクション名に続くエントリには，次に示すプロパティとその値を指定します。

- dmaProp_DisplayName
  追加するサブクラス名を値として指定します。
  ここでは，dmaClass_DocVersion クラスのサブクラスとして usrClass_PropTextSearch クラスを作成するので，「dmaClass_DocVersion/usrClass_PropTextSearch」と指定しています。
- dmaProp_DescriptiveText プロパティ
  追加するサブクラスを簡単に説明する文字列を値として指定します。指定を省略した場合，dmaProp_DisplayName プロパティに指定した値から「スーパークラス名 /」を除いたサブクラス

名，すなわち，usrClass_PropTextSearch が仮定されます。
ここでは，記述を省略しています。

- dmaProp_Ids プロパティ
  追加するサブクラスに付与する GUID を値として指定します。指定を省略した場合，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを必ず指定してください。
  ここでは，記述を省略しています。

＜サブクラスにプロパティを追加するためのアクションの説明＞

ここでは，[AddProperty/ クラス名] のエントリとして追加するプロパティを記述します。

[AddProperty/ **クラス名**]

全文検索機能付き文字列型プロパティを追加する場合，「クラス名」にはプロパティを追加するクラス名を指定します。
ここでは，「usrClass_PropTextSearch」を指定しています。

＜サブクラスにプロパティを追加するためのプロパティの説明＞

ほかのクラスに同じプロパティが定義されていない場合，次のプロパティは [AddProperty/ クラス名] に指定してください。これらのプロパティは，[AssumeProperty/DMA_DATATYPE_STRING] には指定できません。

- dmaProp_DisplayName
  追加するプロパティ名を値として指定します。なお，この値の指定は省略できません。
- dmaProp_DescriptiveText プロパティ
  追加するプロパティを簡単に説明する文字列を値として指定します。指定を省略した場合，dmaProp_DisplayName プロパティに指定したプロパティ名が仮定されます。
  ここでは，記述を省略しています。
- dmaProp_Ids プロパティ
  追加するプロパティに付与する GUID を値として指定します。指定を省略した場合，メタ情報の追加コマンド（EDMAddMeta）を実行するときに，-g オプションを必ず指定してください。
  ここでは，記述を省略しています。
- dmaProp_DataType プロパティ
  追加するプロパティのデータ型を示す文字列を値として指定します。全文検索機能付き文字列型プロパティを追加する場合は，必ず「DMA_DATATYPE_STRING」を指定します。この指定は省略できません。
- edmProp_SearchExtention プロパティ
  全文検索機能付き文字列型プロパティを追加する場合に指定します。全文検索機能付き文字列型プロパティを追加する場合は，必ず「FreeWordIndex」を指定します。この指定は省略できません。

4.8　RD エリア定義情報ファイル

この節では，RD エリア定義情報ファイルについて説明します。

RD エリア定義情報ファイルには，次のサンプルファイルが提供されています。

- /opt/HiEDMS/sample/EDM_RDAREA.txt（アクセス制御機能を使用しない場合）
- /opt/HiEDMS/sample/EDM_RDAREA_ACL.txt（アクセス制御機能を使用する場合）

DocumentBroker の実行環境に合ったサンプルファイルを選択し，編集してください。

4.8.1　RD エリア定義情報ファイルの概要

RD エリア定義情報ファイルは，データベース定義文中の RD エリア名を，ユーザが指定した RD エリア名で出力させるために作成するファイルです。作成したファイルは，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびメタ情報の追加コマンド（EDMAddMeta）の -r オプションで指定します。指定したファイルの内容が，-o オプションで指定したデータベース定義文格納ファイル中の RD エリア名に反映されます。これによって，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびメタ情報の追加コマンド（EDMAddMeta）で出力されるデータベース定義文中の RD エリアの手動による変更が不要になります。

4.8.2　RD エリア定義情報ファイルの記述形式

ここでは，RD エリア定義情報ファイルの記述形式について説明します。

RD エリア定義情報ファイルは，次に示す六つのセクションによって構成されます。

● [TableArea] セクション

● [IndexArea] セクション

● [LobArea] セクション

● [SGMLTEXTLobArea] セクション

● [NgramIndexArea] セクション

● [SearchExtentionPropertyIndexArea] セクション

セクションの記述規則を次に示します。

● 印刷可能な ASCII コードで記述します。

●「;」（セミコロン）および「#」（シャープ）で始まる行はコメントとして扱われます。

以降，RD エリア定義情報ファイルを構成する各セクションについて説明します。

（1）[TableArea] セクション

表として定義されるクラスと RD エリア名を対応づけます。

次の記述形式に従って記述します。

＜記述形式＞

```
class=class_name,area=area_name
```

class_name（クラス名）

HiRDB の表として定義されるクラスの名称（dmaProp_DisplayName プロパティに設定した値）を指定します。

HiRDB の表として定義されるクラスは次のとおりです。

- dmaClass_DocVersion クラス
- dmaClass_Rendition クラス
- dmaClass_ContentReference クラス
- dmaClass_ContentTransfer クラス
- dmaClass_ConfigurationHistory クラス
- dmaClass_Container クラス
- dmaClass_DirectContainmentRelationship クラス
- dmaClass_ReferentialContainmentRelationship クラス
- dmaClass_VersionSeries クラス
- dmaClass_VersionDescription クラス
- edmClass_IndependentPersistence クラス
- edmClass_ComponentDocVersion クラス
- edmClass_ContainerVersion クラス
- edmClass_VersionTraceableContainer クラス
- edmClass_VersionTracedDocVersion クラス
- edmClass_VersionTracedComponentDocVersion クラス
- edmClass_VersionTraceableContainmentRelationship クラス
- edmClass_OIID クラス
- edmClass_ACL クラス（アクセス制御機能の使用時だけ）
- edmClass_PublicACL クラス（アクセス制御機能の使用時だけ）
- edmClass_BindRelationship クラス（アクセス制御機能の使用時だけ）
- edmClass_ContentTransfers クラス（マルチファイル管理機能の使用時だけ）
- edmClass_ContentFileLink クラス（File Link 連携機能の使用時だけ）
- edmClass_ContentReference クラス（リファレンスファイル管理機能の使用時だけ）
- edmClass_Relationship クラス
- edmClass_VTRelationship クラス
- ユーザが追加したサブクラス

area_name（RD エリア名）

表を格納するユーザ用 RD エリア名を指定します。

（2） [IndexArea] セクション

HiRDB のインデクスとして定義されるプロパティと RD エリア名を対応づけます。

次の記述形式に従って記述します。

＜記述形式＞

class=class_name,prop=property_name[,prop=property_name,...]，area=area_name

class_name（クラス名）

「(1) [TableArea] セクション」のクラス名を参照してください。

property_name（プロパティ名）

インデクスのキーに選択するプロパティの名称（dmaProp_DisplayName プロパティに設定した値）

を指定します。
DocumentBroker は，次に示すプロパティに対してインデクスを定義します。

dmaProp_OIID プロパティ

このプロパティは，「(1) [TableArea] セクション」の class_name（クラス名）の説明に記述しているクラスのうち，edmClass_ACL クラスを除くクラスに定義されます。

OIIDPropertyDescription プロパティ

このプロパティは，edmClass_ACL クラスに定義されます。

dmaProp_This プロパティ

このプロパティは，次に示すクラスおよびそのサブクラスに定義されます。

- edmClass_PublicACL クラス
- edmClass_Relationship クラス
- edmClass_VTRelationship クラス
- edmClass_ContainerVersion クラスおよびそのサブクラス
- edmClass_VersionTraceableContainer クラスおよびそのサブクラス
- dmaClass_ConfigurationHistory クラスおよびそのサブクラス
- dmaClass_Container クラスおよびそのサブクラス
- dmaClass_DocVersion クラスおよびそのサブクラス
- edmClass_ComponentDocVersion クラスおよびそのサブクラス
- edmClass_VersionTracedComponentDocVersion クラスおよびそのサブクラス
- edmClass_VersionTracedDocVersion クラスおよびそのサブクラス

ThisPropertyDescription プロパティ

このプロパティは，次に示すクラスおよびそのサブクラスに定義されます。

- dmaClass_ContentReference クラス
- dmaClass_ContentTransfer クラス
- edmClass_ContentFileLink クラス
- dmaClass_DirectContainmentRelationship クラス
- dmaClass_ReferentialContainmentRelationship クラス
- dmaClass_Rendition クラス
- dmaClass_VersionDescription クラス
- dmaClass_VersionSeries クラス
- edmClass_ContentReference クラス
- edmClass_ContentTransfers クラス
- edmClass_VersionTraceableContainmentRelationship クラス
- edmClass_Relationship クラス
- edmClass_VTRelationship クラス
- edmClass_IndependentPersistence クラスおよびそのサブクラス
- edmClass_Struct クラスのサブクラス

dmaProp_ParentContainer プロパティ

このプロパティは，次に示すクラスおよびそのサブクラスに定義されます。

- edmClass_ContainerVersion クラスおよびそのサブクラス
- edmClass_VersionTraceableContainer クラスおよびそのサブクラス
- dmaClass_ConfigurationHistory クラスおよびそのサブクラス
- dmaClass_Container クラスおよびそのサブクラス
- dmaClass_DocVersion クラスおよびそのサブクラス

- edmClass_ComponentDocVersion クラスおよびそのサブクラス
- edmClass_VersionTracedComponentDocVersion クラスおよびそのサブクラス
- edmClass_VersionTracedDocVersion クラスおよびそのサブクラス

dmaProp_Head プロパティ

このプロパティは，次に示すクラスに定義されます。

- dmaClass_DirectContainmentRelationship クラス
- dmaClass_ReferentialContainmentRelationship クラス
- edmClass_VersionTraceableContainmentRelationship クラス
- edmClass_Relationship クラス
- edmClass_VTRelationship クラス

dmaProp_Head プロパティおよび dmaProp_Tail プロパティ

このプロパティは，次に示すクラスに定義されます。

- edmClass_Relationship クラス
- edmClass_VTRelationship クラス

dmaProp_Tail プロパティ

このプロパティは，次に示すクラスに定義されます。

- dmaClass_DirectContainmentRelationship クラス
- dmaClass_ReferentialContainmentRelationship クラス
- edmClass_VersionTraceableContainmentRelationship クラス
- edmClass_Relationship クラス
- edmClass_VTRelationship クラス

edmProp_HeadVTConfigurationHistory プロパティおよび dmaProp_Tail プロパティ

このプロパティは，edmClass_VTRelationship クラスに定義されます。

edmProp_TailVTConfigurationHistory プロパティ

このプロパティは，edmClass_VTRelationship クラスに定義されます。

dmaProp_VersionSeries プロパティ

このプロパティは，dmaClass_VersionDescription クラスに定義されます。

dmaProp_Version プロパティ

このプロパティは，dmaClass_VersionDescription クラスに定義されます。

edmProp_VTVersionSeries プロパティ

このプロパティは，次に示すクラスに定義されます。

- edmClass_VersionTraceableContainmentRelationship クラス

edmProp_Parent プロパティ

このプロパティは，次に示すクラスおよびそのサブクラスに定義されます。

- dmaClass_Rendition クラス
- dmaClass_ContentReference クラス
- dmaClass_ContentTransfer クラス
- edmClass_ContentReference クラス
- edmClass_ContentTransfers クラス
- edmClass_ContentFileLink クラス
- edmClass_Struct クラスのサブクラス

edmProp_RenditionStatus プロパティ

このプロパティは，dmaClass_Rendition クラスに定義されます。

edmProp_OwnerId **プロパティ**

このプロパティは，アクセス制御機能を使用する場合に定義されます。次に示すクラスおよびそのサブクラスに定義されます。

- edmClass_PublicACL クラス
- edmClass_ContainerVersion クラスおよびそのサブクラス
- edmClass_VersionTraceableContainer クラスおよびそのサブクラス
- dmaClass_ConfigurationHistory クラスおよびそのサブクラス
- dmaClass_Container クラスおよびそのサブクラス
- dmaClass_DocVersion クラスおよびそのサブクラス
- edmClass_ComponentDocVersion クラスおよびそのサブクラス
- edmClass_IndependentPersistence クラスおよびそのサブクラス
- edmClass_VersionTracedComponentDocVersion クラスおよびそのサブクラス
- edmClass_VersionTracedDocVersion クラスおよびそのサブクラス

edmProp_PrimaryGroupId **プロパティ**

このプロパティは，アクセス制御機能を使用する場合に定義されます。次に示すクラスおよびそのサブクラスに定義されます。

- edmClass_ContainerVersion クラスおよびそのサブクラス
- edmClass_VersionTraceableContainer クラスおよびそのサブクラス
- dmaClass_ConfigurationHistory クラスおよびそのサブクラス
- dmaClass_Container クラスおよびそのサブクラス
- dmaClass_DocVersion クラスおよびそのサブクラス
- edmClass_ComponentDocVersion クラスおよびそのサブクラス
- edmClass_IndependentPersistence クラスおよびそのサブクラス
- edmClass_VersionTracedComponentDocVersion クラスおよびそのサブクラス
- edmClass_VersionTracedDocVersion クラスおよびそのサブクラス

edmProp_BindObject **プロパティ**

このプロパティは，次に示すクラスに定義されます。

- edmClass_ACL クラス
- edmClass_BindRelationship クラス

edmProp_ACLId **プロパティ**

このプロパティは，edmClass_BindRelationship クラスに定義されます。

edmProp_ReferenceType **プロパティ**

このプロパティは，edmClass_ContentReference クラスに定義されます。

area_name（RD **エリア名**）

インデクスを格納するユーザ用 RD エリア名を指定します。

（3） [LobArea] セクション

BLOB 列として定義されるプロパティと LOB 列格納用 RD エリア名を対応づけます。

次の記述形式に従って記述します。

＜記述形式＞

class=class_name,prop=property_name,area=area_name

class_name（**クラス名**）

次に示すクラス名を指定します。
- dmaClass_ContentTransfer クラス

property_name（**プロパティ名**）
edmProp_Content プロパティを指定します。

area_name（RD **エリア名**）
文書実体を格納する LOB 列格納用 RD エリア名を指定します。

（4） [SGMLTEXTLobArea] セクション

全文検索機能付き文書クラスと全文検索用の文書内容格納用 RD エリア名を対応づけます。次の記述形式に従って記述します。

＜記述形式＞
class=class_name,area=area_name

class_name（**クラス名**）
全文検索機能付き文書クラスの名称（dmaProp_DisplayName プロパティに設定した値）を指定します。

area_name（RD **エリア名**）
全文検索用の文書内容格納用 RD エリア名を指定します。

（5） [NgramIndexArea] セクション

全文検索機能付き文書クラスに定義される全文検索インデクスと全文検索インデクス格納用 RD エリア名を対応づけます。次の記述形式に従って記述します。

＜記述形式＞
class=class_name,area=area_name

class_name（**クラス名**）
全文検索機能付き文書クラスの名称（dmaProp_DisplayName プロパティに設定した値）を指定します。

area_name（RD **エリア名**）
全文検索インデクス格納用 RD エリア名を指定します。

（6） [SearchExtentionPropertyIndexArea] セクション

全文検索機能付き文字列型プロパティに定義される全文検索インデクスと全文検索インデクス格納用 RD エリア名を対応づけます。次の記述形式に従って記述します。

＜記述形式＞
class=class_name,prop=property_name,area=area_name

class_name（**クラス名**）
全文検索機能付き文字列型プロパティを追加したクラスの名称（dmaProp_DisplayName プロパティに設定した値）を指定します。

property_name（**プロパティ名**）
全文検索機能付き文字列型プロパティの名称（dmaProp_DisplayName プロパティに設定した値）を

指定します。

area_name（RD エリア名）

全文検索インデクス格納用 RD エリア名を指定します。

4.8.3 RD エリア定義情報ファイルの記述例

RD エリア定義情報ファイルの記述例を次に示します。

```
[TableArea]
class=dmaClass_DocVersion,area=USER01
class=dmaClass_ContentTransfer,area=USER02
class=dmaClass_DocVersion/usrClass_ProbInfoDoc,area=USER01
class=dmaClass_DocVersion/usrClass_DocTextSearch,area=USER01
class=edmClass_ContentReference,area=USER03

[IndexArea]
class=dmaClass_DocVersion,prop=dmaProp_OIID,area=INDEX01
class=dmaClass_DocVersion,prop=dmaProp_This,area=INDEX01
class=dmaClass_DocVersion,prop=dmaProp_ParentContainer,area=INDEX01
class=dmaClass_ContentTransfer,prop=dmaProp_OIID,area=INDEX02
class=dmaClass_ContentTransfer,prop=dmaProp_This,area=INDEX02
class=dmaClass_ContentTransfer,prop=edmProp_Parent,area=INDEX02
class=dmaClass_DocVersion/usrClass_ProbInfoDoc,prop=dmaProp_OIID,area=INDEX01
class=dmaClass_DocVersion/usrClass_ProbInfoDoc,prop=dmaProp_This,area=INDEX01
class=dmaClass_DocVersion/
usrClass_ProbInfoDoc,prop=dmaProp_ParentContainer,area=INDEX01
class=dmaClass_DocVersion/
usrClass_ProbInfoDoc,prop=usrProp_DocumentID,area=INDEX01
class=dmaClass_DocVersion/usrClass_DocTextSearch,prop=dmaProp_OIID,area=INDEX01
class=dmaClass_DocVersion/usrClass_DocTextSearch,prop=dmaProp_This,area=INDEX01
class=dmaClass_DocVersion/
usrClass_DocTextSearch,prop=dmaProp_ParentContainer,area=INDEX01
class=edmClass_ContentReference,prop=dmaProp_OIID,area=INDEX03
class=edmClass_ContentReference,prop=ThisPropertyDescription,area=INDEX03
class=edmClass_ContentReference,prop=edmProp_Parent,area=INDEX03
class=edmClass_ContentReference,prop=edmProp_ReferenceType,area=INDEX03

[LobArea]
class=dmaClass_ContentTransfer,prop=edmProp_Content,area=LOB01

[SGMLTEXTLobArea]
class=dmaClass_DocVersion/usrClass_DocTextSearch,area=LOB03
[NgramIndexArea]
class=dmaClass_DocVersion/usrClass_DocTextSearch,area=LOB04

[SearchExtentionPropertyIndexArea]
class=dmaClass_DocVersion/
usrClass_PropTextSearch,prop=usrProp_DocSummary,area=SEARCHINDEX1
```

4.8.4 RD エリア定義情報ファイルの注意事項

RD エリア定義情報ファイルの注意事項について説明します。

- 指定された RD エリア名が HiRDB に定義されているかどうかについてはチェックしません。
- 指定されたクラス名，プロパティ名の妥当性についてはチェックしません。
- [LobArea] セクション，[SGMLTEXTLobArea] セクション，[NgramIndexArea] セクション，および [SearchExtentionPropertyIndexArea] セクションに指定する RD エリア名は，ユニークな名称を指定してください。重複している場合，該当するエリア名には BLOBAREA という文字列を出力します。

- RD エリア定義情報ファイルの解析を実行して 10 件の誤りを検知した場合，それ以降の解析処理は実行されません。
- RD エリア定義情報ファイルには，空白を含むクラス名，プロパティ名，および RD エリア名は指定できません。空白を含むクラス名，プロパティ名，および RD エリア名を指定した場合，DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）実行時に KMBR10226-W のメッセージを出力するか，文書空間構築コマンド（EDMCBuildDocSpace）実行時に KMBR16920-E メッセージ（エラー情報：rc=5）もしくは KMBR10226-W メッセージを出力するか，またはデータベース定義ユティリティ実行時にデータベースのエラーでエラーとなります。

4.9 インデクス情報ファイル

この節では，インデクス情報ファイルについて説明します。

インデクス情報ファイルは，ユーザが追加するプロパティにインデクスを定義し，インデクスの作成をデータベース定義に追加する場合に作成してください。

4.9.1 インデクス情報ファイルの概要

サブクラスに追加するプロパティをキーにして検索したい場合，プロパティにインデクスを定義します。インデクス情報ファイルは，ユーザが追加するプロパティにインデクスを定義する場合に，インデクスの情報を記述するファイルです。

作成したインデクス情報ファイルをメタ情報の追加コマンド（EDMAddMeta），および DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）実行時に入力ファイルとして指定することで，データベース定義文にインデクスの作成を定義できます。インデクス情報ファイルで定義できる内容を次に示します。

● ユーザが定義するプロパティに対するインデクスの作成

● 全文検索インデクス用プロパティに対する差分インデクスの作成

4.9.2 インデクス名をデータベース定義の名称に使用する場合の規則

定義情報ファイルの dmaProp_DisplayName プロパティに記述したサブクラス名やプロパティ名を，データベース定義の名称定義として使用する場合の規則を説明します。ここでは，インデクス名（表を検索するためのキーとして列に付けた索引の名称）を定義する場合の規則について説明します。

インデクス名を定義する場合，英小文字，英大文字，数字，空白，「_」（下線文字），および「-」（マイナス記号）で指定します。インデクス名の指定では，DocumentBroker で付けるインデクス名を含めてデータベースのスキーマで一意に付ける必要があります。また，インデクス名を指定する場合は，次に示す規則およびデータベース定義の制約に従ってください。

- ユーザがインデクスを追加する場合，インデクス情報ファイルで指定したインデクス名が，データベース定義文のインデクス名になります。
- インデクス名には，「dmaClass」および「edmClass」で始まる名称を指定しないでください。
- インデクス名の先頭 1 バイトは，英小文字または英大文字で指定します。
- インデクス名の末尾は，空白にできません。
- インデクス名は，1 ～ 30 バイトで指定します。

4.9.3 全文検索での差分インデクス機能を使用するための設定

ここでは，差分インデクス機能を使用する場合に必要なインデクス情報ファイルの設定について説明します。

HiRDB Text Search Plug-in は，登録性能を向上させるために，小容量の一時的な登録用インデクスを作成し，登録した文書を追加できる機能を持っています。この機能を**差分インデクス機能**といいます。差分インデクス機能を使用する場合，全文検索インデクス用プロパティに対して，差分インデクスの作成を定義する必要があります。この場合，差分インデクスの作成，および作成する差分インデクスの容量をインデクス定義情報ファイルに定義することによって，差分インデクスの作成を定義したデータベース定義文を出力できます。

差分インデクスの作成を定義できる全文検索インデクス用プロパティを次に示します。

- 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

4.9.4 インデクス情報ファイルの記述形式

ここでは，インデクス情報ファイルの記述形式について説明します。

インデクス情報ファイルは，次に示す二つのセクションと，各セクションに指定するエントリによって構成されます。

● [Index] セクション

● [ContentSearchIndex] セクション

セクションとエントリの記述規則を次に示します。

● 印刷可能な ASCII コードで記述します。

● セクション名は，[ ]（角括弧）で囲んで指定します。一つのセクションは，セクション名を指定してから，次のセクション名を指定するまで，またはファイルの終端までの範囲です。

●「;」（セミコロン）および「#」（シャープ）で始まる行はコメントとして扱われます。

● インデクス情報ファイルの解析を実行して 10 件の誤りを検知した場合，それ以降の解析処理は実行されません。

以降，インデクス情報ファイルを構成する各セクションと，セクションごとに指定するエントリについて説明します。

```
〔[Index]〕
class=<class_name>〔,UNIQUE〕,prop=<property_name>〔,order={ASC｜DESC}〕
                            〔,prop=<property_name>〔,order={ASC｜DESC}〕...〕
                            〔,EXCEPT〕
                            〔,index=<index_name>〕

〔[ContentSearchIndex]
class=<class_name>,prop=<system_property_name>
                  〔,subindex=<numeric>〕
                  〔,index=<index_name>〕
]
```

(1) インデクス情報ファイルのセクションとエントリ

インデクス情報ファイルを構成するセクションとエントリについて説明します。

(a) 定義内容とセクションおよびエントリの対応

インデクス情報ファイルに定義する内容によって，セクションおよびエントリの設定は異なります。インデクス情報ファイルで定義する内容に対応したセクション名およびエントリ名を次に示します。

● ユーザが定義するプロパティに対するインデクスの作成
[Index] セクションで定義します。次のエントリを設定する必要があります。
- class エントリ
- prop エントリ

- UNIQUE エントリ
- order エントリ
- EXCEPT エントリ
- index エントリ

● 全文検索インデクス用プロパティに対する差分インデクスの作成
[ContentSearchIndex] セクションで定義します。次のエントリを設定する必要があります。

- class エントリ
- prop エントリ
- subindex エントリ
- index エントリ

(b) セクションおよびエントリの説明

インデクス情報ファイルを構成するセクションおよびエントリについて，説明します。

[Index]

[Index] セクションは，ユーザが定義するプロパティにインデクスを作成する場合に定義します。
[Index] セクションの各エントリについて説明します。

class=<class_name>

インデクスを定義するクラスの名称（クラスに定義される dmaProp_DisplayName プロパティの値）を指定します。

prop=<property_name>

インデクスのキーに選択するプロパティの名称（プロパティに定義される dmaProp_DisplayName プロパティの値）を指定します。
prop= は，最大 16 個指定できます。複数個を指定すると，複数列インデクスの定義となります。
なお，ここで指定した各プロパティの定義長の合計はチェックしません。

UNIQUE

インデクスのキーに選択するプロパティを一意に制約する場合に指定します。
この指定を省略した場合，インデクスのキーに選択するプロパティを一意に制約しません。

order={ASC|DESC}

インデクスを昇順に定義する場合は ASC，降順に定義する場合は DESC を指定します。

EXCEPT

NULL 値だけで構成されるキー値を除外してインデクスを定義する場合に指定します。指定を省略した場合，NULL 値だけで構成されるキー値を除外しません。

index=<index_name>

インデクスのインデクス名を指定します。インデクス名の名称定義の方法については，「4.9.2 インデクス名をデータベース定義の名称に使用する場合の規則」を参照してください。

[ContentSearchIndex]

[ContentSearchIndex] セクションは，全文検索インデクス用プロパティに差分インデクスを作成する場合に定義します。これらのインデクスを作成できる全文検索インデクス用プロパティを次に示します。

全文検索インデクス用プロパティに対する差分インデクスの作成

次に示す全文検索インデクス用プロパティに対して差分インデクスの作成を定義できます。

- 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

次に，[ContentSearchIndex] セクションの各エントリについて説明します。

class=<class_name>

インデクスの作成を定義する全文検索機能付き文書クラスの名称（クラスに定義される dmaProp_DisplayName プロパティの値）を指定します。

prop=<system_property_name>

インデクスのキーに選択する全文検索インデクス用プロパティの名称（プロパティに定義される dmaProp_DisplayName プロパティの値）を指定します。
なお，プロパティの詳細については，「2.3.4　全文検索機能付き文書クラスの追加」を参照してください。

subindex=<numeric>

全文検索インデクスに対して，差分インデクスの作成を定義するとともに，作成する差分インデクスの容量を定義します。<numeric> に指定できる値は，40,000 ～ 102,400（キロバイト）の符号なしの整数値です。<numeric> に指定できる値の範囲は，HiRDB Text Search Plug-in の差分インデクスで指定できる範囲と同じです。なお，この指定を省略した場合，差分インデクスの作成を定義する SQL 文は出力されません。
差分インデクスの詳細については，マニュアル「HiRDB Text Search Plug-in」を参照してください。

index=<index_name>

全文検索インデクス用プロパティに定義するインデクスのインデクス名を指定します。
指定を省略した場合，DocumentBroker が提供するインデクス名の名称定義に従ってインデクス名が定義されます。インデクス名の名称定義については，「4.9.2　インデクス名をデータベース定義の名称に使用する場合の規則」を参照してください。

4.9.5　インデクス情報ファイルの記述例

インデクス情報ファイルの記述例を次に示します。

```
# Sample file for creating index
# Index definition
[Index]
class=dmaClass_DocVersion/
usrClass_ProbInfoDoc,UNIQUE,prop=usrProp_DocumentID,order=ASC
class=dmaClass_DocVersion/usrClass_ProbInfoDoc,prop=usrProp_Category
class=dmaClass_Container/usrClass_Container,prop=usrProp_ContainerName,EXCEPT
class=dmaClass_DocVersion/
usrClass_ProbInfoDoc,prop=usrProp_DocumentID,order=DESC,prop=usrProp_Category
# Content Search Index definition

[ContentSearchIndex]
class=dmaClass_DocVersion/usrClass_DocTextSearch,property=edmProp_TextIndex
```

4.10　クラス定義情報ファイル

この節では，クラス定義情報ファイルについて説明します。

クラス定義情報ファイルは，クラス定義情報ファイル作成コマンド（EDMCrtSimMeta）を実行すると " 実行環境ディレクトリ /etc/meta_files" に作成されます。

クラス定義情報ファイルは，次の場合に作成してください。

- Java クラスライブラリを使用する場合
- DocumentBroker オブジェクト操作ツールを使用する場合

4.10.1　クラス定義情報ファイルの概要

クラス定義情報ファイルは，DocumentBroker に定義されている DMA オブジェクトのクラス，そのサブクラスのクラス，およびプロパティから，GUID，データ型，基本単位などの情報を記述したファイルです。

Java クラスライブラリおよび DocumentBroker オブジェクト操作ツールは，これらのクラスおよびプロパティに対応する GUID とプロパティの定義情報を使用して検索します。クラス定義情報ファイルは，これらのクラスおよびプロパティに対応する GUID とプロパティの定義情報を，DocumentBroker サーバを介さないで取得できるようにするために使用します。

また，クラス定義情報ファイル作成コマンド（EDMCrtSimMeta）を実行すると，接続先の DocumentBroker サーバで管理されているメタ情報からクラス定義情報ファイルを出力できます。

クラス定義情報ファイル作成コマンド（EDMCrtSimMeta）については，「7.3　コマンドの文法」を参照してください。

次にクラス定義情報ファイルの作成と更新について説明します。

クラス定義情報ファイルの作成

クラス定義情報ファイルは，クラス定義情報ファイル作成コマンド（EDMCrtSimMeta）を実行して作成します。

クラス定義情報ファイルの更新

「3.11.3　データベースシステムの設定手順」以外でクラスやプロパティを追加・更新してメタ情報ファイルを更新した場合，クラス定義情報ファイル作成コマンド（EDMCrtSimMeta）を実行して更新します。

4.10.2　クラス定義情報ファイルの出力形式

ここでは，クラス定義情報ファイルの出力形式について説明します。

(1) 出力ディレクトリとファイル名

クラス定義情報ファイルは，" 実行環境ディレクトリ /etc/meta_files" に作成されます。なお，クラス定義情報ファイル作成コマンド（EDMCrtSimMeta）で出力ディレクトリを指定した場合，その指定に従って出力されます。出力されるファイル名は，**文書空間識別子（GUID）+ サフィックス** (".ini") です。

例えば，文書空間識別子が 673D2BE0-D1FD-11D0-AB59-08002BE29E1D の場合，出力されるファイル名は，673D2BE0-D1FD-11D0-AB59-08002BE29E1D.ini になります。

(2) 出力情報

クラス定義情報ファイルに出力される情報を次に示します。

1. DocumentBroker **が定義するクラスで検索可能なクラス**
   - dmaClass_DocVersion クラス
   - edmClass_ContentSearch クラス
   - dmaClass_ConfigurationHistory クラス
   - dmaClass_Container クラス
   - dmaClass_DirectContainmentRelationship クラス
   - dmaClass_ReferentialContainmentRelationship クラス
   - edmClass_ComponentDocVersion クラス
   - edmClass_ContainerVersion クラス
   - edmClass_VersionTraceableContainer クラス
   - edmClass_VersionTraceableContainmentRelationship クラス
   - edmClass_VersionTracedComponentDocVersion クラス
   - edmClass_VersionTracedDocVersion クラス
   - edmClass_Relationship クラス
   - edmClass_PublicACL クラス
2. DocumentBroker **が定義するクラスでサブクラスの作成が可能なクラス**
   - dmaClass_ConfigurationHistory クラス
   - dmaClass_Container クラス
   - dmaClass_DocVersion クラス
   - edmClass_ComponentDocVersion クラス
   - edmClass_ContainerVersion クラス
   - edmClass_IndependentPersistence クラス
   - edmClass_Struct クラス
   - edmClass_VersionTraceableContainer クラス
   - edmClass_VersionTracedDocVersion クラス
   - edmClass_VersionTracedComponentDocVersion クラス
3. DocumentBroker **が定義するクラスでユーザがプロパティを追加定義可能なクラス**
   2. で示されたクラス，および次に示すクラスにプロパティを追加定義できます。
   - dmaClass_DirectContainmentRelationship クラス
   - dmaClass_ReferentialContainmentRelationship クラス
   - edmClass_VersionTraceableContainmentRelationship クラス
   - edmClass_PublicACL クラス
   - edmClass_Relationship クラス
4. **ユーザが定義するサブクラス**
5. DocumentBroker **が定義するプロパティでデータ型が** Integer32 **型，**String **型，**Object **型で検索可能または選択可能なプロパティ**
   メタ情報ファイルのプロパティ定義を記述する PropertyDescriptionNUMBLE セクションの IsSearchable エントリの値が「1」，または IsSelectable エントリの値が「1」であるプロパティ
6. **ユーザが定義するプロパティ**
7. **システム定義の情報**

クラス定義情報ファイルで定義されている内容を次に示します。それぞれの <> 内に定義情報が設定され

ます。なお，<> は非終端記号です。

```
[Version]
Version=text=<Version>-<OS種別>

[CreateDate]
CreateDate=date=<YYYY><MM><DD>T<hh><mm><ss>Z

[SystemDefinition]
DocSpaceCharacterSet=text={SJIS | UTF-8}

<CLASS_DESCRIPTION>::=
(
  [<CLASS NAME>]
  ClassId=guid=<CLASS ID>
  SuperClass=text=<SUPER CLASS NAME>
  (<PROPERTY NAME>=guid=<PROPERTY ID>)...
)...

<PROPERTY DESCRIPTION>::=
(
  [<PROPERTY ID>]
  dmaProp_DataType=int=<DMA DATATYPE>
  [
   dmaProp_RequiredClass=obj=<CLASS NAME>
   [dmaProp_Cardinality=int=<DMA CARDINALITY>]
  ]
  [=<SELECTIONS DATATYPE>=<SELECTIONS VALUE>]...
)...
```

次に，クラス定義情報ファイルに設定されている定義情報を説明します。

<Version>-<OS 種別 >

DocumentBroker のバージョンおよびオペレーティングシステムが設定されます。「=text=」が区切り文字として設定されます。<Version> および <OS 種別 > に設定される値を次に示します。

<Version>

DocumentBroker のバージョンが設定されます。

<OS 種別 >

オペレーティングシステムの種別が設定されます。

CreateDate

クラス定義情報ファイルを作成した日時が設定されます。CreateDate エントリに「=date=」が区切り文字として設定されます。<YYYY><MM><DD>，T，<hh><mm><ss> および Z について，それぞれ次に説明します。

<YYYY><MM><DD>

クラス定義情報ファイルが作成された日付（年，月，日）を示します。<YYYY>，<MM> および <DD> に設定される値を次に示します。

<YYYY>

クラス定義情報ファイルが作成された年（4 けた）を示します。0000 ～ 9999 の範囲で値が設定されます。

<MM>

クラス定義情報ファイルが作成された月（2 けた）を示します。01 ～ 12 の範囲で値が設定されます。

<DD>

クラス定義情報ファイルが作成された日（2 けた）を示します。01 ～ 31 の範囲で値が設定されます。

T

クラス定義情報ファイルが作成された日付と時刻の区切り文字を示します。

<hh><mm><ss>

クラス定義情報ファイルが作成された時刻（時，分，秒）を示します。

<hh>

クラス定義情報ファイルが作成された時刻を 24 時間表示で示します。00 ～ 23 の範囲で値が設定されます。

<mm>

クラス定義情報ファイルが作成された時刻（分）を示します。00 ～ 59 の範囲で値が設定されます。

<ss>

クラス定義情報ファイルが作成された時刻（秒）を示します。00 ～ 59 の範囲で値が設定されます。

Z

クラス定義情報ファイルが作成された日付と時刻の表示の終端を示します。

DocSpaceCharacterSet

DocumentBroker サーバの文書空間で使用する文字コード種別が設定されます。「=text=」を区切り文字として，SJIS または UTF-8 が設定されます。SJIS，および UTF-8 について説明します。

SJIS

DocumentBroker サーバの文書空間で使用する文字コード種別が Shift-JIS であることを示します。

UTF-8

DocumentBroker サーバの文書空間で使用する文字コード種別が UTF-8 であることを示します。

<CLASS DESCRIPTION>

クラス定義に関する情報が設定されます。<CLASS NAME>，<CLASS ID>，<SUPER CLASS NAME>，<PROPERTY NAME> および <Property ID> について，それぞれ次に説明します。

<CLASS NAME>

クラスの DisplayName プロパティの値が設定されます。これは，メタ情報ファイルのクラス定義が記述された [ClassDescription] セクションの dmaProp_DisplayName エントリの値です。また，この値は，「[」（左角括弧）と「]」（右角括弧）で囲まれて表示されます。

<CLASS ID>

クラスのクラス識別子（GUID）が設定されます。これは，メタ情報ファイルのクラス定義が記述された [ClassDescription] セクションの dmaProp_Ids エントリから参照されるセクションの GUID です。また，この値は，「=guid=」を区切り文字として設定されます。GUID の形式を次に示します。

GUID **の形式：**XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

「X」は 16 進数値文字を示します。「-」はマイナス記号です。

<SUPER CLASS NAME>

スーパークラスの DisplayName プロパティの値が設定されます。これは，ClassDescription オブジェクトの SuperclassDescription プロパティから参照される ClassDescription オブジェクトの dmaProp_DisplayName プロパティの値です。また，この値は，「=text=」を区切り文字とし

て設定されます。

<PROPERTY NAME>

プロパティの名称が設定されます。これは，メタ情報ファイルの [PropertyDescription] セクションの dmaProp_DisplayName エントリに指定された値です。

<Property ID>

プロパティ識別子（GUID）が設定されます。これは，[PropertyDescription] セクションの dmaProp_Ids エントリから参照するセクションに指定された GUID です。GUID の形式については，< CLASS ID> を参照してください。

<PropertyDescription>

プロパティの定義情報が設定されます。

<PROPERTY ID>::=<GUID>

プロパティ識別子（GUID）が設定されます。これは，[PropertyDescription] セクションの dmaProp_Ids エントリから参照するセクションに指定された GUID です。また，この値は，「[」（左角括弧）と「]」（右角括弧）で囲まれて表示されます。GUID の形式については，< CLASS ID> を参照してください。

<DMA DATATYPE>

プロパティのデータ型が設定されます。
[PropertyDescription] セクションの dmaProp_DataType エントリに指定された値が dmaProp_DataType エントリに「=int=」を区切り文字として設定されます。設定されるデータ型を次に示します。

- DMA_DATATYPE_BOOLEAN：Boolean 型
- DMA_DATATYPE_ID：Id 型
- DMA_DATATYPE_INTEGER32：Integer32 型
- DMA_DATATYPE_OBJECT：Object 型
- DMA_DATATYPE_STRING：String 型

<CLASS NAME>

配布オブジェクトのクラスが設定されます。列挙によって取得するすべてのオブジェクトを作成する基になったクラスのクラス名が設定されます。
[PropertyDescription] セクションの dmaProp_DataType が Object 型の場合に，dmaProp_RequiredClass エントリに指定された値が設定されます。この値は，dmaProp_RequiredClass エントリに「=obj=」を区切り文字として設定されます。

<DMA CARDINALITY>

プロパティの基本単位が設定されます。これは，[PropertyDescription] セクションの dmaProp_Cardinality エントリに指定された値です。また，この値は，dmaProp_Cardinality エントリに「=int=」を区切り文字として設定されます。設定される基本単位を次に示します。

- DMA_CARDINALITY_SINGLE:Scalar 型
- DMA_CARDINALITY_ENUM:Enumeration 型
- DMA_CARDINALITY_LIST:List 型
- 255:VariableArray 型

<SELECTIONS DATATYPE>

プロパティに設定できる値のデータ型が設定されます。
dmaProp_DataType プロパティに指定するデータ型が DMA_DATATYPE_INTEGER32 または DMA_DATATYPE_STRING の場合だけ設定されます。DMA_DATATYPE_INTEGER32 は「int」，DMA_DATATYPE_STRING は，「text」が設定されます。

dmaProp_DataType に設定されたデータ型に従って「=<SELECTIONSDATATYPE>=」の形式の区切り文字が設定され，そのあとに値が設定されます。

<SELECTIONS VALUE>

プロパティに設定された値が設定されます。この値は，印刷可能な ASCII コードで記述されます。

4.10.3 クラス定義情報ファイルの出力例

クラス定義情報ファイルの出力例を次に示します。

```
[Version]
Version=text=031100-AIX5.3

[CreateDate]
CreateDate=date=20000831T095801Z

[SystemDefinition]
DocSpaceCharacterSet=text=SJIS

[dmaClass_ConfigurationHistory]
ClassId=guid=01a3a8c6-7aec-11d1-a31b-0020af9fbb1c
SuperClass=text=dmaClass_Containable
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

[dmaClass_DocVersion]
ClassId=guid=01a3a8c2-7aec-11d1-a31b-0020af9fbb1c
SuperClass=text=dmaClass_Versionable
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

[dmaClass_Container]
ClassId=guid=01a3a8ca-7aec-11d1-a31b-0020af9fbb1c
SuperClass=text=dmaClass_Containable
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

[dmaClass_DirectContainmentRelationship]
ClassId=guid=01a3a8cd-7aec-11d1-a31b-0020af9fbb1c
SuperClass=text=dmaClass_ContainmentRelationship
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_Head=guid=045c3750-7aad-11d1-85e5-00a024e8a452
dmaProp_Tail=guid=045c374d-7aad-11d1-85e5-00a024e8a452

[dmaClass_ReferentialContainmentRelationship]
ClassId=guid=01a3a8ce-7aec-11d1-a31b-0020af9fbb1c
SuperClass=text=dmaClass_ContainmentRelationship
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_Head=guid=045c3750-7aad-11d1-85e5-00a024e8a452
dmaProp_Tail=guid=045c374d-7aad-11d1-85e5-00a024e8a452

[edmClass_IndependentPersistence]
ClassId=guid=bb683094-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_DMA
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452

[edmClass_Struct]
ClassId=guid=bb6830c1-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_DMA
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452

[edmClass_ContainerVersion]
```

```
ClassId=guid=bb683085-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_Versionable
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

[edmClass_VersionTraceableContainmentRelationship]
ClassId=guid=bb683086-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_ContainmentRelationship
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_Head=guid=045c3750-7aad-11d1-85e5-00a024e8a452
dmaProp_Tail=guid=045c374d-7aad-11d1-85e5-00a024e8a452
edmProp_VTMode=guid=bb6830a4-0bf0-11d2-9a68-0000e20838e7
edmProp_VTVersionSeries=guid=bb6830a3-0bf0-11d2-9a68-0000e20838e7
edmProp_VTConfigurationHistory=guid=bb6830ac-0bf0-11d2-9a68-0000e20838e7

[edmClass_ContentSearch]
ClassId=guid=bb683083-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_DMA
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
edmProp_Content=guid=bb68309a-0bf0-11d2-9a68-0000e20838e7
edmProp_DocVersionOIID=guid=bb68309b-0bf0-11d2-9a68-0000e20838e7

[edmClass_VersionTracedDocVersion]
ClassId=guid=bb6830c9-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_DocVersion
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

[edmClass_ComponentDocVersion]
ClassId=guid=bb6830ca-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_DocVersion
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

[edmClass_VersionTracedComponentDocVersion]
ClassId=guid=bb6830cb-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_DocVersion
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

[edmClass_VersionTraceableContainer]
ClassId=guid=bb6830cc-0bf0-11d2-9a68-0000e20838e7
SuperClass=text=dmaClass_Container
dmaProp_OIID=guid=045c376f-7aad-11d1-85e5-00a024e8a452
dmaProp_This=guid=045c37ac-7aad-11d1-85e5-00a024e8a452
dmaProp_Parent=guid=045c3773-7aad-11d1-85e5-00a024e8a452
dmaProp_ParentContainer=guid=045c3776-7aad-11d1-85e5-00a024e8a452

;; dmaProp_OIID
[045c376f-7aad-11d1-85e5-00a024e8a452]
dmaProp_DataType=int=DMA_DATATYPE_STRING
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; dmaProp_This
[045c37ac-7aad-11d1-85e5-00a024e8a452]
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_RequiredClass=obj=NULL
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; dmaProp_Parent
[045c3773-7aad-11d1-85e5-00a024e8a452]
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_RequiredClass=obj=NULL
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; dmaProp_ParentContainer
```

```
[045c3776-7aad-11d1-85e5-00a024e8a452]
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_RequiredClass=obj=NULL
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; dmaProp_Head
[045c3750-7aad-11d1-85e5-00a024e8a452]
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_RequiredClass=obj=NULL
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; dmaProp_Tail
[045c374d-7aad-11d1-85e5-00a024e8a452]
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_RequiredClass=obj=NULL
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; edmProp_VTMode
[bb6830a4-0bf0-11d2-9a68-0000e20838e7]
dmaProp_DataType=int=DMA_DATATYPE_INTEGER32
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; edmProp_VTVersionSeries
[bb6830a3-0bf0-11d2-9a68-0000e20838e7]
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_RequiredClass=obj=NULL
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; edmProp_VTConfigurationHistory
[bb6830ac-0bf0-11d2-9a68-0000e20838e7]
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_RequiredClass=obj=NULL
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; edmProp_Content
[bb68309a-0bf0-11d2-9a68-0000e20838e7]
dmaProp_DataType=int=DMA_DATATYPE_STRING
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; edmProp_DocVersionOIID
[bb68309b-0bf0-11d2-9a68-0000e20838e7]
dmaProp_DataType=int=DMA_DATATYPE_STRING
dmaProp_Cardinality=int=DMA_CARDINALITY_SINGLE

;; end of file
```

4.11　ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイル

この節では，ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルについて説明します。

ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルは，DocumentBroker で提供されているクラスおよびプロパティ以外にユーザクラスおよびユーザプロパティを定義して，ユーザアプリケーションプログラムを開発する場合に作成してください。

4.11.1　ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの概要

ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルは，ユーザ定義識別子ファイルの作成コマンド（EDMCreateIds）を実行して作成します。ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルには，次の内容が出力されます。

- **ユーザ定義識別子ファイル**
  ユーザ定義のクラス，およびプロパティのそれぞれに対応するユーザ定義識別子が出力されます。
- **ユーザ定義識別子変数ファイル**
  ユーザ定義のクラス，およびプロパティに対応するユーザ定義識別子を設定した変数の宣言が出力されます。

ユーザ定義識別子ファイルの作成コマンド（EDMCreateIds）については，「7.3　コマンドの文法」を参照してください。

4.11.2　ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力形式

ここでは，ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力形式について説明します。

ユーザ定義識別子ファイルに出力される情報を次に示します。

● ユーザが定義したサブクラス

● ユーザが定義したプロパティ

4.11.3　ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力例

ユーザ定義識別子ファイルの作成コマンド（EDMCreateIds）の実行時に指定したオプションごとに出力される，ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力例を次に示します。ユーザ定義識別子ファイルの作成コマンド（EDMCreateIds）については，「7.3　コマンドの文法」を参照してください。

（1）-i オプションだけを指定した場合のユーザ定義識別子ファイルの出力例

-i オプションにユーザ定義識別子ファイル名「userids.h」を指定した場合のコマンドの入力例とユーザ定義識別子ファイルの出力例を次に示します。

コマンドの入力例

```
$ EDMCreateIds -i userids.h
```

ユーザ定義識別子ファイル（userids.h）の出力例

```
#ifndef _USERIDS_H
#define _USERIDS_H
#define usrClass_ManualVal ¥
   {0x01A3A891,0x7AEC,0x11D1, ¥
    {0xA3,0x1B,0x00,0x20,0xAF,0x9F,0xBB,0x1C}}
#endif
```

(2) -c オプションの指定値ごとのユーザ定義識別子変数ファイルの出力例

-c オプションの指定値ごとのユーザ定義識別子変数ファイルの出力例について説明します。なお，ユーザ定義識別子ファイルは，-i オプションだけを指定した場合と同じ内容が出力されます。ユーザ定義識別子ファイルの出力例については，「(1) -i オプションだけを指定した場合のユーザ定義識別子ファイルの出力例」を参照してください。

(a) -c オプションに「A」を指定した場合のユーザ定義識別子変数ファイルの出力例

-i オプションにユーザ定義識別子ファイル名「userids.h」，-v オプションにユーザ定義識別子変数ファイル名「useridvar.h」，-c オプションに「A」を指定した場合のコマンドの入力例およびユーザ定義識別子変数ファイルの出力例を次に示します。

コマンドの入力例

```
$ EDMCreateIds -i userids.h -v useridvar.h -c A
```

文字コードセットが char に対応したユーザ定義識別子変数ファイル（useridvar.h）の出力例

```
#ifndef _USERIDVAR_H
#define _USERIDVAR_H
#if !defined(DMATYPESA_H)
#   include <dmatypesA.h>
#endif
#if !defined(DMACOM_H)
#   include <dmacom.h>
#endif
#if !defined(DMAIFACEA_H)
#   include <dmaifaceA.h>
#endif
#if !defined(DMAIDS_H)
#   include <dmaids.h>
#endif
#if !defined(_USERIDS_H)
#   include "userids.h"
#endif

#if defined(EDMAPP_INIT_ID)
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_ = x_##Val;
#else
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_;
#endif

EDMAPP_DECL_ID(usrClass_Manual)

#endif
```

(b) -c オプションに「B」を指定した場合のユーザ定義識別子変数ファイルの出力例

-i オプションにユーザ定義識別子ファイル名「userids.h」，-v オプションにユーザ定義識別子変数ファイル名「useridvar.h」，-c オプションに「B」を指定した場合のコマンドの入力例およびユーザ定義識別子変数ファイルの出力例を次に示します。

コマンドの入力例

```
$ EDMCreateIds -i userids.h -v useridvar.h -c B
```

文字コードセットが wchar_t に対応したユーザ定義識別子変数ファイル（useridvar.h）の出力例

```
#ifndef _USERIDVAR_H
#define _USERIDVAR_H
#if !defined(DMATYPES_H)
#   include <dmatypes.h>
#endif
#if !defined(DMACOM_H)
#   include <dmacom.h>
#endif
#if !defined(DMAIFACE_H)
#   include <dmaiface.h>
#endif
#if !defined(DMAIDS_H)
#   include <dmaids.h>
#endif
#if !defined(_USERIDS_H)
#   include "userids.h"
#endif

#if defined(EDMAPP_INIT_ID)
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_ = x_##Val;
#else
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_;
#endif

EDMAPP_DECL_ID(usrClass_Manual)

#endif
```

(c) -c オプションに「T」を指定した場合のユーザ定義識別子変数ファイルの出力例

-i オプションにユーザ定義識別子ファイル名「userids.h」，-v オプションにユーザ定義識別子変数ファイル名「useridvar.h」，-c オプションに「T」を指定した場合のコマンドの入力例およびユーザ定義識別子変数ファイルの出力例を次に示します。

コマンドの入力例

```
$ EDMCreateIds -i userids.h -v useridvar.h -c T
```

文字コードセットが char および wchar_t に対応したユーザ定義識別子変数ファイル（useridvar.h）の出力例

```
#ifndef _USERIDVAR_H
#define _USERIDVAR_H
#if !defined(DMATYPEST_H)
#   include <dmatypesT.h>
#endif
#if !defined(DMACOM_H)
#   include <dmacom.h>
#endif
#if !defined(DMAIFACET_H)
```

```
#   include <dmaifaceT.h>
#endif
#if !defined(DMAIDS_H)
#   include <dmaids.h>
#endif
#if !defined(_USERIDS_H)
#   include "userids.h"
#endif

#if defined(EDMAPP_INIT_ID)
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_ = x_##Val;
#else
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_;
#endif

EDMAPP_DECL_ID(usrClass_Manual)

#endif
```

（3）-i オプション，-v オプション，-c オプション，および -p オプションを指定した場合のユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力例

-i オプションにユーザ定義識別子ファイル名「userids.h」，-v オプションにユーザ定義識別子変数ファイル名「useridvar.h」，-c オプションに「T」，および -p オプションに「Pre_」を指定した場合のコマンドの入力例，ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイルの出力例を次に示します。

コマンドの入力例

```
$ EDMCreateIds -i userids.h -v useridvar.h -c T -p Pre_
```

ユーザ定義識別子ファイル（userids.h）の出力例

```
#ifndef _USERIDS_H
#define _USERIDS_H
#define Pre_usrClass_ManualVal ¥
   {0x01A3A891,0x7AEC,0x11D1, ¥
    {0xA3,0x1B,0x00,0x20,0xAF,0x9F,0xBB,0x1C}}
#endif
```

文字コードセットが char および wchar_t に対応したユーザ定義識別子変数ファイル（useridvar.h）の出力例

```
#ifndef _USERIDVAR_H
#define _USERIDVAR_H
#if !defined(DMATYPEST_H)
#   include <dmatypesT.h>
#endif
#if !defined(DMACOM_H)
#   include <dmacom.h>
#endif
#if !defined(DMAIFACET_H)
#   include <dmaifaceT.h>
#endif
#if !defined(DMAIDS_H)
#   include <dmaids.h>
#endif
#if !defined(_USERIDS_H)
#   include " userids.h "
#endif

#if defined(EDMAPP_INIT_ID)
```

```
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_ = x_##Val;
#else
#  define EDMAPP_DECL_ID(x_) DMA_EXTERN_C DmaId x_;
#endif

EDMAPP_DECL_ID(Pre_usrClass_Manual)

#endif
```

4.12 ファイル転送サービス環境定義ファイル（ftpsv.ini）

この節では，ファイル転送サービス環境定義ファイルについて説明します。

ファイル転送サービス環境定義ファイルには，次のサンプルファイルが提供されています。ファイル転送機能を使用する場合，環境に応じたサンプルファイルを編集してください。

FtpSvSetup コマンドで指定した実行環境ディレクトリ /etc/ftpsv.ini

4.12.1 ファイル転送サービス環境定義ファイルの概要

DocumentBroker サーバに接続する DocumentBroker クライアントが異なるマシン上に存在する場合，DocumentBroker サーバと DocumentBroker 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。

ファイル転送サービス環境定義ファイルは，同時に割り当て可能なクライアントの数の最大値など，ファイル転送機能を使用するために必要な環境を定義するファイルです。

4.12.2 ファイル転送サービス環境定義ファイルの記述形式

ここでは，ファイル転送サービス環境定義ファイルの記述形式について説明します。

ファイル転送サービス環境定義ファイルは，次に示す二つのセクションと，各セクションに指定するエントリによって構成されます。

- [FtpService] セクション
- [FtpProcessXXXX] セクション

セクションとエントリの記述規則を次に示します。

- 印刷可能な ASCII コードで記述します。
- セクション名は，[ ]（角括弧）で囲んで指定します。一つのセクションは，セクション名を指定してから，次のセクション名を指定するまで，またはファイルの終端までの範囲です。
- エントリは，「エントリ名 = 指定値」の形式で指定します。
- 「;」（セミコロン）で始まる行はコメント行として扱われます。

記述形式を次に示します。

[< セクション名 >]
< エントリ名 >=< 値 >

なお，同一名のセクションを複数指定した場合，最初に指定したセクションが有効になります。また，指定できないセクションを指定した場合，その指定は無視されます。

各セクション内で同一名のエントリを複数指定した場合，最初に指定したエントリが有効になります。また，各エントリの値として指定できるのは，1,023 バイトまでです。

以降，ファイル転送サービス環境定義ファイルを構成する各セクションと，セクションごとに指定するエントリについて説明します。

（1）[FtpService] セクション

FtpSessionMax エントリ

同時に割り当て可能なクライアントの数の最大値を，1 ～ 1,024 の値で指定してください。静的モードの場合は，ファイル転送サービス起動コマンドで起動したすべてのファイル転送サービスプロセスで，同時に割り当て可能なクライアントの最大数を指定してください。動的モードの場合は，一つのファイル転送サービスプロセスで同時に割り当て可能なクライアントの最大数を指定してください。
このエントリを省略した場合，またはエントリに不正な値を指定した場合は，「64」が仮定されます。
なお，クライアントへの割り当ては，クライアントアプリケーションでファイル転送を要求してから，文書空間への接続解除まで有効です。

FtpOrbBoaOption エントリ

ファイル転送サービス監視プロセスの ORB および BOA のオプションを指定します。指定を省略した場合，「-OAthreadMax 64 -OAlocalipc 0」が仮定されます。ORB および BOA のオプション以外の値を指定した場合，ファイル転送サービス監視プロセスの起動に失敗します。指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。
なお，このエントリは，TPBroker V3 環境の場合に指定するエントリです。TPBroker V5 と連携して動作する環境では指定できません。TPBroker V5 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V5 環境では，VisiBroker プロパティとして FtpVBProperty エントリに指定してください。

FtpVBProperty エントリ

ファイル転送サービス監視プロセスの VisiBroker プロパティを指定します。指定を省略した場合，「-Dvbroker.se.iiop_tp.scm.iiop_tp.manager.type=Socket
-Dvbroker.se.iiop_tp.scm.iiop_tp.dispatcher.threadMax=64」が仮定されます。VisiBroker プロパティ以外の値を指定した場合，ファイル転送サービス監視プロセスの起動に失敗します。指定できるプロパティについては，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。
なお，このエントリは，TPBroker V5 環境の場合に指定するエントリです。TPBroker V3 と連携して動作する環境では指定できません。TPBroker V3 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V3 環境では，ORB ／ BOA オプションとして FtpOrbBoaOption エントリに指定してください。

FtpProcessOrbBoaOption エントリ

すべてのファイル転送サービスプロセスに共通する ORB および BOA のオプションを指定します。指定を省略した場合，「-OAthreadMax 64 -OAlocalipc 0」が仮定されます。ORB および BOA のオプション以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。
なお，このエントリは，TPBroker V3 環境の場合に指定するエントリです。TPBroker V5 と連携して動作する環境では指定できません。TPBroker V5 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V5 環境では，VisiBroker プロパティとして FtpProcessVBProperty エントリに指定してください。

FtpProcessVBProperty エントリ

すべてのファイル転送サービスプロセスに共通する VisiBroker プロパティを指定します。指定を省略した場合，「-Dvbroker.se.iiop_tp.scm.iiop_tp.manager.type=Socket
-Dvbroker.se.iiop_tp.scm.iiop_tp.dispatcher.threadMax=64」が仮定されます。VisiBroker プロパティ以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。指定できるプロパティについては，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)

プログラマーズリファレンス」を参照してください。
なお，このエントリは，TPBroker V5 環境の場合に指定するエントリです。TPBroker V3 と連携して動作する環境では指定できません。TPBroker V3 環境で指定した場合は，指定が無視され省略値も有効になりません。TPBroker V3 環境では，ORB ／ BOA オプションとして FtpProcessOrbBoaOption エントリに指定してください。

(2) [FtpProcessXXXX] セクション

XXXX は，0001 ～ 0020 を示します。[FtpProcessXXXX] セクションでは，静的モードで開始される各ファイル転送サービスプロセスに固有の動作を定義します。

ファイル転送サービス開始コマンド（FtpSvStart）の -n オプションに指定した数までのセクションが有効になります。ファイル転送サービスプロセスごとに動作を定義することで，ポート番号を指定してファイル転送サービスプロセスを複数起動できます。例えば，ファイル転送サービス開始コマンドの -n オプションに 2 を指定した場合，[FtpProcess0001] セクションおよび [FtpProcess0002] セクションの定義内容が有効になります。

[FtpProcessXXXX] セクションでは，次のエントリを指定できます。

OrbBoaOption エントリ

各ファイル転送サービスプロセスに固有の ORB および BOA のオプションを指定します。指定を省略した場合に仮定される値はありません。ORB および BOA のオプション以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。
[FtpService] セクションの FtpProcessOrbBoaOption エントリの値と，このエントリの値を連結した値が，ORB および BOA のオプションになります。指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。
なお，このエントリは，TPBroker V3 環境の場合に指定するエントリです。TPBroker V5 と連携して動作する環境では指定できません。TPBroker V5 環境で指定した場合は，指定が無視されます。TPBroker V5 環境では，VisiBroker プロパティとして VBProperty エントリに指定してください。

VBProperty エントリ

各ファイル転送サービスプロセスに固有の VisiBroker プロパティを指定します。指定を省略した場合に仮定される値はありません。VisiBroker プロパティ以外の値を指定した場合，ファイル転送サービスプロセスの起動に失敗します。
[FtpService] セクションの FtpProcessVBProperty エントリの値と，このエントリの値を連結した値が，VisiBroker プロパティになります。指定できるプロパティについては，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。
なお，このエントリは，TPBroker V5 環境の場合に指定するエントリです。TPBroker V3 と連携して動作する環境では指定できません。TPBroker V3 環境で指定した場合は，指定が無視されます。TPBroker V3 環境では，ORB ／ BOA オプションとして OrbBoaOption エントリに指定してください。

4.12.3 ファイル転送サービス環境定義ファイルの記述例

TPBroker V3 と連携して動作する環境の場合

ファイル転送サービス開始コマンドの -n オプションに「2」を指定し，個々のファイル転送サービスプロセスに BOA が使用するポート番号を定義する場合の，ファイル転送サービス環境定義ファイルの記述例を次に示します。

```
[FtpProcess0001]
OrbBoaOption = -OAport 14006

[FtpProcess0002]
OrbBoaOption = -OAport 14007
```

TPBroker V5 と連携して動作する環境の場合

ファイル転送サービス開始コマンドの -n オプションに「2」を指定し，個々のファイル転送サービスプロセスに POA が使用するポート番号を定義する場合の，ファイル転送サービス環境定義ファイルの記述例を次に示します。

```
[FtpProcess0001]
VBProperty = -Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=14006

[FtpProcess0002]
VBProperty = -Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=14007
```

4.13 サービスプロセス定義ファイル

この節では，サービスプロセス定義ファイルについて説明します。

サービスプロセス定義ファイルは，" 実行環境ディレクトリ /etc/process.ini" に提供されています。

4.13.1 サービスプロセス定義ファイルの概要

サービスプロセス定義ファイルは，サービスプロセスごとに動作を定義するファイルです。例えば，サービスプロセスごとにポート番号を指定して，起動できるようになります。

なお，サービスプロセスごとに動作を定義しない場合，サービスプロセス定義ファイルを編集する必要はありません。また，サービスプロセス定義ファイルが存在しない場合，DocumentBroker サーバは，サービスプロセスごとに動作を定義しないものとして動作します。

4.13.2 サービスプロセス定義ファイルの記述形式

ここでは，サービスプロセス定義ファイルの記述形式について説明します。

サービスプロセス定義ファイルは，次に示す二つのセクションと，セクションに指定するエントリによって構成されます。

- [Entry0001] セクション
- [ProcessXXXX] セクション

セクションとエントリの記述規則を次に示します。

- 印刷可能な ASCII コードで記述します。
- セクション名は，[ ]（角括弧）で囲んで指定します。一つのセクションは，セクション名を指定してから，次のセクション名を指定するまで，またはファイルの終端までの範囲です。
- エントリは，「エントリ名 = 指定値」の形式で指定します。
- 「;」（セミコロン）で始まる行はコメント行として扱われます。

記述形式を次に示します。

[< セクション名 >]
< エントリ名 >=< 値 >

なお，同一名のセクションを複数指定した場合，最初に指定したセクションが有効になります。また，指定できないセクションを指定した場合，その指定は無視されます。

各セクション内で同一名のエントリを複数指定した場合，最初に指定したエントリが有効になります。また，各エントリの値として指定できるのは，1,023 バイトまでです。

以降，サービスプロセス定義ファイルを構成する各セクションと，セクションごとに指定するエントリについて説明します。

（1） [Entry0001] セクション

[Entry0001] セクションでは，サービスプロセスの動作を定義します。

このセクション名は必ず記述してください。このセクションは，[ProcessXXXX] セクションより前に記述してください。

DBConnectionClose エントリ

DocumentBroker サーバを停止した時に，明示的に DB コネクションを切断するかどうかを指定します。

- Yes
  DB コネクションを切断します。
- NO
  DB コネクションを切断しません。

指定を省略した場合，またはファイルが存在しない場合，「NO」が仮定されます。

(2) [ProcessXXXX] セクション

XXXX は，0001 ～ 0020 を示します。[ProcessXXXX] セクションでは，各サービスプロセスに固有の動作を定義します。

DocumentSpace 構成定義ファイルの Process エントリに指定した数までのセクションが有効になります。例えば，Process エントリに「2」を指定した場合，[Process0001] セクションおよび [Process0002] セクションの定義内容が有効になります。

[ProcessXXXX] セクションでは，次のエントリを指定できます。

OrbBoaOption エントリ

各サービスプロセスに固有の ORB および BOA のオプションを指定します。指定を省略した場合に仮定される値はありません。ORB および BOA のオプション以外の値を指定した場合，DocumentBroker サーバの起動に失敗します。DocumentSpace 構成定義ファイルの ProcessOrbBoaOption エントリの値と，このエントリの値を連結した値が，ORB および BOA のオプションになります。指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。

なお，このエントリは，TPBroker V3 環境の場合に指定するエントリです。TPBroker V5 と連携して動作する環境では指定できません。TPBroker V5 環境で指定した場合は，指定が無視されます。TPBroker V5 環境では，VisiBroker プロパティとして VBProperty エントリに指定してください。

VBProperty エントリ

各サービスプロセスに固有の VisiBroker プロパティを指定します。指定を省略した場合に仮定される値はありません。VisiBroker プロパティ以外の値を指定した場合，DocumentBroker サーバの起動に失敗します。DocumentSpace 構成定義ファイルの ProcessVBProperty エントリの値とこのエントリの値を連結した値が，VisiBroker プロパティになります。指定できるプロパティについては，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。

なお，このエントリは，TPBroker V5 環境の場合に指定するエントリです。TPBroker V3 と連携して動作する環境では指定できません。TPBroker V3 環境で指定した場合は，指定が無視されます。TPBroker V3 環境では，ORB ／ BOA オプションとして OrbBoaOption エントリに指定してください。

4.13.3 サービスプロセス定義ファイルの記述例

TPBroker V3 と連携して動作する環境の場合

DocumentSpace 構成定義ファイルの Process エントリに「2」を指定し，個々のサービスプロセスに BOA が使用するポート番号を定義する場合の，サービスプロセス定義ファイルの記述例を次に示します。

```
[Entry0001]

[Process0001]
OrbBoaOption = -OAport 14005

[Process0002]
OrbBoaOption = -OAport 14006
```

TPBroker V5 と連携して動作する環境の場合

DocumentSpace 構成定義ファイルの Process エントリに「2」を指定し，個々のサービスプロセスに POA が使用するポート番号を定義する場合の，サービスプロセス定義ファイルの記述例を次に示します。

```
[Entry0001]

[Process0001]
VBProperty = -Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=14005

[Process0002]
VBProperty = -Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=14006
```

4.14 プロパティマッピング定義ファイル

この節では，プロパティマッピング定義ファイルについて説明します。

プロパティマッピング定義ファイルは，XML 文書管理機能を使用する場合に作成してください。

4.14.1 プロパティマッピング定義ファイルの概要

XML 文書管理機能を利用するためには，次のプログラムに必要な定義ファイルを作成して，実行時に指定する必要があります。

- HiRDB Adapter for XML
- Preprocessing Library for Text Search

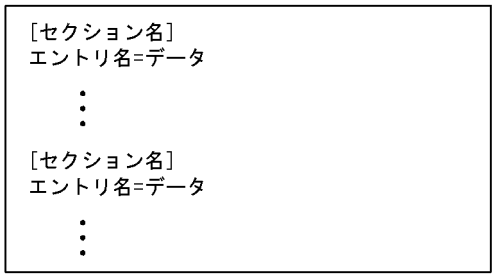
プロパティマッピング定義ファイルは，XML 文書管理機能で使用する定義ファイルに含まれるファイルの一つで，XML 文書中のタグ名とその内容をマッピングするクラス名，プロパティ名の対応関係の定義（プロパティマッピング定義）を記述したファイルです。この定義ファイルを基に，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）がマッピング定義ファイルを生成します。

4.14.2 プロパティマッピング定義ファイルの記述形式

ここでは，プロパティマッピング定義ファイルの記述形式について説明します。

プロパティマッピング定義の記述形式を次の図に示します。

図 4-3 プロパティマッピング定義の記述形式



プロパティマッピング定義はセクション形式で指定します。プロパティマッピング定義は複数のセクションで構成され，それぞれのセクションは複数のエントリで構成されます。なお，1 行の長さは，2,048 バイトまで（改行文字を含む）です。

(1) セクションの文法

セクションを記述する場合の文法を次に示します。

セクション名 ::= **文字列**
　予約セクション名：EDM_MAPPING_CLASS_LIST
　その他のセクション名 ::={ **各セクションのエントリで定義したセクション名** }

セクション名に指定できる文字を次に示します。

- 半角英数字
- 半角記号

ただし，「[」（左角括弧）および「]」（右角括弧）を除きます。

セクションの種類は，次に示す 3 種類があります。

- マッピング先クラス一覧セクション
- マッピング先クラス用セクション
- マッピング先 VariableArray 型プロパティ用セクション

以降，それぞれのセクションの文法について説明します。

(a) マッピング先クラス一覧セクション

マッピング先となるクラスを指定するセクションです。セクション名には，EDM_MAPPING_CLASS_LIST を指定してください。このセクションは，プロパティマッピング定義中に 1 か所だけ存在します。

エントリには，マッピング先クラス名と，マッピング内容を記述するセクションの名称（マッピング先クラス用セクション名）を定義します。エントリには使用するすべてのクラスについて定義します。エントリに定義するクラスは，dmaClass_ConfigurationHistory クラスのサブクラスおよび dmaClass_DocVersion クラスのサブクラスです。それぞれ一つずつまたはどちらか一つについて定義します。

エントリの内容を次に示します。

● エントリの内容

マッピング先クラス名 = マッピング先クラス用セクション名

このエントリは，必ず指定してください。

(b) マッピング先クラス用セクション

マッピング先となるプロパティ名とタグ名（属性名）の対応を定義するセクションです。セクション名には，「(a) マッピング先クラス一覧セクション」で定義したマッピング先クラス用セクション名を指定します。使用するすべてのマッピング先クラスについて，このセクションを定義します。エントリには，次に示す内容を定義します。

- カレントタグ階層の指定（EDM_TAG エントリ）
- マッピング先となるプロパティ名とタグ名の対応の定義（マッピング先プロパティ名エントリ）

マッピング先に指定できるプロパティはユーザ追加プロパティだけです。マッピング先が VariableArray 型プロパティの場合，タグ名との対応は別セクションで定義するので，このセクションでは VariableArray 型プロパティ名とそのプロパティ用のセクションの名称（マッピング先 VariableArray 型プロパティ用セクション名）を定義します（VariableArray 型プロパティ名エントリ）。エントリの内容を次に示します。

● エントリの内容

- EDM_TAG= **タグ文字列**
- **マッピング先プロパティ名 =**
  { タグ文字列 | マッピング先 VariableArray **型プロパティ用セクション名 }**
  マッピング先プロパティ名エントリは，必ず指定してください。

(c) マッピング先 VariableArray 型プロパティ用セクション

マッピング先 VariableArray 型プロパティ中の要素プロパティ名とタグ名（属性名）の対応を定義するセクションです。セクション名には「(b) マッピング先クラス用セクション」で定義したマッピング先 VariableArray 型プロパティ用セクション名を指定します。「(b) マッピング先クラス用セクション」で定義したすべてのマッピング先 VariableArray 型プロパティについて，このセクションを定義します。エン

トリには次に示す内容を定義します。

- マッピング対象となるタグ（属性）群（子タグ群）を取りまとめるタグ（親タグ）の定義（EDM_RECORD エントリ）
- カレントタグ階層の指定（EDM_TAG エントリ）
- マッピング先となるプロパティ名とタグ名の対応の定義（マッピング先プロパティ名エントリ）

まず，EDM_RECORD エントリで親タグを定義し，次に各要素プロパティ名とタグ（属性）名の対応を定義します。親タグは，すべての子タグの上位にあり，繰り返し出現するタグである必要があります。なお，EDM_RECORD エントリはセクションの先頭だけに記述できます。エントリの内容を次に示します。

● エントリの内容

- EDM_RECORD= **タグ文字列**
- EDM_TAG= **タグ文字列**
- **マッピング先プロパティ名 = タグ文字列**
  EDM_RECORD エントリおよびマッピング先プロパティ名エントリは必ず指定してください。

(2) エントリの文法

エントリを記述する場合の文法を次に示します。

エントリ名 ::= **文字列**
予約エントリ名：EDM_TAG **および** EDM_RECORD
その他のエントリ名 ::={ **クラス名** | **プロパティ名** }

データ ::={ **タグ文字列** |EDM_RECORD **用タグ文字列** | **セクション名** }

タグ文字列 ::="{ **タグ名** [/ **タグ名**]...[/ **属性名**] | **属性名** }"
次のどちらかの形式で指定します。

- 1 個以上のタグ名，および 0 個または 1 個の属性名を指定します。
- 1 個の属性名を指定します。

タグ名とタグ名との間，およびタグ名と属性名との間は「/」（スラント）で区切ります。

EDM_RECORD **用タグ文字列** ::=" **タグ名** [/ **タグ名**]..."
EDM_RECORD エントリ専用の記述形式です。タグ名は 1 回以上指定します。タグ名とタグ名との間は「/」（スラント）で区切ります。

タグ名 ::= { **文字列** | .}
タグ名または「.」（ピリオド）を指定します。

属性名 ::= @ **文字列**

注意

- エントリ名に指定できる文字はメタ情報のクラス名（edmProp_DisplayName プロパティに指定した値）として指定できる文字です。
- タグ名および属性名に指定できる文字列は，XML の名前文字列として使用できる文字列です。XML の名前文字列は，1 文字目は，半角英字，全角文字（全角数字を除く），「_」（下線文字），または「:」（コロン）です。2 文字目以降は，半角英数字，全角文字（全角数字を除く），「_」（下線文字），「:」（コロン），「.」（ピリオド），または「-」（マイナス記号）です。

エントリの種類は，次に示す 5 種類があります。

- クラス名エントリ

- EDM_TAG エントリ
- EDM_RECORD エントリ
- プロパティ名エントリ
- VariableArray 型プロパティ名エントリ

以降，それぞれのエントリの文法について説明します。

(a) クラス名エントリ

マッピング先クラス一覧セクション内だけで記述できるエントリです。エントリ名でマッピング先クラスを定義すると同時に，データでマッピング内容を記述するセクションの名称を定義します。

クラス名には，次のクラスを指定できます。

- dmaClass_ConfigurationHistory クラスを継承したユーザ追加のサブクラス
- dmaClass_DocVersion クラスを継承したユーザ追加のサブクラス
- edmClass_ComponentDocVersion クラスを継承したユーザ追加のサブクラス
- edmClass_VersionTracedDocVersion クラスを継承したユーザ追加のサブクラス
- edmClass_VersionTracedComponentDocVersion クラスを継承したユーザ追加のサブクラス

指定内容を次に示します。

● 指定内容
クラス名 = セクション名

(b) EDM_TAG エントリ

プロパティ名エントリの解析時にカレントとして扱うタグ階層を指定するエントリです。このエントリ以降に出現するプロパティ名エントリでは，指定されたタグはこのエントリで指定したタグの下の階層に存在するものとして解析されます。マッピング先クラス用セクションとマッピング先 VariableArray 型プロパティ用セクションで記述できます。

このエントリの指定は，次の EDM_TAG エントリが出現するか，セクションが終了するまで有効です。ただし，このエントリの指定後にマッピング先 VariableArray 型プロパティ用セクション名が定義されている場合，そのマッピング先 VariableArray 型プロパティ用セクションには，このエントリの指定内容が引き継がれます。

マッピング先 VariableArray 型プロパティ用セクションでこのエントリを指定した場合，指定したタグ階層は EDM_RECORD エントリで指定したタグの下の階層に存在するものとして解析されます。

このエントリで「.」（ピリオド）だけを指定すると「セクションの基本の EDM_TAG エントリ指定にする」という意味になります。このように指定すると，マッピング先クラス用セクション内ではカレントとして扱うタグ階層がないことを示し，マッピング先 VariableArray 型プロパティ用セクション内では EDM_RECORD エントリの指定タグ階層を示します。指定内容を次に示します。

● 指定内容
EDM_TAG= **タグ文字列**

(c) EDM_RECORD エントリ

VariableArray 型プロパティにマッピングする子タグ群の親の位置に当たる繰り返しタグ（親タグ）を定義するエントリです。VariableArray 型プロパティの配列数は，ここで定義したタグの出現回数になります。このエントリは上位セクションの EDM_TAG エントリの指定を引き継ぐので，定義したタグの前には引き継いできたタグ階層が追加されます（上位セクションの EDM_TAG ＋指定タグ）。マッピング先 VariableArray 型プロパティ用セクションの先頭に必ず記述してください。

このエントリ以降に出現する EDM_TAG エントリおよびプロパティ名エントリでは，指定されたタグは，すべてこのエントリで指定したタグの下の階層に存在するものとして解析されます。このエントリの指定は，セクションが終了するまで有効です。

このエントリを記述する場合の注意事項を次に示します。

- このエントリには属性名および「.」（ピリオド）は指定できません。
- このエントリの指定タグと，上位セクションから引き継いだ EDM_TAG 指定の合計が 2 階層以上必要です。

指定内容を次に示します。

● 指定内容
EDM_RECORD= **タグ文字列**

(d) プロパティ名エントリ

マッピング先プロパティとマッピングするタグ（属性）を定義するエントリです。指定値によってマッピングする対象が異なります。プロパティ名のタグの指定形式について次の表に示します。

表 4-22 プロパティ名のタグの指定形式

タグの指定形式	マッピング対象
タグ名	指定したタグの開始タグおよび終了タグに囲まれた文字列（タグ間のテキスト）をマッピングします。
属性名	指定した属性の値をマッピングします。対象タグは上位の EDM_TAG エントリおよび EDM_RECORD エントリで指定したタグです。
「.」（ピリオド）	タグ間テキストをマッピングします。対象タグは上位の EDM_TAG エントリおよび EDM_RECORD エントリで指定したタグです。

「(b) EDM_TAG エントリ」および「(c) EDM_RECORD エントリ」で説明したように，プロパティ名エントリで指定したタグは，これらの二つのエントリが上位に存在するかどうかで解析のしかたが変わります。上位エントリによる指定タグの解析の違いについて次の表に示します。

表 4-23 上位エントリによる指定タグの解析の違い

上位エントリ	解析結果および説明
EDM_TAG エントリ有り	EDM_TAG **＋指定タグ** **説明** このエントリで指定したタグの前に EDM_TAG エントリで指定したタグ階層を追加して解析します。
EDM_RECORD エントリ有り	**上位セクションの** EDM_TAG ＋ EDM_RECORD **＋指定タグ** **説明** このエントリで指定したタグの前に EDM_RECORD エントリで指定したタグを追加して解析します。
両方有り	**上位セクションの** EDM_TAG ＋ EDM_RECORD ＋ EDM_TAG **＋指定タグ** **説明** このエントリで指定したタグの前に EDM_RECORD エントリで指定したタグと EDM_TAG エントリで指定したタグを追加して解析します。
両方なし	**指定タグだけ** **説明** このエントリで指定したタグをそのまま解析します。解析結果にタグ名は必要なので，この場合，属性名だけの指定や，「.」（ピリオド）の指定はできません。

このエントリはマッピング先クラス用セクションとマッピング先 VariableArray 型プロパティ用セクションで記述できます。マッピング先 VariableArray 型プロパティ用セクションでは，EDM_RECORD エントリが上位セクションの EDM_TAG エントリを引き継いでいるので，タグの内容には注意する必要があります。

マッピング先として指定できるプロパティはユーザ追加プロパティだけです。ただし，マッピング対象となる属性にはマッピング元 XML タグ定義（DCD）によってデータ型が設定されているので，プロパティと属性のデータ型の組み合わせによってはマッピングできない場合があります。マッピングできるデータ型の組み合わせを次の表に示します。

表 4-24　マッピングできるデータ型の組み合わせ

プロパティのデータ型	マッピング対象のマッピング元 XML タグ定義（DCD）データ型
DMA_DATATYPE_INTEGER32	string,number,int,fixed,decimal,i1,i2,i4,ui1,ui2,ui4,fixed14.4（値が整数値を表す文字列であること）
DMA_DATATYPE_BOOLEAN	同上（ただし値が '0' または '1' であること）
DMA_DATATYPE_STRING	entity,entities,bin.hex,bin.base64 以外の型

注 1
タグ間テキストのマッピング元 XML タグ定義（DCD）データ型は string 型です。

注 2
マッピング元 XML タグ定義（DCD）データ型の boolean 型は，DMA_DATATYPE_STRING 型にだけマッピングできます。

指定内容を次に示します。

● 指定内容
プロパティ名 = タグ文字列

(e) VariableArray 型プロパティ名エントリ

マッピング先 VariableArray 型プロパティのマッピング内容を記述するセクションのセクション名を定義するエントリです。マッピング先クラス用セクション内だけで記述できます。指定内容を次に示します。

● 指定内容
プロパティ名 = セクション名

(3) コメントの文法

「;」（セミコロン）で始まる行はコメント行として扱われます。

4.14.3 プロパティマッピング定義ファイルの記述例

次の図に示す XML 文書を例にして，オブジェクトへのプロパティマッピングをする場合のプロパティマッピング定義の方法を示します。

図 4-4　XML 文書のタグとオブジェクトに指定するプロパティの値との関連

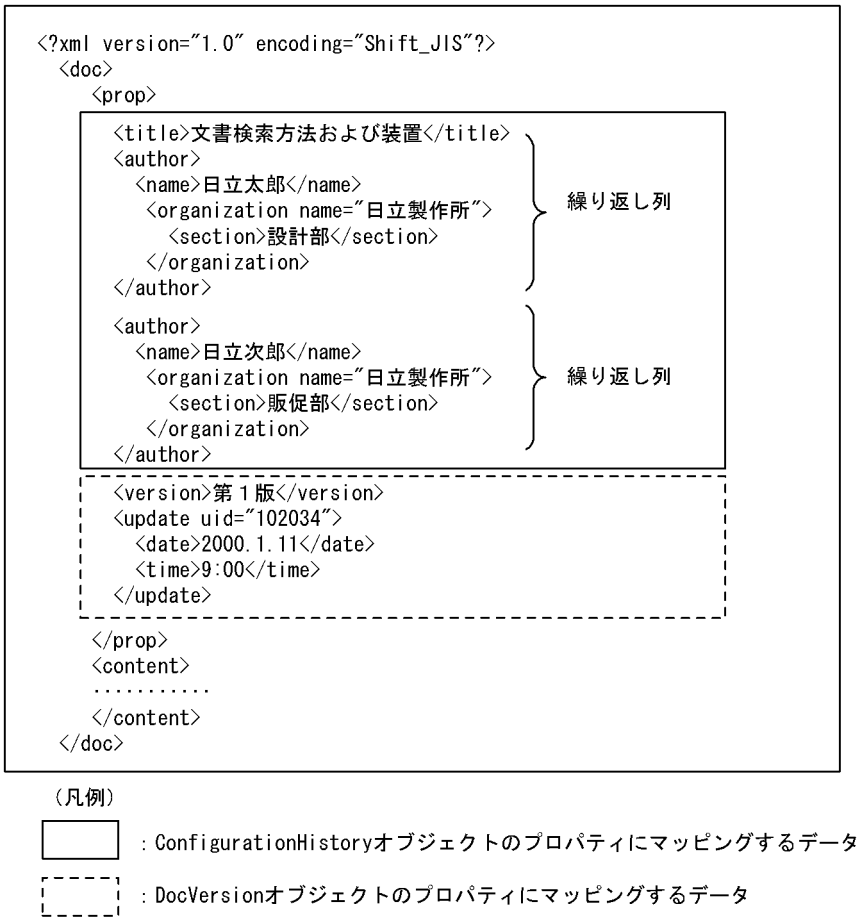


図 4-4 の例で示したプロパティの値を設定するユーザクラスおよびユーザプロパティと，プロパティマッピング定義ファイルの定義例を次の図に示します。

図 4-5 プロパティの値を設定するユーザクラスおよびユーザプロパティとプロパティマッピング定義ファイルの定義例

プロパティの値を設定するユーザクラスおよびユーザプロパティ

・マッピング先に指定するユーザ追加サブクラス
　usrClass_CHクラス（dmaClass_ConfigurationHistoryクラスを継承）
　usrClass_DVクラス（dmaClass_DocVersionクラスを継承）

・usrClass_CHクラスでマッピングに使用するプロパティ
　usrProp_Titleプロパティ（String型）
　usrProp_Authorプロパティ（VariableArray型）
　　usrProp_Nameプロパティ（String型）
　　usrProp_organizationプロパティ（String型）
　　usrProp_Sectionプロパティ（String型）

・usrClass_DVクラスでマッピングに使用するプロパティ
　usrProp_Versionプロパティ（String型）
　usrProp_Idプロパティ（Integer32型）
　usrProp_Dateプロパティ（String型）
　usrProp_Timeプロパティ（String型）

・マッピング対象とするタグ属性のDCDデータ型
　organizationタグのname属性（string型）
　updateタグのuid属性（int型）

プロパティマッピング定義ファイルの定義例

```
[EDM_MAPPING_CLASS_LIST]      } マッピング先クラス一覧セクション
usrClass_CH=CH_PROPLIST       } usrClass_CHクラス（ConfigurationHistoryオブジェクト）と
usrClass_DV=DV_PROPLIST       } usrClass_DVクラス（DocVersionオブジェクト）を指定

[CH_PROPLIST]                 } マッピング先クラス用セクション
EDM_TAG="doc/prop"            } 基本のタグはdoc/prop
usrProp_Title="title"         } titleタグをマッピング
usrProp_Author=VA_AUTHORS     } VariableArray型プロパティ用セクションを指定

[DV_PROPLIST]                 } マッピング先クラス用セクション
EDM_TAG="doc/prop/version"    } 基本のタグはdoc/prop/version
usrProp_Version="."           } タグ間テキストをマッピング
EDM_TAG="doc/prop/update"     } EDM_TAGの再設定
usrProp_Id="@uid"             } uid属性をマッピング
usrProp_Date="date"           } dateタグをマッピング
usrProp_Time="time"           } timeタグをマッピング

[VA_AUTHORS]                  } マッピング先VariableArray型プロパティ用セクション
EDM_RECORD="author"           } 基本のタグはdoc/prop/author
usrProp_Name="name"           } nameタグをマッピング
EDM_TAG="organization"        } 基本のタグはdoc/prop/author/organization
usrProp_organization="@name"  } organizationのname属性をマッピング
usrProp_Section="section"     } sectionタグをマッピング
```

4.14.4 プロパティマッピング定義ファイルの注意事項

プロパティマッピング定義ファイルでマッピング元のタグ名（属性名）を指定する場合は，XML 文書のルートタグから記述することをお勧めします。このように記述することによって，プロパティマッピング定義が不明確になることを防げます。

また，EDM_TAG エントリや EDM_RECORD エントリが存在する場合，タグの解析結果がルートタグから始まるように指定してください。

なお，ルートタグから記述しない場合，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）が生成するマッピング定義の内容が冗長になることがあります。

4.15 文書クラス定義ファイル

この節では，文書クラス定義ファイルについて説明します。

文書クラス定義ファイルのサンプルファイルは，"/opt/HiEDMS/sample/EDMclassdef.csv" に提供されています。

4.15.1 文書クラス定義ファイルの概要

文書クラス定義ファイルは，サブクラスおよびプロパティを追加するときに，追加するサブクラスやプロパティの定義を文書クラスとして定義したり，文書空間を構築するためのデータベース容量を見積もるときに使用する情報を定義したりするファイルです。作成したファイルは，文書空間定義コマンド（EDMCDefDocSpace）の -f オプションで指定します。

"/opt/HiEDMS/sample/" に格納されている「EDMclassdef.csv」を，システム管理者が Microsoft(R) Excel などで編集して，CSV 形式のファイルを作成してください。

4.15.2 追加できるクラス，およびプロパティの種類

ここでは，文書クラス定義ファイルで指定できるクラスやプロパティについて説明します。

(1) クラスの種類

文書クラス定義ファイルで定義できるクラスの種類は，次のとおりです。

表 4-25 文書クラス定義ファイルで定義できるクラスの種類

クラス分類	クラスの種類
可変長配列	可変長配列
独立データ	独立データ
コンテナ	バージョンなしコンテナ
	バージョンなし構成管理コンテナ
	バージョン付き構成管理コンテナ
文書	バージョンなし文書
	バージョン付き文書

(2) プロパティを追加できるクラス

文書クラス定義ファイルでプロパティを追加できるクラスは，ユーザが追加したクラス，および次に示す DocumentBroker が提供するクラスです。

DocumentBroker が提供するクラスを文書クラス定義ファイルで指定する場合は，次のクラス名に置き換えて指定します。

表 4-26 文書クラス定義ファイルで指定するクラス名

指定するクラス名	システム提供のクラス名
aclClass_DCR	dmaClass_DirectContainmentRelationship
aclClass_RCR	dmaClass_ReferentialContainmentRelationship
aclClass_VCR	edmClass_VersionTraceableContainmentRelationship

指定するクラス名	システム提供のクラス名
aclClass_PACL	edmClass_PublicACL
aclClass_RS	edmClass_Relationship

(3) プロパティとして定義できるデータ型

文書クラス定義ファイルで追加できるプロパティのデータ型は，次のとおりです。

ただし，全文検索インデクス用プロパティを追加するときには，データ型を指定する必要はありません。

表 4-27 文書クラス定義ファイルで追加できるプロパティのデータ型

追加できるプロパティのデータ型	説明
String	String 型
Integer32	Integer32 型
Boolean	Boolean 型
VArray	VariableArray 型 配列の要素は，クラスの種類（ClassType エントリ）が可変長配列のクラス（Struct）です。

VariableArray 型のプロパティを追加できるクラスは次のとおりです。

- dmaClass_ConfigurationHistory クラスのサブクラス
- dmaClass_Container クラスのサブクラス
- dmaClass_DocVersion クラスのサブクラス
- edmClass_ContainerVersion クラスのサブクラス
- edmClass_IndependentPersistence クラスのサブクラス
- edmClass_VersionTracedDocVersion クラスのサブクラス

VariableArray 型以外のプロパティを追加できるクラスについては，「(2) プロパティを追加できるクラス」を参照してください。

(4) 全文検索機能を使用する場合のプロパティの追加

全文検索機能を使用する場合，全文検索機能を使用する文書クラスに，全文検索インデクス用プロパティを追加する必要があります。文書クラス定義ファイルで指定できる全文検索インデクス用プロパティを次に示します。全文検索インデクス用プロパティを文書クラス定義ファイルで指定する場合は，次のプロパティ名に置き換えて指定します。

表 4-28 文書クラス定義ファイルで指定するプロパティ名

指定するプロパティ名	システム提供のプロパティ名
aclProp_TI	edmProp_TextIndex
aclProp_SI	edmProp_StIndex
aclProp_CTI	edmProp_ConceptTextIndex
aclProp_CSI	edmProp_ConceptStIndex

全文検索機能を使用できるのは，クラスの種類にバージョンなし文書，またはバージョン付き文書を指定した場合です。また，一つのクラスに複数の全文検索インデクス用プロパティは指定できません。

全文検索インデクス用プロパティの詳細については，「2.3.4　全文検索機能付き文書クラスの追加」を参照してください。

全文検索機能を使用する場合に必要なそのほかのプロパティは，文書空間定義コマンド（EDMCDefDocSpace）で自動的に定義されるため，プロパティとして追加する必要はありません。全文検索インデクス用プロパティの指定によって，自動的に定義されるプロパティを次に示します。

表 4-29　文書空間定義コマンド（EDMCDefDocSpace）で自動的に定義されるプロパティ

指定するプロパティ名	自動的に定義されるプロパティ
aclProp_TI	edmProp_Score，edmProp_RawScore，edmProp_DocLength，edmProp_ContentIndexStatus
aclProp_SI	
aclProp_CTI	edmProp_Score，edmProp_RawScore，edmProp_DocLength，edmProp_ContentIndexStatus，edmProp_ScoreConcept
aclProp_CSI	

(5) インデクスを追加できるプロパティ

文書クラス定義ファイルでインデクスを追加できるプロパティは，ユーザが追加した全文検索インデクス用プロパティおよび VariableArray 型以外のプロパティです。インデクスの指定の詳細については，「4.15.4　文書クラス定義ファイルの注意事項」を参照してください。

(6) プロパティを追加する場合の注意事項

文書クラス定義ファイルでは，文字列型プロパティに対する全文検索機能を使用するための全文検索機能付き文字列型プロパティを追加できません。文字列型プロパティに対する全文検索機能を使用する場合は，定義情報ファイルを使用し，メタ情報の追加コマンド（EDMAddMeta）を実行して，全文検索機能付き文字列型プロパティを追加してください。

定義情報ファイルの詳細については，「4.7　定義情報ファイル」を参照してください。メタ情報の追加コマンド（EDMAddMeta）の詳細については，「7.3　コマンドの文法」の「EDMAddMeta（メタ情報の追加）」を参照してください。

4.15.3　文書クラス定義ファイルの記述形式

ここでは，文書クラス定義ファイルの記述形式について説明します。

(1) 記述規則

文書クラス定義ファイルの記述規則を次に示します。

- 印刷可能な ASCII コードで記述します。
- 行の終わりは，<EOF> または行末文字です。
- 行末文字は，<CR> ＋ <LF> です。
- 文書クラス定義ファイルの各エントリ値の先頭には「"」（引用符）を使用できません。
- 行には，ヘッダ，ボディ，およびコメントがあります。値が設定されていない行（コンマだけが続く行）や，行の終わりだけの行は無視します。

(2) 記述形式

文書クラス定義ファイルは，ヘッダ，ボディ，およびコメントによって構成されます。ヘッダ，ボディ，およびコメントの詳細について説明します。

ヘッダ

ヘッダは，エントリの名称を列記する行です。エントリ名称は省略できません。それぞれのエントリの詳細は，「(3) エントリ」を参照してください。

ObjType エントリは先頭（1 列目）に指定してください。ObjType エントリ以外のエントリは，順序を変えることができます。

ヘッダは，ボディより前の行に 1 行だけ指定します。

ボディ

ボディは，クラス，プロパティおよびインデクスの定義を記述する行です。

ボディは，ヘッダのエントリに対応して指定します。例えば，1 番目（1 列目）に ObjType エントリの値を指定し，2 番目（2 列目）に AclibClassName エントリの値を指定します。

一つのクラスを定義する場合，クラス，プロパティ，インデクスの順で定義（行を記述）します。

コメント

コメントは，注釈を記述できる行です。次の場合，コメントとなります。

- **#**（シャープ）で始まる行
- **;**（セミコロン）で始まる行

(3) エントリ

各エントリの詳細について説明します。

ObjType エントリ

オブジェクトタイプに次のどれかを指定します。

- Class

  クラスを定義するとき，ObjType エントリに Class を指定します。

  バージョンなしクラスの定義は，1 行で指定します。

  バージョン付きクラスの定義は，TargetType エントリの値に CH と VR をそれぞれの行に指定します。一つのバージョンだけで使用するクラスで，DmaClassName エントリで指定するデータベースの表識別子となるサブクラス名および GUID エントリで指定する GUID を省略する場合は，VR を指定する行を省略できます。

- Prop

  プロパティを定義するとき，ObjType エントリに Prop を指定します。

  プロパティの定義は，プロパティを指定するクラスの定義よりあとの行に指定します。

  プロパティの定義は，1 行で指定します。

- Index

  インデクスを定義するとき，ObjType エントリに Index を指定します。

  インデクスの定義は，インデクスがキーとするプロパティの定義よりあとの行に指定します。

  インデクスが一つのプロパティをキーとする設定は，1 行で指定します。

  インデクスが複数のプロパティをキーとする設定は，プロパティの数分の行を指定します。インデクスが二つのプロパティをキーとするときの例を，次に示します。

```
#インデクスが二つのプロパティをキーとするときの例
Index,usrClass_DesignDoc,,,,uP_DocName,,ix_DesignDoc,,,,,,,,,
   ,,,,,,,,,,,,,,,TRUE,
Index,usrClass_DesignDoc,,,,uP_DocWriter,,ix_DesignDoc,,,,,,,,,
   ,,,,,,,,,,,,,,,TRUE,
```

ObjType エントリが Index の行を 2 行指定します。

それぞれの IndexName エントリに同じ値※を指定します。例では，ix_DesignDoc を指定しています。

優先順位の高い順にプロパティを指定します。例では，第 1 キー uP_DocName，第 2 キー

uP_DocWriter と指定しています。

注※

複数のプロパティをキーとするインデクスのインデクス名を指定するときは，それぞれの IndexName エントリに同じインデクス名を指定します。

一方，複数のプロパティをキーとするインデクスのインデクス名を自動で作成したいときは，それぞれの IndexName エントリに同じ記号を指定します。記号は，「001AUTO」のように数字（10 けた以内）と文字列 AUTO をつなげた形式で指定します。自動で作成されるインデクス名の詳細については，IndexName エントリを参照してください。

ClassType エントリ

クラスを定義する場合，クラスの種類を指定します。

- Struct
  可変長配列のクラスです。
- IndP
  独立データのクラスです。
- RCntr
  バージョンなしコンテナのクラスです。
- VTCntr
  バージョンなし構成管理コンテナのクラスです。
- CRCntr
  バージョン付き構成管理コンテナのクラスです。
- Doc
  バージョンなし文書のクラスです。
- VDoc
  バージョン付き文書のクラスです。

AclibClassName エントリ

文書クラス定義ファイル中でのクラスを識別するためのクラス名（ACLib クラス名）を指定します。

- ACLib クラス名は，重複しないようにしてください。
- 「dmaClass」，「edmClass」，「aclClass」で始まる文字列は，システムの予約文字列であるため使用できません。
- 指定できる文字列は，英数字，「_」（下線文字），「ー」（ハイフン）です。
- 先頭 1 バイトは，英小文字または英大文字で指定します。
- クラスの定義で DmaClassName エントリの指定を省略する場合，自動で作成するサブクラス名（DmaClassName）が DocumentBroker のサブクラス名に対応する表識別子の規則を満たすように，ACLib クラス名を指定してください。サブクラス名に対応する表識別子の規則については，「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」を参照してください。

  ACLib クラス名を usrClass とすると，次のとおりにデータベースの表識別子となるサブクラス名を作成します。

  ・usrClass
  　バージョンなしクラスの場合

  ・usrClass_CH
  　バージョン付きクラスの場合の全バージョン共通で使用するデータベースの表識別子となるサブクラス名

  ・usrClass_VR
  　バージョン付きクラスの場合の一つのバージョンで使用するデータベースの表識別子となるサブクラス名

- 指定できる長さは，1 ～ 28 バイトです。ただし，クラスの定義で DmaClassName エントリの指定を省略する場合は，データベースの表識別子となるサブクラス名の作成規則に従って，文書空間定義コマンド（EDMCDefDocSpace）で AclibClassName エントリの指定からサブクラス名（DmaClassName）を決定するので，バージョン付きクラスの場合，指定できる長さは 1 ～ 25 バイトとなります。指定する名称は，DocumentBroker のサブクラス名に対応する表識別子の規則に従ってください。サブクラス名に対応する表識別子の規則については，「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」を参照してください。

DmaClassName エントリ

クラスを定義する場合，データベースの表識別子となるサブクラス名を指定します。指定する名称は，DocumentBroker のサブクラス名に対応する表識別子の規則に従ってください。サブクラス名に対応する表識別子の規則については，「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」を参照してください。

TargetType エントリ

バージョン付きクラスを定義する場合，またはバージョン付きクラスにプロパティやインデクスを定義する場合，次のどちらかで対象のデータベースの表識別子となるサブクラス名を指定します。

- CH
  全バージョンで共通に使用するクラス，またはプロパティやインデクスです。
- VR
  一つのバージョンだけで使用するクラス，またはプロパティやインデクスです。

MaxVersionCount エントリ

このエントリにはエントリの値は指定しないで，ヘッダでエントリ列だけ指定してください。

AvVersionCount エントリ

バージョン付きクラスを定義する場合，文書のバージョンの平均数を指定します。値は 1 ～ 2,147,483,647 の範囲で指定してください。

InstanceCount エントリ

コンテナのクラス以外のクラスを定義する場合，オブジェクト数を指定します。値は 1 ～ 2,147,483,647 の範囲で指定してください。

RootCntrCount エントリ，AvCntrCountPerCntr エントリ，AvCntrTreeHeight エントリ，AvRefCountPerCntr エントリ

コンテナのクラスを定義する場合，次の値を設定します。値は 1 ～ 2,147,483,647 の範囲で指定してください。

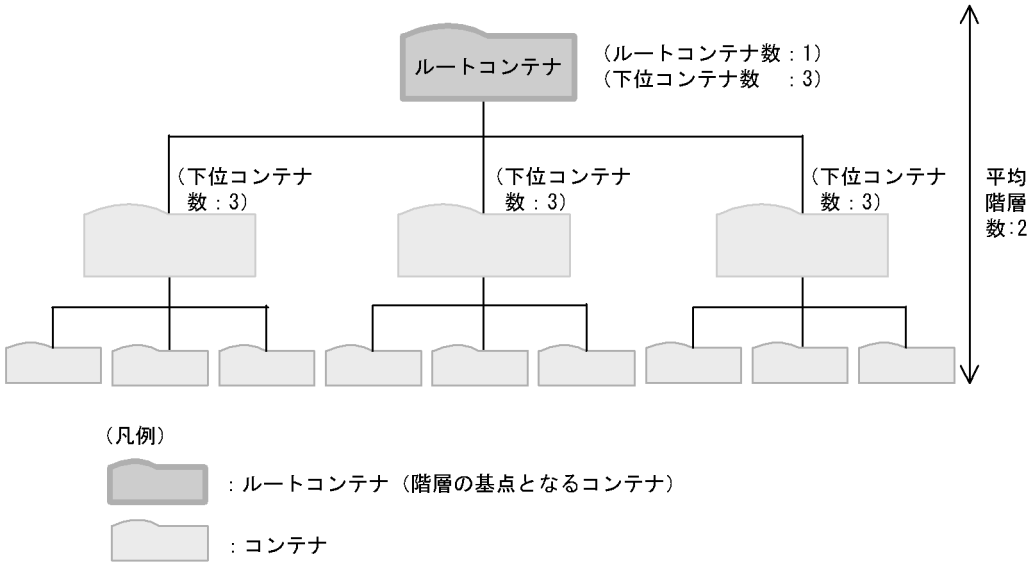
- RootCntrCount エントリ
  コンテナクラスのルートコンテナ数を指定します。
- AvCntrCountPerCntr エントリ
  コンテナごとに格納するコンテナの平均数を指定します。
- AvCntrTreeHeight エントリ
  ルートとなるコンテナからの階層の平均数を指定します。
- AvRefCountPerCntr エントリ
  コンテナごとに参照型で格納するコンテナと文書の平均数を指定します。

RootCntrCount エントリの指定を省略した場合，「1」が仮定されます。ほかのエントリは省略できません。

RootCntrCount エントリ，AvCntrCountPerCntr エントリ，AvCntrTreeHeight エントリの指定例を次に示します。次の図では，RootCntrCount エントリの指定は「1」，AvCntrCountPerCntr エント

リの指定は「3」，AvCntrTreeHeight エントリの指定は「2」となります。

図 4-6 RootCntrCount エントリ，AvCntrCountPerCntr エントリ，AvCntrTreeHeight エントリの指定例



複数のコンテナクラスのオブジェクトで一つの階層構造とする場合は，ユーザでコンテナクラスごとのオブジェクト数を算出し，コンテナクラスごとに AvCntrCountPerCntr エントリの値として設定します。このとき，RootCntrCount エントリおよび AvCntrTreeHeight エントリの値には，最小値（値：1）を設定します。

AvRenditionCount エントリ

バージョンなし文書またはバージョン付き文書のクラスを定義する場合，このエントリにレンディションの平均数を指定します。1 ～ 10 で指定してください。

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，2 ～ 10 で指定してください。ContentType エントリに「Multi」を指定した場合，マルチレンディション機能は使用できません。そのため，このエントリには「1」を指定してください。1 以外を指定した場合は，指定したレンディション数を使用する場合の見積もり（見積もり基礎情報ファイルの出力）結果となるため注意してください。

MaxContentSize エントリ

このエントリにはエントリの値は指定しないで，ヘッダでエントリ列だけ指定してください。

AvContentSize エントリ

バージョンなし文書またはバージョン付き文書のクラスを定義する場合，コンテントの平均サイズ（バイト単位）を指定します。値は 0 ～ 2,147,483,647（バイト）の範囲で指定します。

ContentType エントリの指定によって，このエントリの値が反映される情報が異なります。

ContentType エントリの指定ごとの，見積もり基礎情報ファイルの出力情報の種別（Kind エントリ）を次に示します。

- ContentType エントリが「Single」の場合，指定した値が見積もり基礎情報ファイルの「BLOb」の種別の値に反映されます。
- ContentType エントリが「Multi」の場合，指定した値が見積もり基礎情報ファイルの「BLOb」の種別の値に反映されます。
- ContentType エントリが「Reference」の場合，指定した値が見積もり基礎情報ファイルの「Disk」の種別の値に反映されます。
- ContentType エントリが「FileLink」の場合，指定した値が見積もり基礎情報ファイルの「Disk」

の種別の値に反映されます。

SuperClassType **エントリ**

バージョンなし文書またはバージョン付き文書のクラスを定義する場合，定義するクラスのスーパークラスを次のどちらかから指定します。

- DV
  dmaClass_DocVersion クラスです。
- VTDV
  edmClass_VersionTracedDocVersion クラスです。

指定を省略した場合，「DV」が仮定されます。

PctOfRefDoc **エントリ，**AvRefDocCount **エントリ**

バージョンなし文書またはバージョン付き文書のクラスを定義する場合，PctOfRefDoc エントリに文書間リレーションのリレーション元文書となる文書の割合を指定します。0 ～ 100（%）で指定してください。指定を省略した場合，「0」が仮定されます。

AvRefDocCount エントリにリレーション元文書ごとのリレーション先文書の平均数を指定します。値は 1 ～ 2,147,483,647 の範囲で指定してください。指定を省略した場合，「1」が仮定されます。

ComponentClassName **エントリ，**MaxComponentCount **エントリ，**AvComponentCount **エントリ**

これらのエントリにはエントリの値は指定しないで，ヘッダでエントリ列だけ指定してください。

ContentType **エントリ**

ContentType エントリに，バージョンなし文書またはバージョン付き文書のクラスを定義する場合，使用するレンディションの文書の種別を指定します。このエントリは，表のレコード数の見積もりに使用します。

- Single
  シングルファイル文書を使用するクラスです。
- Multi
  マルチファイル文書を使用するクラスです。
- Reference
  リファレンスファイル文書を使用するクラスです。
- FileLink
  FileLink 文書を使用するクラスです。

指定を省略した場合，「Single」が仮定されます。

定義したクラス中で，種別の異なるコンテントを混在して使用する場合，見積もり基礎情報として出力される表のレコード数は，このエントリで指定された種別に対応する表のレコード数として出力されます。そのため，文書の種別ごとのオブジェクト数を計算して，計算したオブジェクト数を文書の種別に対応する表のレコード数に割り振ってください。レコード数の算出式については，「表 2-11 ユーザ用 RD エリア分のデータベースリソースの所要量」を参照してください。

また，文書のコンテントサイズは文書の種別によって，見積もり基礎情報ファイルの平均文書コンテントサイズの出力情報の種別（Kind エントリ）が異なります。指定した文書の種別によって，該当する出力情報の値として換算してください。文書の種別が異なる文書を混在して使用できる文書の種別は，「Single」，「Reference」および「FileLink」の場合です。

文書の種別と表との対応を次に示します。

- Single：dmaClass_ContentTransfer
  レコード数の算出式のクラス：ContentTransfer クラス
  平均文書コンテントサイズが該当する出力情報の種別：BLOb
- Reference：edmClass_ContentReference

レコード数の算出式のクラス：ContentReference クラス
平均文書コンテントサイズが該当する出力情報の種別：Disk

- FileLink：edmClass_ContentFileLink
  レコード数の算出式のクラス：ContentFileLink クラス
  平均文書コンテントサイズが該当する出力情報の種別：Disk

GUID エントリ

クラスを定義する場合，データベースの表識別子となるサブクラス名の GUID を指定します。
プロパティを定義する場合，プロパティの GUID を指定します。
指定を省略した場合，自動で GUID を割り当てます。
定義済みのプロパティを再定義する場合，このエントリは省略します。
「xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx」（8-4-4-4-12）の形式で指定してください。x は 0 ～ 9，および a ～ f（小文字）で表される 16 進数の値です。A ～ F（大文字）で指定することもできます。このとき大文字と小文字を区別しません。

PropertyName エントリ

プロパティを定義する場合，定義するプロパティ名を指定します。
インデクスを定義する場合，インデクスのキーにするプロパティ名を指定します。
「aclProp」で始まる文字列は，システムの予約文字列であるためプロパティ名に使用できません。指定する名称は，DocumentBroker のプロパティ名に対応する列名の規則に従ってください。プロパティ名に対応する列名の規則については，「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」を参照してください。

DataType エントリ

プロパティのデータ型を指定します。

- String
  プロパティのデータ型は，String 型です。
- Integer32
  プロパティのデータ型は，Integer32 型です。
- Boolean
  プロパティのデータ型は，Boolean 型です。
- VArray
  プロパティのデータ型は，VariableArray 型（配列の要素は可変長配列のクラス）です。

全文検索インデクス用プロパティの場合は，指定する必要はありません。

DefaultValue エントリ

プロパティのデフォルト値を指定します。指定する値には，定義情報ファイルの記述形式の区切り文字は指定できません。定義情報ファイルの記述形式については，「4.7　定義情報ファイル」を参照してください。
プロパティで同一のデフォルト値を設定する場合は，このエントリの値の指定を省略するか，すでに定義されているプロパティのデフォルト値と同じ値を設定してください。
クラスごとにプロパティのデフォルト値を設定する場合は，文書空間情報ファイルの SpPropDefaultValue エントリに「ON」を設定してください。このエントリには，クラスごとのプロパティのデフォルト値を設定してください。
指定を省略した場合，すでに定義されている値がデフォルト値として使用されます。

- DataType エントリが String 型のとき，255 バイト以内の任意の文字列で指定します。デフォルト値を空文字にするには，値を指定しないでください。また，デフォルト値を NULL 値とするには，NULL を指定してください。NULL という文字列はデフォルト値にできません。

- DataType エントリが Integer32 型のとき，-2,147,483,648 ～ 2,147,483,647 の範囲で指定します。
- DataType エントリが Boolean 型のとき，TRUE，FALSE，1，0 のどれかを指定します。
- DataType エントリが VariableArray 型のとき，デフォルト値を指定できません。VariableArray 型のプロパティのデフォルト値は，可変長配列の要素数が 0 の状態です。
- DataType エントリが VariableArray 型以外のとき，デフォルト値の指定を省略すると，DataType エントリの指定によって次のとおりになります。
  ・String 型のとき：空文字
  ・Integer32 型のとき：0
  ・Boolean 型のとき：FALSE

AvLength エントリ，MaxLength エントリ

String 型のプロパティを定義するとき，AvLength エントリに文字列の平均長（バイト単位）を指定し，MaxLength エントリに文字列の最大長（バイト単位）を指定します。値は 1 ～ 32,000 の範囲で指定してください。

StructClassName エントリ，MaxElementCount エントリ，AvElementCount エントリ

VariableArray 型のプロパティを定義するとき，StructClassName エントリに ClassType エントリが Struct の ACLib クラス名を指定し，MaxElementCount エントリに可変長配列の最大要素数を指定し，AvElementCount エントリに可変長配列の平均要素数を指定します。
VariableArrayOfObject オブジェクトの要素を StructClassName エントリに指定したクラスに格納する場合，最大要素数は -1，平均要素数は，1 ～ 2,147,483,647 の範囲で指定してください。
VariableArrayOfObject オブジェクトの要素をこのプロパティに格納する場合，最大要素数は 2 ～ 30,000，平均要素数は 1 ～ 30,000 の範囲で指定してください。
StructClassName エントリに指定する ACLib クラス名のクラスは，ACLib クラス名を使用するプロパティ定義より前に定義してください。

IndexName エントリ

インデクスを定義する場合，インデクス名を指定します。

- 指定を省略した場合，自動でインデクス名を作成します。自動で作成されるインデクス名はクラス名をプリフィクスとし，1 から始まる数値を 32 進数に変換した値をサフィックスとした文書空間で一意となる値となります。
  定義クラス名が「usrClass」の場合，最初のインデクスに自動付与するインデクス名は「usrClass01」となります。
- 指定する名称は，DocumentBroker のインデクス名をデータベース定義の名称に使用する場合の規則に従ってください。インデクス名をデータベース定義の名称に使用する場合の規則については，「4.9.2　インデクス名をデータベース定義の名称に使用する場合の規則」を参照してください。

OrderType エントリ

インデクスを定義する場合，インデクスのキー値の順序に次のどちらかを指定します。

- ASC：昇順
- DESC：降順

指定を省略した場合，「ASC」が仮定されます。

IsUnique エントリ

インデクスを定義する場合，次のどちらかを指定します。なお，複数のプロパティをキーとするときには，インデクスの 1 行目の定義にだけ記述できます。

- TRUE または 1
  キー値を一意に制約します。

- FALSE または 0
  キー値の重複を許します。

指定を省略した場合，「FALSE」が仮定されます。

IsExcept エントリ

インデクスを定義する場合，次のどちらかを指定します。なお，複数のプロパティをキーとするときには，インデクスの 1 行目の定義にだけ記述できます。

- TRUE または 1
  NULL 値だけのキーをインデクスから除外します。
- FALSE または 0
  NULL 値だけのキーもインデクスに追加します。

指定を省略した場合，「FALSE」が仮定されます。

(4) ClassType エントリのクラスごとに指定するエントリの一覧

クラスを定義する場合の ClassType エントリのクラスごとに指定するエントリの一覧を次の表に示します。

表 4-30 ClassType エントリのクラスごとに指定するエントリの一覧

エントリ名	ClassType エントリのクラス （() 内は，TargetType エントリの値）								
	Struct	IndP	RCntr	VTCntr	CRCntr		Doc	VDoc	
					(CH) ※1	(VR) ※1		(CH) ※1	(VR) ※1
AclibClassName	◎	◎	◎	◎	◎※2	◎※2	◎	◎※2	◎※2
DmaClassName	○	○	○	○	○	○	○	○	○
MaxVersionCount	—	—	—	—	—	—	—	—	—
AvVersionCount	—	—	—	—	◎	—	—	◎	—
InstanceCount	—	◎	—	—	—	—	◎	◎	—
RootCntrCount	—	—	○	○	○	—	—	—	—
AvCntrCountPerCntr	—	—	◎	◎	◎	—	—	—	—
AvCntrTreeHeight	—	—	◎	◎	◎	—	—	—	—
AvRefCountPerCntr	—	—	◎	◎	◎	—	—	—	—
AvRenditionCount	—	—	—	—	—	—	◎	◎	—
MaxContentSize	—	—	—	—	—	—	—	—	—
AvContentSize	—	—	—	—	—	—	◎	◎	—
SuperClassType	—	—	—	—	—	—	○	○	—
PctOfRefDoc	—	—	—	—	—	—	○	○	—
AvRefDocCount	—	—	—	—	—	—	○	○	—
ComponentClassName	—	—	—	—	—	—	—	—	—
MaxComponetCount	—	—	—	—	—	—	—	—	—
AvComponentCount	—	—	—	—	—	—	—	—	—
ContentType	—	—	—	—	—	—	○	○	—
GUID	○	○	○	○	○	○	○	○	○

（凡例）
◎：必ず指定してください。
○：指定を省略できます。
－：指定できません。

注※ 1
バージョン付きクラスを定義する場合，2 行で記述します。

注※ 2
バージョン付きクラスを定義する場合，2 行目の ACLib クラス名は 1 行目と必ず一致させてください。

(5) DataType エントリのデータ型ごとに指定するエントリの一覧

プロパティを定義する場合の DataType エントリのデータ型ごとに指定するエントリの一覧を次の表に示します。

表 4-31　DataType エントリのデータ型ごとに指定するエントリの一覧

エントリ名	DataType エントリのデータ型				
	String	Integer32	Boolean	VArray	全文検索機能のプロパティ
TargetType	◎※ 1	◎※ 1	◎※ 1	◎※ 1	◎※ 2
GUID	○	○	○	○	－
PropertyName	◎	◎	◎	◎	◎
DefaultValue	○	○	○	－	－
AvLength	◎	－	－	－	－
MaxLength	◎	－	－	－	－
StructClassName	－	－	－	◎	－
MaxElementCount	－	－	－	◎	－
AvElementCount	－	－	－	◎	－

（凡例）
◎：必ず指定してください。
○：指定を省略できます。
－：指定できません。

注※ 1
バージョン付きクラスのとき必ず指定してください。バージョンなしクラスのときは指定できません。

注※ 2
バージョン付きクラスのとき，「VR」を必ず指定してください。バージョンなしクラスのときは省略できます。

4.15.4　文書クラス定義ファイルの注意事項

文書クラス定義ファイルを記述する場合の注意事項を示します。

- 各エントリ指定値の前後の空白は，トリミング（空白を取り除く）されます。
- 定義済みのプロパティを別のクラスにも定義するときは，次のエントリに定義済みのプロパティと同じ値を指定してください。
  - PropertyName エントリ
  - DataType エントリ
  - MaxLength エントリ（String 型のとき）

- StructClassName エントリ（VariableArray 型のとき）
- MaxElementCount エントリ（VariableArray 型のとき）

GUID エントリおよび DefaultValue エントリには，二つ目の定義以降，省略するか同じ値を指定してください。

● 一つのクラスに追加できるインデクス数は次のとおりです。

表 4-32　文書クラス定義ファイルでユーザが追加したクラスに定義できるインデクス数

クラスの種類	TargetType エントリの値	追加できるインデクス数（個）
独立データ	－	1 ～ 251
バージョンなしコンテナ	－	1 ～ 250
バージョンなし構成管理コンテナ	－	1 ～ 250
バージョン付き構成管理コンテナ	CH	1 ～ 250
	VR	1 ～ 250
バージョンなし文書	－	1 ～ 250
バージョン付き文書	CH	1 ～ 250
	VR	1 ～ 250

（凡例）
－：該当しない。

表 4-33　文書クラス定義ファイルで DocumentBroker が提供するクラスを指定する場合に定義できるインデクス数

指定するクラス名	追加できるインデクス数（個）
aclClass_DCR	1 ～ 251
aclClass_RCR	1 ～ 251
aclClass_VCR	1 ～ 250
aclClass_PACL	1 ～ 252
aclClass_RS	1 ～ 251

● 複数のプロパティをキーとするインデクスでのプロパティは，1 ～ 16（個）で指定してください。

● 文書空間定義コマンド（EDMCDefDocSpace）で出力する定義情報ファイル中のプロパティ定義の値を次に示します。
- dmaProp_IsSelectable プロパティ：1
- dmaProp_IsSearchable プロパティ：1
- dmaProp_IsOrderable プロパティ：1

そのほかのプロパティについては，デフォルト値となります。そのほかのプロパティのデフォルト値については，「4.7.4(2)　定義情報ファイルの記述例」を参照してください。デフォルト値を変更する場合は，文書空間定義コマンド（EDMCDefDocSpace）で出力する定義情報ファイルをカスタマイズして使用してください。

4.16　文書空間情報ファイル

この節では，文書空間情報ファイルについて説明します。

文書空間情報ファイルのサンプルファイルは，"/opt/HiEDMS/sample/EDMdocinfo.txt" に提供されています。

4.16.1　文書空間情報ファイルの概要

文書空間情報ファイルは，DocumentBroker の文書空間の定義および構築で使用する情報を定義するファイルです。作成したファイルは，文書空間定義コマンド（EDMCDefDocSpace）の -s オプションで指定します。

"/opt/HiEDMS/sample" に格納されている「EDMdocinfo.txt」を，システム管理者がテキストエディタなどで編集してください。

以降，文書空間情報ファイルで指定するエントリを示します。各エントリの詳細については，「4.16.2　文書空間情報ファイルの記述形式」を参照してください。

（1）[System] セクション

システム全体のパラメタを定義します。[System] セクションを構成する各エントリを次の表に示します。

表 4-34　[System] セクションのエントリー覧

種類	設定項目	エントリ名	指定内容	省略の可否
アクセス制御機能	アクセス制御機能の使用の有無	AccessControl	アクセス制御機能の使用の有無	×
	パブリック ACL	PublicACLCount	パブリック ACL の作成オブジェクト数	○
		AvBindPublicACLCount	パブリック ACL の平均バインド数	○
	ローカル ACL	AvLocalACECount	ローカル ACL の平均 ACE 数	○
	セキュリティ ACL	AvSecurityACECount	セキュリティ ACL の平均 ACE 数	○
文書のコンテント	ログ取得モード	LogModeType	文書のコンテントのログ取得モード	○
リファレンスファイル管理機能	リファレンスファイル管理機能の使用の有無	ReferenceFile	リファレンスファイル管理機能の使用の有無	○
FileLink 連携機能	FileLink 連携機能の使用の有無	FileLink	FileLink 連携機能の使用の有無	○
マルチファイル管理機能	ファイルの最大数	MultiFileMaxCount	マルチファイル管理機能を使用する場合の 1 文書で管理できるファイルの最大数	○
	平均ファイル数	AvMultiFileCount	マルチファイル管理機能を使用する場合の 1 文書の平均ファイル数	△※
文書間リレーション管理機能	インデクス定義	MultiRelationIndex	複数列インデクスの指定	○
ユーザ識別子の最大長の拡張	ユーザ識別子の最大長	UserIDMaxLength	ユーザ識別子の最大長	○
グループ識別子の最大長の拡張	グループ識別子の最大長	GroupIDMaxLength	グループ識別子の最大長	○

種類	設定項目	エントリ名	指定内容	省略の可否
プロパティのデフォルト値の設定	プロパティのデフォルト値の設定方法	SpPropDefaultValue	プロパティのデフォルト値の設定方法	○
文書空間で使用する文字コード種別	文書空間で使用する文字コード種別	DocSpaceCharacterSet	文書空間で使用する文字コード種別	○

（凡例）
○：指定を省略できます。
△：使用環境の条件によって，指定の省略の可否が変わります。
×：指定を省略できません。

注※
MultiFileMaxCount エントリを指定した場合，省略できません。

(2) [RdAreaName] セクション

RD エリア名を定義します。[RdAreaName] セクションを構成する各エントリを次の表に示します。

表 4-35 [RdAreaName] セクションのエントリー覧

種類	エントリ名	指定内容	省略の可否
メタ情報格納用の RD エリア	MetaTblName	メタ情報の表を格納する RD エリアの RD エリア名	○
	MetaIdxName	メタ情報の表に定義されるインデクスを格納する RD エリアの RD エリア名	○
システム用 RD エリア	SysTblName	システムが定義する表を格納する RD エリアの RD エリア名	○
	SysIdxName	システムが定義する表に定義されるインデクスを格納する RD エリアの RD エリア名	○
ユーザ用 RD エリア	UsrTblName	ユーザが定義する表を格納する RD エリアの RD エリア名	○
	UsrIdxName	ユーザが定義する表に定義されるインデクスを格納する RD エリアの RD エリア名	○
LOB 列格納用 RD エリア	DocTblName	文書用 LOB エリアを格納する RD エリアの RD エリア名	○
	MultiDocTblName	マルチファイル文書用 LOB エリアを格納する RD エリアの RD エリア名	○
全文検索用 RD エリア	SgmlTextName	全文検索用の文書内容を格納する RD エリアの RD エリア名	○
	NgramIdxName	全文検索用のインデクスを格納する RD エリアの RD エリア名	○

（凡例）
○：指定を省略できます。

4.16.2 文書空間情報ファイルの記述形式

ここでは，文書空間情報ファイルの記述形式について説明します。

(1) 記述規則

文書空間情報ファイルの記述規則を次に示します。

- 印刷可能な ASCII コードで記述します。
- 空白行は無視します。
- 「#」（シャープ）および「；」（セミコロン）で始まる行はコメントとして扱います。
- 各行は行の終端を除いて，2,046 バイト以内で記述します。
- 行の終わりは，<EOF> または行末文字です。
- 行末文字は，<CR> ＋ <LF> です。
- ファイルの終端は <EOF> です。

(2) 記述形式

文書空間情報ファイルは，次に示す二つのセクションと，各セクションに指定するエントリによって構成されています。

- [System] セクション
- [RdAreaName] セクション

セクションとエントリの記述規則を次に示します。

- セクション名は，[ ]（角括弧）で囲んで指定します。一つのセクションは，セクション名を指定してから，次のセクションを指定するまで，またはファイルの終端までの範囲です。
- エントリは，「エントリ名 = 値」の形式で指定します。
- セクション名およびエントリ名は 127 バイト以内，値は 1,023 バイト以内で指定します。
- 指定できないセクションを指定した場合，指定したセクションはすべて無視されます。
- セクションを重複して記述している場合，エラーになります。
- セクション内でエントリを重複して記述している場合，エラーになります。
- 指定できないエントリを指定した場合，エラーになります。
- エントリの指定を無視する場合にも，値をチェックします。

記述形式を次に示します。

[< セクション名 >]
< エントリ名 >=< 値 >

以降，文書空間情報ファイルを構成する各セクションと，セクションごとに指定するエントリについて説明します。

(3) [System] セクション

システム全体のパラメタを定義します。[System] セクションを構成する各エントリは次のとおりです。

AccessControl エントリ

アクセス制御機能の使用の有無を指定します。

- ON

  アクセス制御機能を使用します。この値を指定した場合は，セキュリティ運用者を設定してくださ

い。セキュリティ運用者は，セキュリティ定義ファイルおよびユーザ権限定義ファイルを作成してください。

- OFF
  アクセス制御機能を使用しません。

なお，このエントリは省略できません。

PublicACLCount **エントリ**

パブリック ACL を作成するオブジェクト数を指定します。0 ～ 2,147,483,647 の範囲で指定します。
AccessControl エントリに ON を指定した場合，有効になります。OFF を指定した場合，指定を無視します。
指定を省略した場合，「0」が仮定されます。

AvBindPublicACLCount **エントリ**

パブリック ACL の平均バインド数を指定します。0 ～ 10 の範囲で指定します。
AccessControl エントリに ON を指定した場合，有効になります。OFF を指定した場合，指定を無視します。
指定を省略した場合，「0」が仮定されます。

AvLocalACECount **エントリ**

ローカル ACL の平均 ACE 数を指定します。0 ～ 64 の範囲で指定します。
AccessControl エントリに ON を指定した場合，有効になります。OFF を指定した場合，指定を無視します。
指定を省略した場合，「0」が仮定されます。

AvSecurityACECount **エントリ**

セキュリティ ACL の平均 ACE 数を指定します。0 ～ 64 の範囲で指定します。
AccessControl エントリに ON を指定した場合，有効になります。OFF を指定した場合，指定を無視します。
指定を省略した場合，「0」が仮定されます。

LogModeType **エントリ**

文書のコンテントのログ取得モードを指定します。文書クラス定義ファイルのクラス定義の ContentType エントリに Single または MultiFile を指定する場合だけ有効になります。

- ALL
  ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。ログ取得モードで運用すると，ロールバックおよびロールフォワードに必要なデータベースの更新ログを取得します。
- PARTIAL
  更新前ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。更新前ログ取得モードで運用すると，ロールバックに必要なデータベースの更新ログを取得します。
- NONE
  ログレスモードでユーザ LOB 用 RD エリアを運用するときに指定します。ログレスモードで運用すると，データベースの更新ログを取得しません。

指定を省略した場合，「ALL」が仮定されます。

ReferenceFile **エントリ**

リファレンスファイル管理機能の使用の有無を指定します。文書クラス定義ファイルのクラス定義の ContentType エントリに Reference を指定する場合，このエントリには ON を設定します。

- ON
  リファレンスファイル管理機能を使用します。

- OFF
  リファレンスファイル管理機能を使用しません。

指定を省略した場合，「OFF」が仮定されます。

FileLink エントリ

FileLink 連携機能の使用の有無を指定します。文書クラス定義ファイルのクラス定義の ContentType エントリに FileLink を指定する場合，このエントリには ON を設定します。

- ON
  FileLink 連携機能を使用します。
- OFF
  FileLink 連携機能を使用しません。

指定を省略した場合，「OFF」が仮定されます。

MultiFileMaxCount エントリ

マルチファイル管理機能を使用する場合に，一つの文書で管理できるファイルの最大数を指定します。2 ～ 4,096 の範囲で指定します。
文書クラス定義ファイルのクラス定義の ContentType エントリに Multi を指定する場合，このエントリに値を設定します。
指定を省略した場合，マルチファイル管理機能は使用できません。

AvMultiFileCount エントリ

マルチファイル管理機能を使用する場合に，一つの文書の平均ファイル数を指定します。文書クラス定義情報ファイルのクラス定義でバージョンなし文書またはバージョン付き文書のクラスを定義して，文書の種別にマルチファイル文書を使用するクラスを指定して，文書空間定義コマンド（EDMCDefDocSpace）を実行した場合の見積もり基礎情報ファイルの行の数は，このエントリの指定によって決まります。
このエントリは，MultiFileMaxCount エントリを指定していない場合，無効となります。また，このエントリは，MultiFileMaxCount エントリを指定した場合，必須となります。MultiFileMaxCount エントリの指定値以内で指定してください。
なお，このエントリは省略できません。

MultiRelationIndex エントリ

複数列インデクスを指定して，複数の同じ文書オブジェクトを，同時に一つの文書オブジェクトに対して関連づけるかどうかを指定します。

- ON
  複数の異なる文書オブジェクトを，同時に一つの文書オブジェクトに対して関連づける場合にも指定できます。次に示すクラスのプロパティに対して複数列インデクスを定義します。
  edmClass_Relationship クラス
  dmaProp_Head プロパティ
- OFF
  複数列インデクスを指定しません。この場合，単一列インデクスが定義されます。

指定を省略した場合，「OFF」が仮定されます。

UserIDMaxLength エントリ

ユーザ識別子の最大長を 255 バイト以上に拡張する場合に指定します。ユーザ識別子の最大長をバイト単位で指定します。255 ～ 512（バイト）の範囲で指定します。
指定を省略した場合，ユーザ識別子の最大長は 254 バイトになります。

GroupIDMaxLength **エントリ**

グループ識別子の最大長を 255 バイト以上に拡張する場合に指定します。グループ識別子の最大長をバイト単位で指定します。255 ～ 512（バイト）の範囲で指定します。
指定を省略した場合，グループ識別子の最大長は 254 バイトになります。

SpPropDefaultValue **エントリ**

プロパティのデフォルト値をクラスごとに設定する場合に指定します。
複数のクラスに対して同じ名前のプロパティを追加する場合に，クラスごとにプロパティのデフォルト値を設定できます。

- ON
  クラスごとにプロパティのデフォルト値を設定します。
- OFF
  プロパティで同一のデフォルト値を設定します。

プロパティのデフォルト値は，文書クラス定義ファイルの DefaultValue エントリに指定します。文書クラス定義ファイルについては，「4.16　文書クラス定義ファイル」を参照してください。
指定を省略した場合，「OFF」が仮定されます。

DocSpaceCharacterSet **エントリ**

文書空間で使用する文字コード種別を指定します。

- SJIS
  文書空間で使用する文字コード種別を Shift-JIS とします。
- UTF-8
  文書空間で使用する文字コード種別を UTF-8 とします。

指定を省略した場合，「SJIS」が仮定されます。

（4）[RdAreaName] セクション

RD エリア名を定義します。[RdAreaName] セクションを構成する各エントリは次のとおりです。

MetaTblName **エントリ**

メタ情報の表を格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。
RD エリアに割り当てる表の数が 500（HiRDB での最大値）を超えると，指定値の後ろに 7 けたの追番が付いた別名称（例：METATBL0000001）が順次割り当てられます※。
指定を省略した場合，「METATBL」が仮定されます。

MetaIdxName **エントリ**

メタ情報の表に定義されるインデクスを格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。
RD エリアに割り当てるインデクス数が 500（HiRDB での最大値）を超えると，指定値の後ろに 7 けたの追番が付いた別名称（例：METAIDX0000001）が順次割り当てられます※。
指定を省略した場合，「METAIDX」が仮定されます。

SysTblName **エントリ**

システムが定義する表を格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。
RD エリアに割り当てる表の数が 500（HiRDB での最大値）を超えると，指定値の後ろに 7 けたの追番が付いた別名称（例：SYSTBL0000001）が順次割り当てられます※。

指定を省略した場合，「SYSTBL」が仮定されます。

SysIdxName エントリ

システムが定義する表に定義されるインデクスを格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。

RD エリアに割り当てるインデクス数が 500（HiRDB での最大値）を超えると，指定値の後ろに 7 けたの追番が付いた別名称（例：SYSIDX0000001）が順次割り当てられます※。

指定を省略した場合，「SYSIDX」が仮定されます。

UsrTblName エントリ

ユーザが定義する表を格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。

RD エリアに割り当てる表の数が 500（HiRDB での最大値）を超えると，指定値の後ろに 7 けたの追番が付いた別名称（例：USRTBL0000001）が順次割り当てられます※。

指定を省略した場合，「USRTBL」が仮定されます。

UsrIdxName エントリ

ユーザが定義する表に定義されるインデクスを格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。

RD エリアに割り当てるインデクス数が 500（HiRDB での最大値）を超えると，指定値の後ろに 7 けたの追番が付いた別名称（例：USRIDX0000001）が順次割り当てられます※。

指定を省略した場合，「USRIDX」が仮定されます。

DocTblName エントリ

文書用 LOB エリアを格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。

指定を省略した場合，「DOC」が仮定されます。

MultiDocTblName エントリ

マルチファイル文書用 LOB エリアを格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。

指定を省略した場合，「MULTIDOC」が仮定されます。

SgmlTextName エントリ

全文検索用の文書内容を格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。

全文検索用の表が複数指定された場合は，指定値の後ろに 7 けたの追番が付いた別名称（例：SGMLTXT0000001）が順次割り当てられます※。

指定を省略した場合，「SGMLTXT」が仮定されます。

NgramIdxName エントリ

全文検索用のインデクスを格納する RD エリアの RD エリア名を指定します。1 ～ 23（バイト）の文字列を指定します。

全文検索用の表が複数指定された場合は，指定値の後ろに 7 けたの追番が付いた別名称（例：NGRAMIDX0000001）が順次割り当てられます※。

指定を省略した場合，「NGRAMIDX」が仮定されます。

注

- RD エリア名には，すでに使用されている RD エリア名を指定しないでください。すでに使用されてい

る RD エリア名を指定した場合のエラーチェックは行いません。そのため，RD エリアに割り当てる表およびインデクス数のチェックに RD エリアで使用されている表およびインデクス数は，RD エリアで使用されていないものとして，指定された表およびインデクスによって 1 からカウントされます。

- 次の RD エリア名には，ほかのエントリと重複しない RD エリア名を指定してください。
  - LOB エリア格納用（DocTblName エントリ，MultiDocTblName エントリ）
  - 全文検索用文書の内容格納用（SgmlTextName エントリ）
  - 全文検索インデクス格納用（NgramIdxName エントリ）
- 自動的に付与される名称（7 けたの追番が付与された別名称）と一致する名称は，各エントリに指定できません。
- 同じ表のインデクスが，複数の RD エリアに割り当てられないように調整されるため，インデクス数が一つの RD エリアに対して 500 未満となる場合があります。
- 指定できる RD エリア名称は，DocumentBroker のエリア名の規則に従ってください。空白またはハイフンを含む RD エリア名は指定できません。空白を含む RD エリア名を指定した場合，EDMCBuildDocSpace コマンド実行時に KMBR16920-E のメッセージ（エラー情報：rc=5）または KMBR10226-W のメッセージが出力されてエラーとなるか，またはデータベース定義ユティリティ実行時にデータベースのエラーとなります。ハイフンを含む RD エリア名を指定した場合，文書空間構築コマンド（EDMCBuildDocSpace）実行時，またはデータベース定義ユティリティ実行時にデータベースのエラーとなります。

注※

割り当てられた名称は，見積もり基礎情報ファイルで確認できます。割り当てられた RD エリア名称を変更したい場合には，見積もり基礎情報ファイルの内容を基に，文書空間定義コマンド（EDMCDefDocSpace）で出力される RD エリア定義情報ファイルを修正してください。

（5）[RdAreaName] セクションで指定する RD エリアのエントリ指定と，割り当てられる領域の関係

文書空間定義コマンド（EDMCDefDocSpace）のデフォルトの指定では，メタ情報格納用の RD エリア，システム用 RD エリア，ユーザ用 RD エリアの分類ごとに RD エリア名を割り当て，RD エリアの分類ごとにインデクス用の RD エリア名を割り当てます。RD エリアの分類と割り当てられる RD エリア名の関係を次の表に示します。

表 4-36　RD エリアの分類と割り当てられる RD エリア名の関係

RD エリアの分類	RD エリアに格納する要素	割り当てられる RD エリア名
メタ情報格納用の RD エリア	メタ情報を格納する表	MetaTblName エントリの指定値
	メタ情報を格納する表に設定するインデクス	MetaIdxName エントリの指定値
システム用 RD エリア	システムが定義する表	SysTblName エントリの指定値
	システムが定義する表に設定するインデクス	SysIdxName エントリの指定値
ユーザ用 RD エリア	ユーザが定義するサブクラスに対応する表	UsrTblName エントリの指定値
	ユーザが定義するサブクラスに対応する表に設定されるインデクス	UsrIdxName エントリの指定値
	HiRDB 繰り返し列のインデクス	_※

注※　インデクスは指定できないため，RD エリア名は割り当てられません。

4.16.3 文書空間情報ファイルの記述例

文書空間情報ファイルの記述例を次に示します。

```
[System]
AccessControl = OFF
PublicACLCount = 0
AvBindPublicACLCount = 0
AvLocalACECount = 0
AvSecurityACECount = 0
LogModeType = ALL
ReferenceFile = OFF
FileLink = OFF
;MultiFileMaxCount = 2
;AvMultiFileCount = 2
MultiRelationIndex = OFF
;UserIDMaxLength = 512
;GroupIDMaxLength = 512
;SpPropDefaultValue = OFF
DocSpaceCharacterSet = SJIS

[RdAreaName]
MetaTblName = METATBL
MetaIdxName = METAIDX
SysTblName = SYSTBL
SysIdxName = SYSIDX
UsrTblName = USRTBL
UsrIdxName = USRIDX
DocTblName = DOC
MultiDocTblName = MULTIDOC
SgmlTextName = SGMLTXT
NgramIdxName = NGRAMIDX
```

4.17 見積もり基礎情報ファイル

この節では，見積もり基礎情報ファイルについて説明します。

4.17.1 見積もり基礎情報ファイルの概要

見積もり基礎情報ファイルは，文書空間を構築するためのデータベース容量を見積もるのに使用する行の数などの基礎情報を，文書空間定義コマンド（EDMCDefDocSpace）で出力するファイルです。見積もり基礎情報ファイル（EDMestimate.csv）は，" 実行環境ディレクトリ /env" に出力されます。

なお，このファイルに出力される情報は，文書クラス定義ファイルおよび文書空間情報ファイルの値を基に決定されます。この値からデータベースの容量を見積もる場合，容量に対しての安全率を考慮してください。

4.17.2 見積もり基礎情報ファイルの出力形式

ここでは，見積もり基礎情報ファイルの出力形式，出力情報，およびエントリ情報について説明します。

(1) 出力形式

見積もり基礎情報ファイルの出力形式を次に示します。出力情報は種別ごとに分類され，コンマ区切りで出力されます。

種別,表識別子,インデクス名,レコード数,レコード長,列数,キー種別,平均文書コンテントサイズ,RDエリア名,列名,データ型，列長,平均文字列長

(2) 出力情報

見積もり基礎情報ファイルの出力情報を次の表に示します。

表 4-37 見積もり基礎情報ファイルの出力情報

出力情報	エントリ名	内容
種別	Kind	次のどれかが出力されます。 • MetaTable：メタ情報を格納する表 • SystemTable：システムで使用する表 • UserTable：ユーザが定義した表 • MetaIndex：メタ情報を格納する表に定義されるインデクス • SystemIndex：システムが定義する表に設定されるインデクス • UserIndex：ユーザが定義するサブクラスに対応する表に設定されるインデクス • BLOb：BLOB 列が定義された表 • SgmlText：全文検索用のテキスト • NgramIndex：全文検索用のインデクス • Disk：リファレンスファイル文書または FileLink 文書のコンテント格納用のディスク容量
表識別子	Table Name	データベースに定義される表の名前
インデクス名	Index Name	データベースに定義されるインデクスの名前
レコード数	Record Count	表に格納するレコードの総数
レコード長	Record Length	レコードのデータサイズ
列数	Column Count	表に定義する列の総数

出力情報	エントリ名	内容
キー種別	Key Kind	次のどちらかの値が出力されます。 • 1：UNIQUE キー • 0：重複キー
平均文書コンテントサイズ	AvContent Size	登録する文書のコンテントの平均サイズ
RD エリア名	Area Name	表やインデクスなどが格納される RD エリア名
列名	Column Name	可変長文字列型で平均文字列長が 256 バイト以上の列の名前，またはインデクスを定義する列の名前
データ型	Data Type	列のデータ型 次のどれかが出力されます。 • INT：整数 • CHAR：固定長文字列 • VARCHAR：可変長文字列 • MVARCHAR：可変長混在文字列
列長	Column Length	列の定義長※
平均文字列長	AvColumn Length	可変長文字列型の列の平均文字列長（可変長データの分岐ページ数，またはインデクスのキー長の計算に使用）

注※

列の定義長が出力されます。インデクスのキー長は，ユーザが算出してください。

(3) エントリ情報

種別ごとに出力されるエントリ情報を次の表に示します。

表 4-38 見積もり基礎情報ファイルのエントリ情報

種別（Kind）	出力情報											
	表識別子（Table Name）	インデクス名（Index Name）	レコード数（Record Count）	レコード長（Record Length）	列数（Column Count）	キー種別（Key Kind）	平均文書コンテントサイズ（AvContent Size）	RD エリア名（Area Name）	列名（Column Name）	データ型（Data Type）	列長（Column Length）	平均文字列長（AvColumn Length）
MetaTable	○	—	○	○	○	—	—	○	—	—	—	—
SystemTable	○	—	○	○	○	—	—	○	—	—	—	—
SystemTable ※1	○	—	—	—	—	—	—	○	○	○	○	○
UserTable	○	—	○	○	○	—	—	○	—	—	—	—
UserTable ※1	○	—	—	—	—	—	—	○	○	○	○	○
MetaIndex ※2	○	—	—	—	—	○	—	○	○	○	○	—
SystemIndex ※2	○	○	—	—	—	○	—	○	○	○	○	○※3
UserIndex ※2	○	○	—	—	—	○	—	○	○	○	○	○※3
BLOb	○	—	—	—	—	—	○	○	—	—	—	—

種別（Kind）	出力情報											
	表識別子（Table Name）	インデクス名（Index Name）	レコード数（Record Count）	レコード長（Record Length）	列数（Column Count）	キー種別（Key Kind）	平均文書コンテントサイズ（AvContent Size）	RD エリア名（Area Name）	列名（Column Name）	データ型（Data Type）	列長（Column Length）	平均文字列長（AvColumn Length）
SgmlText ※4	○	－	－	－	－	－	○	○	－	－	－	－
NgramIndex ※4	○	－	－	－	－	－	○	○	－	－	－	－
Disk ※5	○	－	－	－	－	－	○	－	－	－	－	－

（凡例）

○：出力されます。

－：出力されません。

注※ 1

平均文字列長が 256 バイト以上の列がある場合です。

注※ 2

複数列インデクスの場合，列情報が複数出力されます。

注※ 3

ユーザが定義した可変長文字列型の列の場合だけ出力されます。

注※ 4

全文検索用の文書を指定した場合だけ出力されます。

注※ 5

リファレンスファイル文書または FileLink 文書の場合だけ出力されます。

4.17.3 見積もり基礎情報ファイルの出力例

見積もり基礎情報ファイルの出力例を次に示します。

```
; Estimate Basic Information File
; 2004/09/24 13:09:29

Kind,Table Name,Index Name,Record Count,Record Length,Column Count,Key
Kind,AvContent Size,Area Name,Column Name,Data Type,Column Length,AvColumn
Length
MetaTable,EDMS_METAINI,,2190,158,3,,,METATBL,,,,
MetaTable,EDMS_META_edms,,891,527,5,,,METATBL,,,,
MetaTable,EDMS_META_edmsys,,71,527,5,,,METATBL,,,,
       :
SystemTable,edmClass_OIID,,1,20,3,,,SYSTBL,,,,
SystemTable,dmaClass_ConfigHistory,,0,228,6,,,SYSTBL,,,,
SystemTable,dmaClass_VerDescription,,9755,180,6,,,SYSTBL,,,,
SystemTable,dmaClass_Rendition,,3900,380,6,,,SYSTBL,,,,
SystemTable,dmaClass_ContentReference,,0,596,5,,,SYSTBL,,,,
       :
UserTable,usrClass_IndP1,,100,465,8,,,USRTBL,,,,
UserTable,usrClass_IndP1,,,,,,,USRTBL,uP_String_512,MVARCHAR,512,300
UserTable,usrClass_RCntr,,1111,313,9,,,USRTBL,,,,
UserTable,usrClass_VTCntr,,1111,317,10,,,USRTBL,,,,
UserTable,usrClass_CRCntr_CH,,1111,365,10,,,USRTBL,,,,
UserTable,usrClass_CRCntr_VR,,5555,184,7,,,USRTBL,,,,
       :
MetaIndex,EDMS_META_edms,,,,,1,,METAIDX,SECTION_NAME,VARCHAR,128,
```

```
MetaIndex,EDMS_META_edms,,,,,1,,METAIDX,OCCUR,INT,4,
MetaIndex,EDMS_META_edmsys,,,,,1,,METAIDX,SECTION_NAME,VARCHAR,128,
MetaIndex,EDMS_META_edmsys,,,,,1,,METAIDX,OCCUR,INT,4,
       :
SystemIndex,edmClass_OIID,edmClass_OIIDAp,,,,1,,SYSIDX,edmProp_AppId,CHAR,2,
SystemIndex,dmaClass_ConfigurationHistory,dmaClass_ConfigHistoryId,,,,1,,SYSIDX
,dmaProp_OIID,CHAR,16,
SystemIndex,dmaClass_ConfigurationHistory,dmaClass_ConfigHistoryPc,,,,0,,SYSIDX
,dmaProp_ParentContainer,CHAR,52,
SystemIndex,dmaClass_ConfigurationHistory,dmaClass_ConfigHistoryTh,,,,0,,SYSIDX
,dmaProp_This,CHAR,52,
       :
UserIndex,usrClass_Doc,usrClass_Doc01,,,,1,,USRIDX,uP_String,MVARCHAR,128,
UserIndex,usrClass_Doc,usrClass_Doc01,,,,1,,USRIDX,uP_Integer32,INT,4,
       :
BLOb,dmaClass_ContentTransfer,,,,,,9308,DOC,,,,
```

4.18 クライアントアプリケーション動作定義ファイル（application.ini）

この節では，クライアントアプリケーション動作定義ファイルに指定できるセクションおよびエントリと，その記述形式について説明します。

クライアントアプリケーション動作定義ファイルのサンプルファイルは，次の場所に格納されています。

"/opt/HiEDMS/client/sample/application.ini"

4.18.1 クライアントアプリケーション動作定義ファイルの概要

クライアントアプリケーション動作定義ファイルは，クライアントアプリケーションの動作環境を定義するためのファイルです。作成したファイルは，次のディレクトリに格納し，クライアントアプリケーションの実行者に読み取り権限を付与します。

環境変数「EDMAPPDEFPATH」に指定したディレクトリ　または "/opt/HiEDMS/client/etc"

環境変数「EDMAPPDEFPATH」が指定されていない場合，"/opt/HiEDMS/client/etc" ディレクトリ以下を参照します。

なお，クライアントアプリケーション動作定義ファイルが，上記ディレクトリ以下に存在しない場合，クライアントアプリケーションの動作環境は定義されていないものとして動作します。

4.18.2 クライアントアプリケーション動作定義ファイルの記述形式

ここでは，クライアントアプリケーション動作定義ファイルの記述形式について説明します。

(1) 記述形式

クライアントアプリケーション動作定義ファイルは，次に示す二つのセクションと，各セクションに指定するエントリによって構成されています。

- [Public] セクション
- [クライアントアプリケーションプロセス名] セクション

セクションとエントリの記述規則を次に示します。

- 印刷可能な ASCII コードで記述します。
- セクション名は，[　] （角括弧）で囲んで指定します。一つのセクションは，セクション名を指定してから，次のセクションを指定するまで，またはファイルの終端までの範囲です。
- エントリは，「エントリ名 = 値」の形式で指定します。
- 「;」（セミコロン）で始まる行はコメント行として扱われます。
- 同じセクション名を指定した場合，2 番目以降に指定したセクションは無視されます。
- 指定できないセクションを指定した場合，指定したセクションはすべて無視されます。
- セクション内で同一名のエントリを複数指定した場合，最初に指定したエントリが有効になります。
- 指定できないエントリを指定した場合，指定したエントリは無視されます。
- 値は 1,023 バイト以内で指定します。

● 値が指定されていないエントリ，または空白が指定されているエントリは無視されます。

記述形式を次に示します。

[< セクション名 >]
< エントリ名 >=< 値 >

以降，クライアントアプリケーション動作定義ファイルを構成する各セクションと，セクションごとに指定するエントリについて説明します。

(2) [Public] セクション

DocumentBroker Development Kit または DocumentBroker Runtime をインストールした環境下で動作するすべてのクライアントアプリケーションの動作環境を指定します。

[Public] セクションを構成するエントリは次のとおりです。

VBProperty エントリ

クライアントアプリケーションプロセスの VisiBroker プロパティを指定します。指定できるプロパティについては，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R) プログラマーズリファレンス」を参照してください。
TPBroker V3 と連携している場合にこのエントリを指定しても無視されます。TPBroker V3 と連携している場合は，OrbOption エントリを指定してください。エントリの指定を省略したときに仮定される値はありません。
なお，エントリ値に，TPBroker が不正とみなす VisiBroker プロパティを指定した場合は，クラスライブラリのメソッドで次のエラーが返却されます。
major_code=ERR_DMA
minor_code=DMARC_NETWORK_UNAVAILABLE

OrbOption エントリ

クライアントアプリケーションプロセスの ORB オプションを指定します。指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。
TPBroker V5 と連携している場合にこのエントリを指定しても無視されます。TPBroker V5 と連携している場合は，VBProperty エントリを指定してください。エントリの指定を省略したときに仮定される値はありません。
なお，エントリ値に，TPBroker が不正とみなす ORB オプションを指定した場合は，クラスライブラリのメソッドで次のエラーが返却されます。
major_code=ERR_DMA
minor_code=DMARC_NETWORK_UNAVAILABLE

(3) [クライアントアプリケーションプロセス名] セクション

DocumentBroker Development Kit または DocumentBroker Runtime をインストールした環境下で動作するクライアントアプリケーションの動作環境を，クライアントアプリケーション名ごとに指定します。なお，このセクションを指定したプロセス名のクライアントアプリケーションプロセスでは，[Public] セクションに指定したエントリは無効になります。

[クライアントアプリケーションプロセス名] セクションを構成するエントリは次のとおりです。

VBProperty エントリ

クライアントアプリケーションプロセスの VisiBroker プロパティを指定します。指定できるプロパティについては，マニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)

プログラマーズリファレンス」を参照してください。
TPBroker V3 と連携している場合にこのエントリを指定しても無視されます。TPBroker V3 と連携している場合は，OrbOption エントリを指定してください。エントリの指定を省略したときに仮定される値はありません。
なお，エントリ値に，TPBroker が不正とみなす VisiBroker プロパティを指定した場合は，クラスライブラリのメソッドで次のエラーが返却されます。
major_code=ERR_DMA
minor_code=DMARC_NETWORK_UNAVAILABLE

OrbOption エントリ

クライアントアプリケーションプロセスの ORB オプションを指定します。指定できるオプションについては，マニュアル「VisiBroker for C++ プログラマーズガイド」を参照してください。
TPBroker V5 と連携している場合にこのエントリを指定しても無視されます。TPBroker V5 と連携している場合は，VBProperty エントリを指定してください。エントリの指定を省略したときに仮定される値はありません。
なお，エントリ値に，TPBroker が不正とみなす ORB オプションを指定した場合は，クラスライブラリのメソッドで次のエラーが返却されます。
major_code=ERR_DMA
minor_code=DMARC_NETWORK_UNAVAILABLE

このセクションに指定するクライアントアプリケーションプロセス名の記述規則を次に示します。

- プロセス名から拡張子を除いた名称を指定してください。部分的に一致するプロセス名を指定した場合は無効になります。
  例えば，プロセス名が ProcessA.exe の場合は，ProcessA と指定します。
- 大文字と小文字で区別されます。
- プロセス名に Public は指定できません。
- プロセス名に]（右角括弧）を含む名称は指定できません。
- プロセス名の前後に空白を含む名称は指定できません。
  例えば，プロセス名として「△ ProcessA」や，「ProcessA △」（△：空白）を指定できません。

4.18.3 クライアントアプリケーション動作定義ファイルの記述例

クライアントアプリケーション動作定義ファイルの記述例を次に示します。

（1） TPBroker V5 と連携する場合

（例）

ネットワーク上の ORB ドメインを定義するポート番号を指定します。

```
[Public]
VBProperty = -Dvbroker.agent.port=14000
[ProcessA]
VBProperty = -Dvbroker.agent.port=14005
[ProcessB]
VBProperty = -Dvbroker.agent.port=14006
```

ProcessA および ProcessB 以外のクライアントアプリケーションプロセスには，[Public] セクションの指定値が適用されます。

(2) TPBroker V3 と連携する場合

(例)

ネットワーク上の ORB ドメインを定義するポート番号を指定します。

```
[Public]
OrbOption = -ORBagentPort 14000
[ProcessA]
OrbOption = -ORBagentPort 14005
[ProcessB]
OrbOption = -ORBagentPort 14006
```

ProcessA および ProcessB 以外のクライアントアプリケーションプロセスには，[Public] セクションの指定値が適用されます。

5 DocumentBroker の起動と終了

この章では，DocumentBroker の起動と終了の方法について説明します。

5.1 DocumentBroker の起動方法
5.2 DocumentBroker の終了方法

5.1 DocumentBroker の起動方法

この節では，DocumentBroker の起動方法について説明します。

起動手順

次の手順に従って，DocumentBroker を起動してください。なお，手順 1.，手順 2. および手順 3. の順序は問いません。

1. **データベースシステムを起動する。**
2. **TPBroker のスマートエージェントを起動する。**
3. **LDAP 対応のディレクトリサービスを起動する。**
   ユーザ管理機能に LDAP 対応のディレクトリサービスを使用する場合，実行してください。ユーザ管理機能については，「3.7　ユーザ管理機能の設定」を参照してください。
4. **DocumentBroker を起動する。**
   DocumentBroker 起動コマンド（EDMStart）を実行します。DocumentBroker 起動コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。

5.2 DocumentBroker の終了方法

この節では，DocumentBroker の終了方法について説明します。

終了手順

次の手順に従って，DocumentBroker を終了してください。なお，手順 2.，手順 3. および手順 4. の順序は問いません。

1. DocumentBroker **を終了する。**
   DocumentBroker 終了コマンド（EDMStop）を実行します。DocumentBroker 終了コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。
2. LDAP **対応のディレクトリサービスを終了する。**
   ユーザ管理機能に，LDAP 対応のディレクトリサービスを使用する場合，実行してください。
   ユーザ管理機能については，「3.7　ユーザ管理機能の設定」を参照してください。
3. TPBroker **のスマートエージェントを終了する。**
4. **データベースシステムを終了する。**

6 運用と障害対策

この章では，DocumentBroker で使用するデータベースの運用方法，およびリファレンスファイル管理機能を使用する場合の運用方法について説明します。また，障害対策，および障害情報の取得についても説明します。

6.1 **データベースの運用**

6.2 DocumentBroker **のクラス定義の変更**

6.3 **リファレンスファイル管理機能を使用する場合の運用**

6.4 **障害対策**

6.5 **アクセスログに関する運用**

6.6 **エラーログに関する運用**

6.1 データベースの運用

この節では，データベースの運用方法について説明します。

6.1.1 バックアップと回復

DocumentBroker で使用する表のバックアップと回復は，データベースシステムの機能を利用してください。なお，使用できるデータベースシステムは HiRDB のため，データのバックアップと回復は HiRDB の機能を利用してください。HiRDB でのバックアップの取得方法と回復方法の詳細については，マニュアル「HiRDB　システム運用ガイド」を参照してください。

また，ハードウェア障害の発生に備えて，DocumentBroker がインストールされているシステムのレジストリ情報などの環境情報のバックアップも取得するようにしてください。

(1) バックアップと回復について

DocumentBroker は，データベース上のレコードを組み合わせることで，アプリケーションで扱う論理的なオブジェクトや，オブジェクト間の関連を実現しています。したがって，障害時の回復には，これらの関連の整合性を保持するようにバックアップおよび回復を行う必要があります。

(2) バックアップと回復の方法

DocumentBroker では，オブジェクト間の関連を，文書空間単位に管理しています。また，文書空間はデータベースのスキーマに対応しており，文書空間単位で運用できます。

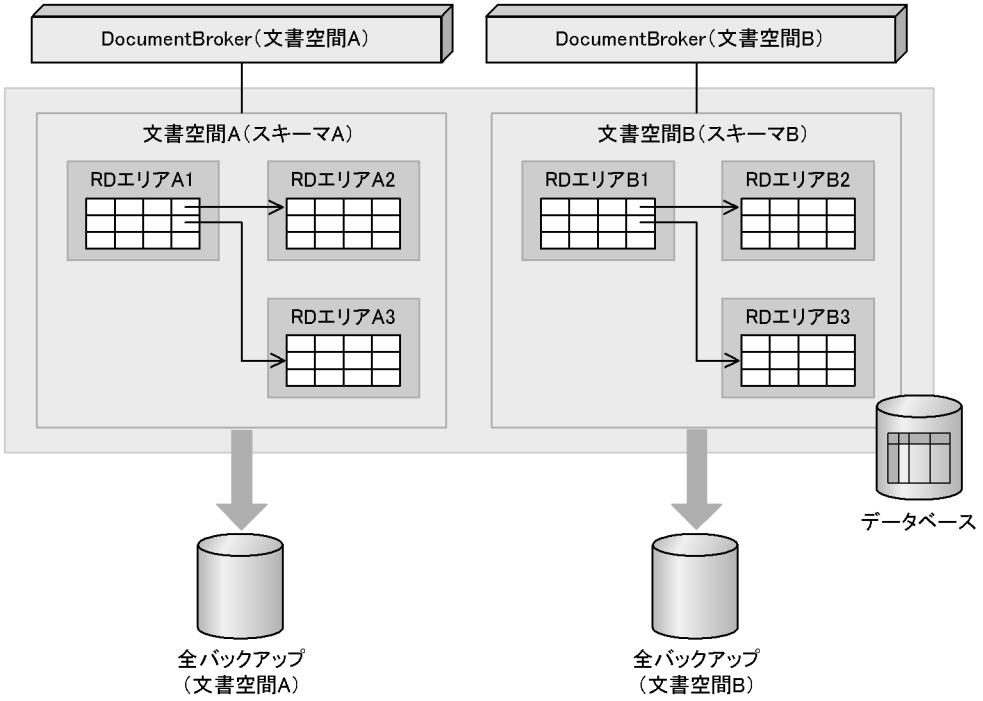
DocumentBroker では，アプリケーションで扱うすべてのデータをデータベース上に格納しています。格納しているデータの障害回復には，データベースの機能を利用してください。

オブジェクト間の関連の整合性を保持したバックアップと回復の方法としては，文書空間単位でバックアップと回復を行う方法と RD エリア単位のバックアップと回復を行う方法があります。次に，これらの方法を説明します。

(a) 文書空間単位のバックアップと回復

文書空間単位でのバックアップの取得の概念を次の図に示します。

図 6-1 文書空間単位でのバックアップの取得



文書空間（スキーマ）に属する全データのバックアップを，同期を取って取得します。障害が発生した場合は，全データをバックアップしたデータを基に回復します。ただし，同期を取って全データのバックアップを取得する運用では，文書空間に属するデータ量が多い場合，バックアップおよび回復に必要とする時間の増大や，障害回復までに，停止する業務範囲が大きいなどの問題が発生します。

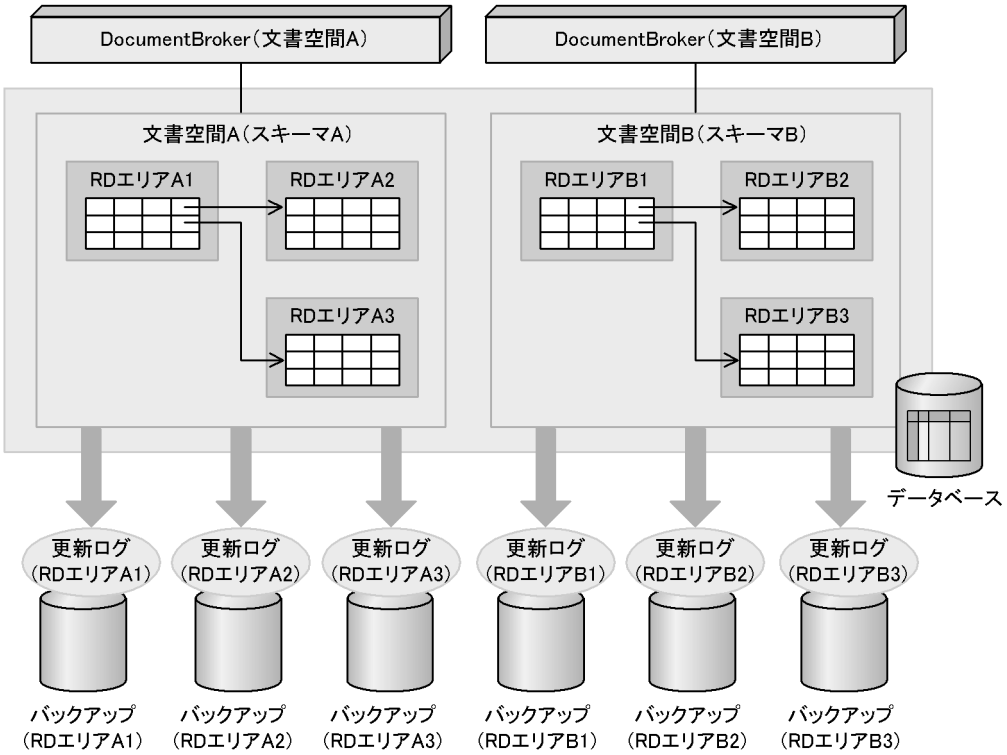
この運用の場合，更新ログの取得方法として次に示す方法のどちらかを選択してください。

- **ロールバックおよびロールフォワードに必要なログを取得する。**
  この方法の場合，バックアップから取得したデータに対して，更新ログでの任意の同期点までの論理的な整合性を保って回復します。
- **ロールバックに必要なログだけを取得する。**
  この方法の場合，最後に全データのバックアップを取得した時点まで回復します。

(b) RD エリア単位のバックアップと最新同期点までの回復

RD エリア単位でのバックアップの取得の概念図を次の図に示します。

図 6-2 RD エリア単位でのバックアップ取得



文書空間を構成する個々の RD エリアまたはディスクのパーティション単位にバックアップします。障害が発生した場合は，バックアップしたデータを基に，障害の発生した RD エリアのデータを回復します。この時，ほかの RD エリアのデータとの関連の整合性を保証するため，データベースの更新ログから最新の同期点まで回復します。このため，更新ログの取得方法は，ロールバックおよびロールフォワードに必要なログを取得する設定にしてください。

この方式では，バックアップした時に RD エリア相互の同期を取る必要がないため，対象データが多い場合にも，日々のバックアップ対象を全体の一部ずつとするなど，柔軟なバックアップの運用ができます。

(3) データベース運用上の注意事項

LOB 列格納用 RD エリアの更新ログの取得方法は，データベース定義文を作成するコマンド（EDMCrtSql および EDMAddMeta）の -u オプションで指定します。この -u オプションの引数によって，更新ログの取得方法を指定します。取得方法には，ログ取得モード，更新前ログ取得モード，およびログレスモードがあります。

なお，ログ取得モード（ロールバックおよびロールフォワードに必要なログを取得）を指定した場合は，更新前ログ取得モード（ロールバックに必要なログを取得）を指定した場合と比べて，次の注意が必要です。

(a) ログファイル容量

当該 LOB 列に対する追加・更新のデータ量の分だけログが増加するため，ログ容量（特にアンロードログファイルのディスク容量）が不足しないようにファイルサイズ（バイト）を見積もってください。1 文書当たりの見積もりの概算を次に示します。

（文書ファイル数 ＋ 4）× 平均ファイルサイズ（バイト）

システムファイルの容量見積もりの詳細については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

(b) ログ取得性能

ログの出力量の増加による性能劣化を抑えるため，次の 2 点を検討してください。

LOB 列格納用 RD エリアへのグローバルバッファの割り当て

グローバルバッファが指定されていない場合は，データの追加および更新ごとにログがファイルに出力されるため入出力が増加します。しかし，グローバルバッファが指定されている場合は，コミット時にログのファイルをバッファリングして出力するため，前者と比べて入出力の増加を抑えた運用ができます。グローバルバッファのオペランドの設定の詳細については，マニュアル「HiRDB　システム定義」を参照してください。

HiRDB のログ入出力バッファ長の拡大

ログ量の増加に伴ってログ用バッファの使用量が増え，ログバッファの空き待ちになる場合が想定されます。このため，ログ入出力バッファ長の拡大を検討してください。サーバ共通定義の詳細については，マニュアル「HiRDB　システム定義」を参照してください。

(4) バックアップの取得

データベースシステムの機能を利用して，同期を取って，DocumentBroker が使用している表一式のバックアップを取得してください。バックアップ対象になる表やエリアなどの情報は，データベースの初期設定で使用したデータベース定義文を基に事前に用意してください。

また，ハードウェア障害の発生に備えて，DocumentBroker がインストールされているシステムのレジストリ情報などの環境情報のバックアップも取得するようにしてください。

(5) リストアの方法

取得したバックアップのリストアは，データベースシステムの機能を利用して実行してください。リストアを実行する前には，必ず DocumentBroker を停止させてください。

6.1.2 ジャーナルファイルの運用

障害が発生した場合は，データベースシステムが取得するジャーナルファイルを利用してデータを回復してください。

なお，使用できるデータベースシステムは HiRDB です。したがって，HiRDB が取得するジャーナルファイルを利用して，データを回復してください。HiRDB のジャーナルファイルの運用方法については，マニュアル「HiRDB　システム運用ガイド」を参照してください。

6.1.3 レプリケーション

データベースの内容をほかのデータベースに反映する機能のことを，レプリケーション機能といいます。レプリケーション機能を利用することで，DocumentBroker が使用するデータベースのデータを抽出して，ほかのデータベースへ反映できます。

なお，使用できるデータベースシステムは HiRDB です。したがって，HiRDB のレプリケーション機能を利用します。HiRDB のレプリケーション機能を利用するには，HiRDB に HiRDB Datareplicator を組み込む必要があります。HiRDB のレプリケーション機能については，マニュアル「HiRDB Datareplicator」を参照してください。

(1) レプリケーションの方法

データベースシステムのレプリケーション機能を利用して，ほかのデータベースへデータを反映できます。レプリケーションの対象となる表やエリアなどの情報は，データベースの初期設定で使用したデータベース定義文を基に事前に用意してください。

(2) クラス，プロパティとデータベースの表，列との対応関係

データベースシステムの定義系 SQL を利用して，クラス，プロパティとデータベースの表，列との対応が取れます。使用できるデータベースシステムは，HiRDB です。したがって，ここでは HiRDB を利用する場合の対応関係について説明します。

DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql）では，メタ情報ファイルに記述されているクラス識別子を，オブジェクトを格納する RDB 表名として設定します。また，メタ情報ファイルに記述されているプロパティ識別子を表の各列名として設定します。

メタ情報ファイルでは，クラスの詳細情報を定義する「ClassDescription」やクラスのプロパティの詳細情報を定義する「PropertyDescription」の中に，クラスまたはプロパティ自身を説明する文字列を設定する項目「dmaProp_DescriptiveText」を指定できます。通常，「dmaProp_DescriptiveText」の値には，クラス識別子またはプロパティ識別子に対応して，ユーザが使用するクラス，プロパティの説明，または通称（意味のある名称）を指定します。これは，ユーザが定義情報ファイルを使用して，サブクラスおよびプロパティを追加する場合でも同様です。

EDMCrtSql コマンドを実行すると，HiRDB の定義系 SQL である COMMENT を使用して，「dmaProp_DescriptiveText」の値を各表および列の注釈としてデータベース定義文に出力します。ただし，データベース定義文に注釈を出力するには，EDMCrtSql コマンドを実行する時に，-c オプションを指定する必要があります。EDMCrtSql コマンドの使用方法については，「7.3　コマンドの文法」を参照してください。ユーザは HiRDB のディクショナリの SQL_TABLES 表および SQL_COLUMNS 表を検索して，表や列に付けられている注釈を参照できます。

したがって，レプリケーション機能を利用する場合は，表や列に付けられた注釈を参照して，DocumentBroker で作成したクラスおよびプロパティとの対応に注意してください。

6.2 DocumentBroker のクラス定義の変更

この節では，DocumentBroker の運用中に，サブクラスやプロパティの追加，削除など，データベースの表を定義・削除する場合の操作手順および注意事項について説明します。

この節で「クラスの定義」および「プロパティの定義」とは，メタ情報として格納している DocumentBroker のクラスおよびプロパティの定義を表します。「表」および「列」は，データベースの表および列を表します。

また，定義系 SQL とは HiRDB の定義系 SQL を表します。

6.2.1 クラスの定義を変更する場合について

DocumentBroker の運用中にクラスを追加したり，プロパティを追加したりする場合があります。DocumentBroker では，次のような場合にクラスの定義を変更します。

- クラスの追加
- クラスの削除（オブジェクトが存在しない場合）
- クラスの定義の変更（オブジェクトが存在しない場合）
  - プロパティの追加
  - プロパティの定義の変更
  - プロパティの削除
- クラスの削除（オブジェクトが存在する場合）
- クラスの定義の変更（オブジェクトが存在する場合）
  - プロパティの追加

次に，これらのクラスの定義を変更する手順を説明します。

6.2.2 クラスの追加

クラスを追加する場合，追加するクラスの定義を記述した定義情報ファイルを作成します。そして，作成した定義情報ファイルを指定してメタ情報の追加コマンド（EDMAddMeta）で，メタ情報へのクラスの定義の追加，および追加するクラスに対応する表を作成するデータベース定義文の生成を実行します。

(1) クラスの追加手順

クラスを追加する手順を次に示します。

1. **定義情報ファイルを作成します。**
   追加するクラスの定義を記述した定義情報ファイルを作成します。作成した定義情報ファイルは，メタ情報の追加コマンド（EDMAddMeta）の -f オプションに指定します。
   定義情報ファイルの作成については，「4.7　定義情報ファイル」を参照してください。

2. **インデクス情報ファイルを作成します。**
   追加するクラスのプロパティにインデクスを定義する場合，インデクス情報ファイルを作成します。メタ情報の追加コマンド（EDMAddMeta）の -i オプションに，作成したインデクス情報ファイル名を指定して実行することで，インデクス情報ファイルの内容を基に，データベース定義文に定義系 SQL「CREATE INDEX」が出力されます。なお，データベース定義文は，-o オプションで指定するデータベース定義文格納ファイルに出力されます。
   インデクス情報ファイルの作成については，「4.9　インデクス情報ファイル」を参照してください。

3. **RD エリア定義情報ファイルを作成します。**
   データベース定義文中の RD エリア名称を，ユーザが指定した RD エリア名で出力させるためには，RD エリア定義情報ファイルを作成します。
   メタ情報の追加コマンド（EDMAddMeta）の -r オプションに，作成した RD エリア定義情報ファイル名を指定することで，RD エリア定義情報ファイルの内容がデータベース定義文の RD エリア名に反映されます。なお，データベース定義文は，-o オプションで指定するデータベース定義文格納ファイルに出力されます。
   RD エリア定義情報ファイルの作成については，「4.8　RD エリア定義情報ファイル」を参照してください。

4. **メタ情報のバックアップを取得します。**
   メタ情報を更新する前に，メタ情報ファイルの出力コマンド（EDMPrintMeta）でメタ情報のバックアップを取得します。
   メタ情報ファイルの出力コマンド（EDMPrintMeta）には，-F オプションおよび -l オプションを指定します。-l オプションには，メタ情報のバックアップを出力するディレクトリ名を指定して，データベースからメタ情報を出力します。

5. **データベースのバックアップを取得します。**
   クラスに対応する表を作成するので，データベースのバックアップを取得します。バックアップを取得する必要のある RD エリアについては，マニュアル「HiRDB　システム運用ガイド」の，障害発生に備えた運用に関する説明での，定義系 SQL「CREATE TABLE」の実行についての記述を参照してください。

6. **メタ情報の追加コマンド（EDMAddMeta）を実行します。**
   手順 1. ～手順 3. で作成した定義情報ファイル，インデクス情報ファイル，および RD エリア定義情報ファイルを指定して，メタ情報の追加コマンド（EDMAddMeta）を実行して，クラスの定義をメタ情報へ追加します。
   メタ情報の追加コマンド（EDMAddMeta）の -o オプションで，追加するクラスに対応する表を作成する定義系 SQL「CREATE TABLE」を格納するデータベース定義文格納ファイルを指定します。
   次に，出力されたデータベース定義文格納ファイルの修正について説明します。

   出力されたデータベース定義文格納ファイルの修正

   メタ情報の追加コマンド（EDMAddMeta）は，データベース定義文格納ファイルへ定義系 SQL「CREATE TABLE」のすべてのオプションを出力しません。このため，必要なオプションは，データベース定義文格納ファイルをテキストエディタなどで修正して追加してください。メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL については，「(2)　メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL」を参照してください。

   次に，基本単位が VariableArray 型のプロパティを定義した場合，データベース定義文格納ファイルに出力されるデータベース定義文について説明します。

   基本単位が VariableArray 型のプロパティの定義

   メタ情報の追加コマンド（EDMAddMeta）は，追加するクラスのプロパティとして，次に示す条件を満たすプロパティを定義している場合，基本単位が VariableArray 型のプロパティの要素を格納するために定義する edmClass_Struct クラスのサブクラスに対応する表を作成する定義系 SQL「CREATE TABLE」を出力します。

   ・基本単位が VariableArray 型のプロパティが存在する。

   ・基本単位が VariableArray 型のプロパティの要素を，クラスに対応する表とは別の表に格納するように指定する。

   ・基本単位が VariableArray 型のプロパティは，新規に追加するプロパティの定義である。

   ただし，基本単位が VariableArray 型のプロパティの要素を格納する edmClass_Struct クラスの

サブクラスの追加は，クラスの追加（ここで説明している作業）の前に行ってください。
基本単位が VariableArray 型のプロパティを定義するクラスの定義を行う場合にメタ情報の追加コマンド（EDMAddMeta）が出力するデータベース定義文については，「6.2.7　メタ情報の追加・削除コマンドの動作」を参照してください。

7. **データベース定義ユティリティを実行します。**
手順 6. で作成したデータベース定義文を入力ファイルとして指定し，HiRDB のデータベース定義ユティリティ（pddef）を実行して定義するクラスに対応する表を追加します。データベース定義ユティリティ（pddef）は，DocumentSpace 構成定義ファイルの PdUser エントリに指定するユーザが実行してください。
HiRDB のデータベース定義ユティリティ（pddef）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。

8. **クラス定義情報ファイルを作成します。**
追加したクラスの定義を反映したクラス定義情報ファイルを作成するため，クラス定義情報ファイルの作成コマンド（EDMCrtSimMeta）を実行します。作成したクラス定義情報ファイルを DocumentBroker オブジェクト操作ツールの動作環境に配置します。
クラス定義情報ファイルの作成については，「4.10　クラス定義情報ファイル」を参照してください。

9. **表へオブジェクトを格納します。**
追加したクラスのオブジェクトを作成するには，次の方法があります。
   - **DocumentBroker Object Loader を使用したオブジェクトの登録**
   大量のオブジェクトを作成するには DocumentBroker Object Loader を使用します。
   オブジェクトを登録するためには，DocumentBroker Object Loader の実行に必要な入力データを作成する必要があります。入力データは，DocumentBroker Object Loader が提供する入力ファイル生成ユティリティ（EDMCrtLDF コマンド，EDMCrtLCF コマンド）で作成してください。入力ファイル生成ユティリティについては，マニュアル「DocumentBroker Object Loader Version 3」を参照してください。
   - **UAP によるオブジェクトの作成**
   UAP によるオブジェクトの作成方法については，マニュアル「DocumentBroker Version 3　クラスライブラリ　C++　解説」を参照してください。
   - **DocumentBroker オブジェクト操作ツールによるオブジェクトの作成**
   DocumentBroker オブジェクト操作ツールによるオブジェクトの作成方法については，マニュアル「DocumentBroker Version 3　オブジェクト操作ツール」を参照してください。

(2) メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL

メタ情報の追加コマンド（EDMAddMeta）が出力する，HiRDB のデータベース定義ユティリティ（pddef）の入力となる定義系 SQL を示します。下線が付いている個所は，メタ情報の追加コマンド（EDMAddMeta）が出力しない定義系 SQL を示します。

(a) メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL「CREATE TABLE」

```
CREATE 〔FIX〕 TABLE 〔認可識別子.〕表識別子(表要素〔,表要素〕…)
    {IN{表格納用RDエリア名
          |(表格納用RDエリア名)
          |(〔(表格納用RDエリア名)格納条件,〕…
          (表格納用RDエリア名)〔格納条件〕)}
       |PARTITIONED BY 列名
          IN(〔(表格納用RDエリア名)境界値,〕…
                (表格納用RDエリア名)境界値,
                (表格納用RDエリア名))
       |〔FIX〕 HASH 〔ハッシュ関数名〕 BY 列名 〔,列名〕…
          IN(表格納用RDエリア名,表格納用RDエリア名,…)}〕
```

```
        〔表オプション〕…
        〔表制約定義〕…
表要素::={列定義｜表制約定義}
   列定義::=列名 データ型〔ARRAY [最大要素数]〕
           〔{列データ抑制指定｜〔列回復制約〕
              {IN {LOB列格納用RDエリア名
                     ｜(LOB列格納用RDエリア名)
                     ｜((LOB列格納用RDエリア名)
                     〔,(LOB列格納用RDエリア名)〕…)}
                ｜抽象データ型定義内LOB格納用RDエリア指定}}〕
             〔プラグイン指定〕
             〔列制約…〕
   列データ抑制指定::=〔SUPPRESS〕
   列回復制約::=〔RECOVERY〔{ALL｜PARTIAL｜NO}〕〕
列制約::={非NULL値制約指定
          ｜単一列一意性制約定義〔インデクスオプション〕…}
   非NULL値制約指定::={〔NULL｜NOT NULL〔WITH DEFAULT〕〕
                      ｜〔〔NOT NULL〕 WITH DEFAULT〕}
   単一列一意性制約定義::=
   〔UNIQUE〕 CLUSTER KEY〔{ASC｜DESC}〕
             〔IN{インデクス格納用RDエリア名
               ｜(インデクス格納用RDエリア名)
               ｜((インデクス格納用RDエリア名)
               ｜〔,(インデクス格納用RDエリア名)〕…)}〕
    インデクスオプション::={PCTFREE=未使用領域の比率
                           ｜UNBALANCED SPLIT}
表制約定義::=複数列一意制約定義
          〔インデクスオプション〔インデクスオプション〕〕
  複数列一意性制約定義::=
          〔UNIQUE〕 CLUSTER KEY(列名〔{ASC｜DESC}〕
          〔,列名〔{ASC｜DESC}〕〕…)
          〔IN{インデクス格納用RDエリア名
             ｜(インデクス格納用RDエリア名)
             ｜((インデクス格納用RDエリア名)
                〔,(インデクス格納用RDエリア名)〕…)}〕
格納条件::=列名{=｜<>｜^=｜!=｜<｜>=｜>｜>=}
          {定数｜(定数〔,定数〕…)}
  ハッシュ関数名::={HASH1｜HASH2｜HASH3｜HASH4｜HASH5｜HASH6}
表オプション::=
  PCTFREE={未使用領域の比率
       ｜(〔未使用領域の比率〕,
           セグメント内の空きページ比率)}
  LOCK{ROW｜PAGE}
  SUPPRESS〔DEC〔IMAL〕〕
  WITHOUT ROLLBACK
抽象データ型定義内LOB格納用RDエリア指定::=
 ALLOCATE(属性名〔..属性名〕…
   IN {LOB属性格納用RDエリア名
      ｜(LOB属性格納用RDエリア名)
      ｜((LOB属性格納用RDエリア名)
         〔,(LOB属性格納用RDエリア名)〕)}
     〔,属性名〔..属性名〕…
              IN {LOB属性格納用RDエリア名
                 ｜(LOB属性格納用RDエリア名)
                 ｜((LOB属性格納用RDエリア名)
                    〔,(LOB属性格納用RDエリア名)〕)}〕…)
プラグイン指定::=PLUGIN プラグインオプション
```

(b) メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL「CREATE INDEX 形式 1」

```
CREATE 〔UNIQUE〕 INDEX 〔認可識別子.〕インデクス識別子
  ON 〔認可識別子.〕表識別子(列名〔{ASC｜DESC}〕
                 〔,列名〔{ASC｜DESC}〕〕…)
  〔IN{RDエリア名
      ｜(RDエリア名)
      ｜((RDエリア名)〔,(RDエリア名)〕…)}〕
  〔インデクスオプション〕…
```

```
インデクスオプション::={PCTFREE=未使用領域の比率
                      |UNBALANCED SPLIT
                      |EMPTY
                      |除外値指定}
除外値指定::=EXCEPT VALUES (NULL〔,NULL〕…)
```

(c) メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL「CREATE INDEX　形式 2」

```
CREATE INDEX 〔認可識別子.〕インデクス識別子
   USING TYPE 〔認可識別子.〕インデクス型識別子
   ON 〔認可識別子.〕表識別子（列名）
   IN{RDエリア名
       |(RDエリア名)
       |((RDエリア名)〔,(RDエリア名)〕…)}
   〔PLUGIN プラグインオプション〕
```

6.2.3 クラスの削除（オブジェクトが存在しない場合）

オブジェクトが存在しないクラスを削除するには，メタ情報の削除コマンド（EDMDelMeta）によってクラスの定義の削除，およびクラスに対応する表を削除するデータベース定義文の生成を実行します。

ただし，基本単位が VariableArray 型のプロパティの要素を格納するために定義する edmClass_Struct クラスのサブクラスの削除は，次に示す条件を満たしている必要があります。

edmClass_Struct クラスのサブクラスの削除の条件

削除する edmClass_Struct クラスのサブクラスを要素とする，基本単位が VariableArray 型のプロパティを定義しているクラスが存在しない。

条件に満たない edmClass_Struct クラスのサブクラスだけの削除は，行わないでください。

オブジェクトが存在しないクラスを削除する手順を次に示します。

1. **メタ情報のバックアップを取得します。**
   「6.2.2(1)　クラスの追加手順」のメタ情報のバックアップの取得に関する記述を参照してください。

2. **データベースのバックアップを取得します。**
   クラスに対応する表を削除するので，データベースのバックアップを取得します。バックアップを取得する必要がある RD エリアについては，マニュアル「HiRDB　システム運用ガイド」の，障害発生に備えた運用に関する説明での，定義系 SQL「DROP TABLE」の実行についての記述を参照してください。

3. **メタ情報の削除コマンド（EDMDelMeta）を実行します。**
   メタ情報の削除コマンド（EDMDelMeta）でクラスの定義を削除します。
   メタ情報の削除コマンド（EDMDelMeta）の -r オプション，-s オプションおよび -o オプションを指定して実行します。-o オプションにはクラスに対応する表を削除する定義系 SQL「DROP TABLE」を格納するデータベース定義文格納ファイルを指定します。
   メタ情報の削除コマンド（EDMDelMeta）は，削除するクラスに対して次に示す条件を満たすプロパティを定義している場合，基本単位が VariableArray 型のプロパティの要素を格納するために定義している edmClass_Struct クラスのサブクラスに対応する表を削除する定義系 SQL「DROP TABLE」を出力します。
   - 基本単位が VariableArray 型のプロパティが存在する。
   - 基本単位が VariableArray 型のプロパティの要素をクラスに対応する表とは別の表に格納する指定である。
   - ほかのクラスで基本単位が VariableArray 型の同じプロパティの定義を使用していない。

   この場合，基本単位が VariableArray 型のプロパティの要素を格納する edmClass_Struct クラスのサ

ブクラスも削除してください。
基本単位が VariableArray 型のプロパティを定義しているクラスを削除する場合，メタ情報の削除コマンド（EDMDelMeta）が出力するデータベース定義文については，「6.2.7　メタ情報の追加・削除コマンドの動作」を参照してください。

4. **データベース定義ユティリティを実行します。**
手順 3. で作成したデータベース定義文を入力ファイルとして指定し，HiRDB のデータベース定義ユティリティ（pddef）を実行して，クラスに対応する表を削除します。詳細については「6.2.2(1)　クラスの追加手順」のデータベース定義ユティリティの実行に関する記述を参照してください。

5. **クラス定義情報ファイルを作成します。**
削除したクラスの定義を反映したクラス定義情報ファイルを DocumentBroker オブジェクト操作ツールの動作環境に配置します。詳細については，「6.2.2(1)　クラスの追加手順」のクラス定義情報ファイルの作成に関する記述を参照してください。

6. **DocumentBroker Object Loader の入力データを作成します。**
削除したクラスの定義を反映した DocumentBroker Object Loader の入力データを作成します。入力データは，DocumentBroker Object Loader を使用している場合に，クラス定義と入力ファイルを一致させるために作成します。詳細については，「6.2.2(1)　クラスの追加手順」の DocumentBroker Object Loader の入力データに関する記述を参照してください。

6.2.4　クラスの定義の変更（オブジェクトが存在しない場合）

オブジェクトが存在しないクラスの定義の変更では，プロパティの追加，プロパティの定義の変更，またはプロパティの削除のどれかがあります。どの場合についても，クラスの定義を変更するには，次に示す方法で行います。

● 変更前のクラスを削除します。

● 変更後のクラスを追加します。

オブジェクトが存在しないクラスの定義を変更する手順を次に示します。

1. **定義情報ファイルを作成します。**
変更後のクラスの定義を記述した定義情報ファイルを，変更前の定義情報ファイルを基に作成します。作成する定義情報ファイルのクラスの定義，プロパティの定義のセクションに指定するエントリ「dmaProp_Ids」の値であるクラス識別子，プロパティ識別子は，定義の変更前と同じ値を指定してください。
定義情報ファイルの作成については，「4.7　定義情報ファイル」を参照してください。

2. **インデクス情報ファイルを作成します。**
「6.2.2(1)　クラスの追加手順」のインデクス情報ファイルの作成に関する記述を参照してください。

3. **RD エリア定義情報ファイルを作成します。**
「6.2.2(1)　クラスの追加手順」の RD エリア定義情報ファイルの作成に関する記述を参照してください。

4. **メタ情報のバックアップを取得します。**
「6.2.2(1)　クラスの追加手順」のメタ情報のバックアップの取得に関する記述を参照してください。

5. **データベースのバックアップを取得します。**
「6.2.2(1)　クラスの追加手順」および「6.2.3　クラスの削除（オブジェクトが存在しない場合）」のデータベースのバックアップの取得に関する記述を参照してください。

6. **メタ情報の削除コマンド（EDMDelMeta）を実行します。**

定義を変更するクラスの定義をメタ情報の削除コマンド（EDMDelMeta）で削除します。詳細については，「6.2.3　クラスの削除（オブジェクトが存在しない場合）」を参照してください。

7. **データベース定義ユティリティを実行します。**
   手順 6. で作成したデータベース定義文を入力ファイルとし，HiRDB のデータベース定義ユティリティ（pddef）を実行して定義を変更するクラスのオブジェクトを格納する表を削除します。詳細については，「6.2.2(1)　クラスの追加手順」のデータベース定義ユティリティの実行に関する記述を参照してください。

8. **メタ情報の追加コマンド（EDMAddMeta）を実行します。**
   手順 1.，手順 2.，および手順 3. で作成した定義情報ファイル，およびインデクス定義を指定して，変更するクラスの定義をメタ情報の追加コマンド（EDMAddMeta）で追加します。詳細については，「6.2.2(1)　クラスの追加手順」のメタ情報の追加コマンド（EDMAddMeta）の実行に関する記述を参照してください。

9. **データベース定義ユティリティを実行します。**
   手順 8. で作成したデータベース定義文を入力ファイルとし，HiRDB のデータベース定義ユティリティ（pddef）を実行して，定義を変更するクラスに対応する表を追加します。詳細については，「6.2.2(1)　クラスの追加手順」のデータベース定義ユティリティの実行に関する記述を参照してください。

10. **クラス定義情報ファイルを作成します。**
    変更したクラスの定義を反映したクラス定義情報ファイルを DocumentBroker オブジェクト操作ツールの動作環境に配置します。詳細については，「6.2.2(1)　クラスの追加手順」のクラス定義情報ファイルの作成に関する記述を参照してください。

11. **表へオブジェクトを格納します。**
    「6.2.2(1)　クラスの追加手順」の表へのオブジェクトの格納に関する記述を参照してください。

6.2.5　クラスの削除（オブジェクトが存在する場合）

オブジェクトが存在するクラスを削除する方法を次に示します。

● 削除するクラスのすべてのオブジェクトを削除します。

● クラスを削除します。

オブジェクトを削除するには，クライアントアプリケーション構築製品が提供するメソッドを使用します。例えば，DocumentBroker Development Kit が提供する C++ クラスライブラリを使用している場合は，CdbrDMA::RemoveObject メソッドを使用します。クラスを削除するには，「6.2.3　クラスの削除（オブジェクトが存在しない場合）」を参照してください。

CdbrDMA::RemoveObject 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　C++　リファレンス　基本機能編」を参照してください。

6.2.6　クラスの定義の変更（オブジェクトが存在する場合）

オブジェクトが存在するクラスの定義の変更は，プロパティを追加することで実現します。ここでは，プロパティの追加について説明します。

(1) プロパティの追加

オブジェクトが存在するクラスにプロパティを追加するには，追加するプロパティの定義を記述した定義情報ファイルを作成します。そして，作成した定義情報ファイルをメタ情報の追加コマンド（EDMAddMeta）で指定して，プロパティの定義の追加，およびプロパティに対応する列を追加するデー

タベース定義文の生成を実行します。追加するプロパティとプロパティに対応する表の列の並びについては，「6.2.8　追加するプロパティと列の並び」を参照してください。

プロパティの追加の手順を次に示します。

(2) プロパティの追加手順

プロパティの追加の手順では次のことに注意する必要があります。

● 抽象データ型

全文検索機能を使用していない定義済みの dmaClass_DocVersion クラスのサブクラスは，HiRDB の抽象データ型で定義する列を使用するため，全文検索機能を使用する場合に必要な次のプロパティを追加できません。

- 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
- edmProp_Content 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
- 全文検索機能付き文字列型プロパティ

なお，全文検索機能を使用している定義済みの dmaClass_DocVersion クラスのサブクラスには，データ型が Boolean 型，Integer32 型，および String 型のプロパティを追加できます。

オブジェクトが存在するクラスの定義を変更する手順を次に示します。

1. **定義情報ファイルを作成します。**
   追加するプロパティの定義を記述する定義情報ファイルを作成します。
   定義情報ファイルの作成については，「4.7　定義情報ファイル」を参照してください。

2. **インデクス情報ファイルを作成します。**
   追加するプロパティに対応する列の値は，手順 5. で示すように NULL 値になります。これによって，追加するプロパティにインデクスを定義する定義系 SQL「CREATE INDEX」で「除外値指定」はできません。インデクス情報ファイルの作成については，「6.2.2(1)　クラスの追加手順」のインデクス情報ファイルの作成に関する記述を参照してください。

3. **メタ情報のバックアップを取得します。**
   「6.2.2(1)　クラスの追加手順」のメタ情報のバックアップの取得に関する記述を参照してください。

4. **データベースのバックアップを取得します。**
   表に列を追加するので，データベースのバックアップを取得します。バックアップを取得する必要がある RD エリアについては，マニュアル「HiRDB　システム運用ガイド」の，障害発生に備えた運用に関する説明での，定義系 SQL「ALTER TABLE」の実行についての記述を参照してください。

5. **メタ情報の追加コマンド（EDMAddMeta）を実行します。**
   手順 1.，および手順 2. で作成した定義情報ファイル，およびインデクス情報ファイルを指定して，メタ情報の追加コマンド（EDMAddMeta）を実行して，プロパティを追加します。
   メタ情報の追加コマンド（EDMAddMeta）の -o オプションで，列を追加する定義系 SQL「ALTER TABLE」を格納するデータベース定義文格納ファイルを指定します。メタ情報の追加コマンド（EDMAddMeta）は，データベース定義文格納ファイルへ定義系 SQL「ALTER TABLE」の「ADD」句のすべてのオプションを出力しません。このため，必要なオプションは，データベース定義文格納

ファイルをテキストエディタなどで修正して追加してください。メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL については，「(3) メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL」を参照してください。
次に，追加したプロパティの値について説明します。ここでは，例として，C++ クラスライブラリのメソッドを使用して説明します。
定義系 SQL「ALTER TABLE」の「ADD」句で追加する列の値は，NULL 値が設定されます。このため，追加するプロパティの値の基本単位が「Single」で，追加するプロパティのデータ型が「Boolean 型」，「Integer32 型」，または「String 型」の場合，追加した列の値は「NULL 値」となっているので，追加したプロパティの値を参照するメソッドである CdbrDMA::GetPropertyValues メソッドで取得する構造体 SDBR_PROP のメンバ uniValue の値は「NULL」となります。なお，CdbrDMA::GetPropertyValuesAndLock メソッドの場合も同様です。
また，追加するプロパティの基本単位が VariableArray 型の場合，CdbrVariableArray::GetCount メソッドで取得する要素数は 0 となります。
メタ情報の追加コマンド（EDMAddMeta）で追加するプロパティ定義に指定する初期値は，UAP でオブジェクトを新規に作成する場合の初期値であり，追加するプロパティには値を設定しません。

6. **データベース定義ユティリティを実行します。**
   手順 5. で作成したデータベース定義文を入力ファイルとして指定し，HiRDB のデータベース定義ユティリティ（pddef）を実行して，追加するプロパティに対応する列を追加します。詳細については，「6.2.2(1) クラスの追加手順」のデータベース定義ユティリティの実行に関する記述を参照してください。

7. **クラス定義情報ファイルを作成します。**
   追加したプロパティの定義を反映したクラス定義情報ファイルを DocumentBroker オブジェクト操作ツールの動作環境に配置します。詳細については，「6.2.2(1) クラスの追加手順」のクラス定義情報ファイルの作成に関する記述を参照してください。

8. **DocumentBroker Object Loader の入力データを作成します。**
   追加したプロパティの定義を反映した DocumentBroker Object Loader の入力データを作成します。入力データは，DocumentBroker Object Loader を使用している場合に，クラス定義と入力ファイルを一致させるために作成します。詳細については，「6.2.2(1) クラスの追加手順」の DocumentBroker Object Loader の入力データに関する記述を参照してください。

(3) メタ情報の追加コマンド（EDMAddMeta）が出力する定義系 SQL

メタ情報の追加コマンド（EDMAddMeta）が出力する HiRDB のデータベース定義ユティリティ（pddef）の入力となる定義系 SQL を次に示します。下線が付いている個所は，メタ情報の追加コマンド（EDMAddMeta）が出力しない定義系 SQL を示します。

```
ALTER TABLE 〔認可識別子.〕表識別子
      {ADD {列名 データ型〔ARRAY ［最大要素数］〕
                {{〔NULL | NOT NULL〔WITH DEFAULT〕〕
                  | 〔〔NOT NULL〕 WITH DEFAULT〕}
                〔WITH PROGRAM〕
```

6.2.7 メタ情報の追加・削除コマンドの動作

基本単位が VariableArray 型であるプロパティをメタ情報の追加コマンド（EDMAddMeta）で追加する場合と，メタ情報の削除コマンド（EDMDelMeta）で削除する場合の動作を次の表に示します。

表 6-1 メタ情報の追加コマンド（EDMAddMeta）の出力内容

格納先	ほかのクラスからの使用	メタ情報		定義系 SQL	
		プロパティの定義	edmClass_Struct クラスのサブクラスの定義	別表の作成	ARRAY で定義する列の追加
別表	あり	追加しない	追加しない※	出力しない	－
	なし	追加する	追加しない※	出力する	－
ARRAY	あり	追加しない	追加しない※	－	出力する
	なし	追加する	追加しない※	－	出力する

注※
要素を格納するクラスの追加は，プロパティを追加する前に行ってください。

表 6-2 メタ情報の削除コマンド（EDMDelMeta）の出力内容

格納先	ほかのクラスからの使用	メタ情報		定義系 SQL	
		プロパティの定義	edmClass_Struct クラスのサブクラスの定義	別表の削除	ARRAY で定義する列の削除
別表	あり	削除しない	削除しない※	出力しない	－
	なし	削除する	削除しない※	出力する	－
ARRAY	あり	削除しない	削除しない※	－	出力する
	なし	削除する	削除しない※	－	出力する

注※
要素を格納するクラスを使用しているすべてのプロパティを削除したあと，要素を格納するクラスを削除してください。

次に，表 6-1 および表 6-2 の各列について説明します。

格納先
基本単位を VariableArray 型で定義するプロパティの要素の格納先を示します。

別表
プロパティを定義するクラスに対応する表とは別の表に格納することを示します。

ARRAY
プロパティを定義するクラスに対応する表の HiRDB の繰り返し列に格納することを示します。

ほかのクラスからの使用
削除するプロパティの定義を複数のクラスで使用しているかどうかを示します。

あり
ほかのクラスから使用していることを示します。

なし
ほかのクラスから使用していないことを示します。

メタ情報
メタ情報ファイルに対する各コマンドの操作を「プロパティの定義」,「edmClass_Struct のサブクラスの定義」ごとに示します。

プロパティの定義

基本単位を VariableArray 型で定義しているプロパティの定義をメタ情報へ追加・削除するかどうかを示します。

edmClass_Struct クラスのサブクラスの定義

基本単位を VariableArray 型で定義しているプロパティの定義をメタ情報へ追加・削除する場合，要素を格納するために定義する edmClass_Struct クラスのサブクラスの定義を追加・削除するかどうかを示します。

定義系 SQL

各コマンドが出力する定義系 SQL を示します。

別表の作成・削除

基本単位を VariableArray 型で定義しているプロパティの要素をクラスに対応する表とは別の表に格納するため，その表を作成・削除する定義系 SQL を出力するかどうかを示します。

ARRAY で定義する列の追加・削除

基本単位を VariableArray 型で定義しているプロパティの要素をクラスに対応する表の繰り返し列で定義する列に格納するため，その列を追加・削除する定義系 SQL を出力するかどうかを示します。

6.2.8 追加するプロパティと列の並び

メタ情報の追加コマンド（EDMAddMeta）で定義しているクラスへ追加するプロパティは，クラスの定義でのプロパティの並びとクラスに対応する表での列の並びが異なります。

DocumentBroker Object Loader では，制御ファイルのセクション「DataMapping」に指定するプロパティの並びと入力データファイルのデータの並びが一致していれば，クラスの定義でのプロパティの並びとクラスに対応する表の列の並びを意識する必要はありません。

DocumentBroker は，ユーザが定義したクラスのオブジェクトを制御するため，クラスの定義に現れないプロパティをクラスへ追加しています。作成済みのクラスにプロパティを追加すると，定義しているクラスのプロパティの末尾へ新たにプロパティが追加されます。しかし，ユーザが定義しているクラスに対応する表の列には，DocumentBroker がオブジェクトを制御するため，クラスの定義に現れないプロパティを追加しているので，DocumentBroker が追加したプロパティに対応する列のあとに，ユーザが追加するプロパティに対応する列が表に追加されます。

例えば，作成済みの usrClass_DocVersion クラスに usrProp_Subtitle プロパティを追加する場合，クラスの定義と表の対応は次の図のようになります。

図 6-3 クラスの定義と表の関係

ユーザが定義するusrClass_DocVersionクラスにusrProp_Subtitleプロパティを追加する。

クラスの定義

```
class usrClass_DocVersion       public:dmaClass_DocVersion
{
public:
  String            dmaProp_OlID;
  ・・・

  List<Object>      dmaProp_Renditions;
  String            usrProp_Title;
  String            usrProp_Subtitle;     // クラスの末尾に追加
}
```

usrClass_DocVersionクラスに対応する表

usrClass_DocVersion				
OlID	・・・	Title	DocumentBrokerが定義する列	Subtitle
・・・	・・・	・・・	・・・	・・・
・・・	・・・	・・・	・・・	・・・

usrProp_Subtitle プロパティは，クラスの定義では，プロパティの末尾に追加されています。usrClass_DocVersion クラスに対応する表では，usrProp_Subtitle プロパティに対応する列は，DocumentBroker が定義する列のあとに追加されています。

6.3 リファレンスファイル管理機能を使用する場合の運用

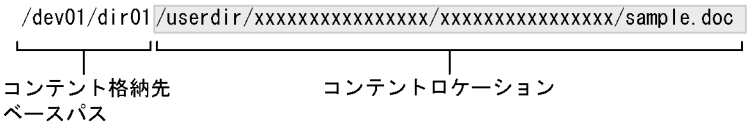
この節では，リファレンスファイル管理機能を使用する場合の，コンテントの格納先ディレクトリの運用方法，およびファイルシステムのバックアップとリストアについて説明します。

6.3.1 コンテントの格納先ディレクトリの運用方法

コンテントの格納先ディレクトリの運用方法として，コンテントロケーションとコンテント格納先ベースパスの管理について説明します。

コンテントの格納先ディレクトリは，コンテント格納先ベースパスとコンテントロケーションから構成されます。コンテントの格納先ディレクトリの例を次の図に示します。

図 6-4 コンテントの格納先ディレクトリの例



この例は，コンテント格納先ベースパスにディレクトリ「/dev01/dir01」を設定し，またコンテント格納先パスにディレクトリ「userdir」を設定して，コンテント「sample.doc」が格納された場合を示しています。

● コンテントロケーションの管理

コンテントロケーションとして，コンテント格納先ベースパスからの相対パスがデータベースに登録されます。コンテントロケーションはデータベースで管理されるため，コンテントロケーションのディレクトリ構成を変更しないでください。

なお，DocumentBroker サーバを運用するマシンの移行などによってコンテントの格納先を変更するときは，コンテントロケーションのディレクトリ構成は変更しないでください。コンテントロケーションのディレクトリ構成を変更すると，ファイルシステム上のコンテント格納先とデータベースで管理するコンテントロケーションが不一致となり，リファレンスファイル文書を操作したときエラーになります。

● コンテント格納先ベースパスの管理

コンテント格納先ベースパスは，ユーザが管理するディレクトリです。そのため，コンテント格納先ベースパスを変更しても，データベースで管理されるコンテントロケーションと不一致にならないので，ユーザがコンテント格納先ベースパスを変更できます。ただし，ファイルシステムのディスクの移行などでコンテント格納先ベースパスを変更する場合は，移行先のディスクでも，コンテントロケーションを移行前と同様のディレクトリ構成にする必要があります。このため，移行前のコンテントロケーションのディレクトリ構成を移行後のコンテント格納先にコピーしてください。

なお，ファイルシステムのディスク移行時などに，コンテントの格納先ベースパスを誤って変更した場合，またはコンテント格納先ベースパスを誤って指定した場合，ファイルシステム上のコンテント格納先とデータベースで管理するコンテントロケーションが不一致となり，リファレンスファイル文書を操作したときエラーになります。

6.3.2 ファイルシステムのバックアップとリストア

コンテントを格納するファイルシステムのバックアップおよびリストアは，OS で提供されるシステムツールなどを利用して実施します。

ファイルシステムのバックアップおよびリストアは，必ず DocumentBroker サーバの停止中に実施してください。また，必ずデータベースのバックアップおよびリストアと同期して実施する必要があります。
データベースのバックアップとリストアについては，「6.1.1　バックアップと回復」を参照してください。

6.4 障害対策

この節では，障害が発生した時に DocumentBroker が出力する情報，障害が発生した場合に考えられる要因とその対処方法について説明します。

なお，トランザクションの処理など，データベースシステムの機能に依存している対処方法については，マニュアル「HiRDB　システム運用ガイド」，またはマニュアル「HiRDB　UAP 開発ガイド」を参照してください。

6.4.1 障害が発生した時に DocumentBroker が出力する情報

ここでは，障害が発生した時に，DocumentBroker が出力する情報について説明します。

なお，Java クラスライブラリが出力する情報については，マニュアル「DocumentBroker Version 3　クラスライブラリ　Java　解説」を参照してください。

(1) メッセージ

運用中のメッセージは，syslog ファイルに出力されます。コマンド実行中のメッセージについては，標準エラー出力に出力されます。メッセージの詳細については，マニュアル「DocumentBroker Version 3 メッセージ」を参照してください。

(2) トレースファイル

発生した障害の切り分けや，原因究明のためにトレースファイルを出力します。関数トレースやメッセージも出力されます。

トレースファイルの出力方法や出力内容については，環境変数で設定できます。

(a) 出力先ディレクトリ

トレースファイルは，環境変数「_HIEDMS_TRACE_DIR」に指定したディレクトリに出力されます。文書空間で使用する文字コード種別が UTF-8 の場合，環境変数「_HIEDMS_TRACE_DIR」には，印刷可能な ASCII コードで値を指定してください。

環境変数「_HIEDMS_TRACE_DIR」を指定した場合の出力先ディレクトリは，次のとおりです。

サーバのトレースファイル
環境変数「_HIEDMS_TRACE_DIR」に指定したディレクトリ /server

クライアントのトレースファイル
環境変数「_HIEDMS_TRACE_DIR」に指定したディレクトリ /client

デフォルトの出力先ディレクトリは，次のとおりです。

サーバのトレースファイル
実行環境ディレクトリ /spool/server

クライアントのトレースファイル
次に示す優先順位で出力先が決まります。
1. 実行環境ディレクトリ /spool/client
2. /var/DocBroker/spool/client
3. カレントディレクトリ

(b) 出力ファイル名

トレースファイルの出力ファイル名は，環境変数に指定できません。次のファイル名で出力されます。

サーバのトレースファイル名
EDMRasTrace"PID"_"NO".log

クライアントのトレースファイル名
EDMRasTraceCL"PID"_"NO".log

注意：「PID」はプロセス識別子です。「NO」はファイル番号です。

(c) トレースファイルの保存日数

トレースファイルは，環境変数「_HIEDMS_TRACE_KEEP_DAYS」に指定した日数だけ保存されます。1 ～ 365 の日数を指定します。デフォルトは 70（日）です。DocumentBroker サーバ起動時，トレースファイルの出力先ディレクトリに，環境変数「_HIEDMS_TRACE_KEEP_DAYS」で指定した保存日数を超えるトレースファイルが存在する場合，これらのファイルは削除されます。

(d) 切り替えファイル数

トレースを出力する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指定できます。環境変数「_HIEDMS_TRACE_NUM」に，切り替えることができるファイルの数を 0，または 2 ～ 16 の値で指定します。デフォルトは 2 です。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します。このとき，ファイルは上書きされます。0 を指定した場合，ファイルは作成されません。この場合，障害の切り分けや障害の原因究明などに必要な情報は出力されませんの注意してください。

(e) ファイルサイズ

トレースを出力するファイルのサイズを，環境変数「_HIEDMS_TRACE_SIZE」に 4,096 ～ 2,147,483,647 の値で指定します。デフォルトは 1,048,576（1 メガバイト）です。

(f) トレースの出力レベル

トレースの出力レベルを変更できます。環境変数「_HIEDMS_TRACE_LEVEL」にトレースレベルを指定します。デフォルトは 10 です。トレースレベルと出力情報について，次の表に示します。

表 6-3 トレースレベルと出力情報

トレースレベル	出力情報
-1※	• トレースヘッダ
0	• エラーなどの必須情報 • サーバの開始 / 終了
10（デフォルト）	• トレースレベルが 0 の場合に出力される情報 • ユーザインターフェースの情報 • ほかのプログラムとのインターフェースの情報 • データベースへの接続 / 切断

注※
トレースレベルが -1 の場合，環境変数「_HIEDMS_TRACE_NUM」にデフォルトの値，または 2 以上の値が指定されていても，トレースを出力するファイルの数は一つです。

6.4.2 サーバマシンでの障害

サーバマシンで障害が発生した場合に考えられる障害の要因と対処について説明します。

(1) 障害の要因

CPU 障害，メモリ障害，オペレーティングシステム障害などによって，サーバがダウンするような場合があります。

(2) DocumentBroker での対処

DocumentBroker は，データベースに格納されていないセッションおよびメモリ中のオブジェクトを破棄します。クライアントアプリケーションへは，TPBroker から異常が通知されます。

なお，コミットされていない仕掛かり中のトランザクションの処理については，データベースシステムの機能に依存します。

(3) ユーザの対処

次のように対処してください。

● 障害が発生しているサーバから出力されるメッセージなどを参照して，障害を取り除いてください。

● そのあと，DocumentBroker を再起動してください。

6.4.3 データベースでの障害

データベースで障害が発生した場合に考えられる障害の要因と対処について説明します。

(1) 障害の要因

次の要因が考えられます。

媒体障害
ディスク障害やジャーナル障害などの発生が考えられます。

システムの停止
通信障害や異常終了によるシステムの停止が考えられます。

(2) DocumentBroker での対処

DocumentBroker は次のような処理を実行します。

● データベースに格納されていないオブジェクトの状態は維持します。

● トランザクションの状態は変更しません。

ただし，データベースシステム上のトランザクションの処理については，データベースシステムの機能に依存します。

(3) ユーザの対処

次のように対処してください。

● クライアントアプリケーションで，トランザクションのロールバックを実行してください。

● DocumentBroker を停止してください。

● 障害が発生しているサーバから出力されるメッセージなどを参照して，障害を取り除いてください。

● 障害が発生しているデータ格納媒体を復旧してください。

● DocumentBroker を再起動してください。

6.4.4 セッション障害

セッションに関して障害が発生した場合に考えられる障害の要因と対処について説明します。

(1) 障害の要因

通信障害，CORBA 層の障害，クライアントアプリケーションの障害などによってセッションが切断される場合があります。

なお，CORBA 層で障害を検知できない場合は，DocumentSpace 構成定義ファイルの SessionTimeOut エントリにセッションのアイドル時間の最大値を指定して，タイムアウトを発生させることもできます。DocumentSpace 構成定義ファイルについては，「4.2 DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

(2) DocumentBroker の対処

DocumentBroker は次のような処理を実行します。

- データベースに格納されていないセッションおよびメモリ中のオブジェクトを破棄します。
- コミットされていない仕掛かり中のトランザクションをロールバックします。

(3) ユーザの対処

次のように対処してください。

- 障害が発生しているサーバから出力されるメッセージなどを参照して，障害を取り除いてください。クライアントアプリケーションの障害の場合は，クライアントアプリケーションの障害を取り除いてください。
- セッションが切断されたクライアントから，再度ログインしてください。

6.4.5 定義情報の障害

定義情報に関して障害が発生した場合に考えられる障害の要因と対処について説明します。

(1) 障害の要因

定義情報の格納媒体の障害や定義内容に誤った定義がされている場合があります。

(2) DocumentBroker の対処

DocumentBroker の起動に失敗します。「(3) ユーザの対処」を参照して，障害の要因を取り除いてください。

(3) ユーザの対処

次のように対処してください。

- 障害が発生しているサーバから出力されるメッセージなどを参照して，障害を取り除いてください。
- 定義情報の格納媒体に障害が発生している場合は，バックアップから定義情報を回復してください。
- DocumentBroker を再起動してください。

なお，メタ情報の定義内容が正しいかどうかを確認する場合には，メタ情報の記述内容の確認コマンド（EDMChkMeta）を使用できます。また，定義されているクラス・プロパティとデータベースに作成されている表・列の定義が一致しているかどうかを確認する場合には，データベースの表・列の確認コマンド

（EDMChkTbl）を使用できます。詳細については，「7.3　コマンドの文法」の「EDMChkMeta（メタ情報の記述内容の確認）」および「EDMChkTbl（データベースの表・列の確認）」を参照してください。

6.4.6　ログの障害

ログに関して障害が発生した場合に考えられる障害の要因と対処について説明します。

（1）障害の要因

ログ情報が出力される媒体の障害が考えられます。

（2）DocumentBroker の対処

エラーメッセージを出力して，処理は継続します。

（3）ユーザの対処

次のように対処してください。

- DocumentBroker を停止してください。
- 出力されているエラーメッセージなどを参照して，障害を取り除いてください。
- DocumentBroker を再起動してください。

6.4.7　DocumentBroker での障害

DocumentBroker で障害が発生した場合に考えられる障害の要因と対処について説明します。

（1）障害の要因

サーバ監視プロセスまたはサービスプロセスの異常終了が考えられます。

（2）DocumentBroker の対処

DocumentBroker は次のような処理を実行します。

サーバ監視プロセスが異常終了した場合

DocumentBroker はすべてのプロセスを停止して，データベースに格納されていないセッションおよびメモリ中のオブジェクトを破棄します。クライアントアプリケーションへは，TPBroker から異常が通知されます。

なお，コミットされていない仕掛かり中のトランザクションの処理については，データベースシステムの機能に依存します。

サービスプロセスが異常終了した場合

DocumentBroker はサービスプロセスを再起動します。

異常終了したサービスプロセスに接続していたすべてのクライアントに対しては，データベースに格納されていないオブジェクトを破棄して，TPBroker から異常が通知されます。

なお，コミットされていない仕掛かり中のトランザクションの処理については，データベースシステムの機能に依存します。

（3）ユーザの対処

次のように対処してください。

- 出力されているエラーメッセージなどを参照して，障害を取り除いてください。

● サーバ監視プロセスが異常終了した場合は，DocumentBroker を再起動してください。

6.4.8 サーバ側で発生した障害情報の取得

DocumentBroker の運用中に障害が発生した場合，障害の種類を切り分けたり障害の原因を究明したりするために，トレースなどのトラブルシュート情報を取得する必要があります。

サーバ側で発生した障害に関する障害情報を取得するには，サーバ側の障害情報取得コマンド（EDMGetRas）を使用できます。このコマンドを使用すると，必要なトラブルシュート情報を，指定したディレクトリに退避できます。

ここでは，サーバ側の障害情報取得コマンド（EDMGetRas）を実行して取得できる情報の種類，構成および取得情報のカスタマイズ方法について説明します。

（1） 取得できる障害情報の種類

サーバ側の障害情報取得コマンド（EDMGetRas）で取得できる障害情報について，次に示します。

なお，これらの情報のうち，どの情報を取得するかは，サーバ側の障害情報取得コマンド（EDMGetRas）実行時にオプションとして指定する収集種別によって指定できます。

● **$DOCBROKERDIR/spool 下の全ファイル**
システム保守情報を取得できます。

● **$DOCBROKERDIR/etc 下の全ファイル**
ユーザ環境情報を取得できます。

● **$DOCBROKERDIR/tmp 下の全ファイル**
テンポラリファイルを取得できます。

● **$_HIEDMS_TRACE_DIR 下の全ファイル**
トレースファイルを取得できます。
このディレクトリ下のファイルは，環境変数「_HIEDMS_TRACE_DIR」が設定されている場合だけ取得できます。なお，この環境変数が設定されていない場合，トレースファイルは，$DOCBROKERDIR/spool 下に出力されています。

● **詳細エラーログファイル**
クライアントが出力する詳細エラーログファイルを取得できます。
環境変数「DBR_DETAIL_ERRORLOG_DIR」が設定されている場合だけ取得できます。文書空間で使用する文字コード種別が UTF-8 の場合，環境変数「DBR_DETAIL_ERRORLOG_DIR」には，印刷可能な ASCII コードで値を設定してください。

● **OS 情報**
次の情報を取得できます。
- uname -a コマンドの実行結果情報
- lslpp -l コマンドの実行結果情報（AIX の場合）
- rpm -q -a コマンドの実行結果情報（Linux の場合）
- /etc/hosts ファイル

● **環境情報**
env コマンドによって出力される情報を取得できます。

● **TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）**
TPBroker のトレース情報を取得できます。この情報は，TPBroker の環境変数「VBROKER_ADM」が設定されている場合だけ取得できます。なお，この環境変数が設定されていない場合，取得できませ

ん。

● getrascustom.ini の [Path] セクションで指定されたファイル
getrascustom.ini の [Path] セクションで指定されたファイルを取得できます。
詳細については，「(3) 取得する障害情報のカスタマイズ（getrascustom.ini）」を参照してください。

● DocumentBroker Server のバージョン情報（DocBro_Version.txt）
DocumentBroker Server のバージョン情報を取得できます。

(2) 障害情報が出力されるファイル

障害情報は，コマンド実行時にオプションとして指定した出力先ディレクトリ下に作成される，次のファイルに出力されます。

● EDM_< ホスト名 >_<YYYYMMDDhhmmss>.tar.Z（AIX の場合）または EDM_< ホスト名 >_<YYYYMMDDhhmmss>.tar.gz (Linux の場合)
取得した障害情報を圧縮したファイルです。

● EDM_< ホスト名 >_<YYYYMMDDhhmmss>.txt
EDMGetRas コマンド内で発行される OS コマンドの実行結果を格納したファイルです。

ファイル名の文字列の意味は次のとおりです。

< ホスト名 >
コマンドを実行したホスト名（最大 64 バイト）を表す文字列です。

<YYYYMMDDhhmmss>
コマンドを実行した時刻を表す文字列です。
YYYY：西暦年号（4 けた），MM：月（2 けた），DD：日（2 けた），hh：時（2 けた），mm：分（2 けた），ss：秒（2 けた）で構成されます。

(3) 取得する障害情報のカスタマイズ（getrascustom.ini）

取得する障害情報の種類をカスタマイズする場合は，サーバの障害情報取得カスタマイズファイル（getrascustom.ini）を編集します。このファイルは，次のディレクトリに格納されています。

getrascustom.ini の格納ディレクトリ（サーバ側）
実行環境ディレクトリ/etc

getrascustom.ini の構成について説明します。

同一名称のセクションを複数指定した場合，最初に指定したセクションが有効になります。また，指定できないセクションを指定した場合，その指定は無視されます。

なお，各エントリの値として指定できるエントリ値の長さは，1,023 バイトまでです。「;」で始まる行はコメント行として扱われます。

[Path] セクション
コマンドで取得する情報以外に取得したい情報がある場合に指定します。
「エントリ名 = フルパス」の形式で指定してください。エントリ名は，32 バイト以内の文字列で指定してください。
指定したパスがファイルの場合は，そのファイルだけを収集します。ディレクトリの場合は，そのディレクトリを階層ごと収集します。
[Path] セクションで指定した情報は，次のファイルに出力されます。
- エントリ名 .tar

getrascustom.ini の記述例

```
[Path]
syslog=/var/adm/syslog
```

6.4.9 クライアント側で発生した障害情報の取得

DocumentBroker の運用中に障害が発生した場合，障害の種類を切り分けたり障害の原因を究明したりするために，トレースなどのトラブルシュート情報を取得する必要があります。

クライアント側で発生した障害に関する障害情報を取得するには，クライアント側の障害情報取得コマンド（EDMGetRasCL）を使用できます。このコマンドを使用すると，必要なトラブルシュート情報を指定したディレクトリに退避できます。

ここでは，クライアント側の障害情報取得コマンド（EDMGetRasCL）を実行して取得できる障害情報の種類，構成および取得情報のカスタマイズ方法について説明します。

Java クラスライブラリが出力するトレースファイルは取得の対象ではありません。

（1） 取得できる障害情報の種類

クライアント側の障害情報取得コマンド（EDMGetRasCL）で取得できる障害情報について，次に示します。

なお，これらの情報のうち，どの情報を取得するかは，クライアント側の障害情報取得コマンド（EDMGetRasCL）実行時にオプションとして指定する収集種別によって指定できます。

- **$_HIEDMS_FTPDIR/spool 下の全ファイル**
  ファイル転送サービスの保守情報を取得できます。
  このディレクトリ下のファイルは，環境変数「_HIEDMS_FTPDIR」が設定されている場合だけ取得できます。

- **$_HIEDMS_FTPDIR/etc 下の全ファイル**
  ファイル転送サービスのユーザ環境情報を取得できます。
  このディレクトリ下のファイルは，環境変数「_HIEDMS_FTPDIR」が設定されている場合だけ取得できます。

- **$_HIEDMS_FTPDIR/tmp 下の全ファイル**
  ファイル転送サービスのテンポラリファイルを取得できます。
  このディレクトリ下のファイルは，環境変数「_HIEDMS_FTPDIR」が設定されている場合だけ取得できます。

- **クライアントおよびファイル転送サービスのトレースファイル**
  クライアントおよびファイル転送サービスのトレースファイルを取得できます。
  取得するトレースファイルは，次の順序で検索されます。
  1. $_HIEDMS_TRACE_DIR
  2. $DOCBROKERDIR/spool
  3. /var/DocBroker/spool/client
  4. カレントディレクトリ

  環境変数「_HIEDMS_TRACE_DIR」を設定していない場合は，getrascustom.ini ファイルに出力先を指定することをお勧めします。

- **詳細エラーログファイル**
  クライアントが出力する詳細エラーログファイルを取得できます。
  環境変数「DBR_DETAIL_ERRORLOG_DIR」が設定されている場合だけ取得できます。文書空間で

使用する文字コード種別が UTF-8 の場合，環境変数「DBR_DETAIL_ERRORLOG_DIR」には，印刷可能な ASCII コードで値を設定してください。

● **OS 情報**

次の情報を取得できます。

- uname -a コマンドの実行結果情報
- lslpp -l コマンドの実行結果情報（AIX の場合）
- rpm -q -a コマンドの実行結果情報（Linux の場合）
- /etc/hosts ファイル

● **環境情報**

env コマンドによって出力される情報を取得できます。

● **TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）**

TPBroker のトレース情報を取得できます。この情報は，TPBroker の環境変数「VBROKER_ADM」が設定されている場合だけ取得できます。なお，この環境変数が設定されていない場合，取得できません。

● **getrascustom.ini の [Path] セクションで指定されたファイル**

getrascustom.ini の [Path] セクションで指定されたファイルを取得できます。
詳細については，「(3) 取得する障害情報のカスタマイズ（getrascustom.ini）」を参照してください。

● **DocumentBroker Development Kit および DocumentBroker Runtime のバージョン情報（DocBro_Version.txt）**

DocumentBroker Development Kit および DocumentBroker Runtime のバージョン情報を取得できます。

(2) 障害情報が出力されるファイル

障害情報は，コマンド実行時にオプションとして指定した出力先ディレクトリ下に作成される，次のファイルに出力されます。

● EDMCL_< ホスト名 >_<YYYYMMDDhhmmss>.tar.Z（AIX の場合）または EDM_< ホスト名 >_<YYYYMMDDhhmmss>.tar.gz (Linux の場合)
取得した障害情報を圧縮したファイルです。

● EDMCL_< ホスト名 >_<YYYYMMDDhhmmss>.txt
EDMGetRasCL コマンド内で発行される OS のコマンドの実行結果を格納したファイルです。

ファイル名の文字列の意味は次のとおりです。

< ホスト名 >
　コマンドを実行したホスト名（最大 64 バイト）を表す文字列です。

<YYYYMMDDhhmmss>
　コマンドを実行した時刻を表す文字列です。
　YYYY：西暦年号（4 けた），MM：月（2 けた），DD：日（2 けた），hh：時（2 けた），mm：分（2 けた），ss：秒（2 けた）で構成されます。

(3) 取得する障害情報のカスタマイズ（getrascustom.ini）

取得する障害情報の種類をカスタマイズする場合は，クライアントの障害情報取得カスタマイズファイル（getrascustom.ini）を編集します。このファイルは，次のディレクトリに格納されています。

getrascustom.ini の格納ディレクトリ（クライアント側）

```
/opt/HiEDMS/client/etc
```

このファイルの記述内容については，サーバ側の障害情報の取得コマンドで使用する getrascustom.ini と同じです。詳細については，「6.4.8(3) 取得する障害情報のカスタマイズ（getrascustom.ini)」を参照してください。

なお，クライアント側の障害情報の取得コマンド（EDMGetRasCL）で参照される getrascustom.ini を編集する場合，次の手順で編集してください。

1. **getrascustom.ini をローカルディレクトリにコピーします。**
2. **コピーした getrascustom.ini を編集して保存します。**
3. **編集が完了した getrascustom.ini を格納したディレクトリのフルパスを，環境変数「_HIEDMS_GETRAS_INI」として設定します。**
   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 コードで値を設定してください。

6.4.10 データベースで発生したデッドロック情報およびタイムアウト情報の取得

DocumentBroker の運用中にデータベース（HiRDB）でデッドロックやタイムアウトが発生した場合，障害の種類を切り分けたり，障害の原因を究明したりするために，トレースなどのトラブルシュート情報を取得します。

データベースで発生したデッドロックおよびタイムアウトに関する情報を取得するためには，データベースのデッドロックおよびタイムアウトの監視コマンド（EDMLckWatcher）を使用できます。このコマンドを使用すると，必要なトラブルシュート情報を，指定したディレクトリに退避できます。また，退避した情報を圧縮することもできます。

データベースのデッドロックおよびタイムアウトの監視コマンド（EDMLckWatcher）で取得できる情報や出力ディレクトリの詳細については，「7.3 コマンドの文法」の「EDMLckWatcher（データベースのデッドロックおよびタイムアウトの監視)」を参照してください。

6.5 アクセスログに関する運用

この節では，アクセスログ取得機能を使用する場合の設定について説明します。

6.5.1 アクセスログの取得

ここでは，アクセスログの出力先とアクセスログを取得するための設定について説明します。

(1) アクセスログファイル

アクセスログファイルは，実行環境ディレクトリ /spool/aclog に格納されています。

(2) アクセスログを取得する場合の制御情報の設定

アクセスログを取得する場合の制御情報は，DocumentSpace 構成定義ファイルの［Entry0001］セクションを構成する次のエントリの指定方法に依存します。

- AcLogUse エントリ
- AcLogLevel エントリ
- AcLogFileCount エントリ
- AcLogFileSize エントリ

アクセスログの管理方法に従って，最適な設定をしてください。各エントリの指定方法については，「4.2 DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

(3) 注意事項

アクセスログ取得機能は，DocumentBroker サーバを再起動した場合，継続して最終更新ファイルにアクセスログを出力します。

6.5.2 アクセスログの出力形式

アクセスログファイルの出力形式を次に示します。

yyyy/mm/dd hh:mm:ss. ms UserID LogInformation

なお，LogInformation 内は，スペースで区切られて出力されます。カラム位置は調整されません。次に，アクセスログの各項目を説明します。

yyyy/mm/dd
: アクセスした日付（年，月，日）を示します。

hh:mm:ss. ms
: アクセスした時刻（時，分，秒，ミリ秒）を示します。

UserID
: アクセスしたユーザのユーザ識別子を示します。

LogInformation
: 出力ログ情報を示します。

6.5.3 出力ログ情報

DocumentBroker サーバの処理に対して出力される出力ログ情報を次の表に示します。

表 6-4 出力ログ情報

操作	出力ログ情報	出力レベル
セッションの確立	CONNECT < 文書空間識別子 >	W，R，E
セッションの切断	DISCONNECT	W，R，E
オブジェクトの作成	CREATE OBJECT <OIID>	W，R，E
オブジェクトの削除	DELETE OBJECT <OIID>	W，R，E
オブジェクトとの接続	CONNECT OBJECT <OIID>	R，E
オブジェクトの接続の解除	RELEASE OBJECT <OIID>	R，E
オブジェクト一覧の取得	LIST OBJECTS <OIID>	R，E
オブジェクトを包含するコンテナ一覧の取得	LIST PARENTS <OIID>	R，E
コンテナに包含されるオブジェクト一覧の取得	LIST CHILDREN <OIID>	R，E
	LIST CHILDREN <OIID>< バージョン識別子 >	R，E
オブジェクト間の関連づけの設定	LINK <OIID><OIID>	W，R，E
	LINK < リンク識別子 ><OIID><OIID>	W，R，E
	LINK <OIID><OIID>< バージョン識別子 >	W，R，E
オブジェクト間の関連づけの解除	UNLINK < リンク識別子 ><OIID>	W，R，E
	UNLINK < リンク識別子 ><OIID><OIID>	W，R，E
	UNLINK < リンク識別子 ><OIID><OIID>< バージョン識別子 >	W，R，E
ACL のバインド	BIND ACLS <OIID>	W，R，E
	BIND ACLS <OIID>< バージョン識別子 >	W，R，E
ACL のバインドの解除	UNBIND ACLS <OIID>	W，R，E
	UNBIND ACLS <OIID>< バージョン識別子 >	W，R，E
ACL 一覧の取得	LIST ACLS <OIID>	R，E
	LIST ACLS <OIID>< バージョン識別子 >	R，E
バージョンのチェックイン	CHECKIN VERSION <OIID>	W，R，E
バージョンのチェックアウト	CHECKOUT VERSION <OIID>	W，R，E
チェックアウトの取り消し	REVOKE VERSION <OIID>	W，R，E
バージョンの削除	DELETE VERSION <OIID>< バージョン識別子 >	W，R，E
バージョンの固定	SET MODE FIX <OIID>	W，R，E
	SET MODE FIX < リレーション識別子 > <OIID>	W，R，E
	SET MODE FIX < リンク識別子 > <OIID>	W，R，E
バージョンの固定の解除	SET MODE FLOATING <OIID>	W，R，E
	SET MODE FLOATING < リンク識別子 ><OIID>	W，R，E
バージョン一覧の取得	LIST VERSIONS <OIID>	R，E
バージョン管理情報一覧の取得	LIST VERSIONABLES <OIID>	R，E
リザベーションの取得	GET RESERVATION <OIID>	R，E

操作	出力ログ情報	出力レベル
リレーション情報の作成	CREATE RELATION <OIID><OIID>	W，R，E
	CREATE RELATION <OIID>< バージョン識別子 ><OIID>< バージョン識別子 >	W，R，E
リレーション情報の削除	DELETE RELATION <OIID>	W，R，E
リレーション情報の一覧取得	LIST RELATIONS <OIID>	R，E
	LIST RELATIONS <OIID>< バージョン識別子 >	R，E
	LIST RELATIONS < バージョン識別子 >	R，E
レンディションの作成	CREATE RENDITION <OIID>	W，R，E
	CREATE RENDITION <OIID>< バージョン識別子 >	W，R，E
レンディションの削除	DELETE RENDITION <OIID>	W，R，E
	DELETE RENDITION <OIID>< バージョン識別子 >	W，R，E
マスタレンディションの変更	CHANGE MASTER RENDITION <OIID>	W，R，E
	CHANGE MASTER RENDITION <OIID>< バージョン識別子 >	W，R，E
レンディションの一覧取得	LIST RENDITIONS <OIID>	R，E
	LIST RENDITIONS <OIID>< バージョン識別子 >	R，E
プロパティの設定	SET PROPERTIES <OIID>	W，R，E
	SET PROPERTIES <OIID>< バージョン識別子 >	W，R，E
	SET PROPERTIES <OIID>< レンディションタイプ >	W，R，E
	SET PROPERTIES < リレーション識別子 ><OIID>	W，R，E
	SET PROPERTIES <OIID>< リンク識別子 >	W，R，E
プロパティの取得	GET PROPERTIES <OIID>	R，E
	GET PROPERTIES <OIID>< バージョン識別子 >	R，E
	GET PROPERTIES <OIID>< リンク識別子 >	R，E
ファイルのアップロード	UPLOAD CONTENT <OIID>	W，R，E
	UPLOAD CONTENT <OIID>< バージョン識別子 >	W，R，E
ファイルのダウンロード	DOWNLOAD CONTENT <OIID>	R，E
	DOWNLOAD CONTENT <OIID>< バージョン識別子 >	R，E
全文検索インデクスの作成	CREATE INDEX <OIID>	W，R，E
	CREATE INDEX <OIID>< バージョン識別子 >	W，R，E
全文検索インデクスの削除	DELETE INDEX <OIID>	W，R，E
	DELETE INDEX <OIID>< バージョン識別子 >	W，R，E
edmSQL の構文チェック	PREPARE	R，E
問い合わせの実行	EXECUTE SEARCH	R，E
	EXECUTE SEARCH <From 句のクラス識別子の並び（16 個まで）>	R，E

操作	出力ログ情報	出力レベル
問い合わせの実行と問い合わせ結果の取得	EXECUTE SEARCH & GET RESULT <From 句のクラス識別子の並び（16 個まで）>	R，E
問い合わせ結果の削除	DELETE RESULT	R，E
エラー	ERROR <major_code><minor_code>	E

（凡例）

W：Write レベル。作成，削除，更新，リンクなどのアクセスレベルについて，アクセスログへの出力を指定する。

R：Read レベル。Write レベルの出力に加えて，セッション確立／切断，プロパティ参照，検索などのアクセスレベルについて，アクセスログへの出力を指定する。

E：Error レベル。Read レベルの出力に加えて，当該 API でエラーが発生した場合についてもアクセスログへの出力を指定する。

次に，出力ログ情報の詳細について説明します。

（1） 文書空間識別子

DocumentSpace 構成定義ファイルの SerialID エントリに指定した値です。

（2） OIID

操作したオブジェクトの OIID です。

（3） バージョン識別子

操作したオブジェクトのバージョンに対する情報を次の表に示します。

表 6-5　バージョン識別子

操作内容	出力文字
バージョン識別子を指定した操作	指定されたバージョン識別子
バージョンなしオブジェクトに対する操作	出力なし
バージョン全体に対する操作	出力なし

（4） リンク識別子

コンテインメントを解除した，または構成管理モードを設定したコンテインメントのリンク識別子です。

（5） リレーション識別子

文書間に設定されたリレーションを識別するための識別子です。

（6） レンディションタイプ

操作したレンディションのレンディションタイプです。レンディションタイプを指定しないで操作した場合は出力されません。

（7） From 句のクラス識別子の並び（16 個まで）

検索を実行する場合，検索条件として指定する From 句のクラス識別子（36 バイト）です。このクラス識別子は，連続して 16 個まで出力されます。出力されたそれぞれのクラス識別子の間には，半角スペース（1 バイト）が入ります。出力形式（△を半角スペースとします）を次に示します。

From 句のクラス識別子の並びの出力形式

クラス識別子△クラス識別子△・・・△クラス識別子

（8） major_code **および** minor_code

クラスライブラリの戻り値です。戻り値の詳細は，マニュアル「DocumentBroker Version 3　メッセージ」を参照してください。

6.6 エラーログに関する運用

この節では，エラーログを使用する場合の設定について説明します。

6.6.1 エラーログの取得

ここでは，エラーログの出力先とエラーログを取得するための設定について説明します。

(1) エラーログファイル

DocumentBroker サーバで発生したエラー情報は，エラーログファイルとして，**実行環境ディレクトリ**/spool/errlog に出力されます。エラー情報とは，出力されるメッセージのうち「error」で示されるメッセージのことです。

(2) エラーログを取得する場合の制御情報の設定

エラーログを取得する場合の制御情報は，DocumentSpace 構成定義ファイルの［DocSpace］セクションを構成する次のエントリの指定方法に依存します。

- ErrLogFileCount エントリ
- ErrLogFileSize エントリ

エラーログの管理方法に従って，最適な設定をしてください。各エントリの指定方法については，「4.2 DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。

6.6.2 エラーログの出力形式

エラーログファイルの出力形式を次に示します。

yyyy/mm/dd hh:mm:ss.ms Pid Tid ClassName MethodName LogInformation

なお，LogInformation 内は，スペースで区切られて出力されます。カラム位置は調整されません。次に，エラーログの各項目を説明します。

yyyy/mm/dd
: エラーが発生した日付（年，月，日）を示します。

hh:mm:ss.ms
: エラーが発生した時間（時，分，秒，ミリ秒）を示します。

Pid
: エラーが発生したサーバプロセス識別子を示します。

Tid
: エラーが発生したサーバプロセスのスレッド識別子を示します。

ClassName
: メッセージを出力したクラス名を示します。

MethodName
: メッセージを出力したメソッド名を示します。

LogInformation
: 出力されたエラーメッセージを示します。

7 コマンドリファレンス

この章では，DocumentBroker で使用するコマンドの種類と文法について説明します。

7.1 **コマンドの種類**

7.2 **コマンドの形式**

7.3 **コマンドの文法**

7.1 コマンドの種類

この節では，DocumentBroker で使用するコマンドの種類について説明します。DocumentBroker で使用するコマンドは，システム運用コマンド，データベース運用コマンド，システム導入支援コマンド，およびトラブルシュートコマンドに分けられます。コマンドを実行できるユーザは，システム管理者だけです。また，これらのコマンドを実行する場合，DocumentBroker の実行環境ディレクトリ下（$DOCBROKERDIR/bin）に格納されているコマンドを使用してください。ただし，「EDMGetRasCL コマンド」は，AIX を使用する場合で，TPBroker V3 を使用するときは「/opt/HiEDMS/client/bin」に格納されているコマンドを使用してください。AIX を使用する場合で，TPBroker V5 を使用するときは「/opt/HiEDMS/client/bin_tp5」に格納されているコマンドを使用してください。Linux を使用する場合は，「/opt/HiEDMS/client/bin」に格納されているコマンドを使用してください。

文書空間で使用する文字コード種別が UTF-8 の場合，使用できるコマンドに制限があります。使用できるコマンドの詳細は，「付録 I　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機能およびコマンド」を参照してください。

7.1.1 システム運用コマンド

システム運用コマンドの一覧を次の表に示します。

表 7-1　システム運用コマンド一覧

コマンド	機能	実行のタイミング
EDMRefresher	サービスプロセスのリフレッシュ	起動中
EDMSetup	DocumentBroker の実行環境の作成と削除	停止中
EDMStart	DocumentBroker の起動	停止中
EDMStop	DocumentBroker の終了	起動中
EDMUsrView	文書空間に接続しているユーザ一覧出力	起動中（引数の指定によっては停止中も実行可能）

（凡例）

起動中：
DocumentBroker サーバの起動中にコマンドを実行することを示します。

停止中：
DocumentBroker サーバの停止中にコマンドを実行することを示します。

7.1.2 データベース運用コマンド

データベース運用コマンドの一覧を次の表に示します。

表 7-2　データベース運用コマンド一覧

コマンド	機能	実行のタイミング	排他モード
EDMAddMeta	メタ情報の追加	停止中	EX
EDMCreateIds	ユーザ定義識別子ファイルの作成		SH
EDMCrtSimMeta	クラス定義情報ファイルの作成		SH
EDMCrtSql	DocumentBroker 用データベース定義文の作成		SH

コマンド	機能	実行のタイミング	排他モード
EDMDelMeta	メタ情報の削除		EX
EDMInitMeta	メタ情報の初期設定		EX
EDMPrintMeta	メタ情報ファイルの出力		EX
EDMRegEnvId	DocumentBroker 実行環境の情報の登録		EX
EDMXmlMap	XML 定義ファイルの追加／更新／削除		SH

（凡例）

停止中：

DocumentBroker サーバの停止中にコマンドを実行できます。

EX：

排他モードが EX，SH のコマンドは同時に実行できません。

該当するコマンドを実行中の場合，EDMStart コマンドは失敗します。

SH：

排他モードが EX のコマンドは同時に実行できません。

排他モードが SH のコマンドは同時に実行できます。

該当するコマンドを実行中の場合，EDMStart コマンドは失敗します。

7.1.3 システム導入支援コマンド

システム導入支援コマンドの一覧を次に示します。

表 7-3 システム導入支援コマンド一覧

コマンド	機能	実行のタイミング
EDMCDefDocSpace	文書空間の定義	停止中 なお，次のコマンドと同時に実行できません。 • EDMAddMeta • EDMChkMeta • EDMChkTbl • EDMCreateIds • EDMCreateSimMeta • EDMCrtSql • EDMDelMeta • EDMInitMeta • EDMPrintMeta • EDMRegEnvId • EDMXmlMap
EDMCBuildDocSpace	文書空間の構築	停止中 なお，次のコマンドと同時に実行できません。 • EDMAddMeta • EDMChkMeta • EDMChkTbl • EDMCreateIds • EDMCreateSimMeta • EDMCrtSql • EDMDelMeta • EDMInitMeta • EDMPrintMeta • EDMRegEnvId • EDMXmlMap

（凡例）

停止中：

DocumentBroker サーバの停止中にコマンドを実行できます。

7.1.4 トラブルシュートコマンド

トラブルシュートコマンドの一覧を次の表に示します。

表 7-4 トラブルシュートコマンド一覧

コマンド	機能	実行のタイミング
EDMChkMeta	メタ情報の記述内容の確認	停止中 なお，次のコマンドと同時に実行できません。 • EDMAddMeta • EDMCDefDocSpace • EDMCBuildDocSpace • EDMChkTbl • EDMDelMeta • EDMInitMeta • EDMPrintMeta • EDMRegEnvId
EDMChkTbl	データベースの表・列の確認	停止中 なお，次のコマンドと同時に実行できません。 • EDMAddMeta • EDMCDefDocSpace • EDMCBuildDocSpace • EDMChkMeta • EDMDelMeta • EDMInitMeta • EDMPrintMeta • EDMRegEnvId
EDMGetRas	サーバ側の障害情報の取得	起動中または停止中
EDMGetRasCL	クライアント側の障害情報の取得	
EDMLckWatcher	データベースのデッドロックおよびタイムアウトの監視	

（凡例）

起動中：

DocumentBroker サーバの起動中にコマンドを実行できます。

停止中：

DocumentBroker サーバの停止中にコマンドを実行できます。

7.2 コマンドの形式

この節では，コマンドの入力形式，使用方法および注意事項について説明します。

(1) 入力形式

コマンドの入力形式を次に示します。

コマンド名称〔オプション…〕

(a) コマンド名

コマンド名は，実行するコマンドのファイル名です。

(b) オプション

オプションの入力形式の規則を次に示します。なお，説明文で使用する「$」はシェルのプロンプト，「cmd」はコマンド名を表します。

オプションの形式

オプションはマイナス記号で始まる文字列で，次に示すように，引数を取らないか，または 1 個の引数を取ります。
形式 1：－オプションフラグ
形式 2：－オプションフラグ＜空白またはタブ＞フラグ引数
（凡例）
オプションフラグ：1 文字の英数字で，英大文字・小文字は区別される
フラグ引数：オプションフラグに対する引数

オプションの指定規則

- フラグ引数を取らないオプションフラグは，一つのマイナス記号のあとにまとめて指定できません。
  誤った指定例：$ cmd -abc
  正しい指定例：$ cmd -a -b -c
- フラグ引数を必要とするオプションフラグのフラグ引数は省略できません。
  例えば，オプションフラグ -a がフラグ引数を取る場合，次のように入力すると -b はフラグ引数とみなされます。

  $ cmd -a -b
- オプションフラグとフラグ引数の間には空白またはタブが必要です。
  誤った指定例：$ cmd -afile
  正しい指定例：$ cmd -a file
- 同じオプションフラグを 2 回以上指定できません。例えば，「$ cmd -a 1 -a 2」とは入力できません。
- マイナス記号だけのオプションは入力できません。例えば，「$ cmd -」と入力すると「-」はコマンド引数とみなされます。
- 文書空間で使用する文字コード種別が UTF-8 の場合に，オプションのフラグ引数としてファイルのパスを指定するとき，パスは印刷可能な ASCII コードで記述します。

(2) 入出力

(a) 入力

入力は，すべてコマンドのオプション，引数の並びです。

(b) 出力

コマンド処理が正常に終了した場合の出力は，すべて標準出力に対して実行します。また，終了コードの一覧を次の表に示します。

表 7-5 終了コード一覧

終了コード	意味
0	正常終了
1	警告つき正常終了
2	引数エラー
それ以外	そのほかのエラー

コマンド処理がエラーになった場合（終了コードが「0」以外）は，メッセージをすべて標準エラー出力に出力します。引数エラーになった場合（終了コードが「2」）は，標準エラー出力にコマンドの使用方法（USAGE）を出力します。出力形式は次のとおりです。

出力形式

Usage:xxxx yyyy

- xxxx：コマンド名称が出力されます。
- yyyy：コマンドの指定形式が表示されます。

出力例

```
Usage:command -a -b option_arg_1 〔-c〕〔-d option_arg_2〕…
        {-e | -f option_arg_3}
```

(3) コマンドの実行可能ファイルに対するアクセス権限

コマンドの実行可能ファイルにアクセスしてコマンドを実行できるユーザはシステム管理者だけです。

7.3 コマンドの文法

この節では，コマンドの文法について説明します。なお，各コマンドは，アルファベット順に説明します。

EDMRefresher コマンド，EDMStop コマンド，および EDMUsrView コマンドは，DocumentBroker サーバの起動中に実行してください。これ以外のコマンドは，DocumentBroker サーバの停止中に実行してください。なお，EDMGetRas コマンド，EDMGetRasCL コマンドおよび EDMLckWatcher コマンドは，DocumentBroker サーバ起動中または DocumentBroker サーバ停止中のどちらでも実行できます。

使用できるデータベースシステムは HiRDB です。したがって，コマンドの文法は HiRDB を使用する場合の記述形式になっています。

EDMAddMeta（メタ情報の追加）

機能

引数に指定した定義情報ファイルの内容を基に，サブクラスとプロパティを追加します。サブクラスまたはプロパティを追加した場合は，自動的に動作環境メタ情報ファイルも更新されます。

形式

```
EDMAddMeta   〔-g〕
              -f 定義情報ファイル名
             〔-D〕
             〔-o データベース定義文格納ファイル
             〔-c〕
             〔-i インデクス情報ファイル名〕
             〔-r RDエリア定義情報ファイル名〕
             〔-u { ALL | PARTIAL | NO }〕
             〔-n〕
             〕
```

オプション

-g

GUID 値を自動設定する場合に指定します。このオプションを省略する場合は，定義情報ファイル内の GUID を定義する個所に GUID 値を指定しておく必要があります。なお，メタ情報ファイル内に GUID 値を指定しているときにこのオプションを指定した場合，メタ情報ファイル内に指定している GUID 値を優先します。

-f 定義情報ファイル名

定義情報ファイル名を相対パスまたは絶対パスで指定します。定義情報ファイルについては，「4.7 定義情報ファイル」を参照してください。

-D

プロパティのデフォルト値をクラスごとに設定する場合に指定します。
複数のクラスに対して同じ名前のプロパティを追加する場合に，クラスごとにプロパティのデフォルト値を設定できます。
-f オプションに指定する定義情報ファイルに，［AddProperty/ クラス名］セクションに続くエントリとして，プロパティのデフォルト値を記述します。
デフォルト値を記述するプロパティを次に示します。

- dmaProp_PropertyDefaultString **プロパティ**
- dmaProp_PropertyDefaultInteger32 **プロパティ**

- dmaProp_PropertyDefaultBoolean **プロパティ**

定義情報ファイルの記述方法については，「4.7　定義情報ファイル」を参照してください。
-D オプションを指定したときと省略したとき，それぞれで有効になるプロパティを次に示します。

-D オプションを指定したとき

定義情報ファイルに設定したクラスごとのプロパティのデフォルト値が有効になります。
定義情報ファイルで，クラスごとのプロパティのデフォルト値の設定を省略すると，データ型ごとに設定したプロパティのデフォルト値の記述が有効になります。

-D オプションを省略したとき

メタ情報に定義済みのプロパティのデフォルト値が有効になります。

-o データベース定義文格納ファイル

作成したデータベース定義文を格納するファイルを相対パスまたは絶対パスで指定します。このオプションを指定した場合，定義情報ファイルの内容によって，データベース定義文に出力される SQL 文が異なります。

- サブクラスを追加する場合
  定義系 SQL「CREATE TABLE」文がデータベース定義文に出力されます。
- 追加済みのクラスにプロパティを追加する場合
  定義系 SQL「ALTER TABLE」文がデータベース定義文に出力されます。

-c

-o オプションを指定した場合だけ指定できるオプションです。
表名，列名に対して注釈を付ける SQL 文を出力する場合に指定します。注釈となる文字列は，定義情報ファイルに指定したクラスおよびプロパティそれぞれの dmaProp_DescriptiveText プロパティに指定されている値（文字列）です。なお，dmaProp_DescriptiveText プロパティの指定を省略した場合は，dmaProp_DisplayName プロパティに指定した値になります。

-i インデクス情報ファイル名

-o オプションを指定した場合だけ指定できるオプションです。
追加するプロパティに対して定義系 SQL「CREATE INDEX」文（インデクス定義文）を出力する場合に，インデクス情報ファイル名を相対パスまたは絶対パスで指定します。インデクス情報ファイルについては，「4.9　インデクス情報ファイル」を参照してください。

-r RD エリア定義情報ファイル名

-o オプションを指定した場合だけ指定できるオプションです。
RD エリア定義情報ファイル名を相対パスまたは絶対パスで指定します。
-o オプションで指定したデータベース定義文格納ファイル中の RD エリア名を，指定した RD エリア定義情報ファイルの情報に従って出力する場合に指定します。RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。

-u { ALL | PARTIAL | NO }

-o オプションを指定した場合だけ指定できるオプションです。
コンテント格納用 RD エリア，および SGMLTEXT データ格納用 RD エリアに対して，データベースの更新ログ取得方式を指定します。

ALL

ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。ログ取得モードで運用すると，ロールバックおよびロールフォワードに必要なデータベースの更新ログを取得します。

PARTIAL

更新前ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。更新前ログ取得モードで運用すると，ロールバックに必要なデータベースの更新ログを取得します。

NO

ログレスモードでユーザ LOB 用 RD エリアを運用する場合に指定します。ログレスモードで運用すると，データベースの更新ログを取得しません。

なお，このオプションを省略した場合，「ALL」が仮定されます。

指定するデータベースの更新ログ取得方式によって，障害が発生した場合のユーザ LOB 用 RD エリアの回復方法は異なります。また，必要となるログ容量も異なります。障害が発生した場合のユーザ LOB 用 RD エリアの回復方法については，マニュアル「HiRDB　システム運用ガイド」を参照してください。また，ログ容量については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

-n

-o オプションを指定した場合だけ指定できるオプションです。

OIID を格納するプロパティに対応する列に，非ナル値制約を与える場合に指定します。非ナル値制約とは，指定した列の値に，ナル値を許さない制約のことです。ただし，次の OIID を格納するプロパティに対応する列には，非ナル値制約を与えません。

- 複数のクラスを一つの表に格納するクラスの場合の，二つ目以降のクラスの OIID を格納するプロパティに対応する列
- HiRDB の繰り返し列に定義される OIID を格納するプロパティに対応する列

注意事項

● このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

● このコマンドを実行する前に，データベースを起動しておいてください。

● このコマンドを実行する前には，EDMInitMeta コマンドでメタ情報をデータベースに登録しておく必要があります。

● データベースの初期設定をする場合は，EDMCrtSql コマンドを実行する前に，このコマンドを実行してください。

● このコマンドはデータベース自身の定義であるエリアの定義は実行しません。必要なエリアの定義は，出力されるデータベース定義文に追加してください。

● このコマンドで追加した定義を削除するには，EDMDelMeta コマンドを実行してください。または，コマンド実行前に取得したデータベースのバックアップをリストアしてください。したがって，コマンドを実行する前にはデータベースのバックアップを取得してください。

● 次に示す場合は，インデクス格納 RD エリアの名称として「USR_INDEX_AREA」が出力されます。

- -i オプションにインデクス情報ファイル名を指定し，-r オプションを省略してこのコマンドを実行した場合
- RD エリア定義情報ファイルの［IndexArea］セクションに，インデクス情報ファイルに記述した項目に対応する RD エリアの定義を記述していない状態で，-i オプションにインデクス情報ファイル名を指定し，-r オプションに RD エリア定義情報ファイル名を指定してこのコマンドを実行した場合

● 実行結果であるデータベース定義文に対してエリア定義などの必要な項目を追加，修正してからデータベースを作成してください。ただし，-r オプションを指定した場合は，RD エリア定義情報ファイルの指定に従って，RD エリア名が出力されます。

- EDMInitMeta コマンドの -v オプションで指定した値によって，出力するデータベース定義文は異なります。
- HiRDB のオンライン再編成機能を使用する場合，-n オプションを指定してください。なお，メタ情報の表および OIID を格納する表は，オンライン再編成機能の対象外です。オンライン再編成機能の対象外の表は，オンライン再編成機能の対象となる表と RD エリアを分けてください。オンライン再編成機能の詳細については，マニュアル「HiRDB Staticizer Option」を参照してください。

EDMCBuildDocSpace（文書空間の構築）

機能

文書空間定義コマンド（EDMCDefDocSpace）の実行によって作成された定義情報を使用して，文書空間を新規に構築します。

文書空間定義コマンド（EDMCDefDocSpace）とこのコマンドを使用して文書空間を構築する場合には，さまざまな定義ファイルを作成して，メタ情報の初期設定コマンド（EDMInitMeta），メタ情報の追加コマンド（EDMAddMeta），DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびクラス定義情報ファイルの作成コマンド（EDMCrtSimMeta）を使用する必要はありません。

このコマンドの実行後に出力される情報は次のとおりです。

出力情報

- データベース定義文格納ファイル（EDMtable.txt）
  データベース定義文を記述したファイルです。必要に応じて，システム管理者がファイルを編集してください。ファイルを編集する場合，このコマンドで，-m print オプションを指定して実行してください。
- クラス定義情報ファイル（< 文書空間識別子 >.ini）
  詳細については，「4.10　クラス定義情報ファイル」を参照してください。

形式

```
EDMCBuildDocSpace
              [ -m { print | exec } ]
              [ -o  出力ディレクトリ名 ]
```

オプション

-m { print | exec }

このコマンドを実行するモードを指定します。
なお，このオプションを省略した場合，「print」が仮定されます。

print

データベース定義文格納ファイルおよびクラス定義情報ファイルを出力します。

exec

データベース定義文格納ファイルおよびクラス定義情報ファイルを作成し，データベース定義を実行します。

データベース定義文格納ファイルおよびクラス定義情報ファイルは，" 実行環境ディレクトリ /env" に出力されます。任意のディレクトリ下にも出力する場合には，-o オプションを指定します。
-m print オプションを指定してデータベース定義文格納ファイルを出力した場合は，HiRDB のデータベース定義ユティリティを使用してデータベース定義を実行してください。

-o 出力ディレクトリ名

データベース定義文格納ファイルおよびクラス定義情報ファイルを任意のディレクトリ下に出力する場合に，ファイルを格納するディレクトリのパス名を相対パスまたは絶対パスで指定します。
このオプションの指定の有無に関係なく，" 実行環境ディレクトリ /env" にデータベース定義文格納ファイル（EDMtable.txt）およびクラス定義情報ファイル（< 文書空間識別子 >.ini）を出力します。

注意事項

- このコマンドを実行する前に，データベースを起動しておいてください。
- このコマンドを実行する前に，文書空間定義コマンド（EDMCDefDocSpace）を実行して，文書空間を構築するための定義情報を作成してください。
- 指定する出力ディレクトリ名のパス長は，末尾のパスの区切り文字を含まないで 218 バイト以内で指定してください。
- コマンド実行後に出力されるファイル名と同一名のファイルがある場合には，出力ファイル名 .bak の名称でバックアップファイルを作成します。このバックアップファイルは，出力ファイルをカスタマイズしたあとコマンドを再度実行した場合に，カスタマイズした内容を確認したいときなどに使用します。ただし，バックアップファイルをそのまま使用して，HiRDB のデータベース定義ユティリティを実行しないでください。コマンド実行時にバックアップファイルがある場合，バックアップファイルは上書きされます。
- このコマンドを実行するときには，DocumentSpace 構成定義ファイルに次のエントリを必ず指定してください。
  - Process エントリ
  - SerialId エントリ
  - DbType エントリ
  - PdHost エントリ
  - PdNamePort エントリ
  - PdUser エントリ
- 文書空間定義コマンド（EDMCDefDocSpace）の実行後，出力される RD エリア定義情報ファイル中の RD エリア名は，文書クラス定義ファイルで指定したクラスやインデクスの数に応じて，文書空間情報ファイルで指定した RD エリア名＋通番の名称に変更されている場合があります。この RD エリア名を変更する場合には，" 実行環境ディレクトリ /env" の RD エリア定義情報ファイルの内容をカスタマイズしてから，このコマンドを実行してください。
- このコマンドの実行後に出力されるデータベース定義文格納ファイルをカスタマイズする場合は，-m print オプションを指定してコマンドを実行してください。その後，出力されたデータベース定義文格納ファイルの内容を変更してください。変更したデータベース定義は，HiRDB のデータベース定義ユティリティを使用して実行します。
- 文書空間定義コマンド（EDMCDefDocSpace）の実行後に出力されるインデクス情報ファイルは，編集しないでください。インデクスの定義を変更する場合には，文書空間定義コマンド（EDMCDefDocSpace）の入力情報となる文書クラス定義ファイルを変更し，再度，文書空間定義コマンド（EDMCDefDocSpace）を実行します。
- アクセス制御機能を使う場合，セキュリティ定義ファイルやユーザ権限定義ファイルは，ユーザが編集してください。セキュリティ定義ファイルやユーザ権限定義ファイルの変更は，文書空間定義コマンド（EDMCDefDocSpace）とこのコマンドの実行には影響しません。

EDMCDefDocSpace（文書空間の定義）

機能

文書空間を新規に構築する場合に必要な定義情報を作成します。

このコマンドを実行すると，追加するクラスや格納する文書数などの情報（文書クラス定義ファイル）や，ユーザが定義する文書空間の構成情報（文書空間情報ファイル）を基に，文書空間を構築するための定義情報やデータベースの容量を見積もるための基礎情報が出力されます。これらの情報を基に，文書空間構築コマンド（EDMCBuildDocSpace）を実行すると，文書空間を構築できます。

このコマンドと文書空間構築コマンド（EDMCBuildDocSpace）を使用して文書空間を構築する場合には，さまざまな定義ファイルを作成して，メタ情報の初期設定コマンド（EDMInitMeta），メタ情報の追加コマンド（EDMAddMeta），DocumentBroker 用データベース定義文の作成コマンド（EDMCrtSql），およびクラス定義情報ファイルの作成コマンド（EDMCrtSimMeta）を使用する必要はありません。

このコマンドの実行後に出力される情報は次のとおりです。次のファイルは，" 実行環境ディレクトリ /env" に出力されます。

出力情報

- 見積もり基礎情報ファイル（EDMestimate.csv）
  データベースの見積もり基データとして使用するファイルです。詳細については，「4.17　見積もり基礎情報ファイル」を参照してください。
- 定義情報ファイル（EDMdefine.txt）
  文書空間構築コマンド（EDMCBuildDocSpace）の入力情報となるファイルです。詳細については，「4.7　定義情報ファイル」を参照してください。
- RD エリア定義情報ファイル（EDMrdarea.txt）
  文書空間構築コマンド（EDMCBuildDocSpace）の入力情報となるファイルです。詳細については，「4.8　RD エリア定義情報ファイル」を参照してください。
- インデクス情報ファイル（EDMindex.txt）
  文書空間構築コマンド（EDMCBuildDocSpace）の入力情報となるファイルです。詳細については，「4.9　インデクス情報ファイル」を参照してください。

形式

```
EDMCDefDocSpace
               -f 文書クラス定義ファイル名
               -s 文書空間情報ファイル名
              〔-e { all | resource } 〕
```

オプション

-f 文書クラス定義ファイル名

文書クラス定義ファイルは，構築する文書空間に追加するクラスを定義したファイルです。文書クラス定義ファイル名を相対パスまたは絶対パスで指定します。このファイルのサンプルファイルは，"/opt/HiEDMS/sample/EDMclassdef.csv " に提供されています。
詳細については，「4.15　文書クラス定義ファイル」を参照してください。

-s 文書空間情報ファイル名

文書空間情報ファイルは，文書空間のアクセス制御情報や使用する RD エリア名を定義するファイルです。文書空間情報ファイル名を相対パスまたは絶対パスで指定します。このファイルのサンプルファイルは，"/opt/HiEDMS/sample/EDMdocinfo.txt" に提供されています。

詳細については，「4.16　文書空間情報ファイル」を参照してください。

-e { all | resource }

文書空間定義時の実行モードを指定します。
なお，このオプションを省略した場合，「all」が仮定されます。

all

データベース容量見積もりの基礎情報および文書空間を構築するための定義情報を出力するときに指定します。

resource

データベース容量見積もりの基礎情報だけを出力するときに指定します。

注意事項

- コマンド実行後に出力されるファイル名と同一名のファイルがある場合には，出力ファイル名 .bak の名称でバックアップファイルを作成します。このバックアップファイルは，出力ファイルをカスタマイズしたあとコマンドを再度実行した場合に，カスタマイズした内容を確認したいときなどに使用します。ただし，バックアップファイルをそのまま使用して文書空間構築コマンド（EDMCBuildDocSpace）を実行しないでください。このコマンド実行時にバックアップファイルがある場合は，バックアップファイルは上書きされます。
- このコマンドを実行するときには，DocumentSpace 構成定義ファイルに次のエントリを必ず指定してください。
  - SerialId エントリ
  - Process エントリ
  - DbType エントリ
  - PdHost エントリ
  - PdNamePort エントリ
  - PdUser エントリ
- このコマンドの実行後に出力される RD エリア定義情報ファイル中の RD エリア名は，文書クラス定義ファイルで指定したクラスやインデクスの数に応じて，文書空間情報ファイルで指定した RD エリア名＋通番の名称に変更されている場合があります。この RD エリア名を変更する場合には，RD エリア定義情報ファイルの内容をカスタマイズしてから，文書空間構築コマンド（EDMCBuildDocSpace）を実行してください。
- このコマンドの実行後に出力されるインデクス情報ファイルは，編集しないでください。インデクスの定義を変更する場合には，このコマンドの入力情報となる文書クラス定義ファイルを変更し，再度，このコマンドを実行します。
- アクセス制御機能を使う場合，セキュリティ定義ファイルやユーザ権限定義ファイルは，ユーザが編集してください。セキュリティ定義ファイルやユーザ権限定義ファイルの変更は，このコマンドおよび文書空間構築コマンド（EDMCBuildDocSpace）の実行には影響しません。
- 実行モード（-e オプション）に resource を指定して見積もり基礎情報ファイルを出力した場合は，-e オプションに all を指定して文書空間を構築するための定義情報を事前に作成していても，データベース容量見積もりの基礎情報と文書空間を構築するための定義情報が不整合とならないよう，再度，-e オプションに all を指定してこのコマンドを実行し，文書空間を構築するための定義情報を作成してください。

EDMChkMeta（メタ情報の記述内容の確認）

機能

メタ情報の定義を追加または変更する場合に，コマンドのオプションに指定するメタ情報ファイルの記述内容を確認します。確認した記述内容に誤りがある場合，標準エラー出力にメッセージを出力します。
チェックされる内容の詳細については，「付録 H　メタ情報の記述内容」を参照してください。

形式

```
EDMChkMeta  -f メタ情報ファイル名
          〔-c クラス名(,クラス名)...|-s セクション名(,セクション名)...〕
          〔-r { OBJECT | OBJCLASS | ALL }〕
```

オプション

-f

記述内容を確認したいオブジェクト定義を記述するメタ情報ファイルを指定します。このとき，メタ情報ファイル名は，コマンドを実行するディレクトリからの相対パス，絶対パス，またはカレントディレクトリで指定します。メタ情報ファイルについては，「4.6　メタ情報ファイル」を参照してください。

-c

確認するクラス名を指定します。
クラス名には，-f オプションに指定するメタ情報ファイルのオブジェクト定義を記述するセクションを指定します。つまり，メタ情報の追加コマンド（EDMAddMeta）の定義情報ファイルに指定するアクション名，［AddSubClass］セクションの dmaProp_DisplayName プロパティの値を指定します。
-c オプションは，オブジェクト定義，オブジェクト定義から参照するクラス定義，およびプロパティ定義を確認する場合に指定します。
-c オプションに指定するクラス名は，-f オプションに指定するメタ情報ファイルに存在する必要があります。

-s

確認するセクション名を指定します。
セクション名には，-f オプションに指定するメタ情報ファイルのクラス定義，またはプロパティ定義を記述するセクションを指定します。
-s オプションは，クラス定義，またはプロパティ定義を確認する場合に指定します。ただし，クラス定義，またはプロパティ定義から，オブジェクト定義については確認できません。したがって，クラス定義，またはプロパティ定義を変更し，-s オプションを指定してこのコマンドを実行した場合，オブジェクト定義は確認されません。クラス定義，またはプロパティ定義の変更とあわせてオブジェクト定義を確認するためには，-c オプションでこのコマンドを実行してください。
また，-s オプションに指定するセクション名は，-f オプションに指定するメタ情報に存在する必要があります。

-r { OBJECT | OBJCLASS | ALL }

メタ情報のチェック範囲を OBJECT，OBJCLASS，ALL の中から指定します。指定する値のチェック範囲と，指定する値が有効な場合について次に説明します。

OBJECT

-c オプション，および -s オプションで指定するクラス，およびセクションをチェックします。つまり，指定したクラスだけをチェックします。

オブジェクト定義を一部変更した場合に指定します。

OBJCLASS

OBJECT を指定した場合にチェックする範囲に加えて，-c オプションおよび -s オプションで指定するクラス，およびセクションをチェックするために参照するクラス定義のセクションについてもチェックします。
ユーザクラスの追加やクラス定義を変更した場合に指定します。

ALL

-c オプションおよび -s オプションで指定するクラス，およびセクションから参照が可能なすべてのセクションをチェックします。
システム提供のクラスおよびセクションも含めて，メタ情報全体をチェックする場合に指定します。

なお，このオプションを省略した場合，「ALL」が仮定されます。また，指定できない値を指定した場合，Usage が出力されます。

注意事項

● -f オプションに指定するメタ情報ファイルと同じディレクトリにすべてのメタ情報ファイルを格納する必要があります。

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

● 次のコマンドの実行中は，このコマンドを実行しないでください。
- EDMAddMeta コマンド
- EDMChkTbl コマンド
- EDMDelMeta コマンド
- EDMInitMeta コマンド
- EDMPrintMeta コマンド
- EDMRegEnvId コマンド
- EDMCDefDocSpace コマンド
- EDMCBuildDocSpace コマンド

● edmsys.ini，edmsysclass.ini，edmsysprop.ini，edmnmclass.ini，および edmnmprop.ini のメタ情報ファイルについては，このコマンドを実行した場合にエラーが発生しても定義に問題はありませんので，そのまま利用してください。これらのファイルは内部処理で使用するため，オブジェクト定義を指定するメタ情報ファイルとはフォーマットが異なります。

● -r オプションに OBJECT を指定する場合，あらかじめクラス定義に誤りがないかどうかをチェックしておく必要があります。-r オプションに OBJECT を指定した場合，-c オプションおよび -s オプションで指定したクラスおよびセクションについてはチェックしますが，クラス定義についてはチェックしません。しかし，オブジェクト定義をチェックするときはクラス定義を参照するため，クラス定義に誤りがあると -c オプションおよび -s オプションで指定したクラスおよびセクションを正しくチェックできません。

● -c オプションおよび -s オプションを指定しない場合，-f オプションで指定したメタ情報ファイルのすべてのセクションをチェックします。

● このコマンドの実行中に，メッセージ「Ignore: An error occurred in initialization of a trace object.」が標準エラー出力に出力される場合があります。このメッセージが出力された場合，コマンドの処理は続行されますが，トレースファイルは出力されません。このメッセージが出力された場合，メモリ不足が原因と考えられるため，ほかの実行中のアプリケーションを終了させるか，または実メモリを増設してから，このコマンドを再度実行してください。

● このコマンドの実行中に，メッセージ「Ignore: An error occurred in initialization of a message object.」が標準エラー出力に出力される場合があります。このメッセージが出力された場合，コマンドの処理は続行されますが，メッセージが正しく表示されないことがあります。このメッセージが出力された場合，メモリ不足が原因と考えられるため，ほかの実行中のアプリケーションを終了させるか，または実メモリを増設してから，このコマンドを再度実行してください。

EDMChkTbl（データベースの表・列の確認）

機能

定義されているクラス・プロパティとデータベースに作成している表・列の定義が一致しているかどうかを確認します。一致していない場合，標準エラー出力にメッセージを出力します。

なお，データベースに格納されたオブジェクト（永続オブジェクト）を格納するデータベースは，DocumentSpace 構成定義ファイルの DbType エントリの値を参照します。この値に「HIRDB」が指定されているかどうかを確認して実行してください。

形式

```
EDMChkTbl 〔-p〕 -f メタ情報ファイル名 〔-c クラス名(,クラス名)...〕
```

オプション

-p

データベースに格納しているメタ情報を基にクラス・プロパティの定義と，データベースに作成している表・列の定義が一致しているかどうかを確認する場合に指定します。このオプションを指定する場合，-f オプションにはオブジェクト定義を記述しているメタ情報ファイル名だけを指定します。このオプションを省略する場合，-f オプションに指定するメタ情報ファイルとデータベースに作成している表・列の定義が一致しているかどうかを確認します。

-f

オブジェクト定義が記述されているメタ情報ファイルを指定します。メタ情報ファイル名は，-p オプションを省略する場合，コマンドを実行するディレクトリからの相対パス，絶対パス，またはカレントディレクトリで指定します。すべてのメタ情報ファイルは，-f オプションに指定するオブジェクト定義を記述するメタ情報ファイルと同じディレクトリに格納されている必要があります。
なお，-p オプションを指定する場合，オブジェクト定義を記述しているメタ情報ファイル名だけを指定します。メタ情報ファイルについては，「4.6　メタ情報ファイル」を参照してください。

-c

クラスおよびプロパティの定義が，データベースの表および列の定義と一致しているかどうかを確認するクラス名を指定します。つまり，メタ情報の追加コマンド（EDMAddMeta）の定義情報ファイルに指定するアクション名，[AddSubClass] セクションの dmaProp_DisplayName プロパティの値を指定します。
-f オプションに指定するメタ情報ファイルには，このオプションで指定するクラスの記述が必要です。
このオプションを省略する場合，永続オブジェクトを格納するすべてのクラスを対象に，クラスおよびプロパティの定義と，データベースの表および列が一致しているかどうかを確認します。
このオプションは，幾つかのクラスを指定して，指定するクラスの定義が表および列の定義と一致しているかどうかを確認する場合に使用します。

注意事項

● このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数

「DOCBROKERDIR」に指定してください。

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

● このコマンドを実行する前に，データベースを起動しておいてください。

● このコマンドを実行する前に，メタ情報の登録とデータベースの表の作成を完了しておく必要があります。

● 次のコマンドの実行中は，このコマンドを実行しないでください。
- EDMAddMeta コマンド
- EDMChkMeta コマンド
- EDMDelMeta コマンド
- EDMInitMeta コマンド
- EDMPrintMeta コマンド
- EDMRegEnvId コマンド
- EDMCDefDocSpace コマンド
- EDMCBuildDocSpace コマンド

● このコマンドの実行中に，メッセージ「Ignore: An error occurred in initialization of a trace object.」が標準エラー出力に出力される場合があります。このメッセージが出力された場合，コマンドの処理は続行されますが，トレースファイルは出力されません。このメッセージが出力された場合，メモリ不足が原因と考えられるため，ほかの実行中のアプリケーションを終了させるか，または実メモリを増設してから，このコマンドを再度実行してください。

● このコマンドの実行中に，メッセージ「Ignore: An error occurred in initialization of a message object.」が標準エラー出力に出力される場合があります。このメッセージが出力された場合，コマンドの処理は続行されますが，メッセージが正しく表示されないことがあります。このメッセージが出力された場合，メモリ不足が原因と考えられるため，ほかの実行中のアプリケーションを終了させるか，または実メモリを増設してから，このコマンドを再度実行してください。

EDMCreateIds（ユーザ定義識別子ファイルの作成）

機能

DocumentBroker の動作環境メタ情報ファイルに定義されているユーザ定義のクラス，およびプロパティにそれぞれ対応する識別子を，指定するファイルに出力します。なお，このコマンドを実行した場合の出力例については，「4.11　ユーザ定義識別子ファイルおよびユーザ定義識別子変数ファイル」を参照してください。

形式

```
EDMCreateIds  -i ユーザ定義識別子ファイル名
             〔-v ユーザ定義識別子変数ファイル名
              -c { A | B | T }〕
             〔-p プリフィクス 〕
```

オプション

-i ユーザ定義識別子ファイル名

ユーザ定義のクラス，およびプロパティのそれぞれに対応するユーザ定義識別子を出力する，ユーザ定義識別子ファイルを指定します。ユーザ定義識別子ファイル名を，相対パスまたは絶対パスで指定します。ユーザ定義識別子ファイル名は，英小文字，英大文字，数字，「.」（ピリオド），「_」（下線文字），および「-」（マイナス記号）で指定します。

ユーザ定義のクラスのクラス名は，動作環境メタ情報ファイルでクラス定義が記述されている，ClassDescription の dmaProp_DisplayName エントリの値です。ユーザ定義のプロパティのプロパティ名は，動作環境メタ情報ファイルでプロパティ定義が記述されている，PropertyDescription の dmaProp_DisplayName の値です。これらのユーザ定義のクラスおよびプロパティのユーザ定義識別子は，dmaProp_Ids プロパティのリストの要素として指定するエントリであるユーザ定義識別子の値となります。
また，このユーザ定義識別子ファイルには，ヘッダの二重インクルードを防止するマクロをファイルの先頭と末尾に出力します。マクロ名は，「'_'+ ユーザ定義識別子ファイル名」です。ただし，英小文字は英大文字に，「.」（ピリオド）および「-」（マイナス記号）は「_」（下線文字）に，変換されます。

-v ユーザ定義識別子変数ファイル名

ユーザ定義のクラス，およびプロパティに対応するユーザ定義識別子を設定した変数の宣言を出力する，ユーザ定義識別子変数ファイルを指定します。ユーザ定義識別子変数ファイル名を，相対パスまたは絶対パスで指定します。ユーザ定義識別子変数ファイル名は，英小文字，英大文字，数字，「.」（ピリオド），「_」（下線文字），および「-」（マイナス記号）で指定します。なお，ユーザ定義識別子変数ファイル名として，-i オプションに指定するユーザ定義識別子ファイルと同じファイルを指定した場合，エラーになります。
また，このユーザ定義識別子変数ファイルには，ヘッダの二重インクルードを防止するマクロをファイルの先頭と末尾に出力します。マクロ名は，「'_'+ ユーザ定義識別子変数ファイル名」です。ただし，英小文字は英大文字に，「.」（ピリオド）および「-」（マイナス記号）は「_」（下線文字）に，変換されます。
出力されたユーザ定義識別子変数ファイルをインクルードする場合，EDMAPP_INIT_ID を定義するかどうかによって，変数を宣言だけするか，変数の宣言と定義をするかが異なります。複数のファイルでユーザ定義識別子変数ファイルをインクルードする場合，変数の重複した定義を避けるため，EDMAPP_INIT_ID を定義するファイルは一つだけにしてください。

EDMAPP_INIT_ID を定義する場合の記述形式

```
#define  EDMAPP_INIT_ID
#include  "ユーザ定義識別子ファイル名"
```

変数が，宣言・定義されます。

EDMAPP_INIT_ID を定義しない場合の記述形式

```
#include  "ユーザ定義識別子ファイル名"
```

変数が宣言されます。

-c { A | B | T }

このオプションは，-v オプションを指定した場合，有効になります。-v オプションを指定した場合は，このオプションを必ず指定してください。
コマンドを実行して作成するファイルで，文字コードセットに対応してインクルードするヘッダファイルの種別を指定します。インクルードするヘッダファイルは，二重インクルードを防止するマクロ定義とともに出力します。このオプションに指定する値が対応する文字コードセットについて，次に説明します。

A

char 対応ヘッダファイルをインクルードします。

B

wchar_t 対応ヘッダファイルをインクルードします。

T

char および wchar_t 対応ヘッダファイルをインクルードします。

インクルードするファイルと二重インクルードを防止するマクロの対応関係を次の表に示します。

表 7-6 インクルードファイル名と二重インクルードを防止するマクロの対応関係

-c オプションの引数	インクルードファイル名	二重インクルードを防止するマクロ定義
A	dmatypesA.h	DMATYPESA_H
	dmacom.h	DMACOM_H
	dmaifaceA.h	DMAIFACEA_H
	dmaids.h	DMAIDS_H
B	dmatypes.h	DMATYPES_H
	dmacom.h	DMACOM_H
	dmaiface.h	DMAIFACE_H
	dmaids.h	DMAIDS_H
T	dmatypesT.h	DMATYPEST_H
	dmacom.h	DMACOM_H
	dmaifaceT.h	DMAIFACET_H
	dmaids.h	DMAIDS_H

-p プリフィクス

コマンドの実行によって作成したファイル中のマクロ名にプリフィクスを付ける場合，プリフィクス名を指定します。プリフィクス名は，英小文字，英大文字，数字，および「_」（下線文字）で指定します。ただし，先頭の文字は，英小文字または英大文字を指定してください。

注意事項

- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは，DocumentBroker サーバの停止中に実行してください。

EDMCrtSimMeta（クラス定義情報ファイルの作成）

機能

メタ情報ファイルから，Java クラスライブラリや DocumentBroker オブジェクト操作ツールなどで使用するクラス定義情報ファイルを作成します。クラス定義情報ファイルは，接続する文書空間識別子にサフィックス「.ini」を付加したファイル名で，オプションで指定したディレクトリに出力されます。なお，クラス定義情報ファイルについては，「4.10　クラス定義情報ファイル」を参照してください。

形式

```
EDMCrtSimMeta　〔-l 出力先ディレクトリ名〕
```

オプション

-l 出力先ディレクトリ名

クラス定義情報ファイルを出力するディレクトリを，相対パスまたは絶対パスで指定します。このオプションを省略した場合，クラス定義情報ファイルを，" 実行環境ディレクトリ /etc/meta_files" に出力します。

注意事項

● このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

EDMCrtSql（DocumentBroker 用データベース定義文の作成）

機能

データベースに登録されているメタ情報を基に，DocumentBroker のデータベースを初期設定するために必要なデータベース定義文を作成します。また，オプションの指定によって，定義系 SQL「CREATE INDEX」文を作成します。

形式

```
EDMCrtSql  -o データベース定義文格納ファイル名
          〔-t HIRDB〕
          〔-c〕
          〔-i インデクス情報ファイル名〕
          〔-j インデクスのコメント情報出力ファイル名〕
          〔-r RDエリア定義情報ファイル名〕
          〔-m〕
          〔-u { ALL | PARTIAL | NO}〕
          〔-n〕
```

オプション

-o データベース定義文格納ファイル名

作成したデータベース定義文を格納するファイルのパス名を指定します。

-t HIRDB

永続オブジェクトを格納するデータベースを指定します。使用できるデータベースシステムは HiRDB です。したがって，フラグ引数として「HIRDB」を指定してください。

-c

表名，列名に対して注釈を付ける SQL 文を出力する場合に指定します。注釈となる文字列は，メタ情報ファイルの ClassDescription セクションおよび PropertyDescription セクションの「dmaProp_DescriptiveText」に指定されている値（文字列）です。なお，この値が指定されていないクラスに対応する表およびプロパティに対応する列には，注釈を付ける SQL 文は出力されません。

-i インデクス情報ファイル名

インデクス定義をするための情報を記述したファイルのパス名を指定します。インデクス情報ファイルについては，「4.9　インデクス情報ファイル」を参照してください。

-j インデクスのコメント情報出力ファイル名

インデクスのコメントを出力するファイル名を指定します。このファイルには，インデクス定義文，クラス名およびプロパティ名が出力されます。

-r RD エリア定義情報ファイル名

RD エリア定義情報ファイル名を相対パスまたは絶対パスで指定します。

-o オプションで指定したデータベース定義文格納ファイル中の RD エリア名を，指定した RD エリア定義情報ファイルの情報に従って出力する場合に指定します。RD エリア定義情報ファイルについて

は，「4.8　RD エリア定義情報ファイル」を参照してください。

-m

複数の異なる文書オブジェクトを，同時に一つの文書オブジェクトに対して関連づける場合に指定します。この場合，次に示すクラスのプロパティに対して複数列インデクスを定義します。

edmClass_Relationship **クラス**

dmaProp_Head プロパティ

なお，このオプションの指定を省略した場合，それぞれのプロパティに対して単一列インデクスが定義されます。

-u { ALL | PARTIAL | NO }

コンテント格納用 RD エリア，および SGMLTEXT データ格納用 RD エリアに対して，データベースの更新ログ取得方式を指定します。

ALL

ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。ログ取得モードで運用すると，ロールバックおよびロールフォワードに必要なデータベースの更新ログを取得します。

PARTIAL

更新前ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。更新前ログ取得モードで運用すると，ロールバックに必要なデータベースの更新ログを取得します。

NO

ログレスモードでユーザ LOB 用 RD エリアを運用する場合に指定します。ログレスモードで運用すると，データベースの更新ログを取得しません。

なお，このオプションを省略した場合，「ALL」が仮定されます。

指定するデータベースの更新ログ取得方式によって，障害が発生した場合のユーザ LOB 用 RD エリアの回復方法は異なります。また，必要となるログ容量も異なります。障害が発生した場合のユーザ LOB 用 RD エリアの回復方法については，マニュアル「HiRDB　システム運用ガイド」を参照してください。また，ログ容量については，マニュアル「HiRDB　システム導入・設計ガイド」を参照してください。

-n

OIID を格納するプロパティに対応する列に非ナル値制約を与える場合に指定します。非ナル値制約とは，指定した列の値に，ナル値を許さない制約のことです。ただし，次の OIID を格納するプロパティに対応する列には，非ナル値制約を与えません。

- edmClass_OIID クラスの OIID を格納するプロパティに対応する列
- 複数のクラスを一つの表に格納するクラスの場合の，二つ目以降のクラスの OIID を格納するプロパティに対応する列
- HiRDB の繰り返し列に定義される OIID を格納するプロパティに対応する列

注意事項

● このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

● このコマンドを実行する前に，データベースを起動しておいてください。

● このコマンドを実行する前には，EDMInitMeta コマンドでメタ情報をデータベースに登録しておく必要があります。

- このコマンドは，データベース自身の定義であるエリア定義は実行しません。必要なエリアの定義は，出力されるデータベース定義文に追加してください。
- 次に示す場合は，インデクス格納 RD エリアの名称として「USR_INDEX_AREA」が出力されます。
  - -i オプションにインデクス情報ファイル名を指定し，-r オプションを省略してこのコマンドを実行した場合
  - RD エリア定義情報ファイルの［IndexArea］セクションに，インデクス情報ファイルに記述した項目に対応する RD エリアの定義を記述していない状態で，-i オプションにインデクス情報ファイル名を指定し，-r オプションに RD エリア定義情報ファイル名を指定してこのコマンドを実行した場合
- 実行結果であるデータベース定義文に対してエリア定義などの必要な項目を追加，修正してからデータベースを作成してください。ただし，-r オプションを指定した場合は，RD エリア定義情報ファイルの指定に従って，RD エリア名が出力されます。
- マルチファイル管理機能を使用する環境の場合，edmClass_ContentTransfers クラスに対応するデータベース定義文が出力されます。
- リファレンスファイル管理機能を使用する環境の場合，edmClass_ContentReference クラスに対応するデータベース定義文が出力されます。
- File Link 連携機能を使用する環境の場合，edmClass_ContentFileLink クラスに対応するデータベース定義文が出力されます。
- HiRDB のオンライン再編成機能を使用する場合，-n オプションを指定してください。なお，メタ情報の表および OIID を格納する表は，オンライン再編成機能の対象外です。オンライン再編成機能の対象外の表は，オンライン再編成機能の対象となる表と RD エリアを分けてください。オンライン再編成機能の詳細については，マニュアル「HiRDB Staticizer Option」を参照してください。

EDMDelMeta（メタ情報の削除）

機能

EDMInitMeta コマンドで登録したメタ情報をデータベース（HiRDB）から削除します。メタ情報を削除した場合（サブクラスやプロパティの削除）は，自動的に動作環境メタ情報ファイルも更新されます。

形式

```
EDMDelMeta  {-r サブクラス名
             -s スーパークラス名
             -o データベース定義文格納ファイル |
             -p プロパティ名
             -q クラス名
             -o データベース定義文格納ファイル |
             -t 〔-e〕〔-b〕}
```

オプション

-r サブクラス名

サブクラスを削除する場合，サブクラス名を指定します。

-s スーパークラス名

-r オプションに指定したサブクラスのスーパークラス名を指定します。

-o データベース定義文格納ファイル

作成したデータベース定義文を格納するファイル名を相対パスまたは絶対パスで指定します。

-p **プロパティ名**

プロパティを削除する場合，プロパティ名を指定します。

-q **クラス名**

-p オプションに指定したプロパティを持つクラスのクラス名を指定します。

-t

メタ情報の表をすべて削除する場合に指定します。

-e

複数の実行環境から同一の文書空間にアクセスしている場合，それらの実行環境の情報をすべて削除するときに指定します。このオプションを指定できるのは，実行環境識別子が 0 の実行環境だけです。

このオプションを指定すると，登録されている実行環境の情報が削除されるため，複数の実行環境から同一の文書空間にアクセスする実行環境を新規に登録し直す必要があります。

-b

システム導入支援機能によって文書空間を構築した場合の構築情報をすべて削除するときに指定します。

＜削除する情報とオプションの組み合わせ＞

ユーザが作成したサブクラスを削除する

ユーザが作成したサブクラスを削除するには，-r オプション，-s オプションおよび -o オプションを指定します。サブクラスを削除した場合には，自動的に動作環境メタ情報ファイルも更新されます。

ユーザが追加したプロパティを削除する

ユーザが追加したプロパティを削除するには，-p オプション，-q オプションおよび -o オプションを指定します。ただし，全文検索用のプロパティは削除できません。また，プロパティを削除した場合には，自動的に動作環境メタ情報ファイルも更新されます。

文書空間単位でメタ情報を削除する

文書空間単位でメタ情報を削除するには，-t オプションを指定します。文書空間単位のメタ情報の削除によって，HiRDB の表などを削除する場合は，該当する文書空間用のスキーマを削除してください。

注意事項

● このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

● このコマンドを実行する前に，データベースを起動しておいてください。

● このコマンドを実行する前に，次に示す条件に従ってオブジェクトを削除してください。

- クラスを削除する場合
  削除するクラスを基に作成されたオブジェクトを削除します。
- プロパティを削除する場合
  プロパティを削除するクラスを基に作成されたオブジェクトを削除します。

● このコマンド実行時，edmClass_Struct クラスのサブクラスを削除する場合，またはほかのサブクラスに定義されているプロパティと同じ VariableArray 型のプロパティを削除する場合，データベース定義文格納ファイルは作成されません。

EDMGetRas（サーバ側の障害情報の取得）

機能

DocumentBroker で障害が発生した場合に，障害調査に必要な情報を取得します。なお，このコマンドで取得できるのは，サーバ側の情報です。

このコマンドを実行して取得できる情報（ファイル）は，次のとおりです。

- $DOCBROKERDIR/spool 下の全ファイル
- $DOCBROKERDIR/etc 下の全ファイル
- $DOCBROKERDIR/tmp 下の全ファイル
- $_HIEDMS_TRACE_DIR 下の全ファイル
- 詳細エラーログファイル
- OS 情報
- 環境情報
- TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）
- getrascustom.ini の [Path] セクションで指定されたファイル
- DocumentBroker Server のバージョン情報

それぞれの情報の詳細については，「6.4.8　サーバ側で発生した障害情報の取得」を参照してください。

形式

```
EDMGetRas  -d 出力先ディレクトリ名
          〔-l 収集種別 〕
          〔-s 開始時刻 〕
```

オプション

-d 出力先ディレクトリ名

障害情報の出力先ディレクトリ名をフルパスで指定します。このコマンドで収集した障害情報を，指定したディレクトリに出力します。
出力先ディレクトリの名称は任意です。このコマンドを実行するユーザが，作成するディレクトリに対して読み取り，書き込み，および検索を実行できるように権限を設定してください。
出力先ディレクトリとして，収集対象となるディレクトリの下位のディレクトリを指定しないでください。

-l 収集種別

収集する障害情報の種別を 1 ～ 3 で指定します。指定が省略された場合には，収集種別 1 で動作します。
収集種別ごとに取得できる情報の種類について，次の表に示します。なお，それぞれの情報の詳細については，「6.4.8　サーバ側で発生した障害情報の取得」を参照してください。

表 7-7 EDMGetRas コマンドで収集種別ごとに取得できる情報の種類

収集種別	取得できる情報
1	$DOCBROKERDIR/spool 下の全ファイル $DOCBROKERDIR/etc 下の全ファイル $DOCBROKERDIR/tmp 下の全ファイル $_HIEDMS_TRACE_DIR 下の全ファイル※1 詳細エラーログファイル※1 OS 情報 環境情報 TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）※1 getrascustom.ini※2 の [Path] セクションで指定されたファイル DocumentBroker Server のバージョン情報
2	$DOCBROKERDIR/spool 下の全ファイル $_HIEDMS_TRACE_DIR 下の全ファイル※1 詳細エラーログファイル※1 TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）※1 getrascustom.ini※2 の [Path] セクションで指定されたファイル DocumentBroker Server のバージョン情報
3	$DOCBROKERDIR/spool 下の全ファイル $_HIEDMS_TRACE_DIR 下の全ファイル※1 詳細エラーログファイル※1 getrascustom.ini※2 の [Path] セクションで指定されたファイル DocumentBroker Server のバージョン情報

注※ 1

環境変数の設定によっては取得できません。詳細については，「6.4.8　サーバ側で発生した障害情報の取得」を参照してください。

注※ 2

詳細については，「6.4.8　サーバ側で発生した障害情報の取得」を参照してください。

-s 開始時刻

トレースファイルを収集する場合に，指定した時刻からコマンドを実行した時刻までに更新されたファイルだけを収集するときに指定します。
このオプションが有効になるのは，取得する障害情報のうち，次のファイルです。

- $DOCBROKERDIR/spool 下の全ファイル
- $_HIEDMS_TRACE_DIR 下の全ファイル
- TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）
- 詳細エラーログファイル

指定が省略された場合，更新時刻に関係なく，すべてのファイルが収集対象になります。
開始時刻は，次の書式で指定してください。

開始時刻の書式

```
'YYYY/MM/DD[△hh:mm]'
```

開始時刻は「'」（シングルクォーテーション）で囲んで指定してください。
△には，半角の空白を指定します。
YYYY には，西暦年号を指定します。0001 ～ 9999 の範囲の数字を指定してください。
MM には，月を指定します。01 ～ 12 の範囲の数字を指定してください。
DD には，日を指定します。01 ～ 31 の範囲の数字を指定してください。
hh には，時を指定します。00 ～ 23 の範囲の数字を指定してください。

mm には，分を指定します。00 ～ 59 の範囲の数字を指定してください。

開始時刻の指定例

指定例を次に示します。

例1（省略なし）：-s '2000/12/24 10:30'
2000年12月24日10:30AMからコマンドを実行した時刻までに更新されたファイルが対象になります。
例2（[△hh:mm]を省略）：-s '2000/12/24'
2000年12月24日00:00AMからコマンドを実行した時刻までに更新されたファイルが対象になります。

注意事項

- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは，DocumentBroker サーバの停止中または起動中どちらでも実行できます。
- 収集対象となるディレクトリ（$DOCBROKERDIR/spool など）が存在しなかったり，アクセス権（コマンド実行者に対する，読み出し，実行権限）がなかった場合には，その処理はスキップされ，以降の処理が続行されます。getrascustom.ini の [Path] セクションに指定した場合も同様です。

EDMGetRasCL（クライアント側の障害情報の取得）

機能

DocumentBroker で障害が発生した場合に，障害調査に必要な情報を取得します。なお，このコマンドで取得できるのは，クライアント側の情報です。なお，Java クラスライブラリが出力するトレースファイルは取得の対象ではありません。

このコマンドを実行して取得できる情報（ファイル）は，次のとおりです。

- $_HIEDMS_FTPDIR/spool 下の全ファイル
- $_HIEDMS_FTPDIR/etc 下の全ファイル
- $_HIEDMS_FTPDIR/tmp 下の全ファイル
- クライアントおよびファイル転送サービスのトレースファイル
- 詳細エラーログファイル
- OS 情報
- 環境情報
- TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）
- getrascustom.ini の [Path] セクションで指定されたファイル
- DocumentBroker Development Kit および DocumentBroker Runtime のバージョン情報

それぞれの情報の詳細については，「6.4.9　クライアント側で発生した障害情報の取得」を参照してください。

形式

```
EDMGetRasCL -d 出力先ディレクトリ名
           〔-l 収集種別 〕
           〔-s 開始時刻 〕
```

オプション

-d 出力先ディレクトリ名

障害情報の出力先ディレクトリ名をフルパスで指定します。このコマンドで収集した障害情報を，指

定したディレクトリに出力します。
出力先ディレクトリの名称は任意です。このコマンドを実行するユーザが，作成するディレクトリに対して読み取り，書き込み，および検索を実行できるように権限を設定してください。
出力先ディレクトリとして，収集対象となるディレクトリの下位のディレクトリを指定しないでください。

-l 収集種別

収集する障害情報の種別を 1 ～ 3 で指定します。指定が省略された場合には，収集種別 1 で動作します。
収集種別ごとに取得できる情報の種類について，次の表に示します。なお，それぞれの情報の詳細については，「6.4.9　クライアント側で発生した障害情報の取得」を参照してください。

表 7-8　EDMGetRasCL コマンドで収集種別ごとに取得できる情報の種類

収集種別	取得できる情報
1	$_HIEDMS_FTPDIR/spool 下の全ファイル※1 $_HIEDMS_FTPDIR/etc 下の全ファイル※1 $_HIEDMS_FTPDIR/tmp 下の全ファイル※1 クライアントおよびファイル転送サービスのトレースファイル 詳細エラーログファイル※1 OS 情報 環境情報 TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）※1 getrascustom.ini※2 の [Path] セクションで指定されたファイル DocumentBroker Development Kit および DocumentBroker Runtime のバージョン情報
2	$_HIEDMS_FTPDIR/spool 下の全ファイル※1 クライアントおよびファイル転送サービスのトレースファイル 詳細エラーログファイル※1 TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）※1 getrascustom.ini※2 の [Path] セクションで指定されたファイル DocumentBroker Development Kit および DocumentBroker Runtime のバージョン情報
3	$_HIEDMS_FTPDIR/spool 下の全ファイル※1 クライアントおよびファイル転送サービスのトレースファイル 詳細エラーログファイル※1 getrascustom.ini※2 の [Path] セクションで指定されたファイル DocumentBroker Development Kit および DocumentBroker Runtime のバージョン情報

注※ 1

環境変数の設定によっては取得できません。詳細については，「6.4.9　クライアント側で発生した障害情報の取得」を参照してください。

注※ 2

詳細については，「6.4.9　クライアント側で発生した障害情報の取得」を参照してください。

-s 開始時刻

トレースファイルを収集する場合に，指定した時刻からコマンドを実行した時刻までに更新されたファイルだけを収集するときに指定します。
このオプションが有効になるのは，取得する障害情報のうち，次のファイルです。指定が省略された場合，更新時刻に関係なく，すべてのファイルが収集対象になります。

- $_HIEDMS_FTPDIR/spool 下の全ファイル
- クライアントおよびファイル転送サービスのトレースファイル
- 詳細エラーログファイル

- TPBroker のトレース（$VBROKER_ADM/../log 下の全ファイル）

開始時刻は，次の書式で指定してください。

開始時刻の書式

'YYYY/MM/DD[△hh:mm]'

開始時刻は「'」（シングルクォーテーション）で囲んで指定してください。
△には，半角の空白を指定します。
YYYY には，西暦年号を指定します。0001 ～ 9999 の範囲の数字を指定してください。
MM には，月を指定します。01 ～ 12 の範囲の数字を指定してください。
DD には，日を指定します。01 ～ 31 の範囲の数字を指定してください。
hh には，時を指定します。00 ～ 23 の範囲の数字を指定してください。
mm には，分を指定します。00 ～ 59 の範囲の数字を指定してください。

開始時刻の指定例

指定例を次に示します。
例1（省略なし）：-s '2000/12/24 10:30'
2000年12月24日10:30AMからコマンドを実行した時刻までに更新されたファイルが対象になります。
例2（[△hh:mm]を省略）：-s '2000/12/24'
2000年12月24日00:00AMからコマンドを実行した時刻までに更新されたファイルが対象になります。

注意事項

● 収集対象となるディレクトリ（$_HIEDMS_FTPDIR/spool など）が存在しなかったり，アクセス権（コマンド実行者に対する，読み出し，実行権限）がなかったりした場合には，その処理はスキップされ，以降の処理が続行されます。getrascustom.ini の [Path] セクションに指定した場合も同様です。

● このコマンドは，クライアントアプリケーションを起動するユーザが実行してください。

EDMInitMeta（メタ情報の初期設定）

機能

DocumentBroker のメタ情報を表にして，DocumentSpace 構成定義ファイルで指定されているデータベース（HiRDB）に登録します。また，動作環境メタ情報ファイルを，" 実行環境ディレクトリ /etc/meta_files" の下に作成します。

なお，マルチファイル管理機能，リファレンスファイル管理機能や File Link 連携機能を使用するかどうかについても，このコマンドで設定します。

形式

```
EDMInitMeta   〔-d〔-e〕〔-b〕〕
               -f メタ情報ファイル名
              〔-r ユーザ表用RDエリア名〕
              〔-i ユーザインデクス用RDエリア名〕
              〔-v VariableArrayの型プロパティの要素の格納先〕
              〔-A〕
              〔-R〕
              〔-L〕
              〔-u { DisplayName | ID | DBAlias}〕
              〔-c 一つの文書に格納するファイルの最大数 〕
              〔-n ユーザ識別子の最大長〕
              〔-g グループ識別子の最大長〕
              〔-C { SJIS | UTF-8 }〕
```

オプション

-d

メタ情報をデータベースに再登録する場合に指定します。このオプションを指定すると，既存のメタ情報をデータベースから削除して，新規にメタ情報を登録します。

-e

複数の実行環境から同一の文書空間にアクセスしている場合，それらの実行環境の情報をすべて削除するときに指定します。このオプションを指定できるのは，実行環境識別子が 0 の実行環境だけです。
このオプションを指定すると，登録されている実行環境の情報が削除されるため，複数の実行環境から同一の文書空間にアクセスする実行環境を新規に登録し直す必要があります。

-b

システム導入支援機能によって文書空間を構築した場合の構築情報をすべて削除するときに指定します。
このオプションを指定した場合，文書空間の構築情報を削除するため，システム導入支援機能で作成した環境ではなくなります。そのため，今後，システム導入支援機能は使用できなくなります。

-f メタ情報ファイル名

メタ情報ファイル名を相対パスまたは絶対パスで指定します。文書空間に作成するオブジェクトの定義ファイルを指定します。
メタ情報を新規登録する場合，メタ情報ファイルとして，DocumentBroker で提供されているメタ情報ファイル（実行環境ディレクトリ /etc/edms.ini）を指定してください。
メタ情報を再登録する場合，メタ情報ファイルとして，動作環境メタ情報ファイル（実行環境ディレクトリ /etc/meta_files/edms.ini）を指定してください。
なお，指定するメタ情報ファイルから参照しているメタ情報ファイルも同じディレクトリに格納してください。メタ情報ファイルについては，「4.6　メタ情報ファイル」を参照してください。

-r ユーザ表用 RD エリア名

メタ情報の表の行を格納するユーザ用 RD エリア名を指定します。省略した場合，メタ情報の表は，定義されている表数がいちばん少ないユーザ用 RD エリアに格納されます。

-i ユーザインデクス用 RD エリア名

メタ情報の表に定義されるインデクスを格納するユーザ用 RD エリア名を指定します。省略した場合，インデクスはメタ情報の表が格納されている RD エリアに格納されます。

-v VariableArray 型のプロパティの要素の格納先

基本単位が VariableArray 型であるプロパティを使用する場合，そのプロパティの要素の格納先として，次のどれかの文字列を指定します。

Own

VariableArray 型のプロパティが定義されるクラスに該当する表とは別の表へ格納する場合に指定します。

HIRDB

VariableArray 型のプロパティが定義されるクラスに該当する表の HiRDB の繰り返し列へ格納する場合に指定します。HiRDB の繰り返し列の最大要素数は，VariableArray 型のプロパティを定義する場合の指定に依存します。

Both

Own および HIRDB の両方を使用する場合に指定します。VariableArray 型のプロパティを定義する場合の指定によって，別の表か，HiRDB の繰り返し列かを設定します。

オプションを省略した場合，Own を仮定します。ただし，-f オプションに動作環境メタ情報ファイルを指定した場合，そのメタ情報に定義されている格納先に従います。

また，既存の VariableArray 型のプロパティの格納先から，ほかの格納先への変更が可能なのは，Own から Both への変更，および HiRDB から Both への変更だけです。格納先だけを変更する場合，次に示す形式でコマンドを実行します。

```
EDMInitMeta -d -f $DOCBROKERDIR/etc/meta_files/edms.ini 〔-r ユーザ表用RDエリア名〕〔-i ユーザインデクス用RDエリア名〕 -v Both
```

-A

アクセス制御機能を使用する場合に指定します。このオプションを省略した場合，アクセス制御機能は使用できません。

-R

リファレンスファイル管理機能を使用する場合に指定します。このオプションを省略した場合，リファレンスファイル管理機能は使用できません。

なお，メタ情報を再登録する場合には，リファレンスファイル管理機能を使用するかどうかの設定を変更できません。また，-f オプションに動作環境メタ情報ファイル（実行環境ディレクトリ /etc/meta_files/edms.ini）を指定してメタ情報を再登録する場合，動作環境メタ情報ファイルのリファレンスファイル管理機能を使用するかどうかの設定に合わせて，このオプションを指定する必要があります。

-L

File Link 連携機能を使用する場合に指定します。このオプションを省略した場合，File Link 連携機能は使用できません。

なお，メタ情報を再登録する場合には，File Link 連携機能を使用するかどうかの設定を変更できません。

また，Linux の場合および TPBroker V5 を使用している場合，このオプションを指定できません。

-u { DisplayName | ID | DBAlias}

データベース定義の名称定義の方法を指定します。

DisplayName

データベース定義の名称定義は，クラス名，プロパティ名などとします。

ID

データベース定義の名称定義は，GUID 値を変換した ID とします。

DBAlias

データベースの名称定義は，クラス名やプロパティ名などの別名とします。-f オプションに DBAlias を使用するように移行した環境の動作環境メタ情報ファイルを指定する場合だけ指定してください。

このオプションの指定を省略した場合，「DisplayName」が仮定されます。

なお，このコマンドでは，データベース定義の名称定義の方法を変更できません。このため，-f オプションに動作環境メタ情報ファイルを指定する場合，そのメタ情報ファイル中のデータベース定義の名称定義の方法に合わせてこのオプションを指定する必要があります。

なお，-u オプションで「DisplayName」を指定した場合のデータベース定義の名称定義については，「付録 F　システムクラスおよびシステムプロパティの名称定義の規則」を参照してください。

-c 一つの文書に格納するファイルの最大数

マルチファイル管理機能を使用する場合に指定します。一つの文書に含まれるコンテントとして格納

するファイルの最大数を指定します。
2 ～ 4,096 の範囲の 10 進数値で指定してください。
このオプションを省略した場合，マルチファイル管理機能は使用できません。
なお，メタ情報を再登録する場合には，マルチファイル管理機能を使用するかどうかの設定を変更できません。また，一つの文書のコンテントとして格納するファイルの最大数も変更できません。-f オプションに動作環境メタ情報ファイル（実行環境ディレクトリ /etc/meta_files/edms.ini）を指定してメタ情報を再登録する場合，-c オプションは次のように扱われます。

マルチファイル管理機能を使用する環境の場合

-c オプションの指定は無視されます。一つの文書のコンテントに格納するファイルの最大数は，動作環境メタ情報ファイルに定義されている値になります。

マルチファイル管理機能を使用しない環境の場合

このオプションは指定できません。指定するとエラーになります。

マルチファイル管理機能を使用する場合に，一つの文書のコンテントに格納するファイルの最大数を確認するには，動作環境メタ情報ファイル（実行環境ディレクトリ /etc/meta_files/edms.ini）の［bb683102-0bf0-11d2-9a68-0000e20838e7］セクションの dmaProp_MaximumElements エントリの値を参照してください。dmaProp_MaximumElements エントリの値が，一つの文書のコンテントに格納するファイルの最大数です。

-n **ユーザ識別子の最大長**

ユーザ識別子の最大長を 255 バイト以上に拡張する場合に指定します。ユーザ識別子の最大長をバイト単位で指定します。
指定できる値の範囲は 255 ～ 512 の 10 進数値です。このオプションを省略した場合，ユーザ識別子の最大長は 254 バイトになります。
なお，メタ情報を再登録する場合は，ユーザ識別子の最大長を変更できません。-f オプションに動作環境メタ情報ファイル（$DOCBROKERDIR/etc/meta_files/edms.ini）を指定してメタ情報を再登録する場合，-n オプションは次のように扱われます。

ユーザ識別子の最大長を拡張している環境の場合

-n オプションの指定は無視されます。ユーザ識別子の最大長は，動作環境メタ情報ファイルに定義されている値になります。

ユーザ識別子の最大長を拡張していない環境の場合

このオプションは指定できません。指定するとエラーになります。

-g **グループ識別子の最大長**

グループ識別子の最大長を 255 バイト以上に拡張する場合に指定します。グループ識別子の最大長をバイト単位で指定します。
指定できる値の範囲は 255 ～ 512 の 10 進数値です。このオプションを省略した場合，グループ識別子の最大長は 254 バイトになります。
なお，メタ情報を再登録する場合は，グループ識別子の最大長を変更できません。-f オプションに動作環境メタ情報ファイル（$DOCBROKERDIR/etc/meta_files/edms.ini）を指定してメタ情報を再登録する場合，-g オプションは次のように扱われます。

グループ識別子の最大長を拡張している環境の場合

-g オプションの指定は無視されます。グループ識別子の最大長は，動作環境メタ情報ファイルに定義されている値になります。

グループ識別子の最大長を拡張していない環境の場合

このオプションは指定できません。指定するとエラーになります。

-C { SJIS | UTF-8}

文書空間で使用する文字コード種別を指定します。

SJIS

文書空間で使用する文字コード種別を Shift-JIS とします。

UTF-8

文書空間で使用する文字コード種別を UTF-8 とします。

このオプションの指定を省略した場合，Linux 以外のときは，「SJIS」が仮定されます。Linux のときは，「UTF-8」が仮定されます。

文書空間で使用する文字コード種別を変更する場合は，-d オプションと -C オプションを指定してこのコマンドを実行します。

注意事項

- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは，DocumentBroker サーバの停止中に実行してください。
- このコマンドを実行する前に，データベースを起動しておいてください。
- このコマンドを実行すると，メタ情報ファイル一つに対して一つの表を作成します。作成される表には，「EDMS_META_XXXX」という名称が付けられます。「XXXX」には，メタ情報ファイル名から拡張子を除いた文字列が設定されます。例えば，「edms.ini」に対する表は「EDMS_META_edms」という名称が付けられます。
- -d オプションを指定してこのコマンドを実行した場合は，必ず EDMCrtSql コマンドを実行して，出力されたデータベース定義文を基に表を再定義してください。
- 既存の VariableArray 型のプロパティの格納先を Own から Both，または HIRDB から Both 以外へ変更したい場合，次の順序でコマンドを実行して，出力されたデータベース定義文を基に表を再定義する必要があります。
  1. -t オプションを指定して，EDMDelMeta コマンドを実行します。
  2. EDMInitMeta コマンドを実行します。
  3. EDMAddMeta コマンドを実行します。
  4. EDMCrtSql コマンドを実行します。
- DocumentBroker サーバのバージョンが 02-40 のメタ情報ファイルを -f オプションに指定して実行環境を構築すると，実行環境識別子に 0 が割り当てられる場合があります。その条件を次に示します。
  - 新規に実行環境を構築した場合
  - -e オプションを指定した場合
  - -e オプションを指定した EDMDelMeta の実行後，EDMInitMeta を実行した場合
- 複数の実行環境から一つの文書空間にアクセスする実行環境として登録済みの環境で，EDMDelMeta の実行後に，EDMInitMeta を実行してメタ情報を再登録した場合，実行環境識別子は変更されません。複数の実行環境から一つの文書空間にアクセスする実行環境を新規に作成する場合，「3.15　複数の実行環境を構築する場合の設定」の手順に従ってください。この手順に従わないで次の操作を実行した場合，不正に実行環境識別子に 0 が割り当てられます。
  - 実行環境識別子が 0 である実行環境ディレクトリ /etc 下をコピーして，新たに実行環境を作成した場合
  - 複数の実行環境から一つの文書空間にアクセスする実行環境を登録（EDMRegEnvId -r）しないで，EDMInitMeta を実行した場合
  - 複数の実行環境から一つの文書空間にアクセスする実行環境を登録（EDMRegEnvId -r）しないで，実行環境ファイルを出力（EDMRegEnvId -p）した場合

- 複数の実行環境から一つの文書空間にアクセスする実行環境の実行環境識別子を削除（EDMRegEnvId -d）したあと，実行環境を登録（EDMRegEnvId -r）しないで，実行環境ファイルを出力（EDMRegEnvId -p）した場合

EDMLckWatcher（データベースのデッドロックおよびタイムアウトの監視）

機能

データベース（HiRDB）のデッドロック情報およびタイムアウト情報が出力されていないかを監視し，出力されている場合は，デッドロック情報，タイムアウト情報および関連するサービスプロセスのトレース情報を取得します。

このコマンドを実行すると取得できる情報（ファイル）を次に示します。

- HiRDB のデッドロック情報
- HiRDB のタイムアウト情報
- DocumentBroker のサーバのトレースファイル

デッドロック情報およびタイムアウト情報の詳細については，マニュアル「HiRDB　システム運用ガイド」を参照してください。DocumentBroker のサーバのトレースファイルについては，「6.4.1(2) トレースファイル」を参照してください。

また，このコマンドを自動で実行すると，DocumentBroker サーバが起動（EDMStart コマンドを実行）したタイミングで監視が開始され，DocumentBroker サーバが停止（EDMStop コマンドを実行）したタイミングで監視が終了されます。このコマンドを自動で実行する場合は，DocumentSpace 構成定義ファイル（docspace.ini）の DbLockWatcher エントリに「Accept」を設定してください。なお，このコマンドは手動でも実行できます。

形式

```
EDMLckWatcher 〔-i HiRDBの運用ディレクトリ名〕
              〔-l 格納先ディレクトリ名〕
              〔-s 監視時間の間隔〕
              〔-c〕
```

オプション

-i HiRDB の運用ディレクトリ名

デッドロック情報およびタイムアウト情報を取得する HiRDB の運用ディレクトリを指定します。指定したディレクトリを基に，"HiRDB の運用ディレクトリ /spool/pdlckinf" から情報を取得します。
このオプションの指定を省略した場合，環境変数「PDDIR」に設定したディレクトリが有効になります。
このオプションの指定および環境変数「PDDIR」の設定がない場合，エラーになります。

-l 格納先ディレクトリ名

デッドロック情報およびタイムアウト情報の格納先ディレクトリ名をフルパスで指定します。指定されたディレクトリの下にプレフィクス付きのディレクトリを作成し，その下に収集した情報を格納します。プレフィクス付きディレクトリのディレクトリ名を次に示します。

lck_<YYYYMMDDhhmmssxxx>

<YYYYMMDDhhmmssxxx> はコマンドを実行した時刻を表す文字列です。YYYY は西暦年号

（4 けた），MM は月（2 けた），DD は日（2 けた），hh は時（2 けた），mm は分（2 けた），ss は秒（2 けた），xxx はミリ秒（3 けた）を示します。

存在するディレクトリを指定してください。存在しないディレクトリを指定するとエラーになります。

DocumentBroker のシステム管理者に対して，格納先ディレクトリに書き込み権限を設定してください。書き込み権限が設定されていない場合，ファイルを格納できません。

このオプションの指定を省略した場合，環境変数「_HIEDMS_LCKINF_DIR」に設定したディレクトリに格納されます。

このオプションの指定および環境変数「_HIEDMS_LCKINF_DIR」の設定がない場合，エラーになります。

-s 監視時間の間隔（単位：分）

デッドロックおよびタイムアウトを監視する時間の間隔を分単位で指定します。

指定できる値の範囲は 1 ～ 120 の 10 進数値です。範囲外の値を指定するとエラーになります。

このオプションの指定を省略した場合，環境変数「_HIEDMS_LCKWATCH_TIME」の設定値が有効になります。

このオプションの指定および環境変数「_HIEDMS_LCKWATCH_TIME」の設定がない場合，「10」が仮定されます。

-c

デッドロック情報およびタイムアウト情報を退避後，圧縮する場合に指定します。

圧縮されたファイルは，デッドロック情報およびタイムアウト情報が格納されたディレクトリと同じディレクトリに，次のファイル名で格納されます。

lck_<YYYYMMDDhhmmssxxx>.tar.Z

<YYYYMMDDhhmmssxxx> はコマンドを実行した時刻を表す文字列です。YYYY は西暦年号（4 けた），MM は月（2 けた），DD は日（2 けた），hh は時（2 けた），mm は分（2 けた），ss は秒（2 けた），xxx はミリ秒（3 けた）を示します。

このオプションの指定を省略した場合，環境変数「_HIEDMS_LCKINF_ARCHIVE」の設定値が有効になります。

このオプションの指定および環境変数「_HIEDMS_LCKINF_ARCHIVE」の設定がない場合，デッドロック情報およびタイムアウト情報は圧縮されません。

設定が必要な環境変数

コマンドを実行する場合に設定が必要な環境変数を次の表に示します。このコマンドを自動で実行する場合，表に示す環境変数の設定値に従ってコマンドが実行されます。そのため，DocumentBroker サーバを起動する前に，必ず環境変数を設定してください。また，このコマンドを手動で実行する場合，コマンドのオプションの指定を省略すると，表に示す環境変数の設定値に従ってコマンドが実行されます。

表 7-9　EDMLckWatcher コマンドを実行する場合に設定が必要な環境変数

環境変数	説明
PDDIR	HiRDB の運用ディレクトリを設定します。 コマンドを自動で実行する場合，この環境変数の設定は省略できません。設定がないと，エラーになります。
_HIEDMS_LCKINF_DIR	デッドロック情報およびタイムアウト情報の格納先ディレクトリを設定します。 文書空間で使用する文字コード種別が UTF-8 の場合，印刷可能な ASCII コードで値を設定してください。 コマンドを自動で実行する場合，この環境変数の設定は省略できません。設定がないと，エラーになります。

環境変数	説明
_HIEDMS_LCKWATCH_TIME	デッドロックおよびタイムアウトを監視する時間の間隔を設定します。 コマンドを自動で実行する場合，この環境変数の設定を省略すると「10」が仮定されます。
_HIEDMS_LCKINF_ARCHIVE	デッドロック情報およびタイムアウト情報を圧縮するかどうかを設定します。 デッドロック情報およびタイムアウト情報を圧縮する場合は，「Y」または「y」を設定します。「Y」および「y」以外の値を設定すると，デッドロック情報およびタイムアウト情報は圧縮されません。 コマンドを自動で実行する場合，この環境変数の設定を省略すると，デッドロック情報およびタイムアウト情報は圧縮されません。

実行例

コマンドの実行例を次に示します。

```
$EDMLckWatcher -i /usr/HiRDB_P -l /var/tmp -s 10 -c
```

注意事項

- このコマンドを実行できるのは，DocumentBroker サーバと HiRDB サーバを同一マシンで運用している場合だけです。
- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは多重実行できません。自動で実行する場合，同時に手動では実行できません。
- このコマンドは，DocumentBroker サーバの起動中に実行してください。サーバが停止中に実行するとエラーになります。
- このコマンドが起動しているかどうかは，OS の ps コマンドで確認できます。実行例を次に示します。

```
$ps -ef | grep EDMLckWatcher
```

  また，このコマンドを手動で実行した場合にコマンドを停止するときは，上記の ps コマンドの実行結果に出力されるプロセス ID を指定して OS の kill コマンドを実行するか，このコマンドを実行したコマンドプロンプトで，Ctrl ＋ C キーを押してください。

- このコマンドを自動で実行する場合は，DocumentSpace 構成定義ファイル（docspace.ini）の DbLockWatcher エントリに「Accept」を設定します。DbLockWatcher エントリの指定がないと，自動では実行されません。DocumentSpace 構成定義ファイル（docspace.ini）の詳細は，「4.2 DocumentSpace 構成定義ファイル（docspace.ini）」を参照してください。
- このコマンドを手動で実行する場合，コマンドを実行した時刻以降に出力されるデッドロック情報およびタイムアウト情報が監視の対象になります。

EDMPrintMeta（メタ情報ファイルの出力）

機能

データベースに登録済みのメタ情報を基に，動作環境メタ情報ファイルを出力します。

形式

```
EDMPrintMeta　〔-F〕　〔-l 出力先ディレクトリ名〕
```

オプション

-F

すべてのメタ情報ファイルを出力します。このオプションを省略すると，dmaclass.ini などのユーザの追加した定義を含まないメタ情報ファイルは出力されません。

-l 出力先ディレクトリ名

メタ情報ファイルの出力先を指定します。このオプションを省略した場合，" 実行環境ディレクトリ /etc/meta_files" 下に出力します。

注意事項

- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは，DocumentBroker サーバの停止中に実行してください。
- このコマンドを実行する前に，データベースを起動しておいてください。
- 出力するメタ情報ファイルのセクションは，アルファベット順に並びます。この順序は，最初のメタ情報ファイルと異なります。

EDMRefresher（サービスプロセスのリフレッシュ）

機能

すべてのサービスプロセスをリフレッシュします。リフレッシュとは，DocumentBroker サーバを停止することなく，クライアントからの要求に対して文書空間へのサービスを供給しながら，サービスプロセスを順次再起動することです。

サービスプロセスは，EDMStart コマンドで起動した順に，一つずつリフレッシュされます。各サービスプロセスがリフレッシュ対象になったとき，次のどちらかの条件を満たしていると，リフレッシュされます。

- **接続中のユーザがいない，または接続中のユーザはいるがトランザクションはすべて決着している**※
  この場合，猶予時間内であっても，サービスプロセスがリフレッシュされます。

  注※
  どちらの状態でリフレッシュされるのかは，DocumentSpace 構成定義ファイルの RefreshTiming エントリで指定した値によって決定します。

- **コマンド実行後，猶予時間が経過した**
  この場合，接続中のユーザは強制的にサーバとの接続を切断されます。なお，猶予する事象は，DocumentSpace 構成定義ファイルの RefreshTiming エントリで指定した値によって決定します。

形式

```
EDMRefresher　〔-t 猶予時間〕
```

オプション

-t 猶予時間

接続中のユーザがログアウトするための猶予時間，または接続中のユーザが未決着トランザクションを決着するための猶予時間を秒で設定します。0 ～ 7,200 の値を指定してください。指定を省略した場合は，600 を仮定します。0 を指定した場合，接続中のユーザがログアウトする，または接続中のユーザが未決着のトランザクションを決着するのを待たないで，直ちにリフレッシュします。

注意事項

- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは，DocumentBroker サーバの起動中に実行してください。
- このコマンドは，多重実行できません。
- 複数のサービスプロセスがある場合，コマンド実行中にログインしたユーザが，リフレッシュ後のサービスプロセスに接続することは保証されません。リフレッシュ前のサービスプロセスに接続した場合，リフレッシュ実行時にサーバへの接続が強制的に切断される可能性があります。
- このコマンドの実行中にユーザが文書空間に接続した場合，エラーになることがあります。例えば，C++ クラスライブラリを使用している場合，CdbrSession::Connect メソッドで次のどちらかのエラーが返却されることがあります。
  - major_code = ERR_DBR
    minor_code = ERR_NO_SERVICE
  - major_code = ERR_DMA
    minor_code = DMARC_NETWORK_UNAVAILABLE
- サーバへの接続が強制的に切断された場合，API で次のエラーが返却されることがあります。
  major_code = ERR_DBR
  minor_code = ERR_SESSION_NOT_CONNECT
- メモリサイズ監視によるリフレッシュ実行中にこのコマンドを実行した場合，メモリサイズ監視によるリフレッシュの終了を待ってからコマンドによるリフレッシュを実行します。コマンドに指定した猶予時間内にメモリサイズ監視によるリフレッシュが終了しない場合，このコマンドは KMMBR03359-E を出力し終了します。ただし，サーバはメモリサイズ監視によるリフレッシュの終了を待ってから，すべてのサービスプロセスを順次リフレッシュします。
- サービスプロセスをリフレッシュすると，アプリケーションプロセスとサービスプロセスの通信で使用するポート番号がリフレッシュ前後で変更される可能性があります。アプリケーションプロセスがサービスプロセスに接続後，このコマンドを実行し通信ポート番号が変更され，アプリケーションプロセスとリフレッシュ前のサービスプロセスの通信で使用していたポート番号がほかのプログラムに使用されると，以後アプリケーションプロセスとサービスプロセスとの通信で無応答が発生する原因となります。この現象を回避するために，アプリケーションプロセスとサービスプロセスとの通信で使用するポート番号を固定してください。アプリケーションプロセスとサービスプロセスとの通信で使用する固定ポート番号は，サービスプロセス定義ファイルに指定できます。固定ポート番号の指定については，「4.13　サービスプロセス定義ファイル」と，マニュアル「VisiBroker for C++ プログラマーズガイド」またはマニュアル「VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　プログラマーズリファレンス」を参照してください。

EDMRegEnvId（DocumentBroker 実行環境の情報の登録）

機能

DocumentBroker の実行環境についての情報（実行環境識別子，環境変数「DOCBROKERDIR」に指定している実行環境ディレクトリのパス，およびホスト名）をデータベース中の表に登録します。また，登録した実行環境の更新，削除，および登録されている実行環境のファイル出力をします。

なお，このコマンドで登録した実行環境識別子は，該当する実行環境下で作成されるすべてのオブジェクトの OIID に付加されます。

形式

```
EDMRegEnvId  -r 〔-i 実行環境識別子 | -a〕 |
             -u  実行環境識別子          |
             -d                          |
             -p                          |
             -l 〔-o 出力先ファイル名 〕
```

オプション

-r

DocumentBroker の実行環境の情報を登録する場合に指定します。
このオプションを指定すると，データベース中の表に実行環境識別子のレコードが挿入（INSERT）されます。また，実行環境の情報が，実行環境ファイルとして出力されます。

-i 実行環境識別子

登録する実行環境識別子を指定します。1 ～ 254 の範囲で，10 進数の値で指定してください。
このオプションを省略した場合は，1 ～ 254 の範囲で未使用の識別子のうち，最小値の識別子が自動的に採番されます。

-a

実行環境の情報を次のどちらかの方法で削除した場合に，以前登録されていた実行環境識別子で再登録するときに指定します。

- EDMInitMeta コマンドで，-d オプションと -e オプションを指定して実行
- EDMDelMeta コマンドで，-e オプションを指定して実行

-u 実行環境識別子

実行環境の情報を更新する場合に指定します。実行環境を別のマシンに移動したり，実行環境ディレクトリのパスを変更したりした場合は，-u オプションを指定してこのコマンドを実行してください。
このオプションを指定すると，引数に指定した実行環境識別子に対応するレコードのカラム（環境変数「DOCBROKERDIR」に指定している実行環境ディレクトリのパス，およびホスト名）が更新されます。また，実行環境の情報が，実行環境ファイルとして出力されます。

-d

実行環境を削除する場合に指定します。ただし，このコマンドを実行した実行環境の OIID 通番レコードが 0 件の場合だけ削除できます。

-p

実行環境ファイルを，" 実行環境ディレクトリ /etc" に出力する場合に指定します。
このオプションを指定すると，コマンドを実行した実行環境の情報が，実行環境ファイルとして出力されます。
このオプションは，実行環境ファイルが壊れた場合などに使用します。

-l

登録されている実行環境の一覧を表示する場合に指定します。

-o 出力先ファイル名

-l オプションで取得した実行環境の一覧をファイルに出力する場合に指定します。出力先のファイル名を，相対パスまたは絶対パスで指定します。指定されたファイルがすでに存在する場合，上書きされます。指定先のファイルにアクセス権限がない場合，エラーになります。

実行例

-l オプションおよび -o オプションを指定して実行した場合の，出力ファイルの例を次に示します。

```
ID=0      HOST-NAME=----       ENVDIR=----
ID=20     HOST-NAME=Host1      ENVDIR=/user/DocumentBroker
ID=100    HOST-NAME=Host2      ENVDIR=/user/DocumentBroker100
```

注意事項

- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは，DocumentBroker サーバの停止中に実行してください。
- このコマンドを実行する前に，データベースを起動しておいてください。
- このコマンドを実行する前に，EDMPrintMeta -F を実行しておいてください。
- 実行環境識別子としてすでに登録されている識別子を指定して，実行環境を登録しようとすると，メッセージが出力されて，処理が停止されます。次のどちらかの方法で，実行環境識別子が重複しないように登録してください。
  - すでに登録されている実行環境識別子を確認し，未登録の実行環境識別子を指定して，このコマンドを実行する。
    すでに登録されている実行環境識別子を確認するには，-l オプション（ファイル出力する場合は，-l オプションおよび -o オプション）を指定して，このコマンドを実行してください。
  - -i オプションを省略して，このコマンドを実行する。
    1 ～ 254 の範囲で未使用の識別子のうち，最小値の識別子が自動的に採番されます。
- 一度登録された実行環境識別子は削除・変更できません。ただし，実行環境識別子が 0 以外の実行環境については，実行環境でオブジェクトを作成していない場合に（実行環境の OIID 通番レコードが 0 件の場合に），-d オプションを指定してこのコマンドを実行することで，実行環境識別子を削除できます。また，実行環境識別子の削除後，-r オプションを指定してこのコマンドを実行することで，実行環境識別子を変更できます。

EDMSetup（DocumentBroker の実行環境の作成と削除）

機能

指定したディレクトリの下に DocumentBroker の実行環境を作成します。また，DocumentBroker の実行環境を削除します。なお，AIX を使用している場合で，TPBroker V3 を使用しているときは，「/opt/HiEDMS/bin」ディレクトリに格納されている EDMSetup コマンドを実行してください。AIX を使用している場合で，TPBroker V5 を使用しているときは，「/opt/HiEDMS/bin_tp5」ディレクトリに格納されている EDMSetup コマンドを実行してください。Linux を使用している場合は，「/opt/HiEDMS/bin」ディレクトリに格納されている EDMSetup コマンドを実行してください。

形式

```
EDMSetup　〔-m { create | delete }〕　-d ディレクトリ名
```

オプション

-m { create | delete }

実行するモードとして，次のどちらかの文字列を指定します。

create

DocumentBroker の実行環境を作成する場合に指定します。

delete

DocumentBroker の実行環境を削除する場合に指定します。

このオプションを省略した場合，「-m create」を仮定します。

-d ディレクトリ名

実行環境を作成または削除するディレクトリを絶対パスで指定します。実行環境を作成するディレクトリについては，「3.6.1　DocumentBroker 実行環境ディレクトリの作成」を参照してください。なお，インストールディレクトリおよびインストールディレクトリのサブディレクトリを指定した場合は，エラーになります。

実行環境を作成する場合または削除する場合で，処理の内容が異なります。それぞれの場合について，次に説明します。

実行環境を作成する場合

-d オプションで指定したディレクトリの下に DocumentBroker の実行環境を作成します。DocumentBroker 実行環境のディレクトリ構成については，「付録 A.2　DocumentBroker の実行環境ディレクトリの構成（AIX の場合）」または「付録 B.2　DocumentBroker の実行環境ディレクトリの構成（Linux の場合）」を参照してください。ただし，すでに DocumentBroker 実行環境が作成されていた場合は，実行環境を更新します。

実行環境を削除する場合

-d オプションで指定したディレクトリの下から，「bin」，「lib」，「adm」「tmp」，「spool」，および「tools」ディレクトリを削除します。

注意事項

- 実行環境の作成前に，実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。
- このコマンドは，DocumentBroker サーバの停止中に実行してください。
- このコマンドの実行によって作成されるすべてのディレクトリおよびファイルには，インストールディレクトリ下にあるディレクトリおよびファイルと同等のアクセス権が与えられます。
- 実行環境を作成するディレクトリ下に，このコマンドの実行によって作成されるファイルと同じ名称のファイルが存在した場合は，このファイルおよびこのファイルが存在するディレクトリを新規に作成しません。ただし，「adm」ディレクトリ下に同じ名称のファイルが存在した場合だけ，既存の同じ名称のファイルを削除して新規に作成します。
- 実行環境の作成中に，次に示す状態を検知した場合は処理を停止します。ただし，処理が停止するまでに作成されたディレクトリおよびファイルは，削除されません。
  - 実行環境を作成するディレクトリが存在しない。
  - 実行環境を作成するディレクトリにアクセス権が存在しない。
  - 実行環境を作成するディレクトリ下で DocumentBroker がすでに起動している。
  - 実行環境を作成するディレクトリ下に作成対象となるディレクトリ名称と同一のファイルがすでに存在する。

この場合，処理が停止するまでの間に作成されたディレクトリおよびファイルは，削除されません。したがって，必要に応じて「-m delete」を指定して，このコマンドを再度実行したあとに，実行環境を作成し直してください。

- 指定したパスの長さが，512 バイト以上である。
  この場合，ディレクトリおよびファイルは，作成されません。ただし，環境変数「_HIEDMS_TRACE_DIR」が設定されていない場合は，実行環境ディレクトリ /spool/server にトレースファイルが作成されます。

● 実行環境を削除する場合は，-d オプションで指定したディレクトリ下のファイルはコマンドが作成したものかどうかに関係なく，削除されます。削除される対象となるディレクトリ下には不用意にファイルを作成しないでください。

● 実行環境を削除する場合は，-d オプションで指定したディレクトリがあるかどうかはチェックされません。例えば，指定したディレクトリ下に「bin」ディレクトリだけ存在している場合は，そのディレクトリを削除して正常終了とします。ただし，削除対象となるディレクトリやファイルが一つも存在しない場合はエラーになります。

● 実行環境は移動しないでください。また，別ディスクへのリンクも作成しないでください。

● DocumentBroker サーバを再インストールしたり，アンインストールしたりする場合，実行環境の削除（EDMSetup -m delete）を必ず実行してください。実行環境を削除しないで，これらの操作をするとエラーになることがあります。

EDMStart（DocumentBroker の起動）

機能

DocumentBroker サーバを起動します。

形式

```
EDMStart 〔-w 〕
```

オプション

-w

DocumentBroker Server のサービスが開始されてからこのコマンドを終了する場合に指定します。このオプションを指定することで，サービスが開始されたタイミングを知ることができます。
このオプションの指定を省略した場合，DocumentBroker Server のサービスが開始されるのを待たないで，このコマンドを終了します。

注意事項

● このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。環境変数が指定されていない場合，または指定されている実行環境ディレクトリがインストールディレクトリの場合，このコマンドは実行できません。

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

EDMStop（DocumentBroker の終了）

機能

DocumentBroker サーバを終了します。

なお，-l オプションを指定していない場合で，サービスプロセス定義ファイルの〔Entry001〕セクションの DBConnectionClose エントリに「Yes」を指定しているときは，DB コネクションを切断します。

形式

```
EDMStop 〔-l FORCE 〕
```

オプション

-l FORCE

強制終了モードで DocumentBroker サーバを終了する場合（通常終了で終了しない場合）に指定します。

注意事項

- このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。環境変数が指定されていない場合，または指定されている実行環境ディレクトリがインストールディレクトリの場合，このコマンドは実行できません。
- このコマンドは，DocumentBroker サーバの起動中に実行してください。
- DocumentBroker サーバが停止する時間に，DB コネクションの切断処理時間を加算してください。
- DB コネクションの切断でエラーが発生した場合は，KMBR04001-E のメッセージが出力され，DocumentBroker サーバの停止処理は継続されます。EDMStop コマンドは正常終了（終了コードが「0」）となります。

EDMUsrView（文書空間に接続しているユーザー覧出力）

機能

文書空間に接続しているユーザのユーザ情報を取得します。このコマンドを実行すると，取得したユーザ情報を保存したり，表示したりできます。

形式

```
EDMUsrView  {-l <directory name>
            |-v 〔<directory name>〕
                〔-c <connect pass time>〕
                〔-t <transaction pass time>〕
                〔-h <host name>〕
                〔-p <application name>〕
            }
```

オプション

オプションには，機能オプションと表示オプションがあります。次にそれぞれのオプションについて説明していきます。

機能オプション

機能オプションは，一文字の機能文字で表されます。このコマンドでは，-l オプションおよび -v オプションを指定できます。この機能文字の指定によって，ユーザ情報の保存やユーザ情報の表示を選択できます。

-l <directory name>

接続中のユーザのユーザ情報をサービスプロセスごとにファイルに出力します。
引数に directory name を指定します。この引数は省略できません。引数である directory name には，ファイルを出力するディレクトリを絶対パスで，存在するディレクトリを指定してください。ディレクトリが存在しない場合，エラーになります。また，引数に指定するディレクトリにはシステム管理者に対して書き込み権が必要です。書き込み権がない場合はファイルを出力できません。なお，このコマンドの実行者には，引数に指定するディレクトリの読み取り権が必要です。
出力ファイルの形式を次に示します。
" **サービスプロセス通番 _ サービスプロセス** ID.log"
このファイルは，指定されたディレクトリの下に作成された，usr_YYYYMMDDhhmmssxxx ディレクトリ（YYYYMMDDhhmmssxxx はコマンド実行時間）に格納されます。
なお，出力ファイルの内容については，「ユーザ情報の出力形式」を参照してください。

-v 〔<directory name>〕

接続中のユーザのユーザ情報の表示，またはこのコマンドで作成されたファイルの情報を，指定した表示オプションの内容に従って表示します。引数である directory name には，このコマンドによって作成されたディレクトリ（usr_YYYYMMDDhhmmssxxx ディレクトリ）を絶対パスで指定します。ディレクトリが存在しない場合，エラーになります。引数を省略した場合，コマンド実行時に DocumentBroker サーバに接続しているユーザのユーザ情報を表示します。表示されるユーザ情報は，表示オプションの指定に従います。
表示されるユーザ情報の出力形式については，「ユーザ情報の出力形式」を参照してください。

表示オプション

機能オプションとして -v オプションを指定した場合，有効になります。表示オプションには，-c <connect pass time>，-t <transaction pass time>，-h <host name>，および -p <application name> があります。

-c <connect pass time>

ユーザ情報を取得した時間とユーザ情報のレコード内容（コネクト開始時間）を比較して，connect pass time の条件を満たしているユーザ情報を表示します。connect pass time は，次に示す書式で指定できます。

time1-time2

二つの値である time1 および time2 によって時間の範囲を指定します。time1 および time2 の値は，" ＜時＞：＜分＞：＜秒＞ " という形式で指定してください。なお，省略して指定する場合の例を次に示します。

例

- -c 24-48
  24 時間以上，48 時間以内
- -c 12-
  12 時間以上
- -c 0:30-1
  30 分以上，1 時間以内
- -c 0:300-
  300 分以上
- -c 0-12

12 時間以内

-t <transaction pass time>

ユーザ情報を取得した時間とユーザ情報のレコード内容 " トランザクション開始時間 " を比較し，transaction pass time に該当しているユーザ情報を表示します。transaction pass time は，次に示す書式で指定できます。

time1-time2

二つの値である time1 および time2 によって時間の範囲を指定します。time1 および time2 の値は，" ＜時＞：＜分＞：＜秒＞ " という形式で指定してください。なお，省略して指定する場合の例を次に示します。

例

- -t 24-48
  24 時間以上，48 時間以内
- -t 12-
  12 時間以上
- -t 0:30-1
  30 分以上，1 時間以内
- -t 0:300-
  300 分以上
- -t 0-12
  12 時間以内

-h <host name>

host name とユーザ情報のレコード内容（ホスト名）が一致するユーザ情報を表示します。

-p <application name>

application name とユーザ情報のレコード内容（アプリケーション名）が一致するユーザ情報を表示します。

ユーザ情報の出力形式

ユーザ情報は，1 ユーザに対して 1 レコードがファイルに出力されます。1 レコードは，「,」（コンマ）で区切られており，左から順に次のように出力されます。

" サービスプロセス通番 "，" サービスプロセス ID"，**" ホスト名 "**，**" アプリケーション名 "**，"AP **プロセス** ID"，**" コネクト開始時間 "**，**" セッション** ID**（ユーザ名）"**，**" トランザクション開始時間 "**，"HiRDB **クライアントアプリケーションプログラム名（**DB **コネクション名）"**

ユーザ名が特定できない契機では，括弧内には何も出力されません。接続しているユーザがいない場合は，次のように出力されます。

" サービスプロセス通番 "，**" サービスプロセス** ID"，**"（**NO USER**）"**

ユーザ情報が取得できなかった場合は，次のように出力されます。

" サービスプロセス通番 "，**" サービスプロセス** ID"，**"（**INFORMATION NOT FOUND**）"**

実行例

このコマンドの実行例を次に示します。

機能オプションに l を指定した場合の実行例

```
$EDMUsrView -l /var/tmp
KMBR03352-I DocumentBrokerに接続しているユーザの一覧を出力します。
出力先ディレクトリ名のプレフィクスはusr_20000103132732886となります。
KMBR03353-I DocumentBrokerに接続しているユーザの一覧を出力しました。
```

機能オプションに v を指定した場合の実行例

```
$EDMUsrView -v /var/tmp/usr_20000103132732886
KMBR03352-I DocumentBrokerに接続しているユーザの一覧を出力します。

0001,3302,neko,IKTP,3304,2000/01/03
13:26:38.403,00000ce63833a9ba0005349a(ACUser1),,
0002,3303,sagano,MUTP,3305,2000/01/03
13:29:40.312,000070c1387024fe0007e057(ACUser2),,

KMBR03353-I DocumentBrokerに接続しているユーザの一覧を出力しました。
```

注意事項

● このコマンドを実行する前に，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定してください。

機能オプションに l を指定した場合

- このオプションを指定したコマンドは多重実行できません。
- サーバからの応答待ち時間を環境変数「_HIEDMS_USRVIEW_TIMEOUT」に指定できます（デフォルト 60 秒）。
- DocumentBroker サーバからの応答待ち時間を超過した場合，このコマンドはタイムアウトによって異常終了しますが，取得したユーザ情報およびこのコマンドによって作成されたディレクトリは消去しません。
- SIG_INT による割り込みが発生した場合，取得できたユーザ情報およびコマンドによって作成されたディレクトリを消去します。

機能オプションに v を指定した場合

● 引数 directory name を省略した場合の注意事項を示します。

- コマンドは，多重実行できません。
- DocumentBroker サーバが起動している必要があります。
- DocumentBroker サーバからの応答待ち時間を環境変数「_HIEDMS_USRVIEW_TIMEOUT」に指定できます（デフォルト 60 秒）。
- DocumentBroker サーバからの応答待ち時間を超過した場合，コマンドはタイムアウトによって異常終了し，取得できたユーザ情報だけが表示されます。

● 引数 directory name を指定した場合の注意事項を示します。

- コマンドは，多重実行できます。
- DocumentBroker サーバが停止していても，コマンドを実行できます。
- 指定されたディレクトリ下に，取得ユーザ情報管理ファイル .usrview.info が存在しない場合，エラーになります。
- <connect pass time>，および <transaction pass time> に指定できる時間の上限は，9999:9999:9999 です。これ以上の値を指定したり不正な文字列を指定したりした場合，"<connect pass time> is

invalid parameter"，または "<transaction pass time> is invalid parameter" が表示されて，Usage エラーになります。また，"::0-" など，省略のしかたが不正な場合についても同様のエラーになります。

EDMXmlMap（XML 定義ファイルの追加／更新／削除）

機能

DocumentBroker の使用する XML 定義ファイルを追加，更新，および削除します。次に示す四つの機能を持ちます。

1. 新規マッピング定義の追加および更新機能
   HiRDB Adapter for XML のマッピング定義ファイルを作成します。
2. フィルタリング定義ファイルの追加および更新機能
3. マッピング定義一覧の表示機能
4. 指定マッピング定義の削除機能

形式

```
EDMXmlMap  {   -p プロパティマッピング定義ファイル名（DPM）
               -m マッピング定義名
               -c マッピング元XMLタグ定義ファイル名（DCD）
              〔-t フィルタリング定義ファイル名（TFD）〕
              〔-u〕
             |
               -m マッピング定義名
               -t フィルタリング定義ファイル名（TFD）
              〔-u〕
             |
               -l
             |
               -d マッピング定義名
           }
```

オプション

-p

プロパティマッピング定義ファイル（DPM）のパス名を指定します。

-m

マッピング定義名を指定します。追加したマッピング定義を，この定義名で使用できるようになります。また，指定した定義名は，次に示すファイルのファイル名の一部として使用されます。

- 作成するマッピング定義ファイル（XMP）
- 複製（コピー）されるマッピング元 XML タグ定義ファイル（DCD）
- 複製（コピー）されるフィルタリング定義ファイル（TFD）

ただし，定義名には，全角文字を指定できません。また，パス名の長さには OS に依存した上限値があるため，指定するマッピング定義名には，ディレクトリ名と文書空間識別子および拡張子の長さも考慮して，作成するファイルがパス名の上限値を超えないように値を指定してください。なお，マッピング定義名は，HiRDB Adapter for XML でマッピング名として使用しますので，マッピング名の規則に従った値を指定してください。

-c

使用するマッピング元 XML タグ定義ファイル（DCD）のパス名を指定します。
指定された DCD ファイルのコピーを，$DOCBROKERDIR/etc/xml_files/' 文書空間識別子 '_' マッピ

ング定義名 '.dcd として作成します。

-t

使用するフィルタリング定義ファイル（TFD）のパス名を指定します。
指定された TFD ファイルのコピーを，$DOCBROKERDIR/etc/xml_files/' 文書空間識別子 '_' マッピング定義名 '.tfd として作成します。

-u

マッピング定義の上書きの実行を指定します。このオプションを指定しないと次に示す場合にはエラーになり，マッピング定義の上書きを実行できません。

1. 同名のマッピング定義がすでに XMS ファイルに登録されている場合
2. DCD ファイルの複製（コピー）時，同名の DCD ファイルが出力先ディレクトリに存在する場合
3. TFD ファイルの複製（コピー）時，同名の TFD ファイルが出力先ディレクトリに存在する場合
4. 作成する XMP ファイルと同名のファイルが，出力先ディレクトリに存在する場合

このオプションを指定すると上書き不可のエラーは回避できますが，上記の四つすべての条件について判定されません。このため，このオプションを指定する場合は，マッピング定義の上書きを実行しても問題がないかどうか十分注意してください。

-l

マッピング定義の一覧の表示を指定します。このオプションを指定する場合，単独で指定してください。ほかのオプションと同時に指定した場合，エラーになります。

-d

マッピングセット定義ファイル（XMS）中から削除するマッピング定義を指定します。特定のマッピング定義を削除したい場合に指定します。このオプションを指定する場合，単独で指定してください。ほかのオプションと同時に指定した場合，エラーになります。

コマンドの使用方法

このコマンドには，4 種類の使用方法があります。

- 新規マッピング定義を追加してマッピングセット定義を更新する場合
- フィルタリング定義ファイルを追加してマッピング定義を更新する場合
- マッピング定義を一覧表示する場合
- マッピング定義を削除する場合

それぞれの方法について説明します。

新規マッピング定義を追加してマッピングセット定義を更新する場合

新規マッピング定義を追加してマッピングセット定義を更新する場合は，次に示す形式でコマンドを実行します。

形式

```
EDMXmlMap    -p プロパティマッピング定義ファイル名（DPM）
             -m マッピング定義名
             -c マッピング元XMLタグ定義ファイル名（DCD）
            〔-t フィルタリング定義ファイル名（TFD）〕
            〔-u〕
```

実行する処理

指定されたプロパティマッピング定義ファイル（DPM）を解析してマッピング定義に変換し，マッピング定義ファイル（XMP）を作成します。さらに，マッピング元 XML タグ定義ファイル（DCD）とフィルタリング定義ファイル（TFD）を登録用に名称を変更して複製（コピー）します。また，作成

したマッピング定義を指定されたマッピング定義名で使用できるように，マッピングセット定義ファイル（XMS）を更新します。
コマンドが作成および参照するファイルの名称を次に示します。

- マッピング定義ファイル（XMP）
  '**文書空間識別子**'_'**マッピング定義名**'.xmp
- マッピング元 XML タグ定義ファイル（DCD）
  '**文書空間識別子**'_'**マッピング定義名**'.dcd
- フィルタリング定義ファイル（TFD）
  '**文書空間識別子**'_'**マッピング定義名**'.tfd
- マッピングセット定義ファイル（XMS）
  '**文書空間識別子名**'.xms

これらの XML 定義ファイルは，次に示すディレクトリに格納されます。
$DOCBROKERDIR/etc/xml_files
登録済みのマッピング定義を使用した場合，この登録済みのマッピング定義を更新します。この時点で，TFD ファイルを指定しない場合は，TFD ファイルは登録済みのファイルが継続して使用されます。

実行例

コマンドの実行例を説明します。

例

$DOCBROKERDIR/etc/xml_files の下に「MAPPING.dpm」という名称の DPM ファイルと「BASE.dcd」という名称の DCD ファイルを格納し，「SAMPLE」というマッピング定義名でコマンドを実行します。

- この場合，新規に作成されるのは '文書空間識別子'_SAMPLE という名前を持つ DCD ファイルと XMP ファイルの組です。
- 作成したマッピング定義を使用するには，クライアント環境に「文書空間識別子」のプリフィクスの付いた XMS ファイル，XMP ファイル，および DCD ファイルを複製（コピー）する必要があります。

コマンドの指定形式を次に示します。

コマンドの指定形式

```
EDMXmlMap  -p $DOCBROKERDIR/etc/xml_files/SAMPLE.dpm
           -m SAMPLE
           -c $DOCBROKERDIR/etc/xml_files/BASE.dcd
```

このコマンドの実行によって作成されるファイルを次の表に示します。

表 7-10　コマンドの実行によって作成されるファイル

コマンドの実行状態	コマンド実行前	コマンド実行後
ディレクトリに格納されるファイル	• MAPPING.dpm • BASE.dcd • '文書空間識別子'.xms	• MAPPING.dpm • BASE.dcd • '文書空間識別子'_SAMPLE.dcd（複製） • '文書空間識別子'_SAMPLE.xmp（作成） • '文書空間識別子'.xms（更新）

なお，これらのファイルは次のディレクトリに格納されます。

$DOCBROKERDIR/etc/xml_files/

フィルタリング定義ファイルを追加してマッピング定義を更新する場合

既存のマッピング定義にフィルタリング定義ファイル（TFD）を追加してマッピング定義を更新する場合，次に示す形式でコマンドを実行します。

形式

```
EDMXmlMap  -m マッピング定義名
           -t フィルタリング定義ファイル名（TFD）
          〔-u〕
```

実行する処理

指定されたフィルタリング定義ファイル（TFD）の複製（コピー）が，出力先のディレクトリに，'文書空間識別子 '_' マッピング定義名 '.tfd というファイル名で作成されます。この処理は，TFD ファイルがない登録済みのマッピング定義に TFD ファイルを追加したり，登録済みの TFD ファイルだけを更新したりする場合に実行します。

なお，指定されたマッピング定義がマッピングセット定義ファイル（XMS）に登録されていない場合は，エラーになります。

マッピング定義を一覧表示する場合

すでにマッピングセット定義ファイル（XMS）に登録されているマッピング定義の一覧を表示する場合，次に示す形式でコマンドを実行します。

形式

```
EDMXmlMap   -l
```

実行する処理

マッピングセット定義ファイル（XMS）を参照して，登録されているマッピング定義の情報を一覧表示します。表示される情報を次に示します。

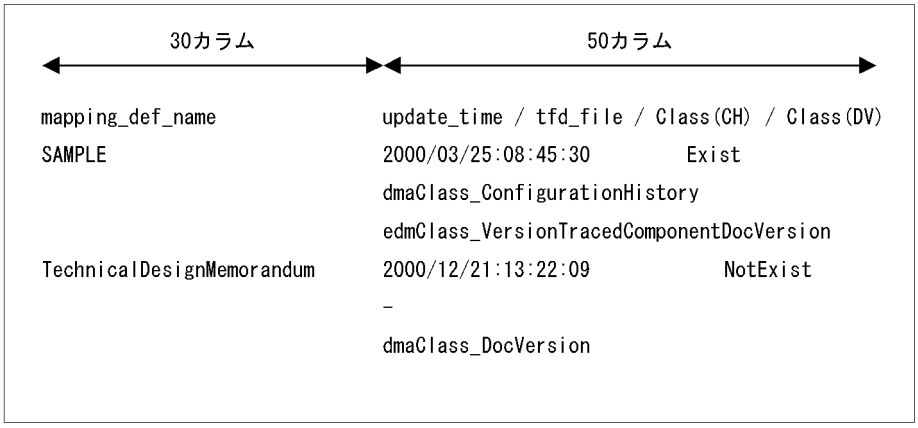
- マッピング定義名
- マッピング定義の追加および更新日時
- マッピング定義使用クラス名（dmaClass_ConfigurationHistory クラスのサブクラス）
- マッピング定義使用クラス名（dmaClass_DocVersion クラスのサブクラス）
- フィルタリング定義ファイル（TFD）の有無

有効なマッピング定義が登録されていない場合には，一覧表示されないで，警告メッセージが出力されます。

実行結果

このコマンドオプションを指定して，コマンドを実行した結果を次の図に示します。

図 7-1 マッピング定義の一覧表示



- 表示は，すべて左寄せで出力されます。
- 更新日時のあとに 4 カラム空けて，TFD ファイルの有無の情報を出力します（「Exist」か「NotExist」のどちらかを出力します）。
- マッピング定義名が 26 カラムを超える場合には，マッピング定義名をすべて出力したあと，4 カラム空けて更新日時を続けて出力します。そのあと，4 カラム空けて TFD ファイルの有無を出力します。
- クラス名は，50 カラムを超えてもそのまま出力します。
- dmaClass_ConfigurationHistory クラス，または dmaClass_DocVersion クラスのサブクラスのマッピング定義が登録されていなければ，「-」（マイナス記号）を出力します。

マッピング定義を削除する場合

登録済みのマッピング定義を使用できないように，マッピング定義一覧から指定マッピング定義を削除する場合，次に示す形式でコマンドを実行します。

形式

```
EDMXmlMap   -d  削除するマッピング定義名
```

実行する処理

指定されたマッピング定義がマッピングセット定義ファイル（XMS）に登録されていれば，その定義情報を削除します。この場合，$DOCBROKERDIR/etc/xml_files に存在する，次に示す名称のファイルを削除します。

- ' 文書空間識別子 '_' マッピング定義名 '.xmp
- ' 文書空間識別子 '_' マッピング定義名 '.dcd
- ' 文書空間識別子 '_' マッピング定義名 '.tfd

なお，指定されたマッピング定義が存在しない場合，エラーになります。
また，登録済みのマッピング定義からフィルタリング定義ファイル（TFD）だけを削除することはできません。

注意事項

● このコマンドは，DocumentBroker サーバの停止中に実行してください。

付録

付録 A　ディレクトリ構成（AIX の場合）

付録 B　ディレクトリ構成（Linux の場合）

付録 C　クラスタリングシステムでの運用

付録 D　全文検索インデクスに UCS-4 の文字を使用する場合のデータベースの設定

付録 E　データベース移行

付録 F　システムクラスおよびシステムプロパティの名称定義の規則

付録 G　ユーザ作成のアクセスルーチンを使用するための関数

付録 H　メタ情報の記述内容

付録 I　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機能およびコマンド

付録 J　DocumentBroker Web Client を使用した DocumentBroker クライアントの開発

付録 K　このマニュアルの参考情報

付録 L　用語解説

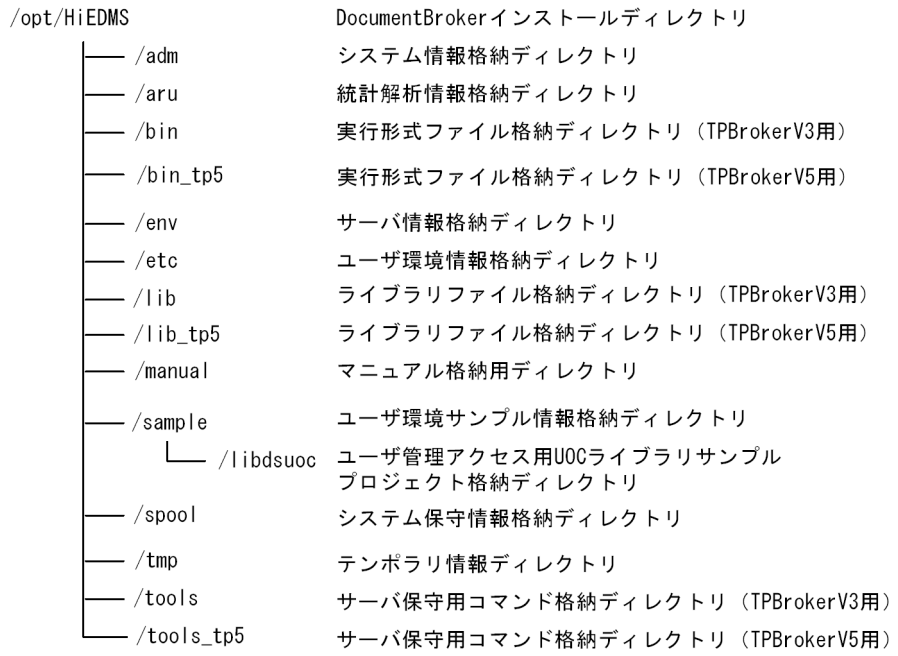
付録 A　ディレクトリ構成（AIX の場合）

DocumentBroker のインストールディレクトリの構成，および実行環境ディレクトリの構成について説明します。

付録 A.1　DocumentBroker のインストールディレクトリの構成（AIX の場合）

DocumentBroker のインストールディレクトリの構成を次の図に示します。

図 A-1　DocumentBroker のインストールディレクトリの構成



次に，各ディレクトリについて説明します。

DocumentBroker インストールディレクトリ（/opt/HiEDMS）

DocumentBroker のインストールディレクトリです。

システム情報格納ディレクトリ（/opt/HiEDMS/adm）

システム情報を格納するディレクトリです。デフォルトの環境情報ファイルを格納します。

統計解析情報格納ディレクトリ（/opt/HiEDMS/aru）

DocumentBroker 統計解析ツールが使用するディレクトリです。詳細については，マニュアル「DocumentBroker Version 3　統計解析ツール」を参照してください。

実行形式ファイル格納ディレクトリ（/opt/HiEDMS/bin）（/opt/HiEDMS/bin_tp5）

運用コマンドや実行形式のファイルを格納するディレクトリです。

サーバ情報格納ディレクトリ（/opt/HiEDMS/env）

DocumentBroker サーバの動作環境に関する情報を格納するディレクトリです。

ユーザ環境情報格納ディレクトリ（/opt/HiEDMS/etc）

ユーザが定義する環境定義ファイルを格納するディレクトリです。環境定義ファイルの詳細については，「付録 A.2　DocumentBroker の実行環境ディレクトリの構成（AIX の場合）」を参照してください。

ライブラリファイル格納ディレクトリ（/opt/HiEDMS/lib）（/opt/HiEDMS/lib_tp5）

DocumentBroker が使用するライブラリファイルを格納するディレクトリです。

マニュアル格納用ディレクトリ（/opt/HiEDMS/manual）

マニュアルのデータをバージョンごとに格納するディレクトリです。DocumentBroker Server で提供するマニュアルは，/server に格納されています。また，DocumentBroker Development Kit で提供するマニュアルは，/devkit に格納されています。

ユーザ環境サンプル情報格納ディレクトリ（/opt/HiEDMS/sample）

DocumentBroker サーバを動作させるための関連プログラムの設定に必要なサンプルを格納するディレクトリです。

ユーザ管理アクセス用 UOC ライブラリサンプルプロジェクト格納ディレクトリ（/opt/HiEDMS/sample/libdsuoc）

次に示すファイルを格納するディレクトリです。

- libdsuoc.c（ソースファイル）
- libdsuoc.h（ヘッダファイル）
- makefile（メイクファイル）
- libdsuoc.a（共用ライブラリ）

システム保守情報格納ディレクトリ（/opt/HiEDMS/spool）

ログ情報などのシステム保守情報を格納するディレクトリです。DocumentBroker サーバが動的に生成します。

テンポラリ情報ディレクトリ（/opt/HiEDMS/tmp）

DocumentBroker サーバの稼働中の作業領域として動的に利用するディレクトリです。

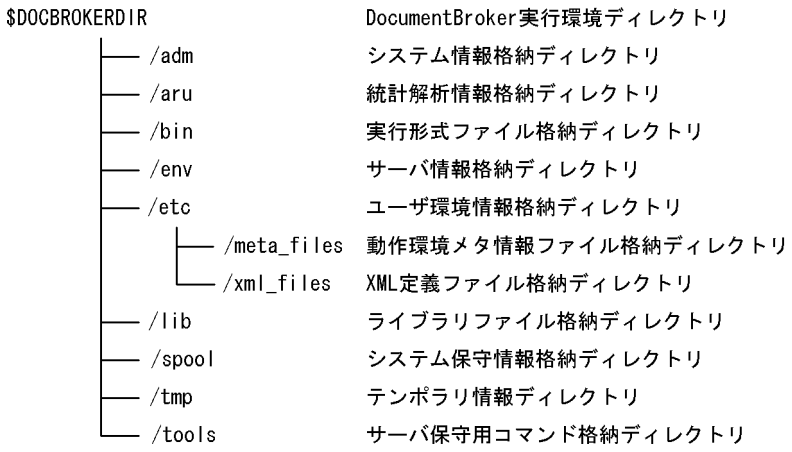
サーバ保守用コマンド格納ディレクトリ（/opt/HiEDMS/tools）（/opt/HiEDMS/tools_tp5）

トラブルシュートコマンドを格納するディレクトリです。

付録 A.2　DocumentBroker の実行環境ディレクトリの構成（AIX の場合）

DocumentBroker の実行環境ディレクトリの構成を次の図に示します。

図 A-2　DocumentBroker の実行環境ディレクトリの構成



次に，各ディレクトリについて説明します。

DocumentBroker 実行環境ディレクトリ（$DOCBROKERDIR）

DocumentBroker の実行環境として設定したディレクトリです。

システム情報格納ディレクトリ（$DOCBROKERDIR/adm）

システム情報を格納するディレクトリです。デフォルトの環境情報ファイルが格納されます。インストールディレクトリの同名ディレクトリ（/opt/HiEDMS/adm）からコピーされます。

統計解析情報格納ディレクトリ（$DOCBROKERDIR/aru）

DocumentBroker 統計解析ツールが使用するディレクトリです。詳細については，マニュアル「DocumentBroker Version 3　統計解析ツール」を参照してください。

実行形式ファイル格納ディレクトリ（$DOCBROKERDIR/bin）

運用コマンドや実行形式のファイルを格納するディレクトリです。インストールディレクトリの同名ディレクトリ「/opt/HiEDMS/bin」（TPBroker V3 の場合）または「/opt/HiEDMS/bin_tp5」（TPBroker V5 の場合）へリンクしています。

サーバ情報格納ディレクトリ（$DOCBROKERDIR/env）

DocumentBroker サーバの動作環境に関する情報を格納するディレクトリです。

ユーザ環境情報格納ディレクトリ（$DOCBROKERDIR/etc）

動作環境メタ情報ファイル，DocumentSpace 構成定義ファイルなど，ユーザが定義する環境定義ファイルを格納するディレクトリです。インストールディレクトリの同名ディレクトリ（/opt/HiEDMS/etc）からコピーされます。なお，動作環境メタ情報ファイルは，直接編集しないでください。メタ情報を変更したい場合は，データベース運用コマンドを使用してください。

動作環境メタ情報ファイル格納ディレクトリ（$DOCBROKERDIR/etc/meta_files）に格納される動作環境メタ情報ファイルを，次の表に示します。

表 A-1　動作環境メタ情報ファイル一覧

ファイル名	説明	改変の可否
edms.ini	文書空間内に生成するオブジェクトの定義ファイル	○
dsclass.ini	文書空間内に定義するクラスの ClassDescription の定義ファイル	×
dsprop.ini	文書空間内に定義するクラスの PropertyDescription の定義ファイル	×

ファイル名	説明	改変の可否
dmaclass.ini	dsclass.ini で定義する ClassDescription 以外の ClassDescription を定義するファイル	×
dmaprop.ini	dsprop.ini で定義する PropertyDescription 以外の PropertyDescription を定義するファイル	×
dmaproto.ini	DocumentBroker の実行に必要な定義ファイル	×
dsqop.ini	検索オペレータの定義ファイル	×
edmqop.ini	拡張検索オペレータの定義ファイル	×
edmclass.ini	文書空間内に定義する拡張クラスの ClassDescription の定義ファイル	×
edmprop.ini	文書空間内に定義する拡張クラスの PropertyDescription の定義ファイル	×
edmsys.ini	DocumentBroker 内部で使用するプロパティの定義ファイル	×
edmsysclass.ini	DocumentBroker 内部で使用するプロパティの ClassDescription の定義ファイル	×
edmsysprop.ini	DocumentBroker 内部で使用するプロパティの PropertyDescription の定義ファイル	×
edmnmclass.ini	クラス名と表識別子の情報ファイル。クラス名を表識別子とする場合に利用するファイル	×
edmnmprop.ini	プロパティ名と列名の情報ファイル。プロパティ名を列名とする場合に利用するファイル	×

（凡例）
○：改変できます。
×：改変できません。

ユーザ環境情報格納ディレクトリに格納される動作環境メタ情報ファイル以外の環境定義ファイルを，次の表に示します。

表 A-2　環境定義ファイル一覧

ファイル名	説明	改変の可否
EXTENSION_MIME.ini	拡張子－ MimeType 対応表	○
PUBID_MIME.ini	公開識別子－ MimeType 対応表	○
SYSID_MIME.ini	システム識別子－ MimeType 対応表	○
slocalreg.ini	System オブジェクトレジストリファイル	○
docspace.ini	DocumentSpace 構成定義ファイル	○
docaccess.ini	セキュリティ定義ファイル	○
userperm.ini ※	ユーザ権限定義ファイル	○
process.ini	サービスプロセス定義ファイル	○
getrascustom.ini	障害情報取得カスタマイズファイル	○
その他	DocumentBroker 実行時に必要とするファイルおよびディレクトリ	×

（凡例）
○：改変できます。
×：改変できません。
注※　サンプルファイルです。定義内容に応じて編集できます。

XML 定義ファイル格納ディレクトリ（$DOCBROKERDIR/etc/xml_files）

XML 定義ファイルを格納するディレクトリです。XML 文書管理機能を使用する場合，このディレクトリに格納されている定義ファイルを DocumentBroker クライアントに転送する必要があります。

ライブラリファイル格納ディレクトリ（$DOCBROKERDIR/lib）

DocumentBroker が使用するライブラリファイルを格納するディレクトリです。インストールディレクトリの同名ディレクトリ「/opt/HiEDMS/lib」（TPBroker V3 の場合）または「/opt/HiEDMS/lib_tp5」（TPBroker V5 の場合）へリンクしています。

システム保守情報格納ディレクトリ（$DOCBROKERDIR/spool）

ログ情報などのシステム保守情報を格納するディレクトリです。
DocumentBroker の起動中は，このディレクトリ下を操作しないでください。DocumentBroker が異常終了することがあります。

テンポラリ情報ディレクトリ（$DOCBROKERDIR/tmp）

DocumentBroker の稼働中の作業領域として動的に利用するディレクトリです。

サーバ保守用コマンド格納ディレクトリ（$DOCBROKERDIR/tools）

DocumentBroker サーバ保守用コマンドを格納するディレクトリです。インストールディレクトリの同名ディレクトリ「/opt/HiEDMS/tools」（TPBroker V3 の場合）または「/opt/HiEDMS/tools_tp5」（TPBroker V5 の場合）へリンクしています。

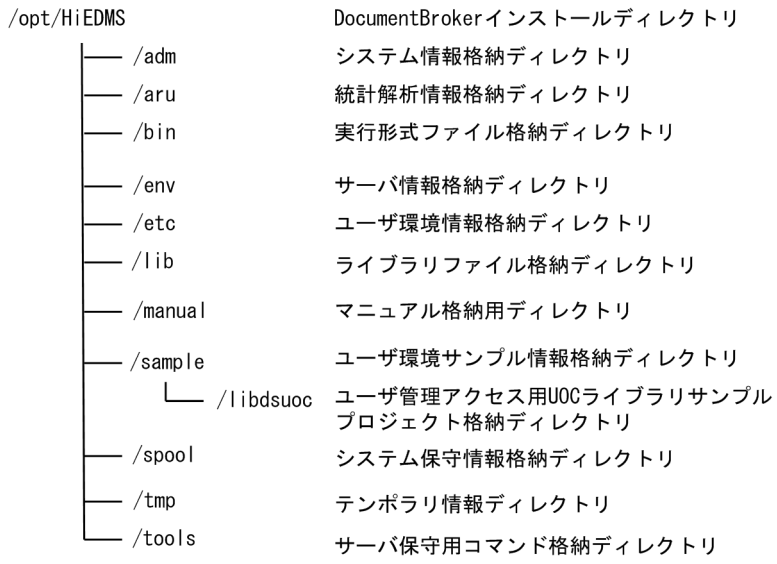
付録 B　ディレクトリ構成（Linux の場合）

DocumentBroker のインストールディレクトリの構成，および実行環境ディレクトリの構成について説明します。

付録 B.1　DocumentBroker のインストールディレクトリの構成（Linux の場合）

DocumentBroker のインストールディレクトリの構成を次の図に示します。

図 B-1　DocumentBroker のインストールディレクトリの構成



次に，各ディレクトリについて説明します。

DocumentBroker インストールディレクトリ（/opt/HiEDMS）

DocumentBroker のインストールディレクトリです。

システム情報格納ディレクトリ（/opt/HiEDMS/adm）

システム情報を格納するディレクトリです。デフォルトの環境情報ファイルを格納します。

統計解析情報格納ディレクトリ（/opt/HiEDMS/aru）

DocumentBroker 統計解析ツールが使用するディレクトリです。詳細については，マニュアル「DocumentBroker Version 3　統計解析ツール」を参照してください。

実行形式ファイル格納ディレクトリ（/opt/HiEDMS/bin）

運用コマンドや実行形式のファイルを格納するディレクトリです。

サーバ情報格納ディレクトリ（/opt/HiEDMS/env）

DocumentBroker サーバの動作環境に関する情報を格納するディレクトリです。

ユーザ環境情報格納ディレクトリ（/opt/HiEDMS/etc）

ユーザが定義する環境定義ファイルを格納するディレクトリです。環境定義ファイルの詳細については，「付録 B.2　DocumentBroker の実行環境ディレクトリの構成（Linux の場合）」を参照してください。

ライブラリファイル格納ディレクトリ（/opt/HiEDMS/lib）

DocumentBroker が使用するライブラリファイルを格納するディレクトリです。

ユーザ環境サンプル情報格納ディレクトリ（/opt/HiEDMS/sample）

DocumentBroker サーバを動作させるための関連プログラムの設定に必要なサンプルを格納するディレクトリです。

ユーザ管理アクセス用 UOC ライブラリサンプルプロジェクト格納ディレクトリ（/opt/HiEDMS/sample/libdsuoc）

次に示すファイルを格納するディレクトリです。

- libdsuoc.c（ソースファイル）
- libdsuoc.h（ヘッダファイル）
- makefile（メイクファイル）
- libdsuoc.so（共用ライブラリ）

システム保守情報格納ディレクトリ（/opt/HiEDMS/spool）

ログ情報などのシステム保守情報を格納するディレクトリです。DocumentBroker サーバが動的に生成します。

テンポラリ情報ディレクトリ（/opt/HiEDMS/tmp）

DocumentBroker サーバの稼働中の作業領域として動的に利用するディレクトリです。

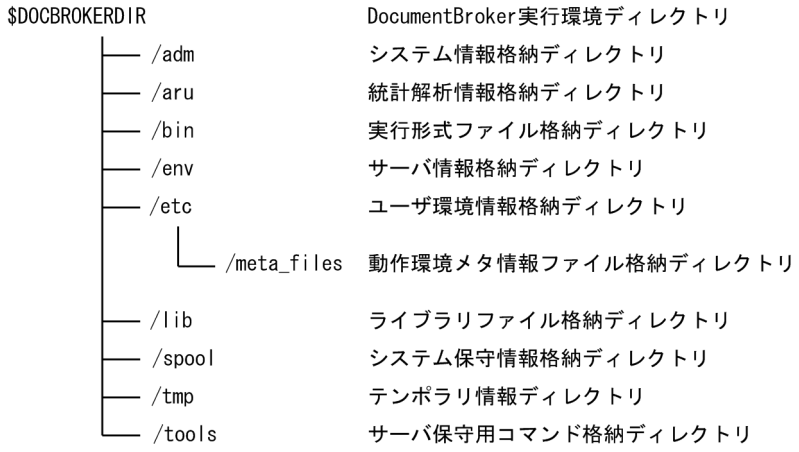
サーバ保守用コマンド格納ディレクトリ（/opt/HiEDMS/tools）

トラブルシュートコマンドを格納するディレクトリです。

付録 B.2　DocumentBroker の実行環境ディレクトリの構成（Linux の場合）

DocumentBroker の実行環境ディレクトリの構成を次の図に示します。

図 B-2　DocumentBroker の実行環境ディレクトリの構成



次に，各ディレクトリについて説明します。

DocumentBroker 実行環境ディレクトリ（$DOCBROKERDIR）

DocumentBroker の実行環境として設定したディレクトリです。

システム情報格納ディレクトリ（$DOCBROKERDIR/adm）

システム情報を格納するディレクトリです。デフォルトの環境情報ファイルが格納されます。インストールディレクトリの同名ディレクトリ（/opt/HiEDMS/adm）からコピーされます。

統計解析情報格納ディレクトリ（$DOCBROKERDIR/aru）

DocumentBroker 統計解析ツールが使用するディレクトリです。詳細については，マニュアル「DocumentBroker Version 3　統計解析ツール」を参照してください。

実行形式ファイル格納ディレクトリ（$DOCBROKERDIR/bin）

運用コマンドや実行形式のファイルを格納するディレクトリです。インストールディレクトリの同名ディレクトリ「/opt/HiEDMS/bin」へリンクしています。

サーバ情報格納ディレクトリ（$DOCBROKERDIR/env）

DocumentBroker サーバの動作環境に関する情報を格納するディレクトリです。

ユーザ環境情報格納ディレクトリ（$DOCBROKERDIR/etc）

動作環境メタ情報ファイル，DocumentSpace 構成定義ファイルなど，ユーザが定義する環境定義ファイルを格納するディレクトリです。インストールディレクトリの同名ディレクトリ（/opt/HiEDMS/etc）からコピーされます。なお，動作環境メタ情報ファイルは，直接編集しないでください。メタ情報を変更したい場合は，データベース運用コマンドを使用してください。

動作環境メタ情報ファイル格納ディレクトリ（$DOCBROKERDIR/etc/meta_files）に格納される動作環境メタ情報ファイルを，次の表に示します。

表 B-1　動作環境メタ情報ファイル一覧

ファイル名	説明	改変の可否
edms.ini	文書空間内に生成するオブジェクトの定義ファイル	○
dsclass.ini	文書空間内に定義するクラスの ClassDescription の定義ファイル	×
dsprop.ini	文書空間内に定義するクラスの PropertyDescription の定義ファイル	×
dmaclass.ini	dsclass.ini で定義する ClassDescription 以外の ClassDescription を定義するファイル	×
dmaprop.ini	dsprop.ini で定義する PropertyDescription 以外の PropertyDescription を定義するファイル	×
dmaproto.ini	DocumentBroker の実行に必要な定義ファイル	×
dsqop.ini	検索オペレータの定義ファイル	×
edmqop.ini	拡張検索オペレータの定義ファイル	×
edmclass.ini	文書空間内に定義する拡張クラスの ClassDescription の定義ファイル	×
edmprop.ini	文書空間内に定義する拡張クラスの PropertyDescription の定義ファイル	×
edmsys.ini	DocumentBroker 内部で使用するプロパティの定義ファイル	×
edmsysclass.ini	DocumentBroker 内部で使用するプロパティの ClassDescription の定義ファイル	×
edmsysprop.ini	DocumentBroker 内部で使用するプロパティの PropertyDescription の定義ファイル	×
edmnmclass.ini	クラス名と表識別子の情報ファイル。クラス名を表識別子とする場合に利用するファイル	×
edmnmprop.ini	プロパティ名と列名の情報ファイル。プロパティ名を列名とする場合に利用するファイル	×

（凡例）
○：改変できます。
×：改変できません。

ユーザ環境情報格納ディレクトリに格納される動作環境メタ情報ファイル以外の環境定義ファイルを，次の表に示します。

表 B-2　環境定義ファイル一覧

ファイル名	説明	改変の可否
slocalreg.ini	System オブジェクトレジストリファイル	○
docspace.ini	DocumentSpace 構成定義ファイル	○
docaccess.ini	セキュリティ定義ファイル	○
userperm.ini ※	ユーザ権限定義ファイル	○
process.ini	サービスプロセス定義ファイル	○
getrascustom.ini	障害情報取得カスタマイズファイル	○
その他	DocumentBroker 実行時に必要とするファイルおよびディレクトリ	×

（凡例）
○：改変できます。
×：改変できません。
注※　サンプルファイルです。定義内容に応じて編集できます。

ライブラリファイル格納ディレクトリ（$DOCBROKERDIR/lib）

DocumentBroker が使用するライブラリファイルを格納するディレクトリです。インストールディレクトリの同名ディレクトリ「/opt/HiEDMS/lib」へリンクしています。

システム保守情報格納ディレクトリ（$DOCBROKERDIR/spool）

ログ情報などのシステム保守情報を格納するディレクトリです。
DocumentBroker の起動中は，このディレクトリ下を操作しないでください。DocumentBroker が異常終了することがあります。

テンポラリ情報ディレクトリ（$DOCBROKERDIR/tmp）

DocumentBroker の稼働中の作業領域として動的に利用するディレクトリです。

サーバ保守用コマンド格納ディレクトリ（$DOCBROKERDIR/tools）

DocumentBroker サーバ保守用コマンドを格納するディレクトリです。インストールディレクトリの同名ディレクトリ「/opt/HiEDMS/tools」へリンクしています。

付録 C　クラスタリングシステムでの運用

DocumentBroker をクラスタリングシステムで運用すると，障害が発生した場合のシステムの可用性やシステムのパフォーマンスを高めることができます。DocumentBroker では，次に示すクラスタリングシステムが使用できます。なお，Linux の場合はクラスタリンクシステムでの運用はできません。

● HACMP
AIX の場合に使用するクラスタリングシステムです。

ここでは，DocumentBroker をそれぞれのクラスタリングシステムで運用するための方法について説明します。

付録 C.1　HACMP によるクラスタリングシステムでの運用

DocumentBroker Server は，HACMP を使用して，クラスタリングシステムで運用できます。

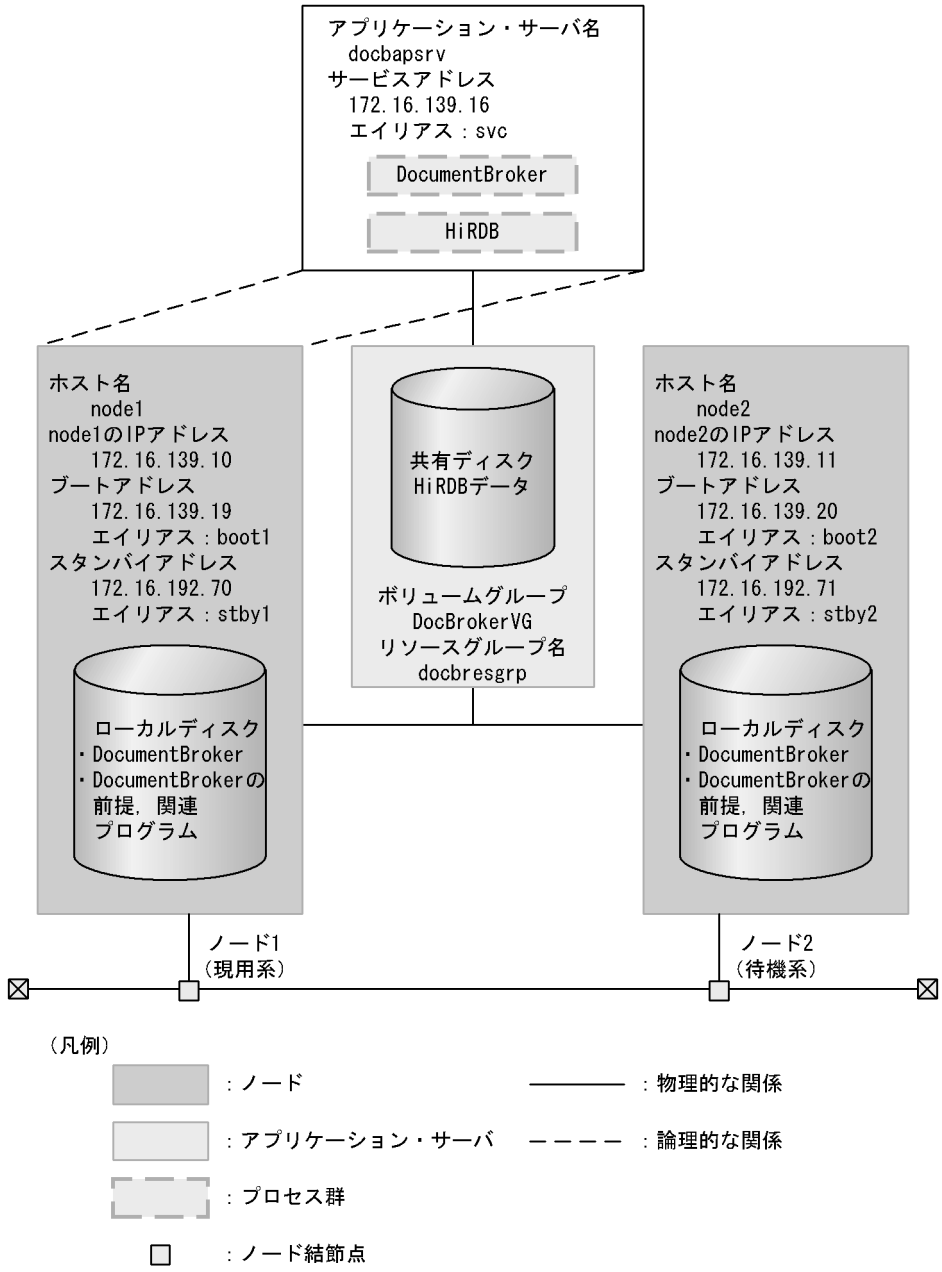
HACMP の詳細については，HACMP に関するマニュアルを参照してください。また，DocumentBroker Server の前提プログラムおよび関連プログラムをクラスタリングシステム構成で運用する場合の詳細については，各プログラムのマニュアルを参照してください。

(1) システム構成例

HACMP を使用して，DocumentBroker Server 環境をクラスタリングシステムで構成した例を，次の図に示します。

なお，以降，「付録 C.1　HACMP によるクラスタリングシステムでの運用」内では，すべてこの構成例の内容を基に説明します。

図 C-1　HACMP を使用したクラスタリングシステム構成例



HACMP を使用したクラスタリングシステム構成例でのクラスタ構成項目について，次の表に示します。

表 C-1　クラスタ構成項目

項目	値
クラスタ名	docbcluster
クラスタ ID	1001
現用系ノードの IP アドレスとホスト名	172.16.139.10　node1
現用系ノードのブートアドレスとエイリアス	172.16.139.19　boot1
現用系ノードのスタンバイアドレスとエイリアス	172.16.192.70　stby1
待機系ノードの IP アドレスとホスト名	172.16.139.11　node2

項目	値
待機系ノードのブートアドレスとエイリアス	172.16.139.20　boot2
待機系ノードのスタンバイアドレスとエイリアス	172.16.192.71　stby2
サービスアドレスとエイリアス	172.16.139.16　svc
アプリケーション・サーバ名	docbapsrv
共用ディスクのボリュームグループ	DocBrokerVG
リソース・グループ名	docbresgrp
IP ベース・ネットワークのネットワーク名	netLAN1
非 IP ベース・ネットワークのネットワーク名	netrs1

HACMP を使用したクラスタリングシステム構成例の特徴を次に示します。

- DocumentBroker Server と HiRDB サーバを一つの**アプリケーション・サーバ**（HACMP が起動・終了を実行するアプリケーションの単位）として登録します。
- TPBroker のスマートエージェントは，別ノード上に起動します。
- ノードには，ブートアドレス，スタンバイアドレスを装備します。
- 現用系と待機系のノードは，シリアル・ネットワークで接続します。
- リファレンスファイル管理機能を使用する場合，共有ディスク上でコンテントを管理します。共有ディスク上でコンテントを管理する場合の設定については，「3.16.2　ネットワーク上のマシンの共有ディスクでコンテントを管理する場合の設定」を参照してください。

(2) HACMP による DocumentBroker Server の運用

HACMP を使用したクラスタリングシステムによる運用では，HACMP 環境に DocumentBroker Server および HiRDB サーバを一つのアプリケーション・サーバとして登録します。HACMP を使用したクラスタリングシステムで DocumentBroker Server を運用している場合，次に示す要因の系切り替えを設定できます。

- HACMP による，システム障害（ハードウエア障害，ネットワーク障害など）の検出
- サーバ監視プロセス（EDMDaemon）の消滅
- ユーザによる計画的な系切り替え

なお，サーバ監視プロセスが消滅した場合だけ系を切り替える設定をするためには，アプリケーション・モニター機能を含んだ HACMP/ES をインストールする必要があります。

! 注意事項

サービスプロセス（EDMService）が消滅した場合に系を切り替える設定をしないでください。サービスプロセスの消滅は，DocumentBroker Server 自体が監視しているため，サービスプロセスが障害で消滅した場合は DocumentBroker Server がサービスプロセスを再起動します。

(3) HACMP によるクラスタリングシステムを使用する場合の環境設定の流れ

HACMP によるクラスタリングシステムを使用する場合は，次の流れで環境設定を実行します。

環境設定を実行する前の準備

クラスタリング環境に必要な，共用ディスクの設定やネットワーク構成に関する設定をしてください。

1. DocumentBroker Server の環境設定をします。

2. **始動スクリプト，停止スクリプトおよび監視スクリプトを作成します。**
3. **クラスタを定義します。**
4. **リソースおよびリソース・グループを定義します。**
5. **クラスタを検証します。**

（4） DocumentBroker Server の環境設定

現用系ノードおよび待機系ノードの DocumentBroker Server の環境を設定します。

（a） 注意事項

HACMP によるクラスタリングシステムを運用する場合は，次の点に注意して環境設定を実行してください。

- DocumentBroker Server の環境設定は，クラスタサービスを起動していない状態で実行してください。
- DocumentBroker Server のバージョンをすべてのノードで統一してください。
- DocumentBroker Server は，現用系ノードおよび待機系ノードのローカルディスクにインストールしてください。
- DocumentBroker Server の実行環境は，現用系ノードおよび待機系ノードのローカルディスクに作成してください。
- DocumentBroker Server のインストールディレクトリ，実行環境ディレクトリおよび定義ファイルのパスは，すべてのノードで統一してください。
- DocumentBroker Server の定義ファイルの指定内容は，すべてのノードで統一してください。
  ただし，DocumentBroker Server の環境設定時には，Document Space 構成定義ファイルに定義する HiRDB サーバのホスト名として，現用系ノードと待機系ノードで異なる値を指定して作業する必要があります。環境設定終了後，クラスタサービスを起動する前に，Document Space 構成定義ファイルを更新して，同じ値に変更してください。
  図 B-1 のシステム構成の場合に，それぞれの時点で指定する値を次に示します。

  環境設定時
  現用系ノードの HiRDB サーバのホスト名：node1
  待機系サーバの HiRDB サーバのホスト名：node2

  クラスタサービス起動時
  現用系ノードおよび待機系ノードの HiRDB サーバのホスト名：svc
  なお，svc はサービスアドレスです。

DocumentBroker Server の環境設定方法の詳細については，「3.　環境設定」を参照してください。前提プログラムおよび関連プログラムの環境設定方法については，それぞれのプログラムのマニュアルを参照してください。

（b） 現用系ノードでの環境設定

現用系ノードの DocumentBroker Server の環境設定を実行する前に，共用ディスクのボリュームグループを活動化してください。共用ディスクのボリュームグループの活動化には，varyonvg コマンドを実行します。

環境設定は，通常の手順で実行してください。詳細については，「3.　環境設定」を参照してください。

環境設定が終了したら，待機系の環境設定に備えて，共用ディスクのボリュームグループを非活動化して

ください。共用ディスクのボリュームグループの非活動化には，varyoffvg コマンドを実行します。

(c) 待機系ノードでの環境設定

待機系ノードの DocumentBroker Server の環境設定を実行する前に，共用ディスクのボリュームグループを活動化してください。共用ディスクのボリュームグループの活動化には，varyonvg コマンドを実行します。

待機系の環境設定は，現用系ノードの実行環境 ID の値に応じて，次の手順で実行してください。

現用系ノードの実行環境 ID が 0 の場合

1. DocumentBroker Server をインストールします。
2. OS の環境設定をします。
3. ユーザ管理機能を使用する場合の設定をします。
4. 前提プログラムの環境設定をします。
5. DocumentBroker Server の実行環境を作成します。
6. 現用系ノードの DocumentBroker Server の実行環境の etc ディレクトリ（$DOCBROKERDIR/etc）下の内容（ファイルおよびディレクトリ）を，待機系の DocumentBroker Server の実行環境の etc ディレクトリ（$DOCBROKERDIR/etc）にコピーします。

現用系ノードの実行環境 ID が 1 ～ 254 の場合

1. DocumentBroker Server をインストールします。
2. OS の環境設定をします。
3. ユーザ管理機能を使用する場合の設定をします。
4. 前提プログラムの環境設定をします。
5. DocumentBroker Server の実行環境を作成します。
6. EDMPrintMeta -F コマンド（メタ情報ファイルの出力）を実行します。
7. EDMRegEnvId -r（DocumentBroker 実行環境登録コマンド）を実行します。
   これによって，待機系ノードの実行環境を現用系ノードの実行環境とは異なる実行環境 ID で登録されます。

環境設定が終了したら，共用ディスクのボリュームグループを非活動化してください。共用ディスクのボリュームグループの非活動化には，varyoffvg コマンドを実行します。

(5) スクリプトの作成

クラスタリングシステムを運用するためには，次のスクリプトを作成する必要があります。

● 始動スクリプト
アプリケーション・サーバを始動するスクリプトです。

● 停止スクリプト
アプリケーション・サーバを停止するスクリプトです。

● 監視スクリプト
サービスを HACMP に監視させて，サービスがダウンした場合にフォールオーバさせるスクリプトです。

これらのスクリプトを作成したら，現用系ノードおよび待機系ノードの次のディレクトリに格納してください。

スクリプトの格納先
DocumentBroker 実行環境ディレクトリ /etc（$DOCBROKERDIR/etc）下

ここでは，それぞれのスクリプトの作成手順と作成例を示します。作成例のスクリプトは，次の表に示す前提条件で作成されています。

表 C-2　スクリプト作成例の前提

項目	値
始動スクリプトファイル名	docb_start
停止スクリプトファイル名	docb_stop
監視スクリプトファイル名	docb_monitor
HiRDB サーバ実行環境ディレクトリ	/home2/DocBroker/docbdb/HiRDB_P
DocumentBroker 実行環境ディレクトリ	/home2/DocBroker/docbsv
SecureWay Directory クライアントライブラリのパス	/usr/lib
OSAGENT_PORT	14555
osagent 稼働ホスト IP アドレス	172.16.139.150
HiRDB サーバのホスト名（サービスアドレスを設定）	svc
HiRDB サーバのポート番号	20555
HiRDB サーバデータベースにアクセスするための ユーザ名 / パスワード	"root"/"root"

（a）始動スクリプト（docb_start）

始動スクリプトは，次の順序で処理を実行するように作成してください。

1. **各プログラムのディレクトリの設定**
2. TPBroker **が使用するポート番号と** IP **アドレスの設定**
3. HiRDB **サーバ関連の環境変数の設定**
4. **環境変数** PATH **および** LIBPATH **の設定**
5. HiRDB **サーバの起動**
6. DocumentBroker Server **の起動**
7. **スクリプトを正常終了コード** 0 **で** exit

始動スクリプトの作成例を次に示します。

始動スクリプトの作成例

```
#!/bin/sh

# 1. 各プログラムのディレクトリの設定
CPPDIR=/usr/vacpp
PDDIR=/home2/DocBroker/docbdb2/HiRDB_P
PDCONFPATH=${PDDIR}/conf
DOCBROKERDIR=/home2/DocBroker/docbsv
HICOMDIR=/opt/hitachi/common
TPDIR=/opt/TPBroker
DADIR=/opt/DABroker
PDCLDIR=/opt/HiRDB_P/client
export CPPDIR PDDIR PDCONFPATH DOCBROKERDIR HICOMDIR TPDIR DADIR PDCLDIR

# SecureWay Directory
# この項目はユーザ認証方式にLDAPを使用する場合に設定する
LDAPRTDIR=/usr/lib
export LDAPRTDIR
# 2. TPBroker使用するポート番号と
```

```
#       osagent稼働ホストIPアドレスの設定
OSAGENT_PORT=14555
OSAGENT_ADDR=172.16.139.150
export OSAGENT_PORT OSAGENT_ADDR

# 3. HiRDBサーバ関連の環境変数の設定
PDHOST=svc
PDNAMEPORT=20555
PDUSER='"root"/"root"'
export PDHOST PDNAMEPORT PDUSER

# 4. 環境変数PATHおよびLIBPATHの設定
#       ユーザ認証方式にLDAPを使用しない場合は
#       ${LDAPRTDIR}の設定は不要
PATH=${DOCBROKERDIR}/bin:${PDDIR}/bin:${PATH}
LIBPATH=${DOCBROKERDIR}/lib:${HICOMDIR}/lib:${TPDIR}/lib:${DADIR}/
lib:${PDDIR}/lib:${PDCLDIR}/lib:${LDAPRTDIR}:${CPPDIR}/lib:${LIBPATH}
export PATH LIBPATH

# 5. HiRDBサーバの起動
$PDDIR/bin/pdstart

# 6. DocumentBroker Serverの起動
$DOCBROKERDIR/bin/EDMStart

# 7. 正常終了コード0でexit
  exit 0
```

(b) 停止スクリプト（docb_stop）

停止スクリプトは，次の順序で処理を実行するように作成してください。

1. **各プログラムのディレクトリの設定**
2. TPBroker **が使用するポート番号と** IP **アドレスの設定**
3. HiRDB **サーバ関連の環境変数の設定**
4. **環境変数** PATH **および** LIBPATH **の設定**
5. DocumentBroker Server **の停止**
6. HiRDB **サーバの停止**
7. **スクリプトを正常終了コード** 0 **で** exit

停止スクリプトの作成例を次に示します。

停止スクリプトの作成例

```
#!/bin/sh

# 1. 各プログラムのディレクトリの設定
CPPDIR=/usr/vacpp
PDDIR=/home2/DocBroker/docbdb2/HiRDB_P
PDCONFPATH=${PDDIR}/conf
DOCBROKERDIR=/home2/DocBroker/docbsv
HICOMDIR=/opt/hitachi/common
TPDIR=/opt/TPBroker
DADIR=/opt/DABroker
PDCLDIR=/opt/HiRDB_P/client
export CPPDIR PDDIR PDCONFPATH DOCBROKERDIR HICOMDIR TPDIR DADIR PDCLDIR

# SecureWay Directory
# この項目はユーザ認証方式にLDAPを使用する場合に設定する
LDAPRTDIR=/usr/lib
export LDAPRTDIR
# 2. TPBrokerが使用するポート番号と
#       osagent稼働ホストIPアドレスの設定
OSAGENT_PORT=14555
```

```
OSAGENT_ADDR=172.16.139.150
export OSAGENT_PORT OSAGENT_ADDR

# 3. HiRDBサーバ関連の環境変数の設定
PDHOST=svc
PDNAMEPORT=20555
PDUSER='"root"/"root"'
export PDHOST PDNAMEPORT PDUSER

# 4. 環境変数PATHおよびLIBPATHの設定
# ユーザ認証方式にLDAPを使用しない場合は${LDAPRTDIR}の設定は不要
PATH=${DOCBROKERDIR}/bin:${PDDIR}/bin:${PATH}
LIBPATH=${DOCBROKERDIR}/lib:${HICOMDIR}/lib:${TPDIR}/lib:${DADIR}/
lib:${PDDIR}/lib:${PDCLDIR}/lib:${LDAPRTDIR}:${CPPDIR}/lib:${LIBPATH}
export PATH LIBPATH

# 5. DocumentBroker Serverの停止
$DOCBROKERDIR/bin/EDMStop

# 6. HiRDBサーバの停止
$PDDIR/bin/pdstop -f -q

# 7. 正常終了コード0でexit
  exit 0
```

(c) 監視スクリプト（docb_monitor）

監視スクリプトは，次の順序で処理を実行するように作成してください。

1. SIGTERM **受け付け時にスクリプトを終了するように動作変更します。**
2. DocumentBroker **実行環境ディレクトリを設定します。**
3. DocumentBroker **実行環境ディレクトリから実行されたサーバ監視プロセス（**EDMDaemon**）が存在しているかを確認して，サーバ監視プロセスが消滅していたら** 0 **以外の終了コードで** exit **します。サーバ監視プロセスが存在していたら正常終了コード** 0 **で** exit **します。**

監視スクリプトの作成例を次に示します。なお，ここでは，サーバ監視プロセス（EDMDaemon）が消滅していた場合の終了コードを 255 としています。

監視スクリプトの作成例

```
#!/bin/sh

# 1. SIGTERM受け付け時にスクリプトを終了するよう動作変更
trap exit SIGTERM

# 2. DocumentBroker実行環境ディレクトリの設定
DOCBROKERDIR=/home2/DocBroker/docbsv
export DOCBROKERDIR

# 3. DocumentBroker実行環境ディレクトリから実行された
#       EDMDaemonプロセスが存在しているかを確認して，
#       EDMDaemonプロセスが消滅していたら0以外の終了コード
#       （ここでは255）でexit。
#       存在していたら正常終了コード0でexit。
STATUS=`ps -ef | grep ${DOCBROKERDIR} | grep EDMDaemon | grep -v grep | wc -l`
if [ ${STATUS} -ne 1 ]
then
    exit 255
else
    exit 0
fi
```

(6) クラスタの定義

ここでは，DocumentBroker Server をクラスタとして定義する方法について説明します。

クラスタは，smit コマンドを使用して定義します。smit コマンドについての詳細については，HACMP のマニュアルを参照してください。

クラスタの定義は，現用系ノードまたは待機系ノードのどちらか一方で，すべての手順を実行します。どちらかのノードで実行したあとで，定義の同期を取ることで，もう一方のノードに定義情報を反映します。

以降の説明では，現用系ノードで定義を実行して，定義終了後に待機系ノードの同期を取る手順について説明します。各手順内で，**太字**で示した個所は，入力または選択する必要がある項目です。なお，説明中の値は，図 B-1 のシステムの場合の指定値です。ご使用の環境に合わせて入力または選択してください。

なお，各手順で設定または選択する項目の詳細については，HACMP のマニュアルを参照してください。

(a) クラスタ定義の追加（現用系ノードで実行）

DocumentBroker Server をクラスタとして定義する方法について説明します。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit cm_config_cluster.add
   ```
   クラスタ定義を追加するための画面が表示されます。
2. **「クラスター** ID**」および「クラスター名」を指定します。**
   次のように指定します。

項目	値
クラスター ID	[**1001**]
クラスター名	[**docbcluster**]

3. ENTER **キーを押します。**
   クラスタ定義が追加されます。

(b) クラスタノードの追加（現用系ノードで実行）

クラスタ定義にノードを追加する方法について説明します。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit cm_config_nodes.add
   ```
   クラスタノードを追加するための画面が表示されます。
2. **「ノード名」を指定します。**
   次のように指定します。

項目	値
ノード名	[**node1 node2**]

3. ENTER **キーを押します。**
   クラスタノードが追加されます。

(c) IP ベース・ネットワークの追加（現用系ノードで実行）

IP ベース・ネットワークを追加する方法について説明します。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit cm_config_nets_add_net
   ```
   IP ベース・ネットワークを追加するための画面が表示されます。

2. **「ネットワーク名」,「ネットワークタイプ」,「サブネット」を指定します。**
   次のように指定します。

項目	値
ネットワーク名	[**netLAN1**]
ネットワーク属性	共用
ネットワーク・タイプ	[**ether**]
サブネット	[**172.16.192.0/24 172.16.139.0/24**]

3. ENTER **キーを押します。**
   IP ベース・ネットワークが設定されます。

(d) 非 IP ベース・ネットワークの追加（現用系ノードで実行）

非 IP ベース・ネットワークを追加する方法について説明します。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit config_nets_add_nonip.cmdhdr
   ```
   非 IP ベース・ネットワークを追加するための画面が表示されます。

2. **「ネットワーク名」と「ネットワークタイプ」を指定します。**
   次のように指定します。

項目	値
ネットワーク名	[**netrs1**]
ネットワーク・タイプ	[**rs232**]

3. ENTER **キーを押します。**
   非 IP ベース・ネットワークが設定されます。

(e) IP ベース・アダプターの追加（現用系ノードで実行）

IP ベース・アダプターを追加します。ここでは，現用系ノードで実行します。

追加するアダプターは，ブートアダプター，スタンバイアダプターおよびサービスアダプターです。これらのアダプターを，現用系ノードおよび待機系ノードに対して追加します。

● 現用系ノードにブートアダプターを追加

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit cm_config_ads_add.select
   ```
   新規アダプターを追加するネットワークを選択するための画面が表示されます。

2. **アダプターを追加するネットワークを選択します。**
   ここでは，「(c) IP ベース・ネットワークの追加（現用系ノードで実行）」で追加したネットワークを選択します。
   次のネットワークを選択します。
   netLAN1　(172.16.192.0　172.16.139.0)

3. ENTER **キーを押します。**
   IP ベース・アダプターを追加するための画面が表示されます。

4. **「アダプター** IP **ラベル」,「アダプターの機能」,「アダプター** IP **アドレス」,「ノード名」および「**Netmask**」を指定します。**

次のように指定します。

項目	値
アダプター IP ラベル	[**boot1**]
ネットワーク・タイプ	ether
ネットワーク名	netLAN1
アダプターの機能	[**ブート**]
アダプター IP アドレス	[**172.16.139.19**]
アダプターのハードウェア・アドレス	[]
ノード名	[**node1**]
Netmask	[**255.255.255.0**]

5. ENTER **キーを押します。**
   現用系ノードにブートアダプターが追加されます。

● 待機系ノードにブートアダプターを追加

1. **次の形式で** smit **コマンドを実行します。**
   `smit cm_config_ads_add.select`
   新規アダプターを追加するネットワークを選択するための画面が表示されます。
2. **アダプターを追加するネットワークを選択します。**
   ここでは,「(c) IP ベース・ネットワークの追加（現用系ノードで実行)」で追加したネットワークを選択します。
   次のネットワークを選択します。
   `netLAN1   ( 172.16.192.0  172.16.139.0 )`
3. ENTER **キーを押します。**
   IP ベース・アダプターを追加するための画面が表示されます。
4. **「アダプター** IP **ラベル」,「アダプターの機能」,「アダプター** IP **アドレス」,「ノード名」および「**Netmask**」を指定します。**
   次のように指定します。

項目	値
アダプター IP ラベル	[**boot2**]
ネットワーク・タイプ	ether
ネットワーク名	netLAN1
アダプターの機能	[**ブート**]
アダプター IP アドレス	[**172.16.139.20**]
アダプターのハードウェア・アドレス	[]
ノード名	[**node2**]
Netmask	[**255.255.255.0**]

5. ENTER **キーを押します。**
   待機系ノードにブートアダプターが追加されます。

● 現用系ノードにスタンバイアダプターを追加

1. **次の形式で** smit **コマンドを実行します。**

```
smit cm_config_ads_add.select
```

新規アダプターを追加するネットワークを選択するための画面が表示されます。

2. **アダプターを追加するネットワークを選択します。**
ここでは,「(c) IP ベース・ネットワークの追加（現用系ノードで実行)」で追加したネットワークを選択します。
次のネットワークを選択します。
netLAN1　(172.16.192.0　172.16.139.0)

3. ENTER **キーを押します。**
IP ベース・アダプターを追加するための画面が表示されます。

4. **「アダプター** IP **ラベル」,「アダプターの機能」,「アダプター** IP **アドレス」,「ノード名」および「**Netmask**」を指定します。**
次のように指定します。

項目	値
アダプター IP ラベル	[**stby1**]
ネットワーク・タイプ	ether
ネットワーク名	netLAN1
アダプターの機能	[**スタンバイ**]
アダプター IP アドレス	[**172.16.192.70**]
アダプターのハードウェア・アドレス	[]
ノード名	[**node1**]
Netmask	[**255.255.255.0**]

5. ENTER **キーを押します。**
現用系ノードにスタンバイアダプターが追加されます。

● 待機系ノードにスタンバイアダプターを追加

1. **次の形式で** smit **コマンドを実行します。**
```
smit cm_config_ads_add.select
```
新規アダプターを追加するネットワークを選択するための画面が表示されます。

2. **アダプターを追加するネットワークを選択します。**
ここでは,「(c) IP ベース・ネットワークの追加（現用系ノードで実行)」で追加したネットワークを選択します。
次のネットワークを選択します。
netLAN1　(172.16.192.0　172.16.139.0)

3. ENTER **キーを押します。**
IP ベース・アダプターを追加するための画面が表示されます。

4. **「アダプター** IP **ラベル」,「アダプターの機能」,「アダプター** IP **アドレス」,「ノード名」および「**Netmask**」を指定します。**
次のように指定します。

項目	値
アダプター IP ラベル	[**stby2**]
ネットワーク・タイプ	ether

項目	値
ネットワーク名	netLAN1
アダプターの機能	[**スタンバイ**]
アダプター IP アドレス	[**172.16.192.71**]
アダプターのハードウェア・アドレス	[]
ノード名	[**node2**]
Netmask	[**255.255.255.0**]

5. ENTER **キーを押します。**
   待機系ノードにスタンバイアダプターが追加されます。

● サービスアダプターを追加

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit cm_config_ads_add.select
   ```
   新規アダプターを追加するネットワークを選択するための画面が表示されます。

2. **アダプターを追加するネットワークを選択します。**
   ここでは,「(c) IP ベース・ネットワークの追加（現用系ノードで実行)」で追加したネットワークを選択します。
   次のネットワークを選択します。
   ```
   netLAN1   ( 172.16.192.0  172.16.139.0 )
   ```

3. ENTER **キーを押します。**
   IP ベース・アダプターを追加するための画面が表示されます。

4. **「アダプター** IP **ラベル」,「アダプター** IP **アドレス」および「**Netmask**」を指定します。**
   次のように指定します。

項目	値
アダプター IP ラベル	[**svc**]
ネットワーク・タイプ	ether
ネットワーク名	netLAN1
アダプターの機能	[サービス]
アダプター IP アドレス	[**172.16.139.16**]
アダプターのハードウェア・アドレス	[]
ノード名	[]
Netmask	[**255.255.255.0**]

5. ENTER **キーを押します。**
   サービスアダプターが追加されます。

(f)　非 IP **ベース・アダプターの追加（現用系ノードで実行）**

非 IP ベース・アダプターを追加します。ここでは，現用系ノードで実行します。

非 IP ベース・アダプターは，現用系ノードおよび待機系ノードに対して追加します。

● 現用系ノードに非 IP **ベース・アダプターを追加**

1. **次の形式で** smit **コマンドを実行します。**

```
smit cm_config_ads_add_nonip.select
```

新規アダプターを追加するネットワークを選択するための画面が表示されます。

2. **アダプターを追加するネットワークを選択します。**
   ここでは,「(d) 非 IP ベース・ネットワークの追加（現用系ノードで実行)」で追加したネットワークを選択します。
   次のネットワークを選択します。
   netrs1
3. ENTER **キーを押します。**
   非 IP ベース・アダプターを追加するための画面が表示されます。
4. **「アダプター・ラベル」,「デバイス名」および「ノード名」を指定します。**
   次のように指定します。

項目	値
アダプター・ラベル	[**serial1**]
ネットワーク・タイプ	rs232
ネットワーク名	netrs1
デバイス名	[**/dev/tty2**]
ノード名	[**node1**]

5. ENTER **キーを押します。**
   現用系ノードにブートアダプターが追加されます。

● 待機系ノードに非 IP ベース・アダプターを追加

1. **次の形式で** smit **コマンドを実行します。**

```
smit cm_config_ads_add_nonip.select
```

新規アダプターを追加するネットワークを選択するための画面が表示されます。

2. **アダプターを追加するネットワークを選択します。**
   ここでは,「(d) 非 IP ベース・ネットワークの追加（現用系ノードで実行)」で追加したネットワークを選択します。
   次のネットワークを選択します。
   netrs1
3. ENTER **キーを押します。**
   IP ベース・アダプターを追加するための画面が表示されます。
4. **「アダプターラベル」,「デバイス名」および「ノード名」を指定します。**
   次のように指定します。

項目	値
アダプター・ラベル	[**serial2**]
ネットワーク・タイプ	rs232
ネットワーク名	netrs1
デバイス名	[**/dev/tty2**]
ノード名	[**node2**]

5. ENTER **キーを押します。**

待機系ノードに非 IP ベース・アダプターが追加されます。

(g) クラスタ・トポロジーの同期（現用系ノードで実行）

クラスタ・トポロジーを同期させて，現用系ノードと待機系ノードを同期させる方法について説明します。これによって，現用系ノードで実行した定義が待機系ノードに反映されます。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit configchk.dialog
   ```
   クラスタ・トポロジーを同期させるための画面が表示されます。
2. **クラスタ・トポロジーについて指定します。**
   次のようになっていることを確認します。

項目	値
クラスター検証エラーを無視する	[いいえ]
エミュレートまたは実際	[実際]
クラスターの検証をスキップする	[いいえ]

3. ENTER **キーを押します。**
   クラスタ・トポロジーが同期されて，現用系ノードで定義した内容が待機系ノードに反映されます。

(7) リソースおよびリソース・グループの定義

ここでは，クラスタ内のリソースおよびリソース・グループを定義する方法を説明します。

リソースおよびリソース・グループは，smit コマンドを使用して定義します。smit コマンドについての詳細については，HACMP のマニュアルを参照してください。

リソースおよびリソース・グループの定義は，現用系ノードまたは待機系ノードのどちらか一方で，すべての手順を実行します。どちらかのノードで実行したあとで，定義の同期を取ることで，もう一方のノードに定義情報を反映します。

以降の説明では，現用系ノードで定義を実行して，定義終了後に待機系ノードの同期を取る手順について説明します。各手順内で，**太字**で示した個所は，入力または選択する必要がある項目です。なお，説明中の値は，図 B-1 のシステムの場合の指定値です。ご使用の環境に合わせて入力または選択してください。

各手順で設定または選択する項目の詳細については，HACMP のマニュアルを参照してください。

(a) リソース・グループの追加（現用系ノードで実行）

DocumentBroker Server 用のリソース・グループを追加する方法について説明します。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit cm_add_grp
   ```
   リソース・グループを追加するための画面が表示されます。
2. **「リソース・グループ名」，「ノード関係」および「参加ノード名／デフォルト・ノード優先順位」を指定します。**
   次のように指定します。

項目	値
リソース・グループ名	[**docbresgrp**]
ノード関係	**ローテート**※
参加ノード名／デフォルト・ノード優先順位	[**node1 node2**]

注※
「ローテート」または「カスケード」を選択してください。

3. ENTER **キーを押します。**
リソース・グループが追加されます。

(b) アプリケーション・サーバの追加（現用系ノードで実行）

DocumentBroker Server を登録するアプリケーション・サーバを追加する方法について説明します。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
```
smit claddserv.dialog
```
アプリケーション・サーバを追加するための画面が表示されます。

2. **「サーバ名」，「始動スクリプト」および「停止スクリプト」を指定します。**
次のように指定します。

項目	値
サーバ名	[**docbapsrv**]
始動スクリプト	[**/home2/DocBroker/docbsv/etc/docb_start**]
停止スクリプト	[**/home2/DocBroker/docbsv/etc/docb_stop**]

3. ENTER **キーを押します。**
アプリケーション・サーバが追加されます。

(c) リソース・グループの属性変更（現用系ノードで実行）

DocumentBroker Server 用に追加したリソース・グループの属性を変更する方法について説明します。ここでは，現用系ノードで実行します。

1. **次の形式で** smit **コマンドを実行します。**
```
smit cm_cfg_res.select
```
リソース・グループを選択するための画面が表示されます。

2. **リソース・グループを選択します。**
ここでは，「(a) リソース・グループの追加（現用系ノードで実行）」で追加したノードを選択します。
次のリソース・グループを選択します。
docbresgrp

3. ENTER **キーを押します。**
リソース・グループのリソースおよび属性の変更と，表示をするための画面が表示されます。

4. **「サービス** IP **ラベル」，「ボリューム・グループ」および「アプリケーションサーバ」を指定します。**
次のように指定します。

項目	値
リソース・グループ名	docbresgrp
ノード関係	ローテート
参加ノード名／デフォルト・ノード優先順位	node1 node2
動的ノード優先順位	[]
サービス IP ラベル	[**svc**]
ファイルシステム（デフォルトは「すべて」）	[]
ファイルシステムの整合性検査	fsck
ファイルシステムの回復メソッド	シリアル
エクスポートするファイルシステム／ディレクトリー	[]
NFS マウントするファイルシステム／ディレクトリー	[]
NFS マウント用ネットワーク	[]
ボリューム・グループ	[**DocBrokerVG**]
コンカレント・ボリューム・グループ	[]
ロー・ディスク PVID	[]
Connections サービス	[]
Fast Connect サービス	[]
テープ・リソース	[]
アプリケーション・サーバ	[**docbapsrv**]
高可用性通信リンク	[]
その他のデータ	[]
ボリューム・グループの自動インポート	いいえ
インアクティブ・テークオーバーをアクティブにする	いいえ
フォールバックなしカスケードを使用可能にする	いいえ
9333 ディスク・フェンシングをアクティブにする	いいえ
SSA ディスク・フェンシングをアクティブにする	いいえ
IP 構成の前にファイルシステムをマウントする	いいえ

5. ENTER **キーを押します。**
   リソース・グループの属性が変更されます。

（d）アプリケーション・サーバの追加（現用系ノードで実行）

DocumentBroker Server のサービスを HACMP に監視させて，サービスがダウンした時にフェールオーバさせるためのモニターを追加する方法について説明します。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit clappserv_to_custom_monitor.select
   ```
   モニターするアプリケーション・サーバを選択するための画面が表示されます。

2. **モニターするアプリケーション・サーバを選択します。**
   ここでは，「(b) アプリケーション・サーバの追加（現用系ノードで実行）」で追加したアプリケーション・サーバを選択します。

次のアプリケーション・サーバを選択します。
docbapsrv

3. ENTER **キーを押します。**
ユーザ定義アプリケーション・モニターを追加するための画面が表示されます。

4. **「モニター・メソッド」（監視スクリプト），「モニター間隔」，「モニターを停止するシグナル」，「安定化間隔」，「再始動カウント」，「再始動間隔」，「アプリケーション障害時のアクション」を指定します。**
次のように指定します。

項目	値
アプリケーション・サーバ名	docbapsrv
モニター・メソッド	[**/home2/DocBroker/docbsv/etc/docb_monitor**]
モニター間隔	[**30**] ※1
モニターを停止するシグナル	[**15**] ※2
安定化間隔	[**600**] ※3
再始動カウント	[**0**] ※2
再始動間隔	[**0**] ※2
アプリケーション障害時のアクション	[fallover] ※2
通知メソッド	[]
クリーンアップ・メソッド	[/home2/DocBroker/docbsv/etc/**docb**_stop]
再始動メソッド	[/home2/DocBroker/docbsv/etc/**docb**_start]

注※ 1
「10」秒以上を指定してください。

注※ 2
必ずこの値を指定してください。

注※ 3
「60」秒以上を指定してください。

5. ENTER **キーを押します。**
アプリケーション・モニターが追加されます。

(e) クラスタ・リソースの同期化

クラスタ・リソースについて，現用系ノードと待機系ノードを同期させる方法について説明します。これによって，現用系ノード実行した定義が待機系ノードに反映されます。

1. **次の形式で** smit **コマンドを実行します。**
```
smit clsyncnode.dialog
```
クラスタ・リソースを同期させるための画面が表示されます。

2. **クラスタ・リソースの同期化について指定します。**
次のようになっていることを確認します。

項目	値
クラスター検証エラーを無視する	[いいえ]
クラスター・リソースを構成／構成解除する	[はい]

項目	値
エミュレートまたは実際	[実際]
クラスターの検証をスキップする	[いいえ]

3. ENTER **キーを押します。**
   クラスタ・リソースが同期されて，現用系ノードで定義した内容が待機系ノードに反映されます。

(8) クラスタの検証

クラスタのトポロジーとリソース構成を検証する方法について説明します。

クラスタは，smit コマンドを使用して検証します。smit コマンドについての詳細については，HACMP のマニュアルを参照してください。

クラスタの検証は，現用系ノードまたは待機系ノードのどちらか一方で，すべての手順を実行します。

ここでは，現用系ノードで定義を実行する手順について説明します。各手順内で，**太字**で示した個所は，入力または選択する必要がある項目です。なお，説明中の値は，図 B-1 のシステムの場合の指定値です。ご使用の環境に合わせて入力または選択してください。

各手順で設定または選択する項目の詳細については，HACMP のマニュアルを参照してください。

1. **次の形式で** smit **コマンドを実行します。**
   ```
   smit clverify
   ```
   実行する項目を選択するための画面が表示されます。
2. **「クラスターの検証」を選択します。**
3. ENTER **キーを押します。**
   クラスタを検証するための画面が表示されます。
4. **クラスタの検証方法について指定します。**
   次のようになっていることを確認します。

項目	値
基本 HACMP 検証メソッド （トポロジー，リソース，両方，どちらでもない）	両方
ユーザー定義の検証メソッド	[]
エラー件数	[]
出力を保管するためのログ・ファイル	[]

5. ENTER **キーを実行します。**
   クラスタのトポロジーおよびリソースの構成の内容が検証されます。

(9) クラスタサービスの起動

ここでは，クラスタサービスの起動方法について説明します。クラスタサービスの起動では，現用系ノードおよび待機系ノードそれぞれのクラスタサービスを起動させて利用できるようにして，DocumentBroker Server を開始します。

クラスタサービスは，smit コマンドを使用して始動します。smit コマンドについての詳細については，HACMP のマニュアルを参照してください。

クラスタサービスの始動は，現用系ノードおよび待機系ノードの両方で実行します。なお，クラスタサー

ビスを始動する前に，共用ディスクのボリュームグループを非活動化してください。

1. **次の形式で smit コマンドを実行します。**
   ```
   smit clstart
   ```
   クラスタサービスを始動するための画面が表示されます。
2. **始動方法を指定します。**
   次のようになっていることを確認します。

項目	値
即時始動，システム再始動時に始動，または両方	**即時**
始動時にメッセージをブロードキャストする	いいえ
クラスター・ロック・サービスを始動する	いいえ
クラスター情報デーモンを始動する	**はい**
Reacquire resources after forced down ?	いいえ

（10）運用時の注意事項

ここでは，HACMP によるクラスタリングシステムを運用する場合の注意事項について説明します。

（a）起動・終了の運用

HACMP によるクラスタリングシステムを運用する場合，DocumentBroker Server を起動・終了は，HACMP を使用して実行してください。

クラスタサービスを使用して DocumentBroker Server を起動した場合に EDMStop コマンドで終了しようとすると，システムが不正な状態になります。これは，サーバ監視プロセス（EDMDaemon）が消滅することで，DocumentBroker Server を登録したアプリケーション・サーバに障害が発生したと HACMP にみなされるためです。

（b）運用コマンド・統計解析ツール・トラブルシュートコマンドの運用

運用コマンド，統計解析ツールおよびトラブルシュートコマンドは，現用系ノードで実行してください。待機系ノードでは，これらのコマンドおよびツールは実行できません。

（c）計画的な系切り替えの運用

計画的な系切り替えは，HACMP によって実行してください。詳細については，HACMP のマニュアルを参照してください。

（11）そのほかの注意事項

HiRDB のクライアントの環境変数 PDISLLVL には，値を設定しないか，または「2」を設定してください。

HiRDB のクライアントの環境変数の詳細については，マニュアル「HiRDB　UAP 開発ガイド」を参照してください。

付録 C.2　クラスタリングシステムでのコマンドの実行

DocumentBroker Server の運用コマンドの実行方法は，「7.　コマンドリファレンス」を参照してください。DocumentBroker Server をクラスタリングシステムで運用している場合の注意事項を次に示します。

- クラスタリングシステムの実行系で DocumentBroker Server を運用している場合は，待機系で運用コ

マンドを実行しないでください。

- クラスタリングシステムの実行系で運用コマンドを実行中に，待機系の DocumentBroker への切り替えが発生した場合は，切り替え先で運用コマンドを再度実行する必要があります。

付録 D　全文検索インデクスに UCS-4 の文字を使用する場合のデータベースの設定

ここでは，全文検索インデクスに UCS-4 の文字を使用する場合のデータベースの設定について説明します。なお，次の条件をすべて満たす場合に，全文検索インデクスに UCS-4 の文字を使用できます。

- UCS-2 の範囲外の文字を使用する（UCS-4 の範囲内の文字を使用する）
- データベースの文字コード種別が UTF-8 である
- データベースのインデクスが差分インデクス以外である

データベースの設定には，HiRDB SQL Executer および HiRDB Text Search Plug-in のコマンドを使用します。また，設定を行う前には DocumentBroker サーバを停止してください。

データベースの設定手順を次に示します。

1. **RD エリアの空き容量が十分にあるかを確認します。**
   インデクスの Unicode Type を UCS-4 に変更するとき，インデクスを格納している RD エリアの空き容量が十分にないと，HiRDB Text Search Plug-in のコマンド「phnmodidx（インデクス情報変更ユティリティ）」が異常終了します。そのため，UCS-4 への移行作業を行う前に RD エリアの空き容量を確認してください。
   UCS-4 用のインデクスを使用するときの RD エリア容量の見積もりについては，マニュアル「HiRDB 全文検索プラグイン HiRDB Text Search Plug-in Version 8」を参照してください。
2. **データベースのバックアップを取得します。**
3. **次のディレクトリのバックアップを取得します。**
   ＜ DocumentBroker サーバの実行環境ディレクトリ＞ /etc
4. **HiRDB SQL Executer のコマンド「TABLEINF（指定した表の情報を表示）」を実行し，インデクスに使用する文書クラスのインデクス識別子および RD エリアを確認します。**

TABLEINF (文書クラス名);

＜実行例＞

文書クラス mdmClass_Document のインデクス識別子と RD エリアを確認する場合

```
>TABLEINF "mdmClass_Document";
  :
Index Type      : PLUGIN(MASTER.NGRAM) INDEX
Index Name      : "mdmClass_Document06"
Index ID        : 0x000301e9(197097)
Column          :
    1  "edmProp_TextIndex"
RDAREA Name     : IN (BLOBAREA10)
Option          : PCTFREE=0
  :
```

1. **HiRDB Text Search Plug-in のコマンド「phnidxls（インデクス情報の取得ユティリティ）」を使用して，インデクスの Unicode Type を確認します。**

phnidxls -d インデクス識別子 -r RD エリア名

＜実行例＞

手順 4. で指定した文書クラス mdmClass_Document の Unicode Type を確認する場合

```
phnidxls -d "mdmClass_Document06" -r BLOBAREA10

Index Type                                      CONCEPT
Concept Terms                                        ON
Unicode Type                                       UCS2
RD Area Size                                    14500 Segments
Index File Using Size                             607 Segments
Free Size of Index File                         10039 Segments( 99.98 %)
Available Work Area Size                          729 Segments
Delete Character                                  OFF
Number of Term                                      0
Number of Term for Increment                        0
Index Size for Increment                            0 Segments
Specified File Size for Sub Index                   0 Segments
Sub Index File Using Size                           0 Segments
Number of Documents for Index                       0
Number of Documents for Sub-Index                   0
Ratio of Condensable Documents                   0.00 %
Ratio of Condensed Index                         0.00 %
Number of No Condensed Index                        0 Segments
Delay Status                             CREATEMODE=0
Number of Documents for Unfinished-Index            0
Size of Delay File                                  0
Size of Original Concept File                     579 Segments
Max Page of Original Concept File                3250 Segments
```

1. HiRDB Text Search Plug-in **のコマンド「**phnmodidx**（インデクス情報変更ユティリティ）」を使用して，インデクスの** Unicode Type **を** UCS-4 **に変更します。**

phnmodidx -d インデクス識別子 -v INDEX_TYPE=UCS4

＜実行例＞

手順 4. で指定した文書クラス mdmClass_Document の Unicode Type を UCS-4 に変更する場合

```
>phnmodidx -d "mdmClass_Document06" -v INDEX_TYPE=UCS4

KFPL28904-I Transaction commit
```

1. HiRDB Text Search Plug-in **の** phnidxls **コマンドを使用して，変更後の** Unicode Type **を確認します。**

phnidxls -d インデクス識別子 -r RD エリア名

＜実行例＞

手順 4. で指定した文書クラス mdmClass_Document の変更後の Unicode Type を確認する場合

```
>phnidxls -d "mdmClass_Document06" -r BLOBAREA10

Index Type                                      CONCEPT
Concept Terms                                        ON
Unicode Type                                       UCS4
RD Area Size                                    14500 Segments
Index File Using Size                             635 Segments
Free Size of Index File                         10039 Segments( 99.98 %)
Available Work Area Size                          727 Segments
Delete Character                                  OFF
Number of Term                                      0
Number of Term for Increment                        0
Index Size for Increment                            0 Segments
Specified File Size for Sub Index                   0 Segments
Sub Index File Using Size                           0 Segments
Number of Documents for Index                       0
Number of Documents for Sub-Index                   0
Ratio of Condensable Documents                   0.00 %
Ratio of Condensed Index                         0.00 %
Number of No Condensed Index                        0 Segments
Delay Status                              CREATEMODE=0
Number of Documents for Unfinished-Index            0
Size of Delay File                                  0
Size of Original Concept File                     579 Segments
Max Page of Original Concept File                3250 Segments
```

注意事項

前記の設定作業を行ったあとに全文検索インデクスを再登録してください。全文検索インデクスを再登録しないと，UCS-2 にはなく UCS-4 だけにある文字は検索対象になりません。

参考

データベースの設定時に HiRDB Text Search Plug-in のコマンドが異常終了した場合，次の対策を実施し，再度設定を行ってください。

1. 手順 2. で作成したバックアップを使用して，データベースを回復します。
2. 手順 3. で作成したバックアップを使用して，DocumentBroker のメタデータを回復します。
3. 異常終了時に出力されたエラーメッセージを参照して，エラーの要因を取り除きます。
4. 手順 4. から再度設定を行います。

付録 E　データベース移行

ここでは，次に示すデータベースを移行する方法について説明します。

● アクセス制御機能を使用するためのデータベース移行

● クラス名やプロパティ名などをデータベース定義の名称に使用するためのデータベース移行

● HiRDB の繰り返し列を使用するためのデータベース移行

● マルチファイル管理機能を使用するためのデータベース移行

● リファレンスファイル管理機能を使用するためのデータベース移行

● File Link 連携機能を使用するためのデータベース移行

なお，データベースを移行するためのコマンドを実行できるのは，システム管理者だけです。これらのコマンドを実行する場合，DocumentBroker の実行環境ディレクトリ下（$DOCBROKERDIR/tools）に格納されているコマンドを使用してください。

付録 E.1　アクセス制御機能を使用するためのデータベース移行

アクセス制御非対応のデータベースから，アクセス制御対応のデータベースに移行するための，移行手順およびデータベース移行ツールについて説明します。

(1) 移行手順

データベースの移行手順について説明します。

1. **データベースのバックアップを作成します。**
   - データベースのエラーによってデータベース移行ツールが異常終了した場合，このバックアップからデータベースを回復します。
2. RD **エリア定義情報ファイルを作成します。**
   - 手順 4. 実行後に出力されるデータベース定義文の RD エリア名をあらかじめ指定しない場合は，この作業は不要です。ただし，手順 4. 実行後，データベース定義文格納ファイル内の RD エリア名を変更する必要があります。
   - RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。
3. HiRDB **のデータベース構成変更ユティリティ（**pdmod**）を実行して，**RD **エリアを拡張します。**
   - アクセス制御用のメタ情報の追加，および定義済みの表に対するアクセス制御用の列の追加によって，RD エリアを拡張する必要があります。
   - HiRDB のデータベース構成変更ユティリティ（pdmod）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。
4. EDMChangeACL **コマンドを実行します。**

   データベース定義文が出力されます。
   - この作業は，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定していることを前提としています。
   - このコマンドを実行する前に，データベースを起動してください。
   - このコマンドの実行中に，ほかのコマンドは実行できません。
   - 手順 2. で RD エリア定義情報ファイルを作成していない場合は，このコマンドの実行後，データベース定義文格納ファイル内の RD エリア名を変更してください。
   - EDMChangeACL コマンドの詳細については，「(2)　EDMChangeACL（アクセス制御機能を使用するためのデータベース移行ツール）の文法」を参照してください。

5. **データベース定義文格納ファイルを入力ファイルとして，HiRDB のデータベース定義ユティリティ（pddef）を実行し，スキーマの定義内容を更新します。**
   手順 4. 実行後に出力されたデータベース定義文を基に，スキーマの定義内容が更新され，アクセス制御機能に対応したデータベースのスキーマが HiRDB の表に格納されます。
   - HiRDB のデータベース定義ユティリティ（pddef）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。

（2）EDMChangeACL（アクセス制御機能を使用するためのデータベース移行ツール）の文法

EDMChangeACL コマンドの詳細について説明します。

機能

アクセス制御非対応のデータベースから，アクセス制御対応のデータベースに移行します。また，メタ情報およびデータベース定義を，次に示す定義項目を追加して更新します。

- アクセス制御用のメタ情報の追加
- 定義済みの表に対するアクセス制御用の列の追加
- 追加されたアクセス制御用の列に対するデータの設定

なお，アクセス制御用プロパティを追加したクラスの一覧は，カレントディレクトリの ChangeACLResult.txt に出力されます。

形式

```
EDMChangeACL   {-w | -u ユーザ識別子}
               〔-f セキュリティ定義ファイル名〕
                 -o データベース定義文格納ファイル名
               〔-r RDエリア定義情報ファイル名〕
```

オプション

-w

DocumentBroker Web Client を使用している環境で，DocumentBroker Web Client に登録されているすべてのオブジェクトの所有者のユーザ識別子を，edmProp_Owner プロパティに設定されている値に変更する場合に指定します。

-u ユーザ識別子

DocumentBroker サーバに登録されているすべてのオブジェクトの所有者のユーザ識別子を，このオプションで指定するユーザ識別子に変更する場合に指定します。使用するユーザ管理機能に応じたユーザ識別子を指定してください。
指定できるユーザ識別子の長さの範囲は 1 ～ 254 バイトです。
なお，このオプションに指定するユーザ識別子に対しては，文字列の長さだけをチェックします。

-f セキュリティ定義ファイル名

セキュリティ定義ファイルを絶対パスで指定します。セキュリティ定義ファイルに定義されている次に示すエントリの値に従って，オブジェクトの所有者，グループおよびすべてのユーザに対するアクセス権が設定されます。

- DefaultACFlagOwner エントリ
- DefaultACFlagGroup エントリ
- DefaultACFlagEveryone エントリ

なお，指定を省略した場合，またはセキュリティ定義ファイルにエントリを定義していない場合は，オブジェクトの所有者，グループおよびすべてのユーザに対してフルコントロール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，「4.3　セキュリティ定義ファイル（docaccess.ini）」を参照してください。

-o データベース定義文格納ファイル名

出力されたデータベース定義文を格納するファイルを絶対パスで指定します。

-r RD エリア定義情報ファイル名

RD エリア定義情報ファイルを絶対パスで指定します。-o オプションで指定したデータベース定義文格納ファイル内の RD エリア名を，このオプションで指定する RD エリア定義情報ファイルの定義内容に従って出力する場合に指定します。指定を省略した場合は，このコマンドの実行後，データベース定義文格納ファイル内の RD エリア名を変更してください。

RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。

(3) エラー発生時の対策手順

データベースのエラーによってデータベース移行ツールが異常終了した場合，次に示す手順に従って対策を実施してください。

1. **移行手順 1. で作成したバックアップから，データベースを回復します。**
2. **エラーメッセージを基に，エラーの要因を取り除きます。**
3. **移行手順 3. から再度実行します。**

付録 E.2　クラス名やプロパティ名などをデータベース定義の名称に使用するためのデータベース移行

データベース定義の名称に，GUID 値を変換した ID を使用するデータベースから，クラス名やプロパティ名を使用するデータベースに移行するための，移行手順およびデータベース移行ツールについて説明します。

(1) 移行手順

データベースの移行手順について説明します。

1. **データベースのバックアップを作成します。**
   - データベースのエラーによってデータベース移行ツールが異常終了した場合，このバックアップからデータベースを回復します。
2. **名称定義ファイルを作成します。**
   - クラス名やプロパティ名などをデータベース定義の名称に使用する場合の規則に従って，クラス名およびプロパティ名が指定されている場合，この作業は不要です。
   - 名称定義ファイルについては，「(3)　名称定義ファイル」を参照してください。
3. **RD エリアを拡張する必要がある場合は，HiRDB のデータベース構成変更ユティリティ（pdmod）を実行して，RD エリアを拡張します。**
   - 表識別子および列名に関するメタ情報およびデータベース定義の変更によって，RD エリアを拡張する必要がある場合は，RD エリアを拡張してください。
   - HiRDB のデータベース構成変更ユティリティについては，マニュアル「HiRDB　コマンドリファレンス」を参照してください。
4. **EDMChangeDBDefName コマンドを実行します。**

   データベース定義文が出力されます。
   - この作業は，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定し

ていることを前提としています。
- このコマンドを実行する前に，データベースを起動してください。
- このコマンドの実行中に，ほかのコマンドは実行できません。
- EDMChangeDBDefName コマンドの詳細については，「(2)　EDMChangeDBDefName（クラス名やプロパティ名などをデータベース定義名称に使用するためのデータベース移行ツール）の文法」を参照してください。

5. **データベース定義文格納ファイルを入力ファイルとして，HiRDB のデータベース定義ユティリティ（pddef）を実行し，スキーマの定義内容を更新します。**

手順 4. 実行後に出力されたデータベース定義文を基に，スキーマの定義内容が更新され，クラス名やプロパティ名などをデータベース定義の名称に使用できるデータベースのスキーマが，HiRDB の表に格納されます。
- HiRDB のデータベース定義ユティリティについては，マニュアル「HiRDB　コマンドリファレンス」を参照してください。

(2) EDMChangeDBDefName（クラス名やプロパティ名などをデータベース定義名称に使用するためのデータベース移行ツール）の文法

EDMChangeDBDefName コマンドの詳細について説明します。

機能

データベース定義の名称に使用する値を，GUID 値を変換した ID から，クラス名やプロパティ名に変更します。

このコマンドを初めて実行する場合，すべての表識別子および列名が変更されます。2 回目以降は，-f オプションに指定した名称定義ファイルに定義している表識別子および列名だけが変更されます。

形式

```
EDMChangeDBDefName　〔-f 名称定義ファイル名〕
                      -o データベース定義文格納ファイル名
                    〔-r 表格納用RDエリア名〕
                    〔-i インデクス格納用RDエリア名〕
```

オプション

-f 名称定義ファイル名

名称定義ファイルを絶対パスで指定します。省略した場合，クラス名が表識別子になり，プロパティ名が列名になります。なお，次の場合は名称定義ファイルを必ず指定してください。
- 表識別子として使用できないクラス名，および列名として使用できないプロパティ名を指定している場合
- このコマンドを実行するのが 2 回目以降の場合

名称定義ファイルについては，「(3)　名称定義ファイル」を参照してください。

-o データベース定義文格納ファイル名

出力されたデータベース定義文を格納するファイルを絶対パスで指定します。

-r 表格納用 RD エリア名

このコマンドを初めて実行する場合，表識別子および列名を格納するメタ情報の表が追加されます。このメタ情報の表を格納するユーザ用 RD エリア名を指定します。指定を省略した場合，メタ情報の表は，定義されている表数が最少の公用 RD エリアに格納されます。

-i インデクス格納用 RD エリア名

このコマンドを初めて実行する場合，メタ情報用のインデクスが追加されます。このメタ情報用のインデクスを格納するユーザ用 RD エリア名を指定します。指定を省略した場合，メタ情報用

のインデクスは，表識別子および列名のメタ情報の表が格納されている RD エリアに格納されます。

(3) 名称定義ファイル

名称定義ファイルには，ユーザ定義のクラスに対応する表識別子，およびユーザ定義のプロパティに対応する列名を定義します。ここでは，名称定義ファイルの記述形式および記述例について説明します。

名称定義ファイルの記述形式

名称定義ファイルの記述形式を次に示します。

```
class=クラス名,DBAlias=表識別子
prop=プロパティ名,DBAlias=列名
```

class= **クラス名** ,DBAlias= **表識別子**

ユーザ定義のクラスに対応する表識別子（1 ～ 28 バイト）を指定します。クラス名を表識別子として使用できない場合は，必ず指定してください。

prop= **プロパティ名** ,DBAlias= **列名**

ユーザ定義のプロパティに対応する列名を指定します。プロパティ名を列名として使用できない場合は，必ず指定してください。

次に示すクラス名やプロパティ名などをデータベース定義の名称に使用する場合の規則に従って，列名を指定してください。規則の詳細については，「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」を参照してください。

- VariableArray 型のプロパティに対応する列名は，1 ～ 14 バイトで指定する。
- VariableArray 型の要素のプロパティに対応する列名は，1 ～ 15 バイトで指定する。
- 上記以外のプロパティに対応する列名は，1 ～ 28 バイトで指定する。

なお，このファイルで，行頭が「;」（セミコロン）または「#」（シャープ）の行はコメントとして処理されます。

名称定義ファイルの記述例

名称定義ファイルの記述例を次に示します。

```
#クラス
class=usrClass_ProblemInformationDocument,DBAlias=CProblemInfoDoc
class=usrClass_QuestionAndAnswerDocument,DBAlias=CQandADoc

#VariableArray型のプロパティ
prop=usrProp_WriterInformation,DBAlias=PWriterInfo

#VariableArray型の要素のプロパティ
prop=usrProp_WriterName,DBAlias=PWriterName

#上記以外のプロパティ
prop=usrProp_ProblemInformationDocument,DBAlias=PProblemInfoDoc
prop=usrProp_QuestionAndAnswerDocument,DBAlias=PQandADoc
```

(4) エラー発生時の対策手順

データベースのエラーによってデータベース移行ツールが異常終了した場合，次に示す手順に従って対策を実施してください。

1. **移行手順 1. で作成したバックアップから，データベースを回復します。**
2. **エラーメッセージを基に，エラーの要因を取り除きます。**
3. **移行手順 3. から再度実行します。**

(5) 注意事項

- メタ情報の初期設定コマンド（EDMInitMeta）実行時に，-u オプションに "DisplayName" を指定した場合，このデータベース移行は実行できません。
- データベース移行前に，edmProp_DBAlias プロパティを使用していなくても，データベース移行後に，EDMAddMeta コマンドで edmProp_DBAlias プロパティを追加できます。
- クラスに対応する表を基にビュー表を作成している場合は，ビュー表を削除してから移行作業を実施してください。
- データベース移行時，インデクス名として使用する値は変更しません。したがって，インデクス名は，GUID 値を変換した ID のままです。

付録 E.3　HiRDB の繰り返し列を使用するためのデータベース移行

VariableArray 型のプロパティの要素の格納先を，HiRDB の別表から HiRDB の繰り返し列に移行するための，移行手順およびデータベース移行ツールについて説明します。

(1) 移行手順

データベースの移行手順について説明します。

1. **データベースのバックアップを作成します。**
   - データベースのエラーによってデータベース移行ツールが異常終了した場合，このバックアップからデータベースを回復します。
2. **移行プロパティ定義ファイルを作成します。**
   - 移行プロパティ定義ファイルについては，「(3)　移行プロパティ定義ファイル」を参照してください。
3. EDMChangeVarray **コマンドを実行します。**
   - この作業は，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定していることを前提としています。
   - このコマンドを実行する前に，データベースを起動してください。
   - このコマンドの実行中に，ほかのコマンドは実行できません。
   - EDMChangeVarray コマンドの詳細については，「(2)　EDMChangeVarray（HiRDB の繰り返し列対応機能を使用するためのデータベース移行ツール）の文法」を参照してください。

(2) EDMChangeVarray（HiRDB の繰り返し列対応機能を使用するためのデータベース移行ツール）の文法

EDMChangeVarray コマンドの詳細について説明します。

機能

移行プロパティ定義ファイルに定義した VariableArray 型のプロパティの格納先を，HiRDB の別表から HiRDB の繰り返し列に移行します。また，移行プロパティ定義ファイルに定義した VariableArray 型のプロパティの最大要素数を取得します。

形式

```
EDMChangeVarray  -f 移行プロパティ定義ファイル名
                 〔-o 最大要素数格納ファイル名〕
```

オプション

-f 移行プロパティ定義ファイル名

移行プロパティ定義ファイルを絶対パスで指定します。移行プロパティ定義ファイルについては，「(3)　移行プロパティ定義ファイル」を参照してください。

-o 最大要素数格納ファイル名

移行プロパティ定義ファイルに指定した VariableArray 型のプロパティの最大要素数を取得するためのオプションです。最大要素数を格納するファイル（最大要素数格納ファイル）を絶対パスで指定します。最大要素数格納ファイルには，-f オプションで指定した移行プロパティ定義ファイルに定義されている，VariableArray 型のプロパティの最大要素数が出力されます。なお，このオプションを指定したときは，移行処理は実行されません。
最大要素数格納ファイルの出力例を次に示します。

```
#maxelements file
prop=usrProp_WriterInfo1,maxelements=90
prop=usrProp_WriterInfo2,maxelements=0
```

(3) 移行プロパティ定義ファイル

移行プロパティ定義ファイルには，移行する VariableArray 型のプロパティのプロパティ名および最大要素数を定義します。ここでは，移行プロパティ定義ファイルの記述形式および記述例について説明します。

移行プロパティ定義ファイルの記述形式

移行プロパティ定義ファイルの記述形式を次に示します。

```
prop=VariableArray型プロパティ名，maxelements=最大要素数
```

prop=VariableArray 型プロパティ名

移行する VariableArray 型のプロパティのプロパティ名（dmaProp_DisplayName プロパティの値）を指定します。

maxelements= 最大要素数

移行する VariableArray 型のプロパティの最大要素数（2 ～ 30,000 の符号なし整数値）を指定します。
なお，EDMChangeVarray コマンドで -o オプションを指定すると，-f オプションで指定した移行プロパティ定義ファイルに定義されている，VariableArray 型のプロパティの最大要素数を取得できます。この場合，maxelements は指定を省略できます。

なお，このファイルで，行頭が「;」（セミコロン）または「#」（シャープ）の行はコメントとして処理されます。また，このファイルを解析して 10 件のエラーを検出した場合，その時点で解析処理を中止します。

移行プロパティ定義ファイルの記述例

移行プロパティ定義ファイルの記述例を次に示します。

```
# Sample file for creating index
prop=usrProp_WriterInfo1,maxelements=100
prop=usrProp_WriterInfo2,maxelements=10
```

(4) エラー発生時の対策手順

データベースのエラーによってデータベース移行ツールが異常終了した場合，次に示す手順に従って対策を実施してください。

1. **移行手順 1. で作成したバックアップから，データベースを回復します。**
2. **エラーメッセージを基に，エラーの要因を取り除きます。**
3. **移行手順 3. から再度実行します。**

付録 E.4　マルチファイル管理機能を使用するためのデータベース移行

マルチファイル管理機能非対応のデータベースから，マルチファイル管理機能対応のデータベースに移行するための，移行手順およびデータベース移行ツールについて説明します。

(1) 移行手順

データベースの移行手順について説明します。

1. **データベースのバックアップを作成します。**
   - データベースのエラーによってデータベース移行ツールが異常終了した場合，このバックアップからデータベースを回復します。
2. RD **エリア定義情報ファイルを作成します。**
   - 次のファイルを基に作成してください。
     /opt/HiEDMS/sample/EDM_RDAREA_MULTIFILE.txt
   - 手順 4. 実行後に出力されるデータベース定義文の RD エリア名をあらかじめ指定しない場合は，この作業は不要です。ただし，手順 4. 実行後，データベース定義文格納ファイル内の RD エリア名を変更する必要があります。
   - RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。
3. RD **エリアを追加する必要がある場合は，**HiRDB **のデータベース構成変更ユティリティ（**pdmod**）を実行して，**RD **エリアを追加します。**
   - マルチファイル格納用の表の追加によって，LOB 用 RD エリアを追加する必要があります。
   - HiRDB のデータベース構成変更ユティリティ（pdmod）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。
4. EDMChangeMultiFile **コマンドを実行します。**
   データベース定義文が出力されます。
   - このコマンドを実行する前に，データベースを起動してください。
   - この作業は，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定していることを前提としています。
   - このコマンドの実行中に，ほかのコマンドは実行できません。
   - 手順 2. で RD エリア定義情報ファイルを作成していない場合は，このコマンドを実行したあと，データベース定義文格納ファイル内の RD エリア名を変更してください。
   - EDMChangeMultiFile コマンドの詳細については，「(2)　EDMChangeMultiFile（マルチファイル管理機能を使用するためのデータベース移行ツール）の文法」を参照してください。
5. **データベース定義文格納ファイルを入力ファイルとして，**HiRDB **のデータベース定義ユティリティ（**pddef**）を実行し，スキーマの定義内容を更新します。**
   手順 4. 実行後に出力されたデータベース定義文を基に，スキーマの定義内容が更新され，マルチファイル管理機能に対応したデータベースのスキーマが HiRDB の表に格納されます。
   - HiRDB のデータベース定義ユティリティ（pddef）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。

(2) EDMChangeMultiFile（マルチファイル管理機能を使用するためのデータベース移行ツール）の文法

EDMChangeMultiFile コマンドの詳細について説明します。

機能

マルチファイル管理機能非対応のデータベースから，マルチファイル管理機能対応のデータベースに移行します。

形式

```
EDMChangeMultiFile -c 一つの文書に格納するファイルの最大数
                    -o データベース定義文格納ファイル名
                   〔-r RDエリア定義情報ファイル名〕
                   〔-u ｛ALL | PARTIAL | NO｝〕
```

オプション

-c **一つの文書に格納するファイルの最大数**

一つの文書に含まれるコンテントとして格納するファイルの最大数を指定します。
指定できる値は 2 ～ 4,096 です。

-o **データベース定義文格納ファイル名**

出力されたデータベース定義文を格納するファイルのパス名を指定します。

-r RD **エリア定義情報ファイル名**

RD エリア定義情報ファイルのパス名を指定します。-o オプションで指定したデータベース定義文格納ファイル内の RD エリア名を，このオプションで指定する RD エリア定義情報ファイルの定義内容に従って出力する場合に指定します。指定を省略した場合は，このコマンドを実行したあと，データベース定義文格納ファイル内の RD エリア名を変更してください。
RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。

-u ｛<u>ALL</u> | PARTIAL | NO｝

コンテント格納用 RD エリアに対して，データベースの更新ログを取得する方式を，次のどれかの文字列で指定します。オプションを省略した場合は，「ALL」が仮定されます。

- ALL
  ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。この場合，ロールバックおよびロールフォワードに必要なデータベースの更新ログが取得されます。
- PARTIAL
  更新前ログ取得モードでユーザ LOB 用 RD エリアを運用する場合に指定します。この場合，ロールバックに必要なデータベースの更新ログが取得されます。
- NO
  ログレスモードでユーザ LOB 用 RD エリアを運用する場合に指定します。この場合，データベースの更新ログは取得されません。

(3) エラー発生時の対策手順

データベースのエラーによってデータベース移行ツールが異常終了した場合，次に示す手順に従って対策を実施してください。

1. **移行手順 1. で作成したバックアップから，データベースを回復します。**
2. **エラーメッセージを基に，エラーの要因を取り除きます。**
3. **移行手順 3. から再度実行します。**

(4) EDM_RDAREA_MULTIFILE.txt の内容

EDM_RDAREA_MULTIFILE.txt の内容を次に示します。

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Area Definition for table                                       ;
;  Format : class=class_name,area=area_name                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[TableArea]
```

```
class=edmClass_ContentTransfers,area=USERAREA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Area Definition for Index                                              ;
;  Format : class=class_name,prop=property_name,{prop=property_name,...};
;                             area=area_name                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[IndexArea]
class=edmClass_ContentTransfers,prop=dmaProp_OIID,area=INDEXAREA
class=edmClass_ContentTransfers,prop=ThisPropertyDescription,area=INDEXAREA
class=edmClass_ContentTransfers,prop=edmProp_Parent,area=INDEXAREA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Area Definition for BLOB Column                                        ;
;  Format : class=class_name,prop=edmProp_Content,area=area_name         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[LobArea]
class=edmClass_ContentTransfers,prop=edmProp_Content,area=BLOBAREA
```

付録 E.5　リファレンスファイル管理機能を使用するためのデータベース移行

リファレンスファイル管理機能非対応のデータベースから，リファレンスファイル管理機能対応のデータベースに移行するための，移行手順およびデータベース移行ツールについて説明します。

(1) 移行手順

データベースの移行手順について説明します。

1. **データベースのバックアップを作成します。**
   - データベースのエラーによってデータベース移行ツールが異常終了した場合，このバックアップからデータベースを回復します。
2. RD **エリア定義情報ファイルを作成します。**
   - 次のファイルを基に作成してください。
     /opt/HiEDMS/sample/EDM_RDAREA_REFFILE.txt
   - 手順 4. 実行後に出力されるデータベース定義文の RD エリア名をあらかじめ指定しない場合は，この作業は不要です。ただし，手順 4. 実行後，データベース定義文格納ファイル内の RD エリア名を変更する必要があります。
   - RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。
3. RD **エリアを追加する必要がある場合は，**HiRDB **のデータベース構成変更ユティリティ（**pdmod**）を実行して，**RD **エリアを追加します。**
   - edmClass_ContentReference クラスの追加または EDMSMETAREGENVID クラスの構成変更によって RD エリアの構成を拡張する必要がある場合は，RD エリアを拡張してください。
   - HiRDB のデータベース構成変更ユティリティ（pdmod）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。
4. EDMChangeRefFile **コマンドを実行します。**
   データベース定義文が出力されます。
   - このコマンドを実行する前に，データベースを起動してください。
   - この作業は，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定していることを前提としています。
   - このコマンドの実行中に，ほかのコマンドは実行できません。
   - 手順 2. で RD エリア定義情報ファイルを作成していない場合は，このコマンドを実行したあと，データベース定義文格納ファイル内の RD エリア名を変更してください。
   - このコマンドは，EDMSMETAREGENVID クラスを使用するため，必ず複数の実行環境機能を使用

できる環境で実行してください。
複数の実行環境機能を使用できる環境かどうかは，DocumentBroker のバージョンが 02-40 以降の場合に，DocumentBroker 実行環境の情報の登録コマンド（EDMRegEnvID）に -l オプションを指定して実行することで確認できます。コマンドを実行すると，複数の実行環境機能を使用できる場合は，登録されている実行環境の情報が表示されます。複数の実行環境機能を使用できない場合はエラーとなり，メッセージ KMBR12307-E が出力されます。DocumentBroker 実行環境の情報の登録コマンドについては，「7.3　コマンドの文法」の「EDMRegEnvId（DocumentBroker 実行環境の情報の登録）」を参照してください。メッセージの詳細については，マニュアル「DocumentBroker Version 3　メッセージ」を参照してください。
複数の実行環境機能を使用できない環境からリファレンスファイル管理機能を使用できる環境へ移行する場合は , 複数の実行環境機能を使用できる環境に移行したあと，このコマンドを実行してください。複数の実行環境機能を構築する方法については，「3.15　複数の実行環境を構築する場合の設定」を参照してください。

- このコマンドは，実行環境識別子が 0 の環境で実行してください。
- EDMChangeRefFile コマンドの詳細については，「(2)　EDMChangeRefFile（リファレンスファイル管理機能を使用するためのデータベース移行ツール）の文法」を参照してください。

5. **データベース定義文格納ファイルを入力ファイルとして，HiRDB のデータベース定義ユティリティ（pddef）を実行し，スキーマの定義内容を更新します。**
手順 4. 実行後に出力されたデータベース定義文を基に，スキーマの定義内容が更新され，リファレンスファイル管理機能に対応したデータベースのスキーマが HiRDB の表に格納されます。
- HiRDB のデータベース定義ユティリティ（pddef）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。

(2) EDMChangeRefFile（リファレンスファイル管理機能を使用するためのデータベース移行ツール）の文法

EDMChangeRefFile コマンドの詳細について説明します。

機能

リファレンスファイル管理機能非対応のデータベースから，リファレンスファイル管理機能対応のデータベースに移行します。

形式

```
EDMChangeRefFile -o データベース定義文格納ファイル名
                 〔-r RDエリア定義情報ファイル名〕
```

オプション

-o データベース定義文格納ファイル名

出力されたデータベース定義文を格納するファイルのパス名を指定します。

-r RD エリア定義情報ファイル名

RD エリア定義情報ファイルのパス名を指定します。-o オプションで指定したデータベース定義文格納ファイル内の RD エリア名を，このオプションで指定する RD エリア定義情報ファイルの定義内容に従って出力する場合に指定します。指定を省略した場合は，このコマンドを実行したあと，データベース定義文格納ファイル内の RD エリア名を変更してください。RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。

(3) エラー発生時の対策手順

データベースのエラーによってデータベース移行ツールが異常終了した場合，次に示す手順に従って対策

を実施してください。

1. **移行手順 1. で作成したバックアップから，データベースを回復します。**
2. **エラーメッセージを基に，エラーの要因を取り除きます。**
3. **移行手順 3. から再度実行します。**

（4） EDM_RDAREA_REFFILE.txt の内容

EDM_RDAREA_REFFILE.txt の内容を次に示します。

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Area Definition for table                                                    ;
;    Format : class=class_name,area=area_name                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[TableArea]
class=edmClass_ContentReference,area=USERAREA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Area Definition for Index                                                    ;
;    Format : class=class_name,prop=property_name,{prop=property_name,...};
;                              area=area_name                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[IndexArea]
class=edmClass_ContentReference,prop=dmaProp_OIID,area=INDEXAREA
class=edmClass_ContentReference,prop=ThisPropertyDescription,area=INDEXAREA
class=edmClass_ContentReference,prop=edmProp_ReferenceType,area=INDEXAREA
class=edmClass_ContentReference,prop=edmProp_Parent,area=INDEXAREA
```

付録 E.6　File Link 連携機能を使用するためのデータベース移行

File Link 連携機能非対応のデータベースから，File Link 連携機能対応のデータベースに移行するための，移行手順およびデータベース移行ツールについて説明します。なお，Linux の場合および TPBroker V5 を使用している場合，File Link 連携機能は使用できません。

（1） 移行手順

データベースの移行手順について説明します。

1. **データベースのバックアップを作成します。**
   - データベースのエラーによってデータベース移行ツールが異常終了した場合，このバックアップからデータベースを回復します。
2. **RD エリア定義情報ファイルを作成します。**
   - 「(4)　RD エリア定義情報ファイルの記述例」を参考にして作成してください。
   - 手順 4. 実行後に出力されるデータベース定義文の RD エリア名をあらかじめ指定しない場合は，この作業は不要です。ただし，手順 4. 実行後，データベース定義文格納ファイル内の RD エリア名を変更する必要があります。
   - RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。
3. **RD エリアを追加する必要がある場合は，HiRDB のデータベース構成変更ユティリティ（pdmod）を実行して，RD エリアを追加します。**
   - HiRDB のデータベース構成変更ユティリティ（pdmod）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。
4. **EDMChangeFileLink コマンドを実行します。**
   データベース定義文が出力されます。
   - このコマンドを実行する前に，データベースを起動してください。

- この作業は，DocumentBroker の実行環境ディレクトリを環境変数「DOCBROKERDIR」に指定していることを前提としています。
- このコマンドの実行中に，ほかのコマンドは実行できません。
- 手順 2. で RD エリア定義情報ファイルを作成していない場合は，このコマンドを実行したあと，データベース定義文格納ファイル内の RD エリア名を変更してください。
- EDMChangeFileLink コマンドの詳細については，「(2)　EDMChangeFileLink（File Link 連携機能を使用するためのデータベース移行ツール）の文法」を参照してください。

5. **データベース定義文格納ファイルを入力ファイルとして，HiRDB のデータベース定義ユティリティ（pddef）を実行し，スキーマの定義内容を更新します。**

   手順 4. 実行後に出力されたデータベース定義文を基に，スキーマの定義内容が更新され，File Link 連携機能に対応したデータベースのスキーマが HiRDB の表に格納されます。
   - HiRDB のデータベース定義ユティリティ（pddef）については，マニュアル「HiRDB　コマンドリファレンス」を参照してください。

(2) EDMChangeFileLink（File Link 連携機能を使用するためのデータベース移行ツール）の文法

EDMChangeFileLink コマンドの詳細について説明します。

機能

File Link 連携機能非対応のデータベースから，File Link 連携機能対応のデータベースに移行します。

形式

```
EDMChangeFileLink  -o データベース定義文格納ファイル名
                  〔-r RDエリア定義情報ファイル名〕
```

オプション

-o データベース定義文格納ファイル名

出力されたデータベース定義文を格納するファイルのパス名を指定します。

-r RD エリア定義情報ファイル名

RD エリア定義情報ファイルのパス名を指定します。-o オプションで指定したデータベース定義文格納ファイル内の RD エリア名を，このオプションで指定する RD エリア定義情報ファイルの定義内容に従って出力する場合に指定します。指定を省略した場合は，このコマンドを実行したあと，データベース定義文格納ファイル内の RD エリア名を変更してください。

RD エリア定義情報ファイルについては，「4.8　RD エリア定義情報ファイル」を参照してください。

(3) エラー発生時の対策手順

データベースのエラーによってデータベース移行ツールが異常終了した場合，次に示す手順に従って対策を実施してください。

1. **移行手順 1. で作成したバックアップから，データベースを回復します。**
2. **エラーメッセージを基に，エラーの要因を取り除きます。**
3. **移行手順 3. から再度実行します。**

(4) RD エリア定義情報ファイルの記述例

RD エリア定義情報ファイルの記述例を次に示します。File Link 連携機能を使用する場合の RD エリア定

義情報ファイルは，/opt/HiEDMS/sample 下の EDM_RDAREA_FILELINK.txt です。

```
[TableArea]
#
# FileLinkを利用するための移行で必要なRDエリア
#
class=edmClass_ContentFileLink,area=USERAREA

[IndexArea]
#
# FileLinkを利用するための移行で必要なRDエリア
#
class=edmClass_ContentFileLink,prop=dmaProp_OIID,area=INDEXAREA
class=edmClass_ContentFileLink,prop=ThisPropertyDescription,area=INDEXAREA
class=edmClass_ContentFileLink,prop=edmProp_Parent,area=INDEXAREA
```

付録 F　システムクラスおよびシステムプロパティの名称定義の規則

メタ情報の初期設定コマンド（EDMInitMeta）の -u オプションで「DisplayName」を指定した場合，システムクラスおよびシステムプロパティのデータベース定義での名称定義が決まっています。ここでは，システムクラスおよびシステムプロパティの dmaProp_DisplayName の値，表識別子，および列名を示します。

付録 F.1　システムクラスの名称定義の規則

システムクラスに対応する dmaProp_DisplayName の値と表識別子を次の表に示します。

表 F-1　システムクラスの名称定義の規則

クラス名および dmaProp_DisplayName の値	表識別子
dmaClass_ConfigurationHistory	dmaClass_ConfigHistory
dmaClass_DirectContainmentRelationship	dmaClass_DCRelationship
dmaClass_ReferentialContainmentRelationship	dmaClass_RCRelationship
dmaClass_Rendition	**マスタレンディションの場合** dmaClass_DocVersion の表識別子と同じ **サブレンディションの場合** dmaClass_Rendition
dmaClass_Reservation	dmaClass_VersionSeries
dmaClass_VersionDescription	dmaClass_VerDescription
edmClass_ComponentDocVersion	edmClass_CompoDocVersion
edmClass_VersionTraceableContainer	edmClass_VTContainer
edmClass_VersionTraceableContainmentRelationship	edmClass_VTCRelationship
edmClass_VersionTracedComponentDocVersion	edmClass_VTCompoDocVersion
edmClass_VersionTracedDocVersion	edmClass_VTDocVersion
• dmaClass_Container • dmaClass_ContentReference • dmaClass_ContentTransfer • dmaClass_DocVersion • dmaClass_VersionSeries • edmClass_ContainerVersion • edmClass_ContentTransfers • edmClass_ContentReference • edmClass_ContentFileLink • edmClass_PublicACL • edmClass_Relationship	クラス名および dmaProp_DisplayName の値と同じ

付録 F.2　システムプロパティの名称定義の規則

システムプロパティに対応する dmaProp_DisplayName の値と表識別子を次の表に示します。

表 F-2　システムプロパティの名称定義の規則

プロパティ名および dmaProp_DisplayName の値	列名
プロパティ名 dmaProp_OIID **dmaProp_DisplayName の値** dmaProp_OIID または OIIDPropertyDescription	dmaProp_OIID
プロパティ名 dmaProp_This **dmaProp_DisplayName の値** dmaProp_This または ThisPropertyDescription	dmaProp_This
dmaProp_ConfigurationHistory	dmaProp_ConfigHistory
dmaProp_CurrentOfSeriesCount	dmaProp_CurrentOfSeriesCnt
dmaProp_CurrentVersionDescription	dmaProp_CurrntVerDesc
edmProp_ACL	• edmProp_ACL_edmProp_Subj • edmProp_ACL_edmProp_Permis
edmProp_HeadVersionDescription	edmProp_HeadVerDescription
edmProp_HeadVTConfigurationHistory	edmProp_HeadVTConfigHistory
dmaProp_PrimaryVersionSeries	dmaProp_PrimVerSeries
edmProp_PrimaryGroupPermission	edmProp_PrimGrpPermission
edmProp_PublicACLIds	edmProp_PACLs_edmProp_ACLElm
edmProp_SACL	• edmProp_SACL_edmProp_Subj • edmProp_SACL_edmProp_Permis
edmProp_TailVersionDescription	edmProp_TailVerDescription
edmProp_TailVTConfigurationHistory	edmProp_TailVTConfigHistory
edmProp_VTConfigurationHistory	edmProp_VTConfigHistory

プロパティ名および dmaProp_DisplayName の値	列名
• dmaProp_ComponentType • dmaProp_ContentLocation • dmaProp_Head • dmaProp_IsCurrentVersion • dmaProp_NewVersion • dmaProp_Parent • dmaProp_ParentContainer • dmaProp_RenditionType • dmaProp_RetrievalName • dmaProp_Reservation • dmaProp_ReservedFor • dmaProp_Tail • dmaProp_Version • dmaProp_VersionSeries • edmProp_ConceptTextIndex • edmProp_ConceptStIndex • edmProp_ConfigurationTable • edmProp_Content • edmProp_ContentIndexStatus • edmProp_ContentLocation • edmProp_DocLength • edmProp_EntityName • edmProp_EveryonePermission • edmProp_HeadVTVersionSeries • edmProp_HeadVTMode • edmProp_Interpretation • edmProp_OwnerId • edmProp_OwnerPermission • edmProp_ParentDocVersion • edmProp_Permission • edmProp_PrimaryGroupId • edmProp_PublicID • edmProp_ReferenceType • edmProp_RelationType • edmProp_RenditionsCount • edmProp_RenditionStatus • edmProp_RequiredClassId • edmProp_Role • edmProp_StIndex • edmProp_Subject • edmProp_SystemID • edmProp_TailVTVersionSeries • edmProp_TailVTMode • edmProp_TextIndex • edmProp_VTMode • edmProp_VTVersionSeries	プロパティ名および dmaProp_DisplayName の値と同じ

また，システムプロパティは，次に示す規則にも従っています。

規則 1

HiRDB の繰り返し列を利用した VariableArray 型のプロパティの列名は，「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」の内容に従って生成します。

規則 2

列名には，_1，_2，... という番号が付く場合があります。この番号の意味は，列名に対応するプロパティが属するクラスが，何番目のクラスとして表に格納されているかを表します。番号が付くプロパ

ティを持つクラスを次に示します。

1. マスタ dmaClass_Rendition クラス
   "_1" が列名の最後に付きます。
2. dmaClass_Reservation クラス
   "_1" が列名の最後に付きます。

詳細については,「4.7.2　サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則」を参照してください。

付録 G　ユーザ作成のアクセスルーチンを使用するための関数

ここでは，ユーザ管理システムのアクセスルーチンとして DocumentBroker が定めている関数の形式および仕様について説明します。DocumentBroker で提供する関数を使用して，運用中のユーザ管理システムにアクセスできるようにアクセスルーチンを作成してください。DocumentBroker が UOC で提供する関数の一覧を次の表に示します。

なお，DocumentBroker では，サンプルとして UOC を使用するための関数の例および UOC ライブラリの make ファイルを提供しています。これらは，"/opt/HiEDMS/sample/libdsuoc" に格納されています。

表 G-1　DocumentBroker が UOC で提供する関数の一覧

関数	機能
dbrinitLibrary	UOC ライブラリの初期化
dbrinitSession	UOC セッションの初期化
dbrAuthenticateUser	ユーザとパスワードのチェック
dbrGetUserInformation	ユーザ情報の取得
dbrFinalizeSession	UOC セッションの破棄
dbrFinalizeLibrary	UOC ライブラリの破棄
dbrSetDocSpaceCharacterSet	文書空間の文字コード種別の設定

各関数の詳細では，次に示す項目について説明します。

機能

関数の概要を記述しています。

形式

関数の概要を記述しています。/* */ 内にコメントを記述しています。

機能説明

関数の機能の詳細を記述しています。

引数

引数について次のように記述しています。

引数（入力）

指定する値についての説明

引数（出力）

指定する領域の説明

指定した領域に設定される値の説明

- 引数は，「形式」で記述した文字列です。
- （入力）は，ユーザが値を指定することを示しています。
- （出力）は，ユーザが，DocumentBroker によって設定される値を格納する領域を指定することを示しています。

戻り値

関数を呼び出して，正常な場合に返される値，およびエラーが発生した場合に返される主な値を列挙しています。

注意事項

関数を使用する場合の注意事項を記述しています。

付録 G.1　dbrinitLibrary

機能

UOC ライブラリの初期化

形式

```
int *dbrinitLibrary(
    char          *pszErrMessage    /* エラーメッセージ          */
)
```

機能説明

UOC ライブラリを初期化します。この関数は，プロセス内で 1 回だけ，UOC の中で最初に発行されます。初期化が正常終了した場合，UOC ライブラリハンドルを返却します。

引数

pszErrMessage（出力）

エラーが発生した場合，エラーの詳細を示すメッセージの格納領域を指定します。メッセージの格納領域は，1,024 バイト以上を確保しておく必要があります。この引数に NULL を指定した場合，詳細を示すメッセージは返却されません。

戻り値

```
non NULL      UOCライブラリハンドル
NULL          初期化に失敗しました
```

注意事項

この関数は 1 回だけ発行されます。しかし，アプリケーションからは，マルチスレッド環境で発行されます。そのため，スレッドセーフにしておく必要があります。

付録 G.2　dbrinitSession

機能

UOC セッションの初期化

形式

```
int *dbrinitSession(
     void          *pvhLibrary,     /* UOCライブラリハンドル  */
     const char    *pszUserName,    /* ログインユーザ名        */
     char          *pszErrMessage   /* エラーメッセージ        */
)
```

機能説明

UOC セッションを初期化します。この関数は，セッション作成時にセッション内で 1 回だけ発行されます。初期化が正常に終了した場合，UOC セッションハンドルを返却します。

引数

pvhLibrary（入力）

ライブラリ固有のハンドルを指定します。このパラメタには dbrinitLibrary() で返却された値を指定してください。

pszUserName（入力）

CdbrSession::Connect() で指定されたユーザ名を指定します。NULL または ""（空文字列）が指定された場合，エラーになります。

pszErrMessage（出力）

エラーが発生した場合，エラーの詳細を示すメッセージの格納領域を指定します。メッセージの

格納領域は，1,024 バイト以上を確保しておく必要があります。この引数に NULL を指定した場合，詳細を示すメッセージは返却されません。

戻り値

```
non NULL        UOCライブラリハンドル
NULL            初期化に失敗しました
```

注意事項

この関数は，マルチスレッド環境で発行されます。そのため，スレッドセーフにしておく必要があります。

付録 G.3　dbrAuthenticateUser

機能

ユーザとパスワードのチェック

形式

```
int dbrAuthenticateUser(
     void          *pvhLibrary,     /* UOCライブラリハンドル   */
     void          *pvhSession,     /* UOCセッションハンドル   */
     const char    *pszUserName,    /* ログインユーザ名         */
     const char    *pszPassword,    /* パスワード               */
     char          *pszErrMessage   /* エラーメッセージ         */
)
```

機能説明

引数で指定されたログインユーザ名とパスワードが正しいかどうかをチェックします。

引数

pvhLibrary（入力）

ライブラリ固有のハンドルを指定します。このパラメタには dbrinitLibrary() で返却された値を指定してください。

pvhSession（入力）

セッションハンドルを指定します。このパラメタには，dbrinitSession() で返却された値を指定してください。

pszUserName（入力）

CdbrSession::Connect() で指定されたユーザ名を指定します。NULL または ""（空文字列）が指定された場合，エラーになります。

pszPassword（入力）

CdbrSession::Connect() で指定されたパスワードを指定します。NULL または ""（空文字列）が指定された場合，エラーになります。

pszErrMessage（出力）

エラーが発生した場合，エラーの詳細を示すメッセージの格納領域を指定します。メッセージの格納領域は，1,024 バイト以上を確保しておく必要があります。この引数に NULL を指定した場合，詳細を示すメッセージは返却されません。

戻り値

```
 0              正常終了
-1              ライブラリハンドルが不正です
-2              セッションハンドルが不正です
-3              pszUserNameがNULLまたは""です
                pszPasswordがNULLまたは""です
-4              指定されたユーザは存在しません
```

```
-5              指定されたパスワードは不正です
-6              認証できません
```

上記以外の値は，UOC によって定義されます

注意事項

この関数は，マルチスレッド環境で発行されます。そのため，スレッドセーフにしておく必要があります。

付録 G.4　dbrGetUserInformation

機能

ユーザ情報の取得

形式

```
int dbrGetUserInformation(
     void          *pvhLibrary,      /* UOCライブラリハンドル    */
     void          *pvhSession,      /* UOCセッションハンドル    */
     const char    *pszUserName,     /* ログインユーザ名          */
     char          **ppszUserId,     /* ユーザ識別子              */
     char          **ppszPriGrpId,   /* プライマリグループ        */
                                     /*          識別子返却領域   */
     char          ***pppszGrpIds,   /* グループ識別子返却領域    */
     int           *piGrpCount,      /* 返却グループ識別子要素数  */
     char          *pszErrMessage    /* エラーメッセージ          */
)
```

機能説明

引数 pszUserName で指定されたログインユーザ名に対応するユーザの情報を返却します。

引数

pvhLibrary（入力）

ライブラリ固有のハンドルを指定します。このパラメタには dbrinitLibrary() で返却された値を指定してください。

pvhSession（入力）

セッションハンドルを指定します。このパラメタには，dbrinitSession() で返却された値を指定してください。

pszUserName（入力）

CdbrSession::Connect() で指定されたユーザ名を指定します。NULL または ""（空文字列）が指定された場合，エラーになります。

ppszUserId（出力）

返却されるユーザ識別子を指すポインタのアドレスを指定します。返却された領域は，関数の使用者が free() を使用して解放してください。

ppszPriGrpId（出力）

返却されるプライマリグループ識別子を指すポインタのアドレスを指定します。返却された領域は，関数の使用者が free() を使用して解放してください。プライマリグループ識別子を使用しない場合は，NULL を指定してください。なお，この領域に返却された識別子は，*pppszGrpIds が指すグループ識別子返却領域に返却されることはありません。また，この引数に NULL 以外を指定した場合，引数 piGrpCount の値が 0 以下であっても，ログインユーザはこの引数に示されるプライマリグループに属しているとみなします。

pppszGrpIds（出力）

返却されるグループ情報を指すポインタのアドレスを指定します。返却された領域は，関数の使用者が free() を使用して解放してください。

piGrpCount（出力）

実際に格納された要素数を格納するための領域を指定します。この引数には，引数 ppszPriGrpId で返却されたグループを含みません。

pszErrMessage（出力）

エラーが発生した場合，エラーの詳細を示すメッセージの格納領域を指定します。メッセージの格納領域は，1,024 バイト以上を確保しておく必要があります。この引数に NULL を指定した場合，詳細を示すメッセージは返却されません。

戻り値

0	正常終了
-1	ライブラリハンドルが不正です
-2	セッションハンドルが不正です
-3	pszUserNameがNULLまたは""です
-4	指定されたユーザは存在しません
-5	引数が不正です

上記以外の値は，UOC によって定義されます

注意事項

この関数は，マルチスレッド環境で発行されます。そのため，スレッドセーフにしておく必要があります。

付録 G.5　dbrFinalizeSession

機能

UOC セッションの破棄

形式

```
int dbrFinalizeSession(
     void          *pvhLibrary,     /* UOCライブラリハンドル   */
     void          *pvhSession,     /* UOCセッションハンドル   */
     char          *pszErrMessage   /* エラーメッセージ         */
)
```

機能説明

引数で指定された UOC セッションハンドルを破棄します。この関数は，セッション破棄時に，セッション内で 1 回だけ発行されます。この関数を発行したあとで UOC セッションハンドルを使用して関数を発行しないでください。

引数

pvhLibrary（入力）

ライブラリ固有のハンドルを指定します。このパラメタには dbrinitLibrary() で返却された値を指定してください。

pvhSession（入力）

セッションハンドルを指定します。このパラメタには，dbrinitSession() で返却された値を指定してください。

pszErrMessage（出力）

エラーが発生した場合，エラーの詳細を示すメッセージの格納領域を指定します。メッセージの格納領域は，1,024 バイト以上を確保しておく必要があります。この引数に NULL を指定した場合，詳細を示すメッセージは返却されません。

戻り値

```
 0          正常終了
-1          ライブラリハンドルが不正です
-2          セッションハンドルが不正です
```

上記以外の値は，UOC によって定義されます

注意事項

この関数は，マルチスレッド環境で発行されます。そのため，スレッドセーフにしておく必要があります。

付録 G.6　dbrFinalizeLibrary

機能

UOC ライブラリの破棄

形式

```
int dbrFinalizeLibrary(
     void          *pvhLibrary,     /* UOCライブラリハンドル    */
     char          *pszErrMessage   /* エラーメッセージ          */
)
```

機能説明

引数で指定された UOC ライブラリハンドルを破棄します。この関数は，プロセス終了時に 1 回だけ，ほかの UOC ライブラリの中で最後に発行されます。この関数を発行したあとで UOC ライブラリハンドルを使用して関数を発行しないでください。

引数

pvhLibrary（入力）

ライブラリ固有のハンドルを指定します。このパラメタには dbrinitLibrary() で返却された値を指定してください。

pszErrMessage（出力）

エラーが発生した場合，エラーの詳細を示すメッセージの格納領域を指定します。メッセージの格納領域は，1,024 バイト以上を確保しておく必要があります。この引数に NULL を指定した場合，詳細を示すメッセージは返却されません。

戻り値

```
 0          正常終了
-1          ライブラリハンドルが不正です
```

上記以外の値は，UOC によって定義されます

注意事項

この関数は 1 回だけ発行されます。しかし，アプリケーションからは，マルチスレッド環境で発行されます。そのため，スレッドセーフにしておく必要があります。

この関数は，DocumentBroker サーバの停止完了とは非同期に，EDMService プロセスの終了時に実行され，この関数が処理を終了しないかぎり，EDMService プロセスは終了しません。

EDMService プロセスが終了していない状態で，DocumentBroker サーバを再起動すると，KMBR03006-E が出力され起動エラーとなります。そのため，DocumentBroker サーバの停止が完了したあと，次にサーバが起動されるまでにこの関数の処理を終了しなければなりません。

付録 G.7　dbrSetDocSpaceCharacterSet

機能

文書空間の文字コード種別の設定

形式

```
int  dbrSetDocSpaceCharacterSet(
     void      *pvhLibrary,             /* UOC ライブラリハンドル    */
     int       iDocSpaceCharacterSet,   /* 文書空間の文字コード種別 */
     char      *pszErrMessage           /* エラーメッセージ          */
)
```

機能説明

文書空間の文字コード種別を設定します。この関数は，プロセス内で 1 回だけ，dbrinitLibrary 関数の実行後に発行されます。

引数

pvhLibrary（入力）

ライブラリ固有のハンドルを指定します。このパラメタには dbrinitLibrary() で返却された値を指定してください。

iDocSpaceCharacterSet（入力）

文書空間で使用する文字コード種別を指定します。次のどちらかの値を指定してください。

- 1：文書空間の文字コード種別として Shift-JIS を使用します。
- 2：文書空間の文字コード種別として UTF-8 を使用します。

pszErrMessage（出力）

エラーが発生した場合，エラーの詳細を示すメッセージの格納領域を指定します。メッセージの格納領域は，1,024 バイト以上を確保しておく必要があります。この引数に NULL を指定した場合，詳細を示すメッセージは返却されません。

戻り値

```
 0          正常終了
-1          ライブラリハンドルが不正です
```

上記以外の値は，UOC によって定義されます

注意事項

- この関数は，プロセス内で 1 回だけ，マルチスレッド環境で発行されます。そのため，スレッドセーフにしておく必要があります。
- 文書空間の文字コード種別の設定が不要な場合は，この関数を省略できます。

付録 H　メタ情報の記述内容

ここでは，メタ情報の記述内容確認コマンド（EDMChkMeta）の実行によって，チェックされる内容について説明します。

付録 H.1　ini ファイルのシンタクスの基本項目

ここでは，メタ情報の記述内容の確認コマンド（EDMChkMeta）の実行によってチェックされるメタ情報のシンタクスについて説明します。

```
<Value>::=
{
<Binary Value> | <Boolean Value> | <DateTime Value> | <Float64 Value> | <Id Value>
|
<Integer32 Value> | <Object Value> | <String Value>
}
<Binary Value>::={ 0x[ (<Hex>)... ] | (空値) }
<Boolean Value>::={ 0 | 1 | (空値) }
<DateTime Value>::={ <YYYY><MM><DD>T<hh><mm><ss>Z | (空値) }
<Float64 Value>::={ [<Sign>] ((<Digit>)..."."(<Digit>)...) [ {d | D | e | E
}[<Sign>](<Digit>)*] | (空値) }
<Id Value>::={ <dmaCorrespondingToGuid> | <GUID> }
<Integer32 Value>::={ (-2,147,483,648から2,147,483,647の範囲の10けたの10進数) | (空
値) }
<Object Value>::={ <Reference> | NULL | (空値) }
<String Value>::={ (Printable ascii code)... | NULL | (空値) }
<Hex>::= ( 0 - 9, a-fの16進数の文字 )
<YYYY>::=( 0000-9999の範囲の4けたの10進数)
<MM>::=( 01-12の範囲の2けたの10進数)
<DD>::=( 01-31の範囲の2けたの10進数)
<hh>::=( 00-23の範囲の2けたの10進数 )
<mm>::=( 00-59の範囲の2けたの10進数 )
<ss>::=( 00-59の範囲の2けたの10進数 )
<Digit>::= ( 0 - 9の10進数の文字 )
<GUID>::={ (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx(8-4-4-4-12)形式で記述するGUID) |
<GUID_NULL> | <dmaCorrespondingToGuid> }
<GUID_NULL>::=00000000-0000-0000-0000-000000000000
<Reference>::={  [<Context>@]<Section> | NULL | (空値) }
```

1. <Binary Value>
   データ型が Binary 型の値を 0x で始まる任意の数値（16 進数）で指定します。指定には，0-9 および a-f（小文字）を使用します。指定可能な値の範囲は 128 バイトで，指定可能な長さは 256 バイトです。（空値）は，値を設定していないことを示します。
2. <Boolean Value>
   データ型が Boolean 型の値を指定します。
   - 1：DMA_TRUE
   - 0：DMA_FALSE
3. <DateTime Value>
   データ型が DateTime 型の値を <YYYY><MM><DD>T<hh><mm><ss>Z の形式で指定します。（空値）は，値を指定しないことを示します。
   - <YYYY>：年を 0000 ～ 9999 の範囲の 4 けたで指定します。
   - <MM>：月を 01 ～ 12 の範囲の 2 けたで指定します。
   - <DD>：日を 01 ～ 31 の範囲の 2 けたで指定します。
   - <hh>：時を 00 ～ 23 の範囲の 2 けたで指定します。
   - <mm>：分を 00 ～ 59 の範囲の 2 けたで指定します。

- <ss>：秒を 00 ～ 59 の範囲の 2 けたで指定します。
- T：年月日と時分秒の区切り文字です。
- Z：DateTime 型の値の終端を示します。

4. <Float64 Value>
   データ型が Float64 型の値を，任意の数値（10 進数）で指定します。指定可能な値の範囲は 1.7E +/- 308 で，指定可能なけた数は 15 けたまでです。（空値）は，値を設定していないことを示します。
5. <Id Value>
   データ型が Id 型の値を，「xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx(8-4-4-4-12)」の形式の GUID で指定します。指定には，0-9 および a-f（小文字）を使用します。
   - <GUID_NULL>：GUID の NULL 値を示します。00000000-0000-0000-0000-000000000000 で指定します。
   - <dmaCorrespondingToGuid>：DMA で予約している「dma」で始まるクラスやプロパティを指定します。
6. <Integer32 Value>
   データ型が Integer32 型の値を任意の数値（10 進数）で指定します。指定可能な値の範囲は -2,147,483,648 から 2,147,483,647 で，指定可能なけた数は 10 けたまでです。（空値）は，値を設定していないことを示します。
7. <Object Value>
   データ型が Object 型の値を記述したオブジェクト定義を示すセクションを <Reference> で指定します。<Reference> の内容を次に示します。
   - <Context>：セクションを記述したファイル名を指定します。

   <Context> を省略した場合
   　指定する <Section> が，<Object Value> を記述する同一ファイルに存在すると仮定します。

   <Context> にファイル名だけ指定
   　カレントディレクトリにあるファイルを，<Context> で指定する値で参照します。ただし，環境変数「XDK_METADATA_ROOTDIR」に絶対パスを指定すると，環境変数「XDK_METADATA_ROOTDIR」で指定する値のディレクトリ下のファイルを <Context> の値で参照します。

   <Context> に絶対パスを指定
   　指定する値のファイルを参照します。

   <Context> に相対パスを指定
   　相対パスで指定するファイルを，<Context> の値で参照します。ただし，環境変数「XDK_METADATA_ROOTDIR」に絶対パスを指定すると環境変数「XDK_METADATA_ROOTDIR」で指定する値 + 相対パスでファイルを <Context> の値で参照します。
   - <Section>：セクション名を，印刷可能な ASCII コードで指定します。また，INI ファイル内で一意になるよう指定してください。
   - @：ファイル名とセクション名の区切り文字です。
   - NULL,（空値）：値を設定していないことを示します。
8. <String Value>
   データ型が String 型の値を，任意の印刷可能な ASCII コードで指定します。
   - NULL：値を指定しないことを示します。
   -（空値）："¥0" を指定することを示します。

付録 H.2　クラス定義

ここでは，クラス定義について説明します。

```
<Class Description>::=
(
<左角括弧><Class Name><右角括弧>
ClassFactory=text=dmaClass_ClassDescription
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ClassDescription
dmaProp_This=obj=Self
dmaProp_DisplayName=text=<Class Name>
dmaProp_DescriptiveText=text=<Descriptive Text>
dmaProp_Ids=obj=<Reference>
dmaProp_SuperclassDescription=obj=<Reference>
dmaProp_SuperclassPropertyCount=int=<Count>
dmaProp_ImmediateSubclassDescriptions=obj=<Reference>
dmaProp_NamePropertyIndex=int=0
dmaProp_PropertyDescriptions=obj=<Reference>
dmaProp_HasIncludeSubclasses=bool=0
dmaProp_HasProperSubclassProperties=bool=0
dmaProp_ProperSubclassPropertyDescriptions=obj=<Reference>
)
<Class Name>::=(<Printable ascii code>)...
<Descriptive Text>::=(<Printable ascii code>)...
<Count>::={ 0-2147483647 }
<左角括弧>::=[
<右角括弧>::=]
```

1. <ClassName>
   定義するクラスの名前を，任意の文字列で指定します。クラスの名前の指定は，次に示す形式に従います。
   - DMA が定義するクラスは，「dmaClass_」で始まります。
   - DocumentBroker が定義するクラスは，「edmClass_」で始まります。
   - アプリケーションが定義するクラスは，「dmaClass_」および「edmClass_」以外の名前とします。
2. <Descriptive Text>
   <ClassName> に指定するクラスの説明を，任意の文字列で記述します。
3. dmaProp_Ids=obj=<Reference>
   <ClassName> クラスのクラス識別子を要素として持つ，dmaClass_ListOfId クラスを記述するセクションを指定します。
4. dmaProp_SuperclassDescription=obj=<Reference>
   <ClassName> クラスのスーパークラスのクラス定義を記述する，dmaClass_ClassDescription クラスのセクションを指定します。スーパークラスが存在しない場合は，「NULL」を指定します。
5. dmaProp_SuperclassPropertyCount=int=<Count>
   定義するクラスが，スーパークラスから継承するプロパティの数を指定します。
   - 指定可能な数値は，0 ～ 2,147,483,647 です。
   - 指定可能なけた数は 10 けたです。
6. dmaProp_ImmediateSubclassDescriptions=obj=<Reference>
   定義するクラスのサブクラスのクラス定義を記述する dmaClass_ClassDescription クラスのセクションを要素として持つ dmaClass_ListOfObject クラスを記述するセクションを指定します。サブクラスが存在しない場合は，要素が空の dmaClass_ListOfObject クラスを記述するセクションを指定します。
7. dmaProp_PropertyDescriptions=obj=<Reference>
   定義するクラスのプロパティの定義を記述する dmaClass_PropertyDescription クラスを要素として持つ dmaClass_ListOfObject クラスを記述するセクションを指定します。dmaClass_ListOfObject クラスを記述するセクションのエントリは次の規則に従う必要があります。

- 定義するプロパティのデータ型と一致する dmaClass_PropertyDescriptionXXXX クラスを記述するセクションを，要素のエントリ値に指定します。
- dmaClass_PropertyDescriptionXXXX クラスを記述するセクションをエントリ値とするエントリの並びは，定義するクラスのプロパティの順序になるので，オブジェクトの定義もこのエントリの並びと一致させる必要があります。

8. dmaProp_ProperSubclassPropertyDescriptions=obj=<Reference>
   要素が空の dmaClass_ListOfObject クラスを記述するセクションを指定します。

付録 H.3　プロパティ定義

ここでは，プロパティ定義について説明します。

```
<Property Description>::=
{
<PropertyDescriptionBinary> | <PropertyDescriptionBoolen> |
<PropertyDescriptionDateTime> |
<PropertyDescriptionFloat64> | <PropertyDescriptionId> |
<PropertyDescriptionInteger32> |
<PropertyDescriptionObject> | <PropertyDescriptionString>
}
```

プロパティ定義についてデータ型ごとに説明します。

(1) Binary 型のプロパティ定義

```
<PropertyDescriptionBinary>::=
(
<左角括弧><Property Name><右角括弧>
ClassFactory=text=dmaClass_PropertyDescriptionBinary
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionBinary
dmaProp_DisplayName=text=<Property Name>
dmaProp_DescriptiveText=text=<Descriptive Text>
dmaProp_Ids=obj=<Reference>
dmaProp_DataType=int=DMA_DATATYPE_BINARY
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultBinary=bin=<Binary Value>
dmaProp_PropertySelectionsBinary=obj=<Reference>
dmaProp_MaximumLengthBinary=int=DMA_NO_MAXIMUM
)
<Property Name>::=(<Printable ascii code>)...
<DMA Cardinality>::=
{
DMA_CARDINALITY_SINGLE | DMA_CARDINALITY_LIST | DMA_CARDINALITY_ENUM | 255
}
```

1. <Property Name>
   定義するプロパティの名前を指定します。プロパティの名前は，次の規則に従います。
   - DMA が定義するプロパティは「dmaProp_」で始まります。
   - DocumentBroker が定義するプロパティは，「edmProp_」で始まります。
   - アプリケーションが定義するプロパティは，「dmaProp_」および「edmProp_」以外の名前とします。

2. <Descriptive Text>

<PropertyName> に指定するプロパティの説明を記述します。

3. dmaProp_Ids=obj=<Reference>
   <PropertyName> プロパティのプロパティ識別子を要素として持つ dmaClass_ListOfId オブジェクトを記述するセクションを指定します。
4. dmaProp_Cardinality=int=<DMA_CARDINALITY>
   プロパティの基本単位を指定します。指定できる型は，scalar 型（DMA_CARDINALITY_SINGLE），List 型（DMA_CARDINALITY_LIST），Enumeration 型（DMA_CARDINALITY_ENUM），および VariableArray 型（255）です。
5. dmaProp_IsSelectable=bool=<Boolean Value>
   プロパティが検索結果として選択可能であるかどうかを示す値を指定します。
   - 1：検索結果の 1 項目として指定できるプロパティです。
   - 0：検索結果の 1 項目として指定できないプロパティです。
6. dmaProp_IsSearchable=bool=<Boolean Value>
   プロパティが，問い合わせの条件に使用可能かどうかを示す値を指定します。
   - 1：問い合わせの条件に指定できるプロパティです。
   - 0：問い合わせの条件に指定できないプロパティです。
7. dmaProp_IsOrderable=bool=<Boolean Value>
   問い合わせでソートするプロパティとして使用可能かどうかを示す値を指定します。DMA1.0 Specification では，dmaProp_IsSelectable の値が「1」でなければ，エントリ値に「1」を指定できません。
   - 1：ソートするプロパティとして問い合わせに指定できます。
   - 0：ソートするプロパティとして問い合わせに指定できません。
8. dmaProp_QueryOperatorDescriptions=obj=<Reference>
   定義するプロパティを問い合わせのオペランドに指定する時，使用できるオペレータの定義を記述する dmaClass_OperatorDescription オブジェクトを要素として持つ dmaClass_ListOfObject を記述するセクションを指定します。使用できるオペレータが存在しない場合，要素が空の dmaClass_ListOfObject を記述するセクションを指定します。
9. dmaProp_IsSystemGenerated=bool=<Boolean Value>
   DocumentBroker が値を設定するかどうかを示す値を指定します。
   - 1：DocumentBroker が値を設定します。
   - 0：アプリケーションが値を設定します。

   DocumentBroker は，この値でのプロパティの制御は実行しませんが，DocumentBroker の提供するプロパティで DocumentBroker が値を設定するプロパティには「1」，そうでない場合は「0」を設定します。
   アプリケーションが，この値でプロパティへの値の設定の制御を実行する場合，アプリケーション側で制御を実行する必要があります。制御を実行しない場合，アプリケーションは，この値に反する操作をしないでください。
10. dmaProp_IsReadOnly=bool=<Boolean Value>
    定義するプロパティの値が，IdmaEditProperties インターフェースのメソッドで更新可能かどうかを示す値を指定します。Cardinality が List，Enumeration，および VariableArray の場合，それぞれのオブジェクトの要素が更新可能かどうかを示す値を指定します。この場合，IdmaEditProperties インターフェースのメソッドは，エントリの値に関係なく，戻り値 DMARC_READ_ONLY でリターンすることを示します。ただし，Enumeration オブジェクトを更新するインターフェースは存在しません。
    Cardinality が Single の場合に示す値とその意味を説明します。
    - 1：IdmaEditProperties インターフェースのメソッドで定義するプロパティの更新ができません。

- 0：IdmaEditProperties インターフェースのメソッドで定義するプロパティの更新ができます。

Cardinality が List，または VariableArray の場合に示す値とその意味を説明します。

- 1：IdmaEditListOfXXXX インターフェース，および IdmaEditVariableArrayOfXXXX インターフェースのメソッドで要素を更新できません。
- 0：IdmaEditListOfXXXX インターフェース，および IdmaEditVariableArrayOfXXXX インターフェースのメソッドで要素を更新できます。

DocumentBroker は，この値での List オブジェクト，VariableArray オブジェクトの制御は実行しません。しかし，DocumentBroker が提供するプロパティで更新を許さないプロパティには「1」を，更新を許すプロパティには「0」を指定します。
アプリケーションは，この値で List オブジェクト，VariableArray オブジェクトの制御を実行する場合，アプリケーション側で制御を実行する必要があります。制御を実行しない場合，アプリケーションは，この値に反する操作をしないでください。

11. dmaProp_IsValueRequired=bool=<Boolean Value>
    定義するプロパティに値が必要かどうかを示す値を指定します。
    Cardinality が Single の場合に示す値と意味を説明します。
    - 1：プロパティは，値を持つ必要があります。
    - 0：プロパティは，値を持つ必要はありません。

    Cardinality が List，Enumeration，または VariableArray の場合に示す値と意味を説明します。
    - 1：1 個以上の要素を持つ必要があります。
    - 0：要素を持つ必要はありません。

    DocumentBroker は，この値でのプロパティの制御を実行しませんが，DocumentBroker が提供するプロパティで値の設定が必要なプロパティには「1」，値の設定が不要なプロパティには「0」を指定します。
    アプリケーションがこの値でプロパティへの値の設定を制御する場合，アプリケーション側で制御を実行する必要があります。制御を実行しない場合，アプリケーションは，この値に反する操作をしないでください。
12. dmaProp_IsHidden=bool=<Boolean Value>
    システム管理者だけに表示可能なプロパティであるかどうかを示す値を指定します。
    - 1：システム管理者だけに表示する制限を持つプロパティです。
    - 0：システム管理者だけに表示する制限を持たないプロパティです。

    DocumentBroker は，この値でのプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには「0」を指定します。
    アプリケーションは，この値でプロパティの制御を実行する場合，アプリケーション側で制御を実行する必要があります。制御を実行しない場合，アプリケーションは，この値に反する操作を実行しないでください。
13. dmaProp_PropertyDefaultBinary=bin=<Binary Value>
    dmaClass_PropertyDescriptionBinary クラスが定義するプロパティにデフォルト値として設定する Binary 型の値を指定します。
    DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
    アプリケーションが値を指定する場合は，オブジェクトの定義に指定する値と一致させる必要があります。
14. dmaProp_PropertySelectionsBinary=obj=<Reference>
    dmaClass_PropertyDescriptionBinary クラスが定義するプロパティに指定可能な Binary 型の値を要素として持つ dmaClass_ListOfBinary クラスを記述したエントリを指定します。すべての値が指定可

能な場合，要素が空の dmaClass_ListOfBinary クラスを記述したエントリを指定します。
DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには，要素が空の dmaClass_ListOfBinary クラスを記述したセクションを指定します。
アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションで制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。

(2) Boolean 型のプロパティ定義

```
<PropertyDescriptionBoolen>::=
(
<左角括弧><Property Name><右角括弧>
ClassFactory=text=dmaClass_PropertyDescriptionBoolean
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionBoolean
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj="<Reference>
dmaProp_DataType=int=DMA_DATATYPE_BOOLEAN
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultBoolean=bool=<Boolean Value>
dmaProp_PropertySelectionsBoolean=obj=<Reference>
)
```

1. dmaProp_PropertyDefaultBoolean=bool=<Boolean Value>
   dmaClass_PropertyDescriptionBoolean クラスが定義するプロパティにデフォルト値として設定する Boolean 型の値を指定します。
   DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「0」を指定します。
   アプリケーションが値を指定する場合，クラスの定義に指定する値と一致させる必要があります。
2. dmaProp_PropertySelectionsBoolean=obj=<Reference>
   dmaClass_PropertyDescriptionBinary クラスが定義するプロパティに指定可能な Boolean 型の値を要素として持つ dmaClass_ListOfBoolean クラスを記述したエントリを指定します。すべての値が指定可能な場合，要素が空の dmaClass_ListOfBoolean クラスを記述したエントリを指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには要素が空の dmaClass_ListOfBoolean クラスを記述したセクションを指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションで制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。

(3) DateTime 型のプロパティ定義

```
<PropertyDescriptionDateTime>::=
(
<左角括弧><Property Name><右角括弧>
ClassFactory=text=dmaClass_PropertyDescriptionDateTime
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionDateTime
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj=<Reference>
dmaProp_DataType=int=DMA_DATATYPE_DATETIME
```

```
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultDateTime=date=<DateTime Value>
dmaProp_PropertySelectionsDateTime=obj=<Reference>
dmaProp_PropertyMinimumDateTime=date=<DateTime Value>
dmaProp_PropertyMaximumDateTime=date=<DateTime Value>
)
```

1. dmaProp_PropertyDefaultDateTime=date=<DateTime Value>
   dmaClass_PropertyDescriptionDateTime クラスが定義するプロパティに，デフォルト値として設定する DateTime 型の値を指定します。
   DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションが値を指定する場合，オブジェクトの定義に指定する値と一致させる必要があります。
2. dmaProp_PropertySelectionsDateTime=obj=<Reference>
   dmaClass_PropertyDescriptionDateTime クラスが定義するプロパティに指定可能な DateTime 型の値を要素として持つ dmaClass_ListOfDateTime クラスを記述したエントリを指定します。すべての値が指定可能な場合，要素が空の dmaClass_ListOfDateTime クラスを記述したエントリを指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，要素が空の dmaClass_ListOfDateTime クラスを記述したセクションを指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
3. dmaProp_PropertyMinimumDateTime=date=<DateTime Value>
   プロパティに指定可能な DateTime 型の最小値を指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
4. dmaProp_PropertyMaximumDateTime=date=<DateTime Value>
   プロパティに指定可能な DateTime 型の最大値を指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。

(4) Float64 型のプロパティ定義

```
<PropertyDescriptionFloat64>::=
(
<左角括弧><Property Name><右角括弧>
ClassFactory=text=dmaClass_PropertyDescriptionFloat64
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionFloat64
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj=<Reference>
dmaProp_DataType=int=DMA_DATATYPE_FLOAT64
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
```

```
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultFloat64=float=<Float64 Value>
dmaProp_PropertySelectionsFloat64=obj=<Reference>
dmaProp_PropertyMinimumFloat64=float=<Float64 Value>
dmaProp_PropertyMaximumFloat64=float=<Float64 Value>
)
```

1. dmaProp_PropertyDefaultFloat64=float=<Float64 Value>
   dmaClass_PropertyDescriptionFloat64 オブジェクトが定義するプロパティにデフォルト値として設定する Float64 型の値を指定します。
   DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションが値を指定する場合，オブジェクトの定義に指定する値と一致させる必要があります。
2. dmaProp_PropertySelectionsFloat64=obj=<Reference>
   dmaClass_PropertyDescriptionFloat64 クラスが定義するプロパティに指定可能な Float64 型の値を要素として持つ dmaClass_PropertyDescriptionFloat64 クラスを記述したエントリを指定します。すべての値が指定可能な場合，要素が空の dmaClass_PropertyDescriptionFloat64 クラスを記述したエントリを指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，要素が空の dmaClass_PropertyDescriptionFloat64 クラスを記述したセクションを指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
3. dmaProp_PropertyMinimumFloat64=float=<Float64 Value>
   プロパティに指定可能な Float64 型の最小値を指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
4. dmaProp_PropertyMaximumFloat64=float=<Float64 Value>
   プロパティに指定可能な Float64 型の最大値を指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。

(5) Id 型のプロパティ定義

```
<PropertyDescriptionId>::=
(
<左角括弧><Property Name><右角括弧>
ClassFactory=text=dmaClass_PropertyDescriptionId
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionId
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj=<Reference>
dmaProp_DataType=int=DMA_DATATYPE_ID
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
```

```
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultId=guid=<GUID>
dmaProp_PropertySelectionsId=obj=<Reference>
)
```

1. dmaProp_PropertyDefaultId=guid=<GUID>
   dmaClass_PropertyDescriptionId クラスが定義するプロパティにデフォルト値として設定する Id 型の値を指定します。
   DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションが値を指定する場合は，クラスの定義に指定する値と一致させる必要があります。
2. dmaProp_PropertySelectionsId=obj=<Reference>
   dmaClass_PropertyDescriptionId クラスが定義するプロパティに指定可能な Id 型の値を要素として持つ dmaClass_PropertyDescriptionId オブジェクトを記述したエントリを指定します。すべての値が指定可能な場合，要素が空の dmaClass_PropertyDescriptionId クラスを記述したエントリを指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，要素が空の dmaClass_PropertyDescriptionId クラスを記述したセクションを指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。

(6) Integer32 型のプロパティ定義

```
<PropertyDescriptionInteger32>::=
(
<左角括弧><Property Name><右角括弧>
"ClassFactory=text=dmaClass_PropertyDescriptionInteger32
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionInteger32
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj=<Reference>
dmaProp_DataType=int=DMA_DATATYPE_INTEGER32
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultInteger32=int=<Integer Value>
dmaProp_PropertySelectionsInteger32=obj=<Reference>
dmaProp_PropertyMinimumInteger32=int=<Integer Value>
dmaProp_PropertyMaximumInteger32=int=<Integer Value>
)
```

1. dmaProp_PropertyDefaultInteger32=int=<Integer Value>
   dmaClass_PropertyDescriptionInteger32 オブジェクトが定義するプロパティにデフォルト値として設定する Integer32 型の値を指定します。
   DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。

アプリケーションが値を指定する場合，オブジェクトの定義に指定する値と一致させる必要があります。

2. dmaProp_PropertySelectionsInteger32=obj=<Reference>
   dmaClass_PropertyDescriptionInteger32 クラスが定義するプロパティに指定可能な Integer32 型の値を要素として持つ dmaClass_PropertyDescriptionInteger32 クラスを記述したエントリを指定します。すべての値が指定可能な場合，要素が空の dmaClass_PropertyDescriptionInteger32 クラスを記述したエントリを指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，要素が空の dmaClass_PropertyDescriptionInteger32 クラスを記述したセクションを指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
3. dmaProp_PropertyMinimumInteger32=int=<Integer Value>
   プロパティに指定可能な Integer32 型の最小値を指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
4. dmaProp_PropertyMaximumInteger32=int=<Integer Value>
   プロパティに指定可能な Integer32 型の最大値を指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，DocumentBroker が提供するプロパティには「空値」を指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。

（7） Object 型のプロパティ定義

```
<PropertyDescriptionObject>::=
(
<左角括弧><Property Name><右角括弧>
ClassFactory=text=dmaClass_PropertyDescriptionObject
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionObject
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj=<Reference>
dmaProp_DataType=int=DMA_DATATYPE_OBJECT
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultObject=obj=<Reference>
dmaProp_PropertySelectionsObject=obj=<Reference>
dmaProp_RequiredClass=obj=<Reference>
dmaProp_ReflectivePropertyId=guid=<GUID>
)
```

1. dmaProp_PropertyDefaultObject=obj=<Reference>
   dmaClass_PropertyDescriptionObject クラスが定義するプロパティにデフォルト値として設定する Object 型の値を指定します。
   DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「NULL」を指定します。

アプリケーションが値を指定する場合は，クラスの定義に指定する値と一致させる必要があります。

2. dmaProp_PropertySelectionsObject=obj=<Reference>
   dmaClass_PropertyDescriptionObject クラスが定義するプロパティに指定可能な Object 型の値を要素として持つ dmaClass_PropertyDescriptionObject クラスを記述したエントリを指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，要素が空の dmaClass_PropertyDescriptionObject クラスを記述したセクションを指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
3. dmaProp_RequiredClass=obj=<Reference>
   プロパティから取得されるクラスのクラスの定義を記述する dmaClass_ClassDescription クラスのセクションを指定します。プロパティの値として持つクラスが決まらない場合，「NULL」を指定します。

   Cardinality が Single の場合

   プロパティの値として指定するクラスのクラスの定義を記述する dmaClass_ClassDescription クラスのセクションを指定します。

   Cardinality が List，Enumeration，VariableArray の場合

   ListOfObject オブジェクト，EnumerationOfObject オブジェクト，VariableArrayOfObject オブジェクトの要素となるオブジェクトのクラスの定義を記述する dmaClass_ClassDescription クラスのセクションを指定します。

   DocumentBroker は，この値でプロパティに指定可能なオブジェクトかどうかは確認しません。しかし，DocumentBroker が提供するプロパティには上記の場合に従って，セクションを記述します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
4. dmaProp_ReflectivePropertyId=guid=<GUID>
   dmaClass_PropertyDescriptionObject クラスで定義するプロパティから参照する永続オブジェクトが保持するプロパティで，逆にたどって参照元のオブジェクトを参照できるプロパティの識別子を指定します。逆にたどって参照元のオブジェクトを参照できるプロパティが存在しない場合は＜GUID_NULL＞を指定します。

(8) String 型のプロパティ定義

```
<PropertyDescriptionString>::=
(
<左角括弧><Property Name><右角括弧>
ClassFactory=text=dmaClass_PropertyDescriptionString
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_PropertyDescriptionString
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj=<Reference>
dmaProp_DataType=int=DMA_DATATYPE_STRING
dmaProp_Cardinality=int=<DMA_CARDINALITY>
dmaProp_IsSelectable=bool=<Boolean Value>
dmaProp_IsSearchable=bool=<Boolean Value>
dmaProp_IsOrderable=bool=<Boolean Value>
dmaProp_QueryOperatorDescriptions=obj=<Reference>
dmaProp_IsSystemGenerated=bool=<Boolean Value>
dmaProp_IsReadOnly=bool=<Boolean Value>
dmaProp_IsValueRequired=bool=<Boolean Value>
dmaProp_IsHidden=bool=<Boolean Value>
dmaProp_PropertyDefaultString=text=<String Value>
dmaProp_PropertySelectionsString=obj=<Reference>
dmaProp_MaximumLengthString=int=<Integer Value>
)
```

1. dmaProp_PropertyDefaultString=text=<String Value>

dmaClass_PropertyDescriptionString クラスが定義するプロパティにデフォルト値として設定する String 型の値を指定します。
DocumentBroker は，この値でプロパティを初期化しません。したがって，DocumentBroker が提供するプロパティには「NULL」を指定します。
アプリケーションが値を指定する場合，オブジェクトの定義に指定する値と一致させる必要があります。

2. dmaProp_PropertySelectionsString=obj=<Reference>
   dmaClass_PropertyDescriptionString クラスが定義するプロパティに指定可能な String 型の値を要素として持つ dmaClass_PropertyDescriptionString クラスを記述したエントリを指定します。
   DocumentBroker は，この値でプロパティの制御を実行しません。したがって，要素が空の dmaClass_PropertyDescriptionString クラスを記述したセクションを指定します。
   アプリケーションがこの値でプロパティの制御を実行する場合，アプリケーションが制御を実行する必要があります。制御を実行しない場合，アプリケーションはこの値に反する操作をしないでください。
3. dmaProp_MaximumLengthString=int=<Integer Value>
   プロパティに指定可能な String 型の最大長をバイト数で指定します。

付録 H.4　リストオブジェクト

ここでは，リストオブジェクトについて説明します。

```
<List Object>::=
{
<ListOfBinary> | <ListOfBoolean> | <ListOfDateTime> | <ListOfFloat64> |
<ListOfId> | <ListOfInteger32> | <ListOfObject> | <ListOfString>
}
```

リストオブジェクトについてデータ型ごとに説明します。

（1）dmaClass_ListOfBinary オブジェクト

```
<ListOfBinary>::=
(
<左角括弧> <Section> <右角括弧>
ClassFactory=text=dmaClass_ListOfBinary
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfBinary
[ (=bin=<Binary Value>) ... ]
)
```

1. <Section>
   定義するオブジェクトの名前を指定します。ini ファイル内で一意になるよう指定してください。
2. =bin=<Binary Value>
   dmaClass_ListOfBinary オブジェクトの要素の Binary 型の値をエントリ名は指定しないで，区切り文字を「=bin=」として指定します。このエントリを記述しない場合，要素が空の dmaClass_ListOfBinary オブジェクトとなります。

（2）dmaClass_ListOfBoolean オブジェクト

```
<ListOfBoolean>::=
(
<左角括弧> <Section Name><右角括弧>
ClassFactory=text=dmaClass_ListOfBoolean
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfBoolean
[ (=bool=<Boolean Value>) ... ]
)
```

1. =bool=<Boolean Value>
   dmaClass_ListOfBoolean オブジェクトの要素の Boolean 型の値をエントリ名は指定しないで，区切

り文字を「=bool=」として指定します。このエントリを記述しない場合，要素が空の dmaClass_ListOfBoolean オブジェクトとなります。

(3) dmaClass_ListOfDateTime オブジェクト

```
<ListOfDateTime>::=
(
<左角括弧> <Section Name><右角括弧>
ClassFactory=text=dmaClass_ListOfDateTime
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfDateTime
[ (=date=<DateTime Value>) ... ]
)
```

1. =date=<DateTime Value>
   dmaClass_ListOfDateTime オブジェクトの要素の DateTime 型の値をエントリ名は指定しないで，区切り文字を「=date=」として指定します。このエントリを記述しない場合，要素が空の dmaClass_ListOfDateTime オブジェクトとなります。

(4) dmaClass_ListOfFloat64 オブジェクト

```
<ListOfFloat64>::=
(
<左角括弧> <Section Name> <右角括弧>
ClassFactory=text=dmaClass_ListOfFloat64
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfFloat64
[ (=float=<Float64 Value>) ... ]
)
```

1. =float<Float64 Value>
   dmaClass_ListOfFloat64 オブジェクトの要素の Float64 型の値をエントリ名は指定しないで，区切り文字を「=float=」として指定します。このエントリを記述しない場合，要素が空の dmaClass_ListOfFloat64 オブジェクトとなります。

(5) dmaClass_ListOfId オブジェクト

```
<ListOfId>::=
(
<左角括弧> <Section Name> <右角括弧>
ClassFactory=text=dmaClass_ListOfId
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfId
[ (=guid=<Id Value>) ...]
)
```

1. =guid=<Id Value>
   dmaClass_ListOfId オブジェクトの要素の Id 型の値をエントリ名は指定しないで，区切り文字を「=guid=」として指定します。このエントリを記述しない場合，要素が空の dmaClass_ListOfId オブジェクトとなります。

(6) dmaClass_LisfOfInteger32 オブジェクト

```
<ListOfInteger32>::=
(
<左角括弧><Section Name> <右角括弧>
ClassFactory=text=dmaClass_ListOfInteger32
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfInteger32
[ (=int=<Integer32 Value>) ...]
)
```

1. =int=<Integer32 Value>
   dmaClass_ListOfInteger32 オブジェクトの要素の Integer32 型の値をエントリ名は指定しないで，区切り文字を「=int=」として指定します。このエントリを記述しない場合，要素が空の

dmaClass_ListOfInteger32 オブジェクトとなります。

（7） dmaClass_ListOfObject オブジェクト

```
<ListOfObject>::=
(
<左角括弧><Section Name> <右角括弧>
ClassFactory=text=dmaClass_ListOfObject
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfObject
dmaProp_RequiredClass=obj=<Reference>
[ (=obj=<Object Value>) ...]
)
```

1. =obj=<Object Value>
   dmaClass_ListOfObject オブジェクトの要素の Object 型の値をエントリ名は指定しないで，区切り文字を「=obj=」として指定します。このエントリを記述しない場合，要素が空の dmaClass_ListOfObject オブジェクトとなります。

（8） dmaClass_ListOfString オブジェクト

```
<ListOfString>::=
(
<左角括弧> <Section Name> <右角括弧>
ClassFactory=text=dmaClass_ListOfString
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_ListOfString
[ (=text=<String Value>) ...]
)
```

1. =text=<String Value>
   dmaClass_ListOfString オブジェクトの要素の String 型の値をエントリ名は指定しないで，区切り文字を「=text=」として指定します。このエントリを記述しない場合，要素が空の dmaClass_ListOfString オブジェクトとなります。

付録 H.5　QueryOperator クラス定義

ここでは，QueryOperator クラス定義について説明します。

```
<Query Operator Description>::=
(
<左角括弧><Query Operator Description Name><右角括弧>
ClassFactory=text=dmaClass_QueryOperatorDescription
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_QueryOperatorDescription
dmaProp_This=obj=Self
dmaProp_DisplayName=text=(<Printable ascii code>)...
dmaProp_DescriptiveText=text=(<Printable ascii code>)...
dmaProp_Ids=obj=<Reference>
dmaProp_ResultType=int=<DMA Date Type>
dmaProp_IsList=bool=<Boolean Value>
dmaProp_MinimumOperands=int=<Integer32 Value>
dmaProp_MaximumOperands=int=<Integer32 Value>
dmaProp_IsSafeToEliminate=bool=0
dmaProp_OperandDescriptions=obj=<Reference>
dmaProp_JoinParticipation=int=<Join Participation>
)
<DMA Data Type>::=
{
DMA_DATATYPE_BINARY | DMA_DATATYPE_BOOLEAN | DMA_DATATYPE_DATETIME |
DMA_DATATYPE_FLOAT64 | DMA_DATATYPE_ID | DMA_DATATYPE_INTEGER32 |
DMA_DATATYPE_OBJECT | DMA_DATATYPE_STRING | DMA_DATATYPE_CLASS
}
<Join Participation>::=
{
```

```
DMA_JOIN_PARTICIPATION_NONE | DMA_JOIN_PARTICIPATION_OPERAND |
DMA_JOIN_PARTICIPATION_OPERATOR
}
```

1. dmaProp_ResultType=int=<DMA Date Type>
   オペレータの演算結果の値のデータ型を指定します。指定可能な値は，<DMA Data Type> で示される値だけです。
   - DMA_DATATYPE_BINARY：オペレータによる演算結果の値が，Binary 型です。
   - DMA_DATATYPE_BOOLEAN：オペレータによる演算結果の値が Boolean 型です。
   - DMA_DATATYPE_DATETIME：オペレータによる演算結果の値が DateTime 型です。
   - DMA_DATATYPE_FLOAT64：オペレータによる演算結果の値が Float64 型です。
   - DMA_DATATYPE_ID：オペレータによる演算結果の値が Id 型です。
   - DMA_DATATYPE_INTEGER32：オペレータによる演算結果の値が Integer 型です。
   - DMA_DATATYPE_OBJECT：オペレータによる演算結果の値が Object 型です。
   - DMA_DATATYPE_STRING：オペレータによる演算結果の値が String 型です。
   - DMA_DATATYPE_CLASS：Join オペレータの演算結果の値として指定します。
2. dmaProp_IsList=bool=<Boolean Value>
   - 0：オペレータによる演算結果が scalar 型です。
   - 1：オペレータによる演算結果が List 型です。
3. dmaProp_MinimumOperands=int=<Integer32 Value>
   オペレータが要求するオペランドの最小数を指定します。0 以上の値を指定します。
4. dmaProp_MaximumOperands=int=<Integer32 Value>
   オペレータに指定できるオペランドの最大数を指定します。dmaProp_MinimumOperands と dmaProp_MaximumOperands が両方とも 0 の場合，オペレータは，オペランドが不要なことを示します。指定可能なオペランドが任意の個数の場合，DMA_NO_MAXIMUM を指定します。
5. dmaProp_IsSafeToEliminate
   問い合わせに対応しないオペレータです。
6. dmaProp_OperandDescriptions=obj=<Reference>
   オペレータのオペランドを定義する dmaClass_QueryOperandDescription クラスを要素とする dmaClass_ListOfObject クラスを記述するセクションを指定します。オペレータがオペランドを持たない場合，要素が空の dmaClass_ListOfObject クラスを記述するセクションを指定します。dmaClass_ListOfObject クラスの要素である dmaClass_QueryOperandDescription クラスの並びは，オペレータに指定するオペランドの並びと一致させる必要があります。また，すべて同じ定義であるオペランドをオペレータに指定する場合，dmaClass_ListOfObject クラスの要素である dmaClass_QueryOperandDescription クラスは 1 個です。
7. dmaProp_JoinParticipation=int=<Join Participation>
   オペレータが Join 式に参加できる範囲を示す値を指定します。
   - DMA_JOIN_PARTICIPATION_NONE:From 句に指定できないオペレータを示します。通常の dmaClass_QueryOperator オブジェクトが該当します。
   - DMA_JOIN_PARTICIPATION_OPERAND:From 句の Join オペレータの第 3 オペランドである ON 条件に指定できるオペレータを示します。dmaProp_ResultType が DMA_DATATYPE_BOOLEAN で dmaProp_IsList が 0 である必要があります。
   - DMA_JOIN_PARTICIPATION_OPERATOR:Join オペレータを示します。dmaProp_ResultType で DMA_DATATYPE_CLASS および dmaProp_IsList は 0 である必要があります。

付録 H.6　QueryOperand クラス定義

ここでは，QueryOperand クラス定義について説明します。

```
<Query Operand Description>::=
(
<左角括弧>< Query Operand Description Name><右角括弧>
ClassFactory=text=dmaClass_QueryOperandDescription
dmaProp_ClassDescription=obj=dmaclass.ini@dmaClass_QueryOperandDescription
dmaProp_This=obj=Self
dmaProp_OperandDataType=int=<DMA Date Type>
dmaProp_AllowsSingleton=bool=<Boolean Value>
dmaProp_AllowsList=bool=<Boolean Value>
dmaProp_AllowsConstant=bool=<Boolean Value>
dmaProp_AllowsProperty=bool=<Boolean Value>
dmaProp_AllowsExpression=bool=<Boolean Value>
)
```

1. <Query Operand Description Name>
   定義するオペランドの名前を指定します。定義する INI ファイルでオペランド名が一意になるよう指定してください。
2. dmaProp_OperandDataType=int=<DMA Date Type>
   オペレータに指定可能なオペランドのデータ型を指定します。
3. dmaProp_AllowsSingleton=bool=<Boolean Value>
   問い合わせのオペレータに指定するオペランドが単一の値かどうかを指定します。
   - 1：オペランドが単体であることを示します。
   - 0：オペランドが単体ではないことを示します。

   dmaProp_AllowsSingleton プロパティ，dmaProp_AllowsList プロパティの少なくとも一つに「1」を設定します。
4. dmaProp_AllowsList=bool=<Boolean Value>
   問い合わせのオペレータに指定するオペランドがリストかどうかを指定する Boolean 型の値です。
   設定するのは，次の値のどちらかです。
   - 1：オペランドがリストであることを示します。
   - 0：オペランドがリストではないことを示します。

   dmaProp_AllowsSingleton プロパティ，dmaProp_AllowsList プロパティの少なくとも一つに「1」を設定します。
5. dmaProp_AllowsConstant=bool=<Boolean Value>
   問い合わせのオペレータに指定するオペランドが定数かどうかを指定する Boolean 型の値です。
   設定するのは，次の値のどちらかです。
   - 1：オペランドが定数であることを示します。
   - 0：オペランドが定数ではないことを示します。

   dmaProp_AllowsConstant プロパティ，dmaProp_AllowsProperty プロパティ，または dmaProp_AllowsExpression の少なくとも一つに「1」を設定します。
6. dmaProp_AllowsProperty=bool=<Boolean Value>
   問い合わせのオペレータに指定するオペランドがプロパティかどうかを指定する Boolean 型の値です。
   設定するのは，次の値のどちらかです。
   - 1：オペランドがプロパティであることを示します。
   - 0：オペランドがプロパティではないことを示します。

   dmaProp_AllowsConstant プロパティ，dmaProp_AllowsProperty プロパティ，または dmaProp_AllowsExpression の少なくとも一つに「1」を設定します。

7. dmaProp_AllowsExpression=bool=<Boolean Value>
   問い合わせのオペレータに指定するオペランドが検索条件かどうかを指定する Boolean 型の値です。設定するのは，次の値のどちらかです。
   - 1：オペランドが検索条件であることを示します。
   - 0：オペランドが検索条件ではないことを示します。

   dmaProp_AllowsConstant プロパティ，dmaProp_AllowsProperty プロパティ，または dmaProp_AllowsExpression の少なくとも一つに「1」を設定します。

付録 H.7　オブジェクト定義

ここでは，オブジェクト定義について説明します。

```
<Object>::=
(
 <左角括弧><Section Name><右角括弧>
 (<Property>)...
)
<Property>::=<Property Name>=<Data Type>=<Value>
<Data Type>::= { bin | bool | date | float | guid | int | obj | text }
```

1. <Section Name>
   定義するオブジェクトの名前を指定します。INI ファイル内で一意になるように指定してください。
2. <Property Name>
   dmaClass_ListOfObject オブジェクトの要素に記述する dmaClass_PropertyDescriptionXXXX オブジェクトの並び方に従ってエントリを記述します。dmaClass_ListOfObject オブジェクトの要素は，オブジェクトのクラスの定義を記述する dmaClass_ClassDescription クラスのプロパティである dmaProp_PropertyDescriptions プロパティに指定する値で参照されます。プロパティ名は，dmaClass_PropertyDescriptionXXXX オブジェクトの <Property Name> と一致させてください。
3. <DataType>
   <Property Name> に指定するプロパティの定義を記述する dmaClass_PropertyDescriptionXXXX クラスの dmaProp_DataType エントリに指定するデータ型と一致する値を指定します。データ型とそれに対応する値を次に示します。
   - DMA_DATATYPE_BINARY: bin
   - DMA_DATATYPE_BOOL: bool
   - DMA_DATATYPE_DATETIME: date
   - DMA_DATATYPE_FLOAT64: float
   - DMA_DATATYPE_ID: guid
   - DMA_DATATYPE_INTEGER32: int
   - DMA_DATATYPE_OBJECT: obj
   - DMA_DATATYPE_STRING: text

4. <Value>
   <Property Name> に指定するプロパティの定義を記述する dmaClass_PropertyDescriptionXXXX クラスの dmaProp_DataType エントリに指定するデータ型に一致する値を指定します。データ型とそれに対応する値を次に示します。
   - DMA_DATATYPE_BINARY: <Binary Value>
   - DMA_DATATYPE_BOOLEAN: <Boolean Value>
   - DMA_DATATYPE_DATETIME: <DateTime Value>
   - DMA_DATATYPE_FLOAT64: <Float64 Value>
   - DMA_DATATYPE_ID: <Id Value>

- DMA_DATATYPE_INTEGER32: <Integer32 Value>
- DMA_DATATYPE_OBJECT: <Object Value>
- DMA_DATATYPE_STRING: <String Value>

付録 H.8　メタ情報管理用に確保する共用メモリセグメントサイズ

DocumentBroker が提供するメタ情報ファイルではなく，ユーザが作成したファイルに，ユーザが定義するクラスまたはプロパティの定義を記述した場合，共用メモリサイズを次の見積もり式で算出する必要があります。

メタ情報管理用に確保する共用メモリセグメントサイズ（バイト）

＝ 3,000,000 ＋ 10,640 × A ＋ 12,584 × B ＋ 752 × C ＋ 2,084 × D

（凡例）

A

ユーザが定義するクラス数を表します。

B

ユーザが定義するプロパティ数を表します。

C

ユーザが定義するクラスに定義するユーザ定義のプロパティの総数を表します。この値はメタ情報の追加コマンド（EDMAddMeta）に指定する定義情報ファイル内の [AddProperty/ クラス名] の数です。

D

ユーザが作成するメタ情報ファイル数（0 ～ 2,147,482,151）を表します。

付録 I　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機能およびコマンド

ここでは，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機能およびコマンドについて説明します。なお，文書空間で使用する文字コード種別が Shift-JIS の場合は，すべての機能およびコマンドを使用できます。

(1) 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機能

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機能を次の表に示します。

表 I-1　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機能

分類	機能名	使用可否
文書管理	文書の登録機能	○
	バージョン管理機能	○
	マルチレンディション管理機能	○
	マルチファイル管理機能	○
	コンテナ管理機能	○
	文書の構成管理機能	○
	文書間リレーション管理機能	○
	文書の属性情報の管理機能	○
	リファレンスファイル管理機能	○
	レンディションのコンテント種別変換機能	○
	File Link 連携機能	－
構造化文書管理	XML 文書管理機能	－
独立データの管理	独立データの管理機能	○
アクセス制御	アクセス制御機能	○
統計・解析	アクセスログおよびトレースログの統計・解析機能	－
検索	属性検索	○
	全文検索	○
	構造指定検索	－
	概念検索	○
	文字列型プロパティに対する全文検索	○
変換・登録	オブジェクト一括登録機能	○
	全文検索インデクス一括登録機能	○
	レンディション変換機能	○
	イメージ文書の登録機能	－
収集・検索	分散した文書の収集機能	－
	収集した文書の検索機能	－
分析	文書の分析機能	○
ライフサイクル管理	文書のライフサイクル管理機能	－
環境構築	ファイル転送機能	○
	複数の実行環境機能	○

分類	機能名	使用可否
	システム導入支援機能	○

（凡例）
○：使用できます。
－：使用できません。

(2) 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コマンド

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コマンドを次の表に示します。

表 I-2　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コマンド

分類	コマンド名	使用可否
ファイル転送機能で使用するコマンド	FtpSvSetup	○
	FtpSvStart	○
	FtpSvStop	○
システム運用コマンド	EDMRefresher	○
	EDMSetup	○
	EDMStart	○
	EDMStop	○
	EDMUsrView	○
データベース運用コマンド	EDMAddMeta	○
	EDMCreateIds	○
	EDMCrtSimMeta	○
	EDMCrtSql	○
	EDMDelMeta	○
	EDMInitMeta	○
	EDMPrintMeta	○
	EDMRegEnvId	○
	EDMXmlMap	－
システム導入支援コマンド	EDMCDefDocSpace	○
	EDMCBuildDocSpace	○
トラブルシュートコマンド	EDMChkMeta	○
	EDMChkTbl	○
	EDMGetRas	○
	EDMGetRasCL	○
	EDMLckWatcher	○
データベース移行用のコマンド	EDMChangeACL	－
	EDMChangeDBDefName	－
	EDMChangeVarray	－
	EDMChangeMultiFile	－
	EDMChangeRefFile	－
	EDMChangeFileLink	－

分類	コマンド名	使用可否
統計解析ツールのコマンド※	EDMRptAclog	－
	EDMRptTrace	－

（凡例）

○：使用できます。

－：使用できません。

注※

コマンドの詳細は，マニュアル「DocumentBroker Version 3　統計解析ツール」を参照してください。

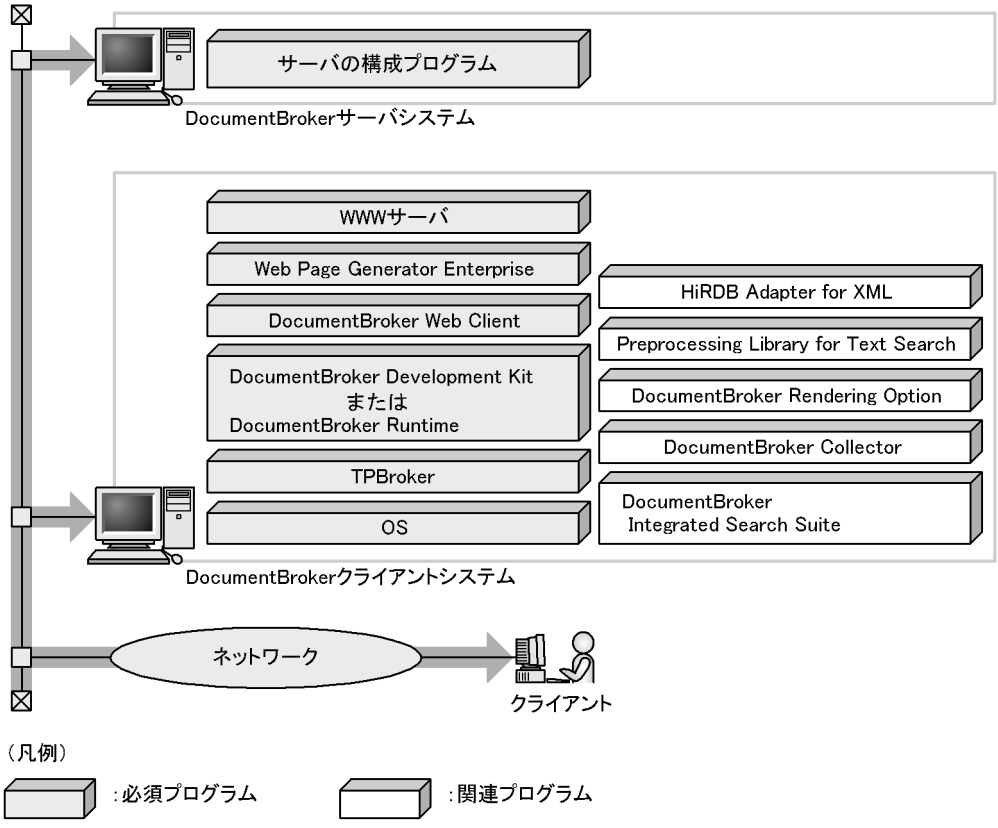
付録 J　DocumentBroker Web Client を使用した DocumentBroker クライアントの開発

DocumentBroker Web Client は，Web Page Generator Enterprise 対応のライブラリを提供しています。Web Page Generator Enterprise で構築した WWW 環境で動作するクライアントアプリケーションを開発・実行できます。また，DocumentBroker Web Client を使用して開発したサンプルの Web アプリケーションも提供しているため，このサンプルを参考にしたりカスタマイズしたりして，業務に適合したクライアントアプリケーションを開発できます。

なお，DocumentBroker クライアントシステムでは，バージョンが 03-00 よりも前の DocumentBroker Development Kit または DocumentBroker Runtime を使用してください。

ここでは，DocumentBroker サーバシステムに必要なプログラム，DocumentBroker クライアントシステムに必要なプログラム，および各機能を使用する場合に必要なプログラムについて説明します。DocumentBroker Web Client で開発したクライアントを構成するプログラムのシステム構成を次の図に示します。

図 J-1　DocumentBroker Web Client で開発したクライアントを構成するプログラムのシステム構成



（1）DocumentBroker サーバシステムに必要なプログラム

DocumentBroker サーバシステムに必要なプログラムについては，「1.3.2　DocumentBroker サーバを構成するプログラム」を参照してください。

(2) DocumentBroker クライアントシステムに必要なプログラム

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。

WWW サーバ

DocumentBroker Web Client は，WWW サーバシステムが構築されている環境で使用します。
DocumentBroker Web Client 使用できる WWW サーバの種類と WWW サーバのバージョンは，Web Page Generator Enterprise に依存します。詳細については，マニュアル「DocumentBroker Web Client Version 2　解説」を参照してください。

Web Page Generator Enterprise

Web Page Generator Enterprise は，WWW ブラウザの画面内容を動的に変更する機能を持つプログラムです。ユーザが指定した URL に従って WWW サーバが Web Page Generator Enterprise を呼び出し，Web Page Generator Enterprise が DocumentBroker Web Client のテンプレートを呼び出します。

DocumentBroker Web Client

DocumentBroker Web Client は，イントラネットまたはインターネットを経由して WWW ブラウザから DocumentBroker サーバにアクセスするためのプログラムです。Web Page Generator Enterprise で構築した WWW 環境で動作するクライアントアプリケーションを開発・実行するための，Web Page Generator Enterprise 対応のライブラリを提供しています。また，DocumentBroker Web Client を使用して開発したサンプルの Web アプリケーションも提供しているため，このサンプルを参考にしたりカスタマイズしたりして，業務に適合したクライアントアプリケーションを開発できます。
DocumentBroker Web Client の詳細については，マニュアル「DocumentBroker Web Client Version 2　解説」，およびマニュアル「DocumentBroker Web Client Version 2　リファレンス」を参照してください。

DocumentBroker Development Kit（クライアント開発環境）または DocumentBroker Runtime（クライアント実行環境）

DocumentBroker Development Kit および DocumentBroker Runtime は，DocumentBroker Web Client が DocumentBroker サーバに対して処理を要求するための API を持つプログラムです。
DocumentBroker Web Client では，DocumentBroker Development Kit が提供する C++ クラスライブラリを使用してクライアントアプリケーションを開発するため，DocumentBroker Web Client の開発環境には DocumentBroker Development Kit が必要です。実行環境だけが必要な場合は，DocumentBroker Runtime を使用します。

TPBroker

CORBA 仕様に基づく分散アプリケーションを開発するためのプログラムです。

OS

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 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 NT の場合：**
  Windows NT Server, Enterprise Edition 4.0 または Windows NT Server Network Operating

System Version 4.0 を使用してください。

(3) XML 文書管理機能を使用する場合に必要なプログラム

XML 文書管理機能を使用する場合は，次のプログラムが必要です。

- HiRDB Adapter for XML
- Preprocessing Library for Text Search

これらのプログラムの説明については，「1.3.3(3)　XML 文書管理機能を使用する場合に必要なプログラム」を参照してください。

(4) 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

レンディション変換要求機能を使用する場合は，次のプログラムが必要です。

- DocumentBroker Rendering Option

このプログラムの説明については，「1.3.3(4)　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」を参照してください。

(5) 分散した文書の収集・検索機能を使用する場合に必要なプログラム

DocumentBroker Collector

ネットワーク上に散在した文書を収集して DocumentBroker に登録する場合に必要なプログラムです。インターネット，イントラネット，ファイルシステムやグループウェアなどのほかの文書管理システムに分散しているさまざまなフォーマットの文書から属性を収集し，XML 形式に変換して DocumentBroker に登録して管理するプログラムです。DocumentBroker Collector の詳細については，マニュアル「DocumentBroker Collector」を参照してください。

DocumentBroker Integrated Search Suite

DocumentBroker Collector によって収集および登録された文書を検索する場合に必要なプログラムです。DocumentBroker Integrated Search Suite が提供する検索テンプレートを使用して WWW ブラウザ上に表示した検索画面から文書を検索できます。DocumentBroker Integrated Search Suite については，マニュアル「DocumentBroker Integrated Search Suite」を参照してください。

付録 K　このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

付録 K.1　関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。なお，本文に記載のマニュアル名称は，「uCosminexus DocumentBroker」を「DocumentBroker」と表記しています。

uCosminexus DocumentBroker **のマニュアル**

- uCosminexus DocumentBroker Version 3　**クラスライブラリ**　C++　**解説**（3000-3-F13）
  uCosminexus DocumentBroker Development Kit で提供するクラスライブラリの機能と，クラスライブラリを使用するために必要なオブジェクトモデル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**　C++　**リファレンス**　**基本機能編**（3000-3-F14）
  uCosminexus DocumentBroker Development Kit が提供するクラスライブラリのクラスの詳細とメソッドの文法について知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 3　**オブジェクト操作ツール**（3000-3-F15）
  uCosminexus DocumentBroker Development Kit または uCosminexus DocumentBroker Runtime で提供するオブジェクト操作ツールの機能について知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 3　**統計解析ツール**（3000-3-F11）
  uCosminexus DocumentBroker Server で提供する統計解析ツールの機能について知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 3　**メッセージ**（3000-3-F12）
  DocumentBroker が出力するメッセージ，およびメッセージに対してユーザが実施する対策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　**解説**（3000-3-F16）
  uCosminexus DocumentBroker Development Kit が提供する Java クラスライブラリの機能，クラスライブラリを使用するために必要なオブジェクトモデル，環境設定，および障害対策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　**リファレンス**（3000-3-F17）
  uCosminexus DocumentBroker Development Kit が提供する Java クラスライブラリのクラス，インターフェース，メソッドの文法およびメッセージについて知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 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 が提供する Java クラスライブラリのサンプル Web アプリケーションの機能，文書管理モデル，および操作方法について知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 3 Standard GUI　**システム導入・運用ガイド**（3020-3-U73）
  uCosminexus DocumentBroker Standard GUI の環境設定，および運用方法について知りたい場合に参照してください。
- uCosminexus DocumentBroker Version 3 Standard GUI　**操作ガイド**（3020-3-U74）
  uCosminexus DocumentBroker Standard GUI の GUI の操作方法など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 Text Search Index Loader の機能，環境設定，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 Life Cycle Suite Version 2（3000-3-796）
  DocumentBroker Life Cycle Suite の機能，環境設定，C++ クラスライブラリ，および分散オブジェクト群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 の機能，環境設定，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 の機能，環境設定，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 Collector（3020-3-C46）
  DocumentBroker Collector の機能，環境設定，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 Integrated Search Suite（3020-3-C47）
  DocumentBroker Integrated Search Suite の機能，環境設定，および操作方法について知りたい場合に参照してください。
- DocumentBroker **イメージ登録** for DocuCentre(R)（3020-3-C48）
  DocumentBroker イメージ登録 for DocuCentre(R) の機能，および環境設定について知りたい場合に参照してください。
- DocumentBroker **イメージ登録** for iR（3020-3-C49）
  DocumentBroker イメージ登録 for iR の機能，および環境設定について知りたい場合に参照してください。
- DocumentBroker **イメージ登録** for imagio and IPSiO（3020-3-C50）
  DocumentBroker イメージ登録 for imagio and IPSiO の機能，および環境設定について知りたい場合に参照してください。
- DocumentBroker **イメージ登録** for Konica（3020-3-C93）
  DocumentBroker イメージ登録 for Konica の機能，および環境設定について知りたい場合に参照してください。
- DocumentBroker **テキスト分析コンポーネント**（3020-3-D68）
  DocumentBroker テキスト分析コンポーネントのコマンド，クラスライブラリの機能と使用方法，および環境設定について知りたい場合に参照してください。
- DocumentBroker **テキスト分析テンプレート**（3020-3-D69）
  DocumentBroker テキスト分析テンプレートの Web アプリケーションプログラムの機能，環境設定，および操作方法について知りたい場合に参照してください。

DocumentBroker Integrated Search Suite for J Version 2（3020-3-D78）
DocumentBroker Integrated Search Suite for J の機能，環境設定方法，および操作方法について知りたい場合に参照してください。

関連製品のマニュアル（HiRDB）

- **スケーラブルデータベースサーバ** HiRDB Version 6 **システム導入・設計ガイド**（UNIX(R) **用**）（3000-6-232）
- **スケーラブルデータベースサーバ** HiRDB Version 7 **システム導入・設計ガイド**（UNIX(R) **用**）（3000-6-272）
- **スケーラブルデータベースサーバ** HiRDB Version 8 **システム導入・設計ガイド**（UNIX(R) **用**）（3000-6-352）
- HiRDB Version 9 **システム導入・設計ガイド**（UNIX(R) **用**）（3000-6-452）

- **スケーラブルデータベースサーバ** HiRDB Version 6 **システム定義（**UNIX(R) **用）**（3000-6-233）
- **スケーラブルデータベースサーバ** HiRDB Version 7 **システム定義（**UNIX(R) **用）**（3000-6-273）
- **スケーラブルデータベースサーバ** HiRDB Version 8 **システム定義（**UNIX(R) **用）**（3000-6-353）
- **スケーラブルデータベースサーバ** HiRDB Version 9 **システム定義（**UNIX(R) **用）**（3000-6-453）
- **スケーラブルデータベースサーバ** HiRDB Version 6 **システム運用ガイド（**UNIX(R) **用）**（3000-6-234）
- **スケーラブルデータベースサーバ** HiRDB Version 7 **システム運用ガイド（**UNIX(R) **用）**（3000-6-274）
- **スケーラブルデータベースサーバ** HiRDB Version 8 **システム運用ガイド（**UNIX(R) **用）**（3000-6-354）
- HiRDB Version 9 **システム運用ガイド（**UNIX(R) **用）**（3000-6-454）
- **スケーラブルデータベースサーバ** HiRDB Version 6 **コマンドリファレンス（**UNIX(R) **用）**（3000-6-235）
- **スケーラブルデータベースサーバ** HiRDB Version 7 **コマンドリファレンス（**UNIX(R) **用）**（3000-6-275）
- HiRDB Version 9 **コマンドリファレンス（**UNIX(R) **用）**（3000-6-455）
- **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）
- **スケーラブルデータベースサーバ** HiRDB Version 7 UAP **開発ガイド（**UNIX(R)/Windows(R) **用）**（3000-6-276）
- **スケーラブルデータベースサーバ** HiRDB Version 8 UAP **開発ガイド**（3020-6-356）
- HiRDB Version 9 UAP **開発ガイド**（3020-6-456）
- **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）
- **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）
- **スケーラブルデータベースサーバ** HiRDB Version 8 SQL **リファレンス**（3020-6-357）
- HiRDB Version 9 SQL **リファレンス**（3020-6-457）
- HiRDB **データ連動機能** HiRDB Datareplicator Version 6（3000-6-243）
- HiRDB **データ連動機能** HiRDB Datareplicator Version 7（3000-6-286）
- HiRDB **データ連動機能** HiRDB Datareplicator Version 8（3020-6-360）
- HiRDB **データ連動拡張機能** HiRDB Datareplicator Extension Version 6（3000-6-244）
- HiRDB **データ連動拡張機能** HiRDB Datareplicator Extension Version 7（3000-6-287）
- HiRDB **データ連動拡張機能** HiRDB Datareplicator Extension Version 8（3020-6-361）
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 2（3000-6-245）
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 7（3000-6-288）
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 8（3020-6-375）
- HiRDB **全文検索プラグイン** HiRDB Text Search Plug-in Version 9（3020-6-481）
- HiRDB Adapter for XML　**ユーザーズガイド**（3000-6-250）
- HiRDB **デジタルコンテンツアクセスプラグイン** HiRDB File Link（UNIX(R) **用）**（3000-6-253）
- **インナレプリカ機能** HiRDB Staticizer Option（3000-6-249）
- **インナレプリカ機能** HiRDB Staticizer Option Version 7（3000-6-282）
- **インナレプリカ機能** HiRDB Staticizer Option Version 8（3000-6-363）
- **インナレプリカ機能** HiRDB Staticizer Option Version 9（3000-6-463）

関連製品のマニュアル（その他）

- JP1/HiCommand Tiered Storage Manager　**ユーザーズガイド**（3020-3-J01）
- Preprocessing Library for Text Search Version 2（3000-7-270）
- **トランザクショナル分散オブジェクト基盤** TPBroker　**ユーザーズガイド**（3020-3-M16）
- VisiBroker Version 3 VisiBroker for C++　**プログラマーズガイド**（3000-3-651）※2
- VisiBroker Version 3 VisiBroker for C++　**プログラマーズガイド**（Windows(R) **用**）（3000-3-678）※2
- VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　**デベロッパーズガイド**（3000-3-936）（3020-3-M45）※3
- VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R)　**プログラマーズリファレンス**（3000-3-937）（3020-3-M46）※3
- **日立ディレクトリサービス　導入編**（3020-3-825）
- **日立ディレクトリサービス　システム管理編**（3020-3-826）
- **日立ディレクトリサービス**　AP **開発編**（3020-3-827）

注※ 1

TPBroker のバージョンが 03-04 以前の場合は，資料番号が「3000-3-660」のマニュアルを参照してください。TPBroker のバージョンが 03-05 以降，かつ 05-00 よりも前の場合は，資料番号が「3000-3-771」のマニュアルを参照してください。TPBroker のバージョンが 05-00 以降の場合は，資料番号が「3020-3-M16」のマニュアルを参照してください。

注※ 2

TPBroker のバージョンが 05-00 よりも前の場合に参照してください。

注※ 3

TPBroker のバージョンが 05-00 以降の場合に参照してください。

関連マニュアルの略称

このマニュアルで使用する関連マニュアルの略称を次に示します。

マニュアル名	略称
DocumentBroker Rendering Option Version 3	DocumentBroker Rendering Option
DocumentBroker Rendering Option Version 3 （活文 PDFstaff 編）	
DocumentBroker イメージ登録 for DocuCentre(R)	DocumentBroker イメージ登録 for DocuCentre
スケーラブルデータベースサーバ HiRDB Version 6 システム導入・設計ガイド（UNIX(R) 用）	HiRDB　システム導入・設計ガイド
スケーラブルデータベースサーバ HiRDB Version 7 システム導入・設計ガイド（UNIX(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 8 システム導入・設計ガイド（UNIX(R) 用）	
HiRDB Version 9 システム導入・設計ガイド（UNIX(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 6 システム定義（UNIX(R) 用）	HiRDB　システム定義
スケーラブルデータベースサーバ HiRDB Version 7 システム定義（UNIX(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 8 システム定義（UNIX(R) 用）	

マニュアル名	略称
HiRDB Version 9 システム定義（UNIX(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 6 システム運用ガイド（UNIX(R) 用）	HiRDB　システム運用ガイド
スケーラブルデータベースサーバ HiRDB Version 7 システム運用ガイド（UNIX(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 8 システム運用ガイド	
HiRDB Version 9 システム運用ガイド	
スケーラブルデータベースサーバ HiRDB Version 6 コマンドリファレンス（UNIX(R) 用）	HiRDB　コマンドリファレンス
スケーラブルデータベースサーバ HiRDB Version 7 コマンドリファレンス（UNIX(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 8 コマンドリファレンス（UNIX(R) 用）	
HiRDB Version 9 コマンドリファレンス（UNIX(R) 用）	
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）	HiRDB　UAP 開発ガイド
スケーラブルデータベースサーバ HiRDB Version 7 UAP 開発ガイド（UNIX(R)/Windows(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 8 UAP 開発ガイド	
HiRDB Version 9 UAP 開発ガイド	
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）	HiRDB　SQL リファレンス
スケーラブルデータベースサーバ HiRDB Version 7 SQL リファレンス（UNIX(R)/Windows(R) 用）	
スケーラブルデータベースサーバ HiRDB Version 8 SQL リファレンス	
HiRDB Version 9 SQL リファレンス	
HiRDB データ連動機能 HiRDB Datareplicator Version 6	HiRDB Datareplicator
HiRDB データ連動機能 HiRDB Datareplicator Version 7	
HiRDB データ連動機能 HiRDB Datareplicator Version 8	
HiRDB データ連動拡張機能 HiRDB Datareplicator Extension Version 6	
HiRDB データ連動拡張機能 HiRDB Datareplicator Extension Version 7	
HiRDB データ連動拡張機能 HiRDB Datareplicator Extension Version 8	
HiRDB 全文検索プラグイン HiRDB Text Search Plug-in Version 2	HiRDB Text Search Plug-in
HiRDB 全文検索プラグイン HiRDB Text Search Plug-in Version 7	
HiRDB 全文検索プラグイン HiRDB Text Search Plug-in Version 8	
HiRDB 全文検索プラグイン HiRDB Text Search Plug-in Version 9	
HiRDB デジタルコンテンツアクセスプラグイン HiRDB File Link（UNIX(R) 用）	HiRDB File Link
インナレプリカ機能 HiRDB Staticizer Option	HiRDB Staticizer Option
インナレプリカ機能 HiRDB Staticizer Option Version 7	
インナレプリカ機能 HiRDB Staticizer Option Version 8	

マニュアル名	略称
インナレプリカ機能 HiRDB Staticizer Option Version 9	
TPBroker　ユーザーズガイド	TPBroker　ユーザーズガイド
TPBroker for C++　ユーザーズガイド	
VisiBroker Version 3 VisiBroker for C++　プログラマーズガイド	VisiBroker for C++　プログラマーズガイド
VisiBroker Version 3 VisiBroker for C++　プログラマーズガイド（Windows(R) 用）	

付録 K.2　このマニュアルでの表記

このマニュアルでは，製品名称を次に示す略称で表記しています。

製品名称	略称
AIX 5L V5.1	AIX
AIX 5L V5.2	
AIX 5L V5.3	
Canon imageRUNNER(R) シリーズ	imageRUNNER
Cosminexus Application Server Version 5	Cosminexus
Cosminexus Application Server Enterprise Version 6	
Cosminexus Application Server Standard Version 6	
uCosminexus Application Server Enterprise Version 6.7	
uCosminexus Application Server Standard Version 6.7	
uCosminexus Application Server Enterprise Version 7	
uCosminexus Application Server Standard Version 7	
DABroker	DABroker
DABroker for C++	
DocumentBroker Collector	DocumentBroker Collector
DocumentBroker Collector for Groupmax	
DocumentBroker Collector for WWW	
DocumentBroker Collector for Lotus Notes	
DocumentBroker Collector for RDB	
DocumentBroker Development Kit Version 3	DocumentBroker Development Kit
DocumentBroker Development Kit Version 3，または DocumentBroker Web Client Version 3 で開発したクライアントアプリケーション	DocumentBroker クライアント
DocumentBroker Development Kit Version 3，または DocumentBroker Web Client Version 2 の GUI	
DocumentBroker Integrated Search Suite for J Version 2	DocumentBroker Integrated Search Suite for J
DocumentBroker Life Cycle Suite Version 2	DocumentBroker Life Cycle Suite
DocumentBroker Life Cycle Suite Development Kit Version 2	
DocumentBroker Object Loader Version 3	DocumentBroker Object Loader
DocumentBroker Runtime Version 3	DocumentBroker Runtime

製品名称	略称	
DocumentBroker Server Version 3	DocumentBroker Server または DocumentBroker サーバ	
DocumentBroker Text Search Index Loader Version 3	DocumentBroker Text Search Index Loader	
DocumentBroker Version 3 Standard GUI	DocumentBroker Standard GUI	
DocumentBroker Version 3 ブジェクト操作ツール	DocumentBroker オブジェクト操作ツール	
DocumentBroker Web Client Version 2	DocumentBroker Web Client	
DocumentBroker イメージ登録 for DocuCentre(R)	DocumentBroker イメージ登録 for DocuCentre	
FUJI XEROX DocuCentre(R) シリーズ	DocuCentre	
FUJI XEROX DocuColor(R) シリーズ		
High Availability ClusterMulti-Processing for AIX	HACMP	
HiRDB Adapter for XML - Enterprise Edition	HiRDB Adapter for XML	
HiRDB Adapter for XML - Standard Edition		
HiRDB Datareplicator Version 6	HiRDB Datareplicator	
HiRDB Datareplicator Version 7		
HiRDB Datareplicator Version 8		
HiRDB Datareplicator Version 9		
HiRDB Datareplicator Extension Version 6		
HiRDB Datareplicator Extension Version 7		
HiRDB Datareplicator Extension Version 8		
HiRDB Datareplicator Extension Version 9		
HiRDB Text Search Plug-in Conceptual Extension Version 2	HiRDB Text Search Plug-in Conceptual Extension	
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		
HiRDB Text Search Plug-in Version 9		
HiRDB/Parallel Server Version 6	HiRDB/Parallel Server	HiRDB サーバ
HiRDB/Parallel Server Version 7		
HiRDB/Parallel Server Version 8		
HiRDB/Single Server Version 6	HiRDB/Single Server	
HiRDB/Single Server Version 7		
HiRDB/Single Server Version 8		
HiRDB Server Version 9	—	
HiRDB/Parallel Server - Object Option Version 6	HiRDB Object Option	
HiRDB/Parallel Server - Object Option Version 7		
HiRDB/Single Server - Object Option Version 6		
HiRDB/Single Server - Object Option Version 7		
HiRDB/Run Time Version 6	HiRDB/Run Time	
HiRDB/Run Time Version 7		
HiRDB/Run Time Version 8		

製品名称	略称
HiRDB/Run Time Version 9	
Hitachi Directory Runtime Version 2	Hitachi Directory Runtime
IBM Directory Server Version 4.1	IBM Directory Server
IBM SecureWay Directory Version 3.2.1	SecureWay Directory
IBM SecureWay Directory Version 3.2.2	
IBM Tivoli Directory Server V5.2	IBM Tivoli Directory Server
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 Directory Server 4	iPlanet Directory Server
Konica Sitios(R) シリーズ	Sitios
Oracle Directory Server Enterprise Edition 11g	Oracle Directory Server
Preprocessing Library for Text Search Version 2	Preprocessing Library for Text Search
Red Hat Enterprise Linux 6 (x86_64)	Linux
RICOH imagio(R) シリーズ	imagio
RICOH IPSiO(R) シリーズ	IPSiO
SANRISE Universal Storage Platform 100	SANRISE USP
SANRISE Universal Storage Platform 600	
SANRISE Universal Storage Platform 1100	
SANRISE Network Storage Controller NSC55	
Sun Cluster 2.2	Sun Cluster
The Microsoft(R) Office	Office
TPBroker Developer for C++	TPBroker V3
TPBroker for C++	
TPBroker	TPBroker V5
WorkCoordinator Library Extension Version 3	WorkCoordinator
WorkCoordinator Library Version 3	
WorkCoordinator Server Version 3	
WorkCoordinator Definer Version 3	WorkCoordinator Definer
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。

- DocumentBroker の製品群を総称して DocumentBroker と表記します。
- DocumentBroker の 01-21 以前のバージョンについては，Version 1 と表記します。
- HiRDB の製品群を総称して HiRDB と表記します。
- AIX および Linux を合わせて UNIX と表記することがあります。
- TPBroker V3 および TPBroker V5 を合わせて TPBroker と表記することがあります。
- それぞれの製品について個別に説明する場合は，製品名称を使用します。
- IBM Directory Server，IBM Tivoli Directory Server，SecureWay Directory，Sun Java System

Directory Server，および Active Directory を合わせて LDAP 対応のディレクトリサービスと表記することがあります。

- リレーショナルデータベース管理システムをデータベースシステムと表記します。

付録 K.3　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
EOF	End of File
ESIS-B	Element Structure Information Set-Binary Format
EUC	Extended UNIX Code
FAM	File Access Module
GIF	Graphics Interchange Format
GUI	Graphical User Interface
GUID	Globally Unique Identifier

英略語	英字での表記
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ID	Identifier
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
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TIFF	Tag Image File Format
UOC	User Own Coding
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
UTF-8	8-bit UCS Transformation Format
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

付録 K.4　KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バ

イト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

付録 L　用語解説

DocumentBroker で使用する用語について説明します。

（英字）

ACE（Access Control Element）

アクセス制御エレメントのことです。

ACFlag（Access Control Flag）

アクセス制御フラグのことです。

ACL（Access Control List）

アクセス制御リストのことです。

AND-NOT 検索

検索オペレータに「AND-NOT」を使用する検索方法です。二つの検索条件を AND-NOT でつないで，左側のオペランドに指定した検索条件は成立するが，右側のオペランドに指定した検索条件は成立しない文書を検索します。例えば，「著者が『日立太郎』であるが，所属は『日立製作所』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。

AND 検索

検索オペレータに「AND」を使用して，検索条件同士の論理積を求める検索方法です。例えば，「著者が『日立太郎』で，文書中に『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を指します。

ConfigurationHistory オブジェクト

バージョン管理に使用する最上位の DMA オブジェクトです。文書のバージョンを管理する VersionSeries オブジェクトを管理します。

Containable オブジェクト

コンテナの包含要素になるオブジェクトです。CdbrContainable クラスのサブクラスを基に作成されたオブジェクトを指します。

Containee

参照型のコンテインメント（ReferentialContainment）の場合，オブジェクト（Container）に包含されるオブジェクトを指します。

Container

参照型のコンテインメントの場合，オブジェクト（Containee）を包含するオブジェクトを指します。

ContainerVersion オブジェクト

包含しているオブジェクト全体を一つの概念的なオブジェクトとして管理する DMA オブジェクトです。また，構成管理型のコンテインメントではオブジェクトを包含するオブジェクトとして利用できます。ContainerVersion オブジェクトは，バージョンを保持できるオブジェクトです。したがって，ContainerVersion オブジェクトのバージョンを上げることで，ある時点でのオブジェクトのまとまりを管理できます。

Container オブジェクト

コンテインメントを利用してオブジェクトを管理する場合に，オブジェクトを包含する DMA オブジェクトの一つです。dmaClass_Container クラスおよびユーザが定義した dmaClass_Container クラスのサブクラスを基に作成します。

ContentElement（dmaClass_ContentElement クラス）

文書のコンテントにアクセスするために使われる DMA オブジェクトの抽象クラスです。ContentElement の四つのサブクラス（edmClass_ContentFileLink クラス，dmaClass_ContentTransfer クラス，edmClass_ContentTransfers クラス，および dmaClass_ContentReference クラス）は，コンテントの格納とアクセス機能を提供します。

ContentFileLink オブジェクト

文書のコンテントを管理するために使用する DMA オブジェクトです。File Link 連携機能を使用する場合に使用します。dmaClass_ContentElement クラスのサブクラスである edmClass_ContentFileLink クラスを基に作成します。

ContentReference オブジェクト

文書のコンテントを管理するために使用する DMA オブジェクトです。コンテントをデータベースに格納しないで，位置情報を永続化して管理する場合に使用します。dmaClass_ContentElement クラスのサブクラスである dmaClass_ContentReference クラスを基に作成します。

ContentTransfers オブジェクト

文書のコンテントを管理するために使用する DMA オブジェクトです。一つの文書のコンテントとして複数のファイルをデータベースに格納（永続化）して管理する場合に使用します。dmaClass_ContentElement クラスのサブクラスである edmClass_ ContentTransfers クラスを基に作成します。

ContentTransfer オブジェクト

文書のコンテントを管理するために使用する DMA オブジェクトです。コンテントをデータベースに格納（永続化）して管理する場合に使用します。dmaClass_ContentElement クラスのサブクラスである dmaClass_ContentTransfer クラスを基に作成します。

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。

DCD（Document Content Description）

W3C で定義している，タグセットを記述するためのタグセットです。マッピング元 XML タグ定義はこの DCD の形式で記述します。DTD と同じように XML 文書の構造を記述しますが，DTD と異なり，DCD 自体が XML のタグセットとして定義されています。また，タグおよびその属性について，データの型を指定できるといった特長を持ちます。

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。

DMA URL

DMA オブジェクトの OIID（Object Instance Identifier）を定義する URL です。

DMA オブジェクト

DMA が規定するオブジェクトモデルに基づいたオブジェクトです。DMA で規定されたクラス（クラス名が dmaClass_ で始まるクラス），DocumentBroker で拡張したクラス（クラス名が edmClass_ で始まるクラス），およびこれらのクラスからユーザが定義したサブクラスを基に作成されます。

DocumentBroker Standard GUI

業種・業務に依存しない標準的な企業内文書管理システムを提供する ECM パッケージ製品です。
文書を審査・承認後に公開する仕組み，公開する文書に電子署名や承認日時を付与する機能などを文書管理システムに取り込めるため，コンプライアンスや情報セキュリティへの対応が容易になります。

DocumentSpace 構成定義ファイル（docspace.ini）

文書空間の構成を定義するために使用するファイルです。

DocVersion オブジェクト

DocumentBroker で扱う文書に相当する DMA オブジェクトです。dmaClass_DocVersion クラスおよびユーザが定義した dmaClass_DocVersion クラスのサブクラスを基に作成します。

edmClass_Relationship クラス

文書間リレーションを表す 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クラスです。

edmSQL

DocumentBroker のオブジェクトを検索するための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 ライクの文法で記述できる edmSQL 文を指定して実行する検索のことです。

FAM（File Access Module）

HiRDB File Link で使用するファイルサーバ上で，コンテントを管理するためのプログラムです。

File Link 文書

HiRDB File Link で管理しているファイルサーバに格納されている一つのファイル，または一つのディレクトリおよびそのサブディレクトリ下のすべてのファイルをコンテントとして持つ文書のことです。データベースでは，コンテントへのリンク情報を管理しています。バージョンなし文書またはバージョン付き文書を File Link 文書として扱えます。

File Link 連携機能

HiRDB File Link と連携してコンテントを管理する機能です。

GUID（Globally Unique Identifier）

DMA のクラス，プロパティ，検索オペレータなどに与えるユニークな識別子です。GUID 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（8-4-4-4-12）」の形式で表されます。

ID ファイル

DocumentBroker オブジェクト操作ツールを使用して複数のオブジェクトを一括して操作する場合，操作対象となるオブジェクトの OIID などを記述するファイルです。

LDAP（Lightweight Directory Access Protocol）

TCP/IP 上で動作する解放型 DAP を提供し，X.500 のデータモデルを保持しています。

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）

文書空間での永続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使用する識別子です。OIID は DMA URL として定義されます。

OR 検索

検索オペレータに「OR」を使用して，検索条件同士の論理和を求める検索方法です。例えば，「作成者が『日立太郎』であるオブジェクトか，作成者の所属が『日立製作所』であるオブジェクトを検索する」というような場合に使用できます。

Parent

直接型のコンテインメントの場合，オブジェクト（Child）を包含するオブジェクトを指します。

RD エリア

データベースの表，インデクスおよびデータディクショナリを格納するデータ領域のことです。データの格納単位の一つで，1 ～ 16 個の HiRDB ファイルから構成されます。

Rendition オブジェクト

文書の表現形式（HTML，PDF など）を管理する DMA オブジェクトです。

Reservation オブジェクト

新しいバージョンを作成する権利を VersionSeries オブジェクトに予約する DMA オブジェクトです。新しいバージョンのチェックインが完了すると，このオブジェクトは削除されて，VersionDescription オブジェクトに置き換えられます。

Scalar 型

プロパティの基本単位の一つです。基本単位が Scalar 型であるプロパティは，データ型に従った値を一つだけ持ちます。

SystemManager オブジェクト

System オブジェクトを管理する DMA オブジェクトです。

System オブジェクト

サーバ内の文書空間を管理する DMA オブジェクトです。

UOC（User Own Coding）

ユーザによって作成されたプログラムのことです。DocumentBroker では，ユーザ認証や，アクセス制御機能のためのユーザ情報取得に UOC を使用できます。

VariableArray 型

プロパティの基本単位の一つです。基本単位が VariableArray 型であるプロパティは，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。また配列の要素は，構造体で管理できます。

Versionable オブジェクト

バージョン管理の対象になるオブジェクトです。

VersionDescription オブジェクト

文書のバージョン（更新履歴）を管理するために使用する DMA オブジェクトです。VersionSeries オブジェクトと Versionable オブジェクトを接続するために使用します。

VersionSeries オブジェクト

文書のバージョン（更新履歴）を管理するために使用する DMA オブジェクトです。連続的な履歴を持つバージョンの構成を保持するオブジェクトです。

VersionTracedComponentDocVersion オブジェクト

直接型，参照型および構成管理型のコンテインメントでコンテナに包含される対象となるためのプロパティを持つ DMA オブジェクトです。dmaClass_DocVersion クラスのサブクラスの edmClass_VersionTracedComponentDocVersion クラスを基に作成します。

VersionTracedDocVersion オブジェクト

直接型，参照型および構成管理型のコンテインメントで，コンテナに包含される対象となるためのプロパティを持つ DMA オブジェクトです。dmaClass_DocVersion クラスのサブクラスの edmClass_VersionTracedDocVersion クラスを基に作成します。

VTContainee

構成管理型のコンテインメントの場合，オブジェクト（VTContainer）に包含されるオブジェクトを指します。

VTContainer
構成管理型のコンテインメントの場合，オブジェクト（VTContainee）を包含するオブジェクトを指します。

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 文書の文書ファイルからプレーンテキスト形式または構造指定検索用の全文検索インデクスデータを作成する機能です。XML 形式のファイルの構文解析もあわせて実行します。

XML プロパティマッピング機能
XML 形式のファイルを構文解析して，DocumentBroker サーバのプロパティマッピング定義に従って XML 形式のファイル内に記述されている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。

（ア行）

アクセス権
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 の文書空間でのオブジェクトの作成や，管理されている文書やコンテナなどの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オブジェクトは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発行する際に，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
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オブジェクトにバインド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）
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。

アクセス制御モデル
アクセス制御機能を利用して運用されている文書管理モデルです。

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。

アドレス呼び出し
文書空間内に構成されている独立した永続オブジェクトに付けられた ID（OIID）を利用して，そのオブジェクトを探索することです。

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方法です。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。

インデクス情報ファイル
ユーザが追加するプロパティにインデクスを定義する場合に，定義するインデクスの情報を記述するファイルです。

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を指します。

永続プロパティ
データベースに存在するプロパティを指します。

オブジェクト作成権
オブジェクト作成権限で，文書空間に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
文書空間に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，オブジェクトを作成するメソッドを実行できます。

オブジェクト操作権限
文書空間内のすべてのオブジェクトを，与えられた権限の範囲で操作する権利で，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 Object 型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。

（カ行）

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。

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・更新する時に使用します。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。

関連オブジェクト
コンテインメントの型（直接型，間接型および構成管理型）を定義するための DMA オブジェクトです。
直接型のコンテインメントの場合，オブジェクトを包含するオブジェクト（Parent）とオブジェクトに包含されるオブジェクト（Child）を DirectContainmentRelationship オブジェクトを使用して関連づけます。参照型のコンテインメントの場合，オブジェクトを包含するオブジェクト（Container）とオブジェクトに包含されるオブジェクト（Containee）を ReferentialContainmentRelationship オブジェクトを使用して関連づけます。
構成管理型のコンテインメントの場合，オブジェクトを包含するオブジェクト（VTContainer）とオブジェクトに包含されるオブジェクト（VTContainee）を VersionTraceableContainmentRelationship オブジェクトを使用して関連づけます。

基本コンテント更新権

基本パーミッションの一つで，オブジェクトのコンテント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。文書に対して設定する場合は，全文検索インデクスを作成，削除する権利を含みます。

基本コンテント参照権

基本パーミッションの一つで，オブジェクトのコンテント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。文書に対して設定する場合は，全文検索を実行する権利を含みます。

基本削除権

基本パーミッションの一つで，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。

基本単位

DMA で規定されているプロパティは，データ型に従った値を 1 個持つか複数個持つかが決められています。これを基本単位といいます。

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，基本コンテント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る）。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，基本コンテント参照権，基本コンテント更新権，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，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，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。コンテナに対して設定する場合は，関連オブジェクトのユーザ定義プロパティの参照と管理されている要素を参照する権利を含みます。

基本リンク権

基本パーミッションの一つで，コンテインメントの設定，変更および関連オブジェクトのユーザ定義プロパティを変更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。

近傍条件検索

検索オペレータに「Prox」を使用する拡張検索（全文検索）です。指定する検索タームが同時に存在する場合に，その検索ターム間の距離を条件として検索します。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ときに使用します。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，基本コンテント参照権および基本コンテント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。

クライアントアプリケーション動作定義機能
あらかじめ定義しておいた動作環境でクライアントアプリケーションを動作させる機能です。

クライアントアプリケーション動作定義ファイル
クライアントアプリケーションの動作環境を定義したファイルです。

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，クラス名またはプロパティ名から GUID，データ型，プロパティの基本単位などの情報を取得するために使用するファイルです。
EDMCrtSimMeta コマンドでも作成できます。Java クラスライブラリや DocumentBroker オブジェクト操作ツールを使用する場合に必要です。

継承（Inheritance）
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。

構成管理型のコンテインメント（VersionTraceableContainment）
構成管理コンテナを利用した文書管理の方法です。構成管理型のコンテインメントで使われる構成管理コンテナは，バージョンを作成できるオブジェクトの構成を管理します。

構成管理コンテナ
構成管理型のコンテインメントで使われるコンテナです。
包含されるオブジェクトのバージョンを固定して管理したり，常に最新のバージョンを追跡して管理したりすることで，包含されるオブジェクト全体の構成を管理するために使用するコンテナです。DMA オブジェクトでは，ContainerVersion オブジェクトに相当します。

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（エレメント）やエレメント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。

コンテインメント（包含）
コンテナを使用したオブジェクトの包含関係を示します。コンテインメントには直接型（DirectContainment），参照型（ReferentialContainment）および構成管理型（VersionTraceableContainment）の 3 種類があります。

コンテナ
コンテインメントで，オブジェクトを包含できるオブジェクトの総称です。また，DMA オブジェクトとしては，Container オブジェクト，VersionTraceableContainer オブジェクトおよび ContainerVersion オブジェクトがあります。

コンテント
一般的には，属性に対する文書のデータ部分を指します。DMA では，ある文書に関連づけられていて，DMA で規定されている Content モデルに従ってアクセスされるオブジェクトの集合を指します。また，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）をコンテントデータといいます。

コンテント格納先パス
リファレンスファイル管理機能を使用する場合に，コンテントの格納先として，コンテント格納先ベースパスからの相対パスで指定するディレクトリパスです。コンテント格納先パスで指定したディレクトリの下に，DocumentBroker がコンテントを管理するためのディレクトリが作成され，その下にコンテントが格納されます。コンテント格納先パスに指定するディレクトリは，DocumentBroker によって作成されます。

コンテント格納先ベースパス
リファレンスファイル管理機能を使用する場合に，コンテントの格納先の基点となるディレクトリを指定するディレクトリパスです。コンテント格納先ベースパスに指定するディレクトリは，ファイルシステム上の任意の領域にユーザが

作成し，ユーザが管理します。コンテント格納先ベースパスに指定したディレクトリの下に，コンテント格納先パスで指定したディレクトリ，および DocumentBroker がコンテントを管理するためのディレクトリが作成され，その下にコンテントが格納されます。

コンテント種別
レンディションのコンテントが，シングルファイル文書のコンテントか，マルチファイル文書のコンテントか，リファレンスファイル文書のコンテントか，または File Link 文書のコンテントかを示します。

コンテントロケーション
リファレンスファイル管理機能を使用する場合に，データベースで管理される，コンテントの格納先の情報です。コンテント格納先ベースパスからの相対パスがコンテントロケーションとしてデータベースに登録されます。

（サ行）

サーババインド
DocumentBroker から LDAP 対応のディレクトリサービスに対して，DN とパスワードを使用して認証を行い，バインドすることです。

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（Subject）
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（SubjectType）
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に，追加登録されたレンディションのことです。マスタレンディション以外のレンディションを指します。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。

参照型のコンテインメント（ReferentialContainment）
コンテナを利用した文書管理の方法です。参照型のコンテインメントで使われるコンテナは，文書にはり付けるインデクスの働きをします。

参照権
組み合わせパーミッションの一つです。基本コンテント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，基本コンテント参照権および基本コンテント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トを参照，更新する権利を設定できます。

システム管理者
DocumentBroker を運用，管理および保守するユーザです。

実行環境制御ファイル
DocumentBroker オブジェクト操作ツールの実行環境を定義するファイルです。

状態フラグ
マルチレンディション文書の，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ト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ト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ト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ト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トが存在しない，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（Owner）

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パーミッションを与えられます。所有者に設定されているユーザは，その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オブジェクトに対して実行できます。また，その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。

シングルファイル文書

データベースに登録されている一つのファイルをコンテントとして持つ文書のことです。

スーパークラス

あるクラスのクラス定義に使われたクラスを，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

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特権を持ち，文書空間のすべての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。

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

全文検索の対象になるテキストデータに対応するプロパティです。edmProp_TextIndex プロパティ，edmProp_StIndex プロパティ，edmProp_ConceptTextIndex プロパティ，edmProp_ConceptStIndex プロパティおよび edmProp_Content プロパティに相当します。

全文検索機能付き文書クラス

全文検索の対象となる文書を作成するためのクラスです。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，ユーザ定義のクラスです。

（タ行）

チェックアウト（check-out）

文書またはコンテナにバージョンを追加するために，次バージョンの追加を予約して，最新バージョンのコピーを要求することです。

チェックイン（check-in）
文書またはコンテナのバージョンの追加を確定することです。

直接型のコンテインメント（DirectContainment）
コンテナを利用した文書管理の方法です。直接型のコンテインメントで使われるコンテナは，文書を格納するフォルダの働きをします。

定義情報ファイル
サブクラスおよびプロパティを追加するときに，追加するオブジェクトの定義情報を記述するファイルです。

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，SecureWay 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，「1.3.2(5)　アクセス制御機能を使用する場合に必要なプログラム」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。

動作環境メタ情報ファイル
DocumentBroker を起動するときに参照するメタ情報です。実行環境ディレクトリ /etc/meta_files に格納されます。

匿名バインド
DocumentBroker から LDAP 対応のディレクトリサービスに対して，匿名ユーザで認証を行い，バインドすることです。

独立データ
ほかのオブジェクトに依存しない，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
クラスライブラリのオブジェクトを構成する複数の DMA オブジェクトのうち，最上位に位置するオブジェクトです。例えば，バージョンなし文書の場合は，DMA オブジェクトの DocVersion オブジェクトがトップオブジェクトです。

（ハ行）

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，基本コンテント参照権，基本コンテント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。

バージョン付き構成管理コンテナ
バージョンを保持できる構成管理コンテナです。DMA オブジェクトでは，ConfigurationHistory オブジェクトに相当します。

バージョン付き文書
複数のバージョンを保持できる文書を表すオブジェクトです。DMA オブジェクトでは，ConfigurationHistory オブジェクトに相当します。

バージョンなし構成管理コンテナ
バージョンを保持しない構成管理コンテナです。DMA オブジェクトでは，ContainerVersion オブジェクトに相当します。

バージョンなし文書
バージョン管理しない文書を表すオブジェクトです。DMA オブジェクトでは，DocVersion オブジェクトに相当します。

パーミッション
オブジェクトの作成，オブジェクトのプロパティ参照，オブジェクトのコンテント更新などの実行可能な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。オブジェクトの操作の範囲を定めるパーミッションには，基本パーミッションと組み合わせ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。
また，LDAP 対応のディレクトリサービスに対して，DN とパスワードを使用して認証を行うことも，バインドといいます。

パブリック ACL
文書空間にオブジェクトとして存在するアクセス制御リスト（ACL）です。複数のオブジェクトが共有できます。

非ナル値制約
非ナル値制約とは，列の値にナル値を許さない制約のことです。

ファイルサーバ
HiRDB File Link で構築したファイルサーバを指します。

フィルタリング定義ファイル（TFD）
Preprocessing Library for Text Search の正規化機能によって構造指定検索用データ（ESIS-B データ）から不要なタグを除去したり，不要なタグ間のテキストをタグごと除去したりする場合に必要な定義ファイルです。

複合データ
複数の異なる型によって表されるデータです。主に，VariableArray 型のプロパティとして設定されている Object 型の要素の値を参照，設定するときに使用します。

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。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。

プロパティマッピング定義ファイル（DPM）
XML 文書中のタグ名とその内容をマッピングするクラス名，およびプロパティ名の対応関係の定義（プロパティマッ

ピング定義）を記述したファイルです。この定義ファイルを基に，XML 定義ファイルの追加／更新／削除コマンド（EDMXmlMap）によってマッピング定義ファイルを生成します。
プロパティマッピング定義は，登録時にマッピング定義名を付けて登録します。プロパティマッピングの実行時には，このマッピング定義名を指定して使用するプロパティマッピング定義を選択します。

文書

DMA オブジェクトでは，dmaClass_DocVersion クラスおよびそのサブクラスを基に作成するオブジェクトのことです。コンテントを保持できます。

文書間リレーション（Relationship）

文書と文書の関連づけです。

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。

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トを，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に登録されたレンディションのうち，主要なレンディションのことです。マルチレンディション文書を参照・更新する場合には，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ときは，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。

マッピングセット定義ファイル（XMS）

HiRDB Adapter for XML が使用する，登録されたマッピング定義の一覧を管理するファイルです。

マッピング定義ファイル（XMP）

HiRDB Adapter for XML が使用できる文法に変換したマッピング定義が記述されているファイルです。ユーザ作成のプロパティマッピング定義ファイル（DPM）とマッピング元 XML タグ定義ファイル（DCD）から生成します。

マッピング元 XML タグ定義ファイル（DCD）

登録する XML 文書の構造を DCD の形式で記述したファイルです。

マルチファイル管理機能

一つの文書に，複数のファイルを登録して管理する機能です。

マルチファイル文書

データベースに登録されている複数のファイルをコンテントとして持つ文書のことです。バージョンなし文書またはバージョン付き文書は，マルチファイル文書として扱えます。

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ト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トを保持する文書です。バージョンなし文書またはバージョン付き文書は，マルチレンディション文書として扱えます。

メタ情報ファイル
DocumentBroker が利用する DMA のクラス，プロパティ，検索オペレータなどの詳細情報を定義したファイルです。クラスおよびプロパティを追加，変更する場合に使用できます。

メタデータ（metadata）
クラス，プロパティおよびオペレータに関する詳細情報を定義するデータです。

メタデータ空間
単一の継承関係を形成するクラスの集合のことです。

（ヤ行）

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，アクセス権の判定に使用されます。

（ラ行）

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集合の要素（文書）をソートして出力する検索です。

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トを管理し，文書のプロパティおよびコンテント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トとして持つ文書のことです。データベースでは，文書のプロパティとコンテントの格納先の情報を管理しています。バージョンなし文書または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扱えます。

リレーション
DMA オブジェクトの Relationship オブジェクトに相当します。文書と文書の間の参照関係を表すオブジェクトです。

リレーション先文書
文書間リレーションを設定される，参照先になる文書です。

リレーション識別子
文書間に設定されたリレーションを識別するための識別子です。
リレーションを削除したり，リレーションのプロパティを更新したりする時に使用します。この識別子は，リレーションを設定した時に DocumentBroker によって設定される識別子です。同じ文書から同じ文書に対して二つのリレーションを設定した場合は，それぞれ異なるリレーション識別子が設定されます。

リレーション元文書
文書間リレーションを設定する元になる文書です。

リンク
DMA オブジェクトの DirectContainmentRelationship オブジェクト，ReferentialContainmentRelationship オブジェクトまたは VersionTraceableContainmentRelationship オブジェクトによって表される，文書とコンテナの関連づけです。

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。

リンク識別子
コンテナオブジェクトへのリンク（関連づけ）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を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時に使用します。
この識別子は関連づけをした時に DocumentBroker によって設定される識別子です。同じコンテナオブジェクトに，同じ要素を 2 度関連づけした場合は，それぞれ異なるリンク識別子が設定されます。

レンディション
文書のコンテントの形式およびそのコンテントをあわせてレンディションと呼びます。DMA オブジェクトの Rendition オブジェクトおよび ContentTransfer オブジェクトに相当します。

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ト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できます。
DocumentBroker では，レンディションタイプとして，MIME 名を指定することを推奨しています。

レンディションのコンテント種別変換機能
レンディションのコンテントの格納先を，最適な格納先へ変換することができる機能です。

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トの文書形式を変換して，サブレンディションのコンテントを作成，登録することです。

ローカル ACL
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。VariableArray 型のプロパティとして設定されます。

ロケール（Locale）
言語や使用する文字コードの種別，特定の国や地域で特別な意味を持つ属性などの定義のことです。ロケールは地域化した形でアプリケーションの拡張性を提供するために使われます。

索引

記号

_HIEDMS_LCKINF_ARCHIVE 403
_HIEDMS_LCKINF_DIR 402
_HIEDMS_LCKWATCH_TIME 403

A

AccessControl エントリ 313
ACE 514
ACFlag 514
ACL 86，514
AclibClassName エントリ 302
AcLogFileCount エントリ 211
AcLogFileSize エントリ 212
AcLogLevel エントリ 210
AcLogUse エントリ 210
AND-NOT 検索 514
AND 検索 24，514
API 514
application.ini 324
AvBindPublicACLCount エントリ 314
AvCntrCountPerCntr エントリ 303
AvCntrTreeHeight エントリ 303
AvComponentCount エントリ 305
AvContentSize エントリ 304
AvElementCount エントリ 307
AvLength エントリ 307
AvLocalACECount エントリ 314
AvMultiFileCount エントリ 315
AvRefCountPerCntr エントリ 303
AvRefDocCount エントリ 305
AvRenditionCount エントリ 304
AvSecurityACECount エントリ 314
AvVersionCount エントリ 303

B

BatchSizeHint エントリ 210
BlobGettingMethod エントリ 217
BlobSubstrMode エントリ 216
BlobSubstrThreshold エントリ 217

C

C++ クラスライブラリ 29
CD-ROM からのインストール 122
ClassType エントリ 302
ComponentClassName エントリ 305
ConfigurationHistory オブジェクト 514
Containable オブジェクト 514
Containee 514
Container 514
ContainerVersion オブジェクト 514
Container オブジェクト 514
ContentElement 515
ContentFileLink オブジェクト 515
ContentReference オブジェクト 515
ContentTransfers オブジェクト 515
ContentTransfer オブジェクト 515
ContentType エントリ 305
CORBA 515
Cosminexus 38，41
Count エントリ 200

D

DABroker および DABroker for C++ 31
DataType エントリ 306
DAT からのインストール 122
DBConnectionClose エントリ 287
DBConnectionPoolCount エントリ 209
DBConnectionPoolDynamic エントリ 209
DBConnectionPoolOver エントリ 209
DBConnectionPoolTiming エントリ 209
DBConnectionPoolWaitTimeOut エントリ 210
DBConnectionScope エントリ 210
DbLockWatcher エントリ 201
dbrAuthenticateUser 473
dbrFinalizeLibrary 476
dbrFinalizeSession 475
dbrGetUserInformation 474
dbrinitLibrary 472
dbrinitSession 472
dbrSetDocSpaceCharacterSet 477
DbType エントリ 208
DB コネクションプール 142
DB コネクションプール機能 142
DCD 515
DefaultACFlagEveryone エントリ 225
DefaultACFlagGroup エントリ 225
DefaultACFlagOwner エントリ 225
DefaultValue エントリ 306
DIT 136
DMA 515
DmaClassName エントリ 303

dmaProp_Cardinality プロパティ 240，244，250
dmaProp_DataType プロパティ 243，250
dmaProp_DescriptiveText プロパティ 243，250，251，252
dmaProp_DisplayName プロパティ 242，243，250，251，252
dmaProp_Head プロパティ 260
dmaProp_Ids プロパティ 243，250，251，252
dmaProp_IsHidden プロパティ 241，244，248
dmaProp_IsOrderable プロパティ 240，244
dmaProp_IsSearchable プロパティ 240，244，247
dmaProp_IsSelectable プロパティ 240，244，247
dmaProp_IsValueRequired プロパティ 241，244
dmaProp_MaximumElements プロパティ 248，250
dmaProp_MaximumLengthStringプロパティ 241，244
dmaProp_OIID プロパティ 259
dmaProp_ParentContainer プロパティ 259
dmaProp_PropertyDefaultBoolean プロパティ 241
dmaProp_PropertyDefaultInteger32 プロパティ 241
dmaProp_PropertyDefaultString プロパティ 241
dmaProp_QueryOperatorDescriptions プロパティ 240，244
dmaProp_RequiredClass プロパティ 250
dmaProp_Tail プロパティ 260
dmaProp_This プロパティ 259
dmaProp_VersionSeries プロパティ 260
dmaProp_Version プロパティ 260
DMA URL 515
DMA オブジェクト 50，515
DN 137
docaccess.ini 224
DOCBROKERDIR[Linux の場合] 134
DOCBROKERDIR〔AIX の場合〕 132
docspace.ini 196
DocSpaceCharacterSet エントリ 316
DocSpaceOrbBoaOption エントリ 200
DocSpaceVBProperty エントリ 201
DocTblName エントリ 317
DocumentBroker Collector 40，502
DocumentBroker Development Kit 28
DocumentBroker Development Kit（クライアント開発環境） 34
DocumentBroker Development Kit で開発したクライアントを構成するプログラム（C++ クラスライブラリの場合） 33
DocumentBroker Development Kit で開発したクライアントを構成するプログラム（Java クラスライブラリの場合） 37
DocumentBroker Integrated Search Suite 502
DocumentBroker Integrated Search Suite for J 40
DocumentBroker Life Cycle Suite 33
DocumentBroker Life Cycle Suite Development Kit（クライアント開発環境） 37
DocumentBroker Life Cycle Suite Runtime（クライアント実行環境） 37
DocumentBroker Object Loader 32
DocumentBroker Rendering Option 36
DocumentBroker Runtime（クライアント実行環境） 34
DocumentBroker Server 31
DocumentBroker Standard GUI 40，41
DocumentBroker Standard GUI〔用語解説〕 515
DocumentBroker Standard GUI を使用する場合のクライアントを構成するプログラム 40
DocumentBroker Text Search Index Loader 33
DocumentBroker Web Client 501
DocumentBroker イメージ登録 for DocuCentre 36
DocumentBroker イメージ登録 for imagio and IPSiO 36
DocumentBroker イメージ登録 for iR 36
DocumentBroker イメージ登録 for Konica 37
DocumentBroker イメージ登録 文字認識オプション 37
DocumentBroker が管理できる XML 文書 82
DocumentBroker サーバを構成するプログラム 29
DocumentBroker 実行環境ディレクトリの作成 131
DocumentBroker 実行環境の情報の登録 406
DocumentBroker テキスト分析コンポーネント 40
DocumentBroker テキスト分析テンプレート 40
DocumentBroker で提供する XML 文書管理機能 77
DocumentBroker の概要 2
DocumentBroker の起動 409
DocumentBroker の起動方法 330
DocumentBroker の機能 7
DocumentBroker のクラス定義の変更 339
DocumentBroker のシステム構成 28
DocumentBroker の実行環境作成コマンドの実行 135
DocumentBroker の実行環境ディレクトリの構成（AIX の場合） 421
DocumentBroker の実行環境ディレクトリの構成（Linux の場合） 426
DocumentBroker の実行環境の作成 131
DocumentBroker の実行環境の作成と削除 407
DocumentBroker の終了 410
DocumentBroker の終了方法 331
DocumentBroker の特長 4
DocumentBroker の目的 2

DocumentBroker 用データベース定義文の作成 388
DocumentSpace 構成定義ファイル 196，515
DocVersion オブジェクト 516

E

EDMAddMeta 375
EDMCBuildDocSpace 378
EDMCDefDocSpace 380
EDMChangeACL 454
EDMChangeDBDefName 456
EDMChangeFileLink 465
EDMChangeMultiFile 460
EDMChangeRefFile 463
EDMChangeVarray 458
EDMChkMeta 382
EDMChkTbl 384
edmClass_Relationship クラス 516
EDMCreateIds 385
EDMCrtSimMeta 387
EDMCrtSql 388
EDMDelMeta 390
EDMGetRas 392
EDMGetRasCL 394
EDMInitMeta 396
EDMLckWatcher 401
EDMPrintMeta 403
edmProp_ACLId プロパティ 261
edmProp_BindObject プロパティ 261
edmProp_ConceptStIndex プロパティ 72，73
edmProp_ConceptTextIndex プロパティ 72，73
edmProp_ContentIndexStatus プロパティ 72，73
edmProp_Content プロパティ 72
edmProp_DocLength プロパティ 72，73
edmProp_HeadVTConfigurationHistory プロパティ 260
edmProp_OwnerId プロパティ 261
edmProp_Parent プロパティ 260
edmProp_PrimaryGroupId プロパティ 261
edmProp_RawScore プロパティ 72，73
edmProp_ReferenceType プロパティ 261
edmProp_RenditionStatus プロパティ 260
edmProp_ScoreConcept プロパティ 73
edmProp_Score プロパティ 72，73
edmProp_StIndex プロパティ 72
edmProp_TailVTConfigurationHistory プロパティ 260
edmProp_TextIndex プロパティ 72
edmProp_VTVersionSeries プロパティ 260
EDMRefresher 404
EDMRegEnvId 406
EDMSetup 407
edmSQL 516
edmSQL 検索 516
EDMStart 409
EDMStop 410
EDMUsrView 410
EDMXmlMap 414
EnbFncFlagOfObjectOperation エントリ 216
Enumeration 型 70
ErrChkFlagOfObjectOperation エントリ 216
ErrLogFileCount エントリ 201
ErrLogFileSize エントリ 201
EXTSHM〔AIX の場合〕 132

F

FAM 516
FileLink エントリ 315
File Link 文書 18，516
File Link 連携機能 18，516
File Link 連携機能を使用するためのデータベース移行 464
File Link 連携機能を使用する場合に必要なプログラム 32，35
File Link 連携機能を使用する場合の設定 182
File Link 連携機能を使用する場合のファイルサーバ容量の見積もり 114
FtpBufferSize エントリ 213
FtpOrbBoaOption エントリ 283
FtpProcessOrbBoaOption エントリ 283
FtpProcessVBProperty エントリ 283
FtpSessionMax エントリ 283
ftpsv.ini 282
FtpSvSetup 173
FtpVBProperty エントリ 283

G

getrascustom.ini 359，361
getrascustom.ini の格納ディレクトリ（クライアント側）361
getrascustom.ini の格納ディレクトリ（サーバ側）359
GroupIDMaxLength エントリ 316
GUID 236，516
GUID エントリ 306

H

HACMP によるクラスタリングシステムでの運用 429
HAX 定義ファイル 183，185
HiRDB 148
HiRDB/Run Time 31
HiRDB Adapter for XML 36
HiRDB File Link 32，35
HiRDB Object Option 32
HiRDB Text Search Plug-in 32
HiRDB Text Search Plug-in Conceptual Extension 32
HiRDB Text Search Plug-in での環境設定 149
HiRDB サーバ 31
HiRDB の繰り返し列を使用するためのデータベース移行 458
HiRDB のユーザ権限の設定 149

I

ID ファイル 516
IndependentPersistence オブジェクト 64
IndexName エントリ 307
ini ファイルのシンタクスの基本項目 478
InstanceCount エントリ 303
IsExcept エントリ 308
IsUnique エントリ 307

J

Java クラスライブラリ 29

L

LANG〔AIX の場合〕131
LANG〔Linux の場合〕133
LD_LIBRARY_PATH〔Linux の場合〕134
LDAP 516
LdapBindPassword エントリ 208
LdapBindUserDN エントリ 208
LdapClientLib エントリ 203
LdapGroupCase エントリ 208
LdapGroupClass エントリ 206
LdapGroupFilterLeft エントリ 206
LdapGroupFilterRight エントリ 206
LdapGroupFromUserAttr エントリ 207
LdapGroupFromUserDn エントリ 207
LdapGroupIdAttrFromUserAttr エントリ 207
LdapGroupIdFromUserAttr エントリ 207
LdapGroupIdFromUserDn エントリ 208
LdapGroupId エントリ 206
LdapGroupIsDnFromUserAttr エントリ 207
LdapGroupMember エントリ 207
LdapGroupRoot エントリ 206
LdapGroupScope エントリ 206
LdapGroupTimeout エントリ 208
LdapGroup エントリ 205
LdapHost エントリ 204
LdapPort エントリ 204
LdapPrefixDn エントリ 204
LdapUserCase エントリ 205
LdapUserClass エントリ 205
LdapUserFilterLeft エントリ 205
LdapUserFilterRight エントリ 205
LdapUserId エントリ 204
LdapUserRoot エントリ 205
LdapUserScope エントリ 205
LdapUserTimeout エントリ 204
LDAP 対応のディレクトリサービス 32
LDAP 対応のディレクトリサービスによるユーザ管理機能を使用する場合 136
LDAP 対応のディレクトリサービスのユーザ認証 137
LIBPATH〔AIX の場合〕132
LifeCycleSuiteConnection エントリ 217
List 型 70
LogModeType エントリ 314

M

MaxComponentCount エントリ 305
MaxContentSize エントリ 304
MaxElementCount エントリ 307
MaxLength エントリ 307
MaxVersionCount エントリ 303
MetaIdxName エントリ 316
MetaTblName エントリ 316
MultiFileMaxCount エントリ 315
MultiRelationIndex エントリ 315

N

NgramIdxName エントリ 317
NODISCLAIM〔AIX の場合〕132
nofile（Linux の場合）222
nofiles（AIX の場合）222
NOT 検索 24
NOT 条件 516

O

ObjType エントリ 301

OIID 516
OIIDPropertyDescription プロパティ 259
OrbBoaOption エントリ 284，287
OrbOption エントリ 325，326
OrderType エントリ 307
OR 検索 24，516
OS 35，39，42，501

P

Parent 516
PctOfRefDoc エントリ 305
PDDIR 402
PdHost エントリ 208
PdNamePort エントリ 209
PdTSPluginOwner エントリ 210
PdUser エントリ 209
Preprocessing Library for Text Search 36
ProcessOrbBoaOption エントリ 202
ProcessVBProperty エントリ 202
Process エントリ 202
PropertyName エントリ 306
PSALLOC〔AIX の場合〕132
PublicACLCount エントリ 314

Q

QueryOperand クラス定義 494
QueryOperator クラス定義 492

R

RD エリア 517
RD エリア構成定義ファイル 230
RD エリア定義情報ファイル 151，257
ReferenceFile エントリ 314
RefreshTiming エントリ 215
Rendition オブジェクト 517
Reservation オブジェクト 517
RootCntrCount エントリ 303

S

Scalar 型 70，517
SecurityAdmin エントリ 224
SelectServerInMultiServer エントリ 213
SerialId エントリ 202
SessionMax エントリ 202
SessionTimeOut エントリ 202
SgmlTextName エントリ 317
SORT 31
SpPropDefaultValue エントリ 316
SRefreshGraceTime エントリ 214
SRefreshLimit エントリ 214
StructClassName エントリ 307
SuperClassType エントリ 305
SysIdxName エントリ 317
SysTblName エントリ 316
SystemManager オブジェクト 517
System オブジェクト 517

T

TargetType エントリ 303
ThisPropertyDescription プロパティ 259
TPBroker 30，35，39，501
TPBroker システムの構築 130
TPBroker での環境設定 130
TZ〔AIX の場合〕131
TZ〔Linux の場合〕134

U

UNIX のパスワードファイルによるユーザ管理機能を使用 139
UOC 140，517
UOCLibrary エントリ 204
UserAuthentication エントリ 203
UserIDMaxLength エントリ 315
userperm.ini 228
UserPermDefFile エントリ 225
UsrIdxName エントリ 317
UsrTblName エントリ 317
UTF-8 の場合に使用できる機能 497
UTF-8 の場合に使用できるコマンド 498

V

VariableArray 型 70，517
VariableArray 型のプロパティが使用するメモリ所要量の見積もり 97
VariableArray 型のプロパティの追加方法 246
VArrayElementCheck エントリ 215
VBProperty エントリ 284，287，325
Versionable オブジェクト 517
VersionDescription オブジェクト 517
VersionSeries オブジェクト 517
VersionTracedComponentDocVersion オブジェクト 517
VersionTracedDocVersion オブジェクト 517
VTContainee 59，60，517
VTContainer 59，60，518

W

W3C 518
Web Page Generator Enterprise 501
WWW サーバ 38，41

X

XDK_HOME〔AIX の場合〕133
XDK_HOME〔Linux の場合〕134
XDK_SHMEM_SIZE〔AIX の場合〕133
XDK_SHMEM_SIZE〔Linux の場合〕135
XdkShmemManage エントリ 212
XdkShmemSize エントリ 212
XML インデクスデータ作成機能 80，518
XML 定義ファイル 183
XML 定義ファイルの DocumentBroker クライアント環境への転送 187
XML 定義ファイルの関連 183
XML 定義ファイルの追加 / 更新 / 削除 414
XML 定義ファイルの登録 185
XML 定義ファイルを作成する場合の注意事項 190
XML ファイル 19
XML ファイルの形式 82
XML プロパティマッピング機能 77，518
XML 文書 19
XML 文書管理機能 19
XML 文書管理機能の環境設定 186
XML 文書管理機能を使用するための操作 187
XML 文書管理機能を使用する場合に必要なプログラム 36
XML 文書管理機能を使用する場合の設定 183
XML 文書管理機能を利用したシステム構築の検討 77
XML 文書の管理に必要なオブジェクト 62

あ

アクセス権 518
アクセス制御エレメント 86，518
アクセス制御機能 20，518
アクセス制御機能付き検索 518
アクセス制御機能の設定 146
アクセス制御機能を使用したシステム構築の検討 85
アクセス制御機能を使用するためのデータベース移行 453
アクセス制御機能を使用する場合に必要なプログラム 32
アクセス制御機能を使用する場合の設定 146
アクセス制御機能を使用する場合のデータベース容量の見積もり 110
アクセス制御情報 518
アクセス制御情報の概要 85
アクセス制御情報変更権 89，518
アクセス制御フラグ 86，518
アクセス制御モデル 518
アクセス制御リスト 86，518
アクセス制御を使用するための管理者 146
アクセスログおよびトレースログの統計・解析機能 22
アクセスログに関する運用 363
アドレス呼び出し 518
アプリケーション・サーバ 431
アンインストール 124
アンバインド 88，519

い

移行プロパティ定義ファイル 459
移行プロパティ定義ファイルの記述形式 459
移行プロパティ定義ファイルの記述例 459
異表記展開検索 519
異表記展開検索〔文書に対する全文検索〕22
異表記展開検索〔文字列型プロパティに対する全文検索〕23
イメージ文書登録機能を使用する場合に必要なプログラム 36
イメージ文書の登録機能 25
インスタンス 67
インストール 121
インストールとアンインストール 121
インデクス情報ファイル 152，265，519
インデクス情報ファイルの記述例 268
インデクス情報ファイルのセクションとエントリ 266
インデクス名をデータベース定義の名称に使用する場合の規則 265

え

永続オブジェクト 519
永続プロパティ 519
エラーログに関する運用 368
エントリ 136
エントリの文法 291

お

オブジェクト一括登録機能 24
オブジェクト一括登録機能を使用する場合に必要なプログラム 32
オブジェクト作成権 519

オブジェクト作成権限 147，228，519
オブジェクト操作権限 147，228，519
オブジェクト定義 495
オブジェクトとデータベースの関連 65
オブジェクトリファレンス 519
オペレーティングシステムでの環境設定 127

か

概念検索 23，519
拡張子－ MimeType 対応表 423
仮想メモリ所要量の見積もり 96
仮のバージョン識別子 519
環境設定 118
環境設定の準備 120
環境設定の流れ 118
環境変数の設定（AIX の場合） 131
環境変数の設定（Linux の場合） 133
監視スクリプト 433
関連オブジェクト 519

き

起動プロセス 43
基本コンテント更新権 520
基本コンテント参照権 520
基本削除権 520
基本単位 70，245，520
基本バージョン管理権 520
基本パーミッション 225，520
基本プロパティ更新権 520
基本プロパティ参照権 520
基本リンク権 520
近傍条件検索 520
近傍条件検索〔文書に対する全文検索〕 22
近傍条件検索〔文字列型プロパティに対する全文検索〕 23

く

組み合わせパーミッション 225，520
クライアントアプリケーション 34，38
クライアントアプリケーション動作定義機能 27，521
クライアントアプリケーション動作定義ファイル 324，521
クライアント側で発生した障害情報の取得 360
クライアント側の障害情報の取得 394
クライアントの障害情報取得カスタマイズファイル 361
クラスおよびプロパティの追加例 74
クラスタリングシステムでの運用 429
クラス定義 480
クラス定義情報ファイル 269，521
クラス定義情報ファイルの更新 269
クラス定義情報ファイルの作成 269，387
クラスの定義を変更する場合 339
クラス名やプロパティ名などをデータベース定義の名称に使用するためのデータベース移行 455
グループ識別子 138，141，144

け

継承 521
検索機能 22
検索ターム〔文書に対する全文検索〕 22
検索ターム〔文字列型プロパティに対する全文検索〕 23

こ

公開識別子－ MimeType 対応表 423
構成管理型のコンテインメント 57，58，521
構成管理コンテナ 13，58，521
構成管理コンテナの管理に必要なオブジェクト 58
構造指定検索 23，80，521
コマンドで生成する定義ファイル 185
コマンドの形式 373
コマンドの文法 375
コメントの文法 294
コンテインメント 57，521
コンテナ 11，521
コンテナ管理機能 11
コンテナの管理に必要なオブジェクト 57
コンテント 521
コンテント格納先パス 179，521
コンテント格納先ベースパス 14，179，521
コンテント種別 522
コンテントロケーション 14，522

さ

サーバ側で発生した障害情報の取得 358
サーバ側の障害情報の取得 392
サーバ監視プロセス 43
サーババインド 522
サービスプロセス 43，142
サービスプロセス監視プロセス 43
サービスプロセス定義ファイル 286
サービスプロセスを使用する場合の設定 176
削除権 522
サブクラス 67，522

サブクラスの追加 68
サブクラス名およびプロパティ名をデータベース定義の名称に使用する場合の規則 233
サブクラス名に対応する表識別子 233
サブジェクト 87，522
サブジェクト種別 522
サブジェクトタイプ 87
サブレンディション 9，522
参照型のコンテインメント 57，522
参照権 522
参照更新権 522

し

システム運用コマンド 370
システム管理者 120，522
システム管理者の登録 127
システムクラスおよびシステムプロパティの名称定義の規則 467
システムクラスの名称定義の規則 467
システム識別子ー MimeType 対応表 423
システム導入支援機能 27
システム導入支援機能での設定 162
システム導入支援コマンド 371
システムプロパティの名称定義の規則 467
実行環境識別子 27
実行環境制御ファイル 522
始動スクリプト 433
ジャーナルファイルの運用 337
出力されるデータベース定義文 155
障害情報取得カスタマイズファイル 359
障害対策 353
状態フラグ 522
所有者 523
シングルファイル文書 10，523

す

スーパークラス 523
スマートエージェントの設定 130

せ

正規化パラメタファイル 153
セキュリティ ACL 88，89，523
セキュリティ運用者 146，523
セキュリティ管理者 146，523
セキュリティ定義ファイル 146，147，224，523
セクションの文法 289
セッション 523
全文検索 523
全文検索インデクス 523
全文検索インデクス一括登録機能 24
全文検索インデクス一括登録機能を使用する場合に必要なプログラム 33
全文検索インデクスに UCS-4 の文字を使用する場合のデータベースの設定 450
全文検索機能 22
全文検索機能付き文書クラス 65，523
全文検索機能付き文書クラスの追加 70
全文検索機能付き文書クラスを作成する場合に必要なオブジェクト 65
全文検索機能付き文字列型プロパティ 23，252
全文検索機能付き文字列型プロパティの追加方法 253
全文検索機能を使用する場合に必要なプログラム 31

そ

相対識別名 137
属性検索機能 22

た

種文章 23

ち

チェックアウト 9，523
チェックイン 9，524
直接型のコンテインメント 57，524

つ

追加するクラスおよびプロパティの検討 67

て

定義情報ファイル 151，233，524
定義ファイルのカスタマイズ 188
定義ファイルの作成・生成の概要 183
停止スクリプト 433
停止プロセス 43
ディレクトリエントリ 136
ディレクトリ構成（AIX の場合）420
ディレクトリ構成（Linux の場合）425
ディレクトリサービス 524
ディレクトリサービスの情報管理の概念 136
データ型の対応 69
データベース移行 453
データベース運用コマンド 370
データベース運用上の注意事項 336
データベースシステムの設定に必要なファイル 151

データベース定義の名称定義の規則 150
データベース定義の名称定義の方法 150
データベースで発生したデッドロック情報およびタイムアウト情報の取得 362
データベースの運用 334
データベースのデッドロックおよびタイムアウトの監視 401
データベースの表・列の確認 384
データベースの文字コード種別の設定 76
データベース容量の見積もり 99
データベース容量の見積もり方法 99
適用業務の検討 48
電子署名機能およびタイムスタンプ機能を使用する場合に必要なプログラム 42

と

同義語展開検索 524
同義語展開検索〔文書に対する全文検索〕 22
同義語展開検索〔文字列型プロパティに対する全文検索〕 23
動作環境メタ情報ファイル 232，422，427，524
匿名バインド 524
独立データ 20，524
独立データの管理機能 20
独立データの管理に必要なオブジェクト 64
特権 524
トップオブジェクト 50，524
トラブルシュートコマンド 372
トレースファイル 353

は

バージョン 8
バージョン管理権 524
バージョン付き構成管理コンテナ 13，524
バージョン付き文書 9，525
バージョン付き文書の管理に必要なオブジェクト 53
バージョンなし構成管理コンテナ 13，525
バージョンなしコンテナ 13
バージョンなし文書 8，525
バージョンなし文書の管理に必要なオブジェクト 52
パーミッション 87，525
バインド 88，525
バックアップと回復 334
バックアップと回復の方法 334
バックアップの取得 337
パブリック ACL 87，525
パラメタのカスタマイズ 127

ひ

非ナル値制約 525

ふ

ファイルサーバ 525
ファイルシステムのバックアップ 351
ファイルシステムのリストア 351
ファイル転送機能 26
ファイル転送サービス環境定義ファイル 282
ファイルの種類 194
フィルタリング定義ファイル 185，186
フィルタリング定義ファイル（TFD）525
複合データ 97，525
複数の実行環境機能 26
プライマリグループ 525
フルコントロール 525
プレーンテキスト 70
プロセス構成 43
プロパティ更新権 525
プロパティ参照権 525
プロパティ定義 481
プロパティの基本単位 70
プロパティの追加 69
プロパティマッピング定義ファイル 185，289
プロパティマッピング定義ファイル（DPM）525
プロパティ名に対応する列名 233
分散した文書の収集・検索機能 25
分散した文書の収集・検索機能を使用する場合に必要なプログラム 40，502
文書 526
文書管理モデルへのアクセス制御の適用 89
文書間リレーション 14，526
文書間リレーション管理機能 13
文書間リレーションの管理に必要なオブジェクト 61
文書空間 526
文書空間識別子の定義 135
文書空間情報ファイル 311
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機能およびコマンド 497
文書空間で使用する文字コード種別の検討 76
文書空間に接続しているユーザ一覧出力 410
文書空間の構成を変更する 223
文書空間の構築 378
文書空間の定義 142，380
文書空間の文字コード種別の設定 76
文書空間を定義する前の準備 142
文書クラス定義ファイル 298
文書の構成管理機能 12
文書の属性情報の管理機能 14

文書の登録機能とバージョン管理機能 8
文書の分析機能 26
文書のライフサイクル 6
文書のライフサイクル管理機能 26
文書のライフサイクル管理機能を使用する場合に必要なプログラム 33，37

へ

変換フラグ 526

ま

マスタレンディション 9，526
マッピングセット定義ファイル 186
マッピングセット定義ファイル（XMS） 526
マッピング定義ファイル 186
マッピング定義ファイル（XMP） 526
マッピング元 XML タグ定義ファイル 185，186
マッピング元 XML タグ定義ファイル（DCD） 526
マルチファイル管理機能 10，526
マルチファイル管理機能を使用するためのデータベース移行 460
マルチファイル文書 10，526
マルチレンディション管理機能 9
マルチレンディション管理機能を使用する場合の排他資源管理テーブルの見積り 115
マルチレンディション機能 526
マルチレンディション文書 9，526
マルチレンディション文書の管理に必要なオブジェクト 55

み

見積もり基礎情報ファイル 320

め

名称定義ファイル 457
名称定義ファイルの記述形式 457
名称定義ファイルの記述例 457
メタ情報 232
メタ情報管理用に確保する共用メモリセグメントサイズ 496
メタ情報の記述内容 478
メタ情報の記述内容の確認 382
メタ情報の削除 390
メタ情報の初期設定 396
メタ情報の追加 375
メタ情報の追加・削除コマンドの動作 347
メタ情報ファイル 151，232，527
メタ情報ファイルの出力 403
メタデータ 527
メタデータ空間 527
メモリ所要量とディスク占有量の見積もり 96

ゆ

ユーザ LOB 用 RD エリアの容量の見積もり 107
ユーザ管理機能の設定 136
ユーザ権限 147，228，527
ユーザ権限定義ファイル 228，527
ユーザ作成のアクセスルーチンを使用する場合 140
ユーザ識別子 138，140，143，145
ユーザ情報 138，143，527
ユーザ情報の取得方法 143
ユーザ定義識別子ファイル 277
ユーザ定義識別子ファイルの作成 385
ユーザ定義識別子変数ファイル 277
ユーザ用 RD エリアの容量の見積もり 99

ら

ランキング検索 23，527

り

リストアの方法 337
リストオブジェクト 490
リファレンスファイル管理機能 14，527
リファレンスファイル管理機能を使用するためのデータベース移行 462
リファレンスファイル管理機能を使用する場合の運用 351
リファレンスファイル管理機能を使用する場合の設定 179
リファレンスファイル管理機能を使用する場合のファイルシステムのディスク容量の見積もり 113
リファレンスファイル文書 14，527
リフレッシュ 44
リレーション 527
リレーション先文書 14，527
リレーション識別子 527
リレーション元文書 14，528
リンク 528
リンク権 528
リンク識別子 528

れ

列名 468
レプリケーション 337
レンディション 9，528
レンディションタイプ 528

レンディションのコンテント種別変換機能 15，528
レンディション変換 528
レンディション変換機能 24
レンディション変換要求機能 25
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 36

ろ

ローカル ACL 87，528
ロケール 528