

Oracle Internet Directory

アプリケーション開発者ガイド

リリース 9.0.2

2002 年 7 月

部品番号 : J05909-01

ORACLE®

Oracle Internet Directory アプリケーション開発者ガイド, リリース 9.0.2

部品番号 : J05909-01

原本名 : Oracle Internet Directory Application Developer's Guide, Release 9.0.2

原本部品番号 : A95193-01

原本著者 : Torrance Brooksfüller, Sheryl Edwards, Richard Smith

原本協力者 : Ramakrishna Bollu, Saheli Dey, Bruce Ernst, Rajinder Gupta, Ashish Kolli, Stephen Lee, David Lin, Radhika Moolky, David Saslav, Valarie Moore

Copyright © 2001, 2002, Oracle Corporation. All rights reserved.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておられません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

目次

はじめに	ix
------------	----

1 概要

Oracle Internet Directory Software Developer's Kit リリース 9.0.2 の概要	1-2
Oracle Internet Directory Software Developer's Kit のコンポーネント	1-2
Oracle Internet Directory のその他のコンポーネント	1-2
サポートされるオペレーティング・システム	1-3

第 I 部 標準的な LDAP API

2 概念

LDAP の歴史	2-2
LDAP モデルの概要	2-2
LDAP ネーミング・モデル	2-2
LDAP 情報モデル	2-4
LDAP 機能モデル	2-5
LDAP セキュリティ・モデル	2-6
Oracle Internet Directory API の概要	2-10
LDAP セッションの初期化	2-12
C API を使用したセッションの初期化	2-12
DBMS_LDAP を使用したセッションの初期化	2-13
C API での LDAP セッション・ハンドル・オプション	2-14
ディレクトリ・サーバーへの認証の有効化	2-15
C API を使用したディレクトリ・サーバーへの認証の有効化	2-15
DBMS_LDAP を使用したディレクトリ・サーバーへの認証の有効化	2-15

DBMS_LDAP を使用した検索	2-17
検索関連操作の流れ	2-18
検索有効範囲	2-20
フィルタ	2-21
DBMS_LDAP を使用したセッション終了の有効化	2-22

3 Oracle Internet Directory の C API

Oracle Internet Directory C API の概要	3-2
Oracle Internet Directory SDK C API SSL 拡張機能	3-2
C API リファレンス	3-4
LDAP C API の概要	3-4
ファンクション	3-7
LDAP セッションの初期化	3-8
LDAP セッション・ハンドル・オプション	3-9
コントロールの使用	3-14
ディレクトリに対する認証	3-16
セッションのクローズ	3-19
LDAP 操作の実行	3-20
操作の中止	3-38
結果の取得と LDAP メッセージの確認	3-40
エラーの処理と結果の解析	3-42
結果リストの参照	3-46
検索結果の解析	3-47
C API の使用例	3-55
SSL モードでの C API の使用方法	3-55
非 SSL モードでの C API の使用方法	3-56
C API を使用したアプリケーションの作成	3-57
必要なヘッダー・ファイルとライブラリ	3-57
サンプル検索ツールの作成	3-57
依存性と制限事項	3-70

4 DBMS_LDAP PL/SQL パッケージ

DBMS_LDAP パッケージの概要	4-2
DBMS_LDAP を使用したアプリケーションの作成	4-2
依存性と制限事項	4-2

DBMS_LDAP サンプル・プログラム	4-3
DBMS_LDAP リファレンス	4-3
サブプログラムの概要	4-3
例外の概要	4-6
データ型の概要	4-8
サブプログラム	4-9

第 II 部 LDAP API に対する Oracle の拡張機能

5 Oracle の拡張機能の概要

LDAP アクセス・モデル	5-2
アプリケーションのインストール・ロジック	5-3
アプリケーションの起動とブートストラップに関するロジック	5-3
アプリケーション実行時のロジック	5-3
アプリケーションの停止ロジック	5-4
アプリケーションの削除ロジック	5-4
LDAP のモデル化されたエンティティ	5-4
ユーザー	5-5
グループ	5-5
サブスクライバ	5-5
API の拡張機能 : 概要と使用モデル	5-6
API の拡張機能 : 前提	5-6
システムの配置	5-7
API の拡張機能 : 機能の分類	5-7
API の拡張機能 : 使用モデル	5-8
インストールと最初の使用	5-11

6 Oracle Internet Directory の Java API

クラスの説明	6-2
User クラス	6-2
Subscriber クラス	6-3
Group クラス	6-4
PropertySetCollection クラス、PropertySet クラスおよび Property クラス	6-5
クラス	6-6
oracle.ldap.util.Base64	6-7

oracle.ldap.util.Group	6-9
oracle.ldap.util.Guid	6-11
oracle.ldap.util.LDIF	6-14
oracle.ldap.util.LDIFAttribute	6-16
oracle.ldap.util.LDIFMigration	6-25
oracle.ldap.util.LDIFReader	6-29
oracle.ldap.util.LDIFRecord	6-32
oracle.ldap.util.LDIFSubstitute	6-35
oracle.ldap.util.LDIFWriter	6-36
oracle.ldap.util.Property	6-41
oracle.ldap.util.PropertySet	6-42
oracle.ldap.util.PropertySetCollection	6-44
oracle.ldap.util.Subscriber	6-46
oracle.ldap.util.User	6-49
oracle.ldap.util.Util	6-54
java.util.Hashtable getAllDASUrl(DirContext ctx)	6-57
oracle.ldap.util.jndi.ConnectionUtil	6-64
例外	6-66
oracle.ldap.util.AcctIPLockedException	6-67
oracle.ldap.util.AcctTotallyLockedException	6-68
oracle.ldap.util.AuthFailureException	6-68
oracle.ldap.util.AuthPasswdChangeWarningException	6-68
oracle.ldap.util.AuthPasswdExpiredException	6-69
oracle.ldap.util.GeneralErrorException	6-69
oracle.ldap.util.InvalidLDIFRecordException	6-70
oracle.ldap.util.InvalidParameterException	6-71
oracle.ldap.util.InvalidRootOrclctxException	6-72
oracle.ldap.util.InvalidSubscriberOrclctxException	6-73
oracle.ldap.util.LoginPolicyFailureException	6-74
oracle.ldap.util.MigrationException	6-74
oracle.ldap.util.MultipleSubscriberException	6-75
oracle.ldap.util.MultipleUserException	6-76
oracle.ldap.util.NoGroupMembersException	6-77
oracle.ldap.util.NoRootOrclctxException	6-77
oracle.ldap.util.NoSubscriberOrclctxException	6-78
oracle.ldap.util.NoSuchGroupException	6-79
oracle.ldap.util.NoSuchSubscriberException	6-80
oracle.ldap.util.NoSuchUserException	6-80

oracle.ldap.util.ParameterException	6-81
oracle.ldap.util.PasswdExpiredException	6-82
oracle.ldap.util.SetPropertiesException	6-82
oracle.ldap.util.SubscriberNotFoundException	6-83
oracle.ldap.util.UtilException	6-84

7 DBMS_LDAP_UTL PL/SQL パッケージ

概要	7-2
DBMS_LDAP_UTL リファレンス	7-3
サブプログラムの概要	7-3
ユーザー関連サブプログラム	7-5
グループ関連サブプログラム	7-21
サブスライバ関連サブプログラム	7-27
その他のサブプログラム	7-32
ファンクション・リターン・コードの概要	7-42
データ型の概要	7-46

8 プロビジョニング統合アプリケーションの開発

前提となる知識	8-2
プロビジョニング統合のための使用モデルの開発	8-2
プロビジョニング統合に関するタスクの開発	8-4
アプリケーションのインストール	8-5
ユーザーの作成と登録	8-5
ユーザーの削除	8-5
アプリケーションの削除	8-7
プロビジョニング・イベント・インタフェースの説明	8-7
LDAP_NTFY ファンクションの定義	8-9
event_ntfy ファンクション	8-10

第 III 部 コマンドライン・ツール

9 コマンドライン・ツールの構文

LDAP データ交換フォーマット (LDIF) の構文	9-2
コマンドライン・ツールの構文	9-4
ldapadd 構文	9-4

ldapaddmt 構文	9-6
ldapbind 構文	9-8
ldapcompare 構文	9-10
ldapdelete 構文	9-11
ldapmoddn 構文	9-13
ldapmodify 構文	9-15
ldapmodifymt 構文	9-20
ldapsearch 構文	9-22
カタログ管理ツールの構文	9-26
プロビジョニング・サブスクリプション・ツールの構文	9-28

10 Oracle Internet Directory サーバーのプラグイン・フレームワーク

概要	10-2
前提となる知識	10-2
概念	10-2
ディレクトリ・サーバー・プラグインの概要	10-2
サーバー・プラグイン・フレームワークの概要	10-3
Oracle Internet Directory でサポートされている操作ベースのプラグイン	10-4
要件	10-6
プラグインの設計	10-6
プラグインの作成	10-7
プラグインのコンパイル	10-9
プラグインの登録	10-10
プラグインの管理	10-13
プラグインの有効化と無効化	10-14
例外処理	10-14
プラグイン LDAP API	10-16
プラグインとレプリケーション	10-17
プラグインと DB ツール	10-17
セキュリティ	10-17
プラグイン LDAP API の仕様	10-18
使用モデルと例	10-18
例 1: 問合せロギングの検索	10-18
例 2: 2 つのディレクトリ情報ツリーの同期化	10-21

タイプ定義と使用モデル	10-23
データベース・オブジェクト・タイプの定義	10-23
プラグイン・モジュール・インタフェースの仕様	10-24
LDAP サーバーのエラー・コード・リファレンス	10-28

A 使用例

データベース・トリガーからの DBMS_LDAP の使用	A-2
検索用の DBMS_LDAP の使用	A-9
Java サンプル・コード	A-13
User クラスのサンプル・コード	A-13
Subscriber クラスのサンプル・コード	A-16
Group クラスのサンプル・コード	A-18
印刷のサンプル・コード	A-20

用語集

索引

はじめに

『Oracle Internet Directory アプリケーション開発者ガイド』では、C Application Program Interface (C API) および PL/SQL Application Program Interface (PL/SQL API) を利用して、アプリケーションで Oracle Internet Directory にアクセスする方法について説明します。

この章では、次の項目について説明します。

- [対象読者](#)
- [このマニュアルの構成](#)
- [関連文書](#)
- [表記規則](#)

対象読者

『Oracle Internet Directory アプリケーション開発者ガイド』は、Oracle Internet Directory サーバーに対してディレクトリ情報の格納および更新を行うアプリケーションを作成する開発者を対象としています。また、このマニュアルは、Oracle Internet Directory C API および PL/SQL API の動作の理解が必要な人も対象としています。

このマニュアルの構成

第 1 章「概要」

Oracle Internet Directory Software Developer's Kit リリース 9.0.2 が対象とする開発者と、コンポーネントについて概要を説明します。また、Oracle Internet Directory のその他のコンポーネントと、サポートしているプラットフォームのリストを示します。

第 2 章「概念」

C API および PL/SQL API で使用可能なすべての主要操作について簡単に概要を説明します。これによって、開発者は、API から独立した観点で Lightweight Directory Access Protocol (LDAP) の概要を理解できます。

第 3 章「Oracle Internet Directory の C API」

Oracle Internet Directory API について説明し、その使用例を紹介します。

第 4 章「DBMS_LDAP PL/SQL パッケージ」

PL/SQL プログラマが LDAP サーバーからデータへアクセスする際に使用できる DBMS_LDAP パッケージについて説明します。また、DBMS_LDAP の使用例も紹介します。

第 5 章「Oracle の拡張機能の概要」

アプリケーションをディレクトリ対応にする方法について説明します。

第 6 章「Oracle Internet Directory の Java API」

Oracle Internet Directory の Java Application Program Interface (Java API) に関する参照資料を説明します。

第 7 章「DBMS_LDAP_UTL PL/SQL パッケージ」

DBMS_LDAP という PL/SQL パッケージに同梱されている PL/SQL API について説明します。また、その使用例も紹介します。

第 8 章「プロビジョニング統合アプリケーションの開発」

Oracle Provisioning Integration Service で提供しているイベントをアプリケーションでできるようにする方法について説明します。

第 9 章「コマンドライン・ツールの構文」

LDAP データ交換フォーマット (LDIF) と LDAP コマンドライン・ツールを使用するための構文、使用方法および使用例を紹介します。

第 10 章「Oracle Internet Directory サーバーのプラグイン・フレームワーク」

Oracle Internet Directory サーバーによるカスタム開発を容易にするプラグイン・フレームワークの使用方法について説明します。

付録 A「使用例」

サンプル・コードを提供します。

用語集

関連文書

詳細は、次の Oracle マニュアルを参照してください。

- 『Oracle Internet Directory 管理者ガイド』
- 『PL/SQL ユーザーズ・ガイドおよびリファレンス』

リリース・ノート、インストレーション・マニュアル、ホワイト・ペーパーまたはその他の関連文書は、OTN-J (Oracle Technology Network Japan) に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

<http://otn.oracle.co.jp/membership/>

OTN-J のユーザー名とパスワードを取得済みであれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

<http://otn.oracle.co.jp/document/>

その他の情報は、次を参照してください。

- 『Chadwick, David, Understanding X.500—The Directory. Thomson Computer Press, 1996』
- 『Howes, Tim and Mark Smith, *LDAP: Programming Directory-enabled Applications with Lightweight Directory Access Protocol*. Macmillan Technical Publishing, 1997』
- 『Howes, Tim, Mark Smith and Gordon Good, Understanding and Deploying LDAP Directory Services. Macmillan Technical Publishing, 1999』
- Internet Assigned Numbers Authority のホームページ
<http://www.iana.org> (オブジェクト識別子の詳細)

- Internet Engineering Task Force (IETF) の次の Web サイト
 - <http://www.ietf.org>
(IETF のホームページ)
 - <http://www.ietf.org/html.charters/ldapext-charter.html>
(ldapext の Charter と LDAP Draft)
 - <http://www.ietf.org/html.charters/ldup-charter.html>
(LDUP の Charter と Draft)
 - <http://www.ietf.org/rfc/rfc2254.txt>
(「The String Representation of LDAP Search Filters」)
 - <http://www.ietf.org/rfc/rfc1823.txt>
(「The LDAP Application Program Interface」)
- <http://www.openldap.org>
(OpenLDAP Community)

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

- [本文の表記規則](#)
- [コード例の表記規則](#)

本文の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則とその使用例を示します。

表記規則	意味	例
太字	太字は、本文中に定義されている用語または用語集に記載されている用語、あるいはその両方を示します。	この句を指定する場合は、 索引構成表 を作成します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージとメソッド、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	<p>この句は NUMBER 列に対してのみ指定できます。</p> <p>BACKUP コマンドを使用すると、データベースのバックアップを作成できます。</p> <p>USER_TABLES データ・ディクショナリ・ビューの TABLE_NAME 列を問い合わせます。</p> <p>DBMS_STATS.GENERATE_STATS プロシージャを使用します。</p>

表記規則	意味	例
固定幅フォントの小文字	<p>固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニット、およびパラメータ値があります。</p> <p>注意：一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。</p>	<p>sqlplus と入力して SQL*Plus をオープンします。</p> <p>パスワードは orapwd ファイルに指定されています。</p> <p>データ・ファイルと制御ファイルのバックアップを /disk1/oracle/dbs ディレクトリに作成します。</p> <p>department_id、department_name および location_id の各列は、hr.departments 表にあります。</p> <p>初期化パラメータ QUERY_REWRITE_ENABLED を true に設定します。</p> <p>oe ユーザーで接続します。</p> <p>これらのメソッドは JRepUtil クラスに実装されます。</p>
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、ブレースホルダまたは変数を示します。	<p>parallel_clause を指定できます。</p> <p>Uold_release.SQL を実行します。old_release は、アップグレード前にインストールしたリリースです。</p>

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインを示します。次のように、固定幅フォントで、通常の本文とは区別して記載しています。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表は、コード例の記載上の表記規則と使用例を示しています。

表記規則	意味	例
[]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (digits [, precision])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。	{ENABLE DISABLE}

表記規則	意味	例
	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち1つを入力します。縦線自体は入力しないでください。	{ENABLE DISABLE} [COMPRESS NOCOMPRESS]
...	水平の省略記号は、次のどちらかを示します。 <ul style="list-style-type: none">■ 例に直接関係のないコード部分が省略されていること。■ コードの一部が繰り返し可能であること。	CREATE TABLE ...AS subquery; SELECT col1, col2, ... ,coln FROM employees;
.	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、示されているとおりに入力してください。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック	イタリックの文字は、特定の値を指定する必要があるプレースホルダまたは変数を示します。	CONNECT SYSTEM/system_password DB_NAME = database_name
大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、小文字でも入力できます。	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。 注意： 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

この章では、Oracle Internet Directory Software Developer's Kit リリース 9.0.2 の対象読者とコンポーネントについて概要を説明します。また、Oracle Internet Directory のその他のコンポーネントと、サポートしているプラットフォームのリストを示します。

この章では、次の項目について説明します。

- [Oracle Internet Directory Software Developer's Kit リリース 9.0.2 の概要](#)
- [Oracle Internet Directory Software Developer's Kit のコンポーネント](#)
- [Oracle Internet Directory のその他のコンポーネント](#)
- [サポートされるオペレーティング・システム](#)

Oracle Internet Directory Software Developer's Kit リリース 9.0.2 の概要

Oracle Internet Directory SDK リリース 9.0.2 は、C、C++ および PL/SQL を使用するアプリケーション開発者を対象とした製品です。Java 開発者は、Sun 社の JNDI プロバイダを使用して、Oracle Internet Directory サーバー内のディレクトリ情報にアクセスできます。

Oracle Internet Directory Software Developer's Kit のコンポーネント

Oracle Internet Directory Software Developer's Kit リリース 9.0.2 の構成内容は次のとおりです。

- LDAP バージョン 3 に準拠した C Application Program Interface (C API)
- PL/SQL Application Program Interface (PL/SQL API) (DBMS_LDAP という PL/SQL パッケージに同梱)
- サンプル・プログラム
- 『Oracle Internet Directory アプリケーション開発者ガイド』(このマニュアル)
- コマンドライン・ツール

Oracle Internet Directory のその他のコンポーネント

次に示す Oracle Internet Directory リリース 9.0.2 のコンポーネントは、Oracle Internet Directory Software Developer's Kit の一部ではありませんが、個別に取得できます。

- Oracle Internet Directory サーバー (LDAP バージョン 3 に準拠したディレクトリ・サーバー)
- Oracle Directory Replication Server
- Oracle Directory Manager (Java ベースの Graphical User Interface)
- Oracle Internet Directory バルク・ツール
- 『Oracle Internet Directory 管理者ガイド』

サポートされるオペレーティング・システム

Oracle Internet Directory は、サーバー、クライアントともに、次のオペレーティング・システムをサポートしています。

- Sun Solaris
- Microsoft Windows
 - Windows NT 4.0
 - Windows 98
 - Windows 2000
- HPUX
- AIX
- Compaq TRU64
- Intel Solaris
- SGI
- DGUX
- UNIXWARE

第 I 部

標準的な LDAP API

第 I 部では、主要な Lightweight Directory Access Protocol Application Program Interface (LDAP API) について説明し、基本的な LDAP プログラミングの概念、Internet Engineering Task Force C Application Program Interface (IETF C API) に関する情報、および PL/SQL Application Program Interface (PL/SQL API) に関する情報を示します。第 I 部は、次の章で構成されています。

- 第 2 章「概念」
- 第 3 章「Oracle Internet Directory の C API」
- 第 4 章「DBMS_LDAP PL/SQL パッケージ」

この章では、C Application Program Interface (C API) および PL/SQL Application Program Interface (PL/SQL API) で使用可能なすべての主要操作について簡単に概要を説明します。これによって、開発者は、API から独立した観点で **Lightweight Directory Access Protocol (LDAP)** の概要を理解できます。この章で説明する概念を把握すると、API の詳細が容易に理解できます。

この章では、次の項目について説明します。

- **LDAP の歴史**
- **LDAP モデルの概要**
- **Oracle Internet Directory API の概要**
- **LDAP セッションの初期化**
- **C API での LDAP セッション・ハンドル・オプション**
- **ディレクトリ・サーバーへの認証の有効化**
- **DBMS_LDAP を使用した検索**
- **DBMS_LDAP を使用したセッション終了の有効化**

LDAP の歴史

LDAP は、X.500 Directory Access Protocol に対する軽量フロントエンドとして開発されました。X.500 Directory Access Protocol を簡素化するため、LDAP は次のように機能します。

- TCP/IP 接続を使用します。これは、X.500 の実装に必要な OSI 通信スタックよりもはるかに軽量です。
- ほとんど使用されない X.500 Directory Access Protocol の冗長な機能を削減しています。
- 単純な書式を使用して、ほとんどのデータ要素を表現します。この書式は、複雑で高度に構造化された X.500 の表現よりも処理が簡単です。
- X.500 で使用されているエンコーディング規則の簡素化バージョンを使用して、ネットワーク上でトランスポートするデータをエンコードします。

LDAP モデルの概要

LDAP では、4 つの基本モデルを定義してその操作を記述します。この項では、次の項目について説明します。

- [LDAP ネーミング・モデル](#)
- [LDAP 情報モデル](#)
- [LDAP 機能モデル](#)
- [LDAP セキュリティ・モデル](#)

LDAP ネーミング・モデル

LDAP ネーミング・モデルによって、ディレクトリ情報を参照および編成できます。ディレクトリ内の各エントリは、**識別名**で一意に識別されます。識別名は、ディレクトリ階層におけるそのエントリの位置を正確に伝えます。この階層は、**ディレクトリ情報ツリー**によって表現されます。

識別名とディレクトリ情報ツリーの関係を理解するには、[図 2-1](#) の例を参照してください。

図 2-1 ディレクトリ情報ツリー

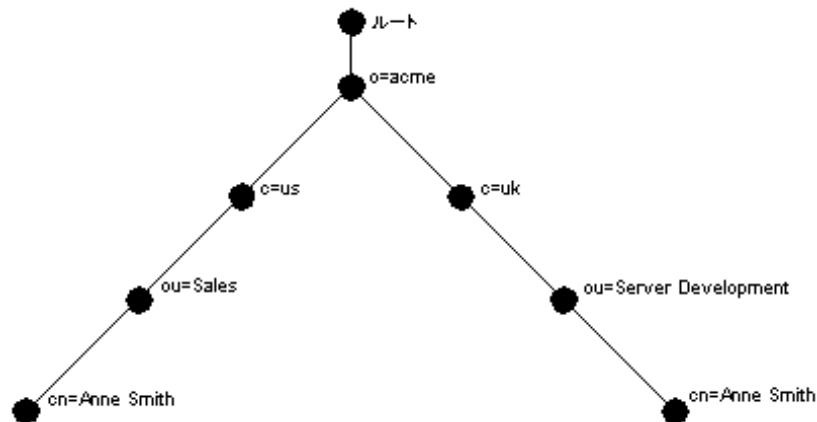


図 2-1 のディレクトリ情報ツリーは、Acme Corporation に所属する、Anne Smith という同じ名前を持つ 2 人の従業員のエントリを図示しています。この図のディレクトリ情報ツリーは、地理的および組織的な線で構造化されています。左の分岐で表されている Anne Smith は米国の販売部門に勤務し、もう一方の Anne Smith は英国のサーバー開発部門に勤務しています。

右の分岐で表されている Anne Smith には、Anne Smith という一般名 (cn) があります。彼女は、Acme という組織 (o) の、英国 (uk) という国 (c) の、サーバー開発という組織単位 (ou) に勤務しています。

この「Anne Smith」エントリの識別名は次のとおりです。

cn=Anne Smith,ou=Server Development,c=uk,o=acme

識別名の慣習的な書式では、左から最下位のディレクトリ情報ツリー・コンポーネント、続いてその次の上位コンポーネントを記述し、ルートのコンポーネントまで順に記述することに注意してください。

識別名内の最下位コンポーネントは**相対識別名**と呼ばれます。たとえば、前述の Anne Smith のエントリの相対識別名は cn=Anne Smith です。同様に、Anne Smith の相対識別名のすぐ上のエントリに対応する相対識別名は ou=Server Development、ou=Server Development のすぐ上のエントリに対応する相対識別名は c=uk です。識別名は、このように各相対識別名をカンマで区切って順に並べたものです。

ディレクトリ情報ツリー全体の中で特定エントリの位置を識別する場合、クライアントは、その相対識別名のみではなく、エントリの完全な識別名を使用することによってそのエントリを一意的に識別します。たとえば、図 2-1 のグローバル組織内では、この 2 人の Anne Smith を混同しないように、それぞれの完全な識別名を使用します。（同一組織単位内に同じ名前の従業員が 2 人いる可能性がある場合は、一意の識別番号で各従業員を識別するなど、他の方法を使用してください。）

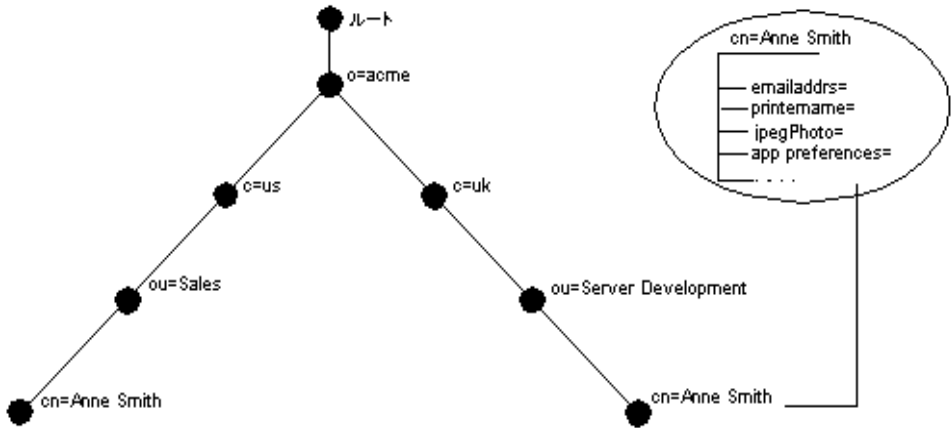
LDAP 情報モデル

LDAP 情報モデルによって、ディレクトリ内の情報の形式や文字が決まります。このモデルの中心はエントリであり、そのエントリは属性で構成されています。ディレクトリ内のオブジェクトに関する情報の各集合は**エントリ**と呼ばれます。たとえば、一般的な電話帳には個人に関するエントリ、図書館のカード式目録には本に関するエントリが含まれています。同様に、オンライン・ディレクトリには、従業員、会議室、E-Commerce パートナまたはプリンタなどの共有ネットワーク・リソースに関するエントリが含まれています。

一般的な電話帳の場合、個人に関するエントリには住所や電話番号などの情報項目が含まれます。オンライン・ディレクトリでは、このような情報項目は**属性**と呼ばれます。たとえば、一般的な従業員エントリの属性には、役職名、電子メール・アドレス、電話番号などがあります。

たとえば、図 2-2 では、英国（uk）の Anne Smith に関するエントリには、その人固有の情報を提供する各種の属性があります。これらの属性はツリーの右側の円の中にリストされています。emailaddr、printername、jpegPhoto および app preferences などの情報が記述されています。さらに、図 2-2 の各黒丸も属性を持つエントリですが、ここではそれぞれの属性は示されていません。

図 2-2 Anne Smith に関するエントリの属性



各属性は、属性の型と 1 つ以上の属性値で構成されます。**属性の型**は、その属性に含まれている情報の種類（例: jobTitle）を示します。**属性値**は、そのエントリに含まれる情報の特定の値です。たとえば、jobTitle 属性に対する値には manager があります。

LDAP 機能モデル

LDAP 機能モデルによって、情報に対して実行できる操作が決まります。次の 3 つの機能があります。

検索および読み込み	読み込み操作では、名前が判明しているエントリの属性を取得します。リスト操作では、指定したエントリの子を列挙します。検索操作では、検索フィルタと呼ばれる選択条件に基づいて、ツリー内に定義されている領域からエントリを選択します。一致した各エントリについて、要求された属性のセットが（値の有無にかかわらず）戻されます。検索対象エントリの範囲は、単一のエントリ、エントリの子またはサブツリー全体にまで広げることができます。別名のエントリは、サーバーの境界を越えている場合でも、検索時に自動的に続行されます。中止操作を定義すると、進行中の操作を取り消すことができます。
変更	<p>このカテゴリでは、ディレクトリを変更するための 4 つの操作を定義します。</p> <ul style="list-style-type: none">■ 変更: 既存のエントリを変更します。属性と値を追加および削除できます。■ 追加: エントリをディレクトリに挿入します。■ 削除: エントリをディレクトリから削除します。■ 相対識別名の変更: エントリの名前を変更します。
認証	このカテゴリではバインド操作を定義します。この操作によって、クライアントはセッションを開始し、その識別情報をディレクトリに示すことができます。単純なクリアテキストのパスワードによる認証から公開鍵ベースの認証に至るまで、様々な認証方式がサポートされています。バインド解除操作によって、ディレクトリ・セッションを終了します。

LDAP セキュリティ・モデル

LDAP セキュリティ・モデルによって、ディレクトリ内の情報を保護できます。

この項では、次の項目について説明します。

- **認証**: ユーザー、ホストおよびクライアントの識別情報が正しく検証されていることを保証する方法
- **アクセス制御と認可**: ユーザーが権限を持つ情報のみを読み込みまたは更新することを保証する方法
- **データ整合性**: 送信中にデータが変更されないことを保証する方法
- **データ・プライバシー**: 送信中にデータが開示されないことを保証する方法
- **パスワード保護**: 4つの暗号化オプションのいずれかによって、ユーザー・パスワードの保護を保証する方法
- **パスワード・ポリシー**: パスワードの使用方法を制御する規則を設定する方法

認証

認証は、ディレクトリ・サーバーが、そのディレクトリに接続しているユーザーの正確な識別情報を設定するプロセスです。このプロセスは、LDAP セッションが `ldap-bind` 操作によって確立されたときに発生します。すべてのセッションには関連付けられているユーザー ID があり、認可 ID とも呼ばれます。

ユーザー、ホストおよびクライアントの識別情報が正しく認識されることを保証するために、Oracle Internet Directory には、匿名、簡易および Secure Sockets Layer (SSL) の3つの認証オプションが用意されています。

匿名認証 ディレクトリをすべての人が使用できる場合は、ユーザーにディレクトリへの匿名ログインを許可できます。**匿名認証**を使用する場合、ユーザーは、ユーザー名とパスワードのフィールドを空白のままにしてログインします。各匿名ユーザーは、匿名ユーザーに指定されたすべての権限を使用できます。

簡易認証 この認証の場合、クライアントは、識別名とパスワード（ネットワークでの送信時には暗号化されない）によって、サーバーに対して自己認証を行います。**簡易認証**では、クライアントが送信した識別名とパスワードと、ディレクトリに格納されている識別名とパスワードが一致していることをサーバーが検証します。

Secure Sockets Layer (SSL) を使用した認証 **Secure Sockets Layer (SSL)** は、ネットワーク接続を保護するための業界標準プロトコルです。SSL は、信頼されている認証局によって検証された**証明書**を交換することによって認証を提供します。証明書は、エンティティの認証情報が正しいことを保証します。エンティティには、エンド・ユーザー、データベース、管理者、クライアントまたはサーバーが可能です。**認証局**は、すべての関係機関によって高いレベルの信頼度を与えられた公開鍵の証明書を作成する機関です。

SSL は、3 つの認証モードで使用できます。

SSL モード	説明
認証なし	クライアントとサーバーのいずれも、他方に対して自己認証を行いません。証明書の送信または交換は行われません。この場合は、SSL 暗号化・復号化のみ使用されます。
サーバー認証	ディレクトリ・サーバーのみ、クライアントに対して自己認証を行います。ディレクトリ・サーバーは、そのサーバーが認証されていることを証明する証明書をクライアントに送信します。
クライアントとサーバーの認証	クライアントとサーバーは、相互に自己認証を行います。クライアントとサーバーは、証明書を交換します。

Oracle Internet Directory 環境では、クライアントとディレクトリ・サーバー間の SSL 認証に、次の 3 つの基本ステップが含まれます。

1. ユーザーは、SSL ポート上の SSL を使用して、ディレクトリ・サーバーへの LDAP 接続を開始します（デフォルトの SSL ポートは 636 です）。
2. SSL は、クライアントとディレクトリ・サーバー間のハンドシェイクを実行します。
3. ハンドシェイクが成功すると、ディレクトリ・サーバーは、そのディレクトリにアクセスするために必要な認可をユーザーが所有していることを検証します。

関連項目： SSL の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

アクセス制御と認可

認可は、ユーザーが権限を持つ情報のみを読み込みまたは更新することを保証するプロセスです。ディレクトリ・セッション内でディレクトリ操作が行われると、ディレクトリ・サーバーは、その操作の実行に必要な権限が（セッションに関連付けられた認可 ID によって識別された）ユーザーに与えられていることを確認します。権限が与えられていない場合、操作は実行できません。この方法によって、ディレクトリ・サーバーは、ディレクトリ・ユーザーによる不正操作からディレクトリ・データを保護しています。この方法はアクセス制御と呼ばれます。

Access Control Information Item（ACI）は、アクセス制御に関連する管理ポリシーを記録したディレクトリ・メタデータです。

ACI は、ユーザー変更可能な操作属性として、Oracle Internet Directory に格納されています。通常、Access Control List（ACL）と呼ばれるこの ACI 属性値のリストは、ディレクトリ・オブジェクトと関連付けられています。このリストにある属性値によって、そのディレクトリ・オブジェクトに対するアクセス・ポリシーが管理されます。

ACI は、ディレクトリ内にテキスト文字列として記述され、格納されています。この文字列は、明確に定義された書式に従う必要があります。ACI 属性の各有効値は、個別のアクセス

制御ポリシーを表します。これらの個々のポリシーのコンポーネントは、ACI ディレクティブまたは ACI と呼ばれ、その書式は ACI ディレクティブ書式と呼ばれます。

アクセス制御ポリシーは規範的です。つまり、このポリシーのセキュリティ・ディレクティブは、[ディレクトリ情報ツリー](#)内の下位エントリすべてに適用されるように設定できます。アクセス制御ポリシーが適用される開始地点は、[アクセス制御ポリシー・ポイント \(ACP\)](#)と呼ばれます。

データ整合性

Oracle Internet Directory は、SSL を使用して、送信中にデータの変更、削除または再実行が行われなかったことを保証します。この SSL 機能は、暗号方式の保護メッセージ・ダイジェストを、[MD5](#) アルゴリズムまたは [Secure Hash Algorithm \(SHA\)](#) を使用した暗号化チェックサムによって生成し、ネットワークを介して送信する各パケットに組み込みます。

データ・プライバシー

Oracle Internet Directory は、SSL で使用可能な[公開鍵暗号](#)を使用して、送信時にデータが開示されないことを保証します。公開鍵暗号では、メッセージの送信側が受信側の公開鍵を使用してメッセージを暗号化します。メッセージが送信されると、受信側は、受信側の秘密鍵を使用してメッセージを復号化します。具体的には、Oracle Internet Directory では、SSL で使用可能な次の 2 つのレベルの暗号化をサポートします。

■ DES40

DES40 アルゴリズムは [DES](#) の変形で、国際的に使用可能な暗号化方式です。このアルゴリズムは、秘密鍵を事前に処理し、40 ビットの有効な鍵を提供します。DES40 は、米国およびカナダ以外で、DES ベースの暗号化アルゴリズムの使用を希望する顧客を対象に設計されています。この機能によって、顧客は地理的条件に関係なく使用するアルゴリズムを選択できます。

■ RC4_40

Oracle は、他の Oracle 製品が使用できる事実上すべての地域に対して、鍵のサイズが 40 ビットの RC4 データ暗号化アルゴリズムを輸出するライセンスを取得しています。この結果、国際企業は、高速暗号化を使用して事業全体を保護することが可能になります。

パスワード保護 パスワードの保護スキームは、インストール時に設定されます。この初期構成は、Oracle Directory Manager または ldapmodify のいずれかを使用して変更できます。パスワード暗号化のタイプを変更するには、スーパー・ユーザーであることが必要です。

パスワードを暗号化するために、Oracle Internet Directory では [MD4](#) アルゴリズムをデフォルトとして使用します。MD4 は、128 ビットのハッシュまたはメッセージ・ダイジェスト値を生成する一方向ハッシュ関数です。このデフォルトは、次のいずれかに変更できます。

- **MD5** – MD4 が改善され、さらに複雑になったバージョン。
- **SHA** – Secure Hash Algorithm。MD5 よりも長い 160 ビットのハッシュを生成します。このアルゴリズムは MD5 よりも若干速度が遅くなりますが、より大きなメッセージ・ダイジェストによって、総当たり攻撃や反転攻撃に対処できます。
- **UNIX Crypt** – UNIX 暗号化アルゴリズム。
- 暗号化なし

指定した値は、**ルート DSE** の `orclCryptoScheme` 属性に格納されます。この属性は単一値です。

ディレクトリ・サーバーへの認証時に、ユーザーはパスワードをクリアテキストで入力します。サーバーはそのパスワードを指定された暗号化アルゴリズムでハッシュし、`userPassword` 属性内のハッシュされたパスワードと照合して検証します。ハッシュされたパスワードの値が一致した場合、サーバーはユーザーを認証します。ハッシュされたパスワードの値が一致しない場合、サーバーはユーザーに対して無効な資格証明であるというエラー・メッセージを送信します。

パスワード・ポリシー パスワード・ポリシーとは、パスワードの使用方法を制御する規則のセットです。ユーザーがディレクトリにバインドすると、ディレクトリ・サーバーはパスワード・ポリシーを使用して、パスワードがポリシーに設定されている様々な要件を満たしていることを確認します。

パスワード・ポリシーを設定するときに、次のようなタイプの規則を設定します。

- 指定のパスワードの最大有効期間
- パスワードの最少文字数
- ユーザーが自分のパスワードを変更できるかどうか

Oracle Internet Directory API の概要

Oracle Internet Directory API は、C API および PL/SQL API として使用できます。

PL/SQL API は、DBMS_LDAP という PL/SQL パッケージに含まれています。このパッケージを使用すると、PL/SQL アプリケーションで、エンタープライズ・ワイドの LDAP サーバー内にあるデータにアクセスできます。ファンクション・コールのネーミングと構文は Oracle Internet Directory C API ファンクションに類似しており、LDAP C-API に関する [Internet Engineering Task Force \(IETF\)](#) の現在の推奨事項に準拠しています。ただし、PL/SQL API に含まれるのは、C API で使用できるファンクションの一部のみです。特に、PL/SQL API で使用できるのは、LDAP サーバーへの同期コールのみです。

図 2-3 は、クライアントのランタイム環境における DBMS_LDAP API の全体的な配置を示しています。

図 2-3 LDAP サーバー・データを共有するアプリケーション

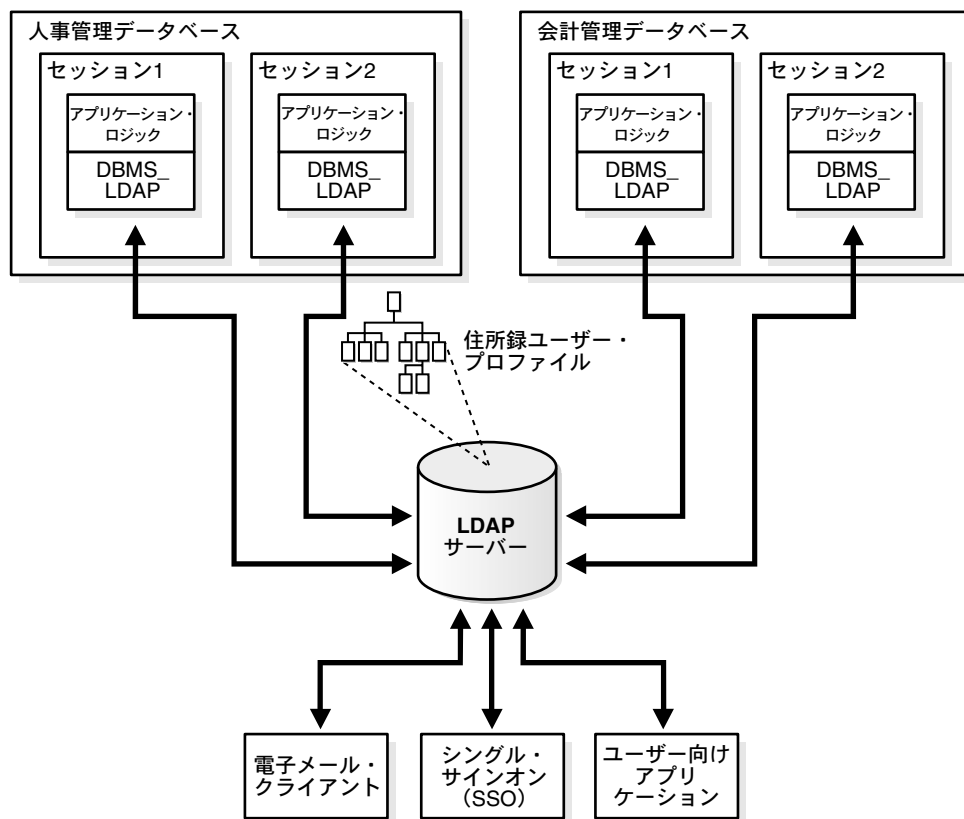


図 2-3 に示すように、API によって、異なる複数のアプリケーション（この例では人事管理と会計管理）で、LDAP サーバーを使用して、従業員の住所録情報とユーザー・プロフィールを共有できます。

このような情報を LDAP サーバーに格納すると、LDAP 対応の非データベース・アプリケーションでも同じ情報を取得できます。図 2-3 の電子メール・クライアント・アプリケーションは、指定された電子メール・アドレスの従業員を検索するのに、同じ従業員住所録データを使用しています。LDAP は一元化したユーザー情報のリポジトリを備えているため、同じ情報をシングル・サインオン・アプリケーションや他のエンタープライズ・ワイドのユーザー向けアプリケーションで使用できます。

要約すると、Oracle Internet Directory API によって、Oracle データベース・アプリケーションは次の処理を実行できます。

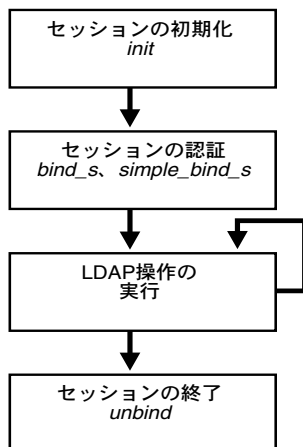
- 企業内の他のプログラムによって公開された LDAP サーバー情報を読み込みます。
- 新規の情報を LDAP サーバーに公開します。この情報は、後で同じアプリケーションまたは他のアプリケーションで使用できます。
- 事前定義された特定の条件に基づいて、LDAP サーバー内の既存の情報を変更または更新します。

一般的に、アプリケーションまたはトリガーでは、次の 4 つの簡単な手順に従って API のファンクションを使用します。

1. ライブラリを初期化し、LDAP セッション・ハンドルを取得します。
2. 必要に応じて、LDAP サーバーに対する認証を行います。
3. いくつかの LDAP 操作を実行し、その結果とエラー（ある場合）を取得します。
4. セッションをクローズします。

図 2-4 は、これらの手順を示しています。

図 2-4 DBMS_LDAP の一般的な使用手順



次の項では、各手順での API の重要な機能について説明します。

LDAP セッションの初期化

すべての LDAP 操作で、クライアントは LDAP サーバーとの LDAP セッションを確立する必要があります。LDAP 操作を実行するには、最初にデータベース・セッションを初期化してから LDAP セッションをオープンする必要があります。

C API を使用したセッションの初期化

`ldap_init()` によって、LDAP サーバーとのセッションが初期化されます。サーバーは、そのサーバーを必要とする操作が実行されるまで実際に接続されないため、初期化した後に様々なオプションを設定できます。

構文

```
LDAP *ldap_init
(
    const char    *hostname,
    int           portno
);
```

パラメータ

表 2-1 ldap_init() のパラメータ

パラメータ	説明
hostname	スペースで区切られたホスト名のリスト、または LDAP サーバーを実行している接続先のホストの IP アドレスを示すドット区切りの文字列が入ります。リスト内の各ホスト名には、ポート番号を含めることもできます。ポート番号とホスト自体はコロン (:) で区切ります。ホストへの接続はリストの順序に従って試みられ、最初にホストへの接続が成功した時点で終了します。 注意: リテラル IPv6[10] アドレスを hostname パラメータに格納するための適切な表現が必要ですが、現在はまだ決定および実装されていません。
portno	接続先の TCP ポート番号が入ります。デフォルトの LDAP ポート 389 は、定数 LDAP_PORT を指定することで取得できます。ホストにポート番号が含まれている場合、このパラメータは無視されます。

ldap_init() と ldap_open() の戻り値はセッション・ハンドルです。このハンドルは、そのセッションに関連する後続のコールに渡す必要がある不透明な構造体へのポインタです。これらのルーチンは、セッションが初期化できない場合に NULL を戻します。この場合、オペレーティング・システムのエラー・レポート・メカニズムによって、コールに失敗した理由をチェックできます。

LDAP v2 サーバーに接続する場合は、次に説明する LDAP バインド・コールの 1 つをセッションで完了してから、他の操作を実行する必要があることに注意してください。LDAP v3 では、他の操作を実行する前にバインド操作を完了する必要はありません。

コール元プログラムでは、次の項で説明するルーチンを呼び出して、セッションの様々な属性を設定できます。

DBMS_LDAP を使用したセッションの初期化

初期化は、DBMS_LDAP.init() ファンクションをコールして行います。init ファンクションの構文は次のとおりです。

```
FUNCTION init (hostname IN VARCHAR2, portnum IN PLS_INTEGER )
RETURN SESSION;
```

LDAP セッションを確立するには、init ファンクションに有効なホスト名とポート番号が必要です。このファンクションは、LDAP セッションにデータ構造を割り当て、コール元に DBMS_LDAP.SESSION タイプのハンドルを戻します。init コールで戻されたハンドルは、API を使用する後続のすべての LDAP 操作で使用する必要があります。DBMS_LDAP API では、LDAP セッション・ハンドルを使用して、オープン接続、未処理の要求および他の情報に関する状態をメンテナンスします。

同一のデータベース・セッションで、必要な数の LDAP セッションを取得できます。通常、同一のデータベース・セッション内で複数の LDAP セッションをオープンするのは、次のような場合です。

- 複数の LDAP サーバーから同時にデータを取得する必要がある場合
- 複数の LDAP 認証を使用してセッションをオープンする必要がある場合

注意： `DBMS_LDAP.init()` のコールで戻されたハンドルは、動的な構造体です。このハンドルは、複数のデータベース・セッションで使用することはできません。ハンドルの値を永続的なフォームに格納し、後で再利用すると、予測できない結果になる場合があります。

C API での LDAP セッション・ハンドル・オプション

`ldap_init()` で戻された LDAP セッション・ハンドルは、LDAP セッションを表す不透明なデータ型へのポインタです。RFC 1823 では、このデータ型はコール元に公開されていた構造体で、構造体のフィールドを設定すると、検索時のサイズや時間の制限などセッションの様々な側面を制御できました。

現在は、コール元でこの構造体に対する重要な変更を行わないように、次に説明する 1 組のアクセッサ・ファンクションによってセッションの様々な側面を制御できます。

`ldap_get_option()` は、セッション全体の様々なパラメータの現在の値にアクセスするために使用します。`ldap_set_option()` は、これらのパラメータの値を設定するために使用します。一部のオプションは読取り専用のため、設定できないことに注意してください。`ldap_set_option()` をコールして読取り専用のオプションを設定しようとするとエラーが発生します。

自動参照追跡が有効な場合（デフォルト）、参照の追跡時に確立された接続は、参照を戻す原因となった最初の要求を送信したセッションに関するオプションを継承することに注意してください。

ディレクトリ・サーバーへの認証の有効化

LDAP に対する操作を実行するユーザーまたはアプリケーションは、LDAP 操作の開始前に認証を受ける必要があります。

C API を使用したディレクトリ・サーバーへの認証の有効化

`ldap_sasl_bind()` ファンクションと `ldap_sasl_bind_s()` ファンクションを使用すると、LDAP に対する一般的な認証と拡張可能な認証を簡易認証セキュリティ・レイヤー [12] を使用して行うことができます。いずれのルーチンもバインドする識別名を、メソッドを示す OID のドット区切り文字列および資格証明を保持している `berval` 構造体として使用します。特別な定数値 `LDAP_SASL_SIMPLE (NULL)` を渡して簡易認証を要求できます。または、簡略化したルーチン `ldap_simple_bind()` または `ldap_simple_bind_s()` を使用できます。

DBMS_LDAP を使用したディレクトリ・サーバーへの認証の有効化

`simple_bind_s` ファンクションおよび `bind_s` ファンクションを使用すると、アプリケーションは、特定の資格証明を使用して、ディレクトリ・サーバーへの認証を受けることができます。`simple_bind_s` ファンクションの構文は次のとおりです。

```
FUNCTION simple_bind_s ( ld IN SESSION, dn IN VARCHAR2, passwd IN VARCHAR2)
RETURN PLS_INTEGER;
```

`simple_bind_s` ファンクションには、`init` ファンクションで取得した LDAP セッション・ハンドルが最初のパラメータとして必要です。また、エントリの LDAP 識別名も必要です。この識別名は、次の内容を示します。

- アプリケーションが認証を受けるときに使用する識別情報
- その識別情報に対するパスワード

`dn` パラメータおよび `passwd` パラメータが `NULL` の場合、LDAP サーバーは `anonymous` と呼ばれる特別な認証をアプリケーションに割り当てます。通常、`anonymous` 認証は、LDAP ディレクトリの最小限の権限に関連付けられています。

バインド操作が完了すると、ディレクトリ・サーバーには、別のバインド操作が実行されるまで、または `unbind_s` を使用して LDAP セッションが終了するまで新規の識別情報が保持されます。LDAP サーバーは、この識別情報を使用して、エンタープライズ管理で指定されたセキュリティ・モデルを規定します。特に、この識別情報は、ユーザーまたはアプリケーションがディレクトリ内で検索、更新または比較を行うための十分な権限を持っているかどうかを LDAP サーバーが判断するのに役立ちます。

バインド操作用のパスワードは、ネットワーク上をクリアテキストで送信されることに注意してください。ネットワークが保護されていない場合は、すべての LDAP 操作について、安全なデータ・トランスポートとともに、SSL 認証の使用を検討してください。SSL 通信を開始するファンクションは `open_ssl` で、その構文は次のとおりです。

```
FUNCTION open_ssl(ld IN SESSION, sslwrl IN VARCHAR2,  
    sslwalletpasswd IN VARCHAR2, sslauth IN PLS_INTEGER)  
RETURN PLS_INTEGER;
```

open_ssl ファンクションは、init をコールした直後にコールし、LDAP TCP/IP 接続が外部に漏れるのを防ぐ必要があります。認証は、Wallet に格納されている証明書の資格証明を使用して暗黙的に行われます。

関連項目： Oracle Wallet Manager の詳細は、『Oracle Internet Directory 管理者ガイド』の付録を参照してください。

次の PL/SQL コード例は、前述した初期化、認証およびクリーンアップの各ファンクションの一般的な使用方法を示しています。

```
DECLARE  
    retval PLS_INTEGER;  
    my_session DBMS_LDAP.session;  
  
BEGIN  
    retval := -1;  
    -- Initialize the LDAP session  
    my_session := DBMS_LDAP.init('yow.acme.com', 389);  
    -- Bind to the directory  
    retval := DBMS_LDAP.simple_bind_s(my_session, 'cn=orcladmin',  
    'welcome');
```

このコード例では、LDAP セッションは、TCP/IP ポート番号 389 で要求をリスニングするコンピュータ yow.acme.com 上にある LDAP サーバーに対して初期化されます。次に、cn=orcladmin の識別情報を使用して認証が実行されます。パスワードは welcome です。これによって LDAP セッションが認証され、通常の LDAP 操作を実行できます。

DBMS_LDAP を使用した検索

検索は、最も頻繁に使用される LDAP 操作です。LDAP 検索操作によって、アプリケーションは複雑な検索条件を使用してディレクトリからエントリを選択して取得できます。このリリースの DBMS_LDAP API に準備されているのは、同期検索機能のみです。これは、検索機能のコール元は、LDAP サーバーが結果セット全体を戻すまでブロックされることを意味します。

DBMS_LDAP API で検索を開始するためのファンクションは次の 2 つです。

- DBMS_LDAP.search_s()
- DBMS_LDAP.search_st()

この 2 つのファンクションの唯一の相違点は、一定の経過時間が過ぎた場合、search_st() はクライアント側のタイムアウトを使用して検索を停止することです。DBMS_LDAP.search_s() の構文は、次のとおりです。

```
FUNCTION search_s
(
    ld          IN  SESSION,
    base        IN  VARCHAR2,
    scope       IN  PLS_INTEGER,
    filter      IN  VARCHAR2,
    attrs       IN  STRING_COLLECTION,
    attronly    IN  PLS_INTEGER,
    res         OUT MESSAGE
)
RETURN PLS_INTEGER;
```

両方のファンクションとも、次の引数を取ります。

引数	説明
ld	有効なセッション・ハンドルです。
base	検索を開始する LDAP サーバー内のベース・エントリの識別名です。
scope	検索対象の DIT の幅と深さです。
filter	検索対象のエントリを選択するためのフィルタです。
attrs	検索で戻されるエントリの属性です。
attronly	1 に設定すると、属性のみが戻されます。
res	追加処理を行うための結果セットを戻す OUT パラメータです。

API では、`search_s` と `search_st` 以外にも、検索結果を取得するためのファンクションがいくつかサポートされています。これらのファンクションについては、次の項で説明します。

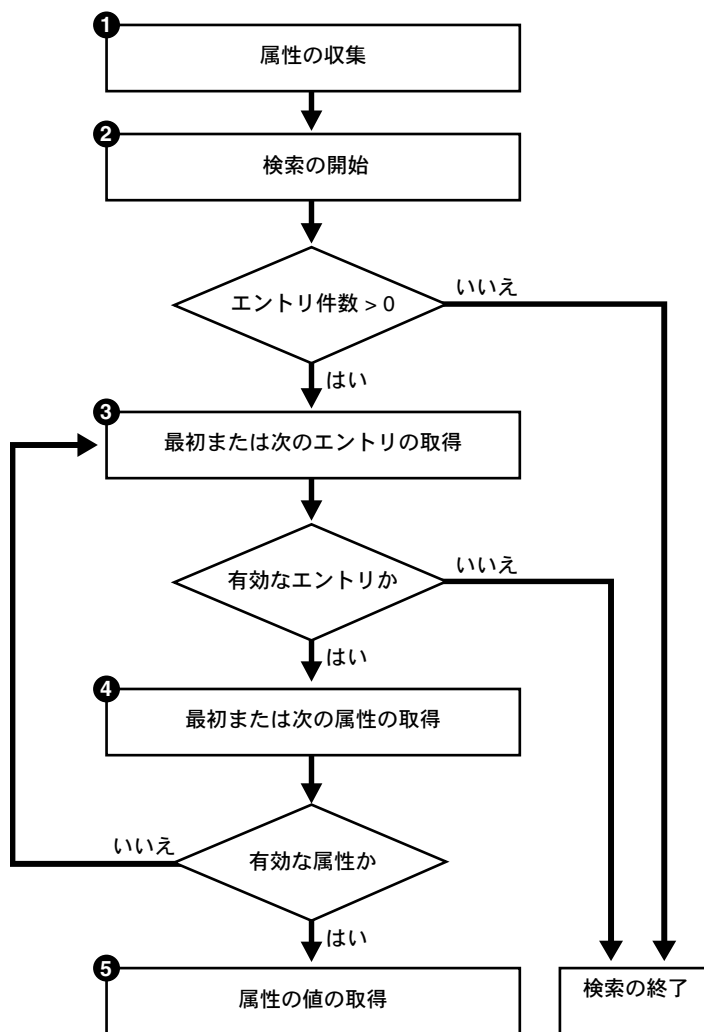
検索関連操作の流れ

一般的な検索操作を開始して結果を取得するために必要なプログラム作業は、次の手順で行います。

1. 検索によって戻される属性を決定し、その属性を `DBMS_LDAP.STRING_COLLECTION` データ型に設定します。
2. 必要なオプションとフィルタを設定して、検索操作を開始します (`DBMS_LDAP.search_s` または `DBMS_LDAP.search_st` を使用します)。
3. 結果セットからエントリを取得します (`DBMS_LDAP.first_entry` または `DBMS_LDAP.next_entry` を使用します)。
4. 手順 3 で取得したエントリの属性を取得します (`DBMS_LDAP.first_attribute` または `DBMS_LDAP.next_attribute` を使用します)。
5. 手順 4 で取得した属性のすべての値を取得し、その値をローカル変数にコピーします (`DBMS_LDAP.get_values` または `DBMS_LDAP.get_values_len` を使用します)。
6. エントリのすべての属性が処理されるまで、手順 4 を繰り返します。
7. すべてのエントリが処理されるまで、手順 3 を繰り返します。

[図 2-5](#) は、前述の手順を詳細に示しています。

図 2-5 検索関連操作の流れ



検索有効範囲

検索の有効範囲は、検索のベースになるエントリの数を決定します。このベースによって、ディレクトリ・サーバーは、エントリが指定のフィルタ条件に一致しているかどうかを調べます。`search_s()` ファンクションまたは `search_st()` ファンクションのいずれかを起動するときは、次の3つのオプションのいずれかを指定できます。

- SCOPE_BASE** ディレクトリ・サーバーは、検索のベースに対応しているエントリのみを検索し、指定したフィルタの条件に一致しているかどうかを調べます。
- SCOPE_ONELEVEL** ディレクトリ・サーバーは、ベース・オブジェクトの直接の子エントリのみを検索し、指定したフィルタの条件に一致しているかどうかを調べます。
- SCOPE_SUBTREE** ディレクトリ・サーバーは、LDAP サブツリー全体を（基本となっているベース・オブジェクトも含めて）検索します。

図 2-6 は、この3つの有効範囲オプションの違いを示しています。

図 2-6 3つの有効範囲オプション

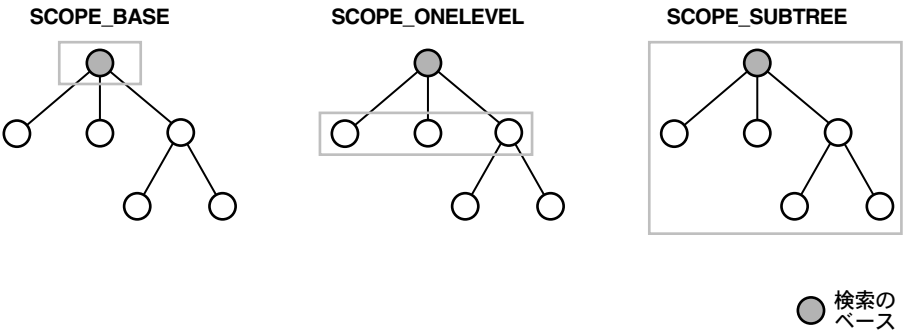


図 2-6 では、検索のベースはグレーの丸で表されています。検索対象のエントリは、薄い色の四角形で囲まれています。

フィルタ

search_s() および search_st() に必要な検索フィルタは、**Internet Engineering Task Force (IETF)** の RFC 1960 で定義されている文字列フォーマットに準拠しています。この項では、フィルタで使用できる様々なオプションについて簡単に説明します。

属性、演算子、値の順で書式を指定する基本的な検索フィルタには、次の 6 種類があります。次の表は、基本的な検索フィルタの要約です。

表 2-2 検索フィルタ

フィルタ・タイプ	書式	例	一致
等価	(attr=value)	(sn=Keaton)	Keaton と完全に等しい姓
近似	(attr~=value)	(sn~=Ketan)	Ketan と近似の姓
部分文字列	(attr=[leading]*[any]*[trailing])	(sn=*keaton*)	文字列 keaton を含む姓
		(sn=keaton*)	keaton で始まる姓
		(sn=*keaton)	keaton で終わる姓
		(sn=ke*at*on)	ke で始まり、at が含まれ、on で終わる姓
以上	(attr>=value)	(sn>=Keaton)	辞書編集順で Keaton 以降の姓
以下	(attr<=value)	(sn<=Keaton)	辞書編集順で Keaton 以前の姓
存在	(attr=*)	(sn=*)	sn 属性を持つすべてのエントリ

表 2-2 の基本フィルタは、ブール演算子や接頭辞表記法を使用して組み合わせることで、さらに複雑なフィルタを構成できます。& 文字は AND、| 文字は OR、! 文字は NOT を表します。

表 2-3 は、基本的なブール演算子の要約です。

表 2-3 ブール演算子

フィルタ・ タイプ	書式	例	一致
AND	(&(<filter1>)(<filter2>)...)	(&(sn=keaton)(objectclass=inetOrgPerson))	姓が Keaton、かつオブジェクト・クラスが InetOrgPerson のエントリ
OR	((<filter1>)(<filter2>)...)	((sn~=ketan)(cn=*keaton))	姓が ketan に近似しているか、または一般名が keaton で終わるエントリ
NOT	(!(<filter>))	(!(mail=*))	メール属性を持たないエントリ

前述の複合フィルタをさらに組み合せて、ネストした複合フィルタを作成できます。

DBMS_LDAP を使用したセッション終了の有効化

LDAP セッション・ハンドルを取得し、目的の LDAP 関連作業をすべて完了した後は、LDAP セッションを破棄する必要があります。これは、DBMS_LDAP.unbind_s() をコールして行います。unbind_s ファンクションの構文は次のとおりです。

```
FUNCTION unbind_s (ld IN SESSION ) RETURN PLS_INTEGER;
```

unbind_s のコールが正常終了すると、LDAP サーバーへの TCP/IP 接続がクローズし、LDAP セッションで使用されたすべてのシステム・リソースが割当て解除されて、整数 DBMS_LDAP.SUCCESS がコール元に戻されます。特定のセッションで unbind_s ファンクションを起動した後は、init をコールしてセッションを再初期化しないかぎり、そのセッションでの LDAP 操作はできません。

Oracle Internet Directory の C API

この章では、Oracle Internet Directory の C Application Program Interface (C API) について説明し、その使用例を紹介します。この章では、次の項目について説明します。

- [Oracle Internet Directory C API の概要](#)
- [C API リファレンス](#)
- [C API の使用例](#)
- [C API を使用したアプリケーションの作成](#)
- [依存性と制限事項](#)

Oracle Internet Directory C API の概要

Oracle Internet Directory SDK C API は、次のものをベースにしています。

- LDAP バージョン 3 C API
- Secure Sockets Layer (SSL) をサポートする Oracle の拡張機能

Oracle Internet Directory API リリース 9.0.2 は、次のモードで使用できます。

- SSL モード SSL を使用してすべての通信を保護する
- 非 SSL モードクライアントからサーバーへの通信を保護しない

API は、TCP/IP を使用して LDAP サーバーに接続します。接続するとき、デフォルトでは暗号化されていないチャネルを使用します。SSL モードを使用するには、Oracle SSL コール・インタフェースを使用する必要があります。API を使用する際に、SSL コールの有無によってどちらのモードを使用するかを決定します。SSL モードと非 SSL モードは簡単に切り替えることができます。

関連項目： この 2 つのモードの使用方法については、3-55 ページの「[C API の使用例](#)」を参照してください。

この項では、次の項目について説明します。

- [Oracle Internet Directory SDK C API SSL 拡張機能](#)
- [LDAP C API の概要](#)

Oracle Internet Directory SDK C API SSL 拡張機能

LDAP API への Oracle SSL 拡張機能は、標準的な SSL プロトコルに基づいています。SSL 拡張機能は回線を通るデータの暗号化と復号化および認証を行います。

認証には次の 3 つのモードがあります。

- 認証なしクライアントとサーバーのどちらも認証せず、SSL による暗号化のみ使用
- サーバー認証クライアントによるサーバーの認証のみ
- クライアントとサーバーの認証クライアントとサーバーが相互に認証

認証のタイプは、SSL インタフェース・コールのパラメータで指定します。

SSL インタフェース・コール

次のコールを行うだけで、SSL は使用可能になります。

```
int ldap_init_SSL(Sockbuf *sb, text *sslwallet, text *sslwalletpasswd, int
sslauthmode)
```

`ldap_init_SSL` コールは、標準的な SSL プロトコルを使用して、クライアントとサーバーの間で必要なハンドシェイクを実行します。コールが成功すると、これ以降のすべての通信は保護された接続で実行されます。

引数	説明
<code>sb</code>	LDAP ハンドルの一部として、 <code>ldap_open</code> コールによって戻されたソケット・バッファ・ハンドル。
<code>sslwallet</code>	ユーザー Wallet の位置。
<code>sslwalletpasswd</code>	Wallet を使用するために必要なパスワード。
<code>sslauthmode</code>	<p>ユーザーが使用できる SSL 認証モード。使用できる値は次のとおりです。</p> <ul style="list-style-type: none"> ■ <code>GSLC_SSL_NO_AUTH</code> – 認証の必要なし ■ <code>GSLC_SSL_ONEWAY_AUTH</code> – サーバー認証のみ必要 ■ <code>GSLC_SSL_TWOWAY_AUTH</code> – クライアントとサーバーの両方の認証が必要 <p>戻り値が 0（ゼロ）のときは、処理は成功です。戻り値が 0（ゼロ）以外のときは、エラーです。エラー・コードは、<code>ldap_err2string</code> ファクションを使用してエラー・メッセージに変換できます。</p>

関連項目： 3-55 ページの「[C API の使用例](#)」を参照してください。

Wallet サポート

SSL の機能を使用するには、使用している認証モードに応じて、サーバーとクライアントそれぞれに Wallet が必要になります。この API のリリース 9.0.2 では、Oracle Wallet のみをサポートしています。Oracle Wallet Manager を使用して Wallet を作成できます。

C API リファレンス

この項では、次の項目について説明します。

- [LDAP C API の概要](#)
- [ファンクション](#)
- [LDAP セッションの初期化](#)
- [LDAP セッション・ハンドル・オプション](#)
- [コントロールの使用](#)
- [ディレクトリに対する認証](#)
- [セッションのクローズ](#)
- [LDAP 操作の実行](#)
- [操作の中止](#)
- [結果の取得と LDAP メッセージの確認](#)
- [エラーの処理と結果の解析](#)
- [結果リストの参照](#)
- [検索結果の解析](#)
- [SSL モードでの C API の使用方法](#)
- [非 SSL モードでの C API の使用方法](#)

LDAP C API の概要

表 3-1 DBMS_LDAP API のサブプログラム

ファンクションまたは プロシージャ	説明
ber_free()	BerElement 構造体のために割り当てられたメモリーの解放
ldap_abandon_ext	非同期操作の取消し
ldap_abandon	

表 3-1 DBMS_LDAP API のサブプログラム (続き)

ファンクションまたは プロシージャ	説明
ldap_add_ext	ディレクトリに新規エントリを追加
ldap_add_ext_s	
ldap_add	
ldap_add_s	
ldap_compare_ext	ディレクトリ内のエントリの比較
ldap_compare_ext_s	
ldap_compare	
ldap_compare_s	
ldap_count_entries	一連の検索結果のエントリ件数をカウント
ldap_count_values	属性の文字列値をカウント
ldap_count_values_len	属性のバイナリ値をカウント
ldap_delete_ext	ディレクトリからのエントリの削除
ldap_delete_ext_s	
ldap_delete	
ldap_delete_s	
ldap_dn2ufn	ユーザーにわかりやすい名前に変換
ldap_err2string	特定のエラー・コードのエラー・メッセージを取得
ldap_explode_dn	識別名を構成要素に分割
ldap_explode_rdn	
ldap_first_attribute	エントリ内の最初の属性の名前を取得
ldap_first_entry	一連の検索結果から最初のエントリを取得
ldap_get_dn	エントリの識別名の取得
ldap_get_option	セッション全体の様々なパラメータの現在の値にアクセス
ldap_get_values	属性の文字列値を取得
ldap_get_values_len	属性のバイナリ値を取得

表 3-1 DBMS_LDAP API のサブプログラム (続き)

ファンクションまたは プロシージャ	説明
ldap_init	LDAP サーバーへの接続のオープン
ldap_open	
ldap_memfree()	LDAP API ファンクション・コールによって割り当てられたメモリーの解放
ldap_modify_ext	ディレクトリ内のエントリの変更
ldap_modify_ext_s	
ldap_modify	
ldap_modify_s	
ldap_msgfree	検索結果あるいは他の LDAP 操作結果のために割り当てられたメモリーの解放
ldap_next_attribute	エントリ内の次の属性の名前を取得
ldap_next_entry	一連の検索結果から次のエントリを取得
ldap_perror	エラー・メッセージの中から指定したメッセージを出力
使用不可	
ldap_rename	ディレクトリ内のエントリの相対識別名を変更
ldap_rename_s	
ldap_result2error	結果メッセージからエラー・コードを取得
使用不可	
ldap_result	非同期操作の結果のチェック
ldap_msgfree	
ldap_msgtype	
ldap_msgid	
ldap_sasl_bind	LDAP サーバーへの一般認証
ldap_sasl_bind_s	

表 3-1 DBMS_LDAP API のサブプログラム（続き）

ファンクションまたは プロシージャ	説明
ldap_search_ext	ディレクトリの検索
ldap_search_ext_s	
ldap_search	
ldap_search_s	
ldap_search_st	タイムアウト値でのディレクトリの検索
ldap_set_option	パラメータの値を設定
ldap_simple_bind	LDAP サーバーへの簡易認証
ldap_simple_bind_s	
ldap_unbind_ext	LDAP セッションの終了
ldap_unbind	
ldap_unbind_s	
ldap_value_free	属性の文字列値のために割り当てられたメモリの解放
ldap_value_free_len	属性のバイナリ値のために割り当てられたメモリの解放

この項では、RFC 1823 に規定されている LDAP C API で使用可能なすべてのコールを示します。

関連項目： 次の URL を参照してください。
<http://www.ietf.org/rfc/rfc1823.txt>（各コールの詳細な説明）

ファンクション

この項では、次の項目について説明します。

- [LDAP セッションの初期化](#)
- [LDAP セッション・ハンドル・オプション](#)
- [ディレクトリに対する認証](#)
- [コントロールの使用](#)
- [セッションのクローズ](#)
- [LDAP 操作の実行](#)
- [操作の中止](#)

- [結果の取得と LDAP メッセージの確認](#)
- [エラーの処理と結果の解析](#)
- [結果リストの参照](#)
- [検索結果の解析](#)

LDAP セッションの初期化

ldap_init

ldap_open

ldap_init() によって、LDAP サーバーとのセッションが初期化されます。サーバーは、そのサーバーを必要とする操作が実行されるまで実際に接続されないため、初期化した後に様々なオプションを設定できます。

構文

```
LDAP *ldap_init
(
    const char      *hostname,
    int             portno
)
;
```

パラメータ

表 3-2 LDAP セッションを初期化するためのパラメータ

パラメータ	説明
hostname	スペースで区切られたホスト名のリスト、または LDAP サーバーを実行している接続先のホストの IP アドレスを示すドット区切りの文字列が入ります。リスト内の各ホスト名には、ポート番号を含めることもできます。ポート番号とホスト自体はコロン (:) で区切ります。ホストへの接続はリストの順序に従って試みられ、最初にホストへの接続が成功した時点で終了します。 注意: リテラル IPv6[10] アドレスを hostname パラメータに格納するための適切な表現が必要ですが、現在はまだ決定および実装されていません。
portno	接続先の TCP ポート番号が入ります。デフォルトの LDAP ポート 389 は、定数 LDAP_PORT を指定することで取得できます。ホストにポート番号が含まれている場合、このパラメータは無視されます。

使用方法

`ldap_init()` および `ldap_open()` の戻り値はセッション・ハンドルです。このハンドルは、そのセッションに関連する後続のコールに渡す必要がある不透明な構造体へのポインタです。これらのルーチンは、セッションが初期化できない場合に `NULL` を戻します。この場合、オペレーティング・システムのエラー・レポート・メカニズムによって、コールに失敗した理由をチェックできます。

LDAP v2 サーバーに接続する場合は、次に説明する LDAP バインド・コールの 1 つをセッションで完了してから、他の操作を実行する必要があることに注意してください。LDAP v3 では、他の操作を実行する前にバインド操作を完了する必要はありません。

コール元プログラムでは、次の項で説明するルーチン呼び出して、セッションの様々な属性を設定できます。

LDAP セッション・ハンドル・オプション

`ldap_init()` で戻された LDAP セッション・ハンドルは、LDAP セッションを表す不透明なデータ型へのポインタです。RFC 1823 では、このデータ型はコール元に公開されていた構造体で、構造体のフィールドを設定すると、検索時のサイズや時間の制限などセッションの様々な側面を制御できました。

現在は、コール元でこの構造体に対する重要な変更を行わないように、次に説明する 1 組のアクセッサ・ファンクションによってセッションの様々な側面を制御できます。

`ldap_get_option`

`ldap_set_option`

`ldap_get_option()` は、セッション全体の様々なパラメータの現在の値にアクセスするために使用します。`ldap_set_option()` は、これらのパラメータの値を設定するために使用します。一部のオプションは読取り専用のため、設定できないことに注意してください。`ldap_set_option()` をコールして読取り専用のオプションを設定しようとするエラーが発生します。

自動参照追跡が有効な場合（デフォルト）、参照の追跡時に確立された接続は、参照を戻す原因となった最初の要求を送信したセッションに関するオプションを継承することに注意してください。

構文

```
int ldap_get_option
(
    LDAP          *ld,
    int           option,
    void          *outvalue
)
;
```

```
int ldap_set_option
(
    LDAP          *ld,
    int           option,
    const void     *invalue
)
;

#define LDAP_OPT_ON      ((void *)1)
#define LDAP_OPT_OFF     ((void *)0)
```

パラメータ

表 3-3 LDAP セッション・ハンドル・オプションのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。このパラメータが NULL の場合は、一連のグローバル・デフォルト値にアクセスします。ldap_init() または ldap_open() を使用して作成された新規の LDAP セッション・ハンドルは、これらのグローバル・デフォルト値の特性を継承します。
option	アクセスまたは設定するオプションの名前です。このパラメータは、表 3-4 で説明されている定数のいずれかである必要があります。定数の後に、その定数の実際の 16 進値をカッコで囲んで記述します。
outvalue	オプションの値を配置する位置のアドレスです。このパラメータの実際の型は、option パラメータの設定値によって異なります。outvalue パラメータが char ** 型および LDAPControl ** 型の場合は、LDAP セッション ld に対応付けられているデータのコピーが戻されます。コール元では、戻されたデータの型に従って ldap_memfree() または ldap_controls_free() をコールし、メモリーを解放する必要があります。
invalue	option パラメータに指定する値へのポインタです。このパラメータの実際の型は、option パラメータの設定値によって異なります。invalue パラメータに関連付けられたデータは API 実装によってコピーされるため、API のコール元はデータを解放するか、ldap_set_option() のコールが正常終了した後にそのデータのコピーを変更できます。invalue パラメータに渡された値が無効な場合、または実装で受け入れることができない場合、ldap_set_option() は -1 を戻してエラーの発生を示す必要があります。

表 3-4 定数

定数	invalue パラメータ の型	outvalue パラメータ の型	説明
LDAP_OPT_API_INFO (0x00)	該当なし (オプションは READ-ONLY)	LDAPAPIInfo *	実行時に LDAP API 実装に関する基本的な情報を取得します。アプリケーションでは、コンパイル時および実行時の両方で使用する特定の API 実装に関する情報を判断することが必要です。このオプションは READ-ONLY で、設定はできません。
LDAP_OPT_DEREF (0x02)	int *	int *	検索時の別名の処理方法を判断します。この定数は、LDAP_DEREF_NEVER (0x00)、LDAP_DEREF_SEARCHING (0x01)、LDAP_DEREF_FINDING (0x02) または LDAP_DEREF_ALWAYS (0x03) のいずれかであることが必要です。LDAP_DEREF_SEARCHING 値の場合、別名は検索時に参照解除されますが、検索のベース・オブジェクトの検索時は参照解除されません。LDAP_DEREF_FINDING 値の場合、別名はベース・オブジェクトの検索時に参照解除されますが、検索時は参照解除されません。このオプションのデフォルト値は LDAP_DEREF_NEVER です。
LDAP_OPT_SIZELIMIT (0x03)	int *	int *	検索で戻されるエントリの最大数です。LDAP_NO_LIMIT (0) 値は上限がないことを意味します。このオプションのデフォルト値は LDAP_NO_LIMIT です。
LDAP_OPT_TIMELIMIT (0x04)	int *	int *	検索を実行する最大秒数です。LDAP_NO_LIMIT (0) 値は上限がないことを意味します。この値は検索要求時のみサーバーに渡され、C LDAP API 実装がローカルで検索結果を待機する時間には影響を与えません。ldap_search_ext_s() または ldap_result() (このマニュアルで後述) に渡された timeout パラメータを使用すると、ローカル側とサーバー側の制限時間を指定できます。このオプションのデフォルト値は LDAP_NO_LIMIT です。

表 3-4 定数（続き）

定数	invalue パラメータ の型	outvalue パラメータ の型	説明
LDAP_OPT_REFERRALS (0x08)	void * (LDAP_OPT_ON または LDAP_OPT_OFF)	int *	LDAP ライブラリが、LDAP サーバーで戻された参照に自動的に従うかどうかを判断します。定数の LDAP_OPT_ON または LDAP_OPT_OFF のいずれかに設定できます。このオプションは、ldap_set_option() に渡される NULL 以外のポインタ値によって有効になります。ldap_get_option() を使用して現在の設定値を読み込むとき、0（ゼロ）値は OFF、0（ゼロ）以外の値は ON を意味します。このオプションのデフォルト値は ON です。
LDAP_OPT_RESTART (0x09)	void * (LDAP_OPT_ON または LDAP_OPT_OFF)	int *	LDAP I/O 操作が未完了のまま異常終了したとき、自動的に再起動するかどうかを判断します。定数の LDAP_OPT_ON または LDAP_OPT_OFF のいずれかに設定できます。このオプションは、ldap_set_option() に渡される NULL 以外のポインタ値によって有効になります。ldap_get_option() を使用して現在の設定値を読み込むとき、0（ゼロ）値は OFF、0（ゼロ）以外の値は ON を意味します。このオプションは、タイマーの停止や他の割込みによって、LDAP I/O 操作が未完了のまま中断する可能性がある場合に便利です。このオプションのデフォルト値は OFF です。
LDAP_OPT_PROTOCOL_VERSION (0x11)	int *	int *	このオプションは、プライマリ LDAP サーバーとの通信時に使用する LDAP プロトコルのバージョンを示します。定数の LDAP_VERSION2 (2) または LDAP_VERSION3 (3) のいずれかを設定できます。バージョンが設定されていない場合のデフォルトは、LDAP_VERSION2 (2) です。
LDAP_OPT_SERVER_CONTROLS (0x12)	LDAPControl **	LDAPControl ***	各要求とともに送信される LDAP サーバー・コントロールのデフォルト・リストです。後の「コントロールの使用」の項を参照してください。
LDAP_OPT_CLIENT_CONTROLS (0x13)	LDAPControl **	LDAPControl ***	LDAP セッションに影響を与えるクライアント・コントロールのデフォルト・リストです。後の「コントロールの使用」の項を参照してください。

表 3-4 定数（続き）

定数	invalue パラメータ の型	outvalue パラメータ の型	説明
LDAP_OPT_API_FEATURE_INFO (0x15)	該当なし (オプションは READ-ONLY)	LDAPAPIFeatureInfo *	実行時に LDAP API 拡張機能に関するバージョン情報を取得します。アプリケーションでは、コンパイル時および実行時の両方で使用する特定の API 実装に関する情報を判断できる必要があります。このオプションは READ-ONLY で、設定はできません。
LDAP_OPT_HOST_NAME (0x30)	char *	char **	プライマリ LDAP サーバーのホスト名（またはホストのリスト）です。有効な構文については、 <code>ldap_init()</code> に対する <code>hostname</code> パラメータの定義を参照してください。
LDAP_OPT_ERROR_NUMBER (0x31)	int *	int *	このセッションで発生した最新の LDAP エラーのコードです。
LDAP_OPT_ERROR_STRING (0x32)	char *	char **	このセッションで発生した最新の LDAP エラーで戻されたメッセージです。
LDAP_OPT_MATCHED_DN (0x33)	char *	char **	このセッションで発生した最新の LDAP エラーで戻された一致する識別名です。

使用方法

`ldap_get_option()` および `ldap_set_option()` は、正常終了した場合は 0（ゼロ）を、エラーが発生した場合は -1 を返します。いずれかのファンクションで -1 が戻された場合は、オプション値 `LDAP_OPT_ERROR_NUMBER` を設定して `ldap_get_option()` をコールすると、特定のエラー・コードを取得できます。オプション値 `LDAP_OPT_ERROR_NUMBER` を設定した `ldap_get_option()` コールに失敗すると、特定のエラー・コードを取得する方法は他にないことに注意してください。

`ldap_get_option()` コールが正常終了した場合、API 実装では、今後の LDAP API コールの動作に影響を与えるような、LDAP セッション・ハンドルの状態または基礎となる実装の状態の変更は行わないでください。`ldap_get_option()` コールに失敗した場合、許容されるセッション・ハンドルの変更は LDAP エラー・コードの設定のみです（`LDAP_OPT_ERROR_NUMBER` オプションによって戻されます）。

`ldap_set_option()` コールに失敗した場合は、今後の LDAP API コールの動作に影響を与えるような、LDAP セッション・ハンドルの状態または基礎となる実装の状態の変更は行わないでください。

この仕様を拡張して新規オプションを指定している規格準拠ドキュメントでは、0x1000 ～ 0x3FFF の間のオプション・マクロの値を使用する必要があります。プライベートな拡張や経験に基づく拡張では、0x4000 ～ 0x7FFF の間のオプション・マクロの値を使用する必要があります。このドキュメントで定義されていない 0x1000 未満および 0x7FFF を超える値は、すべて予約されており使用することはできません。拡張機能の実装を支援するには、次のマクロを C LDAP API 実装で定義する必要があります。

```
#define LDAP_OPT_PRIVATE_EXTENSION_BASE 0x4000 /* to 0x7FFF inclusive */
```

コントロールの使用

コントロールを使用すると、LDAP v3 操作を拡張できます。コントロールは、サーバーに送信したり、LDAP メッセージとともにクライアントに戻すことができます。このようなコントロールは、サーバー・コントロールと呼ばれます。

LDAP API は、クライアント・コントロールを使用してクライアント側の拡張メカニズムもサポートします。このコントロールは、LDAP API の動作にのみ影響し、サーバーに送信されることはありません。両方のタイプのコントロールを表現するため、次の共通のデータ構造が使用されます。

```
typedef struct ldapcontrol
{
    char          *ldctl_oid;
    struct berval  ldctl_value;
    char          ldctl_iscritical;
} LDAPControl;
```

ldapcontrol 構造体のフィールドについて次に説明します。

表 3-5 ldapcontrol 構造体のフィールド

フィールド	説明
ldctl_oid	文字列で表現されたコントロール・タイプです。
ldctl_value	コントロールに対応付けられたデータ（ある場合）です。長さ 0（ゼロ）の値を指定するには、ldctl_value.bv_len を 0（ゼロ）に設定し、ldctl_value.bv_val を長さ 0（ゼロ）の文字列に設定します。コントロールに対応付けられたデータがないことを示すには、ldctl_value.bv_val を NULL に設定します。
ldctl_iscritical	コントロールが重要かどうかを示します。このフィールドが 0（ゼロ）以外の場合は、サーバーまたはクライアント（あるいはその両方）でコントロールが認識されると、その操作のみが実行されます。LDAP のバインド解除操作および中止操作ではサーバーからの応答はないため、これら 2 つの操作でサーバー・コントロールを使用するときは、クライアントでコントロールを重要（critical）とマークしないでください。

LDAP API の一部のコールでは、`ldapcontrol` 構造体、または `ldapcontrol` 構造体の NULL 終了配列を割り当てます。次のルーチンを使用すると、単一コントロールまたはコントロールの配列を解放できます。

```
void ldap_control_free( LDAPControl *ctrl );
void ldap_controls_free( LDAPControl **ctrls );
```

`ctrl` パラメータまたは `ctrls` パラメータが NULL の場合は、このファンクションをコールしても何も実行されません。

セッション全体に影響を与える一連のコントロールは、`ldap_set_option()` ファンクション（前の説明を参照）を使用して設定できます。コントロール・リストは、`ldap_search_ext()` などの一部の LDAP API コールに直接渡すこともできます。その場合、`ldap_set_option()` を使用してセッションに設定したコントロールは無視されます。コントロール・リストは、`ldapcontrol` 構造体へのポインタの NULL 終了配列として表されます。

サーバー・コントロールは、LDAP v3 プロトコル拡張ドキュメントで定義されています。たとえば、コントロールは、サーバー側での検索結果のソートをサポートするために計画されています。

このドキュメントでは、1つのクライアント・コントロールが定義されています（後の項で説明）。他のクライアント・コントロールは、このドキュメントの今後のバージョン、またはこの API を拡張するドキュメントで定義される予定です。

参照プロセスを制御するクライアント・コントロール 3-9 ページの「[LDAP セッション・ハンドル・オプション](#)」で説明したように、アプリケーションでは、`LDAP_OPT_REFERRALS` オプションを設定した `ldap_set_option()` ファンクションを使用して、セッション全体で自動参照追跡を有効にしたり無効にすることができます。これは、要求ごとに自動参照追跡を制御する場合にも役立ちます。この機能を提供するため、1.2.840.113556.1.4.616 の OID を持つクライアント・コントロールがあります。

```
/* OID for referrals client control */
#define LDAP_CONTROL_REFERRALS          "1.2.840.113556.1.4.616"

/* Flags for referrals client control value */
#define LDAP_CHASE_SUBORDINATE_REFERRALS 0x00000020U
#define LDAP_CHASE_EXTERNAL_REFERRALS    0x00000040U
```

参照先クライアント・コントロールを作成するには、LDAPControl 構造体の `ldctl_oid` フィールドを `LDAP_CONTROL_REFERRALS` ("1.2.840.113556.1.4.616") に設定し、`ldctl_value` フィールドを一連のフラグが格納された 4 オクテット値に設定する必要があります。`ldctl_value.bv_len` フィールドは常に 4 に設定する必要があります。`ldctl_value.bv_val` フィールドは、4 オクテットの整数フラグ値を指し示す必要があります。このフラグ値を 0 (ゼロ) に設定すると、自動参照追跡と LDAP v3 リファレンスの両方を無効にできます。これ以外に、このフラグ値を、`LDAP_CHASE_SUBORDINATE_REFERRALS` (0x00000020U) に設定して、LDAP v3 の検索継続リファレンスのみが API 実装によって自動的に追跡されることを示すか、値 `LDAP_CHASE_EXTERNAL_REFERRALS` (0x00000040U) に設定して、LDAP v3 参照のみが自動的に追跡されることを示すか、2 つのフラグ値の論理和 (0x00000060U) に設定して、参照先と参照元の両方が自動的に追跡されることを示すことができます。

ディレクトリに対する認証

次のファンクションを使用して、LDAP ディレクトリ・サーバーに対する LDAP クライアントの認証を行います。

`ldap_sasl_bind`

`ldap_sasl_bind_s`

`ldap_simple_bind`

`ldap_simple_bind_s`

`ldap_sasl_bind()` ファンクションと `ldap_sasl_bind_s()` ファンクションを使用すると、LDAP に対する一般的な認証と拡張可能な認証を簡易認証セキュリティ・レイヤーを使用して行うことができます。いずれのルーチンもバインドする識別名を、メソッドを示すオブジェクトのドット表記の文字列および資格証明を保持している `berval` 構造体として使用します。特別な定数値 `LDAP_SASL_SIMPLE` (NULL) を渡して簡易認証を要求できます。または、簡略化したルーチン `ldap_simple_bind()` または `ldap_simple_bind_s()` を使用できます。

構文

```
int ldap_sasl_bind
(
    LDAP                *ld,
    const char          *dn,
    const char          *mechanism,
    const struct berval *cred,
    LDAPControl         **serverctrls,
    LDAPControl         **clientctrls,
    int                 *msgidp
```

```

);

int ldap_sasl_bind_s(
    LDAP *ld,
    const char *dn,
    const char *mechanism,
    const struct berval *cred,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    struct berval **servercredp
);

int ldap_simple_bind(
    LDAP *ld,
    const char *dn,
    const char *passwd
);

int ldap_simple_bind_s(
    LDAP *ld,
    const char *dn,
    const char *passwd
);

```

次のルーチンは使用できません。詳細は、RFC 1823 を参照してください。

```

int ldap_bind( LDAP *ld, const char *dn, const char *cred, int method );
int ldap_bind_s( LDAP *ld, const char *dn, const char *cred, int method );
int ldap_kerberos_bind( LDAP *ld, const char *dn );
int ldap_kerberos_bind_s( LDAP *ld, const char *dn );

```

パラメータ

表 3-6 ディレクトリに対する認証のパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
dn	バインドするエントリの名前です。
mechanism	LDAP_SASL_SIMPLE (NULL) を指定して簡易認証を取得するか、SASL メソッドを識別するテキスト文字列を指定します。

表 3-6 ディレクトリに対する認証のパラメータ（続き）

パラメータ	説明
cred	認証に使用する資格証明です。このパラメータを使用すると、任意の資格証明を渡すことができます。資格証明の書式と内容は、mechanism パラメータの設定内容によって異なります。
passwd	ldap_simple_bind() ファンクションの場合に、エントリの userPassword 属性と比較するパスワードです。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。
msgidp	ldap_sasl_bind() コールが正常終了した場合は、この結果パラメータが要求のメッセージ ID に設定されます。
servercredp	相互認証が指定されている場合は、この結果パラメータにサーバーから戻された資格証明が入力されます。割り当てられた berval 構造体が戻されるため、ber_bvfree() をコールして解放する必要があります。このフィールドを無視する場合は、NULL を渡す必要があります。

使用方法

使用できないルーチンに関するその他のパラメータは説明していません。詳細は、RFC 1823 を参照してください。

ldap_sasl_bind() ファンクションは、非同期のバインド操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_sasl_bind() によって要求のメッセージ ID が *msgidp に格納されます。後で ldap_result()（詳細は後述）をコールすると、バインドの結果を取得できます。

ldap_simple_bind() ファンクションは、単純な非同期のバインド操作を開始し、開始した操作のメッセージ ID を戻します。後で ldap_result()（詳細は後述）をコールすると、バインドの結果を取得できます。エラーが発生した場合、ldap_simple_bind() は -1 を返し、LDAP 構造体に適切なセッション・エラー・パラメータを設定します。

同期型の ldap_sasl_bind_s() ファンクションおよび ldap_simple_bind_s() ファンクションはいずれも、操作が正常終了した場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。

LDAP v2 サーバーに接続している場合は、バインド・コールが正常に完了するまで、接続に対する他の操作ができないことに注意してください。

後でバインド・コールを使用すると、同じ接続に対して再認証を行うことができます。また、ldap_sasl_bind() または ldap_sasl_bind_s() を連続してコールすると、複数ステップの SASL 処理を実行できます。

セッションのクローズ

ldap_unbind_ext

ldap_unbind

ldap_unbind_s

次のファンクションを使用して、ディレクトリからバインドを解除し、オープンしている接続をクローズして、セッション・ハンドルを解放します。

構文

```
int ldap_unbind_ext( LDAP *ld, LDAPControl **serverctrls,
LDAPControl **clientctrls );
int ldap_unbind( LDAP *ld );
int ldap_unbind_s( LDAP *ld );
```

パラメータ

表 3-7 セッションをクローズするためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。

使用方法

ldap_unbind_ext()、ldap_unbind() および ldap_unbind_s() は、サーバーにバインド解除要求を送信し、LDAP セッション・ハンドルに関連したオープン状態の接続をすべてクローズし、そのセッション・ハンドルに関連したすべてのリソースを解放してから制御を戻します。その意味では、これらのファンクションはすべて同期して動作します。ただし、LDAP バインド解除操作に対するサーバーからの応答はないことに注意してください。これら3つのバインド解除ファンクションは、LDAP_SUCCESS（要求が LDAP サーバーに送信できなかった場合は別の LDAP エラー・コード）を戻します。これらのバインド解除ファンクションのいずれかをコールすると、セッション・ハンドル ld が無効になるため、これ以降に、ld を使用して LDAP API コールを行うことはできません。

`ldap_unbind()` ファンクションと `ldap_unbind_s()` ファンクションは同じように動作します。
`ldap_unbind_ext()` ファンクションを使用すると、サーバー・コントロールとクライアント・コントロールを明示的に含めることができますが、バインド解除要求に対するサーバーからの応答がないため、バインド解除要求で送信されたサーバー・コントロールに対する応答を受信する方法はないことに注意してください。

LDAP 操作の実行

`ldap_search_ext`

`ldap_search_ext_s`

`ldap_search`

`ldap_search_s`

`ldap_search_st`

これらのファンクションを使用して LDAP ディレクトリを検索し、一致した各エントリについて要求された属性セットを戻します。

構文

```
int ldap_search_ext
(
    LDAP          *ld,
    const char    *base,
    int           scope,
    const char    *filter,
    char          **attrs,
    int           attronly,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls,
    struct timeval *timeout,
    int           sizelimit,
    int           msgidp
);

int ldap_search_ext_s
(
    LDAP          *ld,
    const char    *base,
    int           scope,
```



```
        const char    *filter,
        char          **attrs,
        int           attrsonly,
        LDAPControl    **serverctrls,
        LDAPControl    **clientctrls,
        struct timeval *timeout,
        int            sizelimit,
        LDAPMessage     **res
    );

int ldap_search
(
    LDAP          *ld,
    const char    *base,
    int           scope,
    const char    *filter,
    char          **attrs,
    int           attrsonly
);

int ldap_search_s
(
    LDAP          *ld,
    const char    *base,
    int           scope,
    const char    *filter,
    char          **attrs,
    int           attrsonly,
    LDAPMessage     **res
);

int ldap_search_st
(
    LDAP          *ld,
    const char    *base,
    int           scope,
    const char    *filter,
    char          **attrs,
    int           attrsonly,
    struct timeval *timeout,
    LDAPMessage     **res
);
```

パラメータ

表 3-8 検索操作のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
base	検索の開始点となるエントリの識別名です。
scope	LDAP_SCOPE_BASE (0x00)、LDAP_SCOPE_ONELEVEL (0x01) または LDAP_SCOPE_SUBTREE (0x02) のいずれかで、検索範囲を指定します。
filter	検索フィルタを表す文字列です。この値を NULL にすると、すべてのエントリに一致するフィルタ (objectclass=*) を使用するよう指定できます。API のコール元で LDAP v2 を使用している場合、正常に使用できるのはフィルタ機能のサブセットのみであることに注意してください。
attrs	一致した各エントリのどの属性を戻すかを指定する文字列の NULL 終了配列です。このパラメータを NULL にすると、取得可能なユーザー属性がすべて取り出されます。文字列 LDAP_NO_ATTRS ("1.1") を配列内の唯一の文字列として使用すると、サーバーから属性型を戻さないように指定できます。attrs 配列の中で、文字列 LDAP_ALL_USER_ATTRS ("*") をなんらかの操作属性名とともに使用すると、すべてのユーザー属性に加えて、リストした操作属性を戻すように指定できます。
attrsonly	属性の型と値の両方を戻す場合には 0 (ゼロ)、属性の型のみを要求する場合には 0 (ゼロ) 以外を指定する必要があるブール値です。

表 3-8 検索操作のためのパラメータ（続き）

パラメータ	説明
timeout	ldap_search_st() ファンクションの場合は、このパラメータによって、ローカルの検索タイムアウト値を指定します（値が NULL の場合、タイムアウトは無限です）。タイムアウト値 0（ゼロ）が渡されると（tv_sec および tv_usec の両方が 0（ゼロ））、API 実装は LDAP_PARAM_ERROR を戻す必要があります。ldap_search_ext() ファンクションと ldap_search_ext_s() ファンクションの場合は、timeout パラメータによって、ローカルの検索タイムアウト値と検索要求でサーバーに送信される操作制限時間を指定します。timeout パラメータに NULL 値を渡すと、ローカルの無限検索タイムアウト値は使用されずに、LDAP セッション・ハンドルに格納されているグローバルなデフォルト・タイムアウト値（ldap_set_option() の LDAP_OPT_TIMELIMIT パラメータで設定）が要求とともに送信されます。タイムアウト値 0（ゼロ）が渡されると（tv_sec および tv_usec の両方が 0（ゼロ））、API 実装は LDAP_PARAM_ERROR を戻す必要があります。tv_sec の値は 0（ゼロ）だが、tv_usec の値は 0（ゼロ）以外の場合は、操作制限時間として 1 を LDAP サーバーに渡す必要があります。tv_sec の値が 0（ゼロ）以外の場合は、tv_sec の値自体を LDAP サーバーに渡す必要があります。
sizelimit	ldap_search_ext() コールと ldap_search_ext_s() コールの場合は、検索によって戻されるエントリの数をこのパラメータで制限します。LDAP_NO_LIMIT (0) 値は上限がないことを意味します。
res	同期コールを行うとき、コールの終了時に検索結果が入る結果パラメータです。戻される結果がない場合、*res は NULL に設定されます。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。

表 3-8 検索操作のためのパラメータ（続き）

パラメータ	説明
msgidp	<p>ldap_search_ext() コールが正常終了した場合、この結果パラメータは要求のメッセージ ID に設定されます。検索の実行方法に影響する可能性があるセッション・ハンドル ld には、3 つのオプションがあります。3 つのオプションは、次のとおりです。</p> <ul style="list-style-type: none"> ■ LDAP_OPT_SIZELIMIT – 検索で戻されるエントリの最大数です。LDAP_NO_LIMIT (0) 値は上限がないことを意味します。ldap_search_ext() ファンクションまたは ldap_search_ext_s() ファンクションを使用する場合、セッション・ハンドルの値は無視されることに注意してください。 ■ LDAP_OPT_TIMELIMIT – 検索を実行する最大秒数です。LDAP_NO_LIMIT (0) 値は上限がないことを意味します。ldap_search_ext() ファンクションまたは ldap_search_ext_s() ファンクションを使用する場合、セッション・ハンドルの値は無視されることに注意してください。 ■ LDAP_OPT_DEREF – LDAP_DEREF_NEVER (0x00)、LDAP_DEREF_SEARCHING (0x01)、LDAP_DEREF_FINDING (0x02) または LDAP_DEREF_ALWAYS (0x03) のいずれかで、検索時の別名の処理方法を指定します。LDAP_DEREF_SEARCHING 値の場合、別名は検索時に参照解除されますが、検索のベース・オブジェクトの検索時は参照解除されません。LDAP_DEREF_FINDING 値の場合、別名はベース・オブジェクトの検索時に参照解除されますが、検索時は参照解除されません。

使用方法

ldap_search_ext() ファンクションは、非同期の検索操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_search_ext() によって要求のメッセージ ID が *msgidp に格納されます。後で ldap_result()（詳細は後述）をコールすると、検索の結果を取得できます。検索結果は、後述する結果解析ルーチンを使用して解析できます。

ldap_search_ext() ファンクションと同様に、ldap_search() ファンクションは非同期の検索操作を開始し、開始した操作のメッセージ ID を戻します。ldap_search_ext() の場合は、後で ldap_result()（詳細は後述）をコールすると、バインドの結果を取得できます。エラーが発生した場合、ldap_search() は -1 を返し、LDAP 構造体に適切なセッション・エラー・パラメータを設定します。

同期型の ldap_search_ext_s()、ldap_search_s() および ldap_search_st() の各ファンクションはいずれも、操作が正常終了した場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。検索によって戻されるエントリがある場合、res パラ

メータの中に収められます。このパラメータはコール元に対しては不透明です。エントリ、属性、値などは、後述の解析ルーチンをコールすることで抽出できます。`res` に格納された結果が不要になった場合は、後述する `ldap_msgfree()` をコールして解放する必要があります。

`ldap_search_ext()` ファンクションと `ldap_search_ext_s()` ファンクションは、LDAP v3 のサーバー・コントロールとクライアント・コントロールをサポートし、各検索操作に対して様々なサイズと制限時間を簡単に指定できます。`ldap_search_st()` ファンクションは、検索のローカル・タイムアウトを指定するパラメータがあること以外は、`ldap_search_s()` ファンクションと同じです。ローカル検索タイムアウトは、検索完了まで API 実装が待機する時間を制限するために使用します。ローカル検索タイムアウトを経過すると、API 実装は中止操作を送信して検索操作を中止します。

エントリの読み込み

LDAP では、読み込み操作を直接サポートしていません。かわりに、この読み込み操作は、ベースを読み込むエントリの識別名に設定し、有効範囲を `LDAP_SCOPE_BASE` に設定し、さらにフィルタを `"(objectclass=*)"` または `NULL` に設定した検索によってエミュレーションを行います。`attrs` には、戻される属性のリストが格納されます。

エントリの子のリスト表示

LDAP では、リスト操作を直接サポートしていません。かわりに、このリスト操作は、ベースをリスト表示するエントリの識別名に設定し、有効範囲を `LDAP_SCOPE_ONELEVEL` に設定し、さらにフィルタを `"(objectclass=*)"` または `NULL` に設定した検索によってエミュレーションを行います。`attrs` には、子エントリごとに戻される属性のリストが格納されます。

`ldap_compare_ext`

`ldap_compare_ext_s`

`ldap_compare`

`ldap_compare_s`

これらのルーチンを使用して、指定の属性値のアサーションを LDAP エントリと照合して比較します。

構文

```
int ldap_compare_ext
(
    LDAP                *ld,
    const char          *dn,
    const char          *attr,
    const struct berval *bvalue,
    LDAPControl         **serverctrls,
    LDAPControl         **clientctrls,
    int                 *msgidp
);

int ldap_compare_ext_s
(
    LDAP                *ld,
    const char          *dn,
    const char          *attr,
    const struct berval *bvalue,
    LDAPControl         **serverctrls,
    LDAPControl         **clientctrls
);

int ldap_compare
(
    LDAP                *ld,
    const char          *dn,
    const char          *attr,
    const char          *value
);

int ldap_compare_s
(
    LDAP                *ld,
    const char          *dn,
    const char          *attr,
    const char          *value
);
```

パラメータ

表 3-9 比較操作のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
dn	比較の対象となるエントリの名前です。
attr	比較の対象となる属性です。

表 3-9 比較操作のためのパラメータ（続き）

パラメータ	説明
bvalue	指定のエントリで検索された属性と照合して比較される属性値です。このパラメータは拡張ルーチンで使用され、berval 構造体へのポインタとなるため、バイナリ値と比較できます。
value	比較の対象となる文字列の属性値で、ldap_compare() ファンクションと ldap_compare_s() ファンクションで使用します。バイナリ値と比較する必要がある場合は、ldap_compare_ext() または ldap_compare_ext_s() を使用します。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。
msgidp	ldap_compare_ext() コールが正常終了した場合は、この結果パラメータが要求のメッセージ ID に設定されます。

使用方法

ldap_compare_ext() ファンクションは、非同期の比較操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_compare_ext() によって要求のメッセージ ID が *msgidp に格納されます。後で ldap_result()（詳細は後述）をコールすると、比較の結果を取得できます。

ldap_compare_ext() ファンクションと同様に、ldap_compare() ファンクションは非同期の比較操作を開始し、開始した操作のメッセージ ID を戻します。ldap_compare_ext() の場合は、後で ldap_result()（詳細は後述）をコールすると、バインドの結果を取得できます。エラーが発生した場合、ldap_compare() は -1 を返し、LDAP 構造体に適切なセッション・エラー・パラメータを設定します。

同期型の ldap_compare_ext_s() ファンクションおよび ldap_compare_s() ファンクションはいずれも、操作が正常終了した場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。

ldap_compare_ext() ファンクションと ldap_compare_ext_s() ファンクションは、LDAP v3 のサーバー・コントロールとクライアント・コントロールをサポートします。

ldap_modify_ext

ldap_modify_ext_s

ldap_modify

ldap_modify_s

これらのルーチンを使用して、既存の LDAP エントリを変更します。

構文

```
typedef struct ldapmod
{
    int          mod_op;
    char         *mod_type;
    union mod_vals_u
    {
        char          **modv_strvals;
        struct berval **modv_bvals;
    } mod_vals;
} LDAPMod;
#define mod_values      mod_vals.modv_strvals
#define mod_bvalues    mod_vals.modv_bvals

int ldap_modify_ext
(
    LDAP          *ld,
    const char    *dn,
    LDAPMod       **mods,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls,
    int           *msgidp
);

int ldap_modify_ext_s
(
    LDAP          *ld,
    const char    *dn,
    LDAPMod       **mods,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls
);

int ldap_modify
(
```



```

        LDAP      *ld,
        const char *dn,
        LDAPMod   **mods
    );

int ldap_modify_s
(
    LDAP      *ld,
    const char *dn,
    LDAPMod   **mods
);

```

パラメータ

表 3-10 変更操作のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
dn	変更するエントリの名前です。
mods	エントリに対する変更の NULL 終了配列です。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。
msgidp	ldap_modify_ext() コールが正常終了した場合は、この結果パラメータが要求のメッセージ ID に設定されます。

LDAPMod 構造体のフィールドについて次に説明します。

表 3-11

フィールド	説明
mod_op	実行する変更操作です。LDAP_MOD_ADD (0x00)、LDAP_MOD_DELETE (0x01) または LDAP_MOD_REPLACE (0x02) のいずれかになります。このフィールドは、mod_vals 共用体に格納されている値のタイプも示します。mod_bvalues 形式を選択するには、LDAP_MOD_BVALUES (0x80) との論理和をとります。それ以外の場合は、mod_values 形式が使用されます。

表 3-11（続き）

フィールド	説明
mod_type	変更する属性の型です。
mod_vals	追加、削除または置換する値（ある場合）です。mod_op フィールドと定数 LDAP_MOD_BVALUES との論理和によって選択された mod_values または mod_bvalues のいずれかの変数のみ使用できます。mod_values はゼロ終了文字列の NULL 終了配列です。mod_bvalues は berval 構造体の NULL 終了配列で、イメージなどのバイナリ値を渡すために使用できます。

使用方法

LDAP_MOD_ADD 変更の場合は、指定の値がエントリに追加され、必要に応じて属性が作成されます。

LDAP_MOD_DELETE 変更の場合は、指定の値がエントリから削除され、値がない場合は属性が削除されます。すべての属性を削除する場合は、mod_vals フィールドを NULL に設定できます。

LDAP_MOD_REPLACE 変更の場合は、変更後にリストされた値が属性に設定されます。この属性は必要に応じて作成され、mod_vals フィールドが NULL の場合は削除されます。すべての変更は、リストされた順序で実行されます。

ldap_modify_ext() ファンクションは、非同期の変更操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_modify_ext() によって要求のメッセージ ID が *msgidp に格納されます。後で ldap_result()（詳細は後述）をコールすると、変更の結果を取得できます。

ldap_modify_ext() ファンクションと同様に、ldap_modify() ファンクションは非同期の変更操作を開始し、開始した操作のメッセージ ID を戻します。ldap_modify_ext() の場合は、後で ldap_result()（詳細は後述）をコールすると、変更の結果を取得できます。エラーが発生した場合、ldap_modify() は -1 を戻し、LDAP 構造体に適切なセッション・エラー・パラメータを設定します。

同期型の ldap_modify_ext_s() ファンクションおよび ldap_modify_s() ファンクションは、いずれも、操作が正常終了した場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。

ldap_modify_ext() ファンクションと ldap_modify_ext_s() ファンクションは、LDAP v3 のサーバー・コントロールとクライアント・コントロールをサポートします。

ldap_rename

ldap_rename_s

これらのルーチンを使用して、エントリ名を変更します。

```
int ldap_rename
(
    LDAP          *ld,
    const char    *dn,
    const char    *newrdn,
    const char    *newparent,
    int           deleteoldrdn,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls,
    int           *msgidp
);
```

```
int ldap_rename_s
(
    LDAP          *ld,
    const char    *dn,
    const char    *newrdn,
    const char    *newparent,
    int           deleteoldrdn,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls
);
```

次のルーチンは使用できません。詳細は、RFC 1823 を参照してください。

```
int ldap_modrdn
(
    LDAP          *ld,
    const char    *dn,
    const char    *newrdn
);
int ldap_modrdn_s
(
    LDAP          *ld,
    const char    *dn,
    const char    *newrdn
);
int ldap_modrdn2
(
    LDAP          *ld,
    const char    *dn,
    const char    *newrdn,
    int           deleteoldrdn
);
```

```
);
int ldap_modrdn2_s
(
    LDAP          *ld,
    const char    *dn,
    const char    *newrdn,
    int           deleteoldrdn
);
```

パラメータ

表 3-12 改名操作のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
dn	識別名を変更するエントリの名前です。
newrdn	エントリに指定する新規の相対識別名です。
newparent	新規の親、つまり上位のエントリです。このパラメータが NULL の場合は、エントリの相対識別名のみが変更されます。ルート of 識別名は、長さ 0 (ゼロ) の文字列 "" を渡して指定する必要があります。バージョン 2 の LDAP プロトコルを使用するときは、新規の親パラメータは常に NULL にする必要があります。それ以外の場合、サーバーの動作は未定義になります。
deleteoldrdn	newrdn が古い相対識別名と異なる場合、このパラメータは改名ルーチンでのみ使用されます。このパラメータはブール値で、0 (ゼロ) 以外の場合は古い相対識別名が削除されたことを示し、0 (ゼロ) の場合は、エントリの識別されない値として古い相対識別名が保持されていることを示します。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。
msgidp	ldap_rename() コールが正常終了した場合は、この結果パラメータが要求のメッセージ ID に設定されます。

使用方法

ldap_rename() ファンクションは、非同期の識別名変更操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_rename() によって要求の識別名メッセージ ID が *msgidp に格納されます。後で ldap_result() (詳細は後述) をコールすると、改名の結果を取得できます。

同期型の `ldap_rename_s()` ファンクションは、操作が正常終了した場合は定数 `LDAP_SUCCESS` を、失敗した場合は別の `LDAP` エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。

`ldap_rename()` ファンクションと `ldap_rename_s()` ファンクションは、`LDAP v3` のサーバー・コントロールとクライアント・コントロールをサポートします。

ldap_add_ext

ldap_add_ext_s

ldap_add

ldap_add_s

これらのファンクションを使用して、`LDAP` ディレクトリにエントリを追加します。

構文

```
int ldap_add_ext
(
    LDAP          *ld,
    const char    *dn,
    LDAPMod       **attrs,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls,
    int           *msgidp
);

int ldap_add_ext_s
(
    LDAP          *ld,
    const char    *dn,
    LDAPMod       **attrs,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls
);

int ldap_add
(
    LDAP          *ld,
    const char    *dn,
    LDAPMod       **attrs
);
```

```
int ldap_add_s
(
    LDAP          *ld,
    const char    *dn,
    LDAPMod       **attrs
);
```

パラメータ

表 3-13 追加操作のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
dn	追加するエントリの名前です。
attrs	エントリの属性です。ldap_modify() で定義した LDAPMod 構造体を使用して指定します。mod_type フィールドと mod_vals フィールドを入力する必要があります。定数 LDAP_MOD_BVALUES との論理和がとられ、mod_vals 共用体の mod_bvalues ケースの選択に使用されないかぎり、mod_op フィールドは無視されます。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。
msgidp	ldap_add_ext() コールが正常終了した場合は、この結果パラメータが要求のメッセージ ID に設定されます。

使用方法

追加操作を正常に行うためには、追加するエントリの親がすでに存在しているか、親が空（つまり、ルートの識別名と同じ）であることが必要です。

ldap_add_ext() ファンクションは、非同期の追加操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを返します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_add_ext() によって要求のメッセージ ID が *msgidp に格納されます。後で ldap_result()（詳細は後述）をコールすると、追加の結果を取得できます。

ldap_add_ext() ファンクションと同様に、ldap_add() ファンクションは非同期の追加操作を開始し、開始した操作のメッセージ ID を返します。ldap_add_ext() の場合は、後で ldap_result()（詳細は後述）をコールすると、追加の結果を取得できます。エラーが発生した場合、ldap_add() は -1 を返し、LDAP 構造体に適切なセッション・エラー・パラメータを設定します。

同期型の `ldap_add_ext_s()` ファンクションおよび `ldap_add_s()` ファンクションはいずれも、操作が正常終了した場合は定数 `LDAP_SUCCESS` を、失敗した場合は別の LDAP エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。

`ldap_add_ext()` ファンクションと `ldap_add_ext_s()` ファンクションは、LDAP v3 のサーバー・コントロールとクライアント・コントロールをサポートします。

ldap_delete_ext

ldap_delete_ext_s

ldap_delete

ldap_delete_s

これらのファンクションを使用して、LDAP ディレクトリからリーフ・エントリを削除します。

構文

```
int ldap_delete_ext
(
    LDAP          *ld,
    const char     *dn,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls,
    int            *msgidp
);

int ldap_delete_ext_s
(
    LDAP          *ld,
    const char     *dn,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls
);

int ldap_delete
(
    LDAP          *ld,
    const char     *dn
);
```

```
int ldap_delete_s
(
    LDAP          *ld,
    const char    *dn
);
```

パラメータ

表 3-14 削除操作のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
dn	削除するエントリの名前です。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。
msgidp	ldap_delete_ext() コールが正常終了した場合は、この結果パラメータが要求のメッセージ ID に設定されます。

使用方法

削除するエントリは、リーフ・エントリ（つまり、子エントリがないエントリ）であることが必要です。1 回の操作でサブツリー全体を削除する操作は、LDAP ではサポートされていません。

ldap_delete_ext() ファンクションは、非同期の削除操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_delete_ext() によって要求のメッセージ ID が *msgidp に格納されます。後で ldap_result()（詳細は後述）をコールすると、削除の結果を取得できます。

ldap_delete_ext() ファンクションと同様に、ldap_delete() ファンクションは非同期の削除操作を開始し、開始した操作のメッセージ ID を戻します。ldap_delete_ext() の場合は、後で ldap_result()（詳細は後述）をコールすると、削除の結果を取得できます。エラーが発生した場合、ldap_delete() は -1 を戻し、LDAP 構造体に適切なセッション・エラー・パラメータを設定します。

同期型の ldap_delete_ext_s() ファンクションおよび ldap_delete_s() ファンクションはいずれも、操作が正常終了した場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。

ldap_delete_ext() ファンクションと ldap_delete_ext_s() ファンクションは、LDAP v3 のサーバー・コントロールとクライアント・コントロールをサポートします。

ldap_extended_operation

ldap_extended_operation_s

これらのルーチンを使用すると、拡張 LDAP 操作をサーバーに渡し、一般的なプロトコル拡張メカニズムを提供できます。

構文

```
int ldap_extended_operation
(
    LDAP                *ld,
    const char          *requestoid,
    const struct berval *requestdata,
    LDAPControl         **serverctrls,
    LDAPControl         **clientctrls,
    int                 *msgidp
);

int ldap_extended_operation_s
(
    LDAP                *ld,
    const char          *requestoid,
    const struct berval *requestdata,
    LDAPControl         **serverctrls,
    LDAPControl         **clientctrls,
    char                **retoidp,
    struct berval        *retdatap
);
```

パラメータ

表 3-15 拡張操作のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
requestoid	要求を指定する、ドット表記の OID テキスト文字列です。
requestdata	操作に必要な任意のデータです (NULL の場合、サーバーにデータは送信されません)。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。
msgidp	ldap_extended_operation() コールが正常終了した場合は、この結果パラメータが要求のメッセージ ID に設定されます。

表 3-15 拡張操作のためのパラメータ（続き）

パラメータ	説明
retoidp	文字列へのポインタです。この文字列は、サーバーから戻された割当て済み、ドット表記の OID テキスト文字列に設定される文字列です。この文字列は、ldap_memfree() ファンクションを使用して解放する必要があります。OID が戻らない場合、*retoidp は NULL に設定されます。
retdatap	berval 構造体ポインタへのポインタです。このポインタは、サーバーから戻されたデータの割当て済みコピーに設定されます。この berval 構造体は、ber_bvfree() ファンクションを使用して解放する必要があります。データが戻らない場合、*retdatap は NULL に設定されます。

使用方法

ldap_extended_operation() ファンクションは、非同期の拡張操作を開始し、要求が正常に送信された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。正常に送信された場合は、ldap_extended_operation() によって要求のメッセージ ID が *msgidp に格納されます。後で ldap_result()（詳細は後述）をコールすると、拡張操作の結果を取得できます。この結果を ldap_parse_extended_result() に渡すと、結果に含まれている OID とデータを取得できます。

同期型の ldap_extended_operation_s() ファンクションは、操作が正常終了した場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを操作の結果として戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。**retoid** パラメータと **retdata** パラメータには、結果に含まれている OID とデータが入力されます。戻される OID またはデータがない場合、これらのパラメータは NULL に設定されます。

ldap_extended_operation() ファンクションと ldap_extended_operation_s() ファンクションは、LDAP v3 のサーバー・コントロールとクライアント・コントロールをサポートします。

操作の中止

ldap_abandon_ext

ldap_abandon

これらのコールを使用して、進行中の操作を中止します。

構文

```

int ldap_abandon_ext
(
    LDAP          *ld,
    int           msgid,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls
);

int ldap_abandon
(
    LDAP          *ld,
    int           msgid
);

```

パラメータ**表 3-16 操作を中止するためのパラメータ**

パラメータ	説明
ld	セッション・ハンドルです。
msgid	中止する要求のメッセージ ID です。
serverctrls	LDAP サーバー・コントロールのリストです。
clientctrls	クライアント・コントロールのリストです。

使用方法

ldap_abandon_ext() フังก์ションは、msgid パラメータのメッセージ ID を使用して操作を中止し、中止操作が正常終了した場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。エラーとその解析の詳細は、後のエラー処理の項を参照してください。

ldap_abandon() は、クライアント・コントロールまたはサーバー・コントロールを受け入れないこと以外は ldap_abandon_ext() と同じで、中止操作が正常終了した場合は 0（ゼロ）を、失敗した場合は -1 を戻します。

ldap_abandon() コールまたは ldap_abandon_ext() コールが正常終了した後、指定のメッセージ ID を持つ結果は、引き続き ldap_result() をコールしても戻されません。LDAP 中止操作に対するサーバーの応答はありません。

結果の取得と LDAP メッセージの確認

ldap_result

ldap_msgfree

ldap_msgtype

ldap_msgid

ldap_result() を使用して、前に非同期で開始した操作の結果を取得します。**ldap_result()** は、コール方法に応じて、結果メッセージのリスト、つまり、結果セットを実際に戻すことができます。**ldap_result()** ファンクションは、単一の要求に対するメッセージのみ戻します。このため、検索操作以外のすべての LDAP 操作で戻される結果メッセージは 1 つのみです。つまり、結果セットに複数のメッセージが含まれるのは、検索操作の結果が戻された場合のみです。

コール元に戻された結果セットは、そのセットを生成した LDAP 要求との関連をコール元で表示する方法はありません。したがって、**ldap_result()** または同期検索ルーチンをコールして戻された結果セットは、後続の LDAP API コールの影響を受けることはありません（結果セットを解放する **ldap_msgfree()** は除きます）。

ldap_msgfree() は、**ldap_result()** または同期検索ルーチンをコールして前に取得した結果メッセージ（結果セット全体の場合もあります）を解放します。

ldap_msgtype() は LDAP メッセージのタイプを戻します。**ldap_msgid()** は LDAP メッセージのメッセージ ID を戻します。

構文

```
int ldap_result
(
    LDAP          *ld,
    int           msgid,
    int           all,
    struct timeval *timeout,
    LDAPMessage   **res
);
int ldap_msgfree( LDAPMessage *res );
int ldap_msgtype( LDAPMessage *res );
int ldap_msgid( LDAPMessage *res );
```

パラメータ

表 3-17 結果の取得と LDAP メッセージの確認のためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
msgid	結果が戻される操作のメッセージ ID、定数 LDAP_RES_UNSOLICITED (0) (要求に基づかない結果を取得する場合)、または定数 LDAP_RES_ANY (-1) (任意の結果を取得する場合) です。
all	1 回の ldap_result() コールで取得するメッセージの数を指定します。このパラメータは、検索結果を取得する場合にのみ使用されます。定数 LDAP_MSG_ONE (0x00) を渡して、一度に 1 つのメッセージを取得します。すべての結果が 1 つの結果セットとして戻される前に、すべての検索結果を取得するには、LDAP_MSG_ALL (0x01) を渡します。すでに取得したすべてのメッセージを結果セットに戻すには、LDAP_MSG_RECEIVED (0x02) を渡します。
timeout	結果の戻りに対する待機時間を指定するタイムアウトです。NULL 値を指定すると、ldap_result() は結果が戻るまでブロックされます。タイムアウト値に 0 (ゼロ) 秒を指定すると、ポーリング動作になります。
res	ldap_result() の場合は、操作の結果が含まれる結果パラメータです。戻される結果がない場合、*res は NULL に設定されます。ldap_msgfree() の場合は、ldap_result()、ldap_search_s() または ldap_search_st() をコールして前に取得し、解放する結果セットです。res を NULL に設定すると何も処理されず、ldap_msgfree() は 0 (ゼロ) を戻します。

使用方法

正常終了すると、ldap_result() は戻された最初の結果のタイプを res パラメータに戻します。次のいずれかの定数が戻されます。

LDAP_RES_BIND (0x61)

LDAP_RES_SEARCH_ENTRY (0x64)

LDAP_RES_SEARCH_REFERENCE (0x73) — LDAP v3 の新規定数

LDAP_RES_SEARCH_RESULT (0x65)

LDAP_RES_MODIFY (0x67)

LDAP_RES_ADD (0x69)

LDAP_RES_DELETE (0x6B)

LDAP_RES_MODDN (0x6D)

LDAP_RES_COMPARE (0x6F)

LDAP_RES_EXTENDED (0x78) — LDAP v3 の新規定数

`ldap_result()` は、タイムアウトを超過した場合は 0（ゼロ）を、エラーが発生した場合は -1 を返します。エラーの発生に応じて、LDAP セッション・ハンドルのエラー・パラメータが設定されます。

`ldap_msgfree()` は、`res` パラメータが指し示す結果セット内の各メッセージを解放し、最後のメッセージのタイプを返します。`res` が NULL の場合は何も処理されず、値 0（ゼロ）が返されます。

`ldap_msgtype()` は、パラメータとして渡される LDAP メッセージのタイプを返します。前述のタイプのいずれかを返します。エラーの場合は -1 を返します。

`ldap_msgid()` は、パラメータとして渡された LDAP メッセージに対応付けられているメッセージ ID を返します。エラーの場合は -1 を返します。

エラーの処理と結果の解析

`ldap_parse_result`

`ldap_parse_sasl_bind_result`

`ldap_parse_extended_result`

`ldap_err2string`

これらのコールを使用して、結果から情報を抽出し、他の LDAP API ルーチンによって戻されたエラーを処理します。`ldap_parse_sasl_bind_result()` および `ldap_parse_extended_result()` は、通常、`ldap_parse_result()` とともに使用して、SASL バインド操作および拡張操作の結果情報をそれぞれ取得することに注意してください。

構文

```
int ldap_parse_result
(
    LDAP          *ld,
    LDAPMessage   *res,
    int           *errcodep,
    char          **matcheddn,
    char          **errmsgp,
    char          ***referralsp,
    LDAPControl   ***serverctrlsp,
```

```
        int          freeit
    );

int ldap_parse_sasl_bind_result
(
    LDAP          *ld,
    LDAPMessage    *res,
    struct berval  **servercredp,
    int          freeit
);
```

```
int ldap_parse_extended_result
(
    LDAP          *ld,
    LDAPMessage    *res,
    char          **retoidp,
    struct berval  **retdatap,
    int          freeit
);
#define LDAP_NOTICE_OF_DISCONNECTION    "1.3.6.1.4.1.1466.20036"
char *ldap_err2string( int err );
```

次のルーチンは使用できません。詳細は、RFC 1823 を参照してください。

```
int ldap_result2error
(
    LDAP          *ld,
    LDAPMessage    *res,
    int          freeit
);
void ldap_perror( LDAP *ld, const char *msg );
```

パラメータ

表 3-18 エラーの処理と結果の解析を行うためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
res	ldap_result() または API 操作の同期コールの 1 つから戻された LDAP 操作の結果です。
errcodep	この結果パラメータには、LDAPMessage メッセージの LDAP エラー・コード・フィールドの値が入力されます。これは、サーバーでの操作の結果を示します。このフィールドを無視する場合は、NULL を渡す必要があります。
matcheddn	LDAP_NO_SUCH_OBJECT が戻された場合、この結果パラメータには、要求内の名前が認識された程度を示す識別名が入力されます。このフィールドを無視する場合は、NULL を渡す必要があります。一致した識別名の文字列は、このマニュアルで前述した ldap_memfree() をコールして解放する必要があります。
errmsgp	この結果パラメータには、LDAPMessage メッセージのエラー・メッセージ・フィールドの内容が入力されます。エラー・メッセージ・ストリングは、このマニュアルで前に説明した ldap_memfree() をコールして解放する必要があります。このフィールドを無視する場合は、NULL を渡す必要があります。
referralsp	この結果パラメータには、LDAPMessage メッセージの参照フィールドの内容が入力され、要求を再試行するための代替 LDAP サーバーの有無が示されます。この参照配列は、このマニュアルで前に説明した ldap_value_free() をコールして解放する必要があります。このフィールドを無視する場合は、NULL を渡す必要があります。
serverctrlsp	この結果パラメータには、LDAPMessage メッセージからコピーされたコントロールの割当て済み配列が入力されます。このコントロール配列は、前に説明した ldap_controls_free() をコールして解放する必要があります。
freeit	res パラメータを解放するかどうかを判断するブール値です。0（ゼロ）以外の値を渡すと、これらのルーチンは要求された情報を抽出した後に res パラメータを解放します。このパラメータは便宜的に用意されたものです。結果は、後で ldap_msgfree() を使用して解放することもできます。freeit が 0（ゼロ）以外の場合は、res パラメータで示される結果セット全体が解放されます。

表 3-18 エラーの処理と結果の解析を行うためのパラメータ（続き）

パラメータ	説明
servercredp	SASL バインド結果に関するこの結果パラメータには、相互認証が指定されている場合、サーバーから戻された資格証明が入力されます。割り当てられた <code>berval</code> 構造体が戻されるため、 <code>ber_bvfree()</code> をコールして解放する必要があります。このフィールドを無視する場合は、NULL を渡す必要があります。
retoidp	拡張操作に関するこの結果パラメータには、拡張操作の応答名を表現するドット表記の OID テキストが入力されます。この文字列は、 <code>ldap_memfree()</code> をコールして解放する必要があります。このフィールドを無視する場合は、NULL を渡す必要があります。 LDAP_NOTICE_OF_DISCONNECTION マクロは、クライアントに対して便宜的に定義されています。クライアントは、OID が要求に基づかない切断通知 (RFC 2251[2] のセクション 4.4.1 に定義) に使用する OID と一致しているかどうかをチェックします。
retdatap	拡張操作の結果に関するこの結果パラメータには、拡張操作の応答データが含まれる <code>berval</code> 構造体へのポインタが入力されます。このパラメータは、 <code>ber_bvfree()</code> をコールして解放する必要があります。このフィールドを無視する場合は、NULL を渡す必要があります。
err	<code>ldap_err2string()</code> に関する LDAP エラー・コードで、 <code>ldap_parse_result()</code> または他の LDAP API コールによって戻されます。

使用方法

使用できないルーチンに関するその他のパラメータは説明していません。詳細は、RFC 1823 を参照してください。

`ldap_parse_result()`、`ldap_parse_sasl_bind_result()` および `ldap_parse_extended_result()` の各ファンクションは、解析する結果メッセージの検索時に、メッセージ・タイプの LDAP_RES_SEARCH_ENTRY および LDAP_RES_SEARCH_REFERENCE をスキップします。これらのファンクションは、結果が正常に解析された場合は定数 LDAP_SUCCESS を、失敗した場合は別の LDAP エラー・コードを戻します。サーバーで実行された操作の結果を示す LDAP エラー・コードは、`ldap_parse_result()` の `errcodep` パラメータに格納されることに注意してください。複数の結果メッセージが含まれた結果セットがこれらのルーチンに渡されている場合、これらのルーチンは、常にその結果セット内の最初の結果から操作を開始します。

`ldap_err2string()` は、`ldap_parse_result()`、`ldap_parse_sasl_bind_result()`、`ldap_parse_extended_result()` または API 操作の同期コールの 1 つから戻された LDAP エラー・コード（数値）を、エラー説明のためのゼロ終了文字列メッセージに変換するために使用されます。このファンクションは、静的データへのポインタを戻します。

結果リストの参照

ldap_first_message

ldap_next_message

これらのルーチンを使用して、ldap_result() で戻された結果セット内のメッセージ・リストを参照します。検索操作の場合、結果セットには、参照メッセージ、エントリ・メッセージおよび結果メッセージを実際に格納できます。

ldap_count_messages() は、戻されたメッセージの件数をカウントします。前に説明した ldap_msgtype() フังก์ションを使用すると、異なるメッセージ・タイプを区別できます。

```
LDAPMessage *ldap_first_message( LDAP *ld, LDAPMessage *res );
LDAPMessage *ldap_next_message( LDAP *ld, LDAPMessage *msg );
int ldap_count_messages( LDAP *ld, LDAPMessage *res );
```

パラメータ

表 3-19 結果リストを参照するためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
res	同期検索ルーチンのいずれか、または ldap_result() をコールして取得する結果セットです。
msg	ldap_first_message() または ldap_next_message() のコールで前に戻されたメッセージです。

使用方法

ldap_first_message() および ldap_next_message() は、戻された結果セットにメッセージがそれ以上存在しない場合、NULL を戻します。エントリの参照中にエラーが発生した場合も NULL が戻されます。この場合は、エラーを示すためにセッション・ハンドル ld のエラー・パラメータが設定されます。

正常終了した場合、ldap_count_messages() は結果セット内のメッセージ数を戻します。res パラメータが無効な場合など、なんらかのエラーが発生した場合は、-1 が戻されます。ldap_first_message()、ldap_next_message()、ldap_first_entry()、ldap_next_entry()、ldap_first_reference()、ldap_next_reference() によって戻されたメッセージ、エントリまたはリファレンスについて ldap_count_messages() をコールする場合は、結果セットの残りのメッセージ件数をカウントすることもできます。

検索結果の解析

次のコールを使用して、`ldap_search()` および関連のファンクションで戻されるエントリやリファレンスを解析します。これらの結果は、次に説明するルーチンをコールしてアクセスできる不透明な構造体に戻されます。これらのルーチンは、戻されたエントリやリファレンスの参照、エントリの属性の参照、エントリ名の取得、およびエントリ内の指定した属性に対応付けられている値の取得を行うために用意されています。

ldap_first_entry

ldap_next_entry

ldap_first_reference

ldap_next_reference

ldap_count_entries

ldap_count_references

`ldap_first_entry()` ルーチンおよび `ldap_next_entry()` ルーチンは、エントリのリストを検索結果セットから参照して取得します。`ldap_first_reference()` ルーチンおよび `ldap_next_reference()` ルーチンは、継続リファレンスのリストを検索結果セットから参照して取得します。`ldap_count_entries()` は、戻されたエントリの件数をカウントします。`ldap_count_references()` は、戻されたリファレンスの件数をカウントします。

```
LDAPMessage *ldap_first_entry( LDAP *ld, LDAPMessage *res );
LDAPMessage *ldap_next_entry( LDAP *ld, LDAPMessage *entry );
LDAPMessage *ldap_first_reference( LDAP *ld, LDAPMessage *res );
LDAPMessage *ldap_next_reference( LDAP *ld, LDAPMessage *ref );
int ldap_count_entries( LDAP *ld, LDAPMessage *res );
int ldap_count_references( LDAP *ld, LDAPMessage *res );
```

パラメータ

表 3-20 エントリと継続リファレンスを検索結果セットから取得したり、戻されたエントリの件数をカウントするためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
res	検索結果です。同期検索ルーチンのいずれか、または ldap_result() をコールして取得されるものと同一です。
entry	ldap_first_entry() または ldap_next_entry() のコールで前に戻されたエントリです。
ref	ldap_first_reference() または ldap_next_reference() のコールで前に戻されたリファレンスです。

使用方法

ldap_first_entry()、ldap_next_entry()、ldap_first_reference() および ldap_next_reference() は、戻された結果セットにリファレンスがそれ以上存在しない場合、NULL を戻します。エントリまたはリファレンスの参照中にエラーが発生した場合も NULL が戻されます。この場合は、エラーを示すためにセッション・ハンドル ld のエラー・パラメータが設定されます。

ldap_count_entries() の戻り値は、エントリの結果セットに含まれるエントリの数です。res パラメータが無効な場合など、なんらかのエラーが発生した場合は、-1 が戻されます。ldap_first_message()、ldap_next_message()、ldap_first_entry()、ldap_next_entry()、ldap_first_reference()、ldap_next_reference() によって戻されたメッセージ、エントリまたはリファレンスについて ldap_count_entries() をコールする場合は、結果セットの残りのエントリ数をカウントすることもできます。

ldap_count_references() の戻り値は、結果セットに含まれるリファレンスの数です。res パラメータが無効な場合など、なんらかのエラーが発生した場合は、-1 が戻されます。ldap_count_references() をコールすると、結果セット内の残りのリファレンス件数もカウントできます。

ldap_first_attribute

ldap_next_attribute

これらのコールを使用して、エントリとともに戻される属性の型のリストを参照します。

```
char *ldap_first_attribute
(
    LDAP          *ld,
    LDAPMessage   *entry,
    BerElement     **ptr
);
```

```
char *ldap_next_attribute
(
    LDAP          *ld,
    LDAPMessage    *entry,
    BerElement     *ptr
);
void ldap_memfree( char *mem );
```

パラメータ

表 3-21 エントリとともに戻される属性の型を参照するためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
entry	属性を参照する対象となるエントリです。ldap_first_entry() または ldap_next_entry() の戻り値と同一です。
ptr	ldap_first_attribute() では、エントリ内の現在の位置を追跡管理するために内部で使用するポインタのアドレスです。ldap_next_attribute() では、ldap_first_attribute() のコールで前に戻されたポインタです。BerElement 型自体は不透明な構造体です。
mem	ldap_first_attribute() および ldap_next_attribute で戻される属性の型の名前や ldap_get_dn() で戻される識別名など、LDAP ライブラリによって割り当てられたメモリーへのポインタです。mem が NULL の場合は、ldap_memfree() をコールしても何も処理されません。

使用方法

ldap_first_attribute() および ldap_next_attribute() は、属性の最後に達したり、エラーが発生すると NULL を戻します。エラーが発生した場合は、そのエラーを示すためにセッション・ハンドル ld のエラー・パラメータが設定されます。

いずれのルーチンとも、現行の属性名が格納されている割当て済みバッファへのポインタを戻します。このポインタが不要になった場合は、ldap_memfree() をコールして解放する必要があります。

ldap_first_attribute() は、現在位置を追跡管理するために使用する BerElement へのポインタ (ptr パラメータ) を割り当てて戻します。このポインタは、エントリの属性を参照するために、後続の ldap_next_attribute() コールに渡すことができます。ldap_first_attribute() および ldap_next_attribute() の一連のコールを行った後に、ptr パラメータが NULL 以外の場合は、ber_free(ptr, 0) をコールしてこのパラメータを解放する必要があります。このコールでは、2 番目のパラメータとして 0 (ゼロ) を渡すことが重要です。これは、BerElement に対応付けられたバッファは、別に割り当てられたメモリーを指し示していないためです。

戻された属性の型の名前は、関連する値を取り出すための `ldap_get_values()` や関連する関クションのコールに渡すのに適しています。

ldap_get_values

ldap_get_values_len

ldap_count_values

ldap_count_values_len

ldap_value_free

ldap_value_free_len

`ldap_get_values()` および `ldap_get_values_len()` は、指定の属性の値をエントリから取得します。`ldap_count_values()` および `ldap_count_values_len()` は、戻された値の件数をカウントします。

`ldap_value_free()` および `ldap_value_free_len()` は、値を解放します。

構文

```
char **ldap_get_values
(
    LDAP          *ld,
    LDAPMessage   *entry,
    const char     *attr
);

struct berval **ldap_get_values_len
(
    LDAP          *ld,
    LDAPMessage   *entry,
    const char     *attr
);

int ldap_count_values( char **vals );
int ldap_count_values_len( struct berval **vals );
void ldap_value_free( char **vals );
void ldap_value_free_len( struct berval **vals );
```

パラメータ

表 3-22 属性値を取得してその件数をカウントするためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
entry	値を取得する元のエントリです。ldap_first_entry() または ldap_next_entry() の戻り値と同一です。
attr	値を取得する対象となる属性です。ldap_first_attribute() または ldap_next_attribute() の戻り値、またはコール元が提供する文字列 (たとえば、mail) と同一です。
vals	ldap_get_values() または ldap_get_values_len() のコールで前に戻された値です。

使用方法

2 つの形式のコールが用意されています。最初の形式は、バイナリ以外の文字列データに対してのみ使用できます。_len が付く 2 番目の形式は、あらゆる種類のデータで使用できます。

ldap_get_values() および ldap_get_values_len() は、attr パラメータの値がなかったり、エラーが発生した場合、NULL を返します。

ldap_count_values() および ldap_count_values_len() は、vals パラメータが無効だったり、エラーが発生した場合、-1 を返します。

NULL の vals パラメータが ldap_value_free() または ldap_value_free_len() に渡されても、何も処理されません。

戻された値は動的に割り当てられます。不要になった場合は、ldap_value_free() または ldap_value_free_len() をコールして解放する必要があります。

ldap_get_dn

ldap_explode_dn

ldap_explode_rdn

ldap_dn2ufn

ldap_get_dn() は、エントリの名前を取得します。ldap_explode_dn() および ldap_explode_rdn() は、名前を構成要素に分割します。ldap_dn2ufn() は名前をユーザー・フレンドリな形式に変換します。

構文

```
char *ldap_get_dn( LDAP *ld, LDAPMessage *entry );
char **ldap_explode_dn( const char *dn, int notypes );
char **ldap_explode_rdn( const char *rdn, int notypes );
char *ldap_dn2ufn( const char *dn );
```

パラメータ

表 3-23 エントリ名を取得、分割および変換するためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
entry	名前を取得する対象となるエントリです。ldap_first_entry() または ldap_next_entry() の戻り値と同一です。
dn	ldap_get_dn() で戻された識別名など、分割する識別名です。
rdn	ldap_explode_dn() によって配列の構成要素に戻された相対識別名など、分割する相対識別名です。
notypes	ブール値パラメータです。0（ゼロ）以外の場合は、識別名または相対識別名の構成要素から型情報が削除されることを示します（つまり、cn=Babs は Babs になります）。

使用方法

ldap_get_dn() は、識別名の解析でエラーが発生すると NULL を戻します。この場合は、エラーを示すためにセッション・ハンドル ld のエラー・パラメータが設定されます。この関数は、新規に割り当てられた領域へのポインタを戻します。この領域が不要になった場合は、コール元で ldap_memfree() をコールして解放する必要があります。

ldap_explode_dn() は、指定された識別名の相対識別名部分が含まれた、NULL で終了する char * 配列を戻します。型を戻すかどうかは notypes パラメータで指定します。構成要素は、識別名内にある順序で戻されます。戻された配列が不要になった場合は、ldap_value_free() をコールして解放する必要があります。

ldap_explode_rdn() は、指定された相対識別名の構成要素が含まれた、NULL で終了する char * 配列を戻します。型を戻すかどうかは notypes パラメータで指定します。構成要素は、相対識別名内にある順序で戻されます。戻された配列が不要になった場合は、ldap_value_free() をコールして解放する必要があります。

ldap_dn2ufn() は、識別名をユーザーにわかりやすい形式に変換します。戻されたユーザーにわかりやすい名前（UFN）は、新規に割り当てられた領域です。この領域が不要になった場合は、ldap_memfree() をコールして解放する必要があります。

ldap_get_entry_controls

ldap_get_entry_controls() は、LDAP コントロールをエントリから抽出します。

構文

```
int ldap_get_entry_controls
(
    LDAP          *ld,
    LDAPMessage   *entry,
    LDAPControl    ***serverctrlsp
);
```

パラメータ

表 3-24 LDAP コントロールをエントリから抽出するためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
entry	コントロールを抽出する元のエントリです。ldap_first_entry() または ldap_next_entry() の戻り値と同一です。
serverctrlsp	この結果パラメータには、エントリからコピーされたコントロールの割当て済み配列が入力されます。このコントロール配列は、ldap_controls_free() をコールして解放する必要があります。serverctrlsp が NULL の場合、コントロールは戻されません。

使用方法

ldap_get_entry_controls() は、リファレンスが正常に解析されたかどうかを示す LDAP エラー・コードを返します（正常終了の場合は LDAP_SUCCESS）。

ldap_parse_reference

ldap_parse_reference() は、リファレンスおよびコントロールを SearchResultReference メッセージから抽出します。

構文

```
int ldap_parse_reference
(
    LDAP          *ld,
    LDAPMessage   *ref,
    char          ***referralsp,
    LDAPControl    ***serverctrlsp,
    int           freeit
);
```

パラメータ

表 3-25 リファレンスとコントロールを SearchResultReference メッセージから抽出するためのパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
ref	解析するリファレンスです。ldap_result()、ldap_first_reference() または ldap_next_reference() の戻り値と同一です。
referralsp	この結果パラメータには、文字列の割当て済み配列が入力されます。配列の要素は、ref に格納されている参照（通常は LDAP URL）です。この配列が不要になった場合は、ldap_value_free() をコールして解放する必要があります。referralsp が NULL の場合、リファレンス URL は戻されません。
serverctrlsp	この結果パラメータには、ref からコピーされたコントロールの割当て済み配列が入力されます。このコントロール配列は、ldap_controls_free() をコールして解放する必要があります。serverctrlsp が NULL の場合、コントロールは戻されません。
freeit	ref パラメータを解放するかどうかを判断するブール値です。0（ゼロ）以外の値を渡すと、このルーチンは要求された情報を抽出した後に ref パラメータを解放します。このパラメータは便宜的に用意されたものです。結果は、後で ldap_msgfree() を使用して解放することもできます。

使用方法

ldap_parse_reference() は、リファレンスが正常に解析されたかどうかを示す LDAP エラー・コードを戻します（正常終了の場合は LDAP_SUCCESS）。

C API の使用例

SSL モードおよび非 SSL モードでの API の使用例を次に示します。詳細な例は、RFC 1823 に記載されています。また、LDAP 検索を実行するコマンドライン・ツールのためのサンプル・コードも、それぞれのモードでの API の使用方法を示しています。

この項では、次の項目について説明します。

- [SSL モードでの C API の使用方法](#)
- [非 SSL モードでの C API の使用方法](#)

SSL モードでの C API の使用方法

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <gsle.h>
#include <gslc.h>
#include <gsld.h>
#include "gslcc.h"

main()
{
    LDAP          *ld;
    int            ret = 0;
    ...
    /* open a connection */
    if ( (ld = ldap_open( "MyHost", 636 )) == NULL )
        exit( 1 );

    /* SSL initialization */
    ret = ldap_init_SSL(&ld->ld_sb, "file:/sslwallet", "welcome",
                       GSLC_SSL_ONETIME_AUTH );

    if(ret != 0)
    {
        printf(" %s %n", ldap_err2string(ret));
        exit(1);
    }

    /* authenticate as nobody */
    if ( ldap_bind_s( ld, NULL, NULL ) != LDAP_SUCCESS ) {
        ldap_perror( ld, "ldap_bind_s" );
        exit( 1 );
    }
}
```

```
....  
....  
}
```

ldap_init_SSL をコールしているなので、この例でのクライアントからサーバーへの通信は、SSL を使用することによって保護されています。

非 SSL モードでの C API の使用方法

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
#include <netdb.h>  
#include <gsle.h>  
#include <gslc.h>  
#include <gsld.h>  
#include "gslcc.h"  
  
main()  
{  
    LDAP          *ld;  
    int           ret = 0;  
    ....  
  
    /* open a connection */  
    if ( (ld = ldap_open( "MyHost", LDAP_PORT )) == NULL )  
        exit( 1 );  
  
    /* authenticate as nobody */  
    if ( ldap_bind_s( ld, NULL, NULL ) != LDAP_SUCCESS ) {  
        ldap_perror( ld, "ldap_bind_s" );  
        exit( 1 );  
    }  
    ....  
    ....  
}
```

この例では、ldap_init_SSL をコールしていないので、クライアントからサーバーへの通信は保護されていません。

C API を使用したアプリケーションの作成

この項では、次の項目について説明します。

- 必要なヘッダー・ファイルとライブラリ
- サンプル検索ツールの作成

必要なヘッダー・ファイルとライブラリ

C API を使用してアプリケーションを作成するには、次のものが必要となります。

- ヘッダー・ファイルは、`$ORACLE_HOME/ldap/public/ldap.h` にあります。
- ライブラリは、`$ORACLE_HOME/lib/libldapclnt8.a` にあります。

サンプル検索ツールの作成

Oracle Internet Directory SDK リリース 9.0.2 は、C API を使用したアプリケーションの作成方法を説明するために、`samplesearch` というサンプル・コマンドライン・ツールを提供しています。`samplesearch` を使用すると、SSL モードおよび非 SSL モードで LDAP 検索を実行できます。

Source ファイル (`samplesearch.c`) と Make ファイル (`demo_ldap.mk`) はディレクトリ `$ORACLE_HOME/ldap/demo` にあります。

サンプル検索ツールを作成するには、次のコマンドを入力します。

```
make -f demo_ldap.mk build EXE=samplesearch OBJS=samplesearch.o
```

注意： この Make ファイルは、C API を使用して他のクライアント・アプリケーションを作成するときにも使用できます。`samplesearch` を作成するバイナリ・ファイルの名前に、`samplesearch.o` をオブジェクト・ファイルに置き換えてください。

`samplesearch` のサンプル・コードは次のとおりです。

```
/*
NAME
    s0gsldsearch.c - <one-line expansion of the name>
DESCRIPTION
    <short description of component this file declares/defines>
PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line descriptions>
PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line descriptions>
```

```
RETURNS
    <function return values, for .c file with single function>
NOTES
    <other useful comments, qualifications, etc.>
*/
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include "ldap.h"

#define DEFSEP="
#define LDAPSEARCH_BINDDN      NULL
#define LDAPSEARCH_BASE        DEFAULT_BASE
#define DEFAULT_BASE           "o=oracle, c=US"

#ifdef LDAP_DEBUG
extern int ldap_debug, lber_debug;
#endif /* LDAP_DEBUG */

usage( s )
char*s;
{
    fprintf( stderr, "usage: %s [options] filter [attributes...]¥nwhere:¥n", s );
    fprintf( stderr, "    filter¥tRFC-1558 compliant LDAP search filter¥n" );
    fprintf( stderr, "    attributes¥twhitespace-separated list of attributes to
retrieve¥n" );
    fprintf( stderr, "¥t¥t(if no attribute list is given, all are retrieved)¥n" );
    fprintf( stderr, "options:¥n" );
    fprintf( stderr, "    -n¥t¥tshow what would be done but don't actually search¥n"
);
    fprintf( stderr, "    -v¥t¥ttrun in verbose mode (diagnostics to standard
output)¥n" );
    fprintf( stderr, "    -t¥t¥twrite values to files in /tmp¥n" );
    fprintf( stderr, "    -u¥t¥tinclude User Friendly entry names in the output¥n"
);
    fprintf( stderr, "    -A¥t¥tretrieve attribute names only (no values)¥n" );
    fprintf( stderr, "    -B¥t¥tdo not suppress printing of non-ASCII values¥n" );
    fprintf( stderr, "    -L¥t¥tprint entries in LDIF format (-B is implied)¥n" );
#ifdef LDAP_REFERRALS
    fprintf( stderr, "    -R¥t¥tdo not automatically follow referrals¥n" );
#endif /* LDAP_REFERRALS */
    fprintf( stderr, "    -d level¥tset LDAP debugging level to `level'¥n" );
    fprintf( stderr, "    -F sep¥tprint `sep' instead of `=' between attribute names
and values¥n" );
```

```

        fprintf( stderr, "    -S attr¥tsort the results by attribute `attr'¥n" );
        fprintf( stderr, "    -f file¥tperform sequence of searches listed in `file'¥n"
);
        fprintf( stderr, "    -b basedn¥tbase dn for search¥n" );
        fprintf( stderr, "    -s scope¥tone of base, one, or sub (search scope)¥n" );
        fprintf( stderr, "    -a deref¥tone of never, always, search, or find (alias
dereferencing)¥n" );
        fprintf( stderr, "    -l time lim¥ttime limit (in seconds) for search¥n" );
        fprintf( stderr, "    -z size lim¥tsize limit (in entries) for search¥n" );
        fprintf( stderr, "    -D binddn¥tbind dn¥n" );
        fprintf( stderr, "    -w passwd¥tbind passwd (for simple authentication)¥n" );
#ifdef KERBEROS
        fprintf( stderr, "    -k¥t¥tuse Kerberos instead of Simple Password
authentication¥n" );
#endif
        fprintf( stderr, "    -h host¥tldap server¥n" );
        fprintf( stderr, "    -p port¥tport on ldap server¥n" );
        fprintf( stderr, "    -W Wallet¥tWallet location¥n" );
        fprintf( stderr, "    -P Wpasswd¥tWallet Password¥n" );
        fprintf( stderr, "    -U SSLAuth¥tSSL Authentication Mode¥n" );
        return;
    }

static char*binddn = LDAPSEARCH_BINDDN;
static char*passwd = NULL;
static char*base = LDAPSEARCH_BASE;
static char*ldaphost = NULL;
static intldapport = LDAP_PORT;
static char*sep = DEFSEP;
static char*sortattr = NULL;
static intskipsortattr = 0;
static intverbose, not, includeufln, allow_binary, vals2tmp, ldif;
/* TEMP */

main( argc, argv )
intargc;
char**argv;
{
    char*infile, *filtpattern, **attrs, line[ BUFSIZ ];
    FILE*fp;
    intrc, i, first, scope, kerberos, deref, attrsonly;
    intldap_options, timelimit, sizelimit, authmethod;
    LDAP*ld;
    extern char*optarg;
    extern intoptind;
    charlocalHostName[MAXHOSTNAMELEN + 1];
    char *sslwrl = NULL;

```

```
char*sslpaswd = NULL;
int sslauth=0,err=0;

infile = NULL;
deref = verbose = allow_binary = not = kerberos = vals2tmp =
attrstronly = ldif = 0;
#ifdef LDAP_REFERRALS
ldap_options = LDAP_OPT_REFERRALS;
#else /* LDAP_REFERRALS */
ldap_options = 0;
#endif /* LDAP_REFERRALS */
sizelimit = timelimit = 0;
scope = LDAP_SCOPE_SUBTREE;

while ( ( i = getopt( argc, argv,
#ifdef KERBEROS
"KknvtrABLD:s:f:h:b:d:p:F:a:w:l:z:S:"
#else
"nuvtRABLD:s:f:h:b:d:p:F:a:w:l:z:S:W:P:U:"
#endif
)) != EOF ) {
switch( i ) {
case 'n':/* do Not do any searches */
++not;
break;
case 'v':/* verbose mode */
++verbose;
break;
case 'd':
#ifdef LDAP_DEBUG
ldap_debug = lber_debug = atoi( optarg );/* */
#else /* LDAP_DEBUG */
fprintf( stderr, "compile with -DLDPAP_DEBUG for debugging\n" );
#endif /* LDAP_DEBUG */
break;
#ifdef KERBEROS
case 'k':/* use kerberos bind */
kerberos = 2;
break;
case 'K':/* use kerberos bind, 1st part only */
kerberos = 1;
break;
#endif
case 'u':/* include UFN */
++includeufn;
break;
case 't':/* write attribute values to /tmp files */
```



```

        ++vals2tmp;
        break;
case 'R':/* don't automatically chase referrals */
#ifdef LDAP_REFERRALS
        ldap_options &= ~LDAP_OPT_REFERRALS;
#else /* LDAP_REFERRALS */
        fprintf( stderr,
            "compile with -DLdap_REFERRALS for referral support\n" );
#endif /* LDAP_REFERRALS */
        break;
case 'A':/* retrieve attribute names only -- no values */
        ++attrsonly;
        break;
case 'L':/* print entries in LDIF format */
        ++ldif;
        /* fall through -- always allow binary when outputting LDIF */
case 'B':/* allow binary values to be printed */
        ++allow_binary;
        break;
case 's':/* search scope */
        if ( strncasecmp( optarg, "base", 4 ) == 0 ) {
            scope = LDAP_SCOPE_BASE;
        } else if ( strncasecmp( optarg, "one", 3 ) == 0 ) {
            scope = LDAP_SCOPE_ONELEVEL;
        } else if ( strncasecmp( optarg, "sub", 3 ) == 0 ) {
            scope = LDAP_SCOPE_SUBTREE;
        } else {
            fprintf( stderr, "scope should be base, one, or sub\n" );
            usage( argv[ 0 ] );
            exit(1);
        }
        break;

case 'a':/* set alias deref option */
        if ( strncasecmp( optarg, "never", 5 ) == 0 ) {
            deref = LDAP_DEREF_NEVER;
        } else if ( strncasecmp( optarg, "search", 5 ) == 0 ) {
            deref = LDAP_DEREF_SEARCHING;
        } else if ( strncasecmp( optarg, "find", 4 ) == 0 ) {
            deref = LDAP_DEREF_FINDING;
        } else if ( strncasecmp( optarg, "always", 6 ) == 0 ) {
            deref = LDAP_DEREF_ALWAYS;
        } else {
            fprintf( stderr, "alias deref should be never, search, find, or always\n" );
            usage( argv[ 0 ] );
            exit(1);
        }

```

```
        break;

    case 'F':/* field separator */
        sep = (char *)strdup( optarg );
        break;
    case 'f':/* input file */
        infile = (char *)strdup( optarg );
        break;
    case 'h':/* ldap host */
        ldaphost = (char *)strdup( optarg );
        break;
    case 'b':/* searchbase */
        base = (char *)strdup( optarg );
        break;
    case 'D':/* bind DN */
        binddn = (char *)strdup( optarg );
        break;
    case 'p':/* ldap port */
        ldapport = atoi( optarg );
        break;
    case 'w':/* bind password */
        passwd = (char *)strdup( optarg );
        break;
    case 'l':/* time limit */
        timelimit = atoi( optarg );
        break;
    case 'z':/* size limit */
        sizelimit = atoi( optarg );
        break;
    case 'S':/* sort attribute */
        sortattr = (char *)strdup( optarg );
        break;
    case 'W':/* Wallet URL */
        sslwrl = (char *)strdup( optarg );
        break;
    case 'P':/* Wallet password */
        sslpasswd = (char *)strdup( optarg );
        break;
    case 'U':/* SSL Authentication Mode */
        sslauth = atoi( optarg );
        break;
    default:
        usage( argv[0] );
        exit(1);
        break;
}
}
```

```

        if ( argc - optind < 1 ) {
usage( argv[ 0 ] );
        exit(1);
    }
    filtpattern = (char *)strdup( argv[ optind ] );
    if ( argv[ optind + 1 ] == NULL ) {
attrs = NULL;
    } else if ( sortattr == NULL || *sortattr == '¥0' ) {
        attrs = &argv[ optind + 1 ];
    } else {
for ( i = optind + 1; i < argc; i++ ) {
    if ( strcasecmp( argv[ i ], sortattr ) == 0 ) {
break;
    }
}
    if ( i == argc ) {
skipsortattr = 1;
argv[ optind ] = sortattr;
    } else {
optind++;
    }
        attrs = &argv[ optind ];
    }

    if ( infile != NULL ) {
if ( infile[0] == '-' && infile[1] == '¥0' ) {
    fp = stdin;
} else if ( ( fp = fopen( infile, "r" ) ) == NULL ) {
    perror( infile );
    exit( 1 );
}
    }

    if ( ldaphost == NULL ) {
        if ( gethostname( localHostName, MAXHOSTNAMELEN ) != 0 ) {
            perror( "gethostname" );
            exit(1);
        }
        ldaphost = localHostName;
    }

    if ( verbose ) {
printf( "ldap_open( %s, %d )¥n", ldaphost, ldapport );
    }

    if ( ( ld = ldap_open( ldaphost, ldapport ) ) == NULL ) {

```

```
perror( ldaphost );
exit( 1 );
}

if (sslauth > 1)
{
    if (!sslwrl || !sslpasswd)
    {
        printf ("Null Wallet or password given\n");
        exit (0);
    }
}
if (sslauth > 0)
{
    if (sslauth == 1)
        sslauth = GSLC_SSL_NO_AUTH;
    else if (sslauth == 2)
        sslauth = GSLC_SSL_ONEWAY_AUTH;
    else if (sslauth == 3)
        sslauth = GSLC_SSL_TWOWAY_AUTH;
    else
    {
        printf(" Wrong SSL Authentication Mode Value\n");
        exit(0);
    }

    err = ldap_init_SSL(&ld->ld_sb,sslwrl,sslpasswd,sslauth);
    if(err != 0)
    {
        printf(" %s\n", ldap_err2string(err));
        exit(0);
    }
}

ld->ld_deref = deref;
ld->ld_timelimit = timelimit;
ld->ld_sizelimit = sizelimit;
ld->ld_options = ldap_options;

if ( !kerberos ) {
authmethod = LDAP_AUTH_SIMPLE;
} else if ( kerberos == 1 ) {
authmethod = LDAP_AUTH_KRBV41;
} else {
authmethod = LDAP_AUTH_KRBV4;
}
if ( ldap_bind_s( ld, binddn, passwd, authmethod ) != LDAP_SUCCESS ) {
```

```

ldap_perror( ld, "ldap_bind" );
exit( 1 );
}

if ( verbose ) {
printf( "filter pattern: %s\nreturning: ", filtpattern );
if ( attrs == NULL ) {
    printf( "ALL" );
} else {
    for ( i = 0; attrs[ i ] != NULL; ++i ) {
printf( "%s ", attrs[ i ] );
    }
}
putchar( '\n' );
}

if ( infile == NULL ) {
rc = dosearch( ld, base, scope, attrs, attrsonly, filtpattern, "" );
} else {
rc = 0;
first = 1;
while ( rc == 0 && fgets( line, sizeof( line ), fp ) != NULL ) {
    line[ strlen( line ) - 1 ] = '\0';
    if ( !first ) {
putchar( '\n' );
    } else {
first = 0;
    }
    rc = dosearch( ld, base, scope, attrs, attrsonly, filtpattern,
line );
}
if ( fp != stdin ) {
fclose( fp );
}
}

ldap_unbind( ld );
exit( rc );
}

dosearch( ld, base, scope, attrs, attrsonly, filtpatt, value )
LDAP*ld;
char*base;
intscope;
char**attrs;
intattrsonly;
char*filtpatt;

```

```
    char*value;
{
    charfilter[ BUFSIZ ], **val;
    intrc, first, matches;
    LDAPMessage*res, *e;

    sprintf( filter, filtpatt, value );

    if ( verbose ) {
printf( "filter is: (%s)%n", filter );
    }

    if ( not ) {
return( LDAP_SUCCESS );
    }

    if ( ldap_search( ld, base, scope, filter, attrs, attrsonly ) == -1 ) {
ldap_perror( ld, "ldap_search" );
return( ld->ld_errno );
    }

    matches = 0;
    first = 1;
    while ( (rc = ldap_result( ld, LDAP_RES_ANY, sortattr ? 1 : 0, NULL, &res ))
== LDAP_RES_SEARCH_ENTRY ) {
matches++;
e = ldap_first_entry( ld, res );
if ( !first ) {
    putchar( '\n' );
} else {
    first = 0;
}
print_entry( ld, e, attrsonly );
ldap_msgfree( res );
    }
    if ( rc == -1 ) {
ldap_perror( ld, "ldap_result" );
return( rc );
    }
    if ( ( rc = ldap_result2error( ld, res, 0 )) != LDAP_SUCCESS ) {
        ldap_perror( ld, "ldap_search" );
    }
    if ( sortattr != NULL ) {
extern intstrcasecmp();

(void) ldap_sort_entries( ld, &res,
    ( *sortattr == '¥0' ) ? NULL : sortattr, strcasecmp );
    }
```

```

        matches = 0;
        first = 1;
        for ( e = ldap_first_entry( ld, res ); e != NULLMSG;
              e = ldap_next_entry( ld, e ) ) {
matches++;
        if ( !first ) {
            putchar( '\n' );
        } else {
            first = 0;
        }
        print_entry( ld, e, attrsonly );
    }

    if ( verbose ) {
        printf( "%d matches\n", matches );
    }

    ldap_msgfree( res );
    return( rc );
}

```

```

print_entry( ld, entry, attrsonly )
LDAP*ld;
LDAPMessage*entry;
intattrsonly;
{
    char*a, *dn, *ufn, tmpfname[ 64 ];
    inti, j, notascii;
    BerElement*ber;
    struct berval**bvals;
    FILE*tmpfp;
    extern char*mktemp();

    dn = ldap_get_dn( ld, entry );
    if ( ldif ) {
write_ldif_value( "dn", dn, strlen( dn ));
    } else {
        printf( "%s\n", dn );
    }
    if ( includeufn ) {
ufn = ldap_dn2ufn( dn );
        if ( ldif ) {
            write_ldif_value( "ufn", ufn, strlen( ufn ));
        } else {
            printf( "%s\n", ufn );
        }
    }
}

```

```
    }
    free( ufn );
    }
    free( dn );

    for ( a = ldap_first_attribute( ld, entry, &ber ); a != NULL;
          a = ldap_next_attribute( ld, entry, ber ) ) {
    if ( skipsortattr && strcasecmp( a, sortattr ) == 0 ) {
        continue;
    }
    if ( attrsonly ) {
        if ( ldif ) {
            write_ldif_value( a, "", 0 );
        } else {
            printf( "%s¥n", a );
        }
    } else if ( ( bvals = ldap_get_values_len( ld, entry, a ) ) != NULL ) {
        for ( i = 0; bvals[i] != NULL; i++ ) {
            if ( vals2tmp ) {
                sprintf( tmpfname, "/tmp/ldapsearch-%s-XXXXXX", a );
                tmpfp = NULL;

                if ( mktemp( tmpfname ) == NULL ) {
                    perror( tmpfname );
                } else if ( ( tmpfp = fopen( tmpfname, "w" ) ) == NULL ) {
                    perror( tmpfname );
                } else if ( fwrite( bvals[ i ]->bv_val,
                                     bvals[ i ]->bv_len, 1, tmpfp ) == 0 ) {
                    perror( tmpfname );
                } else if ( ldif ) {
                    write_ldif_value( a, tmpfname, strlen( tmpfname ) );
                } else {
                    printf( "%s%s%s¥n", a, sep, tmpfname );
                }

                if ( tmpfp != NULL ) {
                    fclose( tmpfp );
                }
            } else {
                notascii = 0;
                if ( !allow_binary ) {
                    for ( j = 0; j < bvals[ i ]->bv_len; ++j ) {
                        if ( !isascii( bvals[ i ]->bv_val[ j ] ) ) {
                            notascii = 1;
                            break;
                        }
                    }
                }
            }
        }
    }
}
```



```
    }

    if ( ldif ) {
write_ldif_value( a, bvals[ i ]->bv_val,
bvals[ i ]->bv_len );
    } else
    {
printf( "%s%s%s\n", a, sep,
notascii ? "NOT ASCII" : (char *)bvals[ i ]->bv_val );
    }
}
}
gsledePBerBvecfree( bvals );
}
}
}

int
write_ldif_value( char *type, char *value, unsigned long vallen )
{
    char *ldif;

    if (( ldif = gsldlDLdifTypeAndValue( type, value, (int)vallen )) == NULL )
    {
return( -1 );
    }

    fputs( ldif, stdout );
    free( ldif );

    return( 0 );
}
```

依存性と制限事項

この API は、すべてのリリースの Oracle Internet Directory で機能します。Oracle 環境または最低でも NLS などの主要ライブラリが必要です。

SSL で別の認証モードを使用するには、ディレクトリ・サーバーの構成設定をモードに応じて変更する必要があります。

関連項目： それぞれの SSL 認証モードに応じた Oracle Internet Directory サーバーの設定方法の詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

SSL モードで C API を使用する場合は、Wallet を作成するために Oracle Wallet Manager が必要となります。

TCP/IP ソケット・ライブラリが必要です。

次の Oracle ライブラリが必要です。

- Oracle SSL 関連ライブラリ
- Oracle システム・ライブラリ

サンプル・コマンドライン・ツールのリリースの中には、サンプル・ライブラリが含まれています。それらのライブラリはユーザー自身のライブラリに置き換える必要があります。

この製品は、LDAP SDK の仕様（RFC 1823）に記述されている認証方式のみをサポートします。

DBMS_LDAP PL/SQL パッケージ

この章では、PL/SQL プログラマによる Lightweight Directory Access Protocol (LDAP) サーバーからデータへのアクセスを可能にする DBMS_LDAP パッケージについて説明します。また、DBMS_LDAP の使用例も紹介します。この章では、次の項目について説明します。

- [DBMS_LDAP パッケージの概要](#)
- [DBMS_LDAP を使用したアプリケーションの作成](#)
- [依存性と制限事項](#)
- [DBMS_LDAP サンプル・プログラム](#)
- [DBMS_LDAP リファレンス](#)

DBMS_LDAP パッケージの概要

DBMS_LDAP パッケージの PL/SQL Application Program Interface (PL/SQL API) は、[第 3 章「Oracle Internet Directory の C API」](#) で説明されている C Application Program Interface (C API) に基づいています。

Oracle Internet Directory API リリース 9.0.2 は、次のモードで使用できます。

- SSL モード Secure Sockets Layer (SSL) を使用してすべての通信を保護する
- 非 SSL モードクライアントからサーバーへの通信を保護しない

API は、TCP/IP を使用して LDAP サーバーに接続します。接続するとき、デフォルトでは暗号化されていないチャンネルを使用します。SSL モードを使用するには、Oracle SSL コール・インタフェースを使用する必要があります。API を使用する際に、SSL コールの有無によってどちらのモードを使用するかを決定します。SSL モードと非 SSL モードは簡単に切り替えることができます。

DBMS_LDAP を使用したアプリケーションの作成

PL/SQL LDAP API を使用するには、まずデータベースにロードする必要があります。`$ORACLE_HOME/rdbms/admin` ディレクトリにある `catldap.sql` というスクリプトを使用してロードします。そのためには、SQL*Plus コマンドライン・ツールを使用して、SYSDBA で接続する必要があります。

DBMS_LDAP パッケージのロードに使用できるコマンド・シーケンスのサンプルを次に示します。

```
SQL> CONNECT / AS SYSDBA
SQL> @?/rdbms/admin/catldap.sql
```

依存性と制限事項

このリリースの PL/SQL LDAP API には、次の制限事項があります。

- API から取得した LDAP セッション・ハンドルは、データベース・セッションの継続時間内のみ有効です。LDAP セッション・ハンドルを表に書き込んだり、他のデータベース・セッションで再利用することはできません。
- このリリースでは、同期型の LDAP API ファンクションのみサポートされています。
- PL/SQL LDAP API で作業するには、データベースに接続する必要があります。クライアント側の PL/SQL エンジン (Oracle Forms など) ではデータベースへの接続が有効でないかぎり、この API を使用できません。

DBMS_LDAP サンプル・プログラム

今回の Oracle Internet Directory には、リレーショナル環境で DBMS_LDAP を使用する例を示したサンプル・プログラムが含まれています。このサンプルには、次の DBMS_LDAP API の使用例が示されています。

- データベース・トリガーを使用して、リレーショナル表への変更を LDAP に同期化する
- 特定の検索条件に一致する LDAP エントリを取得する

サンプルは、`$ORACLE_HOME/ldap/demo/plsql` ディレクトリに格納されています。

関連項目： サンプルの詳細は、付録 A「使用例」を参照してください。

DBMS_LDAP リファレンス

DBMS_LDAP には、PL/SQL プログラマが LDAP サーバーからデータにアクセスする際に使用できるファンクションとプロシージャが含まれています。この項では、すべての API ファンクションの詳細を説明します。前の各項を読んでから、この項を使用してください。

この項では、次の項目について説明します。

- [サブプログラムの概要](#)
- [例外の概要](#)
- [データ型の概要](#)
- [サブプログラム](#)

サブプログラムの概要

表 4-1 DBMS_LDAP API のサブプログラム

ファンクションまたは プロシージャ	説明
init ファンクション	init() ファンクションを使用すると、LDAP サーバーとのセッションが初期化されます。これにより、実際に LDAP サーバーとの接続が確立されます。
simple_bind_s ファンクション	simple_bind_s ファンクションを使用すると、ユーザー名およびパスワードに基づく、ディレクトリ・サーバーへの簡易認証を実行できます。
bind_s ファンクション	bind_s ファンクションを使用すると、ディレクトリ・サーバーへの高度な認証を実行できます。
unbind_s ファンクション	unbind_s ファンクションは、アクティブな LDAP セッションのクローズに使用します。

表 4-1 DBMS_LDAP API のサブプログラム (続き)

ファンクションまたは プロシージャ	説明
<code>compare_s</code> ファンクション	<code>compare_s</code> ファンクションを使用すると、特定のエントリの特定の属性が特定の値を持っているかどうかをテストできます。
<code>search_s</code> ファンクション	<code>search_s</code> ファンクションを使用すると、LDAP サーバー内で同期検索が実行されます。これを実行すると、サーバーからすべての検索結果が送信されるか、検索要求がサーバーによってタイムアウトになるまで、PL/SQL 環境に制御が戻されません。
<code>search_st</code> ファンクション	<code>search_st</code> ファンクションを使用すると、LDAP サーバー内で、クライアント側のタイムアウトを使用して同期検索が実行されます。これを実行すると、サーバーからすべての検索結果が送信されるか、検索要求がクライアントまたはサーバーによってタイムアウトになるまで、PL/SQL 環境に制御が戻されません。
<code>first_entry</code> ファンクション	<code>first_entry</code> ファンクションを使用すると、 <code>search_s</code> または <code>search_st</code> で戻された結果セット内の最初のエントリを取り出せます。
<code>next_entry</code> ファンクション	<code>next_entry()</code> ファンクションを使用すると、検索操作による結果セット内の次のエントリを取り出すことができます。
<code>count_entries</code> ファンクション	このファンクションは、結果セット内のエントリ数のカウントに使用します。また、 <code>first_entry()</code> ファンクションおよび <code>next_entry()</code> ファンクションと組み合わせて使用すると、結果セットの全探索時に残っているエントリの数をカウントすることもできます。
<code>first_attribute</code> ファンクション	<code>first_attribute()</code> ファンクションを使用すると、結果セットの中から、指定したエントリの最初の属性がフェッチされます。
<code>next_attribute</code> ファンクション	<code>next_attribute()</code> ファンクションを使用すると、結果セットの中から、指定したエントリの次の属性がフェッチされます。
<code>get_dn</code> ファンクション	<code>get_dn()</code> ファンクションを使用すると、結果セットの中から、指定したエントリの X.500 識別名が取り出されます。
<code>get_values</code> ファンクション	<code>get_values()</code> ファンクションを使用すると、特定のエントリの特定の属性に関連する値をすべて取り出せます。
<code>get_values_len</code> ファンクション	<code>get_values_len()</code> ファンクションを使用すると、バイナリ構文を持つ属性の値を取り出せます。

表 4-1 DBMS_LDAP API のサブプログラム (続き)

ファンクションまたは プロシージャ	説明
<code>delete_s</code> ファンクション	<code>delete_s</code> ファンクションを使用すると、LDAP ディレクトリ情報ツリー内のリーフ・エントリを削除できます。
<code>modrdn2_s</code> ファンクション	<code>modrdn2_s()</code> ファンクションを使用すると、エントリの相対識別名を変更できます。
<code>err2string</code> ファンクション	<code>err2string()</code> ファンクションを使用すると、LDAP エラー・コードを、API の動作環境で使用されている各国語の文字列に変換できます。
<code>create_mod_array</code> ファンクション	<code>create_mod_array()</code> ファンクションを使用すると、 <code>modify_s()</code> ファンクションを使用してエントリに適用される変更配列に、メモリーが割り当てられます。
<code>populate_mod_array</code> プロシージャ (文字列バージョン)	追加操作または変更操作に、1 組の属性情報を代入します。DBMS_LDAP. <code>create_mod_array()</code> をコールした後に、このプロシージャをコールする必要があります。
<code>populate_mod_array</code> プロシージャ (バイナリ・バージョン)	追加操作または変更操作に、1 組の属性情報を代入します。DBMS_LDAP. <code>create_mod_array()</code> をコールした後に、このプロシージャをコールする必要があります。
<code>modify_s</code> ファンクション	既存の LDAP ディレクトリ・エントリの同期変更を実行します。 <code>add_s</code> をコールする前に、DBMS_LDAP. <code>create_mod_array()</code> と DBMS_LDAP. <code>populate_mod_array()</code> をコールする必要があります。
<code>add_s</code> ファンクション	LDAP ディレクトリに新規エントリを同期的に追加します。 <code>add_s</code> をコールする前に、DBMS_LDAP. <code>create_mod_array()</code> と DBMS_LDAP. <code>populate_mod_array()</code> をコールする必要があります。
<code>free_mod_array</code> プロシージャ	DBMS_LDAP. <code>create_mod_array()</code> によって割り当てられたメモリーを解放します。
<code>count_values</code> ファンクション	DBMS_LDAP. <code>get_values()</code> によって戻された値の数をカウントします。
<code>count_values_len</code> ファンクション	DBMS_LDAP. <code>get_values_len()</code> によって戻された値の数をカウントします。
<code>rename_s</code> ファンクション	LDAP エントリの名前を同期的に変更します。
<code>explode_dn</code> ファンクション	識別名を個々の構成要素に分割します。
<code>open_ssl</code> ファンクション	すでに確立されている LDAP 接続を介して SSL 接続を確立します。

表 4-1 DBMS_LDAP API のサブプログラム (続き)

ファンクションまたは プロシージャ	説明
msgfree ファンクション	同期検索ファンクションによって戻されたメッセージ・ハンドルに対応付けられている結果セットを解放します。
ber_free ファンクション	BER ELEMENT へのハンドルに対応付けられたメモリーを解放します。

例外の概要

DBMS_LDAP では、次の例外が生成される場合があります。

表 4-2 DBMS_LDAP 例外の概要

例外名	Oracle エラー 番号	例外の原因
<code>general_error</code>	31202	関係する特定の PL/SQL 例外がないエラーが発生すると常呼び出されます。エラー文字列には、ユーザーが使用している各国語で問題が説明されます。
<code>init_failed</code>	31203	<code>DBMS_LDAP.init()</code> でなんらかの問題がある場合に呼び出されます。
<code>invalid_session</code>	31204	DBMS_LDAP パッケージのファンクションおよびプロシージャに無効なセッション・ハンドルが渡された場合に呼び出されます。
<code>invalid_auth_method</code>	31205	<code>DBMS_LDAP.bind_s()</code> で要求された認証方式がサポートされていない場合に呼び出されます。
<code>invalid_search_scope</code>	31206	検索の範囲が無効な場合に、すべての検索ファンクションによって呼び出されます。
<code>invalid_search_time_val</code>	31207	制限時間として渡された値が無効な場合に、時間ベースの検索ファンクションの <code>DBMS_LDAP.search_st()</code> によって呼び出されます。
<code>invalid_message</code>	31208	検索操作によるエントリの取得を結果セット全体にわたって反復するファンクションに、無効なメッセージ・ハンドルが指定された場合に呼び出されます。
<code>count_entry_error</code>	31209	<code>DBMS_LDAP.count_entries</code> で指定した結果セット内のエントリをカウントできない場合に呼び出されます。

表 4-2 DBMS_LDAP 例外の概要 (続き)

例外名	Oracle エラー 番号	例外の原因
get_dn_error	31210	DBMS_LDAP.get_dn によって取り出されるエントリの識別名が NULL である場合に呼び出されます。
invalid_entry_dn	31211	エントリを変更、追加または改名するファンクションに無効なエントリの識別名を指定した場合に呼び出されます。
invalid_mod_array	31212	変更配列を引数として取るファンクションに、無効な変更配列を指定した場合に呼び出されます。
invalid_mod_option	31213	DBMS_LDAP.populate_mod_array で指定した変更オプションが、MOD_ADD、MOD_DELETE または MOD_REPLACE ではなかった場合に呼び出されます。
invalid_mod_type	31214	DBMS_LDAP.populate_mod_array によって変更される属性の型が NULL である場合に呼び出されます。
invalid_mod_value	31215	DBMS_LDAP.populate_mod_array で指定した属性を NULL 値で更新しようとした場合に呼び出されます。
invalid_rdn	31216	相対識別名の値が NULL である場合に、有効な相対識別名を想定するファンクションおよびプロシージャによって呼び出されます。
invalid_newparent	31217	DBMS_LDAP.rename_s によって改名されるエントリの新しい親が NULL である場合に呼び出されます。
invalid_deleteoldrdn	31218	DBMS_LDAP.rename_s の deleteoldrdn パラメータが無効な場合に呼び出されます。
invalid_notypes	31219	DBMS_LDAP.explode_dn の notypes パラメータが無効な場合に呼び出されます。
invalid_ssl_wallet_loc	31220	Wallet の場所が NULL であるが SSL 認証モードが有効な Wallet を必要とする場合に、DBMS_LDAP.open_ssl によって呼び出されます。
invalid_ssl_wallet_password	31221	DBMS_LDAP.open_ssl で指定した Wallet パスワードが NULL である場合に呼び出されます。
invalid_ssl_auth_mode	31222	SSL 認証モードが 1、2 または 3 ではない場合に DBMS_LDAP.open_ssl によって呼び出されます。

データ型の概要

DBMS_LDAP パッケージでは、次のデータ型を使用します。

表 4-3 DBMS_LDAP データ型の概要

データ型	用途
SESSION	LDAP セッションのハンドルを保持するために使用します。API のほとんどすべてのファンクションでは、作業のために、有効な LDAP セッションが必要となります。
MESSAGE	結果セットから取り出されたメッセージのハンドルを保持するために使用します。このデータ型は、エントリの属性および値を処理するすべてのファンクションで使用します。
MOD_ARRAY	modify_s() または add_s() に渡される変更配列のハンドルを保持するために使用します。
TIMEVAL	制限時間を必要とする LDAP API ファンクションに制限時間の情報を渡すために使用します。
BER_ELEMENT	受信メッセージのデコードに使用される BER 構造体のハンドルを保持するために使用します。
STRING_COLLECTION	LDAP サーバーに渡すことができる VARCHAR2 文字列のリストを保持するために使用します。
BINVAL_COLLECTION	バイナリ・データを表す RAW データのリストを保持するために使用します。
BERVAL_COLLECTION	変更配列の代入に使用される BEREAL 値のリストを保持するために使用します。

サブプログラム

init ファンクション

init() ファンクションを使用すると、LDAP サーバーとのセッションが初期化されます。これにより、実際に LDAP サーバーとの接続が確立されます。

構文

```
FUNCTION init
(
    hostname IN VARCHAR2,
    portnum  IN PLS_INTEGER
)
    RETURN SESSION;
```

パラメータ

表 4-4 INIT ファンクションのパラメータ

パラメータ	説明
hostname	スペースで区切られたホスト名のリスト、または LDAP サーバーを実行している接続先のホストの IP アドレスを示すドット表記の文字列が入ります。リスト内の各ホスト名には、ポート番号を含めることもできます。ポート番号とホスト自体はコロン (:) で区切ります。ホストへの接続はリストの順序に従って試みられ、最初にホストへの接続が成功した時点で終了します。
portnum	接続先の TCP ポート番号が入ります。ホストにポート番号が含まれている場合、このパラメータは無視されます。このパラメータを指定せず、ホスト名にもポート番号を含めていない場合は、デフォルトのポート番号 389 が指定されたものとみなされます。

戻り値

表 4-5 INIT ファンクションの戻り値

値	説明
SESSION (ファンクション戻り値)	以後の API のコールに使用できる LDAP セッション・ハンドルです。

例外

表 4-6 INIT ファンクションの例外

例外	説明
init_failed	LDAP サーバーとの通信中に問題が発生した場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

DBMS_LDAP.init() は、LDAP サーバーとのセッションを確立するために最初にコールする必要があるファンクションです。DBMS_LDAP.init() ファンクションの戻り値はセッション・ハンドルです。セッション・ハンドルとは、セッションに関連する後続のコールに渡す必要がある不透明な構造体へのポインタです。このルーチンでセッションを初期化できない場合は NULL が戻され、INIT_FAILED 例外が呼び出されます。init() をコールした後、DBMS_LDAP.bind_s または DBMS_LDAP.simple_bind_s() を使用して認証を行う必要があります。

関連項目

DBMS_LDAP.simple_bind_s()、DBMS_LDAP.bind_s()

simple_bind_s ファンクション

simple_bind_s ファンクションを使用すると、ユーザー名およびパスワードに基づく、ディレクトリ・サーバーへの簡易認証を実行できます。

構文

```
FUNCTION simple_bind_s
(
    ld      IN SESSION,
    dn      IN VARCHAR2,
    passwd  IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-7 SIMPLE_BIND_S ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
dn	ログイン時に使用するユーザー識別名です。
passwd	パスワードを含む文字列です。

戻り値

表 4-8 SIMPLE_BIND_S ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	正常終了した場合の戻り値は DBMS_LDAP.SUCCESS です。問題があった場合は、次の例外のいずれかが呼び出されます。

例外

表 4-9 SIMPLE_BIND_S ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

DBMS_LDAP.simple_bind_s() を使用すると、ディレクトリ識別名およびディレクトリ・パスワードがすでにわかっているユーザーを認証できます。DBMS_LDAP.init() のコールで有効な LDAP セッション・ハンドルを取得してから、このファンクションをコールしてください。

bind_s ファンクション

bind_s ファンクションを使用すると、ディレクトリ・サーバーへの高度な認証を実行できます。

構文

```
FUNCTION bind_s
(
    ld      IN SESSION,
    dn      IN VARCHAR2,
    cred IN VARCHAR2,
    meth IN PLS_INTEGER
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-10 BIND_S ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
dn	ログイン時に使用するユーザー識別名です。
cred	認証に使用する資格証明を含む文字列です。
meth	認証方式です。

戻り値

表 4-11 BIND_S ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	正常終了した場合の戻り値は DBMS_LDAP.SUCCESS です。問題があった場合は、次の例外が呼び出されます。

例外

表 4-12 BIND_S ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル <code>ld</code> が無効な場合に呼び出されます。
invalid_auth_method	要求した認証方式がサポートされていない場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

DBMS_LDAP.bind_s() を使用すると、ユーザーを認証できます。DBMS_LDAP.init() のコールで有効な LDAP セッション・ハンドルを取得してから、このファンクションをコールしてください。

関連項目

DBMS_LDAP.init()、DBMS_LDAP.simple_bind_s()

unbind_s ファンクション

unbind_s ファンクションは、アクティブな LDAP セッションのクローズに使用します。

構文

```
FUNCTION unbind_s
(
    ld IN SESSION
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-13 UNBIND_S ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。

戻り値

表 4-14 UNBIND_S ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	正常終了した場合の戻り値は DBMS_LDAP.SUCCESS です。それ以外の場合は、次の例外のいずれかが呼び出されます。

例外

表 4-15 UNBIND_S ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

`unbind_s()` ファンクションを使用すると、サーバーにバインド解除要求が送信され、LDAP セッションに関連するオープンな接続がすべてクローズされ、セッション・ハンドルに関連するリソースがすべて処理された後に値が戻されます。このファンクションをコールすると、セッション・ハンドル `ld` が無効になるため、これ以降に、`ld` を使用して LDAP API コールを行うことはできません。

関連項目

`DBMS_LDAP.bind_s()`、`DBMS_LDAP.simple_bind_s()`

compare_s ファンクション

compare_s ファンクションを使用すると、特定のエントリの特定の属性が特定の値を持っているかどうかをテストできます。

構文

```
FUNCTION compare_s
(
    ld      IN SESSION,
    dn      IN VARCHAR2,
    attr    IN VARCHAR2,
    value   IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-16 COMPARE_S ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
dn	比較の対象となるエントリの名前です。
attr	比較の対象となる属性です。
value	比較の対象となる文字列の属性値です。

戻り値

表 4-17 COMPARE_S ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	属性の値が指定した値と一致した場合の戻り値は COMPARE_TRUE です。 属性の値が指定した値と一致しない場合の戻り値は COMPARE_FALSE です。

例外

表 4-18 COMPARE_S ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル <code>ld</code> が無効な場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

`compare_s` ファンクションを使用すると、ディレクトリ・サーバーに格納されている特定の属性の値が、特定の値と一致するかどうかを確認できます。この操作は、比較が可能な構文定義を持つ属性についてのみ実行できます。`compare_s` ファンクションは、`init()` ファンクションで有効な LDAP セッション・ハンドルを取得し、`bind_s()` ファンクションまたは `simple_bind_s()` ファンクションを使用してこのセッション・ハンドルを認証してから、コールしてください。

関連項目

DBMS_LDAP.bind_s()

search_s ファンクション

search_s ファンクションを使用すると、LDAP サーバー内で同期検索が実行されます。これを実行すると、サーバーからすべての検索結果が送信されるか、検索要求がサーバーによってタイムアウトになるまでは、PL/SQL 環境に制御が戻されません。

構文

```
FUNCTION search_s
(
    ld          IN  SESSION,
    base        IN  VARCHAR2,
    scope       IN  PLS_INTEGER,
    filter      IN  VARCHAR2,
    attrs       IN  STRING_COLLECTION,
    attronly    IN  PLS_INTEGER,
    res         OUT MESSAGE
)
    RETURN PLS_INTEGER;
```

パラメータ

表 4-19 SEARCH_S ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
base	検索の開始点となるエントリの識別名です。
scope	SCOPE_BASE (0x00)、SCOPE_ONELEVEL (0x01) または SCOPE_SUBTREE (0x02) のいずれかで、検索範囲を指定します。
filter	検索フィルタを表す文字列です。この値を NULL にすると、すべてのエントリに一致するフィルタ (objectclass=*) を使用するように指定できます。
attrs	一致した各エントリのどの属性を戻すかを指定する文字列の集合です。このパラメータを NULL にすると、取得可能なユーザー属性がすべて取り出されます。文字列 NO_ATTRS ("1.1") を配列内の唯一の文字列として使用すると、サーバーが属性型を戻さないように指定できます。attrs 配列の中で、文字列 ALL_USER_ATTRS ("*") をいくつかの操作属性名とともに使用すると、すべてのユーザー属性に加えて、リストした操作属性を戻すように指定できます。
attronly	属性の型と値の両方を戻す場合には 0 (ゼロ)、属性の型のみを要求する場合には 0 (ゼロ) 以外を指定する必要があるブール値です。
res	コールの終了時に検索結果が入る結果パラメータです。戻される結果がない場合、*res は NULL に設定されます。

戻り値

表 4-20 SEARCH_S ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	検索操作が正常終了した場合の戻り値は DBMS_LDAP.SUCCESS です。その他の場合は例外が呼び出されます。
res (OUT パラメータ)	検索が正常終了してエントリがあった場合、このパラメータは NULL 以外の値に設定されます。この値を使用すると、結果セットからエントリを取り出すことができます。

例外

表 4-21 SEARCH_S ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_search_scope	検索範囲が、SCOPE_BASE、SCOPE_ONELEVEL または SCOPE_SUBTREE ではない場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

search_s() ファンクションを使用すると検索操作が発行されます。検索操作が発行されると、サーバーからすべての検索結果が戻されるまでは、ユーザー環境に制御が戻されません。検索によって戻されるエントリがある場合、res パラメータの中に収められます。このパラメータはコール元に対しては不透明です。エントリ、属性、値などは、後述の解析ルーチンをコールすることで抽出できます。

関連項目

DBMS_LDAP.search_st()、DBMS_LDAP.first_entry()、DBMS_LDAP.next_entry

search_st ファンクション

search_st ファンクションを使用すると、LDAP サーバー内で、クライアント側のタイムアウトを使用して同期検索が実行されます。これを実行すると、サーバーからすべての検索結果が送信されるか、検索要求がクライアントまたはサーバーによってタイムアウトになるまでは、PL/SQL 環境に制御が戻されません。

構文

```
FUNCTION search_st
(
    ld          IN  SESSION,
    base        IN  VARCHAR2,
    scope       IN  PLS_INTEGER,
    filter      IN  VARCHAR2,
    attrs       IN  STRING_COLLECTION,
    attrsonly   IN  PLS_INTEGER,
    tv          IN  TIMEVAL,
    res         OUT MESSAGE
)
    RETURN PLS_INTEGER;
```

パラメータ

表 4-22 SEARCH_ST ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
base	検索の開始点となるエントリの識別名です。
scope	SCOPE_BASE (0x00)、SCOPE_ONELEVEL (0x01) または SCOPE_SUBTREE (0x02) のいずれかで、検索範囲を指定します。
filter	検索フィルタを表す文字列です。この値を NULL にすると、すべてのエントリに一致するフィルタ (objectclass=*) を使用するよう指定できます。
attrs	一致した各エントリのどの属性を戻すかを指定する文字列の集合です。このパラメータを NULL にすると、取得可能なユーザー属性がすべて取り出されます。文字列 NO_ATTRS ("1.1") を配列内の唯一の文字列として使用すると、サーバーが属性型を戻さないように指定できます。attrs 配列の中で、文字列 ALL_USER_ATTRS ("*") をいくつかの操作属性名とともに使用すると、すべてのユーザー属性に加えて、リストした操作属性を戻すように指定できます。

表 4-22 SEARCH_ST ファンクションのパラメータ (続き)

パラメータ	説明
attrsonly	属性の型と値の両方を戻す場合には 0 (ゼロ)、属性の型のみを要求する場合には 0 (ゼロ) 以外を指定する必要があるブール値です。
tv	この検索で使用する必要があるタイムアウト値です (秒およびミリ秒単位で表します)。
res	コールの終了時に検索結果が入る結果パラメータです。戻される結果がない場合、*res は NULL に設定されます。

戻り値

表 4-23 SEARCH_ST ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	検索操作が正常終了した場合の戻り値は DBMS_LDAP.SUCCESS です。その他の場合は例外が呼び出されます。
res (OUT パラメータ)	検索が正常終了してエントリがあった場合、このパラメータは NULL 以外の値に設定されます。この値を使用すると、結果セットからエントリを取り出すことができます。

例外

表 4-24 SEARCH_ST ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_search_scope	検索範囲が、SCOPE_BASE、SCOPE_ONELEVEL または SCOPE_SUBTREE ではない場合に呼び出されます。
invalid_search_time_value	タイムアウトに指定した時間の値が無効な場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

このファンクションは DBMS_LDAP.search_s() に類似していますが、タイムアウト値を指定する必要があるという点に違いがあります。

関連項目

DBMS_LDAP.search_s()、DBMS_LDAP.first_entry()、DBMS_LDAP.next_entry

first_entry ファンクション

first_entry ファンクションを使用すると、search_s() または search_st() で戻された結果セット内の最初のエントリを取り出せます。

構文

```
FUNCTION first_entry
(
    ld IN SESSION,
    msg IN MESSAGE
)
RETURN MESSAGE;
```

パラメータ

表 4-25 FIRST_ENTRY ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
msg	検索結果です。同期検索ルーチンのいずれかをコールして取得されるものと同一です。

戻り値

表 4-26 FIRST_ENTRY の戻り値

値	説明
MESSAGE (ファンクション戻り値)	LDAP サーバーから戻されたエントリのリスト中の最初のエントリのハンドルです。エラーがあった場合は NULL に設定され、例外が呼び出されます。

例外

表 4-27 FIRST_ENTRY の例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_message	受信した msg ハンドルが無効な場合に呼び出されます。

使用方法

first_entry() ファンクションは、検索操作からの結果を取り出すために必ず最初にコールする必要があるファンクションです。

関連項目

DBMS_LDAP.next_entry(), DBMS_LDAP.search_s(), DBMS_LDAP.search_st()

next_entry ファンクション

next_entry() ファンクションを使用すると、検索操作による結果セット内の次のエントリを取り出すことができます。

構文

```
FUNCTION next_entry
(
    ld    IN SESSION,
    msg   IN MESSAGE
)
RETURN MESSAGE;
```

パラメータ

表 4-28 NEXT_ENTRY ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
msg	検索結果です。同期検索ルーチンのいずれかをコールして取得されるものと同一です。

戻り値

表 4-29 NEXT_ENTRY ファンクションの戻り値

値	説明
MESSAGE	LDAP サーバーから戻されたエントリのリスト中の次のエントリのハンドルです。エラーがあった場合は NULL に設定されて、例外が呼び出されます。

例外

表 4-30 NEXT_ENTRY ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル 1d が無効な場合に呼び出されます。
invalid_message	受信した msg ハンドルが無効な場合に呼び出されます。

使用方法

next_entry() ファンクションは、必ず first_entry() ファンクションの後にコールする必要があります。また、リスト中の次のエントリをフェッチするには、正常終了した next_entry() のコールの戻り値を、次の next_entry() のコールで msg 引数として使用する必要があります。

関連項目

DBMS_LDAP.first_entry()、DBMS_LDAP.search_s()、DBMS_LDAP.search_st()

count_entries ファンクション

このファンクションは、結果セット内のエントリ数のカウントに使用します。また、first_entry() ファンクションおよび next_entry() ファンクションと組み合わせて使用すると、結果セットの全探索時に残っているエントリ の数をカウントすることもできます。

構文

```
FUNCTION count_entries
(
    ld IN SESSION,
    msg IN MESSAGE
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-31 COUNT_ENTRY ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
msg	検索結果です。同期検索ルーチンのいずれかをコールして取得されるものと同一です。

戻り値

表 4-32 COUNT_ENTRY ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	結果セット中にエントリがある場合の戻り値は 0（ゼロ）以外です。 問題があった場合の戻り値は -1 です。

例外

表 4-33 COUNT_ENTRY ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_message	受信した msg ハンドルが無効な場合に呼び出されます。
count_entry_error	エントリのカウント中に問題があった場合に呼び出されます。

使用方法

`count_entries()` の戻り値は、結果セットに含まれるエントリの数です。`res` パラメータが無効な場合など、なんらかのエラーが発生した場合は、-1 が戻されます。`first_message()`、`next_message()`、`first_entry()`、`next_entry()`、`first_reference()`、`next_reference()` によって戻されたメッセージ、エントリまたはリファレンスを指定して `count_entries()` をコールすれば、結果セットの残りのエントリ数をカウントすることもできます。

関連項目

`DBMS_LDAP.first_entry()`、`DBMS_LDAP.next_entry()`

first_attribute ファンクション

first_attribute() ファンクションを使用すると、結果セットの中から、指定したエントリの最初の属性がフェッチされます。

構文

```
FUNCTION first_attribute
(
    ld          IN SESSION,
    ldapentry   IN MESSAGE,
    ber_elem OUT BER_ELEMENT
)
RETURN VARCHAR2;
```

パラメータ

表 4-34 FIRST_ATTRIBUTE ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
ldapentry	属性を調べる対象となるエントリです。first_entry() または next_entry() の戻り値と同一です。
ber_elem	エントリ内のどの属性が読み取られたのかを記録するために使用される BER ELEMENT のハンドルです。

戻り値

表 4-35 FIRST_ATTRIBUTE ファンクションの戻り値

値	説明
VARCHAR2 (ファンクション戻り値)	属性が存在する場合の戻り値はその属性の名前です。 属性が存在しない場合、またはエラーが発生した場合の戻り値は NULL です。
ber_elem	DBMS_LDAP.next_attribute() で、すべての属性に対して同一処理を反復するために使用するハンドルです。

例外

表 4-36 FIRST_ATTRIBUTE ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル 1d が無効な場合に呼び出されます。
invalid_message	受信した msg ハンドルが無効な場合に呼び出されます。

使用方法

エントリの様々な属性に対して属性名の取出しを繰り返し行うには、first_attribute() のパラメータとして戻された BER_ELEMENT のハンドルを、次の next_attribute() のコールで使用する必要があります。また、first_attribute() のコールで戻された属性の名前を、get_values() または get_values_len() のコールで使用すると、その属性の値を取得できます。

関連項目

DBMS_LDAP.next_attribute()、DBMS_LDAP.get_values()、DBMS_LDAP.get_values_len()、DBMS_LDAP.first_entry()、DBMS_LDAP.next_entry()

next_attribute ファンクション

next_attribute() ファンクションを使用すると、結果セットの中から、指定したエントリの次の属性がフェッチされます。

構文

```
FUNCTION next_attribute
(
    ld          IN SESSION,
    ldapentry   IN MESSAGE,
    ber_elem IN BER_ELEMENT
)
RETURN VARCHAR2;
```

パラメータ

表 4-37 NEXT_ATTRIBUTE ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
ldapentry	属性を調べる対象となるエントリです。first_entry() または next_entry() の戻り値と同一です。
ber_elem	エントリ内のどの属性が読み取られたのかを記録するために使用される BER ELEMENT のハンドルです。

戻り値

表 4-38 NEXT_ATTRIBUTE ファンクションの戻り値

値	説明
VARCHAR2 (ファンクション戻り値)	属性が存在する場合の戻り値はその属性の名前です。

例外

表 4-39 NEXT_ATTRIBUTE ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_message	受信した msg ハンドルが無効な場合に呼び出されます。

使用方法

エントリの様々な属性に対して属性名の取出しを繰り返し行うには、`first_attribute()` のパラメータとして戻された `BER_ELEMENT` のハンドルを、次の `next_attribute()` のコールで使用する必要があります。また、`next_attribute()` のコールで戻された属性の名前を、`get_values()` または `get_values_len()` のコールで使用すると、その属性の値を取得できます。

関連項目

`DBMS_LDAP.first_attribute()`、`DBMS_LDAP.get_values()`、`DBMS_LDAP.get_values_len()`、`DBMS_LDAP.first_entry()`、`DBMS_LDAP.next_entry()`

get_dn ファンクション

get_dn() ファンクションを使用すると、結果セットの中から、指定したエントリの X.500 識別名が取り出されます。

構文

```
FUNCTION get_dn
(
    ld IN SESSION,
    ldapentry IN MESSAGE
)
RETURN VARCHAR2;
```

パラメータ

表 4-40 GET_DN ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
ldapentry	識別名が戻り値となるエントリです。

戻り値

表 4-41 GET_DN ファンクションの戻り値

値	説明
VARCHAR2 (ファンクション戻り値)	エントリの PL/SQL 文字列での X.500 識別名です。 問題があった場合の戻り値は NULL です。

例外

表 4-42 GET_DN ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_message	受信した msg ハンドルが無効な場合に呼び出されます。
get_dn_error	識別名を確認中に問題があった場合に呼び出されます。

使用方法

`get_dn()` ファンクションを使用すると、プログラム・ロジックが結果セット全体にわたって同一処理を反復する間に、エントリの識別名を取り出せます。また、この識別名を `explode_dn()` への入力として使用すると、その識別名の個々の構成要素を取り出せます。

関連項目

`DBMS_LDAP.explode_dn()`

get_values ファンクション

get_values() ファンクションを使用すると、特定のエントリの特定の属性に関連する値をすべて取り出せます。

構文

```
FUNCTION get_values
(
    ld    IN SESSION,
    ldapentry IN MESSAGE,
    attr  IN VARCHAR2
)
RETURN STRING_COLLECTION;
```

パラメータ

表 4-43 GET_VALUES ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
ldapentry	検索結果から戻されたエントリの有効なハンドルです。
attr	値を検索する対象となる属性の名前です。

戻り値

表 4-44 GET_VALUES ファンクションの戻り値

値	説明
STRING_COLLECTION (ファンクション戻り値)	指定した属性のすべての値を含む PL/SQL 文字列の集合です。 指定した属性に関連する値がない場合の戻り値は NULL です。

例外

表 4-45 GET_VALUES ファンクションの例外

例外	説明
invalid session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid message	受信したエントリのハンドルが無効な場合に呼び出されます。

使用方法

`get_values()` ファンクションは、最初に `first_entry()` または `next_entry()` のコールでエントリのハンドルを取り出してからコールしてください。属性の名前は、事前に判明している場合もあれば、`first_attribute()` または `next_attribute()` のコールで判明する場合があります。`get_values()` ファンクションでは、取り出す属性のデータ型は常に文字列であるとみなされます。バイナリ・データ型を取り出すには、`get_values_len()` を使用する必要があります。

関連項目

`DBMS_LDAP.first_entry()`、`DBMS_LDAP.next_entry()`、`DBMS_LDAP.count_values()`、`DBMS_LDAP.get_values_len()`

get_values_len ファンクション

get_values_len() ファンクションを使用すると、バイナリ構文を持つ属性の値を取り出せます。

構文

```
FUNCTION get_values_len
(
    ld    IN SESSION,
    ldapentrymsg IN MESSAGE,
    attr  IN VARCHAR2
)
RETURN BINVAL_COLLECTION;
```

パラメータ

表 4-46 GET_VALUES_LEN ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
ldapentrymsg	検索結果から戻されたエントリの有効なハンドルです。
attr	値の検索対象となる属性の文字列名です。

戻り値

表 4-47 GET_VALUES_LEN ファンクションの戻り値

値	説明
BINVAL_COLLECTION (ファンクション戻り値)	指定した属性のすべての値を含む PL/SQL RAW 型データの集合です。 指定した属性に関連する値がない場合の戻り値は NULL です。

例外

表 4-48 GET_VALUES_LEN ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_message	受信したエントリのハンドルが無効な場合に呼び出されます。

使用方法

`get_values_len()` ファンクションは、`first_entry()` または `next_entry()` のコールでエントリのハンドルを取り出してからコールしてください。属性の名前は、事前に判明している場合もあれば、`first_attribute()` または `next_attribute()` のコールによって初めて判明する場合もあります。このファンクションは、バイナリの属性値および非バイナリの属性値の取出しに使用できます。

関連項目

`DBMS_LDAP.first_entry()`、`DBMS_LDAP.next_entry()`、`DBMS_LDAP.count_values_len()`、`DBMS_LDAP.get_values()`

delete_s ファンクション

delete_s() ファンクションを使用すると、LDAP ディレクトリ情報ツリー内のリーフ・エン
トリを削除できます。

構文

```
FUNCTION delete_s
(
    ld          IN SESSION,
    entrydn IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-49 DELETE_S ファンクションのパラメータ

パラメータ名	説明
ld	有効な LDAP セッションです。
entrydn	削除するエントリの X.500 識別名です。

戻り値

表 4-50 DELETE_S ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	削除操作が正常終了した場合の戻り値は DBMS_LDAP.SUCCESS です。その他の場合は例外が呼び出されます。

例外

表 4-51 DELETE_S ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効な場合に呼び出されます。
invalid_entry_dn	エントリの識別名が無効な場合に呼び出されます。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー 文字列で、エラーの詳細が説明されます。

使用方法

`delete_s()` ファンクションを使用すると、LDAP ディレクトリ情報ツリー内のリーフ・レベルのエントリのみを削除できます。リーフ・レベルのエントリとは、下位に子エントリまたは `ldap` エントリを持たないエントリのことです。このファンクションは、リーフ以外のエントリの削除には使用できません。

関連項目

DBMS_LDAP.modrdn2_s()

modrdn2_s ファンクション

modrdn2_s() ファンクションを使用すると、エントリの相対識別名を変更できます。

構文

```
FUNCTION modrdn2_s
(
    ld IN SESSION,
    entrydn IN VARCHAR2
    newrdn IN VARCHAR2
    deleteoldrdn IN PLS_INTEGER
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-52 MODRDN2_S ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
entrydn	エントリの識別名です（このエントリはディレクトリ情報ツリー内のリーフ・ノードです）。
newrdn	エントリの新しい相対識別名です。
deleteoldrdn	0（ゼロ）以外にした場合は、古い名前から引き継いだ属性値をエントリから削除する必要があることを示すブール値です。

戻り値

表 4-53 MODRDN2_S ファンクションの戻り値

値	説明
PLS_INTEGER (ファンクション戻り値)	操作が正常終了した場合の戻り値は DBMS_LDAP.SUCCESS です。 その他の場合は例外が呼び出されます。

例外

表 4-54 MODRDN2_S ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル 1d が無効な場合に呼び出されます。
invalid_entry_dn	エントリの識別名が無効な場合に呼び出されます。
invalid_rdn	LDAP 相対識別名が無効です。
invalid_deleteoldrdn	LDAP deleteoldrdn が無効です。
general_error	その他すべてのエラーに対応します。この例外に関連するエラー文字列で、エラーの詳細が説明されます。

使用方法

nodrdn2_s() ファンクションを使用すると、ディレクトリ情報ツリーのリーフ・ノードの名前を変更できます。認識の指標となる相対識別名のみが変更されます。LDAP v3 規格でこのファンクションを使用しないでください。同一の基本機能を持つ rename_s() を使用してください。

関連項目

DBMS_LDAP.rename_s()

err2string ファンクション

err2string() ファンクションを使用すると、LDAP エラー・コードを、API の動作環境で使用されている各国語の文字列に変換できます。

構文

```
FUNCTION err2string
(
    ldap_err IN PLS_INTEGER
)
RETURN VARCHAR2;
```

パラメータ

表 4-55 ERR2STRING ファンクションのパラメータ

パラメータ	説明
ldap_err	いずれかの API コールから戻されたエラー番号です。

戻り値

表 4-56 ERR2STRING ファンクションの戻り値

値	説明
VARCHAR2 (ファンクション戻り値)	各国語へ適切に変換された文字列です。この文字列で、エラーの詳細が説明されます。

例外

表 4-57 ERR2STRING ファンクションの例外

例外	説明
該当なし	ありません。

使用方法

このリリースでは、API コールにエラーが発生した場合、例外処理メカニズムによって自動的にこのファンクションがコールされます。

関連項目

該当なし

create_mod_array ファンクション

create_mod_array() ファンクションを使用すると、modify_s() ファンクションまたは add_s() ファンクションを使用してエントリに適用される変更配列に、メモリーが割り当てられます。

構文

```
FUNCTION create_mod_array
(
    num IN PLS_INTEGER
)
RETURN MOD_ARRAY;
```

パラメータ

表 4-58 CREATE_MOD_ARRAY ファンクションのパラメータ

パラメータ	説明
num	追加または変更する属性の数です。

戻り値

表 4-59 CREATE_MOD_ARRAY ファンクションの戻り値

値	説明
MOD_ARRAY (ファンクション戻り値)	このデータ構造により、LDAP 変更配列へのポインタが保持されます。 問題があった場合の戻り値は NULL です。

例外

表 4-60 CREATE_MOD_ARRAY ファンクションの例外

例外	説明
該当なし	LDAP 固有の例外は呼び出されません。

使用方法

このファンクションは、DBMS_LDAP.add_s および DBMS_LDAP.modify_s を使用するための準備段階の 1 つです。add_s または modify_s のコールが終了した後にメモリーを解放するには、DBMS_LDAP.free_mod_array をコールする必要があります。

関連項目

DBMS_LDAP.populate_mod_array()、DBMS_LDAP.modify_s()、DBMS_LDAP.add_s()、DBMS_LDAP.free_mod_array()

populate_mod_array プロシージャ（文字列バージョン）

追加操作または変更操作に、1 組の属性情報を代入します。

構文

```
PROCEDURE populate_mod_array
(
    modptr    IN DBMS_LDAP.MOD_ARRAY,
    mod_op    IN PLS_INTEGER,
    mod_type  IN VARCHAR2,
    modval    IN DBMS_LDAP.STRING_COLLECTION
);
```

パラメータ

表 4-61 POPULATE_MOD_ARRAY（文字列バージョン） プロシージャのパラメータ

パラメータ	説明
modptr	このデータ構造により、LDAP 変更配列へのポインタが保持されます。
mod_op	このフィールドで、実行する変更の型を指定します。
mod_type	このフィールドで、変更を適用する属性型の名前を指定します。
modval	このフィールドで、追加、削除または置換する属性値を指定します。対象は文字列値のみです。

戻り値

表 4-62 POPULATE_MOD_ARRAY（文字列バージョン） プロシージャの戻り値

値	説明
該当なし	

例外

表 4-63 POPULATE_MOD_ARRAY（文字列バージョン）プロシージャの例外

例外	説明
invalid_mod_array	LDAP 変更配列が無効です。
invalid_mod_option	LDAP 変更オプションが無効です。
invalid_mod_type	LDAP 変更型が無効です。
invalid_mod_value	LDAP 変更値が無効です。

使用方法

このファンクションは、DBMS_LDAP.add_s および DBMS_LDAP.modify_s を使用するための準備段階の 1 つです。DBMS_LDAP.create_mod_array をコールした後に、このプロシージャをコールする必要があります。

関連項目

DBMS_LDAP.create_mod_array()、DBMS_LDAP.modify_s()、DBMS_LDAP.add_s()、DBMS_LDAP.free_mod_array()

populate_mod_array プロシージャ (バイナリ・バージョン)

追加操作または変更操作に、1 組の属性情報を代入します。DBMS_LDAP.create_mod_array() をコールした後に、このプロシージャをコールする必要があります。

構文

```
PROCEDURE populate_mod_array
(
    modptr    IN DBMS_LDAP.MOD_ARRAY,
    mod_op    IN PLS_INTEGER,
    mod_type  IN VARCHAR2,
    modbval   IN DBMS_LDAP.BERVAL_COLLECTION
);
```

パラメータ

表 4-64 POPULATE_MOD_ARRAY (バイナリ・バージョン) プロシージャのパラメータ

パラメータ	説明
modptr	このデータ構造により、LDAP 変更配列へのポインタが保持されます。
mod_op	このフィールドで、実行する変更の型を指定します。
mod_type	このフィールドで、変更を適用する属性型の名前を指定します。
modbval	このフィールドで、追加、削除または置換する属性値を指定します。対象はバイナリ値のみです。

戻り値

表 4-65 POPULATE_MOD_ARRAY (バイナリ・バージョン) プロシージャの戻り値

値	説明
該当なし	

例外

表 4-66 POPULATE_MOD_ARRAY (バイナリ・バージョン) プロシージャの例外

例外	説明
invalid_mod_array	LDAP 変更配列が無効です。
invalid_mod_option	LDAP 変更オプションが無効です。
invalid_mod_type	LDAP 変更型が無効です。
invalid_mod_value	LDAP 変更値が無効です。

使用方法

このファンクションは、DBMS_LDAP.add_s および DBMS_LDAP.modify_s を使用するための準備段階の 1 つです。DBMS_LDAP.create_mod_array をコールした後に、このプロシージャをコールする必要があります。

関連項目

DBMS_LDAP.create_mod_array()、DBMS_LDAP.modify_s()、DBMS_LDAP.add_s()、DBMS_LDAP.free_mod_array()

modify_s ファンクション

既存の LDAP ディレクトリ・エントリの同期変更を実行します。

構文

```
FUNCTION modify_s
(
    ld          IN DBMS_LDAP.SESSION,
    entrydn     IN VARCHAR2,
    modptr      IN DBMS_LDAP.MOD_ARRAY
)
    RETURN PLS_INTEGER;
```

パラメータ

表 4-67 MODIFY_S ファンクションのパラメータ

パラメータ	説明
ld	このパラメータは LDAP セッションのハンドルで、DBMS_LDAP.init() のコールが正常終了した場合の戻り値と同一です。
entrydn	このパラメータで、内容を変更するディレクトリ・エントリの名前を指定します。
modptr	このパラメータは LDAP 変更構造体のハンドルで、DBMS_LDAP.create_mod_array() のコールが正常終了した場合の戻り値と同一です。

戻り値

表 4-68 MODIFY_S ファンクションの戻り値

値	説明
PLS_INTEGER	変更操作の成功または失敗を示します。

例外

表 4-69 MODIFY_S ファンクションの例外

例外	説明
invalid_session	LDAP セッションが無効です。
invalid_entry_dn	LDAP エントリ識別名が無効です。
invalid_mod_array	LDAP 変更配列が無効です。

使用方法

このファンクションは、`DBMS_LDAP.create_mod_array()` および `DBMS_LDAP.populate_mod_array()` のコールが正常終了した後にコールする必要があります。

関連項目

`DBMS_LDAP.create_mod_array()`、`DBMS_LDAP.populate_mod_array()`、`DBMS_LDAP.add_s()`、`DBMS_LDAP.free_mod_array()`

add_s ファンクション

LDAP ディレクトリに新規エントリを同期的に追加します。add_s をコールする前に、まず DBMS_LDAP.create_mod_array() と DBMS_LDAP.populate_mod_array() をコールする必要があります。

構文

```
FUNCTION add_s
(
    ld          IN DBMS_LDAP.SESSION,
    entrydn     IN VARCHAR2,
    modptr      IN DBMS_LDAP.MOD_ARRAY
)
    RETURN PLS_INTEGER;
```

パラメータ

表 4-70 ADD_S ファンクションのパラメータ

パラメータ	説明
ld	このパラメータは LDAP セッションのハンドルで、DBMS_LDAP.init() のコールが正常終了した場合の戻り値と同一です。
entrydn	このパラメータで、作成するディレクトリ・エントリの名前を指定します。
modptr	このパラメータは LDAP 変更構造体のハンドルで、DBMS_LDAP.create_mod_array() のコールが正常終了した場合の戻り値と同一です。

戻り値

表 4-71 ADD_S ファンクションの戻り値

値	説明
PLS_INTEGER	変更操作の成功または失敗を示します。

例外

表 4-72 ADD_S ファンクションの例外

例外	説明
invalid_session	LDAP セッションが無効です。
invalid_entry_dn	LDAP エントリ識別名が無効です。
invalid_mod_array	LDAP 変更配列が無効です。

使用方法

追加するエントリの親エントリは、ディレクトリ内にすでに存在する必要があります。
このファンクションは、DBMS_LDAP.create_mod_array() および DBMS_LDAP.populate_mod_array() のコールが正常終了した後にコールする必要があります。

関連項目

DBMS_LDAP.create_mod_array()、DBMS_LDAP.populate_mod_array()、DBMS_LDAP.modify_s()、DBMS_LDAP.free_mod_array()

free_mod_array プロシージャ

DBMS_LDAP.create_mod_array() によって割り当てられたメモリーを解放します。

構文

```
PROCEDURE free_mod_array
(
    modptr IN DBMS_LDAP.MOD_ARRAY
);
```

パラメータ

表 4-73 FREE_MOD_ARRAY プロシージャのパラメータ

パラメータ	説明
modptr	このパラメータは LDAP 変更構造体のハンドルで、DBMS_LDAP.create_mod_array() のコールが正常終了した場合の戻り値と同一です。

戻り値

表 4-74 FREE_MOD_ARRAY プロシージャの戻り値

値	説明
該当なし	

例外

表 4-75 FREE_MOD_ARRAY プロシージャの例外

例外	説明
該当なし	LDAP 固有の例外は呼び出されません。

使用方法

該当なし

関連項目

DBMS_LDAP.populate_mod_array(), DBMS_LDAP.modify_s(), DBMS_LDAP.add_s(), DBMS_LDAP.create_mod_array()

count_values ファンクション

DBMS_LDAP.get_values() によって戻された値の数をカウントします。

構文

```
FUNCTION count_values
(
    values IN DBMS_LDAP.STRING_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-76 COUNT_VALUES ファンクションのパラメータ

パラメータ	説明
values	文字列値の集合です。

戻り値

表 4-77 COUNT_VALUES ファンクションの戻り値

値	説明
PLS_INTEGER	操作の成功または失敗を示します。

例外

表 4-78 COUNT_VALUES ファンクションの例外

例外	説明
該当なし	LDAP 固有の例外は呼び出されません。

使用方法

該当なし

関連項目

DBMS_LDAP.count_values_len()、DBMS_LDAP.get_values()

count_values_len ファンクション

DBMS_LDAP.get_values_len() によって戻された値の数をカウントします。

構文

```
FUNCTION count_values_len
(
    values IN DBMS_LDAP.BINVAL_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-79 COUNT_VALUES_LEN ファンクションのパラメータ

パラメータ	説明
values	バイナリ値の集合です。

戻り値

表 4-80 COUNT_VALUES_LEN ファンクションの戻り値

値	説明
PLS_INTEGER	操作の成功または失敗を示します。

例外

表 4-81 COUNT_VALUES_LEN ファンクションの例外

例外	説明
該当なし	LDAP 固有の例外は呼び出されません。

使用方法

該当なし

関連項目

DBMS_LDAP.count_values(), DBMS_LDAP.get_values_len()

rename_s ファンクション

LDAP エントリの名前を同期的に変更します。

構文

```
FUNCTION rename_s
(
    ld          IN SESSION,
    dn          IN VARCHAR2,
    newrdn     IN VARCHAR2,
    newparent   IN VARCHAR2,
    deleteoldrdn IN PLS_INTEGER,
    serverctrls IN LDAPCONTROL,
    clientctrls IN LDAPCONTROL
)
    RETURN PLS_INTEGER;
```

パラメータ

表 4-82 RENAME_S ファンクションのパラメータ

パラメータ	説明
ld	このパラメータは LDAP セッションのハンドルで、DBMS_LDAP.init() のコールが正常終了した場合の戻り値と同一です。
dn	このパラメータで、改名または移動するディレクトリ・エントリの名前を指定します。
newrdn	このパラメータで、新しい相対識別名を指定します。
newparent	このパラメータで、新しい親の識別名を指定します。
deleteoldrdn	このパラメータで、古い相対識別名を保持するかどうかを指定します。この値を 1 にすると、古い相対識別名が削除されます。
serverctrls	現在はサポートされていません。
clientctrls	現在はサポートされていません。

戻り値

表 4-83 RENAME_S ファンクションの戻り値

値	説明
PLS_INTEGER	操作の成功または失敗を示します。

例外

表 4-84 RENAME_S ファンクションの例外

例外	説明
invalid_session	LDAP セッションが無効です。
invalid_entry_dn	LDAP 識別名が無効です。
invalid_rdn	LDAP 相対識別名が無効です。
invalid_newparent	LDAP の新規の親が無効です。
invalid_deleteoldrdn	LDAP deleteoldrdn が無効です。

使用方法

該当なし

関連項目

DBMS_LDAP.modrdn2_s()

explode_dn ファンクション

識別名を個々の構成要素に分割します。

構文

```
FUNCTION explode_dn
(
    dn          IN VARCHAR2,
    notypes IN PLS_INTEGER
)
    RETURN STRING_COLLECTION;
```

パラメータ

表 4-85 EXPLODE_DN ファンクションのパラメータ

パラメータ	説明
dn	このパラメータで、分割するディレクトリ・エントリの名前を指定します。
notypes	このパラメータで、属性タグを戻すかどうかを指定します。この値を 0（ゼロ）にすると、属性タグは戻されません。

戻り値

表 4-86 EXPLODE_DN ファンクションの戻り値

値	説明
STRING_COLLECTION	文字列の配列です。識別名が分割できない場合は NULL が戻されます。

例外

表 4-87 EXPLODE_DN ファンクションの例外

例外	説明
invalid_entry_dn	LDAP 識別名が無効です。
invalid_notypes	LDAP notypes 値が無効です。

使用方法

該当なし

関連項目

DBMS_LDAP.get_dn()

open_ssl ファンクション

すでに確立されている LDAP 接続を介して SSL 接続を確立します。

構文

```
FUNCTION open_ssl
(
    ld                IN SESSION,
    sslwrl            IN VARCHAR2,
    sslwalletpasswd   IN VARCHAR2,
    sslauth           IN PLS_INTEGER
)
    RETURN PLS_INTEGER;
```

パラメータ

表 4-88 OPEN_SSL ファンクションのパラメータ

パラメータ	説明
ld	このパラメータは LDAP セッションのハンドルで、DBMS_LDAP.init() のコールが正常終了した場合の戻り値と同一です。
sslwrl	このパラメータで、Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
sslwalletpasswd	このパラメータで、Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
sslauth	このパラメータで、SSL 認証モード（SSL 認証なしの場合は 1、サーバー認証の場合は 2、クライアントとサーバーの認証の場合は 3）を指定します。

戻り値

表 4-89 OPEN_SSL ファンクションの戻り値

値	説明
PLS_INTEGER	操作の成功または失敗を示します。

例外

表 4-90 OPEN_SSL ファンクションの例外

例外	説明
invalid_session	LDAP セッションが無効です。
invalid_ssl_wallet_loc	LDAP SSL の Wallet の場所が無効です。
invalid_ssl_wallet_passwd	LDAP SSL の Wallet のパスワードが無効です。
invalid_ssl_auth_mode	LDAP SSL 認証モードが無効です。

使用方法

有効な LDAP セッションを取得するには、まず `DBMS_LDAP.init()` をコールする必要があります。

関連項目

`DBMS_LDAP.init()`

msgfree ファンクション

同期検索ファンクションによって戻されたメッセージ・ハンドルに対応付けられている結果セットを解放します。

構文

```
FUNCTION msgfree
(
    res                IN MESSAGE
)
RETURN PLS_INTEGER;
```

パラメータ

表 4-91 MSGFREE ファンクションのパラメータ

パラメータ	説明
res	メッセージ・ハンドルです。同期検索ルーチンのいずれかをコールして取得されるものと同一です。

戻り値

表 4-92 MSGFREE ファンクションの戻り値

値	説明
PLS_INTEGER	結果セット内にある最後のメッセージのタイプを示します。 このファンクションの戻り値は、次の値のいずれかになります。 <ul style="list-style-type: none">DBMS_LDAP.LDAP_RES_BINDDBMS_LDAP.LDAP_RES_SEARCH_ENTRYDBMS_LDAP.LDAP_RES_SEARCH_REFERENCEDBMS_LDAP.LDAP_RES_SEARCH_RESULTDBMS_LDAP.LDAP_RES_MODIFYDBMS_LDAP.LDAP_RES_ADDDBMS_LDAP.LDAP_RES_DELETEDBMS_LDAP.LDAP_RES_MODDNDBMS_LDAP.LDAP_RES_COMPAREDBMS_LDAP.LDAP_RES_EXTENDED

例外

該当なし。LDAP 固有の例外は呼び出されません。

使用方法

該当なし

関連項目

`DBMS_LDAP.search_s()`、`DBMS_LDAP.search_st()`

ber_free ファンクション

BER ELEMENT へのハンドルに対応付けられたメモリーを解放します。

構文

```
FUNCTION ber_free
(
    ber_elem IN BER_ELEMENT,
    freebuf  IN PLS_INTEGER
)
```

パラメータ

表 4-93 BER_FREE ファンクションのパラメータ

パラメータ	説明
ber_elem	BER ELEMENT へのハンドルです。
freebuf	DBMS_LDAP.first_attribute() から戻された BER ELEMENT を解放している間、このフラグの値は 0（ゼロ）であることが必要です。それ以外の場合、このフラグの値は 1 であることが必要です。 このパラメータのデフォルト値は 0（ゼロ）です。

戻り値

該当なし

例外

該当なし。LDAP 固有の例外は呼び出されません。

使用方法

該当なし

関連項目

DBMS_LDAP.first_attribute()、DBMS_LDAP.next_attribute()

第 II 部

LDAP API に対する Oracle の拡張機能

第 II 部では、Oracle 固有の Application Program Interface (API) について説明し、Oracle9i Application Server リリース 2 (9.0.2) のユーザーやグループなどに関する基本的な概念、Java API の情報 (oracle.ldap.util クラス)、PL/SQL API の情報、およびプロビジョニング・インタフェースに関する情報を示します。第 II 部は、次の章で構成されています。

- 第 5 章「Oracle の拡張機能の概要」
- 第 6 章「Oracle Internet Directory の Java API」
- 第 7 章「DBMS_LDAP_UTL PL/SQL パッケージ」
- 第 8 章「プロビジョニング統合アプリケーションの開発」

Oracle の拡張機能の概要

この章では、アプリケーションをディレクトリ対応にする方法について説明します。この章では、次の項目について説明します。

- [LDAP アクセス・モデル](#)
- [LDAP のモデル化されたエンティティ](#)
- [API の拡張機能 : 概要と使用モデル](#)
- [API の拡張機能 : 前提](#)
- [インストールと最初の使用](#)

LDAP アクセス・モデル

ほとんどのディレクトリ対応アプリケーションは、複数のユーザー要求を同時に処理する中間層です。図 5-1 は、このような中間層環境での Lightweight Directory Access Protocol (LDAP) の使用方法を示しています。

図 5-1 中間層環境での Oracle Internet Directory

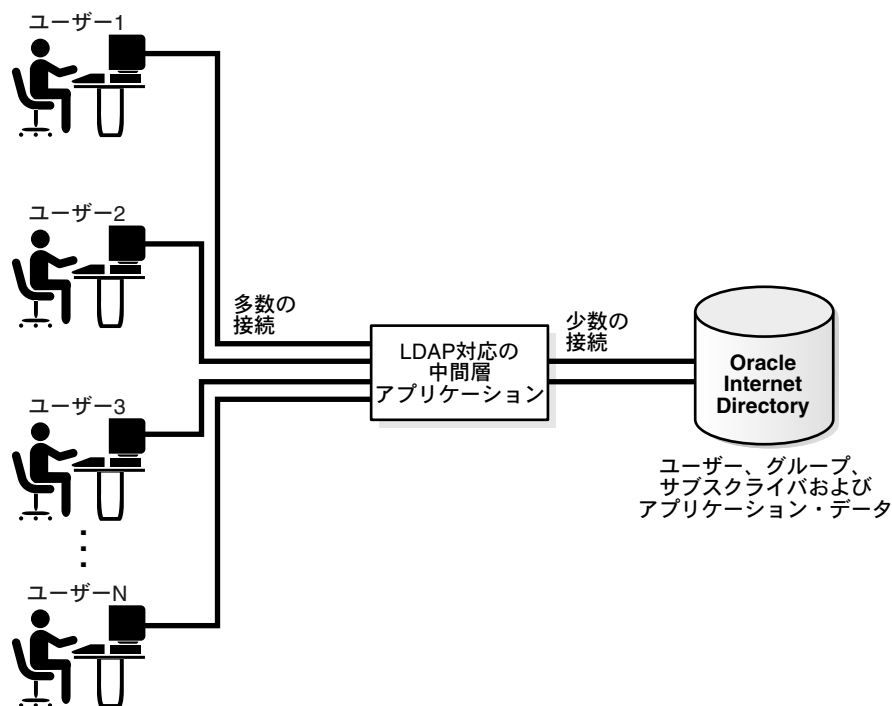


図 5-1 のように、アプリケーションは、複数のユーザーがアクセスできる中間層プログラムまたはバックエンド・プログラムとして機能します。ユーザー要求の実行に LDAP 操作が必要な場合、アプリケーションは、あらかじめ確立されている Oracle Internet Directory への小規模な一連の接続を使用して操作を実行します。

この項では、このアクセス・モデルの実装方法について説明します。次の項目について説明します。

- アプリケーションのインストール・ロジック
- アプリケーションの起動とブートストラップに関するロジック
- アプリケーション実行時のロジック
- アプリケーションの停止ロジック
- アプリケーションの削除ロジック

アプリケーションのインストール・ロジック

1. アプリケーションに対応する識別情報を Oracle Internet Directory に作成します。アプリケーションはこの識別情報を使用して、ほとんどの LDAP 操作を実行します。
2. 特定の LDAP 認可を指定するために、この識別情報を正しい LDAP グループに含めます。その結果、次の操作が可能となります。
 - ユーザー資格証明を受け入れ、Oracle Internet Directory に対して認証します。
 - 特定の LDAP 操作をユーザーのために実行する必要がある場合は、ユーザーの代理、つまりプロキシ・ユーザーとなります。

アプリケーションの起動とブートストラップに関するロジック

アプリケーションは、それ自体を Oracle Internet Directory に対して認証するための資格証明を取得する必要があります。

Oracle Internet Directory に構成メタデータを格納しているアプリケーションは、そのメタデータを取得して、アプリケーションの他の部分を初期化する必要があります。

次に、アプリケーションは、接続のプールを確立してユーザー要求を処理する必要があります。

アプリケーション実行時のロジック

LDAP 操作が必要なすべてのエンド・ユーザー要求について、アプリケーションは次の処理を行う必要があります。

- LDAP 接続のプールから接続を選択します。
- Oracle9iAS Single Sign-On が使用されていない場合は、必要に応じてエンド・ユーザーを認証します。
- エンド・ユーザーの有効な権利を使用して LDAP 操作を実行する必要がある場合は、ユーザーをエンド・ユーザー ID に切り替えます。

- 通常の Application Program Interface (API) またはこの章で説明されている拡張機能を使用して、LDAP 操作を実行します。
- アプリケーションでプロキシ操作を実行した場合は、操作完了後、有効なユーザーがアプリケーションでの識別情報自体であることを確認します。
- LDAP 接続を接続のプールに戻します。

アプリケーションの停止ロジック

未処理の LDAP 操作を中止して、すべての LDAP 接続をクローズします。

アプリケーションの削除ロジック

アプリケーション識別情報とそのアプリケーション識別情報に指定されている関連の LDAP 認可を削除します。

LDAP のモデル化されたエンティティ

LDAP API ヘルプ・アプリケーションに対する Oracle の拡張機能によって、次のエンティティに関する LDAP 情報を取得および設定できます。

ユーザー	1 つ以上のアプリケーションへのアクセス権を持つ Oracle Internet Directory のエンタープライズ・ユーザー。
グループ	通常は 1 つの認可を表す、エンタープライズ・ユーザーの集合。これらの集合をディレクトリ内に格納しているディレクトリ対応のアプリケーションは、グループの位置を特定し、グループ・メンバーシップの問合せを実行できる必要があります。
サブスクリイバ	ホスト環境のモデル化されたエンティティ。通常、サブスクリイバは 1 つ以上の Oracle ホスト対応製品にサブスクリイブする企業です。

次に、これらのエンティティに対して Oracle Internet Directory で必要なアプリケーションについて説明します。次の項目について説明します。

- ユーザー
- グループ
- サブスクリイバ

ユーザー

ディレクトリ対応のアプリケーションは、ユーザーに関連する次の操作について、Oracle Internet Directory にアクセスする必要があります。

- 姓や自宅の住所などと同じ方法で、ユーザー・エントリ自体の属性として格納されているユーザー・エントリ・プロパティ。
- ユーザーに関係するが、DIT 内の別の位置に格納されている拡張ユーザー設定項目。これらのプロパティは、さらに次のように分類できます。
 - すべてのアプリケーションに共通の拡張ユーザー・プロパティ。これらのプロパティは、Oracle コンテキストの共通の位置に格納されています。
 - 特定のアプリケーションに固有の拡張ユーザー・プロパティ。これらのプロパティは、アプリケーション固有の **DIT** に格納されています。
- ユーザーのグループ・メンバーシップの問合せ。
- 単純な名前のユーザーと資格証明の認証。

通常、ユーザーは次のいずれかを使用して識別されます。

- 完全に修飾された LDAP 識別名
- **Global Unique Identifier (GUID)**
- 単純なユーザー名とサブスクライバ名

グループ

Oracle Internet Directory では、グループは識別名の集合としてモデル化されます。ディレクトリ対応のアプリケーションは、グループのプロパティを取得し、指定したユーザーがそのグループのメンバーであることを検証するために、グループに関連する次の操作について、Oracle Internet Directory にアクセスする必要があります。

通常、グループは次のいずれかを使用して識別されます。

- 完全に修飾された LDAP 識別名
- Global Unique Identifier
- 単純なグループ名とサブスクライバ名

サブスクライバ

サブスクライバは、多数の Oracle 製品によって提供されるホスト対応サービスをサブスクライブするエンティティまたは組織です。ディレクトリ対応アプリケーションは、サブスクライバ・プロパティ（ユーザー検索ベースやパスワード・ポリシーなど）を取得して、新しいサブスクライバを作成するために Oracle Internet Directory にアクセスする必要があります。

通常、サブスクライバは次のいずれかを使用して識別されます。

- 完全に修飾された LDAP 識別名
- Global Unique Identifier
- 単純なサブスクライバ名

API の拡張機能 : 概要と使用モデル

前述の項で説明されているように、ディレクトリと統合するすべてのアプリケーションには、準拠する必要がある規則がいくつかあります。この章で説明する API 拡張機能により、これらの規則にアプリケーションを準拠させることができます。

API の拡張機能 : 前提

この章で説明する API の拡張機能は、次の前提に基づいています。

- LDAP 接続が必要なすべての API ファンクションは、各プログラミング言語による適切なメカニズムを使用して、アプリケーションがすでに確立されていると仮定します。
- これらの API ファンクションに渡される LDAP 接続は、指定した操作を実行するために必要な権限を持つ識別情報に関連付けられていると仮定します。つまり、新規 API ファンクションには、それ自体の認可モデルがありません。新規 API ファンクションは、LDAP 認可モデルに依存しています。認可が不十分であることが原因で特定の操作に失敗した場合、単にエラーが起動アプリケーションに転送されます。
- すべての Java ベースのアプリケーションは、LDAP 接続のインタフェースとして Sun 社の JNDI を使用していると仮定します。
- すべての PL/SQL ベースのアプリケーションは、LDAP 接続に Oracle 社の DBMS_LDAP パッケージを使用していると仮定します。

システムの配置

図 5-2 は、既存の API に関連した API の拡張機能の配置を示しています。

図 5-2 Oracle API の拡張機能と既存の API

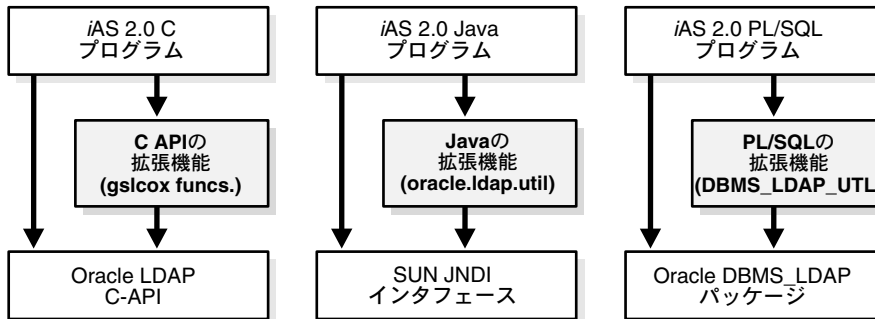


図 5-2 のように、言語（PL/SQL および Java）では、この章で説明する API の拡張機能は既存の API の層の上に位置します。

- PL/SQL プログラムについては、Oracle 社の DBMS_LDAP PL/SQL API
- Java プログラムについては、Sun 社の LDAP JNDI Service Provider

アプリケーションは、接続の確立やクローズなどの通常の操作のために、基礎となる API にアクセスする必要があります。アプリケーションは、既存の API を使用して、API の拡張機能ではカバーされない他の LDAP エントリを検索することもできます。

API の拡張機能 : 機能の分類

API の拡張機能は、操作対象のエンティティに基づいて次のように分類できます。

- ユーザー管理—この機能によって、アプリケーションはユーザーに関する各種プロパティを取得または設定できます。
- グループ管理—この機能によって、アプリケーションはグループのプロパティを問い合わせることができます。
- サブスクライバ管理—この機能によって、アプリケーションは、ユーザー検索ベースなどのサブスクライバ関連のプロパティを取得または設定できます。
- アプリケーション管理—この機能によって、アプリケーションは Oracle Internet Directory のアプリケーション・メタデータを管理できます。
- その他—この機能（GUID の生成）は汎用的に適用できます。

API の拡張機能 : 使用モデル

この章で説明する拡張機能の主なユーザーは、ユーザー、グループ、アプリケーションまたはサブスクリイバについて LDAP 検索の実行が必要な、中間層またはバックエンドのアプリケーションです。この項では、API の拡張機能をこれらのアプリケーションのロジックに統合する方法、つまり API の拡張機能の使用方法のみを説明します。

関連項目： 使用モデルの概要については、5-2 ページの「[LDAP アクセス・モデル](#)」を参照してください。

図 5-3 は、この章で説明されている API の拡張機能を使用するためのプログラムによる制御フローを示しています。

図 5-3 API の拡張機能のプログラム・フロー

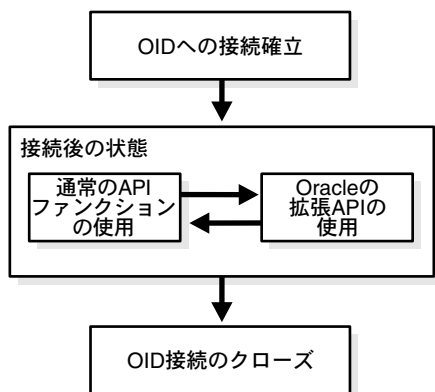


図 5-3 のように、アプリケーションは、最初に Oracle Internet Directory への接続を確立します。その後、既存の API ファンクションと API の拡張機能を交互に使用できます。

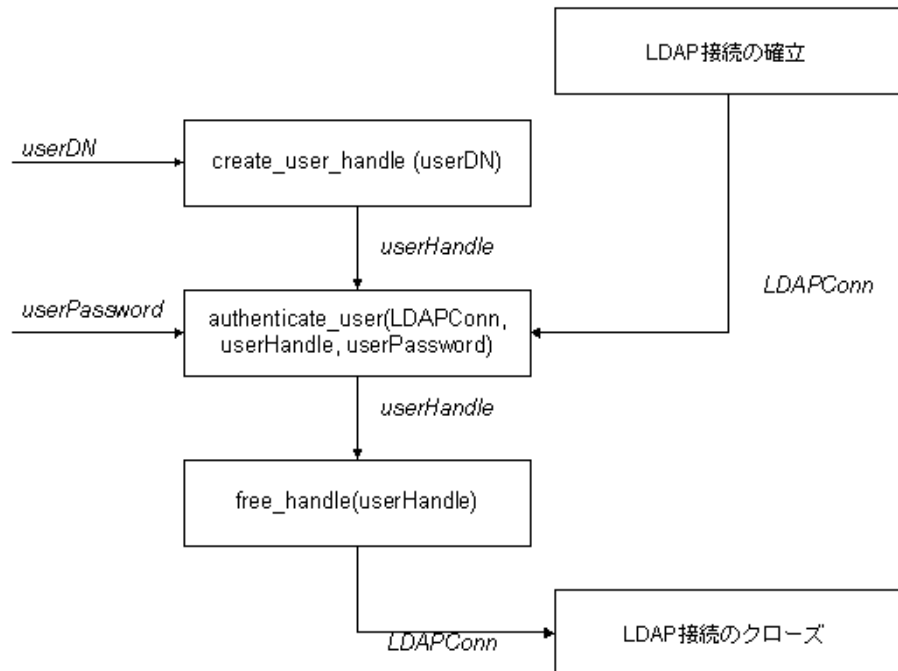
PL/SQL 言語のプログラム抽象化

この章で説明されているほとんどの拡張機能は、ユーザー、グループ、サブスクリバおよびアプリケーションなど、特定の LDAP エンティティに関するデータにアクセスするための補助機能を提供します。多くの場合、開発者は、これらのエンティティの 1 つに対する参照を API ファンクションに渡す必要があります。API のこれらの拡張機能は、ハンドルと呼ばれる不透明なデータ構造を使用します。たとえば、ユーザーの認証が必要なアプリケーションは、次の手順に従います。

1. LDAP 接続を確立するか、接続のプールから接続を取得します。
2. ユーザーの入力に基づいて、ユーザー・ハンドルを作成します。このユーザー・ハンドルは、識別名、GUID または単純な Oracle9iAS Single Sign-On ID の場合があります。
3. LDAP 接続ハンドル、ユーザー・ハンドルおよび資格証明を使用して、ユーザーを認証します。
4. ユーザー・ハンドルを解放します。
5. LDAP 接続をクローズするか、接続をプールに戻します。

図 5-4 は、この使用モデルを示しています。

図 5-4 PL/SQL 言語のプログラム抽象化



Java 言語のプログラム抽象化

LDAP エンティティ（つまり、ユーザー、グループ、サブスクライバおよびアプリケーション）は、ハンドルではなく、`oracle.java.util` パッケージの Java オブジェクトとしてモデル化されます。他のすべてのユーティリティ機能は、個々のオブジェクトとして（GUID など）、あるいはユーティリティ・クラスの静的メンバー関数としてモデル化されます。

たとえば、ユーザーの認証が必要なアプリケーションは、次の手順に従う必要があります。

1. 指定されたユーザー識別名で、`oracle.ldap.util.User` オブジェクトを作成します。
2. 必要なプロパティのすべてを備えた `DirContext` JNDI オブジェクトを作成するか、あるいは `DirContext` オブジェクトのプールから JNDI オブジェクトを取得します。
3. `User.authenticateUser` メソッドを呼び出して、`DirContext` オブジェクトおよびユーザー資格証明への参照を渡します。
4. 既存の `DirContext` オブジェクトのプールから取得した `DirContext` オブジェクトは、そのプールに戻します。

C 言語や PL/SQL とは異なり、Java 言語の使用では、Java ガベージ・コレクション・メカニズムでオブジェクトの解放が実行されるため、オブジェクトを明示的に解放する必要はありません。

ユーザー管理機能

この項では、Java、C 言語および PL/SQL の LDAP API に関するユーザー管理機能について説明します。

Java 前述の例で説明されているように、ユーザーに関連する機能はすべて `oracle.ldap.util.User` と呼ばれる Java クラスで抽象化されます。次の手順は、この機能の高度な使用モデルです。

1. 識別名、GUID または単純な名前に基づいて、`oracle.ldap.util.User` オブジェクトを構成します。
2. 必要の場合は、`User.authenticationUser(DirContext, int, Object)` を呼び出して、ユーザーを認証します。
3. `User.getProperties(DirContext, String[])` を呼び出して、ユーザー・エントリ自体の属性を取得します。
4. `User.getExtendedProperties(DirContext, int, String[])` を呼び出して、ユーザーの拡張プロパティを取得します。`int` は、共有またはアプリケーション固有です。`String[]` は、希望するプロパティのタイプを示すオブジェクトです。`String[]` が NULL の場合は、指定したカテゴリの全プロパティが取得されます。
5. `PropertySetCollection.getProperties(int)` を呼び出して、手順 4 で戻されたプロパティの解析に必要なメタデータを取得します。

6. 拡張プロパティを解析し、アプリケーション固有のロジックを続行します。この解析は、アプリケーション固有のロジックによっても行われます。

インストールと最初の使用

Java API は、LDAP クライアントのインストールの一部としてインストールされます。

PL/SQL API は、Oracle9i データベースのインストールの一部としてインストールされます。PL/SQL API を使用するには、`$ORACLE_HOME/rdbms/admin` に格納されている `catldap.sql` と呼ばれるスクリプトを使用して、PL/SQL API をロードする必要があります。

Oracle Internet Directory の Java API

この章では、Oracle Internet Directory の Java Application Program Interface (Java API) に関する参照資料を示します。

この章では、次の項目について説明します。

- [クラスの説明](#)
- [クラス](#)
- [例外](#)

クラスの説明

この項では、クラスについて説明します。

この項では、次の項目について説明します。

- [User クラス](#)
- [Subscriber クラス](#)
- [Group クラス](#)
- [PropertySetCollection クラス](#)、[PropertySet クラス](#)および[Property クラス](#)

User クラス

User クラスは、サブスクライバの下の特定期間ユーザーを表すために使用されます。識別名、Global Unique Identifier (GUID) または単純な名前に基づいた適切なサブスクライバ ID の作成に加え、識別名、GUID または単純な名前を使用した User オブジェクトを作成できます。単純な名前を使用すると、ルートの Oracle コンテキストおよびサブスクライバの Oracle コンテキストの追加情報がユーザーの識別に使用されます。次に、ユーザーの構成例を示します。

```
User myuser = new User ( ctx,          // A valid InitialDirContext
                        Util.IDTYPE_DN,
                        "cn=user1,cn=users,o=oracle,dc=com",
                        Util.IDTYPE_DN,
                        "o=oracle,dc=com",
                        false);
```

この例の myuser は、その識別名を使用して定義されます。対応するサブスクライバもその識別名で識別されます。既存の Subscriber オブジェクトがある場合は、その Subscriber オブジェクトを使用して User オブジェクトを直接作成することもできます。たとえば、"o=oracle,dc=com" を表す Subscriber オブジェクト myOracleSubscriber の場合は、次のコードを使用して前述の例と同じ User オブジェクトを作成できます。

```
User myuser = new User ( ctx,          // A valid InitialDirContext
                        Util.IDTYPE_DN,
                        "cn=user1,cn=users,o=oracle,dc=com",
                        myOracleSubscriber,
                        false);
```


一部の共通 **User** オブジェクトの使用には、ユーザー・プロパティの設定や取得、およびユーザーの認証が含まれます。次に、ユーザーの認証例を示します。

```
if(myuser.authenticateUser( ctx
                        User.CREDTYPE_PASSWD,
                        "welcome" )) {
    // do work here
}
```

前述の例のユーザーは、クリアテキストのパスワード「welcome」を使用して認証されます。

次に、ユーザーの電話番号を取得する例を示します。

```
String[] userAttrList = {"telephonenumber"};
PropertySetCollection result = myuser.getProperties( ctx,userAttrList );
```

関連項目： **User** クラスの他の使用例は、A-13 ページの「[Java サンプル・コード](#)」を参照してください。

Subscriber クラス

Subscriber クラスは、有効な **Oracle** コンテキストを持つサブスクライバを表すために使用されます。**Subscriber** オブジェクトは、識別名、GUID または単純な名前を使用して作成できます。単純な名前を使用すると、ルート of **Oracle** コンテキストの追加情報がユーザーの識別に使用されます。デフォルトの **Subscriber** オブジェクトの作成もサポートされます。デフォルト・サブスクライバに関連する情報は、ルート of **Oracle** コンテキストに格納されています。次に、サブスクライバの構成例を示します。

```
Subscriber mysub = new Subscriber( ctx, //a valid InitialDirContext
                                Util.IDTYPE_DN,
                                "o=oracle,dc=com",
                                false );
```

この例の **mysub** は、その識別名（"o=oracle,dc=com"）を使用して定義されます。

共通 **Subscriber** オブジェクトを使用すると、サブスクライバ・プロパティを取得できます。たとえば、次のようにします。

```
String[] attrList = { "cn", "orclguid" };
PropertySetCollection result= mysub.getProperties(ctx,attrList);
// do work with result
```

Subscriber オブジェクトは、**User** オブジェクトの構成時にサブスクライバを識別するためにも使用できます。次に、単純な名前「user1」を持つ **User** オブジェクトを、前述の例で作成されたサブスクライバの下に作成する例を示します。

```
myuser1 = new User ( ctx, //a valid InitialDirContext
                    Util.IDTYPE_SIMPLE,
                    "user1",
                    mysub,
                    false );
```

関連項目： Subscriber クラスの他の使用例は、A-13 ページの「[Java サンプル・コード](#)」を参照してください。

Group クラス

Group クラスは、有効なグループ・エントリを表すために使用されます。Group オブジェクトは、その識別名または GUID を使用して作成できます。次に、グループの構成例を示します。

```
Group mygroup = new Group ( Util.IDTYPE_DN,
                            "cn=group1,cn=Groups,o=oracle,dc=com" );
```

この例の mygroup は、その識別名を使用して定義されます。

Group オブジェクトの使用例としては、グループ・プロパティの取得があります。たとえば、次のようにします。

```
PropertySetCollection result = mygroup.getProperties( ctx, null );
```

Group クラスは、メンバーシップに関連する機能もサポートしています。ismember() メソッドを使用すると、ある User オブジェクトがグループのダイレクト・メンバーか、ネストされたメンバーであるかを確認できます。たとえば、次のようにします。

```
if (mygroup.isMember( ctx, // a valid InitialDirContext
                    myuser,
                    true ) ) { // set to true for nested member
    // do work
}
```

myuser は User オブジェクトです。3 番目の引数は、ネストされたメンバーシップが考慮されていることを示すために、true に設定されています。false に設定すると、ダイレクト・メンバーシップのみが考慮されます。

Util.getGroupMembership() を使用して、特定のユーザーが属しているグループのリストを取得することもできます。たとえば、次のようにします。

```
PropertySetCollection result = Util.getGroupMembership( ctx,
                                                         myuser,
                                                         new String[0],
                                                         true );
```

関連項目： Group クラスの他の使用例は、A-13 ページの「[Java サンプル・コード](#)」を参照してください。

PropertySetCollection クラス、PropertySet クラスおよび Property クラス

User クラス、Subscriber クラスおよび Group クラスのほとんどのメソッドは、PropertySetCollection オブジェクトを戻します。このオブジェクトは結果の集合を表します。このオブジェクトは、1 つ以上の Lightweight Directory Access Protocol (LDAP) エントリの集合です。各エントリは PropertySet で表され、識別名で識別されます。PropertySet には、Property でそれぞれを表す属性を含めることができます。Property とは、その Property が表す特定の属性に関する 1 つ以上の値の集合です。次に、これらのクラスの使用例を示します。

```
PropertySetCollection psc = Util.getGroupMembership( ctx,
                                                    myuser,
                                                    null,
                                                    true );

// for loop to go through each PropertySet
for (int i = 0; i < psc.size(); i++ ) {

    PropertySet ps = psc.getPropertySet(i);

    // Print the DN of each PropertySet
    System.out.println("dn:  " + ps .getDN());

    // Get the values for the "objectclass" Property
    Property objectclass = ps.getProperty( "objectclass" );

    // for loop to go through each value of Property "objectclass"
    for (int j = 0; j< objectclass.size(); j++) {

        // Print each "objectclass" value
        System.out.println("objectclass:  " + objectclass.getValue(j));
    }
}
```

myuser は User オブジェクトです。psc には、myuser が属するネストされたグループがすべて含まれます。このコードは結果エントリをループし、各エントリの「objectclass」値をすべて出力します。

関連項目： PropertySetCollection クラス、PropertySet クラスおよび Property クラスの他の使用例は、A-13 ページの「[Java サンプル・コード](#)」を参照してください。

PropertySetCollection クラス、PropertySet クラスおよび Property クラスの使用例は、付録のサンプル・プログラムを参照してください。

クラス

この項では、クラスについて説明します。

この項では、次の項目について説明します。

- [oracle.ldap.util.Base64](#)
- [oracle.ldap.util.Group](#)
- [oracle.ldap.util.Guid](#)
- [oracle.ldap.util.LDIF](#)
- [oracle.ldap.util.LDIFAttribute](#)
- [oracle.ldap.util.LDIFAttribute](#)
- [oracle.ldap.util.LDIFMigration](#)
- [oracle.ldap.util.LDIFReader](#)
- [oracle.ldap.util.LDIFRecord](#)
- [oracle.ldap.util.LDIFSubstitute](#)
- [oracle.ldap.util.LDIFWriter](#)
- [oracle.ldap.util.Property](#)
- [oracle.ldap.util.PropertySet](#)
- [oracle.ldap.util.PropertySetCollection](#)
- [oracle.ldap.util.Subscriber](#)
- [oracle.ldap.util.User](#)
- [oracle.ldap.util.Util](#)

oracle.ldap.util.Base64

構文:

```
public class oracle.ldap.util.Base64
```

説明:

実際のバイトから BASE64 でエンコードされたバイトへのエンコーディング、および BASE64 でエンコードされたバイトから実際のバイトへのデコーディングを提供します。

コンストラクタ

Base64()**構文:**

```
public Base64()
```

メソッド

encode(String inStr)**構文:**

```
public static java.lang.String encode(String inStr)
```

説明:

このメソッドを使用して、文字列を BASE64 でエンコードされた文字列に変換します。

パラメータ:

inStr – BASE64 でエンコードする文字列。

戻り値:

BASE64 でエンコードされた文字列。

decode(String inStr)**構文:**

```
public static java.lang.String decode(String inStr)
```

説明:

このメソッドを使用して、BASE64 でエンコードされた文字列を元の文字列にデコードします。

パラメータ：

inStr — BASE64 でエンコードされた文字列。

戻り値：

元の文字列。

encode(byte[] inBytes)

構文：

```
public static byte[] encode(byte[] inBytes)
```

説明：

渡されたデータ配列を表す、BASE64 でエンコードされた文字の配列を戻します。

パラメータ：

inBytes —エンコードするバイトの配列。

戻り値：

BASE64 でエンコードされたバイト配列。

decode(byte[] inBytes)

構文：

```
public static byte[] decode(byte[] inBytes)
```

説明：

BASE64 でエンコードされた、連続するバイトをデコードします。入力内の無効な記号はすべて無視されます (CRLF、空白)。

パラメータ：

inBytes — BASE64 でエンコードされた連続するバイト。

戻り値：

入力された BASE64 の元のデータ。

oracle.ldap.util.Group

構文：

```
public class oracle.ldap.util.Group
```

コンストラクタ

Group(int inGroupIdType, String inGroupIdName)

構文：

```
public Group(int inGroupIdType, String inGroupIdName)
```

説明：

グループ ID とそのタイプを使用して、グループを構成します。

パラメータ：

inGroupIdType —使用するグループ ID のタイプで、Util.IDTYPE_DN または Util.IDTYPE_GUID。

inGroupIdName —グループ ID。

メソッド

oracle.ldap.util.PropertySetCollection getProperties(DirContext ctx, String[] attrList)

構文：

```
public oracle.ldap.util.PropertySetCollection getProperties(DirContext ctx, String[] attrList)
```

説明：

このグループに関連する、選択された属性を取得します。

パラメータ：

ctx —有効な DirContext。

attrList —取得する属性の配列。NULL は、すべての属性を取得する必要があることを意味します。空の配列は、何も取得しないことを意味します。

戻り値：

結果の PropertySetCollection。

resolve(DirContext ctx)

構文：

```
public void resolve(DirContext ctx)
```

説明：

グループを、その識別名を識別して検証します。

パラメータ：

ctx ー有効な DirContext。

getDn(DirContext ctx)

構文：

```
public java.lang.String getDn(DirContext ctx)
```

説明：

このグループの識別名を戻します。

パラメータ：

ctx ー有効な DirContext。

戻り値：

このグループの識別名。

boolean isMember(DirContext ctx, User user, boolean nested)

構文：

```
public boolean isMember(DirContext ctx, User user, boolean nested)
```

説明：

特定のユーザーが、このグループのメンバーであるかどうかをチェックします。

パラメータ：

ctx ー有効な DirContext。

user ー有効な User オブジェクト。

nested ーネストされたメンバーシップを許可する場合は **true** に設定します。それ以外の場合は、ダイレクト・メンバーシップのみが考慮されます。

戻り値：

指定したユーザーがこのグループのメンバーである場合は **true**、それ以外の場合は **false**。

oracle.ldap.util.Guid

構文:

```
public final class oracle.ldap.util.Guid implements java.lang.Cloneable
```

説明:

このクラスは、GUID またはオブジェクト ID を表します。これは不変のクラスです。

コンストラクタ

Guid()**構文:**

```
public Guid()
```

説明:

デフォルト・コンストラクタです。

Guid(String guid)**構文:**

```
Guid(String guid)
```

説明:

文字列から GUID を構成します。

パラメータ:

guid — GUID の文字列表現。

Guid(byte[] byte_array)**構文:**

```
public Guid(byte[] byte_array)
```

説明:

バイト配列から GUID を構成します。

パラメータ:

byte_array — GUID を表すバイトの配列。このコンストラクタでは、最初にバイト配列の長さが検証されます。

メソッド

newInstance()

構文:

```
public static oracle.ldap.util.Guid newInstance()
```

説明:

新規 GUID を生成します。

戻り値:

GUID クラスの新規インスタンス。

getBytes()

構文:

```
public byte[] getBytes()
```

説明:

GUID のバイト・フォームを戻します。

戻り値:

GUID のバイト・フォームを戻します。

toString()

構文:

```
public final java.lang.String toString()
```

説明:

GUID を文字列フォーマットで取得します。

戻り値:

文字列フォーマットの GUID。

equals (Object o)

構文:

```
public boolean equals(Object o)
```

説明:

GUID を文字列フォーマットで比較します。

パラメータ：

o – GUID オブジェクト。

戻り値：

等しい場合は `true`、それ以外の場合は `false`。

hashCode()

構文：

```
public int hashCode()
```

説明：

ハッシュ用に、このオブジェクトのハッシュ・コードを戻します。

戻り値：

GUID の `int` 型ハッシュ・コード。

Object clone()

構文：

```
public java.lang.Object clone()
```

説明：

GUID オブジェクトのクローンを作成します。

戻り値：

既存の GUID オブジェクトのクローン。

フィールド

GUID_BYTE_SIZE

構文：

```
public static final GUID_BYTE_SIZE
```

説明：

GUID に必要なバイト数。

GUID_STRING_SIZE

構文:

```
public static final GUID_STRING_SIZE
```

説明:

GUID の文字列表現に必要なバイト数。

oracle.ldap.util.LDIF

構文:

```
public class oracle.ldap.util.LDIF
```

説明:

LDAP データ交換フォーマットに関する最も一般的な事項を定義するクラス。

フィールド

RECORD_CHANGE_TYPE_ADD

構文:

```
static final RECORD_CHANGE_TYPE_ADD
```

説明:

レコード変更タイプー add

RECORD_CHANGE_TYPE_DELETE

構文:

```
static final RECORD_CHANGE_TYPE_DELETE
```

説明:

レコード変更タイプー delete

RECORD_CHANGE_TYPE_MODIFY

構文:

```
static final RECORD_CHANGE_TYPE_MODIFY
```

説明:

レコード変更タイプー modify

RECORD_CHANGE_TYPE_MODDN**構文:**

```
static final RECORD_CHANGE_TYPE_MODDN
```

説明:

レコード変更タイプー moddn

ATTRIBUTE_CHANGE_TYPE_ADD**構文:**

```
static final ATTRIBUTE_CHANGE_TYPE_ADD
```

説明:

属性変更タイプー add

ATTRIBUTE_CHANGE_TYPE_DELETE**構文:**

```
static final ATTRIBUTE_CHANGE_TYPE_DELETE
```

説明:

属性変更タイプー delete

ATTRIBUTE_CHANGE_TYPE_REPLACE**構文:**

```
static final ATTRIBUTE_CHANGE_TYPE_REPLACE
```

説明:

属性変更タイプー replace

oracle.ldap.util.LDIFAttribute

構文：

```
public class oracle.ldap.util.LDIFAttribute
```

説明：

LDIF 属性クラスは、属性の名前と値を表します。ディレクトリ・エントリに対して追加または削除する属性、あるいはディレクトリ・エントリ内で変更する属性の指定に使用します。この属性クラスは、ディレクトリの検索でも戻されます。

コンストラクタ

LDIFAttribute(String attrName)

構文：

```
public LDIFAttribute(String attrName)
```

説明：

属性を値なしで構成します。

パラメータ：

attrName — 属性の名前。

LDIFAttribute(LDIFAttribute ldapAttribute)

構文：

```
public LDIFAttribute(LDIFAttribute ldapAttribute)
```

説明：

入力された LDIFAttribute のすべての値のコピーを使用して属性を構成します。

パラメータ：

ldapAttribute — テンプレートとして使用する属性。

LDIFAttribute(String attrName, byte[] attrBytes)

構文：

```
public LDIFAttribute(String attrName, byte[] attrBytes)
```

説明：

バイト・フォーマットの値を使用して属性を構成します。

パラメータ :

attrName —属性の名前。

attrBytes —実際のバイトとしての属性の値。

LDIFAttribute(String attrName, String attrString)

構文 :

```
public LDIFAttribute(String attrName, String attrString)
```

説明 :

単一の文字列値を保持する属性を構成します。

パラメータ :

attrName —属性の名前。

attrString —文字列フォーマットでの属性の値。

LDIFAttribute(String attrName, String[] attrStrings)

構文 :

```
public LDIFAttribute(String attrName, String[] attrStrings)
```

説明 :

文字列値の配列を保持する属性を構成します。

パラメータ :

attrName —属性の名前。

attrStrings —この属性に対する文字列値のリスト。

メソッド

addValue(String attrString)

構文 :

```
public void addValue(String attrString)
```

説明 :

文字列値を属性に追加します。

パラメータ :

attrString —文字列での属性の値。

addValue(byte[] attrBytes)

構文：

```
public synchronized void addValue(byte[] attrBytes)
```

説明：

バイト・フォーマットの値を属性に追加します。

パラメータ：

attrBytes — 属性に追加する実際のバイトでの属性値。

addValue(String[] attrValues)

構文：

```
public synchronized void addValue(String[] attrValues)
```

説明：

文字列値の配列を属性に追加します。

パラメータ：

attrValues — 属性に追加する文字列値の配列。

getByteValues()

構文：

```
public java.util.Enumeration getByteValues()
```

説明：

属性の値に対する列挙子を `byte[]` フォーマットで戻します。

戻り値：

一連の属性値。列挙子の各要素は、バイト型です。

getStringValues()

構文：

```
public java.util.Enumeration getStringValues()
```

説明：

属性の文字列値に対する列挙子を戻します。

戻り値：

文字列値の列挙子。

getBytesValueArray()**構文：**

```
public byte[] getBytesValueArray()
```

説明：

属性の値を `byte[]` の配列として戻します。

戻り値：

属性値のバイト・フォーマットでの配列。

getStringValueArray()**構文：**

```
public java.lang.String[] getStringValueArray()
```

説明：

属性の値を文字列の配列として戻します。

戻り値：

属性値の文字列オブジェクトとしての配列。

setValues(String[] attrValues)**構文：**

```
public void setValues(String[] attrValues)
```

説明：

文字列値を属性値として設定します。

パラメータ：

`attrValues` — 属性値を表す文字列値の配列。

getLangSubtype()**構文：**

```
public java.lang.String getLangSubtype()
```

説明：

言語サブタイプがある場合は、言語サブタイプを戻します。たとえば、属性名が `cn;lang-fr;phonetic` の場合、このメソッドは `lang-fr` 文字列を戻します。

戻り値：

言語サブタイプまたは NULL（名前に言語サブタイプがない場合）。

getBaseName(String attrName)**構文：**

```
public static java.lang.String getBaseName(String attrName)
```

説明：

ベース名を戻します。たとえば、属性名が `cn;lang-fr;phonetic` の場合、このメソッドは `cn` を戻します。

パラメータ：

`attrName` —ベース名を抽出する属性の名前。

戻り値：

ベース名（サブタイプのない属性名など）。

getBaseName()**構文：**

```
public java.lang.String getBaseName()
```

説明：

このオブジェクトのベース名を戻します。たとえば、属性名が `cn;lang-fr;phonetic` の場合、このメソッドは `cn` を戻します。

戻り値：

ベース名（サブタイプのない属性名など）。

getName()**構文：**

```
public java.lang.String getName()
```

説明：

属性の名前を戻します。

戻り値：

属性名。

getSubtypes(String attrName)

構文：

```
public static java.lang.String[] getSubtypes(String attrName)
```

説明：

指定した属性名からサブタイプを抽出します。たとえば、属性名が `cn;lang-fr;phonetic` の場合、このメソッドは `lang-fr` および `phonetic` が含まれる配列を返します。

パラメータ：

`attrName` — サブタイプを抽出する属性の名前。

戻り値：

サブタイプの配列または `NULL`（サブタイプがない場合）。

getSubtypes()

構文：

```
public java.lang.String[] getSubtypes()
```

説明：

このオブジェクトの属性名からサブタイプを抽出します。たとえば、属性名が `cn;lang-fr;phonetic` の場合、このメソッドは `lang-ja` および `phonetic` が含まれる配列を返します。

戻り値：

サブタイプの配列または `NULL`（サブタイプがない場合）。

hasSubtype(String subtype)

構文：

```
public boolean hasSubtype(String subtype)
```

説明：

指定したサブタイプが属性名に含まれているかどうかをレポートします。たとえば、サブタイプ `lang-fr` をチェックして、属性名が `cn;lang-fr` の場合、このメソッドは `true` を返します。

パラメータ :

subtype —チェックする単一のサブタイプ。

戻り値 :

属性名に指定したサブタイプが含まれている場合は **true**。

hasSubtypes(String[] subtypes)

構文 :

```
public boolean hasSubtypes (String[] subtypes)
```

説明 :

指定したすべてのサブタイプが属性名に含まれているかどうかをレポートします。たとえば、サブタイプの lang-fr および **phonetic** をチェックして、属性名が cn;lang-fr;phonetic の場合、このメソッドは **true** を戻します。属性名が cn;phonetic または cn;lang-fr の場合、このメソッドは **false** を戻します。

パラメータ :

subtypes —チェックするサブタイプの配列。

戻り値 :

指定したすべてのサブタイプが属性名に含まれている場合は **true**。

removeValue(String attrString)

構文 :

```
public synchronized void removeValue (String attrString)
```

説明 :

文字列値を属性から削除します。

パラメータ :

attrString —削除する文字列値。

removeValue(byte[] attrBytes)

構文 :

```
public void removeValue (byte[] attrBytes)
```

説明 :

バイト・フォーマットの値を属性から削除します。

パラメータ :

attrBytes - 削除するバイト・フォーマットの値。

size()

構文 :

```
public int size()
```

説明 :

属性の値の数を戻します。

戻り値 :

この属性の値の数。

getChangeType()

構文 :

```
public int getChangeType()
```

説明 :

この属性に関連付けられている変更タイプがある場合、その変更タイプを戻します。

戻り値 :

LDIF クラスに定義されている変更タイプ定数。

setChangeType(int changeType)

構文 :

```
public void setChangeType(int changeType)
```

説明 :

この属性の変更タイプを設定します。

パラメータ :

changeType - LDIF クラスに定義されている変更タイプ定数。

getValue()

構文 :

```
public java.lang.String getValue()
```

説明：

単一値属性の値を返します。属性に複数の値がある場合は、最初の値が返ります。属性に値がない場合は、NULL が返ります。

戻り値：

属性値、あるいは属性に値がない場合は NULL。

contains(String attrString)**構文：**

```
public boolean contains(String attrString)
```

説明：

指定した属性値がこのオブジェクトに含まれているかどうかをレポートします。

パラメータ：

attrString ーチェックが必要な文字列オブジェクトとしての値。

戻り値：

指定した値が属性に含まれている場合は **true**、含まれていない場合は **false**。

contains(byte[] attrBytes)**構文：**

```
public boolean contains(byte[] attrBytes)
```

説明：

指定した属性値がこのオブジェクトに含まれているかどうかをレポートします。

パラメータ：

attrBytes ーチェックが必要なバイト・フォーマット表現としての値。

戻り値：

指定した値が属性に含まれている場合は **true**、含まれていない場合は **false**。

toString()**構文：**

```
public java.lang.String toString()
```

説明：

LDAP エントリ内の属性の文字列表現を取得します。たとえば、次のようにします。
LDIFAttribute type='cn', values='Barbara Jensen,Babs Jensen'

戻り値：

属性の文字列表現。

equals(Object ldifAttr)**構文：**

```
public boolean equals(Object ldifAttr)
```

oracle.ldap.util.LDIFMigration

構文：

```
public class oracle.ldap.util.LDIFMigration
```

説明：

このクラスは、コンポーネント固有のリポジトリにあるユーザー情報を OID に移行するメソッドを提供します。この移行プロセスへの入力、中間 LDAP データ交換フォーマット (LDIF) ファイルです。このファイルには置換が必要な置換変数が含まれています。このプロセスの出力は LDIF ファイルです。このファイルは、既存のツールの 1 つを使用してデータをアップロードするために使用できます。

コンストラクタ

LDIFMigration(String inputFile, Vector subsVect, String outFile)**構文：**

```
public LDIFMigration(String inputFile, Vector subsVect, String outFile)
```

説明：

このメソッドは、LDIF ファイルを読み込んで置換し、LDIF エントリをファイルに書き込むためのオブジェクトを構成します。

パラメータ：

inputFile — 入力ファイルの名前。

subsVect — 置換変数または値が含まれるベクター。

outFile — 出力ファイルの名前。

例外:

MigrationException – I/O エラーや無効な入力パラメータが原因で発生する可能性がある移行エラー。エラーの状況は、エラー・コードおよびこの例外オブジェクトのエラー・メッセージで説明されます。

LDIFMigration(File inpF, Vector subsVect, File outF)

構文:

```
public LDIFMigration(File inpF, Vector subsVect, File outF)
```

説明:

このメソッドは、指定したファイル・オブジェクトから LDIF エントリを読み込んで置換し、指定したファイル・オブジェクトに書き込むためのオブジェクトを構成します。

パラメータ:

inpF – 入力ファイル・オブジェクト。

subsVect – 置換変数または値が含まれるベクター。

outF – 出力ファイル・オブジェクト。

例外:

MigrationException – I/O エラーや無効な入力パラメータが原因で発生する可能性がある移行エラー。エラーの状況は、エラー・コードおよびこの例外オブジェクトのエラー・メッセージで説明されます。

LDIFMigration(InputStream inpS, Vector subsVect, OutputStream outS)

構文:

```
public LDIFMigration(InputStream inpS, Vector subsVect, OutputStream outS)
```

説明:

このメソッドは、指定した入力ストリームから LDIF エントリを読み込んで置換し、指定した出力ストリームに書き込むためのオブジェクトを構成します。

パラメータ:

inpS – LDIF エントリを提供する入力ストリーム。

subsVect – 置換変数または値が含まれるベクター。

outS – LDIF エントリが書き込まれる出力ストリーム。

例外：

`MigrationException` – I/O エラーや無効な入力パラメータが原因で発生する可能性がある移行エラー。エラーの状況は、エラー・コードおよびこの例外オブジェクトのエラー・メッセージで説明されます。

メソッド**setDirContext(DirContext dirCtx).****構文：**

```
public void setDirContext(DirContext dirCtx)
```

説明：

ディレクトリ・コンテキストを設定します。

パラメータ：

`dirCtx` – 置換変数の自動検索のためにディレクトリ属性の間合せが行われるディレクトリ・コンテキスト・オブジェクト。

setUserDN(String dn)**構文：**

```
public void setUserDN(String dn)
```

説明：

現在のユーザー識別名を設定します。

パラメータ：

`dn` – ユーザー・バインドの識別名。

int migrate()**構文：**

```
public int migrate()
```

説明：

このメソッドをコールして、中間 LDIF ファイルを読み込んで置換し、LDIF 出力ファイルに書き込みます。

戻り値：

正常に書き込まれたエントリ数 (`int` 型)。

例外：

MigrationException — LDIF ファイルの読み込みまたは書き込み中にエラーになった場合、この例外が発生します。

int migrate(Subscriber subscriber)

構文：

```
public int migrate(Subscriber subscriber)
```

説明：

このメソッドをコールして、中間 LDIF ファイルを読み込んで置換し、LDIF 出力ファイルに書き込みます。置換変数は、ディレクトリ・サーバーへの接続によって自動的に算出されます。指定した LDAP ホストから決定される置換変数は、`s_SubscriberDN`、`s_UserContainerDN`、`s_GroupContainerDN`、`s_SubscriberOracleContextDN`、`s_RootOracleContextDN` です。

パラメータ：

`subscriber` — 置換変数の算出が必要なサブスクリイバ。

戻り値：

正常に書き込まれたエントリ数 (`int` 型)。

例外：

MigrationException — LDIF ファイルの読み込みまたは書き込み中にエラーになった場合、あるいはディレクトリ操作の実行中に **NamingException** が発生した場合は、この例外が発生します。

cleanup()

構文：

```
public void cleanup()
```

説明：

LDIF の入力ストリームおよび出力ストリームをクローズします。

例外：

MigrationException — 入力ストリームまたは出力ストリームのクローズ中に I/O エラーになった場合は、この例外が発生します。

oracle.ldap.util.LDIFReader

構文:

```
public class oracle.ldap.util.LDIFReader
```

説明:

LDIF は、ディレクトリ・エントリを表すために使用されるファイル形式です。ディレクトリのデータは、この形式でエクスポートして別のディレクトリ・サーバーにインポートできます。LDIF データには、ディレクトリのデータに適用する必要がある一連の変更を記述できます。この形式は、インターネット規約「The LDAP Data Interchange Format (LDIF) - RFC-2849」で説明されています。

このクラスには、LDIF ファイルを読み込んで操作するための一連のメソッドがあります。

コンストラクタ

LDIFReader()**構文:**

```
public LDIFReader()
```

説明:

デフォルト・コンストラクタは、標準入力からデータを読み込みます。入力データは UTF8 フォーマットである必要があります。

例外:

IOException – I/O エラーの場合は、この例外が発生します。

LDIFReader(String file)**構文:**

```
public LDIFReader(String file)
```

説明:

指定したファイルから LDIF データを読み込むための入力ストリームを構成します。

パラメータ:

file – LDIF ファイルの名前。

例外:

IOException – I/O エラーの場合は、この例外が発生します。

LDIFReader(File fileObj)

構文：

```
public LDIFReader(File fileObj)
```

説明：

指定したファイル・オブジェクトから LDIF データを読み込むための入力ストリームを構成します。

パラメータ：

fileObj – LDIF ファイルのファイル・オブジェクト。

例外：

IOException – I/O エラーの場合は、この例外が発生します。

LDIFReader(InputStream ds)

構文：

```
public LDIFReader(InputStream ds)
```

説明：

指定した入力ストリームから LDIF データを読み込むための入力ストリームを構成します。

パラメータ：

ds – LDIF データを提供する入力ストリーム。

例外：

IOException – I/O エラーの場合は、この例外が発生します。

メソッド

nextEntry()

構文：

```
public java.util.Vector nextEntry()
```

説明：

LDIF ファイルの次のエントリを戻します。このメソッドを使用して、LDIF ファイル内の全エントリを反復して取得できます。

戻り値：

名前と値のペアで属性を格納しているベクター・オブジェクトとしての次のエントリ。たとえば、ベクター内の各要素は名前：値のようになります。エントリがない場合は `NULL` が戻されます。

例外：

`IOException` – I/O エラーの場合は、この例外が発生します。

`nextRecord()`**構文：**

```
public synchronized oracle.ldap.util.LDIFRecord nextRecord()
```

説明：

LDIF ファイルの次のレコードを戻します。このメソッドを使用して、LDIF ファイル内の全エントリを反復して取得できます。

戻り値：

`LDIFRecord` オブジェクトとしての次のエントリ。エントリがない場合は `NULL` が戻されます。

例外：

`IOException` – I/O エラーの場合は、この例外が発生します。

`close()`**構文：**

```
public void close()
```

説明：

ストリームをクローズします。

例外：

`IOException` – エラーの場合は、この例外が発生します。

oracle.ldap.util.LDIFRecord

構文:

```
public class oracle.ldap.util.LDIFRecord implements java.lang.Cloneable
```

説明:

LDIFRecord は、LDIF ファイルの単一のエントリを表し、識別名および 0（ゼロ）以上の属性で構成されます。

コンストラクタ

LDIFRecord()

構文:

```
public LDIFRecord()
```

説明:

識別名および属性値を設定せずに LDIFRecord オブジェクトを構成します。

LDIFRecord(String dn)

構文:

```
public LDIFRecord(String dn)
```

説明:

指定した識別名でレコードを構成します。

パラメータ:

dn — 新規エントリの識別名。

addAttribute(LDIFAttribute atr)

構文:

```
public void addAttribute(LDIFAttribute atr)
```

説明:

属性をこのレコードに追加します。

パラメータ:

atr — 追加する LDIFAttribute オブジェクト。

getDN()**構文:**

```
public java.lang.String getDN()
```

説明:

カレント・レコードの識別名を返します。

戻り値:

カレント・レコードの識別名。

setDN(String dn)**構文:**

```
public void setDN(String dn)
```

説明:

このレコードの識別名を設定します。

パラメータ:

dn — カレント・レコードに設定する識別名。

synchronized oracle.ldap.util.LDIFAttribute getAttribute(String attrName)**構文:**

```
public synchronized oracle.ldap.util.LDIFAttribute getAttribute(String attrName)
```

説明:

指定した属性名の `LDIFAttribute` オブジェクトを返します。

パラメータ:

attrName — 属性の名前。

java.util Enumeration getAll()**構文:**

```
public java.util.Enumeration getAll()
```

説明:

このレコード内の属性の列挙を返します。

戻り値：

LDIFAttribute オブジェクトがすべて含まれている列挙。

synchronized java.util Enumeration getIDs()**構文：**

```
public synchronized java.util Enumeration getIDs()
```

説明：

このレコード内の属性 ID の列挙を文字列オブジェクトで取得します。

戻り値：

このレコード・セットの属性 ID の NULL でない列挙。一連の属性が 0（ゼロ）の場合は、空の列挙が返ります。

int getChangeType()**構文：**

```
public int getChangeType()
```

説明：

このレコードの変更タイプを取得します。変更タイプ定数は、LDIF クラスに定義されています。

int size()**構文：**

```
public int size()
```

説明：

このレコード内の属性数を返します。識別名はこの件数に含まれません。

戻り値：

このレコード内の属性数。

Object clone()**構文：**

```
public java.lang.Object clone()
```

説明：

このオブジェクトのレプリカを作成します。

戻り値：

このオブジェクトのレプリカ。

toString()**構文：**

```
public java.lang.String toString()
```

説明：

このオブジェクトを文字列で表現します。

oracle.ldap.util.LDIFSubstitute

構文：

```
public class oracle.ldap.util.LDIFSubstitute
```

説明：

このクラスは、LDIF エントリ内で置換を実行するための一般的なメソッドを提供します。LDIF エントリは、ベクター・オブジェクトで表現されます。置換変数（名前と値）のペアは、ベクターとして指定できます。

コンストラクタ

LDIFSubstitute()**構文：**

```
public LDIFSubstitute()
```

メソッド

Vector substitute**構文：**

```
public static java.util.Vector substitute(Vector ldifEntry, String srchStr, String repStr)
```

説明：

ベクターに含まれている LDIF エントリの置換変数を検索して置換します。

パラメータ：

ldifEntry —属性として要素を持つ LDIF エントリ。

srchStr —置換変数名。

repStr —置換変数値。

戻り値：

置換適用後のベクター LDIF エントリ。

Vector substitute**構文：**

```
public static java.util.Vector substitute(Vector ldifEntry, Vector sAndRep)
```

説明：

ベクターに含まれている LDIF エントリの置換変数を検索して置換します。

パラメータ：

ldifEntry —属性として要素を持つ LDIF エントリ。

sAndRep —置換変数の名前と値のペアが含まれるベクター。

戻り値：

置換適用後のベクター LDIF エントリ。

oracle.ldap.util.LDIFWriter

構文：

```
public class oracle.ldap.util.LDIFWriter
```

説明：

LDIF は、ディレクトリ・エントリを表すために使用されるファイル形式です。ディレクトリのデータは、この形式でエクスポートして別のディレクトリ・サーバーにインポートできます。ディレクトリ・データの LDAP サーバーからのインポートおよびエクスポートによって、ディレクトリ内のデータに適用できる一連の変更を記述できます。この形式は、インターネット規約「The LDAP Data Interchange Format (LDIF) - RFC-2849」で説明されています。

このクラスは、LDIF エントリをファイルに書き込むための一連のメソッドを提供します。

コンストラクタ

LDIFWriter().

構文:

```
public LDIFWriter()
```

説明:

デフォルト・コンストラクタです。出力ストリームを標準出力に作成します。ファイル名が指定されていないため、LDIF データは標準出力にリダイレクトされます。出力データは UTF8 フォーマットで書き込まれます。属性値の文字数が 1 行の最大長を超える場合、折返しは行われません。

例外:

IOException – I/O エラーが発生しました。

LDIFWriter(String file)

構文:

```
public LDIFWriter(String file)
```

説明:

LDIF データの書き込み用に指定したファイルに出力ストリームを作成します。属性値の文字数が 1 行の最大長を超える場合、折返しは行われません。

パラメータ:

file – LDIF ファイルの名前。

例外:

IOException – I/O エラーが発生しました。

LDIFWriter(File fileObj)

構文:

```
public LDIFWriter(File fileObj)
```

説明:

LDIF データの書き込み用に指定したファイルに出力ストリームを作成します。属性値の文字数が 1 行の最大長を超える場合、折返しは行われません。

パラメータ:

fileObj – LDIF ファイルのファイル・オブジェクト。

例外：

IOException – I/O エラーが発生しました。

LDIFWriter(OutputStream out)**構文：**

```
public LDIFWriter(OutputStream out)
```

説明：

LDIF データの書込み用に指定した出力ストリーム・オブジェクトを使用して出力ストリームを作成します。属性値の文字数が 1 行の最大長を超える場合、折返しは行われません。

パラメータ：

out – LDIF データを書き込む必要があるストリーム。

例外：

IOException – I/O エラーが発生しました。

LDIFWriter(String file, boolean wrap)**構文：**

```
public LDIFWriter(String file, boolean wrap)
```

説明：

LDIF データの書込み用に指定したファイルに出力ストリームを作成します。属性値の文字数が 1 行の最大長を超える場合は、次の行に折り返され、行の先頭には空白が入ります。

パラメータ：

file – LDIF ファイルの名前。

wrap – 属性値の文字数が 1 行の最大長を超える場合は、行が折り返されます。

例外：

IOException – I/O エラーが発生しました。

LDIFWriter(File fileObj, boolean wrap)**構文：**

```
public LDIFWriter(File fileObj, boolean wrap)
```

説明：

LDIF データの書込み用に指定したファイルに出力ストリームを作成します。属性値の文字数が 1 行の最大長を超える場合は、次の行に折り返され、行の先頭には空白が入ります。

パラメータ：

fileObj – LDIF ファイルのファイル・オブジェクト。

wrap – 属性値の文字数が 1 行の最大長を超える場合は、行が折り返されます。

例外：

IOException – I/O エラーが発生しました。

LDIFWriter(OutputStream out, boolean wrap)

構文：

```
public LDIFWriter(OutputStream out, boolean wrap)
```

説明：

LDIF データの書き込み用に指定した出力ストリーム・オブジェクトを使用して出力ストリームを作成します。属性値の文字数が 1 行の最大長を超える場合は、次の行に折り返され、行の先頭には空白が入ります。

パラメータ：

out – LDIF データを書き込む必要があるストリーム。

wrap – 属性値の文字数が 1 行の最大長を超える場合は、行が折り返されます。

例外：

IOException – I/O エラーが発生しました。

メソッド

setMaxLineLen(int maxLineLen)

構文：

```
public void setMaxLineLen(int maxLineLen)
```

説明：

このメソッドを使用して、1 行に書き込むことができる最大文字数を設定します。

パラメータ：

maxLineLen – 1 行の最大文字数。

setWrap(boolean wrap)

構文：

```
public void setWrap(boolean wrap)
```

説明：

このメソッドを使用して、属性値の折返しを行うかどうかを指定します。

パラメータ：

wrap — 属性値の文字数が 1 行の最大長を超える場合は、行が折り返されます。

writeEntry(Vector vEntry)

構文：

```
public void writeEntry(Vector vEntry)
```

説明：

このメソッドを使用して、LDIF エントリをファイルに書き込みます。

パラメータ：

vEntry — 要素として属性名と値が含まれるベクター。

例外：

IOException — I/O エラーの場合は、この例外が発生します。

writeComment(String comment)

構文：

```
public void writeComment(String comment)
```

説明：

このメソッドを使用して、コメント行を LDIF ファイルに追加します。

パラメータ：

comment — LDIF ファイルに追加するコメント文字列。

例外：

IOException — I/O エラーの場合は、この例外が発生します。

synchronized void close()

構文：

```
public synchronized void close()
```

説明：

ストリームをクローズします。

例外：

IOException –エラーの場合は、この例外が発生します。

oracle.ldap.util.Property

構文：

```
public class oracle.ldap.util.Property
```

説明：

このクラスは、PropertySet の特定のプロパティを表します。つまり、エントリの属性セットの特定の属性を表します。

メソッド

int size()

構文：

```
public final int size()
```

説明：

このプロパティのサイズ（戻される属性の値の数）を戻します。

戻り値：

このプロパティに属する値の数を示す int 型。0（ゼロ）が戻される場合もあります。

getName()

構文：

```
public final java.lang.String getName()
```

説明：

このプロパティの名前（このプロパティが表す属性の名前）を戻します。

戻り値：

このプロパティの名前を表す文字列。

Object getValue(int i)

構文：

```
public final java.lang.Object getValue(int i)
```

説明：

このプロパティの *i* 番目の値 (*i* 番目の属性値) を戻します。オブジェクトが戻されます。ユーザーはこのオブジェクトを適切にタイプキャストする必要があります。

パラメータ：

i — 取得する値の索引。

戻り値：

このプロパティの *i* 番目の値。

oracle.ldap.util.PropertySet

構文：

```
public class oracle.ldap.util.PropertySet
```

説明：

このクラスは、PropertySetCollection の特定の PropertySet を表します。つまり、検索結果の集合内の特定の検索結果エントリを表します。

メソッド

isEmpty()**構文：**

```
public final boolean isEmpty()
```

説明：

プロパティ・セットにプロパティが含まれていない場合は **true**、それ以外の場合は **false** を戻します。

戻り値：

プロパティ・セットが空かどうかを示すブール値。

size()**構文：**

```
public final int size()
```

説明：

このプロパティ・セットのサイズ（この特定の検索結果エントリについて戻される属性の数）を戻します。

戻り値：

このプロパティ・セットに属するプロパティの数を示す `int` 型。0（ゼロ）が戻される場合があります。

getAttributeNames()**構文：**

```
public final java.lang.String[] getAttributeNames()
```

説明：

すべてのプロパティの名前が含まれる文字列の配列を返します。すべての属性名は、この特定の検索エントリを使用して戻されます。

戻り値：

すべてのプロパティ名が含まれている文字列の配列。

getProperty(int i)**構文：**

```
public final oracle.ldap.util.Property getProperty(int i)
```

説明：

このプロパティ・セットの `i` 番目のプロパティ（この検索エントリの `i` 番目の属性）を返します。

パラメータ：

`i` — 取得するプロパティの索引。

戻り値：

`i` 番目のプロパティを表すプロパティ。

getProperty(String attrID)**構文：**

```
public final oracle.ldap.util.Property getProperty(String attrID)
```

説明：

`attrID` で識別されるプロパティを返します。`attrID` は属性名です。

パラメータ：

`attrID` — 取得する属性名。

戻り値：

属性名 attrID を持つプロパティ。

getDN()

構文：

```
public final java.lang.String getDN()
```

説明：

プロパティ・セットの名前（このプロパティ・セットで表される検索エントリの識別名）を返します。

戻り値：

プロパティの識別名を表す文字列。

oracle.ldap.util.PropertySetCollection

構文：

```
public class oracle.ldap.util.PropertySetCollection
```

説明：

このクラスは、PropertySet の集合を表します。つまり、指定された検索での一連の検索結果エントリを表します。

メソッド

isEmpty()

構文：

```
public final boolean isEmpty()
```

説明：

PropertySetCollection にプロパティ・セットが含まれていない場合は true、それ以外の場合は false を返します。

戻り値：

プロパティ・セットが空かどうかを示すブール値。

size()**構文:**

```
public final int size()
```

説明:

PropertySetCollection のサイズ（検索結果の検索エントリの数）を戻します。

戻り値:

コレクションのプロパティ・セット数を示す int 型。

getDns()**構文:**

```
public final java.lang.String[] getDns()
```

説明:

すべてのプロパティ・セットの名前が含まれる文字列の配列（この検索結果でのすべての検索エントリの識別名）を戻します。

戻り値:

すべてのプロパティ・セット名が含まれている文字列の配列。

getPropertySet(int i)**構文:**

```
public final oracle.ldap.util.PropertySet getPropertySet(int i)
```

説明:

この PropertySetCollection の i 番目のプロパティ・セット（この検索結果の i 番目の検索エントリ）を戻します。

パラメータ:

int - i ー取得するプロパティ・セットの索引。

戻り値:

i 番目のプロパティ・セットを表す PropertySet。

getPropertySet(String dn)**構文:**

```
public final oracle.ldap.util.PropertySet getPropertySet(String dn)
```

説明：

識別名で識別されるプロパティ・セット（指定した識別名を持つ検索エントリ）を返します。

パラメータ：

dn —取得するプロパティ・セットの識別名。

戻り値：

指定した識別名を持つ PropertySet。

oracle.ldap.util.Subscriber

構文：

```
public class oracle.ldap.util.Subscriber
```

コンストラクタ

Subscriber(DirContext ctx, int inSubIdType, String inSubIdName, boolean validate)

構文：

```
Subscriber(DirContext ctx, int inSubIdType, String inSubIdName, boolean validate)
```

説明：

サブスクライバを構成します。

パラメータ：

ctx —有効な DirContext。

inSubIdType —使用するサブスクライバ ID のタイプ。Util.IDTYPE_DN、Util.IDTYPE_SIMPLE または Util.IDTYPE_GUID のうち 1 つを使用します。

inSubIdName —サブスクライバ ID。この値が NULL で、inSubIdType が Util.IDTYPE_DN の場合は、デフォルト・サブスクライバが使用されます。それ以外の場合は、NULL を指定すると例外が発生します。

validate —コンストラクタでユーザーを検証する場合は、true を設定します。

メソッド

getProperties(DirContext ctx, String[] attrList)

構文：

```
public oracle.ldap.util.PropertySetCollection getProperties(DirContext ctx, String[] attrList)
```

説明：

このサブスライバに関連する、選択された属性を取得します。

パラメータ：

ctx — 有効な DirContext。

attrList — 取得する属性の配列。

getExtendedProperties

構文：

```
public oracle.ldap.util.PropertySetCollection getExtendedProperties(DirContext ctx, int propType, String[] attrList, String filter)
```

説明：

このサブスライバの Oracle コンテキストの下にある拡張プロパティを取得します。現在サポートされているのは、共有プロパティのみです。

パラメータ：

ctx — 有効な DirContext。

propType — EXTPROPTYPE_COMMON、EXTPROPTYPE_RESOURCE_ACCESS_TYPE または EXTPROPTYPE_DEFAULT_RAD のうち 1 つを使用します。

attrList — 取得する属性の配列。

filter — 検索条件を絞り込む検索フィルタ。たとえば、EXTPROPTYPE_RESOURCE_ACCESS_TYPE で (orclResourceTypeName=OracleDB) を使用して、OracleDB のプロパティのみを取得します。

resolve(DirContext ctx)

構文：

```
public void resolve(DirContext ctx)
```

説明：

サブスクライバを、その識別名を識別して検証します。

パラメータ：

ctx —有効な DirContext。

getDN(DirContext ctx)**構文：**

```
public java.lang.String getDN(DirContext ctx)
```

説明：

サブスクライバの識別名を戻します（必要な場合は名前を解決します）。

パラメータ：

ctx —有効な DirContext。

getDn(DirContext ctx)**構文：**

```
public java.lang.String getDn(DirContext ctx)
```

説明：

このサブスクライバの識別名を戻します。

フィールド

EXTPROPTYPE_COMMON**構文：**

```
public static EXTPROPTYPE_COMMON
```

説明：

getExtendedProperties() で使用する拡張プリファレンス・タイプー共有サブスクライバ・プロパティ。

EXTPROPTYPE_RESOURCE_ACCESS_TYPE**構文：**

```
public static EXTPROPTYPE_RESOURCE_ACCESS_TYPE
```

説明：

getExtendedProperties() で使用する拡張プリファレンス・タイプ・リソース・アクセス・タイプ。

EXTPROPTYPE_DEFAULT_RAD**構文：**

```
public static EXTPROPTYPE_DEFAULT_RAD
```

説明：

getExtendedProperties() で使用する拡張プリファレンス・タイプ・デフォルト・ユーザー拡張プロパティ。

oracle.ldap.util.User

構文：

```
public class oracle.ldap.util.User
```

コンストラクタ

User(DirContext ctx, int inUserIdType, String inUserName, subscriber inSubscriber, boolean validate)

構文：

```
public User(DirContext ctx, int inUserIdType, String inUserName, int inSubIdType, String inSubIdName, boolean validate)
```

説明：

ユーザーを構成します。

パラメータ：

ctx —有効な DirContext。

inUserIdType —使用するユーザー ID のタイプ。Util.IDTYPE_DN、Util.IDTYPE_SIMPLE または Util.IDTYPE_GUID のうち 1 つを使用します。

inUserName —ユーザー ID。

inSubscriber —有効な Subscriber オブジェクト。

validate —コンストラクタでユーザーを検証する場合は、true に設定します。

User(DirContext ctx, int inUserIdType, String inUserName, int inSubIdType, String inSubIdName, boolean validate)

構文：

```
public User(DirContext ctx, int inUserIdType, String inUserName, int inSubIdType,
String inSubIdName, boolean validate)
```

説明：

ユーザーを構成します。

パラメータ：

ctx ー有効な DirContext。

inUserIdType ー使用するユーザー ID のタイプ。Util.IDTYPE_DN、Util.IDTYPE_SIMPLE または Util.IDTYPE_GUID のうち 1 つを使用します。

inUserName ーユーザー ID。

inSubIdType ー使用するサブスクライバ ID のタイプ。Util.IDTYPE_DN、Util.IDTYPE_SIMPLE または Util.IDTYPE_GUID のうち 1 つを使用します。

inSubIdName ーサブスクライバ ID。

validate ーコンストラクタでユーザーを検証する場合は、true に設定します。

メソッド

getExtendedProperties

構文：

```
public oracle.ldap.util.PropertySetCollection getExtendedProperties(DirContext ctx,
int propType, String[] attrList, String filter)
```

説明：

指定したプロパティ・タイプに基づいて、このユーザーの拡張プロパティの PropertySetCollection を戻します。

パラメータ：

ctx ー有効な DirContext。

propType ー現在サポートされているのは、EXTPROPTYPE_RESOURCE_ACCESS_DESCRIPTOR のみです。

attrList ー取得する属性の文字列配列。NULL は、すべての属性を取得する必要があることを意味します。空の配列は、何も取得しないことを意味します。

filter ー取得する特定のアプリケーションのプロパティを識別する検索フィルタ。

戻り値：

結果が含まれている `PropertySetCollection`。

`getExtendedProperties`**構文：**

```
public oracle.ldap.util.PropertySetCollection getExtendedProperties(DirContext
ctx,int propType, String[] attrList)
```

説明：

指定したプロパティ・タイプに基づいて、このユーザーのすべての拡張プロパティの `PropertySetCollection` を戻します。

パラメータ：

`ctx` —有効な `DirContext`。

`propType` —現在サポートされているのは、`EXTPROPTYPE_RESOURCE_ACCESS_DESCRIPTOR` のみです。

`attrList` —取得する属性の文字列配列。`NULL` は、すべての属性を取得する必要があることを意味します。空の配列は、何も取得しないことを意味します。

戻り値：

結果が含まれている `PropertySetCollection`。

`getProperties`**構文：**

```
public oracle.ldap.util.PropertySetCollection getProperties(DirContext ctx, String[]
attrList)
```

説明：

このユーザーに関連する、選択された属性を取得します。

パラメータ：

`ctx` —有効な `DirContext`。

`attrList` —取得する属性の配列。`NULL` は、すべての属性を取得する必要があることを意味します。空の配列は、何も取得しないことを意味します。

setProperties

構文：

```
public void setProperties(DirContext ctx, ModificationItem[] mods)
```

パラメータ：

ctx —有効な DirContext。
mods —設定する属性の配列。

resolve(DirContext ctx)

構文：

```
public void resolve(DirContext ctx)
```

説明：

ユーザーを、その識別名を識別して検証します。

パラメータ：

ctx —有効な DirContext。

getDn(DirContext ctx)

構文：

```
public java.lang.String getDn(DirContext ctx)
```

説明：

このユーザーの識別名を戻します。

パラメータ：

ctx —有効な DirContext。

戻り値：

ユーザーの識別名。

locateSubscriber(DirContext ctx)

構文：

```
public java.lang.String locateSubscriber(DirContext ctx)
```

説明：

このユーザーが属するサブスクライバの位置を特定します。

パラメータ：

ctx ー有効な DirContext。

戻り値：

サブスクライバの位置。

authenticateUser**構文：**

```
public void authenticateUser(DirContext ctx, int authType, Object cred)
```

説明：

適切な資格証明を使用してユーザーを認証します。

パラメータ：

ctx ー有効な DirContext。

authType ー現在は、User.CREDTYPE_PASSWD がサポートされています。

cred ー authType に基づく資格証明。

フィールド**CREDTYPE_PASSWD****構文：**

```
public static CREDTYPE_PASSWD
```

説明：

認証にユーザー・パスワードを使用します。

EXTPROPTYPE_RESOURCE_ACCESS_DESCRIPTOR**構文：**

```
public static EXTPROPTYPE_RESOURCE_ACCESS_DESCRIPTOR
```

説明：

getExtendedProperties() で使用する拡張プリファレンス・タイプーリソース・アクセス記述子。

oracle.ldap.util.Util

構文:

```
public class oracle.ldap.util.Util
```

コンストラクタ

Util()

構文:

```
public Util()
```

メソッド

PropertySetCollection getEntryDetails

構文:

```
public static oracle.ldap.util.PropertySetCollection getEntryDetails(DirContext ctx,
String base, String filter, int scope, String[] attrList)
```

setEntryDetails

構文:

```
public static void setEntryDetails(DirContext ctx, String base, ModificationItem[]
mods)
```

authenticateUser

構文:

```
public static void authenticateUser(DirContext ctx, User curUser, int authType,
Object cred)
```

説明:

適切な資格証明を使用してユーザーを認証します。

パラメータ:

ctx — 有効な DirContext。

authType — 現在は、Util.CREDTYPE_PASSWD がサポートされています。

cred — authType に基づく資格証明。

getSubscriberDn**構文：**

```
public static java.lang.String getSubscriberDn(DirContext ctx, String subId, int subIdType)
```

説明：

指定したサブスクライバの識別名を返します。識別名がサブスクライバ ID として使用されている場合、単純に識別名を参照して検証が行われます。

パラメータ：

ctx ー有効な DirContext。

subId ーサブスクライバ ID。

subIdType ー使用するサブスクライバ ID のタイプ。IDTYPE_DN、IDTYPE_SIMPLE または Util.IDTYPE_GUID のうち 1 つを使用します。

getUserDn**構文：**

```
public static java.lang.String getUserDn(DirContext ctx, String userId, int userIdType, String subscriberDN)
```

説明：

指定したユーザーの識別名を返します。識別名がユーザー ID として使用されている場合、単純に識別名を参照して検証が行われます。サブスクライバ識別名が NULL の場合は、デフォルト・サブスクライバが使用されます。

パラメータ：

ctx ー有効な DirContext。

userId ーユーザー ID。

userIdType ー使用するユーザー ID のタイプ。IDTYPE_DN、IDTYPE_SIMPLE または Util.IDTYPE_GUID のうち 1 つを使用します。

subscriberDN ーサブスクライバ識別名。

oracle.ldap.util.PropertySetCollection getGroupMembership**構文：**

```
public static oracle.ldap.util.PropertySetCollection getGroupMembership(DirContext ctx, User curUser, String[] attrList, boolean nested)
```

説明：

ユーザーが直接または間接的に属するグループのリストを返します。

パラメータ：

ctx - 有効な DirContext curUser - 有効な User オブジェクト。

curUser - 有効な User オブジェクト。

attrList - 各グループから取得する属性の文字列配列。

nested - ネストされたメンバーシップを検索する場合は、true に設定します。それ以外の場合は、ダイレクト・メンバーシップのみが返されます。

vector2StrArray(Vector list)**構文：**

```
public static java.lang.String[] vector2StrArray(vector list)
```

normalizeDN(String inDn)**構文：**

```
public static java.lang.String normalizeDN(String inDn)
```

getDASUrl(DirContext ctx, String urlTypeDN)**構文：**

```
public static java.lang.String getDASUrl(DirContext ctx, String urlTypeDN)
```

説明：

urlTypeDN で識別された特定の DAS Universal Resource Locator (URL) を返します。

パラメータ：

ctx - 有効な DirContext。

urlTypeDN - 特定の URL タイプを表す Util.DASURL_* のうち 1 つを使用します。

戻り値：

URL を表す文字列。

java.util.Hashtable getAllDASUrl(DirContext ctx)

構文:

```
public static java.util.Hashtable getAllDASUrl(DirContext ctx)
```

説明:

すべての DAS URL が含まれているハッシュテーブルを返します。URL タイプを識別するキーとして `Util.DASURL_*` を使用して、個々の URL をハッシュテーブルから取得できます。

パラメータ:

ctx — 有効な `DirContext`。

戻り値:

すべての DAS URL が含まれているハッシュテーブル。

メソッド

printResults(PropertySetCollection resultSet)**構文:**

```
public static void printResults(PropertySetCollection resultSet)
```

説明:

`PropertySetCollection` で表されたエントリを LDIF フォーマットで出力します。

パラメータ:

resultSet — 有効な `PropertySetCollection`。

getDefaultSubscriber**構文:**

```
public static java.lang.String[] getDefaultSubscriber()
```

createDefaultSubscriber**構文:**

```
public static void createDefaultSubscriber(DirContext ctx, String ohome, String domain, String subscriber)
```

subAndLoadLdif

構文:

```
public static void subAndLoadLdif(DirContext ctx, String filename, Vector subVector)
```

checkInterfaceVersion(String intVersion)

構文:

```
public static boolean checkInterfaceVersion(String intVersion)
```

説明:

このメソッドは、指定したインタフェースのバージョンが現在の API のバージョンでサポートされているかどうかをチェックします。

パラメータ:

intVersion インタフェースのバージョン。

フィールド

API_VERSION

構文:

```
public static API_VERSION
```

説明:

API のバージョン番号。

INTERFACE_VERSION

構文:

```
public static INTERFACE_VERSION
```

説明:

インタフェースのバージョン番号。

IDTYPE_DN

構文:

```
public static IDTYPE_DN
```

説明:

ID は識別名として使用されます。

IDTYPE_SIMPLE**構文:**

```
public static IDTYPE_SIMPLE
```

説明:

使用される ID は単純な ID です。

IDTYPE_GUID**構文:**

```
public static IDTYPE_GUID
```

説明:

使用される ID は GUID です。

IDTYPE_DEFAULT**構文:**

```
public static IDTYPE_DEFAULT
```

説明:

デフォルト値を使用します。

PROPERTIES_ENTRY**構文:**

```
public static PROPERTIES_ENTRY
```

説明:

ユーザー・エントリのプロパティ。

PROPERTIES_DETACHED**構文:**

```
public static PROPERTIES_DETACHED
```

説明:

ユーザーの連結解除されたプロパティ。

CREDTYPE_PASSWD

構文：

```
public static CREDTYPE_PASSWD
```

説明：

認証にユーザー・パスワードを使用します。

DASURL_BASE

構文：

```
public static DASURL_BASE
```

説明：

DAS URL タイプーベース URL。

DASURL_CREATE_USER

構文：

```
public static DASURL_CREATE_USER
```

説明：

DAS URL タイプーユーザーの作成。

DASURL_EDIT_GROUP

構文：

```
public static DASURL_EDIT_GROUP
```

説明：

DAS URL タイプーグループの編集。

DASURL_EDIT_GROUP_GIVEN_GUID

構文：

```
public static DASURL_EDIT_GROUP_GIVEN_GUID
```

説明：

DAS URL タイプー GUID が指定されているグループの編集。

DASURL_GROUP_SEARCH**構文:**

```
public static DASURL_GROUP_SEARCH
```

説明:

DAS URL タイプグループ検索。

DASURL_EDIT_USER**構文:**

```
public static DASURL_EDIT_USER
```

説明:

DAS URL タイプユーザーの編集。

DASURL_GROUP_LOV**構文:**

```
public static DDASURL_GROUP_LOV
```

説明:

DAS URL タイプグループ LOV。

DASURL_DELETE_USER**構文:**

```
public static DASURL_DELETE_USER
```

説明:

DAS URL タイプユーザーの削除。

DASURL_USER_PRIVILEGE**構文:**

```
public static DASURL_USER_PRIVILEGE
```

説明:

DAS URL タイプユーザー権限。

DASURL_CREATE_GROUP

構文：

```
public static DASURL_CREATE_GROUP
```

説明：

DAS URL タイプグループの作成。

DASURL_USER_SEARCH

構文：

```
public static DASURL_USER_SEARCH
```

説明：

DAS URL タイプユーザーの検索。

DASURL_ACCOUNT_INFO

構文：

```
public static DASURL_ACCOUNT_INFO
```

説明：

DAS URL タイプアカウント情報。

DASURL_EDIT_USER_GIVEN_GUID

構文：

```
public static DASURL_EDIT_USER_GIVEN_GUID
```

説明：

DAS URL タイプ GUID が指定されているユーザーの編集。

DASURL_DELETE_USER_GIVEN_GUID

構文：

```
public static DASURL_DELETE_USER_GIVEN_GUID
```

説明：

DAS URL タイプ GUID が指定されているユーザーの削除。

DASURL_DELETE_GROUP_GIVEN_GUID**構文:**

```
public static DASURL_DELETE_GROUP_GIVEN_GUID
```

説明:

DAS URL タイプー GUID が指定されているグループの削除。

DASURL_GROUP_PRIVILEGE**構文:**

```
public static DASURL_GROUP_PRIVILEGE
```

説明:

DAS URL タイプーグループ権限。

DASURL_USER_PRIVILEGE_GIVEN_GUID**構文:**

```
public static DASURL_USER_PRIVILEGE_GIVEN_GUID
```

説明:

DAS URL タイプー GUID が指定されているユーザー権限。

DASURL_PASSWORD_CHANGE**構文:**

```
public static DASURL_PASSWORD_CHANGE
```

説明:

DAS URL タイプーパスワードの変更。

DASURL_USER_LOV**構文:**

```
public static DASURL_USER_LOV
```

説明:

DAS URL タイプーパスワードの変更。

DASURL_GROUP_PRIVILEGE_GIVEN_GUID

構文：

```
public static DASURL_GROUP_PRIVILEGE_GIVEN_GUID
```

説明：

DAS URL タイプー GUID が指定されているグループ権限。

DASURL_DELETE_GROUP

構文：

```
public static DASURL_DELETE_GROUP
```

説明：

DAS URL タイプーグループの削除。

DASURL_CREATE_RESOURCE

構文：

```
public static DASURL_CREATE_RESOURCE
```

説明：

DAS URL タイプーリソースの作成。

oracle.ldap.util.jndi.ConnectionUtil

構文：

```
public class oracle.ldap.util.jndi.ConnectionUtil
```

コンストラクタ

ConnectionUtil()

構文：

```
public ConnectionUtil()
```

メソッド

discoverSSLPort

構文：

```
public static java.lang.String discoverSSLPort(String host, String port, String bindDn, String bindPwd)
```

説明：

このメソッドは、提供された接続情報を使用して既存の非 SSL OID に接続し、デフォルトの configset 1 にある SSL 接続情報を取得します。この時点で SSL OID サーバーが実行されている必要はありません。

パラメータ：

host — 非 SSL OID を実行しているホスト名。

port — 非 SSL OID を実行しているポート番号。

bindDN — バインド識別名 (cn=orcladmin など)。

bindPwd — バインド・パスワード。

getDefaultDirCtx

構文：

```
public static javax.naming.directory.InitialDirContext getDefaultDirCtx(String host, String port, String bindDN, String bindPwd)
```

説明：

提供された接続情報を使用して `InitialDirContext` を戻します。対応する非 SSL OID サーバーが実行されている必要があります。SSL 接続の場合は、かわりに `getSSLDirCtx` を使用してください。

パラメータ：

host — 非 SSL OID を実行しているホスト名。

port — 非 SSL OID を実行しているポート番号。

bindDN — バインド識別名 (cn=orcladmin など)。

bindPwd — バインド・パスワード。

getSSLDIRCtx

構文：

```
public static javax.naming.directory.InitialDirContext getSSLDIRCtx(String host,  
String port, String bindDN, String bindPwd)
```

説明：

提供された接続情報を使用して **InitialDirContext** を戻します。SSL 接続の場合にのみ、このメソッドを使用します。対応する SSL OID サーバーが実行されている必要があります。

パラメータ：

host — 非 SSL OID を実行しているホスト名。

port — 非 SSL OID を実行しているポート番号。

bindDN — バインド識別名 (cn=orcladmin など)。

bindPwd — バインド・パスワード。

例外

この項では、例外について説明します。

この項では、次の項目について説明します。

- [oracle.ldap.util.AcctTotallyLockedException](#)
- [oracle.ldap.util.AuthFailureException](#)
- [oracle.ldap.util.AuthPasswdChangeWarningException](#)
- [oracle.ldap.util.AuthPasswdExpiredException](#)
- [oracle.ldap.util.GeneralErrorException](#)
- [oracle.ldap.util.InvalidLDIFRecordException](#)
- [oracle.ldap.util.InvalidParameterException](#)
- [oracle.ldap.util.InvalidRootOrclctxException](#)
- [oracle.ldap.util.InvalidSubscriberOrclctxException](#)
- [oracle.ldap.util.LoginPolicyFailureException](#)
- [oracle.ldap.util.MigrationException](#)
- [oracle.ldap.util.MultipleSubscriberException](#)
- [oracle.ldap.util.MultipleUserException](#)
- [oracle.ldap.util.NoGroupMembersException](#)

- [oracle.ldap.util.NoRootOrclctxException](#)
- [oracle.ldap.util.NoSubscriberOrclctxException](#)
- [oracle.ldap.util.NoSuchGroupException](#)
- [oracle.ldap.util.NoSuchSubscriberException](#)
- [oracle.ldap.util.NoSuchUserException](#)
- [oracle.ldap.util.ParameterException](#)
- [oracle.ldap.util.PasswdExpiredException](#)
- [oracle.ldap.util.SetPropertiesException](#)
- [oracle.ldap.util.SubscriberNotFoundException](#)
- [oracle.ldap.util.UtilException](#)

oracle.ldap.util.AcctIPLockedException

構文:

```
public class oracle.ldap.util.AcctIPLockedException extends  
oracle.ldap.util.UtilException
```

コンストラクタ

AcctIPLockedException()

構文:

```
public AcctIPLockedException()
```

AcctIPLockedException(String s)

構文:

```
public AcctIPLockedException(String s)
```

oracle.ldap.util.AcctTotallyLockedException

構文:

```
public class oracle.ldap.util.AcctTotallyLockedException extends  
oracle.ldap.util.UtilException
```

コンストラクタ

AcctTotallyLockedException()

構文:

```
public AcctTotallyLockedException()
```

AcctTotallyLockedException(String s)

構文:

```
public AcctTotallyLockedException(String s)
```

oracle.ldap.util.AuthFailureException

構文:

```
public class oracle.ldap.util.AuthFailureException extends  
oracle.ldap.util.UtilException
```

oracle.ldap.util.AuthPasswdChangeWarningException

構文:

```
public class oracle.ldap.util.AuthPasswdChangeWarningException extends  
oracle.ldap.util.UtilException
```

コンストラクタ

AuthPasswdChangeWarningException()

構文:

```
public AuthPasswdChangeWarningException()
```

AuthPasswdChangeWarningException(String s)

構文:

```
public AuthPasswdChangeWarningException(String s)
```

oracle.ldap.util.AuthPasswdExpiredException

```
public class oracle.ldap.util.AuthPasswdExpiredException extends  
oracle.ldap.util.UtilException
```

コンストラクタ**AuthPasswdExpiredException()**

構文:

```
AuthPasswdExpiredException()
```

AuthPasswdExpiredException(String s)

構文:

```
AuthPasswdExpiredException(String s)
```

oracle.ldap.util.GeneralErrorException

構文:

```
public class oracle.ldap.util.GeneralErrorException extends  
oracle.ldap.util.UtilException
```

説明:

一般エラーが発生すると、この例外がスローされます。

コンストラクタ**GeneralErrorException()**

構文:

```
public GeneralErrorException()
```

説明：

詳細メッセージなしで `GeneralErrorException` を構成します。

`GeneralErrorException(String s)`

構文：

```
public GeneralErrorException(String s)
```

説明：

指定した詳細メッセージ付きで `GeneralErrorException` を構成します。

パラメータ：

s — 詳細メッセージ。

`oracle.ldap.util.InvalidLDIFRecordException`

構文：

```
public class oracle.ldap.util.InvalidLDIFRecordException extends  
java.lang.RuntimeException
```

説明：

LDIF レコードの解析中にエラーが発生すると、このクラスのオブジェクトがスローされます。

コンストラクタ

`InvalidLDIFRecordException()`

構文：

```
public InvalidLDIFRecordException()
```

説明：

詳細メッセージなしで `InvalidLDIFRecordException` を構成します。

`InvalidLDIFRecordException(int lineNo, String s)`

構文：

```
public InvalidLDIFRecordException(int lineNo, String s)
```

説明：

指定した詳細メッセージ付きで `InvalidLDIFRecordException` を構成します。

パラメータ：

lineNo — LDIF レコードの番号。

s — 詳細メッセージ。

メソッド

printStackTrace()

構文：

```
void printStackTrace()
```

printStackTrace(PrintStream pout)

構文：

```
void printStackTrace(PrintStream pout)
```

printStackTrace(PrintWriter wout)

構文：

```
void printStackTrace(PrintWriter wout)
```

oracle.ldap.util.InvalidParameterException

構文：

```
public class oracle.ldap.util.InvalidParameterException extends java.lang.Exception
```

説明：

入力パラメータの解析中にエラーが発生すると、このクラスのオブジェクトがスローされます。

コンストラクタ

InvalidParameterException()

構文：

```
public InvalidParameterException()
```

説明：

詳細メッセージなしで `InvalidParameterException` を構成します。

InvalidParameterException(String s)

構文：

```
public InvalidParameterException(String s)
```

説明：

指定した詳細メッセージ付きで `InvalidParameterException` を構成します。

パラメータ

s – 詳細メッセージ。

oracle.ldap.util.InvalidRootOrclctxException

構文：

```
public class oracle.ldap.util.InvalidRootOrclctxException extends  
oracle.ldap.util.UtilException
```

説明：

ルートの Oracle コンテキスト内を検索中に無効なルートの Oracle コンテキストが見つかる
と、この例外がスローされます。

コンストラクタ

InvalidRootOrclctxException()

構文：

```
public InvalidRootOrclctxException()
```

説明：

詳細メッセージなしで `InvalidRootOrclctxException` を構成します。

InvalidRootOrclctxException(String s)

構文：

```
public InvalidRootOrclctxException(String s)
```

説明：

指定した詳細メッセージ付きで `InvalidRootOrclctxException` を構成します。

パラメータ：

s – 詳細メッセージ。

oracle.ldap.util.InvalidSubscriberOrclctxException

構文:

```
public class oracle.ldap.util.InvalidSubscriberOrclctxException extends  
oracle.ldap.util.UtilException
```

説明:

サブスクリバ内で無効な Oracle コンテキストが見つかると、この例外がスローされます。

コンストラクタ

InvalidSubscriberOrclctxException()

構文:

```
public InvalidSubscriberOrclctxException()
```

説明:

詳細メッセージなしで InvalidSubscriberOrclctxException を構成します。

InvalidSubscriberOrclctxException(String s)

構文:

```
public InvalidSubscriberOrclctxException(String s)
```

説明:

指定した詳細メッセージ付きで InvalidSubscriberOrclctxException を構成します。

パラメータ:

s — 詳細メッセージ。

oracle.ldap.util.LoginPolicyFailureException

構文:

```
public class oracle.ldap.util.LoginPolicyFailureException extends  
oracle.ldap.util.UtilException
```

コンストラクタ

LoginPolicyFailureException()

構文:

```
public LoginPolicyFailureException()
```

LoginPolicyFailureException(String s)

構文:

```
public LoginPolicyFailureException(String s)
```

oracle.ldap.util.MigrationException

構文:

```
public class oracle.ldap.util.MigrationException extends java.lang.Exception
```

説明:

移行例外が発生すると、このクラスのオブジェクトがスローされます。

コンストラクタ

MigrationException()

構文:

```
public MigrationException()
```

説明:

詳細メッセージなしで MigrationException を構成します。

MigrationException(String s)

構文:

```
public MigrationException(String s)
```


説明：

指定した詳細メッセージ付きで `MigrationException` を構成します。

パラメータ：

s – 詳細メッセージ。

oracle.ldap.util.MultipleSubscriberException

構文：

```
public class oracle.ldap.util.MultipleSubscriberException extends  
oracle.ldap.util.UtilException
```

説明：

サブスクライバ検索ベースで、同一 ID のサブスクライバが複数見つかり、この例外がスローされます。

コンストラクタ

MultipleSubscriberException()**構文：**

```
public MultipleSubscriberException()
```

説明：

詳細メッセージなしで `MultipleSubscriberException` を構成します。

MultipleSubscriberException(String s)**構文：**

```
public MultipleSubscriberException(String s)
```

説明：

指定した詳細メッセージ付きで `MultipleSubscriberException` を構成します。

パラメータ：

s – 詳細メッセージ。

oracle.ldap.util.MultipleUserException

構文：

```
public class oracle.ldap.util.MultipleUserException extends  
oracle.ldap.util.UtilException
```

説明：

サブスクライバ・ユーザー検索ベースで、同一 ID のユーザーが複数見つかったと、この例外がスローされます。

コンストラクタ

MultipleUserException()

構文：

```
public MultipleUserException()
```

説明：

詳細メッセージなしで MultipleUserException を構成します。

MultipleUserException(String s)

構文：

```
public MultipleUserException(String s)
```

説明：

指定した詳細メッセージ付きで MultipleUserException を構成します。

パラメータ：

s - 詳細メッセージ。

oracle.ldap.util.NoGroupMembersException

構文:

```
public class oracle.ldap.util.NoGroupMembersException extends  
oracle.ldap.util.UtilException
```

説明:

ユーザーのグループ・メンバーシップを検索した結果、特定グループの一意のメンバーではなかった場合に、この例外がスローされます。

コンストラクタ

NoGroupMembersException()

構文:

```
public NoGroupMembersException()
```

説明:

詳細メッセージなしで NoGroupMemberException を構成します。

NoGroupMembersException(String s)

構文:

```
public NoGroupMembersException(String s)
```

説明:

指定した詳細メッセージ付きで NoGroupMemberException を構成します。

パラメータ:

s – 詳細メッセージ。

oracle.ldap.util.NoRootOrclctxException

構文:

```
public class oracle.ldap.util.NoRootOrclctxException extends  
oracle.ldap.util.UtilException
```

説明:

ルートの Oracle コンテキストが存在しなかった場合は、この例外がスローされます。

コンストラクタ

NoRootOrclctxException()

構文：

```
public NoRootOrclctxException()
```

説明：

詳細メッセージなしで NoRootOrclctxException を構成します。

NoRootOrclctxException(String s)

構文：

```
public NoRootOrclctxException(String s)
```

説明：

指定した詳細メッセージ付きで NoRootOrclctxException を構成します。

パラメータ：

s — 詳細メッセージ。

oracle.ldap.util.NoSubscriberOrclctxException

構文：

```
public class oracle.ldap.util.NoSubscriberOrclctxException extends  
oracle.ldap.util.UtilException
```

説明：

サブスライバ内で Oracle コンテキストの位置を特定できなかった場合は、この例外がスローされます。

コンストラクタ

NoSubscriberOrclctxException()

構文：

```
public NoSubscriberOrclctxException()
```

説明：

詳細メッセージなしで NoSubscriberOrclctxException を構成します。

NoSubscriberOrclctxException(String s)**構文：**

```
public NoSubscriberOrclctxException(String s)
```

説明：

指定した詳細メッセージ付きで `NoSubscriberOrclctxException` を構成します。

パラメータ：

s – 詳細メッセージ。

oracle.ldap.util.NoSuchGroupException**構文：**

```
public class oracle.ldap.util.NoSuchGroupException extends  
oracle.ldap.util.UtilException
```

説明：

グループがディレクトリに存在しないために `Group` オブジェクトを解決できなかった場合は、この例外がスローされます。

コンストラクタ**NoSuchGroupException()****構文：**

```
public NoSuchGroupException()
```

説明：

詳細メッセージなしで `NoSuchGroupException` を構成します。

NoSuchGroupException(String s)**構文：**

```
public NoSuchGroupException(String s)
```

説明：

指定した詳細メッセージ付きで `NoSuchGroupException` を構成します。

パラメータ：

s – 詳細メッセージ。

oracle.ldap.util.NoSuchSubscriberException

構文：

```
public class oracle.ldap.util.NoSuchSubscriberException extends  
oracle.ldap.util.UtilException
```

説明：

サブスクライバ検索ベースで、サブスクライバが存在しないために **Subscriber** オブジェクトを解決できなかった場合は、この例外がスローされます。

コンストラクタ

NoSuchSubscriberException()

構文：

```
public NoSuchSubscriberException()
```

説明：

詳細メッセージなしで **NoSuchSubscriberException** を構成します。

NoSuchSubscriberException(String s)

構文：

```
public NoSuchSubscriberException(String s)
```

説明：

指定した詳細メッセージ付きで **NoSuchSubscriberException** を構成します。

パラメータ：

s - 詳細メッセージ。

oracle.ldap.util.NoSuchUserException

構文：

```
public class oracle.ldap.util.NoSuchUserException extends  
oracle.ldap.util.UtilException
```

説明：

指定したサブスクライバのユーザー検索ベースで、ユーザーが存在しないために **User** オブジェクトを解決できなかった場合は、この例外がスローされます。

コンストラクタ

NoSuchUserException()

構文:

```
public NoSuchUserException()
```

説明:

詳細メッセージなしで `NoSuchUserException` を構成します。

NoSuchUserException(String s)

構文:

```
public NoSuchUserException(String s)
```

説明:

指定した詳細メッセージ付きで `NoSuchUserException` を構成します。

パラメータ:

s — 詳細メッセージ。

oracle.ldap.util.ParameterException

構文:

```
public class oracle.ldap.util.ParameterException extends  
oracle.ldap.util.UtilException
```

説明:

入力パラメータの解析中にエラーが発生すると、この例外がスローされます。

コンストラクタ

ParameterException()

構文:

```
public ParameterException()
```

説明:

詳細メッセージなしで `ParameterException` を構成します。

ParameterException(String s)

構文：

```
public ParameterException(String s)
```

説明：

指定した詳細メッセージ付きで `ParameterException` を構成します。

パラメータ：

s — 詳細メッセージ。

oracle.ldap.util.PasswdExpiredException

構文：

```
public class oracle.ldap.util.PasswdExpiredException extends  
oracle.ldap.util.UtilException
```

コンストラクタ

PasswdExpiredException()

構文：

```
public PasswdExpiredException()
```

PasswdExpiredException(String s)

構文：

```
public PasswdExpiredException(String s)
```

oracle.ldap.util.SetPropertiesException

構文：

```
public class oracle.ldap.util.SetPropertiesException extends  
oracle.ldap.util.UtilException
```

説明：

`setProperties()` メソッドの使用中に変更を実行できなかった場合は、この例外がスローされます。

コンストラクタ

SetPropertiesException()

構文：

```
public SetPropertiesException()
```

説明：

詳細メッセージなしで `SetPropertiesException` を構成します。

SetPropertiesException(String s)

構文：

```
public SetPropertiesException(String s)
```

説明：

指定した詳細メッセージ付きで `SetPropertiesException` を構成します。

パラメータ：

s – 詳細メッセージ。

oracle.ldap.util.SubscriberNotFoundException

構文：

```
public class oracle.ldap.util.SubscriberNotFoundException extends  
oracle.ldap.util.UtilException
```

説明：

サブスクライバ検索ベースでサブスクライバの位置を特定できなかった場合は、この例外がスローされます。

コンストラクタ

SubscriberNotFoundException()

構文：

```
public SubscriberNotFoundException()
```

説明：

詳細メッセージなしで `SubscriberNotFoundException` を構成します。

SubscriberNotFoundException(String s)

構文：

```
public SubscriberNotFoundException(String s)
```

説明：

指定した詳細メッセージ付きで `SubscriberNotFoundException` を構成します。

パラメータ：

s — 詳細メッセージ。

oracle.ldap.util.UtilException

構文：

```
public class oracle.ldap.util.UtilException extends java.lang.Exception
```

コンストラクタ

UtilException()

構文：

```
public UtilException()
```

UtilException(String s)

構文：

```
public UtilException(String s)
```

DBMS_LDAP_UTL PL/SQL パッケージ

この章では、Oracle の拡張機能のユーティリティ・ファンクションが含まれている DBMS_LDAP_UTL パッケージの概要について説明します。この章では、次の項目について説明します。

- [概要](#)
- [DBMS_LDAP_UTL リファレンス](#)
- [ファンクション・リターン・コードの概要](#)
- [データ型の概要](#)

概要

この項では、DBMS_LDAP_UTL サブプログラムに関する詳細を示します。各サブプログラムのエントリには、次の情報が含まれています。

表 7-1 ファンクション・エントリ情報

用語	説明
構文	パラメータの順序やタイプなど、ファンクションをコールするための構文を示すコード。
説明	ファンクションの目的に関する簡潔な説明。
コメント	ファンクションに関する詳細情報。ファンクションを使用する上での制限、あるいはアプリケーションでファンクションを使用する際に役立つその他の情報です。
パラメータ	ファンクションの各パラメータに関する説明。パラメータのモードも記載されています。パラメータのモードには、次の値があります。 IN –データを Oracle に渡すパラメータ。 OUT –このコールまたは後続のコールで Oracle からデータを受け取るパラメータ。 IN/OUT –コール時にデータを渡し、このコールまたは後続のコールから戻る際にデータを受け取るパラメータ。
戻り値	ファンクションが戻す値。
使用方法	特定のファンクションの使用に関する注意。
関連ファンクション	関連項目の見出しの下にリストされている関連ファンクション。

DBMS_LDAP_UTL リファレンス

この項では、DBMS_LDAP_UTL ファンクションに関する情報を示します。この項では、次の項目について説明します。

- サブプログラムの概要
- ユーザー関連サブプログラム
- グループ関連サブプログラム
- サブスクリバ関連サブプログラム
- その他のサブプログラム

サブプログラムの概要

表 7-2 DBMS_LDAP_UTL のユーザー関連サブプログラム

ファンクションまたは プロシージャ	用途
authenticate_user ファンクション	Lightweight Directory Access Protocol (LDAP) サーバー に対してユーザーを認証します。
create_user_handle ファンクション	ユーザー・ハンドルを作成します。
set_user_handle_properties ファン クション	指定したプロパティをユーザー・ハンドルに関連付けま す。
get_user_properties ファンクション	LDAP サーバーからユーザー・プロパティを取得します。
set_user_properties ファンクション	ユーザーのプロパティを変更します。
get_user_extended_properties ファン クション	ユーザーの拡張プロパティを取得します。
get_user_dn ファンクション	ユーザーの識別名を取得します。
check_group_membership ファン クション	ユーザーが、指定されたグループのメンバーであるかどう かをチェックします。
locate_subscriber_for_user ファンク ション	指定したユーザーのサブスクリバを取得します。
get_group_membership ファンク ション	ユーザーがメンバーとなっているグループのリストを取得 します。

表 7-3 DBMS_LDAP_UTL のグループ関連サブプログラム

ファンクションまたは プロシージャ	用途
create_group_handle ファンクシ ョン	グループ・ハンドルを作成します。
set_group_handle_properties ファ ンクション	指定したプロパティをグループ・ハンドルに関連付けま す。
get_group_properties ファンクシ ョン	LDAP サーバーからグループ・プロパティを取得します。
get_group_dn ファンクション	グループの識別名を取得します。

表 7-4 DBMS_LDAP_UTL のサブスクライバ関連サブプログラム

ファンクションまたは プロシージャ	用途
create_subscriber_handle ファンク ション	サブスクライバ・ハンドルを作成します。
get_subscriber_properties ファンク ション	LDAP サーバーからサブスクライバ・プロパティを取得し ます。
get_subscriber_dn ファンクション	サブスクライバの識別名を取得します。

表 7-5 DBMS_LDAP_UTL のその他のサブプログラム

ファンクションまたは プロシージャ	用途
normalize_dn_with_case ファンク ション	識別名の文字列を正規化します。
get_property_names ファンクション	PROPERTY_SET のプロパティ名のリストを取得します。
get_property_values ファンクション	プロパティ名の値リストを取得します。
get_property_values_len ファンク ション	プロパティ名のバイナリ値のリストを取得します。
free_propertyset_collection プロ シージャ	PROPERTY_SET_COLLECTION を解放します。
create_mod_propertyset ファンク ション	MOD_PROPERTY_SET を作成します。
populate_mod_propertyset ファン クション	MOD_PROPERTY_SET の構造を移入します。

ファンクションまたは プロシージャ	用途
free_mod_propertyset プロシージャ	MOD_PROPERTY_SET を解放します。
free_handle プロシージャ	ハンドルを解放します。
check_interface_version ファンク ション	インタフェースのバージョンに関するサポートをチェック します。

ユーザー関連サブプログラム

authenticate_user ファンクション

authenticate_user() は、Oracle Internet Directory に対してユーザーを認証するファンクションです。

構文

```
FUNCTION authenticate_user
(
  ld IN SESSION,
  user_handle IN HANDLE,
  auth_type IN PLS_INTEGER,
  credentials IN VARCHAR2,
  binary_credentials IN RAW
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-6 AUTHENTICATE_USER ファンクションのパラメータ

パラメータ名	パラメータ・ タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
auth_type	PLS_INTEGER	認証のタイプ。有効な値は、次のとおりです。 - DBMS_LDAP_UTL.AUTH_SIMPLE
credentials	VARCHAR2	ユーザー資格証明。有効な値は、次のとおりです。 DBMS_LDAP_UTL.AUTH_SIMPLE - パスワード

パラメータ名	パラメータ・タイプ	パラメータの説明
binary_credentials	RAW	バイナリ資格証明。有効な値は、次のとおりです。 DBMS_LDAP_UTL.AUTH_SIMPLE – NULL

戻り値

表 7-7 AUTHENTICATE_USER ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	認証に失敗しました。
DBMS_LDAP_UTL.NO_SUCH_USER	ユーザーが存在しません。
DBMS_LDAP_UTL.MULTIPLE_USER_ENTRIES	指定したユーザーに対して、複数のユーザー識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_SUBSCRIBER_ORCL_CTX	サブスクライバの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.NO_SUCH_SUBSCRIBER	サブスクライバが存在しません。
DBMS_LDAP_UTL.MULTIPLE_SUBSCRIBER_ENTRIES	指定したサブスクライバに対して、複数のサブスクライバ識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.ACCT_TOTALLY_LOCKED_EXCP	ユーザー・アカウントがロックされています。
DBMS_LDAP_UTL.AUTH_PASSWD_CHANGE_WARN	パスワードの変更が必要です。
DBMS_LDAP_UTL.AUTH_FAILURE_EXCP	認証に失敗しました。
DBMS_LDAP_UTL.PWD_EXPIRED_EXCP	ユーザー・パスワードが期限切れです。
DBMS_LDAP_UTL.PWD_GRACELOGIN_WARN	ユーザーの猶予期間ログインです。

値	説明
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()、DBMS_LDAP_UTL.create_user_handle()

create_user_handle ファンクション

create_user_handle() は、ユーザー・ハンドルを作成するファンクションです。

構文

```
FUNCTION create_user_handle
(
  user_hd OUT HANDLE,
  user_type IN PLS_INTEGER,
  user_id IN VARCHAR2,
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-8 CREATE_USER_HANDLE ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
user_hd	HANDLE	ユーザーのハンドルへのポインタです。
user_type	PLS_INTEGER	渡されるユーザー ID のタイプ。この引数に有効な値は、次のとおりです。 <ul style="list-style-type: none">■ - DBMS_LDAP_UTL.TYPE_DN■ - DBMS_LDAP_UTL.TYPE_GUID■ - DBMS_LDAP_UTL.TYPE_NICKNAME
user_id	VARCHAR2	ユーザー・エントリを表すユーザー ID です。

戻り値

表 7-9 CREATE_USER_HANDLE ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

関連項目

DBMS_LDAP_UTL.get_user_properties()、DBMS_LDAP_UTL.set_user_handle_properties()

set_user_handle_properties ファンクション

set_user_handle_properties() は、ユーザー・ハンドルのプロパティを構成するファンクションです。

構文

```
FUNCTION set_user_handle_properties
(
  user_hd IN HANDLE,
  property_type IN PLS_INTEGER,
  property IN HANDLE
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-10 SET_USER_HANDLE_PROPERTIES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
user_hd	HANDLE	ユーザーのハンドルへのポインタです。
property_type	PLS_INTEGER	渡されるプロパティのタイプ。この引数に有効な値は、次のとおりです。 - DBMS_LDAP_UTL.SUBSCRIBER_HANDLE
property	HANDLE	ユーザー・エントリを記述するプロパティです。

戻り値

表 7-11 SET_USER_HANDLE_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.RESET_HANDLE	コール元が既存のハンドル・プロパティをリセットしようとした場合の戻り値です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

使用方法

ユーザー・ハンドルが、TYPE_DN または TYPE_GUID で user_type として作成されている場合は、サブスクリイバ・ハンドルをユーザー・ハンドルのプロパティに設定する必要はありません。

関連項目

DBMS_LDAP_UTL.get_user_properties()

get_user_properties ファンクション

get_user_properties() は、ユーザー・プロパティを取得するファンクションです。

構文

```
FUNCTION get_user_properties
(
  ld IN SESSION,
  user_handle IN HANDLE,
  attrs IN STRING_COLLECTION,
  ptype IN PLS_INTEGER,
  ret_pset_coll OUT PROPERTY_SET_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-12 GET_USER_PROPERTIES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
attrs	STRING_COLLECTION	ユーザーに対してフェッチする属性のリストです。
ptype	PLS_INTEGER	戻すプロパティのタイプ。有効な値は、次のとおりです。 - DBMS_LDAP_UTL.ENTRY_PROPERTIES - DBMS_LDAP_UTL.NICKNAME_PROPERTY
ret-pset_coll	PROPERTY_SET_COLLECTION	コール元が要求した属性が含まれているユーザーの詳細です。

戻り値

表 7-13 GET_USER_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.NO_SUCH_USER	ユーザーが存在しません。
DBMS_LDAP_UTL.MULTIPLE_USER_ENTRIES	指定したユーザーに対して、複数のユーザー識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを返します。

使用方法

このファンクションには、次の要件があります。

- DBMS_LDAP.init() ファンクションで取得した有効な LDAP セッション・ハンドルが必要です。
- ユーザーのタイプが DBMS_LDAP_UTL.TYPE_NICKNAME の場合は、グループ・ハンドルのプロパティに有効なサブスクリバ・ハンドルの設定が必要です。

このファンクションは、NULL のサブスクリプタ・ハンドルをデフォルト・サブスクリプタとして識別しません。デフォルト・サブスクリプタは、引数に NULL の subscriber_id が渡される DBMS_LDAP_UTL.create_subscriber_handle() で取得できます。

グループ・タイプが次のいずれかの場合は、サブスクリバ・ハンドルをユーザー・ハンドルのプロパティに設定する必要はありません。

- DBMS_LDAP_UTL.TYPE_GUID

- DBMS_LDAP_UTL.TYPE_DN

サブスクリバ・ハンドルが設定されている場合、サブスクリバ・ハンドルは無視されます。

関連項目

DBMS_LDAP.init()、DBMS_LDAP_UTL.create_user_handle()

set_user_properties ファンクション

set_user_properties() は、ユーザーのプロパティを変更するファンクションです。

構文

```
FUNCTION set_user_properties
(
  ld IN SESSION,
  user_handle IN HANDLE,
  pset_type IN PLS_INTEGER,
  mod_pset IN PROPERTY_SET,
  mod_op IN PLS_INTEGER
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-14 SET_USER_PROPERTIES ファンクションのパラメータ

パラメータ名	パラメータ・ タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
pset_type	PLS_INTEGER	変更するプロパティ・セットのタイプ。有効な値は、次のとおりです。 - ENTRY_PROPERTIES
mod_pset	PROPERTY_SET	プロパティ・セットに対して実行する変更操作が含まれているデータ構造です。
mod_op	PLS_INTEGER	プロパティ・セットに対して実行する変更操作のタイプ。有効な値は、次のとおりです。 - ADD_PROPERTYSET - MODIFY_PROPERTYSET -DELETE_PROPERTYSET

戻り値

表 7-15 SET_USER_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.NO_SUCH_USER	ユーザーが存在しません。
DBMS_LDAP_UTL.MULTIPLE_USER_ENTRIES	指定したユーザーに対して、複数のユーザー識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.PWD_MIN_LENGTH_ERROR	パスワードの長さが最低限の長さに達していません。
DBMS_LDAP_UTL.PWD_NUMERIC_ERROR	パスワードに数字を含める必要があります。
DBMS_LDAP_UTL.PWD_NULL_ERROR	パスワードは NULL にできません。

値	説明
DBMS_LDAP_UTL.PWD_INHISTORY_ERROR	置換したパスワードと同じパスワードを指定することはできません。
DBMS_LDAP_UTL.PWD_ILLEGALVALUE_ERROR	パスワードに無効な文字が含まれています。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()、DBMS_LDAP_UTL.get_user_properties()

get_user_extended_properties ファンクション

get_user_extended_properties() は、ユーザーの拡張プロパティを取得するファンクションです。

構文

```
FUNCTION get_user_extended_properties
(
  ld IN SESSION,
  user_handle IN HANDLE,
  attrs IN STRING_COLLECTION
  ptype IN PLS_INTEGER,
  filter IN VARCHAR2,
  rep_pset_coll OUT PROPERTY_SET_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-16 GET_USER_EXTENDED_PROPERTIES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
attrs	STRING_COLLECTION	ユーザーに対してフェッチする属性のリストです。
ptype	PLS_INTEGER	戻すプロパティのタイプ。有効な値は、次のとおりです。 - DBMS_LDAP_UTL.EXTPROPTYPE_RESOURCE_ACCESS_DES
filter	VARCHAR2	ファンクションで戻されたユーザー・プロパティをさらに明確にするための LDAP フィルタです。
ret_pset_collection	PROPERTY_SET_COLLECTION	コール元が要求した属性が含まれているユーザーの詳細です。

戻り値

表 7-17 GET_USER_EXTENDED_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.NO_SUCH_USER	ユーザーが存在しません。
DBMS_LDAP_UTL.MULTIPLE_USER_ENTRIES	指定したユーザーに対して、複数のユーザー識別名エントリがディレクトリに存在します。
USER_PROPERTY_NOT_FOUND	ユーザーの拡張プロパティが存在しません。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

値	説明
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()、DBMS_LDAP_UTL.get_user_properties()

get_user_dn ファンクション

get_user_dn は、ユーザーの識別名を戻すファンクションです。

構文

```
FUNCTION get_user_dn
(
  ld IN SESSION,
  user_handle IN HANDLE,
  dn OUT VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-18 GET_USER_DN ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
dn	VARCHAR2	ユーザーの識別名です。

戻り値

表 7-19 GET_USER_DN ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	認証に失敗しました。
DBMS_LDAP_UTL.NO_SUCH_USER	ユーザーが存在しません。
DBMS_LDAP_UTL.MULTIPLE_USER_ENTRIES	指定したユーザーに対して、複数のユーザー識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()

check_group_membership ファンクション

check_group_membership() は、グループに対してユーザーのメンバーシップをチェックするファンクションです。

構文

```
FUNCTION check_group_membership
(
  ld IN SESSION,
  user_handle IN HANDLE,
  group_handle IN HANDLE,
  nested IN PLS_INTEGER
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-20 CHECK_GROUP_MEMBERSHIP ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
group_handle	HANDLE	グループ・ハンドルです。
nested	PLS_INTEGER	ユーザーがグループ内で保持しているメンバーシップのタイプ。有効な値は、次のとおりです。 DBMS_LDAP_UTL.NESTED_MEMBERSHIP DBMS_LDAP_UTL.DIRECT_MEMBERSHIP

戻り値

表 7-21 CHECK_GROUP_MEMBERSHIP ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	ユーザーがメンバーである場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GROUP_MEMBERSHIP	ユーザーがメンバーでない場合の戻り値です。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.get_group_membership()

locate_subscriber_for_user ファンクション

locate_subscriber_for_user() は、指定したユーザーのサブスクライバを取得し、そのサブスクライバへのハンドルを戻すファンクションです。

構文

```
FUNCTION locate_subscriber_for_user
(
  ld IN SESSION,
  user_handle IN HANDLE,
  subscriber_handle OUT HANDLE
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-22 LOCATE_SUBSCRIBER_FOR_USER ファンクションのパラメータ

パラメータ名	パラメータ・ タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
subscriber_handle	HANDLE	サブスクライバ・ハンドルです。

戻り値

表 7-23 LOCATE SUBSCRIBER FOR USER ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.NO_SUCH_SUBSCRIBER	サブスクライバが存在しません。
DBMS_LDAP_UTL.MULTIPLE_SUBSCRIBER_ENTRIES	指定したサブスクライバに対して、複数のサブスクライバ識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.NO_SUCH_USER	ユーザーが存在しません。
DBMS_LDAP_UTL.MULTIPLE_USER_ENTRIES	指定したユーザーに対して、複数のユーザー識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.SUBSCRIBER_NOT_FOUND	指定したユーザーのサブスクライバの位置を特定できません。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.ACCT_TOTALLY_LOCKED_EXCP	ユーザー・アカウントがロックされています。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()、DBMS_LDAP_UTL.create_user_handle()

get_group_membership ファンクション

get_group_membership() は、ユーザーがメンバーになっているグループのリストを戻すファンクションです。

構文

```
FUNCTION get_group_membership
(
  ld IN SESSION
  user_handle IN HANDLE,
  nested IN PLS_INTEGER,
  attr_list IN STRING_COLLECTION,
  ret_groups OUT PROPERTY_SET_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-24 GET_GROUP_MEMBERSHIP ファンクションのパラメータ

パラメータ名	パラメータ・ タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
user_handle	HANDLE	ユーザー・ハンドルです。
nested	PLS_INTEGER	ユーザーがグループ内で保持しているメンバーシップのタイプ。有効な値は、次のとおりです。 DBMS_LDAP_UTL.NESTED_MEMBERSHIP DBMS_LDAP_UTL.DIRECT_MEMBERSHIP
attr_list	STRING_COLLECTION	戻される属性のリストです。
ret_groups	PROPERTY_SET_COLLECTION	グループ・エントリの配列へのポインタを指すポインタです。

戻り値

表 7-25 GET_GROUP_MEMBERSHIP ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()

グループ関連サブプログラム

create_group_handle ファンクション

create_group_handle() は、グループ・ハンドルを作成するファンクションです。

構文

```
FUNCTION create_group_handle
(
  group_hd OUT HANDLE,
  group_type IN PLS_INTEGER,
  group_id IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-26 CREATE_GROUP_HANDLE ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
group_hd	HANDLE	グループのハンドルへのポインタです。
group_type	PLS_INTEGER	渡されるグループ ID のタイプ。この引数に有効な値は、次のとおりです。 - DBMS_LDAP_UTL.TYPE_DN - DBMS_LDAP_UTL.TYPE_GUID - DBMS_LDAP_UTL.TYPE_NICKNAME
group_id	VARCHAR2	グループ・エントリを表すグループ ID です。

戻り値

表 7-27 CREATE_GROUP_HANDLE ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

関連項目

DBMS_LDAP_UTL.get_group_properties()、DBMS_LDAP_UTL.set_group_handle_properties()

set_group_handle_properties ファンクション

set_group_handle_properties() は、グループ・ハンドルのプロパティを構成するファンクションです。

構文

```
FUNCTION set_group_handle_properties
(
  group_hd IN HANDLE,
  property_type IN PLS_INTEGER,
  property IN HANDLE
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-28 SET_GROUP_HANDLE_PROPERTIES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
group_hd	HANDLE	グループのハンドルへのポインタです。
property_type	PLS_INTEGER	渡されるプロパティのタイプ。この引数に有効な値は、次のとおりです。 - DBMS_LDAP_UTL.GROUP_HANDLE
property	HANDLE	グループ・エントリを記述するプロパティです。

戻り値

表 7-29 SET_GROUP_HANDLE_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.RESET_HANDLE	コール元が既存のハンドル・プロパティをリセットしようとした場合の戻り値です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

使用方法

グループ・ハンドルが、TYPE_DN または TYPE_GUID で group_type として作成されている場合は、サブスライバ・ハンドルをグループ・ハンドルのプロパティに設定する必要はありません。

関連項目

DBMS_LDAP_UTL.get_group_properties()

get_group_properties ファンクション

get_group_properties() は、グループ・プロパティを取得するファンクションです。

構文

```
FUNCTION get_group_properties
(
  ld IN SESSION,
  group_handle IN HANDLE,
  attrs IN STRING_COLLECTION,
  ptype IN PLS_INTEGER,
  ret_pset_coll OUT PROPERTY_SET_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-30 GET_GROUP_PROPERTIES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
group_handle	HANDLE	グループ・ハンドルです。
attrs	STRING_COLLECTION	グループに対してフェッチする必要がある属性のリストです。
ptype	PLS_INTEGER	戻されるプロパティのタイプ。有効な値は、次のとおりです。 - DBMS_LDAP_UTL.ENTRY_PROPERTIES
ret_pset_coll	PROPERTY_SET_COLLECTION	コール元が要求した属性が含まれているグループの詳細です。

戻り値

表 7-31 GET_GROUP_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.NO_SUCH_GROUP	グループが存在しません。
DBMS_LDAP_UTL.MULTIPLE_GROUP_ENTRIES	指定したグループに対して、複数のグループ識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションには、次の要件があります。

- DBMS_LDAP.init() ファンクションで取得した有効な LDAP セッション・ハンドルが必要です。
- グループのタイプが DBMS_LDAP_UTL.TYPE_NICKNAME の場合は、グループ・ハンドルのプロパティに有効なサブスクリバ・ハンドルの設定が必要です。

このファンクションは、NULL のサブスクリプタ・ハンドルをデフォルト・サブスクリプタとして識別しません。デフォルト・サブスクリプタは、引数に NULL の subscriber_id が渡される DBMS_LDAP_UTL.create_subscriber_handle() で取得できます。

グループ・タイプが次のいずれかである場合は、サブスクリバ・ハンドルをグループ・ハンドルのプロパティに設定する必要はありません。

- DBMS_LDAP_UTL.TYPE_GUID

- DBMS_LDAP_UTL.TYPE_DN

サブスクリバ・ハンドルが設定されている場合、サブスクリバ・ハンドルは無視されません。

関連項目

DBMS_LDAP.init()、DBMS_LDAP_UTL.create_group_handle()

get_group_dn ファンクション

get_group_dn() は、グループの識別名を戻すファンクションです。

構文

```
FUNCTION get_group_dn
(
  ld IN SESSION,
  group_handle IN HANDLE
  dn OUT VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-32 GET_GROUP_DN ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
group_handle	HANDLE	グループ・ハンドルです。
dn	VARCHAR2	グループの識別名です。

戻り値

表 7-33 GET_GROUP_DN ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.NO_SUCH_GROUP	グループが存在しません。
DBMS_LDAP_UTL.MULTIPLE_GROUP_ENTRIES	指定したグループに対して、複数のグループ識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。

値	説明
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()

サブスクリバ関連サブプログラム

create_subscriber_handle ファンクション

create_subscriber_handle() は、サブスクリバ・ハンドルを作成するファンクションです。

構文

```

FUNCTION create_subscriber_handle
(
  subscriber_hd OUT HANDLE,
  subscriber_type IN PLS_INTEGER,
  subscriber_id IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-34 CREATE_SUBSCRIBER_HANDLE ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
subscriber_hd	HANDLE	サブスクライバのハンドルへのポインタです。
subscriber_type	PLS_INTEGER	渡されるサブスクライバ ID のタイプ。この引数に有効な値は、次のとおりです。 - DBMS_LDAP_UTL.TYPE_DN - DBMS_LDAP_UTL.TYPE_GUID - DBMS_LDAP_UTL.TYPE_NICKNAME - DBMS_LDAP_UTL.TYPE_DEFAULT
subscriber_id	VARCHAR2	サブスクライバ・エントリを表すサブスクライバ ID です。subscriber_type が次の値の場合は、このパラメータを NULL にできます。 - DBMS_LDAP_UTL.TYPE_DEFAULT この場合、デフォルト・サブスクライバはルートの Oracle コンテキストからフェッチされます。

戻り値

表 7-35 CREATE_SUBSCRIBER_HANDLE ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

関連項目

DBMS_LDAP_UTL.get_subscriber_properties()

get_subscriber_properties ファンクション

get_subscriber_properties() は、指定したサブスクライバ・ハンドルのプロパティを取得するファンクションです。

構文

```
FUNCTION get_subscriber_properties
(
  ld IN SESSION,
  subscriber_handle IN HANDLE,
  attrs IN STRING_COLLECTION,
  ptype IN PLS_INTEGER,
  ret_pset_coll OUT PROPERTY_SET_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-36 GET_SUBSCRIBER_PROPERTIES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
subscriber_handle	HANDLE	サブスクライバ・ハンドルです。
attrs	STRING_COLLECTION	サブスクライバに対してフェッチする必要がある属性のリストです。
ptype	PLS_INTEGER	戻すプロパティのタイプ。有効な値は、次のとおりです。 - DBMS_LDAP_UTL.ENTRY_PROPERTIES - DBMS_LDAP_UTL.COMMON_PROPERTIES（サブスクライバの Oracle コンテキストのプロパティを取得します）
ret_pset_coll	PROPERTY_SET_COLLECITON	コール元が要求した属性が含まれているサブスクライバの詳細です。

戻り値

表 7-37 GET_SUBSCRIBER_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.NO_SUCH_SUBSCRIBER	サブスクライバが存在しません。
DBMS_LDAP_UTL.MULTIPLE_SUBSCRIBER_ENTRIES	指定したサブスクライバに対して、複数のサブスクライバ識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()、DBMS_LDAP_UTL.create_subscriber_handle()

get_subscriber_dn ファンクション

get_subscriber_dn() は、サブスクライバの識別名を戻すファンクションです。

構文

```
FUNCTION get_subscriber_dn
(
  ld IN SESSION,
  subscriber_handle IN HANDLE,
  dn OUT VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-38 GET_SUBSCRIBER_DN ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
ld	SESSION	有効な LDAP セッション・ハンドルです。
subscriber_handle	HANDLE	サブスクライバ・ハンドルです。
dn	VARCHAR2	サブスクライバの識別名です。

戻り値

表 7-39 GET_SUBSCRIBER_DN ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.NO_SUCH_SUBSCRIBER	サブスクライバが存在しません。
DBMS_LDAP_UTL.MULTIPLE_SUBSCRIBER_ENTRIES	指定したサブスクライバに対して、複数のサブスクライバ識別名エントリがディレクトリに存在します。

値	説明
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションは、DBMS_LDAP.init() のコールで有効な LDAP セッションを取得してからコールしてください。

関連項目

DBMS_LDAP.init()

その他のサブプログラム

normalize_dn_with_case ファンクション

normalize_dn_with_case() は、識別名から不要な空白文字を削除し、フラグに基づいてすべての文字を小文字に変換するファンクションです。

構文

```
FUNCTION normalize_dn_with_case
(
  dn IN VARCHAR2,
  lower_case IN PLS_INTEGER,
  norm_dn OUT VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-40 NORMALIZE_DN_WITH_CASE ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
dn	VARCHAR2	識別名を指定します。
lower_case	PLS_INTEGER	1 を設定した場合は、正規化された識別名が小文字で戻されます。 0 (ゼロ) を設定した場合は、正規化された識別名の文字列がそのまま戻されます。
norm_dn	VARCHAR2	正規化された識別名です。

戻り値

表 7-41 NORMALIZE_DN_WITH_CASE ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	障害時の戻り値です。

使用方法

このファンクションは、2 つの識別名を比較する際に使用できます。

get_property_names ファンクション

get_property_names() は、プロパティ・セットのプロパティ名のリストを取得するファンクションです。

構文

```
FUNCTION get_property_names
(
  pset IN PROPERTY_SET,
  property_names OUT STRING_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-42 GET_PROPERTY_NAMES ファンクションのパラメータ

パラメータ名	パラメータ・ タイプ	パラメータの説明
pset	PROPERTY_SET	次のいずれかのファンクションで戻される PropertySetCollection 内のプロパティ・セットです。 - DBMS_LDAP_UTL.get_group_membership() - DBMS_LDAP_UTL.get_subscriber_properties() - DBMS_LDAP_UTL.get_user_properties() - DBMS_LDAP_UTL.get_group_properties()
property_names	STRING_ COLLECTION	プロパティ・セットに関連付けられているプロパティ 名のリストです。

戻り値

表 7-43 GET_PROPERTY_NAMES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値で す。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	エラーが発生した場合の戻り値 です。

関連項目

DBMS_LDAP_UTL.get_property values()

get_property_values ファンクション

get_property_values() は、指定したプロパティ名とプロパティに対するプロパティ値（文字列）を取得するファンクションです。

構文

```
FUNCTION get_property_values
(
  pset IN PROPERTY_SET,
  property_name IN VARCHAR2,
  property_values OUT STRING_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-44 GET_PROPERTY_VALUES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
property_name	VARCHAR2	プロパティ名です。
pset	PROPERTY_SET	次のいずれかのファンクションで戻される PropertySetCollection 内のプロパティ・セットです。 - DBMS_LDAP_UTL.get_group_membership() - DBMS_LDAP_UTL.get_subscriber_properties() - DBMS_LDAP_UTL.get_user_properties() - DBMS_LDAP_UTL.get_group_properties()
property_values	STRING_COLLECTION	プロパティ値（文字列）のリストです。

戻り値

表 7-45 GET_PROPERTY_VALUES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	障害時の戻り値です。

関連項目

DBMS_LDAP_UTL.get_property_values_len()

get_property_values_len ファンクション

get_property_values_len() は、指定したプロパティ名とプロパティに対するバイナリ・プロパティ値を取得するファンクションです。

構文

```
FUNCTION get_property_values_len
(
  pset IN PROPERTY_SET,
  property_name IN VARCHAR2,
  property_values OUT BINVAL_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-46 GET_PROPERTY_VALUES_LEN ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
property_name	VARCHAR2	プロパティ名です。
pset	PROPERTY_SET	次のいずれかのファンクションで戻される PropertySetCollection 内のプロパティ・セットです。 - DBMS_LDAP_UTL.get_group_membership() - DBMS_LDAP_UTL.get_subscriber_properties() - DBMS_LDAP_UTL.get_user_properties() - DBMS_LDAP_UTL.get_group_properties()
property_values	BINVAL_COLLECTION	バイナリ・プロパティ値のリストです。

戻り値

表 7-47 GET_PROPERTY_VALUES_LEN ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	障害時の戻り値です。

関連項目

DBMS_LDAP_UTL.get_property_values()

free_propertyset_collection プロシージャ

free_propertyset_collection() は、PropertySetCollection に関連付けられているメモリを解放するプロシージャです。

構文

```
PROCEDURE free_propertyset_collection
(
  pset_collection IN OUT PROPERTY_SET_COLLECTION
);
```

パラメータ

表 7-48 FREE_PROPERTYSET_COLLECTION プロシージャのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
pset_collection	PROPERTY_SET_COLLECTION	次のいずれかのファンクションで戻される PropertySetCollection です。 - DBMS_LDAP_UTL.get_group_membership() - DBMS_LDAP_UTL.get_subscriber_properties() - DBMS_LDAP_UTL.get_user_properties() - DBMS_LDAP_UTL.get_group_properties()

戻り値

該当なし

関連項目

DBMS_LDAP_UTL.get_group_membership()、DBMS_LDAP_UTL.get_subscriber_properties()、DBMS_LDAP_UTL.get_user_properties()、DBMS_LDAP_UTL.get_group_properties()

create_mod_propertyset ファンクション

create_mod_propertyset() は、MOD_PROPERTY_SET データ構造を作成するファンクションです。

構文

```
FUNCTION create_mod_propertyset
(
  pset_type IN PLS_INTEGER,
  pset_name IN VARCHAR2,
  mod_pset OUT MOD_PROPERTY_SET
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-49 CREATE_MOD_PROPERTYSET ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
pset_type	PLS_INTEGER	変更するプロパティ・セットのタイプ。有効な値は、次のとおりです。 - ENTRY_PROPERTIES
pset_name	VARCHAR2	プロパティ・セットの名前。ENTRY_PROPERTIES が変更されている場合は、このパラメータを NULL にできます。
mod_pset	MOD_PROPERTY_SET	プロパティ・セットに対して実行する変更操作が含まれているデータ構造です。

戻り値

表 7-50 CREATE_MOD_PROPERTYSET ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。

値	説明
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。

関連項目

DBMS_LDAP_UTL.populate_mod_propertyset()

populate_mod_propertyset ファンクション

populate_mod_propertyset() は、MOD_PROPERTY_SET データ構造を移入するファンクションです。

構文

```
FUNCTION populate_mod_propertyset
(
  mod_pset IN MOD_PROPERTY_SET,
  property_mod_op IN PLS_INTEGER,
  property_name IN VARCHAR2,
  property_values IN STRING_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-51 POPULATE_MOD_PROPERTYSET ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
mod_pset	MOD_PROPERTY_SET	Mod_PropertySet データ構造です。
property_mod_op	PLS_INTEGER	プロパティに対して実行する変更操作のタイプ。有効な値は、次のとおりです。 - ADD_PROPERTY - REPLACE_PROPERTY - DELETE_PROPERTY
property_name	VARCHAR2	プロパティの名前です。
property_values	STRING_COLLECTION	プロパティに関連付けられている値です。

戻り値

表 7-52 POPULATE_MOD_PROPERTYSET ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.GENERAL_ERROR	認証に失敗しました。
DBMS_LDAP_UTL.PWD_GRACELOGIN_WARN	ユーザーの猶予期間ログインです。

関連項目

DBMS_LDAP_UTL.create_mod_propertyset()

free_mod_propertyset プロシージャ

free_mod_propertyset() は、MOD_PROPERTY_SET データ構造を解放するプロシージャです。

構文

```
PROCEDURE free_mod_propertyset
(
  mod_pset IN MOD_PROPERTY_SET
);
```

パラメータ

表 7-53 FREE_MOD_PROPERTYSET プロシージャのパラメータ

パラメータ名	パラメータ・ タイプ	パラメータの説明
mod_pset	PROPERTY_SET	Mod_PropertySet データ構造です。

戻り値

該当なし

関連項目

DBMS_LDAP_UTL.create_mod_propertyset()

free_handle プロシージャ

free_handle() は、ハンドルに関連付けられているメモリーを解放するプロシージャです。

構文

```
PROCEDURE free_handle
(
  handle IN OUT HANDLE
);
```

パラメータ

表 7-54 FREE_HANDLE プロシージャのパラメータ

パラメータ名	パラメータ・ タイプ	パラメータの説明
handle	HANDLE	ハンドルへのポインタです。

戻り値

該当なし

関連項目

DBMS_LDAP_UTL.create_user_handle()、DBMS_LDAP_UTL.create_subscriber_handle()、DBMS_LDAP_UTL.create_group_handle()

check_interface_version ファンクション

check_interface_version() は、インタフェースのバージョンに関するサポートをチェックするファンクションです。

構文

```
FUNCTION check_interface_version
(
  interface_version IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 7-55 CHECK_INTERFACE_VERSION ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
interface_version	VARCHAR2	インタフェースのバージョンです。

戻り値

表 7-56 CHECK_VERSION_INTERFACE ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	インタフェースのバージョンはサポートされています。
DBMS_LDAP_UTL.GENERAL_ERROR	インタフェースのバージョンはサポートされていません。

ファンクション・リターン・コードの概要

DBMS_LDAP_UTL の各ファンクションは、次の表の値を戻す場合があります。

表 7-57 ファンクション・リターン・コード

名前	リターン・コード	説明
SUCCESS	0	操作は正常終了しました。
GENERAL_ERROR	-1	このエラー・コードは、ここにリストされている以外の障害が発生した場合に戻されます。
PARAM_ERROR	-2	入力パラメータが無効な場合は、すべてのファンクションがこのコードを戻します。
NO_GROUP_MEMBERSHIP	-3	指定したユーザーにグループのメンバーシップがない場合は、ユーザー関連とグループ関連のファンクションからこのコードが戻されます。
NO_SUCH_SUBSCRIBER	-4	ディレクトリにサブスクライバが存在しない場合は、サブスクライバ関連のファンクションからこのコードが戻されます。
NO_SUCH_USER	-5	ディレクトリにユーザーが存在しない場合は、ユーザー関連のファンクションからこのコードが戻されます。

表 7-57 ファンクション・リターン・コード (続き)

名前	リターン・コード	説明
NO_ROOT_ORCL_CTX	-6	ディレクトリにルートの Oracle コンテキストが存在しない場合は、ほとんどのファンクションがこのコードを戻します。
MULTIPLE_SUBSCRIBER_ENTRIES	-7	指定したサブスクライバ・ニックネームに対して複数のサブスクライバ・エントリが検出された場合は、サブスクライバ関連のファンクションからこのコードが戻されます。
INVALID_ROOT_ORCL_CTX	-8	ファンクションに必要なすべての必須情報が、ルート of Oracle コンテキストに含まれていません。
NO_SUBSCRIBER_ORCL_CTX	-9	サブスクライバの Oracle コンテキストが存在しません。
INVALID_SUBSCRIBER_ORCL_CTX	-10	サブスクライバの Oracle コンテキストが無効です。
MULTIPLE_USER_ENTRIES	-11	指定したユーザー・ニックネームに対して複数のユーザー・エントリが検出された場合は、ユーザー関連のファンクションからこのコードが戻されます。
NO_SUCH_GROUP	-12	ディレクトリにグループが存在しない場合は、グループ関連のファンクションからこのコードが戻されます。
MULTIPLE_GROUP_ENTRIES	-13	指定したグループ・ニックネームに対して、複数のグループ・エントリがディレクトリに存在しています。
ACCT_TOTALLY_LOCKED_EXCEPTION	-14	ユーザー・アカウントがロックされている場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。このエラーは、サブスクライバの Oracle コンテキストに設定されているパスワード・ポリシーに基づいています。
AUTH_PASSWD_CHANGE_WARN	-15	ユーザー・パスワードの変更が必要な場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
AUTH_FAILURE_EXCEPTION	-16	ユーザーの認証に失敗した場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。
PWD_EXPIRED_EXCEPTION	-17	ユーザー・パスワードが期限切れの場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。

表 7-57 ファンクション・リターン・コード（続き）

名前	リターン・コード	説明
RESET_HANDLE	-18	エントリ・ハンドルのプロパティをコール元がリセットしようとしている場合は、このコードが戻されます。
SUBSCRIBER_NOT_FOUND	-19	サブスクライバの位置を特定できない場合は、DBMS_LDAP_UTL.locate_subscriber_for_user() ファンクションからこのコードが戻されます。
PWD_EXPIRE_WARN	-20	ユーザー・パスワードの期限切れが近い場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
PWD_MINLENGTH_ERROR	-21	ユーザー・パスワードの変更時に、新規ユーザー・パスワードが最低限の長さに達していない場合は、DBMS_LDAP_UTL.set_user_properties() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
PWD_NUMERIC_ERROR	-22	ユーザー・パスワードの変更時に、新規ユーザー・パスワードに最低 1 文字の数字が含まれていない場合は、DBMS_LDAP_UTL.set_user_properties() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
PWD_NULL_ERROR	-23	ユーザー・パスワードの変更時に、新規ユーザー・パスワードに空のパスワードが指定された場合は、DBMS_LDAP_UTL.set_user_properties() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
PWD_INHISTORY_ERROR	-24	ユーザー・パスワードの変更時に、新規ユーザー・パスワードに以前と同じパスワードが指定された場合は、DBMS_LDAP_UTL.set_user_properties() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
PWD_ILLEGALVALUE_ERROR	-25	ユーザー・パスワードの変更時に、新規ユーザー・パスワードに無効な文字が指定された場合は、DBMS_LDAP_UTL.set_user_properties() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。

表 7-57 ファンクション・リターン・コード（続き）

名前	リターン・コード	説明
PWD_GRACELOGIN_WARN	-26	ユーザー・パスワードが期限切れで、ユーザーに猶予期間ログインが指定されている場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
PWD_MUSTCHANGE_ERROR	-27	ユーザー・パスワードの変更が必要な場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
USER_ACCT_DISABLED_ERROR	-29	ユーザー・アカウントが無効な場合は、DBMS_LDAP_UTL.authenticate_user() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。
PROPERTY_NOT_FOUND	-30	ディレクトリでユーザー・プロパティを検索している場合は、ユーザー関連のファンクションからこのコードが戻されます。

データ型の概要

DBMS_LDAP_UTL パッケージでは、次のデータ型が使用されます。

表 7-58

データ型	用途
HANDLE	エンティティに関するハンドルの保持に使用されます。
PROPERTY_SET	エンティティに関するプロパティの保持に使用されます。
PROPERTY_SET_COLLECTION	PROPERTY_SET 構造体のリストです。
MOD_PROPERTY_SET	エンティティに対する変更操作を保持する構造体です。

プロビジョニング統合アプリケーションの開発

この章では、Oracle Directory Integration Platform の Oracle Provisioning Integration Service が提供するサービスを活用するアプリケーションの開発方法について説明します。

これらのアプリケーションは、Oracle プラットフォームに基づいたレガシー・アプリケーションまたはサード・パーティ・アプリケーションのいずれでもかまいません。

この章では、次の項目について説明します。

- [前提となる知識](#)
- [プロビジョニング統合のための使用モデルの開発](#)
- [プロビジョニング統合に関するタスクの開発](#)
- [プロビジョニング・イベント・インタフェースの説明](#)

前提となる知識

次の内容についての知識が必要です。

- Lightweight Directory Access Protocol (LDAP) の一般的な概念
- Oracle Internet Directory
- Oracle Internet Directory の Oracle9i Application Server への統合
- Delegated Administration Service
- Oracle9i Application Server ドキュメント・セットの『Oracle Internet Directory 管理者ガイド』の「Oracle Provisioning Integration Service」で説明されているユーザー・プロビジョニング・モデル
- Oracle Directory Integration Platform
- SQL、PL/SQL およびデータベース RPC に関する知識

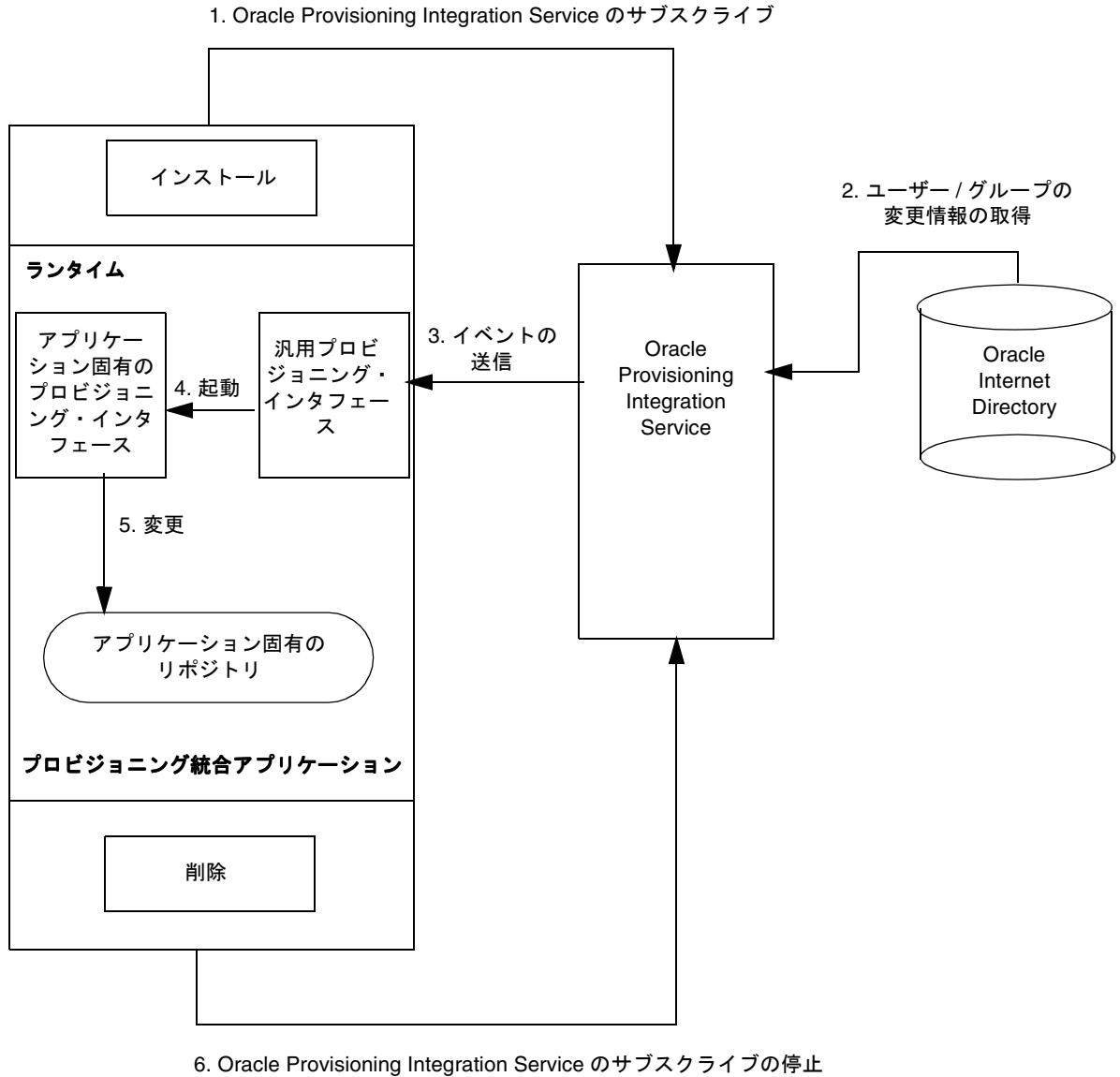
さらに、Oracle9iAS Single Sign-On を理解しておくことをお勧めします。

プロビジョニング統合のための使用モデルの開発

この項では、プロビジョニング統合アプリケーションのためのエージェントの使用モデルの概要について説明します。

[図 8-1](#) は、プロビジョニング・イベントを取得するアプリケーションのライフサイクルを示しています。

図 8-1 アプリケーションでプロビジョニング情報を取得する方法（Oracle Directory Synchronized Provisioning Platform を使用）



1. アプリケーションのインストール時に、次の情報が Oracle Provisioning Integration Service に提供されます。
 - Oracle Internet Directory にアプリケーション・エントリを登録するための情報
 - Oracle Internet Directory にアプリケーション固有のデータベース接続情報を登録するための情報
 - Oracle Provisioning Integration Service がアプリケーションにサービスを提供するための情報（必要な変更の種類やスケジューリング・プロパティなど）
2. Oracle Provisioning Integration Service は、ユーザー情報とグループ情報に対する変更情報を Oracle Internet Directory の変更ログから取得します。これによって、アプリケーションに送信する変更情報を判断します。
3. Oracle Provisioning Integration Service は、汎用プロビジョニング・インタフェースを起動し、データベース接続情報に基づいて、変更情報をアプリケーションに送信します。
4. 汎用プロビジョニング・インタフェースが、アプリケーション固有のロジックを起動します。
5. アプリケーション固有のロジックは、汎用プロビジョニング・イベントをアプリケーション固有のイベントに変換します。次に、アプリケーションのリポジトリで必要な変更を行います。
6. 管理者は、アプリケーションを手動で削除するか、あるいは自動削除処理を使用して削除できます。手動で削除する場合、管理者はプロビジョニング・サブスクリプション・ツールを使用して、プロビジョニング・プラットフォームからのアプリケーションのサブスクライブを停止します。oidprovtool と呼ばれるプロビジョニング・サブスクリプション・ツールが ORACLE_HOME から起動されます。

プロビジョニング統合に関するタスクの開発

同期化したプロビジョニング・アプリケーションを開発するには、次の一般的なタスクを実行します。

1. アプリケーション固有のロジックを開発し、プロビジョニング・システムからのイベントに応じてプロビジョニング・アクティビティを実行します。
2. アプリケーションのインストール手順を変更して、アプリケーションがプロビジョニング・イベントをサブスクライブできるようにします。

この項では、次の項目について説明します。

- [アプリケーションのインストール](#)
- [ユーザーの作成と登録](#)
- [ユーザーの削除](#)
- [アプリケーションの削除](#)

アプリケーションのインストール

インストール後に構成ツールを実行するために、各アプリケーションのインストール・ロジックを変更します。

アプリケーションのインストール時に、アプリケーションはプロビジョニング・サブスクリプション・ツール `oidprovtool` を起動します。このツールを起動する一般的なパターンは、次のとおりです。

```
oidprovtool param1=<p1_value> param2=<p2_value> param3=<p3_value> ...
```

関連項目：

- ツールのパラメータと指定可能な値の詳細は、9-28 ページの「[プロビジョニング・サブスクリプション・ツールの構文](#)」を参照してください。
- インストール後にツールで実行する必要があるタスクの詳細は、8-2 ページの「[プロビジョニング統合のための使用モデルの開発](#)」を参照してください。

ユーザーの作成と登録

最初に、Oracle Internet Directory にユーザーを作成します。次に、そのユーザーをアプリケーションに登録します。

これらのインタフェースのいずれかを使用する場合は、Oracle Provisioning Integration Service を使用可能にして、アプリケーションに現在登録されているユーザーを識別する必要があります。この識別によって、送信される削除イベントは、アプリケーションに登録されているユーザーのみに対応します。

アプリケーション・ロジックを実装して、Oracle Internet Directory の指定のユーザーがアプリケーションに登録されていることを `user_exists` ファンクションで検証できるようにします。

ユーザーの削除

ユーザーの削除イベントは、主に Oracle Provisioning Integration Service によって、Oracle Internet Directory から各種プロビジョニング統合アプリケーションに伝播されます。

アプリケーションは、PL/SQL コールバック・インタフェースを使用して、Oracle Provisioning Integration Service に登録され、次の情報を提供します。

- アプリケーションで使用される PL/SQL パッケージの名前
- そのパッケージにアクセスするための接続文字列

次に、Oracle Provisioning Integration Service はアプリケーション・データベースに接続し、必要な PL/SQL プロシージャを起動します。

図 8-2 は、PL/SQL コールバック・インタフェースのシステムの相互作用を示しています。

図 8-2 PL/SQL コールバック・ベースのアプローチによるユーザーの削除

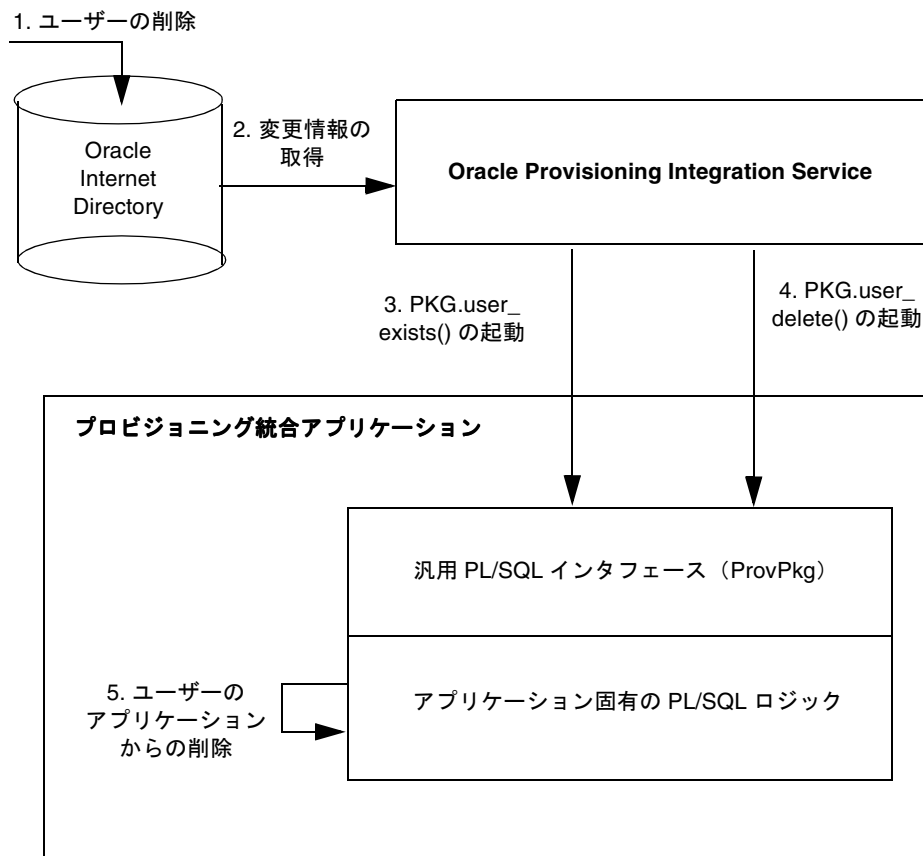


図 8-2 のように、ユーザーのアプリケーションからの削除は、次の手順で構成されています。

1. 管理者は、Oracle Directory Manager または同様のツールを使用して、Oracle Internet Directory 内のユーザーを削除します。
2. Oracle Provisioning Integration Service は、その変更情報を Oracle Internet Directory の変更ログ・インタフェースから取得します。
3. ディレクトリから削除したユーザーが、このアプリケーションに登録されていたかどうかを確認するために、Oracle Provisioning Integration Service は、アプリケーションの

プロビジョニング・イベント・インタフェースの `user_exists()` ファンクションを起動します。

4. ユーザーが登録されている場合、Oracle Provisioning Integration Service は、プロビジョニング・イベント・インタフェースの `user_delete()` ファンクションを起動します。
5. アプリケーション固有の PL/SQL ロジックは、ユーザーおよび関連するフットプリントをアプリケーション固有のリポジトリから削除します。

手順 5 は、プロビジョニング統合アプリケーションの開発者が実行します。

アプリケーションの削除

プロビジョニング・サブスクリプション・ツール (`oidprovtool`) を実行するための削除ロジックを各プロビジョニング統合アプリケーションで使用可能にする必要があります。このツールは、Oracle Provisioning Integration Service へのアプリケーションのサブスクライブを停止します。

プロビジョニング・イベント・インタフェースの説明

8-4 ページの「[プロビジョニング統合に関するタスクの開発](#)」で説明されているように、Oracle Provisioning Integration Service によって生成されたイベントをコンSUMEするためのロジックを開発する必要があります。アプリケーションと Oracle Provisioning Integration Service の間のインタフェースには、表ベースまたは PL/SQL のコールバックを使用できます。

関連項目： これらのインタフェースの使用方法は、8-2 ページの「[プロビジョニング統合のための使用モデルの開発](#)」を参照してください。

PL/SQL コールバック・インタフェースの場合は、Oracle Provisioning Integration Service がアプリケーション固有のデータベースで起動する PL/SQL パッケージを開発する必要があります。パッケージには任意の名前を選択できますが、サブスクリプション時にパッケージを登録する際は、必ず同じ名前を使用してください。次の PL/SQL パッケージ仕様でパッケージを実装します。

```
Rem
Rem      NAME
Rem      ldap_ntfy.pks - Provisioning Notification Package Specification.
Rem

DROP TYPE LDAP_ATTR_LIST;
DROP TYPE LDAP_ATTR;

-- LDAP ATTR
-----
```

```
--
-- Name      : LDAP_ATTR
-- Data Type  : OBJECT
-- DESCRIPTION : This structure contains details regarding
--              an attribute.
--
-----
CREATE TYPE LDAP_ATTR AS OBJECT (
    attr_name      VARCHAR2(255),
    attr_value      VARCHAR2(2048),
    attr_bvalue     RAW(2048),
    attr_value_len  INTEGER,
    attr_type       INTEGER -- (0 - String, 1 - Binary)
    attr_mod_op     INTEGER
);
/
GRANT EXECUTE ON LDAP_ATTR to public;

-----

--
-- Name      : LDAP_ATTR_LIST
-- Data Type  : COLLECTION
-- DESCRIPTION : This structure contains collection
--              of attributes.
--
-----
CREATE TYPE LDAP_ATTR_LIST AS TABLE OF LDAP_ATTR;
/
GRANT EXECUTE ON LDAP_ATTR_LIST to public;

-----

--
-- NAME      : LDAP_NTIFY
-- DESCRIPTION : This a notifier interface implemented by Provisioning System
--              clients to receive information about changes in OID.
--              The name of package can be customized as needed.
--              The functions names within this package SHOULD NOT be changed.
--
-----
CREATE OR REPLACE PACKAGE LDAP_NTIFY AS

    --
    -- LDAP_NTIFY data type definitions
    --
```



```
-- Event Types
USER_DELETE          CONSTANT VARCHAR2(256) := 'USER_DELETE';
USER_MODIFY          CONSTANT VARCHAR2(256) := 'USER_MODIFY';
GROUP_DELETE         CONSTANT VARCHAR2(256) := 'GROUP_DELETE';
GROUP_MODIFY         CONSTANT VARCHAR2(256) := 'GROUP_MODIFY';

-- Return Codes (Boolean)
SUCCESS              CONSTANT NUMBER := 1;
FAILURE              CONSTANT NUMBER := 0;

-- Values for attr_mod_op in LDAP_ATTR object.
MOD_ADD              CONSTANT NUMBER := 0;
MOD_DELETE           CONSTANT NUMBER := 1;
MOD_REPLACE          CONSTANT NUMBER := 2;
```

LDAP_NOTIFY ファクションの定義

user_exists ファクション

ユーザーがアプリケーションに登録されているかどうかをチェックするために、Oracle Provisioning Integration Service によって起動されるコールバック・ファンクションです。

構文

```
FUNCTION user_exists ( user_name    IN VARCHAR2,
                      user_guid    IN VARCHAR2,
                      user_dn      IN VARCHAR2)
```

パラメータ

表 8-1 user_exists ファクションのパラメータ

パラメータ	説明
user_name_	ユーザー識別子です。
user_guid	グローバル・ユーザー識別子です。
user_dn	ユーザー・エントリの識別名の属性です。

戻り値

ユーザーが存在する場合は（任意の）正数を返します。

group_exists ファンクション

グループがアプリケーションに存在するかどうかをチェックするために、Oracle Provisioning Integration Service によって起動されるコールバック・ファンクションです。

構文

```
FUNCTION group_exists ( group_name IN VARCHAR2,
                        group_guid IN VARCHAR2,
                        group_dn   IN VARCHAR2)
RETURN NUMBER;
```

パラメータ

表 8-2 group_exists ファンクションのパラメータ

パラメータ	説明
group_name	グループの単純名です。
group_guid	グループの GUID です。
group_dn	グループ・エントリの識別名です。

戻り値

グループが存在する場合は正数を返します。グループが存在しない場合は 0（ゼロ）を返します。

event_ntfy ファンクション

Oracle Internet Directory でモデル化されたオブジェクトの変更通知イベントを送信するために、Oracle Provisioning Integration Service によって起動されるコールバック・ファンクションです。現在、変更と削除の変更通知イベントは、Oracle Internet Directory 内のユーザーおよびグループに対して送信されます。（Oracle Internet Directory で表される）オブジェクトのイベントを送信する際は、関連する属性が他の詳細とともに送信されます。これらの属性は、属性コンテナのコレクション（配列）として、正規化されていないフォーマットで送信されます。つまり、属性に 2 つの値がある場合は、コレクションの 2 つの行が送信されます。

構文

```
FUNCTION event_ntfy ( event_type  IN VARCHAR2,  
                     event_id    IN VARCHAR2,  
                     event_src   IN VARCHAR2,  
                     event_time  IN VARCHAR2,  
                     object_name IN VARCHAR2,  
                     object_guid IN VARCHAR2,  
                     object_dn   IN VARCHAR2,  
                     profile_id  IN VARCHAR2,  
                     attr_list   IN LDAP_ATTR_LIST )  
RETURN NUMBER;
```

パラメータ

表 8-3 event_ntfy ファンクションのパラメータ

パラメータ	説明
event_type	イベントのタイプ。指定可能な値は、USER_DELETE、USER_MODIFY、GROUP_DELETE、GROUP_MODIFY です。
event_id	イベント ID（変更ログ番号）です。
event_src	このイベントに対して責任のある変更担当者の識別名です。
event_time	このイベントの発生時間です。
object_name	エントリの単純名です。
object_guid	エントリの GUID です。
object_dn	エントリの識別名です。
profile_id	プロビジョニング・エージェントの名前です。
attr_list	エントリの LDAP 属性のコレクションです。

戻り値

正常に終了した場合は正数を戻します。障害時には 0（ゼロ）を戻します。

第III部

コマンドライン・ツール

第III部では、汎用ツールおよび Oracle 固有のツールを含むコマンドライン・ツールについて説明します。第III部は、次の章で構成されています。

- [第9章「コマンドライン・ツールの構文」](#)

コマンドライン・ツールの構文

この章では、LDAP データ交換フォーマット (LDIF) と LDAP コマンドライン・ツールを使用するための構文、使用方法および使用例を紹介します。この章では、次の項目について説明します。

- [LDAP データ交換フォーマット \(LDIF\) の構文](#)
- [コマンドライン・ツールの構文](#)
- [カタログ管理ツールの構文](#)
- [プロビジョニング・サブスクリプション・ツールの構文](#)

LDAP データ交換フォーマット (LDIF) の構文

ディレクトリ・エントリの標準ファイル形式は、次のとおりです。

```
dn: distinguished_name
attribute_type: attribute_value
.
.
.
objectClass: object_class_value
.
.
.
```

プロパティ	値	説明
dn:	RDN,RDN,RDN, ...	相対識別名をカンマで区切ります。
attribute:	attribute_value	この行は、エントリの各属性および複数値属性の各属性値ごとに繰り返します。
objectClass:	object_class_value	この行は、各オブジェクト・クラスごとに繰り返します。

次の例は、従業員のファイル・エントリを示しています。1 行目は識別名です。識別名に続く各行は、属性のニーモニックで始まり、その属性の値が続きます。各エントリが、そのエントリのオブジェクト・クラスを定義する行で終了していることに注意してください。

```
dn: cn=Suzie Smith,ou=Server Technology,o=Acme, c=US
cn: Suzie Smith
cn: SuzieS
sn: Smith
email: ssmith@us.Acme.com
telephoneNumber: 69332
photo:/$ORACLE_HOME/empdir/photog/ssmith.jpg
objectClass: organizational person
objectClass: person
objectClass: top
```

次の例は、組織のファイル・エントリを示しています。

```
dn: o=Acme,c=US
o: Acme
ou: Financial Applications
objectClass: organization
objectClass: top
```


LDIF 形式化の注意事項

次に形式化規則のリストを示します。このリストは、全規則を網羅しているわけではありません。

- 追加対象のエントリに属しているすべての必須属性は、非 NULL 値で LDIF ファイルに記述する必要があります。

ヒント： オブジェクト・クラスの必須属性とオプション属性のタイプを調べるには、Oracle Directory Manager を使用します。『Oracle Internet Directory 管理者ガイド』を参照してください。

- 非表示文字やタブは、BASE64 エンコーディングによる属性値で記述します。
- ファイル内のエントリの間は、空白行で区切る必要があります。
- ファイルには、少なくとも 1 つのエントリが含まれている必要があります。
- 次の行に継続する場合は、継続行を空白またはタブで開始します。
- 個々のエントリの上に空白行を追加してください。
- 写真などのバイナリ・ファイルは、スラッシュ (/) で始まるファイルの絶対アドレスで参照を付けます。
- 識別名には、オブジェクトに対する一意の完全なディレクトリ・アドレスが含まれます。
- 識別名の後にリストされる行には、属性とその値が含まれます。入力ファイルで使われる識別名と属性は、ディレクトリ情報ツリーの既存の構造と一致している必要があります。ディレクトリ情報ツリー内で実装していない属性は、入力ファイルで使わないでください。
- LDIF ファイル内のエントリは、ディレクトリ情報ツリーが上位から下位へ作成されるように順に記述します。エントリがその識別名の上位のエントリに依存している場合は、その子エントリの上に上位エントリを必ず追加してください。
- LDIF ファイル内にスキーマを定義するときは、左カッコと最初のテキストの間、および最後のテキストと右カッコの間に空白を挿入してください。

関連項目： LDIF 形式化規則および LDIF ファイルでのグローバリゼーション・サポートの使用方法については、『Oracle Internet Directory 管理者ガイド』に記載されている各種資料を参照してください。

コマンドライン・ツールの構文

この項では、次のツールの使用方法について説明します。

- [ldapadd 構文](#)
- [ldapaddmt 構文](#)
- [ldapbind 構文](#)
- [ldapcompare 構文](#)
- [ldapdelete 構文](#)
- [ldapmoddn 構文](#)
- [ldapmodify 構文](#)
- [ldapmodifymt 構文](#)
- [ldapsearch 構文](#)

ldapadd 構文

ldapadd コマンドライン・ツールを使用して、エントリとそのオブジェクト・クラス、属性および値をディレクトリに追加できます。既存のエントリに属性を追加するには、[ldapmodify コマンド](#)を使用します。ldapmodify コマンドは、9-15 ページの「[ldapmodify 構文](#)」を参照してください。

関連項目： 入力ファイルを使用してサーバーを構成するために ldapadd を使用する方法は、『Oracle Internet Directory 管理者ガイド』を参照してください。

ldapadd は、次の構文で使います。

```
ldapadd [arguments] -f filename
```

filename は、9-2 ページの「[LDAP データ交換フォーマット \(LDIF\) の構文](#)」で説明する仕様に従って作成された LDIF ファイルの名前です。

次の例では、LDIF ファイル `my_ldif_file.ldi` 内に指定されたエントリを、ディレクトリに追加しています。

```
ldapadd -p 389 -h myhost -f my_ldif_file.ldi
```

オプションの引数	説明
-b	ファイルにバイナリ・ファイル名が含まれていることを指定します。バイナリ・ファイル名はスラッシュで始まります。ツールは、参照先のファイルから実際の値を取り出します。
-c	エラーが発生しても処理を継続する場合に指定します。エラーはレポートされます。(このオプションを使用しない場合、エラーが発生すると <code>ldapadd</code> は停止します。)
-D <i>binddn</i>	ディレクトリに対して認証するときに、 <i>binddn</i> に指定されているエントリとして認証することを指定します。この引数は、 <i>-w password</i> オプションとともに使用します。
-E " <i>character_set</i> "	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバル化・サポートの章を参照してください。
-f <i>filename</i>	LDIF 形式のインポート・データ・ファイルの入力名を指定します。LDIF ファイルのフォーマット方法の詳細は、9-2 ページの「 LDAP データ交換フォーマット (LDIF) の構文 」を参照してください。
-h <i>ldaphost</i>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。
-K	-k と同様ですが、Kerberos バインドの最初のステップのみ実行します。
-k	簡易認証のかわりに、Kerberos 認証を使用して認証します。このオプションを使用可能にするには、Kerberos を定義してコンパイルする必要があります。 有効なチケット認可チケットをすでに所有している必要があります。
-M	ツールに対して、ManageDSAIT コントロールをサーバーに送信するように指示します。ManageDSAIT コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
-n	操作を実際には実行せずに、予測結果を示します。
-O <i>ref_hop_limit</i>	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
-p <i>directory_server_port_number</i>	TCP ポート <i>directory_server_port_number</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート (389) に接続されます。
-P <i>wallet_password</i>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の Secure Sockets Layer (SSL) 接続の場合は必須）。

オプションの引数	説明
-U <i>SSLAuth</i>	SSL 認証モードを指定します。 <ul style="list-style-type: none">1: SSL 認証なし2: サーバー認証3: クライアントとサーバーの認証
-v	冗長モードを指定します。
-V <i>ldap_version</i>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
-w <i>password</i>	接続に必要なパスワードを指定します。
-W <i>wallet_location</i>	Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。 <div>-W "file:/home/my_dir/my_wallet"</div> <div>Windows NT では、このパラメータは次のように設定します。<div>-W "file:C:¥my_dir¥my_wallet"</div></div>

ldapaddmt 構文

ldapaddmt は ldapadd と類似しています。ldapaddmt を使用して、エントリとそのオブジェクト・クラス、属性および値をディレクトリに追加できます。ldapadd と異なるのは、複数のエントリを同時に追加するために複数のスレッドをサポートしている点です。

LDIF エントリの処理中に、ldapaddmt は、カレント・ディレクトリ内の add.log ファイルにエラー・ログを記録します。

ldapaddmt は、次の構文で使用します。

ldapaddmt -T *number_of_threads* -h *host* -p *port* -f *filename*

filename は、9-2 ページの「LDAP データ交換フォーマット (LDIF) の構文」で説明する仕様に従って作成された LDIF ファイルの名前です。

次の例は、5 つの同時スレッドを使用して、ファイル myentries.ldif 内のエントリを処理しています。

ldapaddmt -T 5 -h node1 -p 3000 -f myentries.ldif

注意： 同時スレッドの数が増加すると、LDIF エントリの作成は速くなりますが、システム・リソースはより多く消費されます。

オプションの引数	説明
-b	データ・ファイルにバイナリ・ファイル名が含まれていることを指定します。バイナリ・ファイル名はスラッシュで始まります。ツールは、参照先のファイルから実際の値を取り出します。
-c	エラーが発生しても処理を継続する場合に指定します。エラーはレポートされます。（このオプションを使用しない場合、エラーが発生するとツールは停止します。）
-D <i>binddn</i>	ディレクトリに対して認証するときに、 <i>binddn</i> に指定されているエントリとして認証することを指定します。この引数は、-w <i>password</i> オプションとともに使用します。
-E " <i>character_set</i> "	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバリゼーション・サポートの章を参照してください。
-h <i>ldaphost</i>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。
-K	-k と同様ですが、Kerberos バインドの最初のステップのみ実行します。
-k	簡易認証のかわりに、Kerberos 認証を使用して認証します。このオプションを使用可能にするには、Kerberos を定義してコンパイルする必要があります。 有効なチケット認可チケットをすでに所有している必要があります。
-M	ツールに対して、ManageDSAIT コントロールをサーバーに送信するように指示します。ManageDSAIT コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
-n	操作を実際には実行せずに、予測結果を示します。
-O <i>ref_hop_limit</i>	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
-p <i>ldappport</i>	TCP ポート <i>ldappport</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート（389）に接続されます。
-P <i>wallet_password</i>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。

オプションの引数	説明
-T	エントリを同時に処理するスレッドの数を設定します。
-U <i>SSLAuth</i>	SSL 認証モードを指定します。 <ul style="list-style-type: none">1: SSL 認証なし2: サーバー認証3: クライアントとサーバーの認証
-v	冗長モードを指定します。
-V <i>ldap_version</i>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
-w <i>password</i>	接続に必要なパスワードを指定します。
-W <i>wallet_location</i>	Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。 <div>-W "file:/home/my_dir/my_wallet"</div> <div>Windows NT では、このパラメータは次のように設定します。<div>-W "file:C:¥my_dir¥my_wallet"</div></div>

ldapbind 構文

ldapbind コマンドライン・ツールを使用して、サーバーに対してクライアントを認証できるかどうかを調べることができます。

ldapbind は、次の構文で使⤵します。

ldapbind [*arguments*]

オプションの引数	説明
-D <i>binddn</i>	ディレクトリに対して認証するときに、 <i>binddn</i> に指定されているエントリとして認証することを指定します。この引数は、 <i>-w password</i> オプションとともに使⤵します。
-E " <i>.character_set</i> "	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバリゼーション・サポートの章を参照してください。

オプションの引数	説明
-h <i>ldaphost</i>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。
-n	操作を実際には実行せずに、予測結果を示します。
-p <i>ldapport</i>	TCP ポート <i>ldapport</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート（389）に接続されます。
-P <i>wallet_password</i>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
-U <i>SSLAuth</i>	SSL 認証モードを指定します。 <ul style="list-style-type: none">■ 1: SSL 認証なし■ 2: サーバー認証■ 3: クライアントとサーバーの認証
-V <i>ldap_version</i>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
-w <i>password</i>	接続に必要なパスワードを指定します。
-W <i>wallet_location</i>	Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。 -W "file:/home/my_dir/my_wallet" Windows NT では、このパラメータは次のように設定します。 -W "file:C:¥my_dir¥my_wallet"

ldapcompare 構文

ldapcompare コマンドライン・ツールを使用して、コマンドラインで指定した属性値と、ディレクトリ・エントリの属性値を比較できます。

ldapcompare は、次の構文で使います。

```
ldapcompare [arguments]
```

次の例は、Person Nine のタイトルが associate であるかどうかを通知します。

```
ldapcompare -p 389 -h myhost -b "cn=Person Nine, ou=EuroSinet Suite, o=IMC, c=US" -a title -v associate
```

必須の引数	説明
-a <i>attribute name</i>	比較を実行する属性を指定します。
-b " <i>basedn</i> "	比較を実行するエントリの識別名を指定します。
-v <i>attribute value</i>	比較する属性値を指定します。

オプションの引数	説明
-D <i>binddn</i>	ディレクトリに対して認証するときに、 <i>binddn</i> に指定されているエントリとして認証することを指定します。この引数は、 <i>-w password</i> オプションとともに使用します。
-d <i>debug-level</i>	デバッグ・レベルを設定します。『Oracle Internet Directory 管理者ガイド』のディレクトリ・サーバーの管理の章を参照してください。
-E " <i>character_set</i> "	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバリゼーション・サポートの章を参照してください。.
-f <i>filename</i>	入力ファイル名を指定します。
-h <i>ldaphost</i>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。
-M	ツールに対して、ManageDSAIT コントロールをサーバーに送信するように指示します。ManageDSAIT コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
-O <i>ref_hop_limit</i>	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
-p <i>ldapport</i>	TCP ポート <i>ldapport</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート（389）に接続されます。

オプションの引数	説明
-P <i>wallet_password</i>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
-U <i>SSLAuth</i>	SSL 認証モードを指定します。 <ul style="list-style-type: none">■ 1: SSL 認証なし■ 2: サーバー認証■ 3: クライアントとサーバーの認証
-V <i>ldap_version</i>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
-w <i>password</i>	接続に必要なパスワードを指定します。
-W <i>wallet_location</i>	Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。 <div>-W "file:/home/my_dir/my_wallet"</div> <div>Windows NT では、このパラメータは次のように設定します。<div>-W "file:C:¥my_dir¥my_wallet"</div></div>

ldapdelete 構文

ldapdelete コマンドライン・ツールを使用して、コマンドラインに指定したディレクトリからエントリ全体を削除できます。

ldapdelete は、次の構文で使います。

```
ldapdelete [arguments] ["entry_DN" | -f input_filename]
```

注意： エントリ識別名を指定する場合は、-f オプションを使用しないでください。

次の例では、myhost という名前のホストでポート 389 を使用しています。

```
ldapdelete -p 389 -h myhost "ou=EuroSInet Suite, o=IMC, c=US"
```

オプションの引数	説明
-D <i>binddn</i>	ディレクトリに対して認証するときに、 <i>binddn</i> パラメータに完全識別名を使用します。通常、 <i>-w password</i> オプションとともに使用されます。
-d <i>debug-level</i>	デバッグ・レベルを設定します。『Oracle Internet Directory 管理者ガイド』のディレクトリ・サーバーの管理の章を参照してください。
-E " <i>character_set</i> "	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバリゼーション・サポートの章を参照してください。
-f <i>input_filename</i>	入力ファイル名を指定します。
-h <i>ldaphost</i>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。
-k	簡易認証のかわりに、Kerberos 認証を使用して認証します。このオプションを使用可能にするには、Kerberos を定義してコンパイルする必要があります。 有効なチケット認可チケットをすでに所有している必要があります。
-M	ツールに対して、ManageDSAIT コントロールをサーバーに送信するように指示します。ManageDSAIT コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
-n	削除を実際には実行せずに、予測結果を示します。
-O <i>ref_hop_limit</i>	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
-p <i>ldapport</i>	TCP ポート <i>ldapport</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート（389）に接続されます。
-P <i>wallet_password</i>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
-U <i>SSLAuth</i>	SSL 認証モードを指定します。 <ul style="list-style-type: none"> ■ 1: SSL 認証なし ■ 2: サーバー認証 ■ 3: クライアントとサーバーの認証
-v	冗長モードを指定します。

オプションの引数	説明
<code>-V ldap_version</code>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
<code>-w password</code>	接続に必要なパスワードを指定します。
<code>-W wallet_location</code>	<p>Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。</p> <pre>-W "file:/home/my_dir/my_wallet"</pre> <p>Windows NT では、このパラメータは次のように設定します。</p> <pre>-W "file:C:\my_dir\my_wallet"</pre>

ldapmoddn 構文

ldapmoddn コマンドライン・ツールを使用して、エントリの識別名または相対識別名を変更できます。

ldapmoddn は、次の構文で使用します。

```
ldapmoddn [arguments]
```

次の例では、ldapmoddn を使用して、識別名の相対識別名コンポーネントを "cn=dcpl" から "cn=thanh mai" に変更しています。ポートは 389、myhost という名前のホストを使用しています。

```
ldapmoddn -p 389 -h myhost -b "cn=dcpl,dc=Americas,dc=imc,dc=com" -R "cn=thanh mai"
```

必須の引数	説明
<code>-b "basedn"</code>	変更されるエントリの識別名を指定します。

オプションの引数	説明
<code>-D binddn</code>	ディレクトリに対して認証するときに、binddn に指定されているエントリとして認証します。この引数は、 <code>-w password</code> オプションとともに使用します。
<code>-E "character_set"</code>	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバル化・サポートの章を参照してください。

オプションの引数	説明
-f <i>filename</i>	入力ファイル名を指定します。
-h <i>ldaphost</i>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。
-M	ツールに対して、ManageDSAIT コントロールをサーバーに送信するように指示します。ManageDSAIT コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
-N <i>newparent</i>	相対識別名の新しい親を指定します。
-O <i>ref_hop_limit</i>	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
-p <i>ldapport</i>	TCP ポート <i>ldapport</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート（389）に接続されます。
-P <i>wallet_password</i>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
-r	旧相対識別名を変更エントリ内に値として保持しないことを指定します。この引数が指定されない場合、旧相対識別名は変更エントリ内に属性として保持されます。
-R <i>newrdn</i>	新規相対識別名を指定します。
-U <i>SSLAuth</i>	SSL 認証モードを指定します。 <ul style="list-style-type: none">■ 1: SSL 認証なし■ 2: サーバー認証■ 3: クライアントとサーバーの認証
-V <i>ldap_version</i>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
-w <i>password</i>	接続に必要なパスワードを指定します。

オプションの引数	説明
<code>-W <i>wallet_location</i></code>	Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。 <code>-W "file:/home/my_dir/my_wallet"</code> Windows NT では、このパラメータは次のように設定します。 <code>-W "file:C:¥my_dir¥my_wallet"</code>

ldapmodify 構文

ldapmodify コマンドライン・ツールを使用して、属性を変更できます。

ldapmodify は、次の構文で使います。

`ldapmodify [arguments] -f filename`

filename は、9-2 ページの「[LDAP データ交換フォーマット \(LDIF\) の構文](#)」で説明する仕様に従って作成された LDIF ファイルの名前です。

次の表の引数リストは、すべての引数を網羅しているわけではありません。

オプションの引数	説明
<code>-a</code>	エントリが追加対象で、入力ファイルが LDIF 形式であることを示します。
<code>-b</code>	データ・ファイルにバイナリ・ファイル名が含まれていることを指定します。バイナリ・ファイル名はスラッシュで始まります。
<code>-c</code>	エラーが発生しても処理を継続する場合に指定します。エラーはレポートされます。（このオプションを使用しない場合、エラーが発生すると <code>ldapmodify</code> は停止します。）
<code>-D <i>binddn</i></code>	ディレクトリに対して認証するとき、 <i>binddn</i> に指定されているエントリとして認証することを指定します。この引数は、 <code>-w <i>password</i></code> オプションとともに使用します。
<code>-E "<i>character_set</i>"</code>	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバル化・サポートの章を参照してください。
<code>-h <i>ldaphost</i></code>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。

オプションの引数	説明
-M	ツールに対して、ManageDSAIT コントロールをサーバーに送信するように指示します。ManageDSAIT コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
-n	操作を実際には実行せずに、予測結果を示します。
-o log_file_name	-c オプションとともに使用して、エラーの LDIF エントリをログ・ファイルに書き込みます。ログ・ファイル名の絶対パスを指定する必要があります。
-O ref_hop_limit	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
-p ldapport	TCP ポート ldapport 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート (389) に接続されます。
-P wallet_password	Wallet のパスワードを指定します (サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須)。
-U SSLAuth	SSL 認証モードを指定します。 <ul style="list-style-type: none">■ 1: SSL 認証なし■ 2: サーバー認証■ 3: クライアントとサーバーの認証
-v	冗長モードを指定します。
-V ldap_version	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
-w password	デフォルトの非認証の NULL バインドをオーバーライドします。認証を強制するには、このオプションを -D オプションとともに使用します。
-W wallet_location	Wallet の位置を指定します (サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須)。たとえば Solaris では、このパラメータは次のように設定します。 -W "file:/home/my_dir/my_wallet" Windows NT では、このパラメータは次のように設定します。 -W "file:C:¥my_dir¥my_wallet"

-f フラグを使用して modify、delete および modifyrdn 操作を実行するには、入力ファイル形式に LDIF を使用します (9-2 ページの「[LDAP データ交換フォーマット \(LDIF\) の構文](#)」を参照してください)。仕様は次のとおりです。

複数の変更を行う場合は、入力する各変更の間に、ハイフン (-) のみ含まれた行を挿入します。たとえば、次のようにします。

```
dn:cn=Barbara Fritchey,ou=Sales,o=Oracle,c=US
changetype:modify
add: work-phone
work-phone:510/506-7000
work-phone:510/506-7001
-
delete: home-fax
```

属性値の後の空白など、LDIF 入力ファイルにおける不要な空白は、LDAP 操作が失敗する原因となります。

第1行: 変更レコードの場合は、その1行目にリテラル dn:、その後にエントリの識別名値を記述します。たとえば、次のように記述します。

```
dn:cn=Barbara Fritchey,ou=Sales,o=Oracle,c=US
```

第2行: 変更レコードの場合は、その2行目にリテラル changetype:、その後に変更の種類 (add、delete、modify、modrdn など) を記述します。たとえば、次のように記述します。

```
changetype:modify
```

または

```
changetype:modrdn
```

変更の種類に応じて、次の要件に従って各レコードの残りの部分をフォーマットします。

- **changetype:add**

LDIF 形式を使用します (9-2 ページの「[LDAP データ交換フォーマット \(LDIF\) の構文](#)」を参照してください)。

- **changetype:modify**

この **changetype** に続く行には、前述の第1行で指定したエントリに属する属性に対する変更内容を記述します。属性の変更は、3 種類 (add、delete および replace) を指定できます。それぞれの変更について次に説明します。

- **属性値の追加。** changetype modify のこのオプションは、既存の複数値の属性にさらに値を追加します。属性が存在しない場合は、指定した値で新規属性を追加します。

```
add: attribute name
attribute name: value1
attribute name: value2...
```

たとえば、次のようにします。

```
dn:cn=Barbara Fritchey,ou=Sales,o=Oracle,c=US
changetype:modify
add: work-phone
work-phone:510/506-7000
work-phone:510/506-7001
```

- **値の削除。** `delete` 行のみ記述すると、指定した属性のすべての値が削除されます。属性行を指定した場合は、その属性から特定の値を削除できます。

```
delete: attribute name
[attribute name: value1]
```

たとえば、次のようにします。

```
dn:cn=Barbara Fritchey,ou=Sales,o=Oracle,c=US
changetype:modify
delete: home-fax
```

- **値の置換。** このオプションを使用すると、新しく指定した設定で、属性の値をすべて置換できます。

```
replace:attribute name
[attribute name:value1 ...]
```

`replace` に属性を指定しない場合、ディレクトリは空のセットを追加します。次に、ディレクトリはその空のセットを削除要求と解釈し、エントリから属性を削除することによって対応します。この方法は、存在するかどうか分からない属性を削除する場合に便利です。

たとえば、次のようにします。

```
dn:cn=Barbara Fritchey,ou=Sales,o=Oracle,c=US
changetype:modify
replace: work-phone
work-phone:510/506-7002
```

* `changetype:delete`

この変更タイプは、エントリを削除するときに使用します。第 1 行でエントリを指定し、第 2 行で `changetype` に `delete` を指定しているため、それ以上の入力は必要ありません。

たとえば、次のようにします。

```
dn:cn=Barbara Fritchey,ou=Sales,o=Oracle,c=US
changetype:delete
```

* changetype:modrdn

変更タイプに続く行に、次の形式で新規の相対識別名を指定します。

```
newrdn: RDN
```

たとえば、次のようにします。

```
dn:cn=Barbara Fritchey,ou=Sales,o=Oracle,c=US
changetype:modrdn
newrdn: cn=Barbara Fritchey-Blomberg
```

例 : ldapmodify を使用した属性の追加

この例では、myAttr という新規の属性を追加します。この操作での LDIF ファイルは、次のとおりです。

```
dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: (1.2.3.4.5.6.7 NAME 'myAttr' DESC 'New attribute definition'
EQUALITY caseIgnoreMatch SYNTAX
'1.3.6.1.4.1.1466.115.121.1.15' )
```

1 行目では、識別名を入力して、この新規の属性を配置する場所を指定します。すべての属性とオブジェクト・クラスは、cn=subschemasubentry に格納されます。

2 行目と 3 行目は、新規の属性を追加するための適切な書式を示します。

最後の行は属性の定義です。最初の 1.2.3.4.5.6.7 は、オブジェクトの識別子番号です。この番号は、他のすべてのオブジェクト・クラスと属性に対して一意であることが必要です。次の NAME は、属性の名前です。この例では、属性名は myAttr です。属性名は引用符で囲む必要があります。次は属性の説明です。属性の説明は、引用符で囲んで入力します。この例では、最後に、属性に対するオプションの書式化規則を定義しています。この例の場合、EQUALITY caseIgnoreMatch の一致規則および Directory String の SYNTAX (構文) を追加しています。この例では、Directory String という名前の構文ではなく、オブジェクト ID 番号の 1.3.6.1.4.1.1466.115.121.1.15 を使用しています。

この例のように書式化したファイルに属性情報を入力してください。次に、次のコマンドを実行して、属性を Oracle ディレクトリ・サーバーのスキーマに追加します。

```
ldapmodify -h yourhostname -p 389 -D orcladmin -w "welcome" -v -f /tmp/newattr.ldif
```

この `ldapmodify` コマンドでは、Oracle ディレクトリ・サーバーはポート 389 上で実行されており、スーパー・ユーザーのアカウント名は `orcladmin`、スーパー・ユーザーのパスワードは `welcome`、LDIF ファイルの名前は `newattr.ldif` であることを前提にしています。`yourhostname` には、使用しているコンピュータのホスト名を入力します。

LDIF ファイルがあるディレクトリで作業を行わない場合は、コマンドの最後に LDIF ファイルへの完全ディレクトリ・パスを入力する必要があります。この例では、LDIF ファイルは `/tmp` ディレクトリに格納されています。

ldapmodifymt 構文

`ldapmodifymt` コマンドライン・ツールを使用して、複数のエントリを同時に変更できます。

`ldapmodifymt` は、次の構文で使います。

```
ldapmodifymt -T number_of_threads [arguments] -f filename
```

`filename` は、9-2 ページの「LDAP データ交換フォーマット (LDIF) の構文」で説明する仕様に従って作成された LDIF ファイルの名前です。

関連項目： `ldapmodifymt` で使用されるその他の書式設定の仕様については、9-15 ページの「`ldapmodify` 構文」を参照してください。

たとえば、次のようにします。

```
ldapmodifymt -T 5 -h node1 -p 3000 -f myentries.ldif
```

オプションの引数	説明
-a	エントリが追加対象で、入力ファイルが LDIF 形式であることを示します。(ldapadd を実行している場合、このフラグは必要ありません。)
-b	データ・ファイルにバイナリ・ファイル名が含まれていることを指定します。バイナリ・ファイル名はスラッシュで始まります。
-c	エラーが発生しても処理を継続する場合に指定します。エラーはレポートされます。(このオプションを使用しない場合、エラーが発生すると ldapmodify は停止します。)
-D "binddn"	ディレクトリに対して認証するときに、binddn に指定されているエントリとして認証することを指定します。この引数は、-w password オプションとともに使用します。
-E "character_set"	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバリゼーション・サポートの章を参照してください。

オプションの引数	説明
<code>-h ldaphost</code>	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。
<code>-M</code>	ツールに対して、 <code>ManageDSAIT</code> コントロールをサーバーに送信するように指示します。 <code>ManageDSAIT</code> コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
<code>-n</code>	操作を実際には実行せずに、予測結果を示します。
<code>-O ref_hop_limit</code>	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
<code>-p ldappport</code>	TCP ポート <i>ldappport</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート（389）に接続されます。
<code>-P wallet_password</code>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
<code>-T</code>	エントリを同時に処理するスレッドの数を設定します。
<code>-U SSLAuth</code>	SSL 認証モードを指定します。 <ul style="list-style-type: none"> ■ 1: SSL 認証なし ■ 2: サーバー認証 ■ 3: クライアントとサーバーの認証
<code>-v</code>	冗長モードを指定します。
<code>-V ldap_version</code>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
<code>-w password</code>	デフォルトの非認証の NULL バインドをオーバーライドします。認証を強制するには、このオプションを <code>-D</code> オプションとともに使用します。
<code>-W wallet_location</code>	Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。 <pre>-W "file:/home/my_dir/my_wallet"</pre> <p>Windows NT では、このパラメータは次のように設定します。</p> <pre>-W "file:C:¥my_dir¥my_wallet"</pre>

ldapsearch 構文

ldapsearch コマンドライン・ツールを使用して、ディレクトリ内の特定のエントリを検索して取り出すことができます。

ldapsearch は、次の構文で使います。

```
ldapsearch [arguments] filter [attributes]
```

filter の書式は、RFC-2254 に準拠する必要があります。

関連項目： フィルタの書式の規格については、
<http://www.ietf.org/rfc/rfc2254.txt> を参照してください。

属性は空白で区切ります。属性を何も入力しないと、すべての属性が取り出されます。

必須の引数	説明
-b "basedn"	検索対象のベース識別名を指定します。
-s scope	検索有効範囲を指定します。base、one または sub。

オプションの引数	説明
-A	属性名のみ取り出します（値は取り出しません）。
-a deref	別名参照解除を指定します。never、always、search または find。
-B	非 ASCII 値を出力します。
-D binddn	ディレクトリに対して認証するときに、 <i>binddn</i> に指定されているエントリとして認証することを指定します。この引数は、 <i>-w password</i> オプションとともに使用します。
-d debug level	デバッグ・レベルを指定のレベルに設定します（『Oracle Internet Directory 管理者ガイド』のディレクトリ・サーバーの管理の章を参照してください）。
-E "character_set"	ネイティブ・キャラクタ・セット・エンコーディングを指定します。『Oracle Internet Directory 管理者ガイド』のグローバリゼーション・サポートの章を参照してください。
-f file	<i>file</i> にリストされている検索順を実行します。
-F sep	属性名と値の間に、= ではなく <i>sep</i> を出力します。
-h ldaphost	デフォルトのホスト（ローカル・コンピュータ）ではなく、 <i>ldaphost</i> に接続します。 <i>ldaphost</i> には、コンピュータ名または IP アドレスを指定します。

オプションの引数	説明
-L	エントリを LDIF 形式で出力します（引数 B の内容も含まれます）。
-l <i>timelimit</i>	ldapsearch コマンドが完了するまでの最大待機時間（秒）を指定します。
-M	ツールに対して、ManageDSAIT コントロールをサーバーに送信するように指示します。ManageDSAIT コントロールはサーバーに対して、クライアントに参照を送信しないように指示します。かわりに、参照エントリが正規のエントリとして戻されます。
-n	検索を実際には実行せずに、予測結果を示します。
-O <i>ref_hop_limit</i>	クライアントが処理する必要がある参照ホップ数を指定します。デフォルト値は 5 です。
-p <i>ldapport</i>	TCP ポート <i>ldapport</i> 上のディレクトリに接続します。このオプションを指定しない場合は、デフォルト・ポート（389）に接続されます。
-P <i>wallet_password</i>	Wallet のパスワードを指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。
-S <i>attr</i>	検索結果を属性 <i>attr</i> でソートします。
-t	/tmp のファイルに書き込みます。
-u	わかりやすいエントリ名で出力します。
-U <i>SSLAuth</i>	SSL 認証モードを指定します。 <ul style="list-style-type: none"> ■ 1: SSL 認証なし ■ 2: サーバー認証 ■ 3: クライアントとサーバーの認証
-v	冗長モードを指定します。
-V <i>ldap_version</i>	使用する LDAP プロトコルのバージョンを指定します。デフォルト値は 3 です。これによって、ツールでは LDAP v3 プロトコルが使用されます。この値を 2 に指定すると、LDAP v2 プロトコルが使用されます。
-w <i>bindpassword</i>	バインド・パスワードを指定します（簡易認証の場合）。

オプションの引数	説明
-W <i>wallet_location</i>	Wallet の位置を指定します（サーバー認証、またはクライアントとサーバーの認証の SSL 接続の場合は必須）。たとえば Solaris では、このパラメータは次のように設定します。 -W "file:/home/my_dir/my_wallet" Windows NT では、このパラメータは次のように設定します。 -W "file:C:¥my_dir¥my_wallet"
-z <i>sizelimit</i>	エントリの最大検索数を指定します。

ldapsearch フィルタの例

検索コマンドの作成方法を理解するには、次の例を参考にしてください。

例 1: ベース・オブジェクト検索 次の例は、ルートからディレクトリ上のベース・レベルの検索を実行します。

```
ldapsearch -p 389 -h myhost -b "" -s base -v "objectclass=*" 
```

- -b オプションは、検索するベース識別名を指定します。この場合はルートです。
- -s オプションは、検索がベース検索（base）か、1 レベルの検索（one）か、サブツリー検索（sub）かを指定します。
- "objectclass=*" は、検索時のフィルタを指定します。

例 2: 1 レベルの検索 次の例は、識別名 "ou=HR, ou=Americas, o=IMC, c=US" から開始する 1 レベルの検索を実行します。

```
ldapsearch -p 389 -h myhost -b "ou=HR, ou=Americas, o=IMC, c=US" -s one -v "objectclass=*" 
```

例 3: サブツリー検索 次の例は、サブツリー検索を実行し、"cn=Person" で始まる識別名を持つすべてのエントリを戻します。

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "cn=Person*" 
```

例 4: サイズ制限を使用した検索 次の例では、一致するエントリが 3 つ以上ある場合でも、実際には 2 つのみが取り出されます。

```
ldapsearch -h myhost -p 389 -z 2 -b "ou=Benefits,ou=HR,ou=Americas,o=IMC,c=US" -s one "objectclass=*" 
```

例 5: 指定した属性での検索 次の例では、一致するエントリの DN 属性値のみを戻します。

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "objectclass=*" dn
```

次の例では、surname (sn)、description (description) 属性値とともに、識別名 (dn) 属性値のみが取り出されます。

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "cn=Person*" dn sn description
```

例 6: 指定した属性オプションでのエントリの検索 次の例では、言語コード属性オプションを指定するオプションがある一般名 (cn) 属性を持ったエントリを取り出します。この例では、一般名がフランス語で、R の文字から始まるエントリを取り出します。

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub "cn;lang-fr=R*" 
```

John のエントリには、cn;lang-it 言語コード属性オプションの値が設定されていないものとします。この場合、次の例では John のエントリは戻されません。

```
ldapsearch -p 389 -h myhost -b "c=us" -s sub "cn;lang-it=Giovanni" 
```

例 7: すべてのユーザー属性および指定した操作属性の検索 次の例では、すべてのユーザー属性と、createtimestamp 操作属性および orclguid 操作属性を取り出します。

```
ldapsearch -p 389 -h myhost -b "ou=Benefits,ou=HR,ou=Americas,o=IMC,c=US" -s sub "cn=Person*" * createtimestamp orclguid
```

次の例では、Anne Smith が修正したエントリを取り出します。

```
ldapsearch -h sun1 -b "" "(&(objectclass=*)(modifiersname=cn=Anne Smith))"
```

次の例では、2001 年 4 月 1 日から 2001 年 4 月 6 日の間に修正されたエントリを取り出します。

```
ldapsearch -h sun1 -b "" "(&(objectclass=*)(modifytimestamp>=20000401000000) (modifytimestamp<= 20000406235959))"
```

注意： modifiersname および modifytimestamp は索引付き属性ではないので、catalog.sh を使用してこれら 2 種類の属性に索引付けをします。そして、Oracle Internet Directory サーバーを再起動してから、前述の 2 つの ldapsearch コマンドを発行します。

その他の例： 次の各例は、ホスト sun1 のポート 389 で、識別名 "ou=hr,o=acme,c=us" から開始してサブツリー全体を検索します。

次の例は、objectclass 属性の値を持つエントリをすべて検索します。

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree "objectclass=*"
```

次の例は、objectclass 属性の値が orcle で始まるエントリをすべて検索します。

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree "objectclass=orcle*"
```

次の例は、objectclass 属性が orcle で始まり、cn が foo で始まるエントリを検索します。

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree  
"(&(objectclass=orcle*)(cn=foo*))"
```

次の例は、一般名 (cn) が foo 以外のエントリを検索します。

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree "(!(cn=foo))"
```

次の例は、cn が foo で始まるか、または sn が bar で始まるエントリを検索します。

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree  
"(|(cn=foo*)(sn=bar*))"
```

次の例は、employeenumber が 10,000 以下のエントリを検索します。

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree  
"employeenumber<=10000"
```

カタログ管理ツールの構文

Oracle Internet Directory は、索引を使用して属性を検索できるようにしています。Oracle Internet Directory のインストール時に、エントリ cn=catalogs に、検索で使用できる属性がリストされます。等価の一致規則を持つ属性のみが索引付けできます。

その他の属性を検索フィルタで使用する場合は、使用する属性をカタログ・エントリに追加する必要があります。追加は、Oracle Directory Manager を使用して属性を作成するときに可能です。しかし、属性がすでに存在している場合は、カタログ管理ツールを使用しないと、その属性に索引付けできません。

カタログ管理ツールを実行する前に、LANG 変数の設定を解除してください。カタログ管理ツールの実行終了後、LANG 変数を元の値に設定してください。

LANG の設定を解除する方法は、次のとおりです。

- Korn シェルを使用している場合

```
UNSET LANG
```

- C シェルを使用している場合

```
UNSETENV LANG
```


カタログ管理ツールは、次の構文で使⽤します。

```
catalog.sh -connect net_service_name {add|delete} {-attr attr_name|-file filename}
```

必須の引数	説明
- connect <i>net_service_name</i>	ディレクトリ・データベースに接続するためのネット・サービス名を指定します。
関連項目：『Oracle9i Net Services 管理者ガイド』	

オプションの引数	説明
- add -attr <i>attr_name</i>	指定した属性を索引付けします。
- delete -attr <i>attr_name</i>	指定した属性から索引を削除します。
- add -file <i>filename</i>	指定したファイル内の属性（1 行に 1 つずつ）を索引付けします。
-delete -file <i>filename</i>	指定したファイル内の属性から索引を削除します。

catalog.sh コマンドを⼊力すると、次のメッセージが表示されます。

```
This tool can only be executed if you know the OiD user password.
Enter OiD password:
```

正しいパスワードを⼊力すると、コマンドが実行されます。パスワードに誤りがあると、次のメッセージが表示されます。

```
Cannot execute this tool
```

カタログ管理ツールの実行終了後、LANG 変数を元の値に設定してください。

LANG を設定する方法は、次のとおりです。

- Korn シェルを使⽤している場合


```
SET LANG=appropriate_language; EXPORT LANG
```
- C シェルを使⽤している場合


```
SETENV LANG appropriate_language
```

カタログ管理ツールの実行後にその変更内容を有効にするには、Oracle Internet Directory サーバーを停止して再起動してください。

関連項目： ディレクトリ・サーバーの起動および再起動の方法は、『Oracle Internet Directory 管理者ガイド』の事前⾏する作業の章を参照してください。

プロビジョニング・サブスクリプション・ツールの構文

プロビジョニング・サブスクリプション・ツールを使用して、ディレクトリ内のプロビジョニング・プロファイル・エントリを管理します。このツールを使用して次のアクティビティを実行します。

- 新規プロビジョニング・プロファイルの作成。作成された新規プロビジョニング・プロファイルは、使用可能な状態に設定されるため、DIP で処理できます。
- 既存のプロビジョニング・プロファイルの使用禁止。
- 使用禁止にされているプロビジョニング・プロファイルの使用許可。
- 既存のプロビジョニング・プロファイルの削除。
- 指定したプロビジョニング・プロファイルの現行ステータスの取得。
- 既存のプロビジョニング・プロファイル内にあるすべてのエラーの消去。

プロビジョニング・サブスクリプション・ツールは、プロビジョニング・プロファイル・エントリの位置とスキーマの詳細をツールのコール元から保護します。アプリケーションとサブスクリバの組合せによって、プロビジョニング・プロファイルをコール元の観点から一意に識別します。システムの制約により、1つのサブスクリバの各アプリケーションごとのプロビジョニング・プロファイルは1つのみです。

注意： Windows オペレーティング・システムでシェル・スクリプト・ツールを実行するには、次のいずれかの UNIX エミュレーション・ユーティリティが必要です。

- Cygwin 1.0。次のサイトを参照してください。
<http://sources.redhat.com/cygwin/>
 - MKS Toolkit 5.1 または 6.0。次のサイトを参照してください。
<http://www.datafocus.com/products/>
-

実行可能ファイル名は `oidprovtool` で、`$ORACLE_HOME/bin` に格納されています。

このツールを起動するには、次のコマンドを使用します。

```
oidprovtool param1=param1_value param2=param2_value param3=param3_value ...
```

プロビジョニング・サブスクリプション・ツールでは、次のパラメータを使用できます。

表 9-1 プロビジョニング・サブスクリプション・ツールのパラメータ

名前	操作	必須 (M) / オプション (O)	説明
operation	すべて	M	実行するサブスクリプション操作。このパラメータに適切な値は、create、enable、disable、delete、status および reset です。ツールの起動ごとに 1 つの操作のみ実行できます。
ldap_host	すべて	O	サブスクリプション操作を実行する LDAP サーバーのホスト名。未指定の場合は、デフォルト値の「localhost」が使用されます。
ldap_port	すべて	O	LDAP サーバーが要求をリスニングする TCP/IP ポート。未指定の場合は、デフォルト値の「389」が使用されます。
ldap_user_dn	すべて	M	操作を実行する代替ユーザーの LDAP 識別名。すべてのユーザーに、プロビジョニング・サブスクリプション操作の実行に必要なアクセス権があるとは限りません。プロビジョニング・サブスクリプション操作を実行するアクセス権を LDAP ユーザーに付与または制限する方法は、管理者ガイドを参照してください。
ldap_user_password	すべて	M	操作を実行する代替ユーザーのパスワード。
application_dn	すべて	M	プロビジョニング・サブスクリプション操作の実行対象となるアプリケーションの LDAP 識別名。application_dn パラメータと organization_dn パラメータの組合せによって、サブスクリプション・ツールはプロビジョニング・プロファイルを一意に識別できます。
organization_dn	すべて	M	プロビジョニング・サブスクリプション操作の実行対象となる組織の LDAP 識別名。application_dn パラメータと organization_dn パラメータの組合せによって、サブスクリプション・ツールはプロビジョニング・プロファイルを一意に識別できます。
interface_name	作成のみ	M	PL/SQL パッケージのデータベース・スキーマ名。値の書式は、[Schema].[PACKAGE_NAME] です。

表 9-1 プロビジョニング・サブスクリプション・ツールのパラメータ（続き）

名前	操作	必須 (M) / オプション (O)	説明
interface_type	作成のみ	O	イベントの伝播先インタフェースのタイプ。有効な値は PLSQL です（未指定の場合は、デフォルトとしてこの値が使用されます）。
interface_connect_info	作成のみ	M	データベース接続文字列。この文字列の書式は、[HOST]:[PORT]:[SID]:[USER_ID]:[PASSWORD] です。
interface_version	作成のみ	O	インタフェース・プロトコルのバージョン。有効な値は、1.0 または 1.1 です。1.0 は旧バージョンのインタフェースです。未指定の場合、この値がデフォルトとして使用されます。
interface_additional_info	作成のみ	O	インタフェースに関する追加情報。現在は使用されません。
schedule	作成のみ	O	このプロファイルに関するスケジューリング情報。値は、DIP がこのプロファイルを処理するまでの時間（秒単位）です。未指定の場合は、デフォルト値の 3600 が使用されます。
max_retries	作成のみ	O	プロビジョニング・サービスが、障害が発生したイベントの送信を再試行する回数。未指定の場合は、デフォルト値の 5 が使用されます。
event_subscription	作成のみ	O	このアプリケーションに DIP が通知を送信する対象のイベント。この文字列の書式は、[USER]GROUP:[< 対象のドメイン>]:[DELETE]ADD[MODIFY(< カンマで区切られた属性のリスト>)] です。パラメータを異なる値で複数回リストすると、複数の値を指定できます。未指定の場合は、デフォルトの USER:< 組織の識別名>:DELETEDGROUP:< 組織の識別名>:DELETE が使用されます（つまり、ユーザーとグループの削除通知を組織の識別名の下に送信します）。

Oracle Internet Directory サーバーの プラグイン・フレームワーク

この章では、Oracle Internet Directory サーバーによるカスタム開発を容易にするプラグイン・フレームワークの使用方法について説明します。

この章では、次の項目について説明します。

- [概要](#)
- [前提となる知識](#)
- [概念](#)
- [要件](#)
- [使用モデルと例](#)
- [タイプ定義と使用モデル](#)
- [LDAP サーバーのエラー・コード・リファレンス](#)

概要

Oracle Internet Directory のプラグイン・フレームワークによって、Lightweight Directory Access Protocol (LDAP) の拡張操作の開発が可能になります。たとえば、次のようにします。

- ユーザー情報がディレクトリ・サーバーに格納されていない場合、ユーザーを認証します。
- 特定のカスタム操作を LDAP 操作に連結します。たとえば、一部の LDAP ユーザーは LDAP データ値の妥当性チェックを様々な方法で実行できます。各 `ldapadd` 操作に対して、属性値の妥当性チェックに様々な方法を指定できます。

前提となる知識

Oracle Internet Directory プラグインを開発するには、次の知識が必要です。

- LDAP の一般的な概念
- Oracle Internet Directory
- Oracle Internet Directory の Oracle9i Application Server への統合
- SQL、PL/SQL およびデータベース RPC に関する知識

概念

この項では、プラグインの概念について説明します。

この項では、次の項目について説明します。

- [ディレクトリ・サーバー・プラグインの概要](#)
- [サーバー・プラグイン・フレームワークの概要](#)
- [Oracle Internet Directory でサポートされている操作ベースのプラグイン](#)

ディレクトリ・サーバー・プラグインの概要

Oracle Internet Directory サーバーの機能を拡張するために、ユーザー独自のサーバー・プラグインを記述できます。サーバー・プラグインは、ユーザー独自のファンクションを組み込んだ PL/SQL パッケージ、共有オブジェクト、共有ライブラリまたは Windows NT の動的リンク・ライブラリ (DLL) です (現在のサポート対象は PL/SQL です)。

Oracle Internet Directory サーバーの機能を拡張するユーザー独自のプラグイン・ファンクションは、次のメソッドを使用して記述できます。

- サーバーがデータの LDAP 操作を実行する前に、そのデータを検証できます。
- サーバーによる LDAP 操作が正常に完了した後で、(ユーザーが定義する) アクションを実行できます。
- 拡張操作を定義できます。
- パスワード・ポリシーを定義できます。
- 外部に格納されている資格証明を使用して認証を実行できます。
- ユーザー独自のサーバー・モジュールを定義して、既存のサーバー・モジュールを置換できます。たとえば、ユーザー独自のパスワード構文検査を実装し、Oracle Internet Directory サーバー内に配置できます。
- 特定の属性値を比較する場合に、構文 / 一致規則の代替規則を指定できます。

起動時に、ディレクトリ・サーバーは、ユーザーのプラグイン構成とライブラリをロードし、各種 LDAP 要求の処理中にそのプラグイン・ファンクションをコールします。

サーバー・プラグイン・フレームワークの概要

Oracle Internet Directory サーバーのプラグイン・フレームワークとは、プラグイン・ユーザーがプラグインを開発、構成および適用できる環境です。個々のプラグイン・インスタンスは、プラグイン・モジュールと呼ばれます。

プラグイン・フレームワークには、次のものが含まれます。

- プラグイン構成ツール
- プラグイン・モジュール・インタフェース
- プラグイン LDAP Application Program Interface (API) (ODS.LDAP_PLUGIN パッケージ)

サーバーのプラグイン・フレームワークを使用する手順は、次のとおりです。

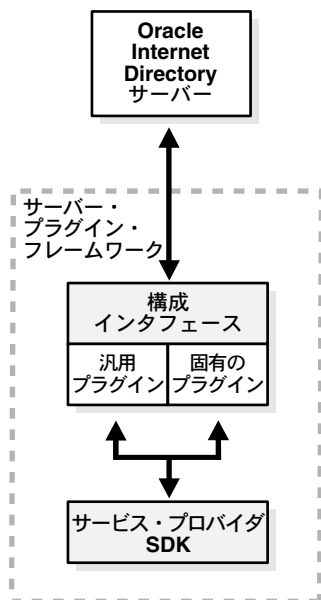
1. ユーザー定義のプラグイン・プロシージャを記述します。このプラグイン・モジュールは、PL/SQL で記述する必要があります。

注意： 現在のサポート対象は PL/SQL です。

2. Oracle Internet Directory のバックエンド・データベースと同じ役割を果たすデータベースに対して、プラグイン・モジュールをコンパイルします。
3. プラグイン・モジュールの実行権限を `ods_server` に付与します。

4. 次の項目が指定されている構成エントリ・インタフェースを使用して、プラグイン・モジュールを登録します。
 - プラグインの名前
 - プラグインのタイプ
5. LDAP サーバーを再起動します。

図 10-1 Oracle Internet Directory サーバーのプラグイン・フレームワーク



Oracle Internet Directory でサポートされている操作ベースのプラグイン

操作ベースのプラグインには、操作前、操作後および操作時の各プラグインがあります。

操作前プラグイン

サーバーは、LDAP 操作を実行する前に、操作前プラグイン・モジュールをコールします。このタイプのプラグインの主な目的は、LDAP 操作で使用する前にデータを検証することです。

操作前プラグインで例外が発生するのは、次のいずれかの場合です。

- リターン・エラー・コードが警告ステータスを示す場合。関連する LDAP 要求は続行されます。
- リターン・コードが障害ステータスを示す場合。要求は続行されません。

関連する LDAP 要求に後で障害が発生した場合、Oracle Internet Directory サーバーは、コミット済みコードをプラグイン・モジュールにロールバックしません。

操作後プラグイン

Oracle Internet Directory サーバーは、LDAP 操作を実行した後に、操作後プラグイン・モジュールをコールします。このタイプのプラグインの主な目的は、特定の操作の実行後にファンクションを起動することです。ログインや通知などが、操作後プラグイン・ファンクションの例です。

操作後プラグインで例外が発生した場合、関連する LDAP 操作はロールバックされません。

関連する LDAP 要求に障害が発生した場合、操作後プラグインはそのまま実行されます。

操作時プラグイン

サーバーは、標準の処理に加え、操作時プラグイン・モジュールをコールします。このタイプのプラグインの主な目的は、既存の機能を補強することです。同一の LDAP トランザクション内の LDAP 操作の一部と考える必要がある特別な操作には、この操作時オプションを使用する必要があります。操作時プラグインは、基本的には関連する LDAP 要求と同一のトランザクション内にあります。LDAP 要求またはプラグイン・プログラムのいずれかに障害が発生した場合は、すべての変更がロールバックされます。

操作時プラグインには、次のタイプがあります。

- 追加
- 置換

たとえば、ldapcompare 操作では、追加時タイプのプラグインを使用できます。Oracle Internet Directory サーバーは、そのサーバー比較コードを実行し、プラグイン開発者が定義したプラグイン・モジュールを実行します。置換時プラグインの場合、Oracle Internet Directory は、それ自体の比較コードは実行せずに、プラグイン・モジュールに従って比較を行い、比較結果を戻します。サーバー比較プロシージャは、プラグイン・モジュールによって置換されます。

置換時プラグインは、ldapcompare および ldapmodify でのみサポートされます。追加時プラグインは、ldapadd、ldapdelete、ldapmodify、ldapcompare、ldapbind および ldapsearch でサポートされます。

要件

この項では、プラグインに関する要件について説明します。

この項では、次の項目について説明します。

- [プラグインの設計](#)
- [プラグインの作成](#)
- [プラグインのコンパイル](#)
- [プラグインの登録](#)
- [プラグインの管理](#)
- [プラグインの有効化と無効化](#)
- [例外処理](#)
- [プラグイン LDAP API](#)
- [プラグインとレプリケーション](#)
- [プラグインと DB ツール](#)
- [セキュリティ](#)

プラグインの設計

プラグインを設計する場合は、次のガイドラインに従います。

- プラグインを使用して、特定の LDAP 操作の実行時に、関連するアクションが確実に実行されるようにします。
- プラグインは、文を発行したユーザーや LDAP アプリケーションに関係なく、プログラムの本体に対して起動される一元化されたグローバル操作に対してのみ使用します。
- 再帰的なプラグインは作成しないでください。たとえば、それ自体が（LDAP PL/SQL API を介して）ldapbind 文を発行する PRE_LDAP_BIND プラグインを作成すると、そのプラグインはリソースがなくなるまで再帰的に実行されます。

注意： LDAP PL/SQL API では、プラグインを慎重に使用してください。プラグインを作成したイベントが発生するたびに、すべての LDAP 要求に対してプラグインが実行されます。

プラグイン操作のタイプ

プラグインは、ldapbind、ldapadd、ldapmodify、ldapcompare、ldapsearch および ldapdelete の各操作に関連付けることができます。

プラグインの名前付け

プラグインの名前（PL/SQL パッケージ名）は、同じデータベース・スキーマ内の他のプラグインやストアド・プロシージャに対して一意であることが必要です。プラグイン名は、表やビューなどの他のデータベース・スキーマ・オブジェクトに対して一意である必要はありません。たとえば、データベース表とプラグインには、同じ名前を指定できます（ただし、混乱を招く可能性があるため、お薦めしません）。

プラグインの作成

プラグイン・モジュールの作成手順は、PL/SQL パッケージを作成する場合と同じです。プラグイン仕様部とプラグイン本体部があります。仕様部は Oracle Internet Directory サーバーとカスタム・プラグインとのインタフェースの役割を担うため、Oracle Internet Directory によって、プラグイン仕様が定義されます。

セキュリティと LDAP サーバーの整合性のために、プラグインをコンパイルできるのは、Oracle Internet Directory サーバーのバックエンド・データベースの役割を担うデータベースに対する Oracle Directory Server (ODS) データベース・スキーマ内のみです。

プラグイン・モジュール・インタフェース・パッケージの仕様

異なるタイプのプラグインには、異なるパッケージ仕様が定義されます。プラグイン・パッケージには名前を指定できます。ただし、プラグイン・プロシージャの各タイプに定義されているシグネチャには従う必要があります。

表 10-1 プラグイン・モジュール・インタフェース

プラグイン項目	ユーザー定義	Oracle Internet Directory で定義
プラグイン・パッケージ名	X	
プラグイン・プロシージャ名		X
プラグイン・プロシージャのシグネチャ		X

関連項目： サンプル・コードは、10-24 ページの「[プラグイン・モジュール・インタフェースの仕様](#)」および 10-18 ページの「[使用モデルと例](#)」を参照してください。

次の表は、操作ベースのプラグインに対する様々な種類のパラメータを示しています。

表 10-2 操作ベースと属性ベースのプラグイン・プロシージャのシグネチャ

起動コンテキスト	プロシージャ名	IN パラメータ	OUT パラメータ
ldapbind 前	PRE_BIND	ldapcontext、bind dn、password	teturn code、error message
ldapbind 時	WHEN_BIND	ldapcontext、bind dn、password	teturn code、error message
ldapbind 後	POST_BIND	ldapcontext、bind result、bind dn、password	teturn code、error message
ldapmodify 前	PRE_MODIFY	ldapcontext、dn、mod structure	teturn code、error message
ldapmodify 時	WHEN_MODIFY	ldapcontext、dn、mod structure	teturn code、error message
ldapmodify 時 (ただし、デフォルト・サーバーの動作を置換)	WHEN_MODIFY_REPLACE	ldapcontext、dn、mod structure	teturn code、error message
ldapmodify 後	POST_MODIFY	ldapcontext、modify result、dn、mod structure	return code、error message
ldapcompare 前	PRE_COMPARE	ldapcontext、dn、attribute、value	return code、error message
ldapcompare 時	WHEN_COMPARE	ldapcontext、dn、attribute、value	return code、error message
ldapcompare 時 (ただし、デフォルト・サーバーの動作を置換)	WHEN_COMPARE_REPLACE	ldapcontext、compare result、dn、attribute、value	compare result、return code、error message
ldapcompare 後	POST_COMPARE	ldapcontext、compare result、dn、attribute、value	return code、error message
ldapadd 前	PRE_ADD	ldapcontext、entry	return code、error message
ldapadd 時	WHEN_ADD	ldapcontext、entry	return code、error message
ldapadd 後	POST_ADD	ldapcontext、add result、entry	return code、error message

表 10-2 操作ベースと属性ベースのプラグイン・プロシージャのシグネチャ (続き)

起動コンテキスト	プロシージャ名	IN パラメータ	OUT パラメータ
ldapdelete 前	PRE_DELETE	ldapcontext、dn	return code、error message
ldapdelete 時	WHEN_DELETE	ldapcontext、dn	return code、error message
ldapdelete 後	POST_DELETE	ldapcontext、delete result、dn	return code、error message
ldapsearch 前	PRE_SEARCH	ldapcontext、base dn、scope、filter	return code、error message
ldapsearch 時	WHEN_SEARCH	ldapcontext、base dn、scope、filter	return code、error message
ldapsearch 後	POST_SEARCH	ldapcontext、search result、base dn、scope、filter	return code、error message

関連項目：

- リターン・コードとエラー・メッセージの有効な値については、10-14 ページの「[エラー処理](#)」を参照してください。
- OUT パラメータのリターン・コードの有効な値については、10-28 ページの「[LDAP サーバーのエラー・コード・リファレンス](#)」を参照してください。
- サポート対象のプロシージャ・シグネチャの詳細は、10-24 ページの「[プラグイン・モジュール・インタフェースの仕様](#)」を参照してください。

プラグインのコンパイル

プラグインのコンパイルは、PL/SQL ストアド・プロシージャとまったく同じです。PL/SQL 無名ブロックは、メモリーにロードされるたびにコンパイルされます。コンパイルは、次の段階で起動されます。

1. 構文検査: PL/SQL の構文がチェックされ、解析ツリーが生成されます。
2. 意味検査: 型がチェックされ、解析ツリーでさらに処理されます。
3. コード生成: P コードが生成されます。

プラグインのコンパイル中にエラーが発生した場合、そのプラグインは作成されません。プラグイン作成時のコンパイル・エラーを参照するには、SQL*Plus または Enterprise Manager で SHOW ERRORS 文を使用するか、あるいは USER_ERRORS ビューでエラーを選択できます。

すべてのプラグイン・モジュールは、ODS データベース・スキーマでコンパイルする必要があります。

依存性

コンパイル済みプラグインには依存性があります。プラグイン本体からコールされる依存オブジェクト（ストアド・プロシージャやファンクションなど）が変更された場合、プラグインは無効になります。依存性の理由から無効となったプラグインは、次回起動するまでに再コンパイルする必要があります。

プラグインの再コンパイル

プラグインを手動で再コンパイルするには、ALTER PACKAGE 文を使用します。たとえば、次の文は my_plugin プラグインを再コンパイルします。

```
ALTER PACKAGE my_plugin COMPILE PACKAGE;
```

権限の付与

プラグイン・モジュールの実行権限を ods_server に付与するには、GRANT EXECUTE 文を使用します。

プラグインの登録

ディレクトリ・サーバーが適切なタイミングでプラグインをコールできるように、プラグインをディレクトリ・サーバーに登録する必要があります。登録するには、プラグインのエントリを cn=plugin、cn=subconfigsubentry に作成します。

orclPluginConfig オブジェクト・クラス

プラグインには、そのオブジェクト・クラスの 1 つとして orclPluginConfig が必要です。このオブジェクト・クラスは構造化オブジェクト・クラスで、そのスーパー・クラスが最上位です。表 10-3 は、プラグインの属性のリストおよび説明を示しています。

表 10-3 プラグインの属性名と属性値

属性名	属性値	必須	オプション
cn	プラグイン・エントリ名。	×	
orclPluginName	プラグイン・パッケージ名。	×	
orclPluginType	次のいずれかの値です。 operational attribute password_policy syntax matchingrule	×	
	関連項目：10-4 ページの「 Oracle Internet Directory でサポートされている操作ベースのプラグイン」		
orclPluginKind	PL/SQL		×
orclPluginEnable	0 = 使用禁止（デフォルト） 1 = 使用可能		×
orclPluginVersion	サポート対象のプラグイン・バージョン番号。		×
orclPluginShareLibLocation	動的リンク・ライブラリのファイル位置。この値が未指定の場合、Oracle Internet Directory サーバーはプラグイン言語を PL/SQL とみなします。		×
orclPluginLDAPOperation	次のいずれかの値です。 ldapcompare ldapmodify ldapbind ldapadd ldapdelete ldapsearch		×
orclPluginTiming	次のいずれかの値です。 pre when post		×

表 10-3 プラグインの属性名と属性値（続き）

属性名	属性値	必須	オプション
orclPluginIsReplace	0 = 使用禁止（デフォルト） 1 = 使用可能 操作時プラグインの場合のみ。		×
orclPluginSubscriberDNList	セミコロンで区切られた識別名のリストで、プラグインの実行を制御します。LDAP 操作のターゲット識別名がリストに含まれている場合は、プラグインが起動されます。		×

コマンドライン・ツールによるプラグイン構成エントリの追加

プラグインは Oracle Internet Directory サーバーに追加する必要があります。その結果、サーバーは適切なタイミングで実行する必要がある追加操作を認識できます。

プラグインが Oracle Internet Directory バックエンド・データベースに対して正常にコンパイルされると、新規エントリが作成され、cn=plugin、cn=subconfigsubentry に配置されます。

次の例では、my_plugin1 という名前の操作ベースのプラグインのエントリが作成されます。LDIF ファイル (my_ldif_file.ldif) は、次のとおりです。

例 1

次の例は、オブジェクトを作成する LDIF ファイルの例です。

```
cn=when_comp,cn=plugin,cn=subconfigsubentry
objectclass=orclPluginConfig
objectclass=top
orclPluginName=my_plugin1
orclPluginType=operational
orclPluginTiming=when
orclPluginLDAPOperation=ldapcompare
orclPluginEnable=1
orclPluginVersion=1.0.1
orclPluginIsReplace=1
cn=when_comp
orclPluginKind=PLSQL
orclPluginSubscriberDNList=dc=COM,c=us;dc=us,dc=oracle,dc=com;dc=org,dc=us;o=IMC,c=US
```


例 2

```
cn=post_mod_plugin, cn=plugin,cn=subconfigsubentry
objectclass=orclPluginConfig
objectclass=top
orclPluginName=my_plugin1
orclPluginType=operational
orclPluginTiming=post
orclPluginLDAPOperation=ldapmodify
orclPluginEnable=1
orclPluginVersion=1.0.1
cn=post_mod_plugin
orclPluginKind=PLSQL
```

次のコマンドを使用して、このファイルをディレクトリに追加します。

```
ldapadd -p 389 -h myhost -D binddn -w password -f my_ldif_file.ldif
```

このエントリをディレクトリに追加すると、ディレクトリ・サーバーはただちにその内容を実行し、コンパイルまたはアクセス権限のエラーをチェックして、そのプラグインを検証します。次に、プラグインに関連するタイミングや LDAP 操作のタイプなど、このプラグインに関する詳細情報を収集します。

注意： プラグイン構成エントリ（cn=plugin、cn=subconfigsubentry など）のメタデータは、一貫性のない状態になることを回避するために、レプリケーション環境ではレプリケートされません。

プラグインの管理

この項では、プラグインの変更とデバッグについて説明します。

プラグインの変更

ストアド・プロシージャと同様、プラグインは明示的に変更できません。プラグインを新しい定義で置換する必要があります。

プラグインを置換する場合は、CREATE PACKAGE 文に OR REPLACE オプションを指定する必要があります。この OR REPLACE オプションによって、既存のプラグインの新規バージョンは、プラグインの元のバージョンに付与されている権限に影響を与えることなく古いバージョンを置換できます。

DROP PACKAGE 文を使用してプラグインを削除してから、CREATE PACKAGE 文を再実行することもできます。

プラグイン名（パッケージ名）が変更されている場合は、新規プラグインを再度登録する必要があります。

プラグインのデバッグ

プラグインは、ストアド・プロシージャで使用可能な同じ機能を使用してデバッグできます。

プラグインの有効化と無効化

プラグインをオンまたはオフに切り替えるには、プラグイン構成オブジェクトの `orclPluginEnable` の値を変更します。たとえば、`cn=post_mod_plugin`、`cn=plugins`、`cn=subconfigsubentry` で `orclPluginEnable` の値を 1 または 0（ゼロ）に変更します。`orclPluginEnable` 値を変更した後は、Oracle Internet Directory サーバーを再起動する必要があります。

例外処理

各プラグイン PL/SQL プロシージャには、エラーを処理し、可能な場合にはリカバリを行うための例外処理ブロックが必要です。

関連項目： PL/SQL プログラミング・ブロックでの例外の使用方法は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

エラー処理

Oracle Internet Directory では、リターン・コード（`rc`）とエラー・メッセージ（`errmsg`）がプラグイン・プロシージャに正しく設定されている必要があります。

リターン・コードにおける有効な値は、次のとおりです。

エラー・コード	説明
0	正常終了
0（ゼロ）より大きい数値	障害（10-28 ページの「 LDAP サーバーのエラー・コード・リファレンス 」も参照）
-1	警告

`errmsg` パラメータは、ユーザーのカスタム・エラー・メッセージを Oracle Internet Directory サーバーに戻すことができる文字列値です。`errmsg` のサイズ制限は、1024 バイトです。Oracle Internet Directory でプラグイン・プログラムが実行されると、Oracle Internet Directory では、実行後にリターン・コードを検査し、エラー・メッセージの表示が必要かどうか判断されます。

たとえば、リターン・コードの値が 0（ゼロ）の場合、エラー・メッセージの値は無視されます。リターン・コードの値が -1 または 0（ゼロ）よりも大きい場合は、次のメッセージがログ・ファイルに記録されるか、標準出力に表示（要求元が LDAP コマンドライン・ツールの場合）されます。

ldap addition info: customized error

Oracle Internet Directory とプラグインの間を処理するプログラム制御

次の表は、プラグイン例外が発生した場合の発生場所とその例外に対する Oracle Internet Directory サーバーの処理を示しています。

表 10-4 プラグイン例外発生時のプログラム制御処理

プラグイン例外の発生場所	Oracle Internet Directory サーバーによる処理
PRE_BIND、PRE_MODIFY、PRE_ADD、PRE_SEARCH、PRE_COMPARE、PRE_DELETE	リターン・コードに従います。 0（ゼロ）より大きい場合（エラー）は、LDAP 操作を実行しません。 -1 の場合（警告）は、LDAP 操作を続行します。
POST_BIND、POST_MODIFY、POST_ADD、POST_SEARCH、WHEN_COMPARE、WHEN_DELETE	LDAP 操作を完了します。ロールバックは行われません。
WHEN_BIND、WHEN_MODIFY、WHEN_ADD、WHEN_SEARCH、WHEN_COMPARE、WHEN_DELETE	LDAP 操作をロールバックします。

次の表は、LDAP 操作に失敗した場合の、障害に対するターゲット・サーバーおよび Oracle Internet Directory サーバーの処理を示しています。

表 10-5 LDAP 操作障害時のプログラム制御処理

LDAP 操作障害の発生場所	Oracle Internet Directory サーバーによる処理
PRE_BIND、PRE_MODIFY、PRE_ADD、PRE_SEARCH、WHEN_COMPARE、WHEN_DELETE	操作前プラグインを完了します。ロールバックは行われません。
POST_BIND、POST_MODIFY、POST_ADD、POST_SEARCH、WHEN_COMPARE、WHEN_DELETE	操作後プラグインが続行されます。LDAP 操作結果は IN パラメータの 1 つです。
WHEN_BIND、WHEN_MODIFY、WHEN_ADD、WHEN_SEARCH、WHEN_COMPARE、WHEN_DELETE	操作時タイプのプラグインの変更はロールバックされます。
操作時置換	プラグイン・プログラム本体への変更はロールバックされます。

プラグイン LDAP API

API アクセスを提供するには、次のように様々な方法があります。

- ユーザーは標準 LDAP PL/SQL API を利用できます。プログラム・ロジックを慎重に計画しないと、プラグインの実行で無限ループが発生する可能性があります。

関連項目： LDAP PL/SQL API の使用に関する情報は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

- Oracle Internet Directory に用意されているプラグイン LDAP API は、該当する LDAP 要求に関連付けられている構成済みのプラグインがある場合、Oracle Internet Directory サーバー内の一連のプラグイン・アクションを実行しません。

プラグイン LDAP API では、プラグイン・モジュール内の同じ Oracle Internet Directory サーバーに再接続するための API が Oracle Internet Directory によって提供されます。つまり、プラグイン・モジュール内で外部の LDAP サーバーに接続する場合は、DBMS_LDAP API を使用できます。このプラグイン自体を実行しているサーバーと同一の Oracle Internet Directory サーバーに接続する場合は、プラグイン LDAP API を使用してバインドおよび認証を行う必要があります。

各プラグイン・モジュールには、Oracle Internet Directory サーバーから渡された `ldapcontext` があります。プラグイン LDAP API をコールする場合は、セキュリティおよびバインドのために、この `ldapcontext` を渡す必要があります。この `ldapcontext` を使用してバインドすると、Oracle Internet Directory サーバーは、この LDAP 要求がプラグイン・モジュールからの要求であることを認識します。このタイプのプラグイン・バインドの場合、Oracle Internet Directory サーバーは後続のプラグインをトリガーしません。Oracle Internet Directory サーバーはこれらのプラグイン・バインドをスーパー・ユーザーのバインドとして処理します。このプラグイン・バインドは慎重に使用してください。

関連項目： コードの例は、10-18 ページの「[プラグイン LDAP API の仕様](#)」を参照してください。

プラグインとレプリケーション

レプリケーション環境では、次のような一貫性のない状態が発生する場合があります。

- プラグイン・メタデータが他のノードにレプリケートされる
- `ldapmodify`、`ldapadd` またはディレクトリのエントリを変更するその他の LDAP 操作がプラグイン・プログラムで使用される
- 関係する一部のノードのみがプラグインをインストールする
- プラグインがディレクトリ・データに依存する特別なチェックを実装する

プラグインと DB ツール

バルク・ツールは、サーバー・プラグインをサポートしていません。

セキュリティ

一部の Oracle Internet Directory サーバーのプラグインでは、厳重なセキュリティを保持するコードをユーザーが用意する必要があります。たとえば、Oracle Internet Directory の `ldapcompare` または `ldapbind` 操作をユーザー独自のプラグイン・モジュールで置換する場合、ユーザーは、セキュリティを維持するための機能が、この操作の実装によって除外されないことを確認する必要があります。

厳重なセキュリティを確保するには、次の処理を行う必要があります。

- プラグイン・パッケージを作成します。
- LDAP 管理者のみがデータベース・ユーザーを制限できるように指定します。
- Access Control List (ACL) を使用して、LDAP 管理者のみがプラグイン構成エントリにアクセスできるように設定します。
- 異なる複数のプラグイン間におけるプログラムの関連性に注意します。

プラグイン LDAP API の仕様

```
CREATE OR REPLACE PACKAGE LDAP_PLUGIN AS
    SUBTYPE SESSION IS RAW(32);
    -- Initializes the LDAP library and return a session handler
    -- for use in subsequent calls.
    FUNCTION init (ldappluginctx IN ODS.plugincontext)
        RETURN SESSION;
    -- Synchronously authenticates to the directory server using
    -- a Distinguished Name and password.
    FUNCTION simple_bind_s (ldappluginctx IN ODS.plugincontext,
                           ld              IN SESSION)
        RETURN PLS_INTEGER;
END LDAP_PLUGIN;
```

使用モデルと例

この項では、問合せロギングを検索する場合、および2つのディレクトリ情報ツリー（DIT）を同期させる場合の2つの例を示します。

例 1: 問合せロギングの検索

状況:すべての ldapsearch コマンドを記録できるかどうかについて、あるユーザーが疑問を持っています。

解答:記録できます。ldapsearch 操作後プラグインを使用して、ユーザーは ldapsearch コマンドのすべてを記録できます。すべての ldapsearch 要求を記録するか、(特定のサブツリー下の) 特定の識別名で検索が発生した場合の ldapsearch 要求すべてを記録できます。

すべての ldapsearch コマンドを記録するには、次の手順を実行します。

1. 準備を行います。

すべての ldapsearch 結果をデータベース表に記録します。このログ表には、次の列が必要です。

- タイムスタンプ
- ベース識別名
- 検索有効範囲
- 検索フィルタ
- 必須属性
- 検索結果

表を作成するには、次の SQL スクリプトを使用します。

```
drop table search_log;
create table search_log
  (timestamp varchar2(50),
   basedn varchar2(256),
   searchscope number(1);
   searchfilter varchar2(256);
   searchresult number(1));
drop table simple_tab;
create table simple_tab (id NUMBER(7), dump varchar2(256));
DROP sequence seq;
CREATE sequence seq START WITH 10000;
commit;
```

2. プラグイン・パッケージ仕様を作成します。

```
CREATE OR REPLACE PACKAGE LDAP_PLUGIN_EXAMPLE1 AS
PROCEDURE post_search
  (ldapplugincontext IN ODS.plugincontext,
   result             IN INTEGER,
   basedN             IN VARCHAR2,
   scope              IN INTEGER,
   filterStr          IN VARCHAR2,
   requiredAttr       IN ODS.strCollection,
   rc                 OUT INTEGER,
   errmsg             OUT VARCHAR2
  );
END LDAP_PLUGIN_EXAMPLE1;
```

3. プラグイン・パッケージ本体を作成します。

```
CREATE OR REPLACE PACKAGE BODY LDAP_PLUGIN_EXAMPLE1 AS
PROCEDURE post_search
  (ldapplugincontext IN ODS.plugincontext,
   result             IN INTEGER,
   basedN             IN VARCHAR2,
   scope              IN INTEGER,
   filterStr          IN VARCHAR2,
   requiredAttr       IN ODS.strCollection,
   rc                 OUT INTEGER,
   errmsg             OUT VARCHAR2
  )
  IS
BEGIN
  INSERT INTO simple_tab VALUES
    (to_char(sysdate, 'Month DD, YYYY HH24:MI:SS'), basedN, scope,
     filterStr, result);
```

```
-- The following code segment demonstrate how to iterate
-- the ODS.strCollection
FOR l_counter1 IN 1..requiredAttr.COUNT LOOP
    INSERT INTO simple_tab
        values (seq.NEXTVAL, 'req attr ' || l_counter1 || ' = ' ||
            requiredAttr(l_counter1));
END LOOP;
rc := 0;
errmsg := 'no post_search plguin error msg';
COMMIT;
EXCEPTION
    WHEN others THEN
        rc := 1;
        errmsg := 'exception: post_search plguin';
END;
END LDAP_PLUGIN_EXAMPLE1;
/
```

4. ods_server に権限を付与します。

```
GRANT EXECUTE ON LDAP_PLUGIN_EXAMPLE1 TO ods_server;
```

5. プラグイン・エントリを Oracle Internet Directory サーバーに登録します。

次のように、LDIF ファイル (register_post_search.ldif) を構成します。

```
cn=post_search,cn=plugin,cn=subconfigsubentry
objectclass=orclPluginConfig
objectclass=top
orclPluginName=ldap_plugin_example1
orclPluginType=operational
orclPluginTiming=post
orclPluginLDAPOperation=ldapsearch
orclPluginEnable=1
orclPluginVersion=1.0.1
cn=post_search
orclPluginKind=PLSQL
```

ldapadd コマンドライン・ツールを使用して、このエントリを追加します。

```
% ldapadd -p port_number -h host_name -D bind_dn -w passwd -v -f register_post_
search.ldif
```

6. Oracle Internet Directory サーバーを再起動します。

例 2: 2 つのディレクトリ情報ツリーの同期化

状況: cn=Products、cn=oraclecontext に依存する 2 つの製品があり、これらの製品内のユーザーは、Oracle Internet Directory で 1 対 1 の関係にあります。最初のディレクトリ情報ツリー（製品 1）内のユーザーが削除された場合は、別のディレクトリ情報ツリー（製品 2）内の対応するユーザーを削除する必要があります（これらのユーザー間には関連性があるため）。

最初のディレクトリ情報ツリーのユーザーを削除するイベントによって、2 番目のディレクトリ情報ツリーのユーザーを削除するトリガーをコールまたは渡すように、Oracle Internet Directory 内にトリガーを設定する方法はありますか。

解答: あります。ldapdelete 操作後プラグインを使用して、2 番目のディレクトリ情報ツリーで発生する 2 番目の削除を処理できます。

最初のディレクトリ情報ツリーに cn=DIT1、cn=products、cn=oraclecontext のネーミング・コンテキストがあり、2 番目のディレクトリ情報ツリーに cn=DIT2、cn=products、cn=oraclecontext のネーミング・コンテキストがある場合、異なる複数ディレクトリ情報ツリーでの 2 ユーザーは、同一の ID 属性を共有します。基本的には、LDAP_PLUGIN と DBMS_LDAP の API を ldapdelete 後のプラグイン・モジュール内で使用して、2 番目のディレクトリ情報ツリー内の対応するユーザーを削除します。

orclPluginSubscriberDNList を cn=DIT1、cn=products、cn=oraclecontext に設定する必要があります。これによって、cn=DIT1、cn=products、cn=oraclecontext でエントリを削除すると、常にプラグイン・モジュールが起動されます。

1. 準備を行います。

両方のディレクトリ情報ツリーのエントリはディレクトリに追加されていると仮定します。たとえば、エントリ id=12345、cn=DIT1、cn=products、cn=oraclecontext は DIT1 に、エントリ id=12345、cn=DIT2、cn=products、cn=oraclecontext は DIT2 にあります。

2. プラグイン・パッケージ仕様を作成します。

```
CREATE OR REPLACE PACKAGE LDAP_PLUGIN_EXAMPLE2 AS
PROCEDURE post_delete
(
  ldapplugincontext IN ODS.plugincontext,
  result            IN INTEGER,
  dn                IN VARCHAR2,
  rc                OUT INTEGER,
  errormsg          OUT VARCHAR2
);
END LDAP_PLUGIN_EXAMPLE2;
/
```

3. プラグイン・パッケージ本体を作成します。

```
CREATE OR REPLACE PACKAGE BODY LDAP_PLUGIN_EXAMPLE2 AS
PROCEDURE post_delete
    (ldapplugincontext IN ODS.plugincontext,
    result IN INTEGER,
    dn IN VARCHAR2,
    rc OUT INTEGER,
    errmsg OUT VARCHAR2
    )
IS
    retval PLS_INTEGER;
    my_session DBMS_LDAP.session;
    newDN VARCHAR2(256);
BEGIN
    retval := -1;
    my_session := LDAP_PLUGIN.init(ldapplugincontext);
    -- bind to the directory
    retval := LDAP_PLUGIN.simple_bind_s(ldapplugincontext, my_session);
    -- if retval is not 0, then raise exception
    newDN := REPLACE(dn, 'DIT1', 'DIT2');
    retval := DBMS_LDAP.delete_s(my_session, newDN);
    -- if retval is not 0, then raise exception
    rc := 0;
    errmsg := 'no post_delete plugin error msg';
EXCEPTION
    WHEN others THEN
        rc := 1;
        errmsg := 'exception: post_delete plugin';
END;
END LDAP_PLUGIN_EXAMPLE2;
/
```

4. プラグイン・エントリを Oracle Internet Directory サーバーに登録します。

LDIF ファイル (register_post_delete.ldif) を次のように構成します。

```
cn=post_delete,cn=plugin,cn=subconfigsubentry
objectclass=orclPluginConfig
objectclass=top
orclPluginName=ldap_plugin_example2
orclPluginType=operational
orclPluginTiming=post
orclPluginLDAPOperation=ldapdelete
orclPluginEnable=1
orclPluginSubscriberDNList=cn=DIT1,cn=oraclecontext,cn=products
orclPluginVersion=1.0.1
cn=post_delete
```

```
orclPluginKind=PLSQL
```

ldapadd コマンドライン・ツールを使用して、次のエントリを追加します。

```
% ldapadd -p port_number -h host_name -D bind_dn -w passwd -v -f register_post_
delete.ldif
```

5. Oracle Internet Directory サーバーを再起動します。

タイプ定義と使用モデル

この項では、データベース・オブジェクトの定義および LDAP_PLUGIN API 仕様の例を示します。

この項では、次の項目について説明します。

- [データベース・オブジェクト・タイプの定義](#)
- [プラグイン・モジュール・インタフェースの仕様](#)

データベース・オブジェクト・タイプの定義

この項では、プラグイン LDAP API で紹介されているオブジェクト・タイプのオブジェクト定義を示します。これらの定義は、すべて ODS のデータベース・スキーマにあります。

```
create or replace type strCollection as TABLE of VARCHAR2(512);
/
```

```
create or replace type pluginContext as TABLE of VARCHAR2(512);
/
```

```
create or replace type attrvalType as TABLE OF VARCHAR2(4000);
/
```

```
create or replace type attrobj as object (
  attrname varchar2(2000),
  attrval attrvalType
);
/
```

```
create or replace type attrlist as table of attrobj;
/
```

```
create or replace type entryobj as object (
  entryname varchar2(2000),
  attr      attrlist
);
/
```

```
create or replace type entrylist as table of entryobj;
/

create or replace type bvalobj as object (
length integer,
val    varchar2(4000)
);
/

create or replace type bvallist as table of bvalobj;
/

create or replace type modobj as object (
operation integer,
type        varchar2(256),
vals        bvallist
);
/

create or replace type modlist as table of modobj;
/
```

プラグイン・モジュール・インタフェースの仕様

ldapbind、ldapsearch、ldapdelete、ldapadd、ldapcompare および ldapmodify の各プラグインを使用するには、次のプロシージャ・シグネチャに従う必要があります。

```
CREATE or replace PACKAGE plugin_test1 AS

PROCEDURE pre_add (ldapplugincontext IN ODS.plugincontext,
    dn          IN VARCHAR2,
    entry       IN ODS.entryobj,
    rc          OUT INTEGER,
    errormsg    OUT VARCHAR2
);

PROCEDURE when_add (ldapplugincontext IN ODS.plugincontext,
    dn          IN VARCHAR2,
    entry       IN ODS.entryobj,
    rc          OUT INTEGER,
    errormsg    OUT VARCHAR2
);

PROCEDURE post_add (ldapplugincontext IN ODS.plugincontext,
    result      IN INTEGER,
    dn          IN VARCHAR2,
    entry       IN ODS.entryobj,
```

```
rc          OUT INTEGER,
errmsg OUT VARCHAR2
);

PROCEDURE pre_modify (ldapplugincontext IN ODS.plugincontext,
dn          IN VARCHAR2,
mods        IN ODS.modlist,
rc          OUT INTEGER,
errmsg OUT VARCHAR2
);

PROCEDURE when_modify (ldapplugincontext IN ODS.plugincontext,
dn          IN VARCHAR2,
mods        IN ODS.modlist,
rc          OUT INTEGER,
errmsg OUT VARCHAR2
);

PROCEDURE when_modify_replace (ldapplugincontext IN ODS.plugincontext,
dn          IN VARCHAR2,
mods        IN ODS.modlist,
rc          OUT INTEGER,
errmsg OUT VARCHAR2
);

PROCEDURE post_modify (ldapplugincontext IN ODS.plugincontext,
result      IN INTEGER,
dn          IN VARCHAR2,
mods        IN ODS.modlist,
rc          OUT INTEGER,
errmsg OUT VARCHAR2
);

PROCEDURE pre_compare (ldapplugincontext IN ODS.plugincontext,
dn          IN VARCHAR2,
attrname IN VARCHAR2,
attrval  IN VARCHAR2,
rc          OUT INTEGER,
errmsg OUT VARCHAR2
);

PROCEDURE when_compare (ldapplugincontext IN ODS.plugincontext,
dn          IN VARCHAR2,
attrname IN VARCHAR2,
attrval  IN VARCHAR2,
rc          OUT INTEGER,
errmsg OUT VARCHAR2
```

```
);

PROCEDURE when_compare_replace (ldapplugincontext IN ODS.plugincontext,
    result OUT INTEGER,
    dn      IN VARCHAR2,
    attrname IN VARCHAR2,
    attrval IN VARCHAR2,
    rc      OUT INTEGER,
    errormsg OUT VARCHAR2
);

PROCEDURE post_compare (ldapplugincontext IN ODS.plugincontext,
    result IN INTEGER,
    dn      IN VARCHAR2,
    attrname IN VARCHAR2,
    attrval IN VARCHAR2,
    rc      OUT INTEGER,
    errormsg OUT VARCHAR2
);

PROCEDURE pre_delete (ldapplugincontext IN ODS.plugincontext,
    dn      IN VARCHAR2,
    rc      OUT INTEGER,
    errormsg OUT VARCHAR2
);

PROCEDURE when_delete (ldapplugincontext IN ODS.plugincontext,
    dn      IN VARCHAR2,
    rc      OUT INTEGER,
    errormsg OUT VARCHAR2
);

PROCEDURE post_delete (ldapplugincontext IN ODS.plugincontext,
    result IN INTEGER,
    dn      IN VARCHAR2,
    rc      OUT INTEGER,
    errormsg OUT VARCHAR2
);

PROCEDURE pre_search (ldapplugincontext IN ODS.plugincontext,
    basedN      IN VARCHAR2,
    scope        IN INTEGER,
    filterStr    IN VARCHAR2,
    requiredAttr IN ODS.strCollection,
    rc           OUT INTEGER,
    errormsg     OUT VARCHAR2
);
```

```
PROCEDURE when_search (ldapplugincontext IN ODS.plugincontext,
    basedN      IN  VARCHAR2,
    scope       IN  INTEGER,
    filterStr    IN  VARCHAR2,
    requiredAttr IN  ODS.strCollection,
    rc          OUT INTEGER,
    errormsg    OUT VARCHAR2
);

PROCEDURE post_search (ldapplugincontext IN ODS.plugincontext,
    result      IN  INTEGER,
    basedN      IN  VARCHAR2,
    scope       IN  INTEGER,
    filterStr    IN  VARCHAR2,
    requiredAttr IN  ODS.strCollection,
    rc          OUT INTEGER,
    errormsg    OUT VARCHAR2
);

PROCEDURE pre_bind (ldapplugincontext IN ODS.plugincontext,
    dnIN VARCHAR2,
    passwdIN VARCHAR2,
    rcOUT INTEGER,
    errormsgOUT VARCHAR2
);

PROCEDURE when_bind (ldapplugincontext IN ODS.plugincontext,
    dnIN VARCHAR2,
    passwdIN VARCHAR2,
    rcOUT INTEGER,
    errormsgOUT VARCHAR2
);

PROCEDURE post_bind (ldapplugincontext IN ODS.plugincontext,
    resultIN INTEGER,
    dnIN VARCHAR2,
    passwdIN VARCHAR2,
    rcOUT INTEGER,
    errormsg OUT VARCHAR2
);

END plugin_test1;
/
```

LDAP サーバーのエラー・コード・リファレンス

```
-----  
---Package specification for DBMS_LDAP  
---      This is the primary interface used by various clients to  
---      make LDAP requests  
-----
```

```
CREATE OR REPLACE PACKAGE DBMS_LDAP AS
```

```
-- ...
```

```
-- possible error codes we can return from LDAP server  
--
```

SUCCESS	CONSTANT NUMBER := 0;
OPERATIONS_ERROR	CONSTANT NUMBER := 1;
PROTOCOL_ERROR	CONSTANT NUMBER := 2;
TIMELIMIT_EXCEEDED	CONSTANT NUMBER := 3;
SIZELIMIT_EXCEEDED	CONSTANT NUMBER := 4;
COMPARE_FALSE	CONSTANT NUMBER := 5;
COMPARE_TRUE	CONSTANT NUMBER := 6;
STRONG_AUTH_NOT_SUPPORTED	CONSTANT NUMBER := 7;
STRONG_AUTH_REQUIRED	CONSTANT NUMBER := 8;
PARTIAL_RESULTS	CONSTANT NUMBER := 9;
REFERRAL	CONSTANT NUMBER := 10;
ADMINLIMIT_EXCEEDED	CONSTANT NUMBER := 11;
UNAVAILABLE_CRITIC	CONSTANT NUMBER := 12;
NO_SUCH_ATTRIBUTE	CONSTANT NUMBER := 16;
UNDEFINED_TYPE	CONSTANT NUMBER := 17;
INAPPROPRIATE_MATCHING	CONSTANT NUMBER := 18;
CONSTRAINT_VIOLATION	CONSTANT NUMBER := 19;
TYPE_OR_VALUE_EXISTS	CONSTANT NUMBER := 20;
INVALID_SYNTAX	CONSTANT NUMBER := 21;
NO_SUCH_OBJECT	CONSTANT NUMBER := 32;
ALIAS_PROBLEM	CONSTANT NUMBER := 33;
INVALID_DN_SYNTAX	CONSTANT NUMBER := 34;
IS_LEAF	CONSTANT NUMBER := 35;
ALIAS_DEREF_PROBLEM	CONSTANT NUMBER := 36;
INAPPROPRIATE_AUTH	CONSTANT NUMBER := 48;
INVALID_CREDENTIALS	CONSTANT NUMBER := 49;
INSUFFICIENT_ACCESS	CONSTANT NUMBER := 50;
BUSY	CONSTANT NUMBER := 51;
UNAVAILABLE	CONSTANT NUMBER := 52;
UNWILLING_TO_PERFORM	CONSTANT NUMBER := 53;
LOOP_DETECT	CONSTANT NUMBER := 54;
NAMING_VIOLATION	CONSTANT NUMBER := 64;
OBJECT_CLASS_VIOLATION	CONSTANT NUMBER := 65;
NOT_ALLOWED_ON_NONLEAF	CONSTANT NUMBER := 66;
NOT_ALLOWED_ON_RDN	CONSTANT NUMBER := 67;
ALREADY_EXISTS	CONSTANT NUMBER := 68;

NO_OBJECT_CLASS_MODS	CONSTANT NUMBER := 69;
RESULTS_TOO_LARGE	CONSTANT NUMBER := 70;
OTHER	CONSTANT NUMBER := 80;
SERVER_DOWN	CONSTANT NUMBER := 81;
LOCAL_ERROR	CONSTANT NUMBER := 82;
ENCODING_ERROR	CONSTANT NUMBER := 83;
DECODING_ERROR	CONSTANT NUMBER := 84;
TIMEOUT	CONSTANT NUMBER := 85;
AUTH_UNKNOWN	CONSTANT NUMBER := 86;
FILTER_ERROR	CONSTANT NUMBER := 87;
USER_CANCELLED	CONSTANT NUMBER := 88;
PARAM_ERROR	CONSTANT NUMBER := 89;
NO_MEMORY	CONSTANT NUMBER := 90;

}

この付録では、サンプル・コードを提供します。

この項では、次の項目について説明します。

- データベース・トリガーからの DBMS_LDAP の使用
- 検索用の DBMS_LDAP の使用
- Java サンプル・コード

データベース・トリガーからの DBMS_LDAP の使用

DBMS_LDAP API をデータベース・トリガーからコールすると、データベースの表に加えられた変更とエンタープライズ・ワイドの LDAP サーバーを同期化することができます。次の例は、挿入、更新および削除のトリガーを使用して、EMP という表に加えられた変更を LDAP サーバーと同期化する方法を示したものです。この例には 2 つの関連ファイルがあります。

- ファイル `trigger.sql` では、表およびその表に関連するトリガーが作成されます。
- ファイル `empdata.sql` では、EMP 表にサンプル・データが挿入されます。EMP 表は、挿入トリガーにより、LDAP サーバーに対して自動的に更新されます。

これらの 2 つのファイルは、`$ORACLE_HOME/ldap/demo` の下位の `plssql` ディレクトリにあります。

trigger.sql ファイル

この SQL ファイルは「EMP」と呼ばれるデータベース表を作成し、表に対するすべての変更を LDAP サーバーで同期させる LDAP_EMP と呼ばれるトリガーをその表に作成します。データベース表への変更は、DBMS_LDAP パッケージを使用して LDAP ディレクトリに反映またはレプリケートされます。

このスクリプトは、次の設定を想定しています。

- LDAP サーバー・ホスト名: NULL (ローカル・ホスト)
- LDAP サーバー・ポート番号: 389
- 従業員レコードのディレクトリ・コンテナ: `o=acme, dc=com`
- ディレクトリ更新のユーザー名 / パスワード: `cn=orcladmin/welcome`

前述の変数は、後述のコードで適切な変数を変更することによって、異なる環境に対応するようにカスタマイズできます。

表定義 データベース表 (EMP) 内の従業員詳細 (列)

EMP_ID	NUMBER
FIRST_NAME	VARCHAR2
LAST_NAME	VARCHAR2
MANAGER_ID	NUMBER
PHONE_NUMBER	VARCHAR2
MOBILE	VARCHAR2
ROOM_NUMBER	VARCHAR2
TITLE	VARCHAR2

LDAP スキーマの定義と関連スキーマ EMP へのマッピング

LDAP ディレクトリでの対応するデータの表現

DN	cn=FIRST_NAME LAST_NAME, o=acme, dc=com
cn	FIRST_NAME LAST_NAME
sn	LAST_NAME
givenname	FIRST_NAME
manager	DN
telephonenumber	PHONE_NUMBER
mobile	MOBILE
employeeNumber	EMP_ID
userpassword	FIRST_NAME
objectclass	person
	organizationalperson
	inetOrgPerson
	top

-Creating EMP table

```
PROMPT Dropping Table EMP ..
drop table EMP;
```

```
PROMPT Creating Table EMP ..
CREATE TABLE EMP (
    EMP_ID      NUMBER, Employee Number
    FIRST_NAME  VARCHAR2(256), First Name
    LAST_NAME   VARCHAR2(256), Last Name
    MANAGER_ID  NUMBER, Manager Number
    PHONE_NUMBER VARCHAR2(256), Telephone Number
    MOBILE      VARCHAR2(256), Mobile Number
    ROOM_NUMBER VARCHAR2(256), Room Number
    TITLE       VARCHAR2(256) Title in the company
);
```

-Creating Trigger LDAP_EMP

```
PROMPT Creating Trigger LDAP_EMP ..
```

```
CREATE OR REPLACE TRIGGER LDAP_EMP
AFTER INSERT OR DELETE OR UPDATE ON EMP
```

```

FOR EACH ROW

DECLARE
    retval    PLS_INTEGER;
    emp_session DBMS_LDAP.session;
    emp_dn    VARCHAR2(256);
    emp_rdn   VARCHAR2(256);
    emp_array DBMS_LDAP.MOD_ARRAY;
    emp_vals  DBMS_LDAP.STRING_COLLECTION ;
    ldap_host VARCHAR2(256);
    ldap_port VARCHAR2(256);
    ldap_user VARCHAR2(256);
    ldap_passwd VARCHAR2(256);
    ldap_base VARCHAR2(256);
BEGIN

    retval      := -1;
    -- Customize the following variables as needed
    ldap_host   := NULL;
    ldap_port   := '389';
    ldap_user   := 'cn=orcladmin';
    ldap_passwd:= 'welcome';
    ldap_base   := 'o=acme,dc=com';
    -- end of customizable settings

    DBMS_OUTPUT.PUT('Trigger [LDAP_EMP]: Replicating changes ');
    DBMS_OUTPUT.PUT_LINE('to directory .. ');
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP Host ',25,' ') || ': ' || ldap_host);
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP Port ',25,' ') || ': ' || ldap_port);

    -- Choosing exceptions to be raised by DBMS_LDAP library.
    DBMS_LDAP.USE_EXCEPTION := TRUE;

    -- Initialize ldap library and get session handle.
    emp_session := DBMS_LDAP.init(ldap_host,ldap_port);

    DBMS_OUTPUT.PUT_LINE (RPAD('Ldap session ',25,' ') || ': ' ||
        RAWTOHEX(SUBSTR(emp_session,1,8)) ||
        '(returned from init)');

    -- Bind to the directory
    retval := DBMS_LDAP.simple_bind_s(emp_session,
        ldap_user,ldap_passwd);

    DBMS_OUTPUT.PUT_LINE(RPAD('simple_bind_s Returns ',25,' ') || ': ' ||
        TO_CHAR(retval));

```

```
-- Process New Entry in the database

IF INSERTING THEN

    -- Create and setup attribute array for the New entry
    emp_array := DBMS_LDAP.create_mod_array(14);

    -- RDN to be - cn="FIRST_NAME LAST_NAME"

    emp_vals(1) := :new.FIRST_NAME || ' ' || :new.LAST_NAME;

    DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
    'cn',emp_vals);

    emp_vals(1) := :new.LAST_NAME;

    DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
    'sn',emp_vals);

    emp_vals(1) := :new.FIRST_NAME;

    DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
    'givenname',emp_vals);

    emp_vals(1) := 'top';
    emp_vals(2) := 'person';
    emp_vals(3) := 'organizationalPerson';
    emp_vals(4) := 'inetOrgPerson';

    DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
    'objectclass',emp_vals);

    emp_vals.DELETE;
    emp_vals(1) := :new.PHONE_NUMBER;

    DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
    'telephonenumber',emp_vals);

    emp_vals(1) := :new.MOBILE;

    DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
    'mobile',emp_vals);

    emp_vals(1) := :new.ROOM_NUMBER;
```

```
DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
                             'roomNumber',emp_vals);

emp_vals(1) := :new.TITLE;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
                             'title',emp_vals);

emp_vals(1) := :new.EMP_ID;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
                             'employeeNumber',emp_vals);

emp_vals(1) := :new.FIRST_NAME;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_ADD,
                             'userpassword',emp_vals);

-- DN for Entry to be Added under 'ldap_base' [o=acme, dc=com]

emp_dn := 'cn=' || :new.FIRST_NAME || ' ' ||
:new.LAST_NAME || ', ' || ldap_base ;
DBMS_OUTPUT.PUT_LINE(RPAD('Adding Entry for DN ',25,' ') || ': ['
                        || emp_dn || ']');

-- Add new Entry to ldap directory
retval := DBMS_LDAP.add_s(emp_session,emp_dn,emp_array);
DBMS_OUTPUT.PUT_LINE(RPAD('add_s Returns ',25,' ') || ': '
                        || TO_CHAR(retval));

-- Free attribute array (emp_array)
DBMS_LDAP.free_mod_array(emp_array);

END IF; -- INSERTING

-- Process Entry deletion in database

IF DELETING THEN

-- DN for Entry to be deleted under 'ldap_base' [o=acme, dc=com]

emp_dn := 'cn=' || :old.FIRST_NAME || ' ' ||
:old.LAST_NAME || ', ' || ldap_base ;
DBMS_OUTPUT.PUT_LINE(RPAD('Deleting Entry for DN ',25,' ') ||
                      ': [' || emp_dn || ']');
```



```

-- Delete entry in ldap directory
retval := DBMS_LDAP.delete_s(emp_session,emp_dn);
        DBMS_OUTPUT.PUT_LINE(RPAD('delete_s Returns ',25,' ') || ': ' ||
        TO_CHAR(retval));

END IF; -- DELETING

-- Process updated Entry in database

IF UPDATING THEN

    -- Since two Table columns(in this case) constitute a RDN
    -- check for any changes and update RDN in ldap directory
    -- before updating any other attributes of the Entry.

    IF :old.FIRST_NAME <> :new.FIRST_NAME OR
       :old.LAST_NAME  <> :new.LAST_NAME THEN

        emp_dn := 'cn=' || :old.FIRST_NAME || ' ' ||
                  :old.LAST_NAME || ', ' || ldap_base;

        emp_rdn := 'cn=' || :new.FIRST_NAME || ' ' || :new.LAST_NAME;

        DBMS_OUTPUT.PUT_LINE(RPAD('Renaming OLD DN ',25,' ') ||
        ': [' || emp_dn || ']');
        DBMS_OUTPUT.PUT_LINE(RPAD(' => NEW RDN ',25,' ') ||
        ': [' || emp_rdn || ']');
        retval := DBMS_LDAP.modrdn2_s(emp_session,emp_dn,emp_rdn,
        DBMS_LDAP.MOD_DELETE);
        DBMS_OUTPUT.PUT_LINE(RPAD('modrdn2_s Returns ',25,' ') || ': ' ||
        TO_CHAR(retval));

    END IF;

    -- DN for Entry to be updated under 'ldap_base' [o=acme, dc=com]

    emp_dn := 'cn=' || :new.FIRST_NAME || ' ' ||
              :new.LAST_NAME || ', ' || ldap_base;

    DBMS_OUTPUT.PUT_LINE(RPAD('Updating Entry for DN ',25,' ') ||
    ': [' || emp_dn || ']');

    -- Create and setup attribute array(emp_array) for updated entry
    emp_array := DBMS_LDAP.create_mod_array(7);

    emp_vals(1) := :new.LAST_NAME;

```

```
DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_REPLACE,
                              'sn',emp_vals);

emp_vals(1) := :new.FIRST_NAME;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_REPLACE,
                              'givenname',emp_vals);

emp_vals(1) := :new.PHONE_NUMBER;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_REPLACE,
                              'telephonenumber',emp_vals);

emp_vals(1) := :new.MOBILE;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_REPLACE,
                              'mobile',emp_vals);

emp_vals(1) := :new.ROOM_NUMBER;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_REPLACE,
                              'roomNumber',emp_vals);

emp_vals(1) := :new.TITLE;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_REPLACE,
                              'title',emp_vals);

emp_vals(1) := :new.EMP_ID;

DBMS_LDAP.populate_mod_array(emp_array,DBMS_LDAP.MOD_REPLACE,
                              'employeeNumber',emp_vals);

-- Modify entry in ldap directory
retval := DBMS_LDAP.modify_s(emp_session,emp_dn,emp_array);

        DBMS_OUTPUT.PUT_LINE(RPAD('modify_s Returns ',25,' ') || ': ' ||
                              TO_CHAR(retval));

-- Free attribute array (emp_array)
DBMS_LDAP.free_mod_array(emp_array);

END IF; -- UPDATING

-- Unbind from ldap directory
retval := DBMS_LDAP.unbind_s(emp_session);
```

```

DBMS_OUTPUT.PUT_LINE(RPAD('unbind_res Returns ',25,' ') || ' : ' ||
                        TO_CHAR(retval));

DBMS_OUTPUT.PUT_LINE('Directory operation Successful .. exiting');

-- Handle Exceptions
EXCEPTION
    WHEN OTHERS THEN
        -- TODO : should the trigger call unbind at this point ??
        -- what if the exception was raised from unbind itself ??

        DBMS_OUTPUT.PUT_LINE(' Error code      : ' || TO_CHAR(SQLCODE));
        DBMS_OUTPUT.PUT_LINE(' Error Message : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE(' Exception encountered .. exiting');

END;
/
-----END OF trigger.sql-----

```

検索用の DBMS_LDAP の使用

次の例は、DBMS_LDAP API を使用して、PL/SQL プログラムの中で LDAP 検索を実行する方法を示したものです。この例では、前述のトリガーの例で作成したエントリを検索します。この例では、ベースが `o=acme,dc=com` であると前提し、サブツリー検索を実行して、そのベース・エントリに従属するエントリをすべて取り出します。次に示すコードは、`$ORACLE_HOME/ldap/demo/plsql` ディレクトリにある `search.sql` という名前のファイルに保存されています。

search.sql ファイル

この SQL ファイルには、LDAP サーバーに対する一般的な検索の実行に必要な PL/SQL コードが含まれています。

このスクリプトは、次の設定を想定しています。

- LDAP サーバー・ホスト名: NULL (ローカル・ホスト)
- LDAP サーバー・ポート番号: 389
- 従業員レコードのディレクトリ・コンテナ: `o=acme,dc=com`
- ディレクトリ更新のユーザー名 / パスワード: `cn=orcladmin/welcome`

注意： データベース・トリガーによって追加されたエントリを確認するために `trigger.sql` スクリプトおよび `empdata.sql` スクリプトを実行した後は、このファイルを実行してください。

```

set serveroutput on size 30000

DECLARE
    retval          PLS_INTEGER;
    my_session      DBMS_LDAP.session;
    my_attrs        DBMS_LDAP.string_collection;
    my_message      DBMS_LDAP.message;
    my_entry        DBMS_LDAP.message;
    entry_index     PLS_INTEGER;
    my_dn           VARCHAR2(256);
    my_attr_name    VARCHAR2(256);
    my_ber_elmt     DBMS_LDAP.ber_element;
    attr_index      PLS_INTEGER;
    i               PLS_INTEGER;
    my_vals         DBMS_LDAP.STRING_COLLECTION ;
    ldap_host       VARCHAR2(256);
    ldap_port       VARCHAR2(256);
    ldap_user       VARCHAR2(256);
    ldap_passwd     VARCHAR2(256);
    ldap_base       VARCHAR2(256);

BEGIN
    retval          := -1;

    -- Please customize the following variables as needed
    ldap_host       := NULL ;
    ldap_port       := '389';
    ldap_user       := 'cn=orcladmin';
    ldap_passwd     := 'welcome';
    ldap_base       := 'o=acme,dc=com';
    -- end of customizable settings

    DBMS_OUTPUT.PUT('DBMS_LDAP Search Example ');
    DBMS_OUTPUT.PUT_LINE('to directory .. ');
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP Host ',25,' ') || ': ' || ldap_host);
    DBMS_OUTPUT.PUT_LINE(RPAD('LDAP Port ',25,' ') || ': ' || ldap_port);

    -- Choosing exceptions to be raised by DBMS_LDAP library.
    DBMS_LDAP.USE_EXCEPTION := TRUE;

    my_session := DBMS_LDAP.init(ldap_host,ldap_port);

    DBMS_OUTPUT.PUT_LINE (RPAD('Ldap session ',25,' ') || ': ' ||
        RAWTOHEX(SUBSTR(my_session,1,8)) ||
        '(returned from init)');

```

```

-- bind to the directory
retval := DBMS_LDAP.simple_bind_s(my_session,
                                   ldap_user, ldap_passwd);

DBMS_OUTPUT.PUT_LINE(RPAD('simple_bind_s Returns ',25,' ') || ': '
                      || TO_CHAR(retval));

-- issue the search
my_attrs(1) := '*'; -- retrieve all attributes
retval := DBMS_LDAP.search_s(my_session, ldap_base,
                              DBMS_LDAP.SCOPE_SUBTREE,
                              'objectclass=*',
                              my_attrs,
                              0,
                              my_message);

DBMS_OUTPUT.PUT_LINE(RPAD('search_s Returns ',25,' ') || ': '
                      || TO_CHAR(retval));
DBMS_OUTPUT.PUT_LINE (RPAD('LDAP message ',25,' ') || ': ' ||
                      RAWTOHEX(SUBSTR(my_message,1,8)) ||
                      '(returned from search_s)');

-- count the number of entries returned
retval := DBMS_LDAP.count_entries(my_session, my_message);
DBMS_OUTPUT.PUT_LINE(RPAD('Number of Entries ',25,' ') || ': '
                      || TO_CHAR(retval));
DBMS_OUTPUT.PUT_LINE('-----');

-- get the first entry
my_entry := DBMS_LDAP.first_entry(my_session, my_message);
entry_index := 1;

-- Loop through each of the entries one by one
while my_entry IS NOT NULL loop
    -- print the current entry
    my_dn := DBMS_LDAP.get_dn(my_session, my_entry);
    -- DBMS_OUTPUT.PUT_LINE ('          entry #' || TO_CHAR(entry_index) ||
    -- ' entry ptr: ' || RAWTOHEX(SUBSTR(my_entry,1,8)));
    DBMS_OUTPUT.PUT_LINE ('          dn: ' || my_dn);
    my_attr_name := DBMS_LDAP.first_attribute(my_session,my_entry,
    my_ber_elt);
    attr_index := 1;
    while my_attr_name IS NOT NULL loop
        my_vals := DBMS_LDAP.get_values (my_session, my_entry,
        my_attr_name);
        if my_vals.COUNT > 0 then

```

```

        FOR i in my_vals.FIRST..my_vals.LAST loop
            DBMS_OUTPUT.PUT_LINE('      ' || my_attr_name || ' : ' ||
                SUBSTR(my_vals(i),1,200));
        end loop;
    end if;
    my_attr_name := DBMS_LDAP.next_attribute(my_session,my_entry,
        my_ber_elmt);
    attr_index := attr_index+1;
end loop;
my_entry := DBMS_LDAP.next_entry(my_session, my_entry);
DBMS_OUTPUT.PUT_LINE('=====');
entry_index := entry_index+1;
end loop;

-- unbind from the directory
retval := DBMS_LDAP.unbind_s(my_session);
DBMS_OUTPUT.PUT_LINE(RPAD('unbind_res Returns ',25,' ') || ' : ' ||
    TO_CHAR(retval));

DBMS_OUTPUT.PUT_LINE('Directory operation Successful .. exiting');

-- Handle Exceptions
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(' Error code      : ' || TO_CHAR(SQLCODE));
        DBMS_OUTPUT.PUT_LINE(' Error Message : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE(' Exception encountered .. exiting');
END;
/
-----END OF search.sql-----

```

Java サンプル・コード

この項では、Java サンプル・コードを示します。

この項では、次の項目について説明します。

- [User クラスのサンプル・コード](#)
- [Subscriber クラスのサンプル・コード](#)
- [Group クラスのサンプル・コード](#)
- [印刷のサンプル・コード](#)

User クラスのサンプル・コード

```
/*
 * SampleUser.java
 *
 * This is a sample usage of the User class in oracle.ldap.util package
 * found in ldapjclnt9.jar. You can define a user using DN, GUID, or
 * a simple name representing the user. The following methods are exercised
 * in this sample program:
 *
 * - User.authenticateUser() - to authenticate a user with the appropriate
 *   credentials
 * - User.getProperties() - to obtain properties of the user
 * - User.setProperties() - to add, replace, or delete properties of the user
 *
 */

import oracle.ldap.util.*;
import oracle.ldap.util.jndi.*;

import java.io.*;
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;

public class SampleUser {

    public static void main(String argv[])
        throws NamingException {

        // Create InitialDirContext

        InitialDirContext ctx = ConnectionUtil.getDefaultDirCtx( "sandal",
                                                                    "3060",
```

```
        "cn=orcladmin",
        "welcome" );

// Create Subscriber object

Subscriber mysub = null;

try {
    // Creation using DN
    mysub = new Subscriber( ctx, Util.IDTYPE_DN, "o=oracle,dc=com", false );
}
catch (UtilException e) {
    /*
     * Exception encountered in subscriber object constructor
     */
}

// Create User Objects

User myuser = null,
    myuser1 = null;

try {
    // Create User using a subscriber DN and the User DN

    myuser = new User ( ctx,
        Util.IDTYPE_DN,
        "cn=user1,cn=users,o=oracle,dc=com",
        Util.IDTYPE_DN,
        "o=oracle,dc=com",
        false );

    // Create User using a subscriber object and the User
    // simple name

    myuser1 = new User ( ctx,
        Util.IDTYPE_SIMPLE,
        "user1",
        mysub,
        false );
}
catch ( UtilException e ) {
    /*
     * Exception encountered in User object constructor
     */
}
```



```
// Authenticate User
try {
    myuser1.authenticateUser(ctx, User.CREDTYPE_PASSWD, "welcome");
}
catch ( UtilException e ) {
    /*
     * Authenticate fails
     */
}

// Perform User operations

try {
    PropertySetCollection result = null;

    // Get telephonenumber of user

    String[] userAttrList = {"telephonenumber"};
    result = myuser1.getProperties(ctx, userAttrList);

    /*
     * Do work with result
     *
     *
     */
    Util.printResults(result);

    // Set telephonenumber of user

    // Create JNDI ModificationItem

    ModificationItem[] mods = new ModificationItem[1];
    mods[0] = new ModificationItem(DirContext.REPLACE_ATTRIBUTE,
                                   new BasicAttribute("telephonenumber", "444-6789"));

    // Perform modification using User object

    myuser.setProperties(ctx, mods);
}
catch ( UtilException e ) {
    /*
     * Exception encountered in User object operations
     */
}
}
} // End of SampleUser.java
```

Subscriber クラスのサンプル・コード

```
/*
 * SampleSubscriber.java
 *
 * This is a sample usage of the Subscriber class in oracle.ldap.util package
 * found in ldapjclnt9.jar. You can define a group using a DN, GUID, or a
 * simple name of the subscriber. The following methods are exercised in
 * this sample program:
 *
 * - Subscriber.getProperties() - to obtain properties of the group
 */

import oracle.ldap.util.*;
import oracle.ldap.util.jndi.*;

import java.io.*;
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;

public class SampleSubscriber {

    public static void main(String argv[])
        throws NamingException {

        // Create InitialDirContext

        InitialDirContext ctx = ConnectionUtil.getDefaultDirCtx( "sandal",
                                                                    "3060",
                                                                    "cn=orcladmin",
                                                                    "welcome" );

        // Create Subscriber object

        Subscriber mysub = null,
                    mysub1 = null,
                    mysub2 = null;

        try {

            // Creation using DN
            mysub = new Subscriber( ctx,

                                    Util.IDTYPE_DN,
                                    "o=oracle,dc=com",
                                    false );
```

```
// Creation using Simple Name
mysub1 = new Subscriber( ctx,
                        Util.IDTYPE_SIMPLE,
                        "Oracle",
                        false );

// Creation using GUID
mysub2 = new Subscriber( ctx,
                        Util.IDTYPE_GUID,
                        "93B37BBC3B1F46F8E034080020F73460",
                        false );
}
catch (UtilException e) {
    /*
     * Exception encountered in subscriber object constructor
     */
}

// Set the attribute list for attributes returned
String[] attrList = { "cn",
                      "orclcommonusersearchbase",
                      "orclguid" };

// Get Subscriber Properties

PropertySetCollection result = null;
try {
    result = mysub.getProperties(ctx,attrList);
}
catch (UtilException e) {
    /*
     * Exception encountered when searching for subscriber properties
     */
}

/*
 * Do work with the result
 */

Util.printResults(result);
}
}
```

Group クラスのサンプル・コード

```
/*
 * SampleGroup.java
 *
 * This is a sample usage of the Group class in oracle.ldap.util package
 * found in ldapjclnt9.jar. You can define a group using DN or GUID.
 * The following methods are exercised in this sample program:
 *
 * - Group.isMember() - to see if a particular user is
 *   a member of this group
 * - Util.getGroupMembership() - to obtain the list of groups which a
 *   particular user belongs to
 * - Group.getProperties() - to obtain properties of the group
 *
 */

import oracle.ldap.util.*;
import oracle.ldap.util.jndi.*;

import java.io.*;
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;

public class SampleGroup {

    public static void main(String argv[])
        throws NamingException {

        // Create InitialDirContext

        InitialDirContext ctx = ConnectionUtil.getDefaultDirCtx( "sandal",
                                                                "3060",
                                                                "cn=orcladmin",
                                                                "welcome" );

        // Create Group Object
        Group mygroup = null;
        try {
            mygroup = new Group ( Util.IDTYPE_DN,
                                "cn=group1,cn=Groups,o=oracle,dc=com" );
        }
        catch ( UtilException e ) {
            /*
             * Error encountered in Group constructor
             */
        }
    }
}
```

```
}

// Create User Object

User myuser = null;
try {
    // Create User using a subscriber DN and the User DN
    myuser = new User ( ctx,
                        Util.IDTYPE_DN,
                        "cn=orcladmin,cn=users,o=oracle,dc=com",
                        Util.IDTYPE_DN,
                        "o=oracle,dc=com",
                        false );
}
catch ( UtilException e ) {
    /*
     * Exception encountered in User object constructor
     */
}

// Perform Group Operations

try {

    // isMember method

    if (mygroup.isMember( ctx,
                          myuser,
                          true ) ) {

        /*
         * myuser is a member of this group
         * Do work
         *
         *
         */
        System.out.println("is member");
    }

    // Get all nested groups that a user belongs to

    PropertySetCollection result = Util.getGroupMembership( ctx,
                                                             myuser,
                                                             new String[0],
                                                             true );

    /*
     * Do work with result
    */
}
```

```
        *           .
        *           .
        *           .
    */
    Util.printResults ( result );

    // Get Group Properties

    result = getProperties( ctx, null );

    /*
     * Do work with result
     *           .
     *           .
     *           .
    */
}
catch ( UtilException e ) {
    /*
     * Exception encountered in getGroupMembership
     */
}
} // End of SampleGroup.java
```

印刷のサンプル・コード

```
/*
 * SamplePrint.java
 *
 * This sample program demonstrates the usage of the PropertySetCollection
 * class which is a key structure used in the oracle.ldap.util package for
 * obtaining search results. A sample printResults() method is implemented
 * that neatly prints out the values of a PropertySetCollection.
 * A PropertySetCollection contains a set of PropertySets. A PropertySet is
 * analogous to an LDAP entry which is identified by the DN. Each PropertySet
 * contains a set of zero or more Properties. A Property is analogous to a
 * particular attribute of an LDAP entry and it may contain one or more
 * values. The printResults() method takes in a PropertySetCollection and
 * navigates through it in a systematic way, printing out the results to
 * the system output.
 *
 */

import oracle.ldap.util.*;
import oracle.ldap.util.jndi.*;
```

```
import java.io.*;
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;

public class SamplePrint {

    public static void printResults( PropertySetCollection resultSet )
    {
        // for loop to go through each PropertySet
        for (int i = 0; i < resultSet.size(); i++ )
        {
            // Get PropertySet
            PropertySet curEntry = resultSet.getPropertySet( i );
            Object obj = null;

            // Print DN of PropertySet
            System.out.println("dn: " + curEntry.getDN());

            // Go through each Property of the PropertySet
            for (int j = 0; j < curEntry.size(); j++)
            {
                // Get Property
                Property curAttr = curEntry.getProperty( j );

                // Go through each value of the Property
                for (int k = 0; k < curAttr.size(); k++)
                {
                    obj = curAttr.getValue(k);
                    if( obj instanceof java.lang.String) {
                        System.out.println( curAttr.getName() + ": "
                                           + (String) obj);
                    }
                    else if (obj instanceof byte[]) {
                        System.out.println( curAttr.getName() + ": "
                                           + (new java.lang.String((byte [])obj)));
                    }
                }
            }
            System.out.println();
        }
    }

} // End of SamplePrint.java
```

用語集

Access Control Information Item (ACI)

どのディレクトリ・データに対して、誰がどのタイプのアクセス権を持っているかを判断する属性。この属性には、エントリに関係する構造型アクセス項目と、属性に関するコンテンツ・アクセス項目に関する 1 組の規則が含まれている。両方のアクセス項目に対するアクセス権限を、1 つ以上のユーザーまたはグループに付与できる。

Access Control List (ACL)

アクセス・ディレクティブのグループ。管理者が定義する。ディレクティブは、特定のクライアントまたはクライアントのグループ、あるいはその両方に対して、特定データへのアクセスのレベルを付与する。

ACI

「[Access Control Information Item \(ACI\)](#)」を参照。

ACL

「[Access Control List \(ACL\)](#)」を参照。

ACP

「[アクセス制御ポリシー・ポイント \(Access Control Policy Point: ACP\)](#)」を参照。

API

「[Application Program Interface \(API\)](#)」を参照。

Application Program Interface (API)

指定したアプリケーションのサービスにアクセスするための一連のプログラム。たとえば、LDAP 対応のクライアントは、LDAP API で使用可能なプログラム・コールを通して、ディレクトリ情報にアクセスする。

Cipher Suite

SSL において、ネットワークのノード間でメッセージ交換に使用される認証、暗号化およびデータ整合性アルゴリズムのセット。SSL ハンドシェイク時に、2 つのノード間で折衝し、メッセージを送受信するときに使用する Cipher Suite を確認する。

configset

「[構成設定エントリ \(configuration set entry\)](#)」を参照。

DES

データ暗号化規格。1970 年代に IBM と米国政府によって公式規格として開発されたブロック暗号。

DIB

「[ディレクトリ情報ベース \(directory information base: DIB\)](#)」を参照。

DIS

「[ディレクトリ統合サーバー \(directory integration server: DIS\)](#)」を参照。

DIT

「[ディレクトリ情報ツリー \(directory information tree: DIT\)](#)」を参照。

DN

「[識別名 \(distinguished name: DN\)](#)」を参照。

DRG

「[ディレクトリ・レプリケーション・グループ \(directory replication group: DRG\)](#)」を参照。

DSA

「[ディレクトリ・システム・エージェント \(directory system agent: DSA\)](#)」を参照。

DSE

「[ディレクトリ固有のエントリ \(directory-specific entry: DSE\)](#)」を参照。

DSA 固有のエントリ。異なる DSA に同じ DIT 名を保持できるが、内容は異なる必要がある。つまり、DSA の内容は、その DSA に固有である。DSE は、それを保持している DSA に固有の内容を含むエントリである。

Global Unique Identifier (GUID)

マルチマスター・レプリケーション環境では、複数のノードでレプリケートされるエントリは、各ノードで同じ識別名を持つ。ただし、識別名が同じでも、各ノードで異なる GUID が割り当てられる。たとえば、同じ識別名を node1 と node2 の両方でレプリケートできるが、

node1 に常駐しているときのその識別名に対する GUID は、node2 におけるその識別名に対する GUID とは異なる。

GUID

「[Global Unique Identifier \(GUID\)](#)」を参照。

Internet Engineering Task Force (IETF)

新しいインターネット標準仕様の開発に従事する主要機関。インターネット・アーキテクチャおよびインターネットの円滑な操作の発展に関わるネットワーク設計者、運営者、ベンダーおよび研究者による国際的な団体である。

Internet Message Access Protocol (IMAP)

プロトコルの 1 種。クライアントは、このプロトコルを使用して、サーバー上の電子メール・メッセージに対するアクセスおよび操作を行う。リモートのメッセージ・フォルダ（メールボックスとも呼ばれる）を、ローカルのメールボックスと機能的に同じ方法で操作できる。

LDAP

「[Lightweight Directory Access Protocol \(LDAP\)](#)」を参照。

LDAP データ交換フォーマット (LDAP Data Interchange Format: LDIF)

LDAP コマンドライン・ユーティリティに使用する入力ファイルをフォーマットするための一連の規格。

LDIF

「[LDAP データ交換フォーマット \(LDAP Data Interchange Format: LDIF\)](#)」を参照。

Lightweight Directory Access Protocol (LDAP)

標準的で拡張可能なディレクトリ・アクセス・プロトコル。LDAP クライアントとサーバーが通信で使用する共通言語。業界標準のディレクトリ製品（Oracle Internet Directory など）をサポートする設計規則のフレームワーク。

MD4

128 ビットのハッシュまたはメッセージ・ダイジェスト値を生成する一方方向ハッシュ関数。1 ビットでもファイルの値が変更された場合、そのファイルの MD4 チェックサムは変更される。元のファイルと同じ結果を MD4 で生成するようにファイルを偽造することはほぼ不可能である。

MD5

MD4 の改善されたバージョン。

MDS

「[マスター定義サイト \(master definition site: MDS\)](#)」を参照。

MTS

「[共有サーバー \(shared server\)](#)」を参照。

OEM

「[Oracle Enterprise Manager](#)」を参照。

OID 制御ユーティリティ (OID Control Utility)

サーバーの起動と停止のコマンドを発行するコマンドライン・ツール。コマンドは、[OID モニター](#)のプロセスによって解析され、実行される。

OID データベース・パスワード・ユーティリティ (OID Database Password Utility)

Oracle Internet Directory が Oracle データベースに接続するときのパスワードの変更に使用されるユーティリティ。

OID モニター (OID Monitor)

Oracle ディレクトリ・サーバー・プロセスの開始、監視および終了を実行する Oracle Internet Directory のコンポーネント。レプリケーション・サーバー（インストールされている場合）および Oracle Directory Integration Server の制御も行う。

Oracle Call Interface (OCI)

Application Program Interface (API) の 1 つ。これにより、第三代言語のネイティブ・プロシージャやファンクション・コールを使用して、Oracle データベース・サーバーにアクセスし、SQL 文の実行のすべての段階を制御するアプリケーションを作成できる。

Oracle Directory Integration Platform

[Oracle Internet Directory](#) のコンポーネントの 1 つ。Oracle Internet Directory のような中央 LDAP ディレクトリの周囲のアプリケーションを統合するために開発されたフレームワーク。

Oracle Directory Integration Server (DIS)

Oracle Directory Integration Platform 環境で、Oracle Internet Directory の変更イベントを監視し、[ディレクトリ統合プロファイル](#)の情報に基づいてアクションを行うデーモン・プロセス。

Oracle Directory Manager

Oracle Internet Directory を管理するための、Graphical User Interface (GUI) を備えた Java ベースのツール。

Oracle Enterprise Manager

Oracle 製品の 1 つ。グラフィカルなコンソール、エージェント、共通サービスおよびツールを組み合わせ、Oracle 製品を管理するための統合された包括的なシステム管理プラットフォームを提供する。

Oracle Internet Directory

分散ユーザーやネットワーク・リソースに関する情報の検索を可能にする、一般的な用途のディレクトリ・サービス。LDAP バージョン 3 と Oracle9i の高度のパフォーマンス、拡張性、耐久性および可用性を組み合わせたもの。

Oracle Net Services

Oracle のネットワーク製品ファミリの基礎。Oracle Net Services を使用すると、サービスやアプリケーションを異なるコンピュータに配置して通信できる。Oracle Net Services の主な機能には、ネットワーク・セッションの確立およびクライアント・アプリケーションとサーバー間のデータ転送がある。Oracle Net Services は、ネットワーク上の各コンピュータに配置される。ネットワーク・セッションの確立後は、Oracle Net Services はクライアントとサーバーのためのデータ伝達手段として機能する。

Oracle PKI 証明書使用 (Oracle PKI certificate usages)

[証明書](#)でサポートされる Oracle アプリケーション・タイプを定義する。

Oracle Wallet Manager

セキュリティ管理者が、クライアントとサーバーにおける公開鍵のセキュリティ資格証明の管理に使用する Java ベースのアプリケーション。

Oracle9i レプリケーション (Oracle9i Replication)

2 つの Oracle データベース間で、データベースの表を継続的に同期化できる Oracle9i の機能。

PKCS #12

[公開鍵暗号](#)規格 (PKCS)。RSA Data Security, Inc. の PKCS #12 は、個人的な認証資格証明を、通常 [Wallet](#) と呼ばれる形式で保管および転送するための業界標準である。

RDN

「[相対識別名](#)」を参照。

SASL

「[Simple Authentication and Security Layer \(SASL\)](#)」を参照。

Secure Hash Algorithm (SHA)

長さが 264 ビット未満のメッセージを取得して、160 ビットのメッセージ・ダイジェスト値を生成するアルゴリズム。このアルゴリズムは MD5 よりも若干遅いが、メッセージ・ダイジェスト値が大きくなることで、総当り攻撃や反転攻撃に対してより強力に保護できる。

Secure Sockets Layer (SSL)

ネットワーク接続を保護するために Netscape Communications Corporation が開発した業界標準プロトコル。SSL では公開鍵インフラストラクチャ (PKI) を使用して、認証、暗号化およびデータの整合性を実現している。

SGA

「[システム・グローバル領域 \(System Global Area: SGA\)](#)」を参照。

SHA

「[Secure Hash Algorithm \(SHA\)](#)」を参照。

Simple Authentication and Security Layer (SASL)

接続ベースのプロトコルに認証サポートを追加する方法。この仕様を使用するために、プロトコルには、ユーザーを識別してサーバーに対して認証を行い、オプションで、後続のプロトコル対話に使用するセキュリティ・レイヤーを取り決めるコマンドが含まれる。このコマンドには、SASL 方式を識別する必須引数がある。

SLAPD

スタンドアロンの LDAP デーモン。

SSL

「[Secure Sockets Layer \(SSL\)](#)」を参照。

subACLSubentry

ACL 情報が含まれた特定のタイプのサブエントリ。

subSchemaSubentry

スキーマ情報が含まれた特定のタイプの[サブエントリ](#)。

TLS

「[Transport Layer Security \(TLS\)](#)」を参照。

Transport Layer Security (TLS)

インターネット上の通信プライバシーを提供するプロトコル。このプロトコルによって、クライアント / サーバー・アプリケーションは、通信時の盗聴、改ざんまたはメッセージの偽造を防止できる。

Trustpoint

「[信頼できる証明書](#)」を参照。

UCS-2

固定幅 16 ビットの [Unicode](#)。各文字は 16 ビットの領域を持つ。Latin-1 文字はこの規格の最初の 256 コード・ポイントであり、Latin-1 の 16 ビット拡張とみなすことができる。

Unicode

汎用キャラクタ・セットのタイプ。16 ビットの領域にエンコードされた 64K 個の文字の集合。既存のほとんどすべてのキャラクタ・セット規格の文字をすべてエンコードする。世界中で使用されているほとんどの記述法を含む。Unicode は Unicode Inc. によって所有および定義される。Unicode は標準的なエンコーディングであり、異なるロケールで値を伝達できることを意味する。しかし、Unicode とすべての Oracle キャラクタ・セットとの間で、情報の損失なしにラウンドトリップ変換が行われることは保証されない。

UNIX Crypt

UNIX 暗号化アルゴリズム。

UTC (Coordinated Universal Time)

世界中のあらゆる場所で共通の標準時間。以前から現在に至るまで広くグリニッジ時 (GMT) または世界時と呼ばれており、UTC は名目上は地球の本初子午線に関する平均太陽時を表す。UTC 形式である場合、値の最後に z が示される (例: 200011281010z)。

UTF-8

文字ごとに連続した 1、2 または 3 バイトを使用する [UCS-2](#) の可変幅エンコーディング。0 ~ 127 の文字 (7 ビット ASCII 文字) は 1 バイトでエンコードされ、128 ~ 2047 の文字では 2 バイト、2048 ~ 65535 の文字では 3 バイトを必要とする。このための Oracle キャラクタ・セット名は UTF-8 (Unicode 2.1 規格用) となる。規格は、文字ごとに連続した 4、5 または 6 バイトを使用する UCS4 文字をサポートする拡張の余地を残している。

Wallet

個々のエンティティに対するセキュリティ資格証明の格納と管理に使用される抽象的な概念。様々な暗号化サービスで使用するために、資格証明の格納と取出しを実現する。Wallet Resource Locator (WRL) は、Wallet の位置を特定するために必要な情報をすべて提供する。

X.509

公開鍵の署名に使用される ISO の一般的な形式。

アクセス制御ポリシー・ポイント (Access Control Policy Point: ACP)

セキュリティ・ディレクティブを含むエントリ。このディレクティブは、[ディレクトリ情報ツリー](#)内の下位エントリすべてに適用される。

アドバンスト・レプリケーション (Advanced Replication: AR)

「[Oracle9i レプリケーション \(Oracle9i Replication\)](#)」を参照。

アドバンスト・レプリケーション (ASR)

「[Oracle9i レプリケーション \(Oracle9i Replication\)](#)」を参照。

暗号化 (cryptography)

データのエンコーディングとデコーディングを行い、保護メッセージを生成する作業。

暗号化 (encryption)

メッセージの内容を、宛先の受信者以外の第三者が読むことのできない形式（暗号文）に変換するプロセス。

一方向関数 (one-way function)

一方向への計算は容易だが、逆の計算、すなわち反対方向への計算は非常に難しい関数。

一方向ハッシュ関数 (one-way hash function)

可変サイズの入力を取得して、固定サイズの出力を作成する**一方向関数**。

一致規則 (matching rule)

検索または比較操作における、検索対象の属性値と格納されている属性値との間の等価性の判断。たとえば、telephoneNumber 属性に関連付けられた一致規則では、(650) 123-4567 を (650) 123-4567 または 6501234567 のいずれかと一致させるか、あるいはその両方と一致させることができる。属性を作成したときに、その属性を一致規則と対応付けることができる。

インスタンス (instance)

「[ディレクトリ・サーバー・インスタンス \(directory server instance\)](#)」を参照。

インポート・エージェント (import agent)

Oracle Directory Integration Platform 環境で、Oracle Internet Directory にデータをインポートするエージェント。

インポート・データ・ファイル (import data file)

Oracle Directory Integration Platform 環境で、[インポート・エージェント](#)によってインポートされたデータを格納するファイル。

エージェント (agent)

「[ディレクトリ統合エージェント \(directory integration agent\)](#)」を参照。

エージェント・プロフィール (agent profile)

Oracle Directory Integration Platform 環境で、次の内容を指定する Oracle Internet Directory のエントリ。

- 統合エージェント用の構成パラメータ
- 接続先ディレクトリと Oracle Internet Directory との間の同期化に適用するマッピング規則

エクスポート・エージェント (export agent)

Oracle Directory Integration Platform 環境で、Oracle Internet Directory からデータをエクスポートするエージェント。

エクスポート・データ・ファイル (export data file)

Oracle Directory Integration Platform 環境で、[エクスポート・エージェント](#)によってエクスポートされたデータを格納するファイル。

エクスポート・ファイル (export file)

「[エクスポート・データ・ファイル \(export data file\)](#)」を参照。

エントリ (entry)

ディレクトリの基本単位で、ディレクトリ・ユーザーに関係のあるオブジェクトに関する情報が含まれている。

応答時間 (response time)

要求の発行から応答の完了までの時間。

オブジェクト・クラス (object class)

名前を持った属性のグループ。属性をエントリに割り当てるときは、その属性を保持しているオブジェクト・クラスをそのエントリに割り当てる。

同じオブジェクト・クラスに関連するオブジェクトはすべて、同じ属性を共有する。

介在者 (man-in-the-middle)

第三者によるメッセージの不正傍受などのセキュリティ攻撃。第三者、つまり介在者は、メッセージを復号化して再暗号化し（元のメッセージを変更する場合と変更しない場合がある）、元のメッセージの宛先である受信者に転送する。これらの処理はすべて、正当な送受信者が気付かないうちに行われる。この種のセキュリティ攻撃は、[認証](#)が行われていない場合にのみ発生する。

外部エージェント (external agent)

Oracle Directory Integration Server に依存しないディレクトリ統合エージェント。Oracle Directory Integration Server は外部エージェントに対して、スケジューリング、マッピングまたはエラー処理の各サービスを提供しない。外部エージェントは、通常、サード・パー

ディのメタディレクトリ・ソリューションを Oracle Directory Integration Platform に統合するときに使用する。

鍵 (key)

暗号化において広く使用されているビット列。データの暗号化と復号化を可能にする。鍵は別の数学的な操作にも使用される。暗号が与えられると、鍵によって、平文から暗号文へのマッピングが判断される。

鍵のペア (key pair)

[公開鍵](#)とそれに対応する [秘密鍵](#)のペア。

「[公開鍵と秘密鍵のペア](#)」を参照。

鍵ペア Wallet (single key-pair wallet)

単一のユーザー [証明書](#)とその関連する [秘密鍵](#)が含まれる PKCS #12 形式の [Wallet](#)。公開鍵は証明書に埋め込まれている。

拡張性 (scalability)

限定された使用可能なハードウェア・リソースに比例したスループットを提供するシステム機能。

簡易認証 (simple authentication)

ネットワークでの送信時に暗号化されない識別名とパスワードを使用して、クライアントがサーバーに対して自己認証を行うプロセス。簡易認証オプションでは、クライアントが送信した識別名とパスワードと、ディレクトリに格納されている識別名とパスワードが一致していることをサーバーが検証する。

管理領域 (administrative area)

ディレクトリ・サーバー上の 1 つのサブツリー。そのエントリは、1 つの管理認可レベル (スキーマ、ACL および共通属性) で制御される。

競合 (contention)

リソースの競合。

兄弟関係 (sibling)

1 つ以上の他のエントリと同じ親を持ったエントリ。

共有サーバー (shared server)

多数のユーザー・プロセスが、非常に少数のサーバー・プロセスを共有できるように構成されたサーバー。これにより、サポートされるユーザー数が増える。共有サーバー構成では、多数のユーザー・プロセスがディスパッチャに接続する。ディスパッチャは、複数の着信ネットワーク・セッション要求を共通キューに送る。複数のサーバー・プロセスの共有プールの中で、あるアイドル状態の共有サーバー・プロセスが共通キューから要求を取り出す。

これは、サーバー・プロセスの小規模プールが大量のクライアントを処理できることを意味する。専用サーバーと対比。

継承 (inherit)

オブジェクト・クラスが別のクラスから導出されたときに、導出元のオブジェクト・クラスの多数の特性も導出（継承）されること。同様に、属性のサブタイプも、そのスーパータイプの特性を継承する。

ゲスト・ユーザー (guest user)

匿名ユーザーではなく、特定のユーザー・エントリも持っていないユーザー。

コールド・バックアップ (cold backup)

データベース・コピー・プロシージャを使用して、新規 **DSA** ノードを既存のレプリケート・システムに追加する手順。

公開鍵 (public key)

公開鍵暗号における一般に公開されるキー。主に暗号化に使用されるが、署名の検証にも使用される。

公開鍵暗号 (public-key cryptography)

公開鍵と秘密鍵を使用する方法に基づいた暗号化。

公開鍵暗号 (public-key encryption)

メッセージの送信側が、受信側の公開鍵でメッセージを暗号化するプロセス。配信されたメッセージは、受信側の秘密鍵で復号化される。

公開鍵と秘密鍵のペア (public/private key pair)

数学的に関連付けられた 2 つの数字のセット。1 つは秘密鍵、もう 1 つは公開鍵と呼ばれる。公開鍵は通常広く使用可能であるのに対して、秘密鍵はその所有者のみ使用可能である。公開鍵で暗号化されたデータは、それに関連付けられた秘密鍵でのみ復号化でき、秘密鍵で暗号化されたデータは、それに関連付けられた公開鍵でのみ復号化できる。公開鍵で暗号化されたデータを、同じ公開鍵で復号化することはできない。

構成設定エントリ (configuration set entry)

ディレクトリ・サーバーの特定インスタンスに関する構成パラメータを保持しているディレクトリ・エントリ。複数の構成設定エントリを格納でき、実行時に参照できる。構成設定エントリは、DSE の subConfigsubEntry 属性で指定されているサブツリー内でメンテナンスされる。DSE 自体は、サーバーの起動対象である関連の**ディレクトリ情報ベース**に常駐している。

コンシューマ (consumer)

レプリケーション更新の宛先となるディレクトリ・サーバー。スレーブと呼ばれることもある。

コンテキスト接頭辞 (context prefix)

ネーミング・コンテキストのルート **DN**。

サービス時間 (service time)

要求の開始から、その要求に対する応答の完了までの時間。

サブエントリ (subentry)

サブツリー内のエントリ・グループに適用可能な情報が含まれているエントリのタイプ。情報には次の3つのタイプがある。

- アクセス制御ポリシー・ポイント
- スキーマ規則
- 共通属性

サブエントリは、管理領域のルート **DN** のすぐ下に位置している。

サブクラス (subclass)

別のオブジェクト・クラスから導出されたオブジェクト・クラス。導出元のオブジェクト・クラスは、その **スーパークラス** と呼ばれる。

サブスキーマ識別名 (subschema DN)

独立したスキーマ定義を持つディレクトリ情報ツリー領域のリスト。

サブタイプ (subtype)

オプションを持たない同じ属性に対して、1つ以上のオプションを持つ属性。たとえば、American English をオプションとして持つ commonName (cn) 属性は、そのオプションを持たない commonName (cn) 属性のサブタイプである。逆に、オプションを持たない commonName (cn) 属性は、オプションを持つ同じ属性の **スーパータイプ** となる。

サプライヤ (supplier)

レプリケーションにおいて、ネーミング・コンテキストのマスター・コピーを保持しているサーバー。マスター・コピーから **コンシューマ**・サーバーに更新を供給する。

参照 (referral)

ディレクトリ・サーバーがクライアントに提供する情報。要求する情報を見つけるためにクライアントが接続する必要がある他のサーバーを示す。

「**ナレッジ参照 (knowledge reference)**」も参照。

識別名 (distinguished name: DN)

ディレクトリ・エントリの一意名。親エントリの個々の名前がすべて、下からルート方向へ順に結合されて構成されている。

思考時間 (think time)

ユーザーが実際にプロセッサを使用していない時間。

システム・グローバル領域 (System Global Area: SGA)

共有メモリー構造の 1 グループ。1 つの Oracle データベース・インスタンスに関するデータと制御情報が含まれている。複数のユーザーが同じインスタンスに同時に接続した場合、そのインスタンスの SGA 内のデータはユーザー間で共有される。したがって、SGA は共有グローバル領域と呼ばれることもある。バックグラウンド・プロセスとメモリー・バッファの組合せは、Oracle インスタンスと呼ばれる。

システム固有のエージェント (native agent)

Oracle Directory Integration Platform 環境で、**ディレクトリ統合サーバー**の制御下で実行される**エージェント**。

システム操作属性 (system operational attribute)

ディレクトリ自体の操作に関係する情報を保持する属性。一部の操作情報は、サーバーを制御するためにディレクトリによって指定される (例: エントリのタイム・スタンプ)。アクセス情報など、その他の操作情報は、管理者が定義し、ディレクトリ・プログラムの処理時に、そのプログラムによって使用される。

従属参照 (subordinate reference)

エントリのすぐ下から始まるネーミング・コンテキストの参照位置を、ディレクトリ情報ツリー内で下位方向に指し示すナレッジ参照。

上位参照 (superior reference)

ディレクトリ情報ツリー内で、参照先の DSA が保持しているすべてのネーミング・コンテキストより上位のネーミング・コンテキストを保持している DSA を上位方向に指し示すナレッジ参照。

証明書 (certificate)

公開鍵に対して識別情報を安全にバインドする ITU x.509 v3 の標準データ構造。証明書は、エンティティの公開鍵が、信頼されている機関 (**認証局**) によって署名されたときに有効となる。この証明書は、そのエンティティの情報が正しいこと、および公開鍵がそのエンティティに実際に属していることを保証する。

証明連鎖 (certificate chain)

エンド・ユーザーまたはサブスクリバの証明書とその認証局の証明書を含む、順序付けられた証明書のリスト。

信頼できる証明書 (trusted certificate)

一定の信頼度を有すると認定された第三者の識別情報。信頼されている証明書は、識別情報の内容がそのエンティティと一致していることを検証するときに使用される。一般的に、信頼されている認証局によってユーザーの証明書が発行される。

スーパークラス (superclass)

別のオブジェクト・クラスの導出元のオブジェクト・クラス。たとえば、オブジェクト・クラス `person` は、オブジェクト・クラス `organizationalPerson` のスーパークラスである。後者の `organizationalPerson` は、`person` のサブクラスであり、`person` に含まれている属性を継承する。

スーパータイプ (supertype)

1 つ以上のオプションを持つ同じ属性に対して、オプションを持たない属性。たとえば、オプションを持たない `commonName (cn)` 属性は、オプションを持つ同じ属性のスーパータイプである。逆に、`American English` をオプションとして持つ `commonName (cn)` 属性は、そのオプションを持たない `commonName (cn)` 属性のサブタイプとなる。

スーパー・ユーザー (super user)

一般的にはディレクトリ情報へのすべてのアクセスが可能な、特別なディレクトリ管理者。

スキーマ (schema)

属性、オブジェクト・クラスおよび対応する一致規則の集合体。

スポンサ・ノード (sponsor node)

レプリケーションにおいて、新規ノードに初期データを供給するために使用されるノード。

スマート・ナレッジ参照 (smart knowledge reference)

ナレッジ参照エントリが検索の有効範囲内にあるときに戻されるナレッジ参照。要求された情報を格納しているサーバーを示す。

スループット (throughput)

Oracle Internet Directory が単位時間ごとに処理する要求の数。通常、「操作 / 秒」(1 秒当りの操作件数) で表される。

スレーブ (slave)

「[コンシューマ \(consumer\)](#)」を参照。

整合性 (integrity)

受信メッセージの内容が、送信時の元のメッセージの内容から変更されていないことを保証すること。

セッション鍵 (session key)

1 つのメッセージまたは 1 つの通信セッションの継続中に使用される、対称鍵暗号方式の鍵。

接続記述子 (connect descriptor)

特別にフォーマットされた、ネットワーク接続の接続先の説明。接続記述子には、宛先サービスおよびネットワーク・ルート情報が含まれる。

宛先サービスを示すには、その Oracle9i リリース 1 (9.0.1) データベースに対応するサービス名、あるいは Oracle リリース 8.0 またはバージョン 7 のデータベースに対応する Oracle システム識別子 (SID) を使用する。ネットワーク・ルートは、少なくとも、ネットワーク・アドレスによってリスナーの位置を提供する。

接続ディレクトリ (connected directory)

Oracle Directory Integration Platform 環境で、それ自体 (たとえば、Oracle Human Resource データベース) と Oracle Internet Directory との間で完全なデータの同期が必要な情報リポジトリ。

相対識別名 (relative distinguished name: RDN)

ローカルの最下位レベルのエントリ名。エントリのアドレスを一意に識別するために使用される他の修飾エントリ名は含まれない。たとえば、cn=Smith,o=acme,c=US では、cn=Smith が相対識別名である。

属性 (attribute)

エントリの性質を説明する断片的な情報項目。1 つのエントリは 1 組の属性から構成され、それぞれが **オブジェクト・クラス** に所属する。さらに、各属性にはタイプと値があり、タイプは属性の情報の種類を説明するものであり、値には実際のデータが格納されている。

属性構成ファイル (attribute configuration file)

Oracle Directory Integration Platform 環境で、接続先ディレクトリに関係のある属性を指定するファイル。

属性値 (attribute value)

エントリで表出される情報の特定の値。たとえば、jobTitle 属性に対する値には manager がある。

属性の型 (attribute type)

属性に含まれている情報の種類 (例: jobTitle)。

その他の情報リポジトリ (other information repository)

Oracle Internet Directory 以外のすべての情報リポジトリ。Oracle Directory Integration Platform 環境では、Oracle Internet Directory が **中央ディレクトリ** として機能する。

待機時間 (latency)

指定したディレクトリ操作が完了するまでのクライアントの待機時間。待機時間は、空費時間として定義される場合がある。ネットワーク通信では、待機時間は、ソースから宛先へパケットが移動する時間として定義される。

待機時間 (wait time)

要求の発行から応答の開始までの時間。

中央ディレクトリ (central directory)

Oracle Directory Integration Platform 環境で、中央リポジトリとして機能するディレクトリ。Oracle Directory Integration Platform 環境では、Oracle Internet Directory が中央ディレクトリになる。

データ整合性 (data integrity)

受信メッセージの内容が、送信時の元のメッセージの内容から変更されていないことを保証すること。

ディレクトリ固有のエントリ (directory-specific entry: DSE)

ディレクトリ・サーバー固有のエントリ。異なるディレクトリ・サーバーに同じ DIT 名を保持できるが、内容は異なる必要がある。つまり、ディレクトリの内容は、そのディレクトリに固有である。DSE は、それを保持しているディレクトリ・サーバーに固有の内容を含むエントリである。

ディレクトリ・サーバー・インスタンス (directory server instance)

ディレクトリ・サーバーの個々の起動のこと。異なるディレクトリ・サーバーの起動（それぞれ、同じまたは異なる構成設定エントリと起動フラグで起動）は、異なるディレクトリ・サーバー・インスタンスと呼ばれる。

ディレクトリ・システム・エージェント (directory system agent: DSA)

ディレクトリ・サーバーを表す X.500 の用語。

ディレクトリ情報ツリー (directory information tree: DIT)

エントリの識別名で構成されるツリー形式の階層構造。

ディレクトリ情報ベース (directory information base: DIB)

ディレクトリに保持されているすべての情報の完全なセット。DIB は、[ディレクトリ情報ツリー](#)内で、階層的に相互に関連するエントリで構成されている。

ディレクトリ同期プロファイル (directory synchronization profile)

Oracle Internet Directory と外部システム間の同期の実現方法を記述した特殊な[ディレクトリ統合プロファイル](#)。

ディレクトリ統合エージェント (directory integration agent)

Oracle Directory Integration Platform 環境で、接続先ディレクトリと Oracle Internet Directory との間で変更を同期化するために、接続先ディレクトリとの対話を行うプログラム。

ディレクトリ統合サーバー (directory integration server: DIS)

Oracle Directory Integration Platform 環境で、Oracle Internet Directory と [接続ディレクトリ](#) との間でデータの同期化を実行するサーバー。

ディレクトリ統合プロファイル (directory integration profile)

Oracle Directory Integration Platform 環境で、Oracle Directory Integration Platform が外部システムとどのように通信し、何を通信するかを示す Oracle Internet Directory のエントリ。

ディレクトリ・ネーミング・コンテキスト (directory naming context)

「[ネーミング・コンテキスト \(naming context\)](#)」を参照。

ディレクトリ・プロビジョニング・プロファイル (Directory Provisioning Profile)

Oracle Directory Integration Platform がディレクトリ対応アプリケーションに送信するプロビジョニング関連通知の性質を記述した特殊な [ディレクトリ統合プロファイル](#)。

ディレクトリ・レプリケーション・グループ (directory replication group: DRG)

レプリケーション承諾のメンバーであるディレクトリ・サーバーの集まり。

デフォルト・ナレッジ参照 (default knowledge reference)

ベース・オブジェクトがディレクトリになく、操作がサーバーによってローカルに保持されていないネーミング・コンテキストで実行されたときに戻される [ナレッジ参照](#)。デフォルト・ナレッジ参照は、一般的にディレクトリ・パーティション化対策についてより多くのナレッジを持つサーバーに送信する。

統合エージェント (integration agent)

「[エージェント \(agent\)](#)」を参照。

同時クライアント (concurrent clients)

Oracle Internet Directory とのセッションを確立しているクライアントの総数。

同時操作 (concurrent operations)

すべての同時クライアントの要求に基づいてディレクトリで実行されている操作の数。一部のクライアントではセッションがアイドル状態の可能性があるので、この数は同時クライアントの数と必ずしも同じではない。

特定管理領域 (specific administrative area)

次の3つの側面を制御する管理領域。

- サブスキーマ管理
- アクセス制御管理
- 共通属性管理

特定管理領域では、この3つの管理面の1つが制御される。特定管理領域は、自律型管理領域の一部である。

匿名認証 (anonymous authentication)

ディレクトリがユーザー名とパスワードの組合せを要求せずにユーザーを認証するプロセス。各匿名ユーザーは、匿名ユーザー用に指定された権限を行使する。

ナレッジ参照 (knowledge reference)

リモート **DSA** に関するアクセス情報（名前とアドレス）およびそのリモート DSA が保持している **DIT** のサブツリーの名前。ナレッジ参照は、参照とも呼ばれる。

認可 (authorization)

オブジェクトまたはオブジェクトのセットへのアクセスのためにユーザー、プログラムまたはプロセスに与えられる許可。

認証 (authentication)

コンピュータ・システム内のユーザー、デバイスまたはその他のエンティティの識別情報を検証するプロセス。多くの場合、システム内のリソースへのアクセスを許可する前提条件として使用される。

認証局 (certificate authority: CA)

他のエンティティ（ユーザー、データベース、管理者、クライアント、サーバーなど）が本物であることを証明する信頼性のあるサード・パーティ。認証局は、ユーザーの識別情報を検証し、認証局の秘密鍵を使用して署名した証明書を発行する。

ネーミング・コンテキスト (naming context)

完全に1つのサーバーに常駐しているサブツリー。サブツリーは連続している必要がある。つまり、サブツリーの最上位の役割を果たすエントリから始まり、下位方向にリーフ・エントリまたは従属ネーミング・コンテキストへの**ナレッジ参照**（参照とも呼ばれる）のいずれかまでを範囲とする必要がある。単一のエントリからディレクトリ情報ツリー全体までを範囲とすることができる。

ネーミング属性 (naming attribute)

異なるタイプの **RDN** の値を保持する特別な属性。ネーミング属性は、そのニーモニック・ラベル（通常 cn、sn、ou、o、c など）で識別できる。たとえば、ネーミング属性 c は、ネーミング属性 country（国）のニーモニックで、特定の国の値に対応する相対識別名が保持されている。

ネット・サービス名 (net service name)

接続記述子に変換されるサービスの単純な名前。ユーザーは、接続するサービスに対する接続文字列内のネット・サービス名に従ってユーザー名およびパスワードを渡すことによって、接続要求を開始する。次に例を示す。

```
CONNECT username/password@net_service_name
```

必要に応じて、ネット・サービス名を次のような様々な場所に格納できる。

- 各クライアントのローカル構成ファイル（tnsnames.ora）
- ディレクトリ・サーバー
- Oracle Names サーバー
- NDS、NIS または CDS などの外部ネーミング・サービス

パーティション (partition)

一意の重複していないディレクトリ・ネーミング・コンテキスト。1つのディレクトリ・サーバーに格納されている。

パートナ・エージェント (partner agent)

Oracle Directory Integration Server がマッピング、スケジューリングおよびエラー処理を実行するためのディレクトリ統合エージェント。

バインド (binding)

ディレクトリに対して認証を行うプロセス。

ハッシュ (hash)

アルゴリズムを使用してテキスト文字列から生成された数値。ハッシュ値は、テキスト文字列より大幅に短くなる。ハッシュの数値は、セキュリティの目的とデータに対する高速アクセスの目的で使用する。

ハンドシェイク (handshake)

2 台のコンピュータが通信セッションを開始するために使用するプロトコル。

秘密鍵 (private key)

公開鍵暗号における秘密鍵。主に復号化に使用されるが、デジタル署名とともに暗号化にも使用される。

フィルタ (filter)

データ（通常、検索対象のデータ）を限定する方法。フィルタは常に識別名で表される。

例:cn=susie smith, o=acme, c=us

フェイルオーバー (failover)

障害の認識とリカバリのプロセス。

復号化 (decryption)

暗号化されたメッセージ（暗号文）の内容を、元の可読書式（平文）に変換する処理。

プロキシ・ユーザー (proxy user)

通常、ファイアウォールなどの中間層を備えた環境で利用されるユーザー。このような環境では、エンド・ユーザーは中間層に対して認証を行う。この結果、中間層はエンド・ユーザーにかわってディレクトリにログインする。プロキシ・ユーザーには ID を切り替える権限があり、一度ディレクトリにログインすると、エンド・ユーザーの ID に切り替える。次に、その特定のエンド・ユーザーに付与されている認可を使用して、エンド・ユーザーのかわりに操作を実行する。

プロビジョニング・アプリケーション (provisioned applications)

ユーザーおよびグループの情報が Oracle Internet Directory に一元化される環境にあるアプリケーション。これらのアプリケーションは、一般的に Oracle Internet Directory 内の該当する情報に対する変更と関連付けられる。

プロビジョニング・エージェント (provisioning agent)

Oracle 固有のプロビジョニング・イベントを外部またはサード・パーティのアプリケーション固有のイベントに変換するアプリケーションまたはプロセス。

プロファイル (profile)

「[ディレクトリ統合プロファイル \(directory integration profile\)](#)」を参照。

並行性 (concurrency)

複数の要求を同時に処理できる機能。並行性メカニズムの例には、スレッドやプロセスなどがある。

平文 (plaintext)

暗号化されていないメッセージ・テキスト。

変更ログ (change logs)

ディレクトリ・サーバーに加えられた変更を記録するデータベース。

マスター・サイト (master site)

レプリケーションにおいて、マスター定義サイト以外のサイトで、LDAP レプリケーションのメンバーであるサイト。

マスター定義サイト (master definition site: MDS)

レプリケーションにおいて、管理者が構成スクリプトを実行する Oracle Internet Directory のデータベース。

マッピング規則ファイル (mapping rules file)

Oracle Directory Integration Platform 環境で、Oracle Internet Directory の属性と [接続ディレクトリ](#) の属性との間のマッピングを指定するファイル。

メタディレクトリ (metadirectory)

企業のすべてのディレクトリ間で情報を共有するディレクトリ・ソリューション。すべてのディレクトリを1つの仮想ディレクトリに統合する。集中的に管理できるため、管理コストを削減できる。ディレクトリ間でデータが同期化されるため、企業内のデータに一貫性があり最新であることが保証される。

猶予期間ログイン (grace login)

パスワード期限切れ前の指定された期間内に行われるログイン。

リモート・マスター・サイト (remote master site: RMS)

レプリケート環境における [マスター定義サイト](#) 以外のサイトで、Oracle9i レプリケーションのメンバーであるサイト。

リレーショナル・データベース (relational database)

構造化されたデータの集合。同一の列のセットを持つ1つ以上の行で構成される表にデータが格納される。Oracle では、複数の表のデータを容易にリンクできる。このため、Oracle はリレーショナル・データベース管理システム、つまり RDBMS と呼ばれる。Oracle はデータを複数の表に格納し、さらに表間の関係を定義できる。このリンクは両方の表に共通の、1つ以上のフィールドに基づいて行われる。

ルート DSE (root DSE)

「[ルート・ディレクトリ固有のエントリ \(root directory specific entry\)](#)」を参照。

ルート・ディレクトリ固有のエントリ (root directory specific entry)

ディレクトリに関する操作情報を格納するエントリ。情報は複数の属性に格納されている。

レジストリ・エントリ (registry entry)

Oracle Internet Directory サーバーの起動（**ディレクトリ・サーバー・インスタンス**と呼ばれる）に関連する実行時情報が含まれているエントリ。レジストリ・エントリはディレクトリ自体に格納され、対応するディレクトリ・サーバー・インスタンスが停止するまで保持される。

レプリカ (replica)

ネーミング・コンテキストの個々のコピー。1 つのサーバー内に格納されている。

レプリケーション承諾 (replication agreement)

ディレクトリ・レプリケーション・グループ内のディレクトリ・サーバー間におけるレプリケーションの関係を記述する特別なディレクトリ・エントリ。

A

Access Control Information Item (ACI), 2-8
属性, 2-7
ディレクティブ
書式, 2-8
Access Control List (ACL), 2-7
ACI, 「Access Control Information Item (ACI)」を
参照
ACL, 「Access Control List (ACL)」を参照
add.log, 9-6

C

C API, 3-1
SSL モードでの使用方法, 3-55
概要, 3-4
サンプル検索ツール, 3-57
使用例, 3-55
非 SSL モードでの使用方法, 3-56
ファンクション
abandon, 3-38
abandon_ext, 3-38
add, 3-33
add_ext, 3-33
add_ext_s, 3-33
add_s, 3-33
compare, 3-25
compare_ext, 3-25
compare_ext_s, 3-25
compare_s, 3-25
count_entries, 3-47
count_references, 3-47
count_values, 3-50
count_values_len, 3-50

delete, 3-35
delete_ext, 3-35
delete_ext_s, 3-35
delete_s, 3-35
dn2ufn, 3-51
err2string, 3-42
explode_dn, 3-51
explode_rdn, 3-51
extended_operation, 3-37
extended_operation_s, 3-37
first_attribute, 3-48
first_entry, 3-47
first_message, 3-46
first_reference, 3-47
get_dn, 3-51
get_entry_controls, 3-53
get_option, 3-9
get_values, 3-50
get_values_len, 3-50
init, 3-8
init_ssl コール, 3-3
modify, 3-28
modify_ext, 3-28
modify_ext_s, 3-28
modify_s, 3-28
msgfree, 3-40
msgid, 3-40
msgtype, 3-40
next_attribute, 3-48
next_entry, 3-47
next_message, 3-46
next_reference, 3-47
open, 3-8
parse_extended_result, 3-42
parse_reference, 3-53

- parse_result, 3-42
- parse_sasl_bind_result, 3-42
- rename, 3-31
- rename_s, 3-31
- result, 3-40
- sasl_bind, 3-16
- sasl_bind_s, 3-16
- search, 3-20
- search_ext, 3-20
- search_ext_s, 3-20
- search_s, 3-20
- search_st, 3-20
- set_option, 3-9
- simple_bind, 3-16
- simple_bind_s, 3-16
- unbind, 3-19
- unbind_ext, 3-19
- unbind_s, 3-19
- value_free, 3-50
- value_free_len, 3-50
- リファレンス, 3-4
- C API の使用例, 3-55
- catldap.sql, 4-2
- changetype
 - add, 9-17
 - delete, 9-18
 - modify, 9-17
 - modrdn, 9-19

D

- DBMS_LDAP
 - アプリケーションの作成, 4-2
 - 概要, 4-1
 - 使用例
 - Java サンプル・コード, A-13
 - 概要, A-1
 - 検索, A-9
 - データベース・トリガー, A-2
- DBMS_LDAP_UTL
 - 概要, 7-1
 - グループ関連サブプログラム
 - create_group_handle ファンクション, 7-21
 - get_group_dn ファンクション, 7-26
 - get_group_properties ファンクション, 7-24

- set_group_handle_properties ファンクション, 7-23
- 概要, 7-4
- サブスクリバ関連サブプログラム
 - create_subscriber_handle ファンクション, 7-27
 - get_subscriber_dn ファンクション, 7-31
 - get_subscriber_properties ファンクション, 7-29
 - 概要, 7-4
- その他のサブプログラム
 - check_interface_version ファンクション, 7-41
 - create_mod_propertyset ファンクション, 7-38
 - free_handle プロシージャ, 7-41
 - free_mod_propertyset プロシージャ, 7-40
 - free_propertyset_collection プロシージャ, 7-37
 - get_property_names ファンクション, 7-33
 - get_property_values_len ファンクション, 7-36
 - get_property_values ファンクション, 7-35
 - normalize_dn_with_case ファンクション, 7-32
 - populate_mod_propertyset ファンクション, 7-39
 - 概要, 7-4
- データ型, 7-46
- ファンクション・リターン・コード, 7-42
- ユーザー関連サブプログラム
 - authenticate_user ファンクション, 7-5
 - check_group_membership ファンクション, 7-17
 - create_user_handle ファンクション, 7-7
 - get_group_membership ファンクション, 7-20
 - get_user_dn ファンクション, 7-15
 - get_user_extended_properties ファンクション, 7-13
 - get_user_properties ファンクション, 7-9
 - locate_subscriber_for_user ファンクション, 7-18
 - set_user_handle_properties ファンクション, 7-8
 - set_user_properties ファンクション, 7-11
 - 概要, 7-3
- リファレンス, 7-3
- DBMS_LDAP パッケージ, 2-10, 4-1
- 使用した検索, 2-17
- DES40 暗号化, 2-8
- DN, 「識別名」を参照

J

Java, 1-2
Java API リファレンス
 概要, 6-1
 クラス, 6-6
 クラスの説明, 6-2
 Group クラス, 6-4
 PropertySetCollection クラス, 6-5
 PropertySet クラス, 6-5
 Property クラス, 6-5
 Subscriber クラス, 6-3
 User クラス, 6-2
 例外, 6-66
JNDI, 1-2
JPEG イメージ、ldapadd を使用した追加, 9-6

K

Kerberos 認証, 9-5, 9-7, 9-12

L

LDAP
 機能モデル, 2-5
 検索フィルタ、IETF 準拠, 9-22
 情報モデル, 2-4
 セキュリティ・モデル, 2-6
 セッション
 初期化, 2-12, 3-8
 セッション・ハンドル・オプション, 3-9
 C API, 2-14
 操作、実行, 3-20
 データ交換フォーマット (LDIF), 9-2
 構文, 9-2
 ネーミング・モデル, 2-2
 バージョン 2 C API, 3-2
 メッセージ、結果の取得と確認, 3-40
 歴史, 2-2
ldapadd, 9-4
 JPEG イメージの追加, 9-6
 エントリの追加, 9-4
 構文, 9-4
ldapaddmt, 9-6
 構文, 9-6
 複数エントリを同時に追加, 9-6
 ログ, 9-6

ldapbind, 9-8
 構文, 9-8
ldap-bind 操作, 2-6
ldapcompare, 9-10
 構文, 9-10
ldapdelete, 9-11
 エントリの削除, 9-11
 構文, 9-11
ldapmoddn, 9-13
 構文, 9-13
ldapmodify, 9-15
 LDIF ファイル, 9-4, 9-6, 9-15, 9-20
 エントリの削除, 9-18
 グループ・エントリの作成, 9-17
 構文, 9-15
 属性値の置換, 9-18
 複数値の属性への値の追加, 9-17
 変更の種類, 9-17
ldapmodifymt, 9-20
 構文, 9-20
 使用, 9-20
 マルチスレッド処理, 9-21
ldapsearch, 3-57
 構文, 9-22
 フィルタ, 9-24
ldapsearch フィルタの例, 9-24
LDAP の歴史, 2-2
LDAP モデルの概要, 2-2
LDIF
 形式化規則, 9-3
 形式化の注意事項, 9-3
 構文, 9-2
 使用, 9-2
 ファイル、ldapmodify コマンド, 9-4, 9-6, 9-15, 9-20

M

MD4、パスワード暗号化, 2-8
MD5、パスワード暗号化, 2-9

O

OpenLDAP Community, xii
Oracle Directory Manager, 1-2
 属性の型のリスト, 9-3
Oracle Directory Replication Server, 1-2

- Oracle Internet Directory、コンポーネント、1-2
- Oracle Internet Directory サーバー、1-2
- Oracle Internet Directory でサポートされるオペレーティング・システム、1-3
- Oracle SSL 関連ライブラリ、3-70
- Oracle SSL コール・インタフェース、3-2、4-2
- Oracle SSL 拡張機能、3-2
- Oracle Wallet、3-3
- Oracle Wallet Manager、3-3
 - Wallet を作成するために必要、3-70
- Oracle Wallet、位置の変更、9-6、9-8、9-9、9-11、9-13、9-15、9-16、9-21、9-24
- Oracle Wallet パラメータ変更、9-6、9-8、9-9、9-11、9-13、9-15、9-16、9-21、9-24
- Oracle インスタンス、Glossary-13
- Oracle システム・ライブラリ、3-70
- Oracle の拡張機能
 - API の拡張機能
 - 概要と使用モデル、5-6
 - 機能の分類、5-7
 - 使用モデル、5-8
 - 前提、5-6
 - LDAP アクセス・モデル、5-2
 - LDAP のモデル化されたエンティティ
 - 概要、5-4
 - グループ、5-5
 - サブスクライバ、5-5
 - ユーザー、5-5
 - アプリケーション
 - インストール・ロジック、5-3
 - 起動とブートストラップに関するロジック、5-3
 - 削除ロジック、5-4
 - 実行時のロジック、5-3
 - 停止ロジック、5-4
 - 概要、5-1
 - プログラム抽象化
 - Java 言語、5-10
 - PL/SQL 言語、5-9
 - ユーザー管理機能、5-10

P

- PKI 認証、2-8
- PL/SQL API、4-1、4-2
 - C API の一部を含む、2-10
 - 依存性と制限事項、4-2

- 概要、4-3
- 検索方法、A-9
- サブプログラム、4-9
- データ型の概要、4-8
- データベース・トリガーからの使用、A-2
- データベースへのロード、4-2
- ファンクション
 - add_s、4-50
 - ber_free、4-62
 - bind_s、4-12
 - compare_s、4-16
 - count_entries、4-25
 - count_values、4-53
 - count_values_len、4-54
 - create_mod_array、4-42
 - dbms_ldap.init、4-10
 - delete_s、4-37
 - err2string、4-41
 - explode_dn、4-57
 - first_attribute、4-27
 - first_entry、4-22
 - get_dn、4-31
 - get_values、4-33
 - get_values_len、4-35
 - init、4-9
 - modify_s、4-48
 - modrdn2_s、4-39
 - msgfree、4-60
 - next_attribute、4-29
 - next_entry、4-23
 - open_ssl、4-58、4-60、4-62
 - rename_s、4-55
 - search_s、4-18
 - search_st、4-20
 - simple_bind_s、4-10
 - unbind_s、4-14
- プロシージャ
 - free_mod_array、4-52
 - populate_mod_array (バイナリ・バージョン)、4-46
 - populate_mod_array (文字列バージョン)、4-44
- リファレンス、4-3
- 例外の概要、4-6
- PL/SQL の使用例、4-2

R

RC4_40 暗号化, 2-8
RDN, 「相対識別名」を参照
RFC 1823, 3-70

S

SDK コンポーネント, 1-2
SHA (Secure Hash Algorithm)、パスワード暗号化, 2-9
Smith, Mark, xi
SQL*Plus, 4-2
SSL
 Oracle の拡張機能, 3-2
 暗号化と復号化, 3-2
 orclsslwalleturl パラメータの変更, 9-6, 9-8, 9-9, 9-11, 9-13, 9-15, 9-16, 9-21, 9-24
 Wallet, 3-3
 位置の変更, 9-6, 9-8, 9-9, 9-11, 9-13, 9-15, 9-16, 9-21, 9-24
 インタフェース・コール, 3-2
 クライアントとサーバーの認証, 2-7
 厳密認証, 2-8
 サーバー認証, 2-7
 使用可能, 9-6, 9-8, 9-9, 9-16, 9-21
 デフォルト・ポート, 2-7
 認証なし, 2-7
 認証モード, 3-2
 ハンドシェイク, 3-3
SSL をサポートする Oracle の拡張機能, 3-2

T

TCP/IP ソケット・ライブラリ, 3-70

U

UNIX Crypt、パスワード暗号化, 2-9

W

Wallet
 SSL, 3-3
 位置の変更, 9-6, 9-8, 9-9, 9-11, 9-13, 9-15, 9-16, 9-21, 9-24
 サポート, 3-3

あ

アクセス制御, 2-6, 2-7
 認可, 2-7
アプリケーション、作成
 C API を使用, 3-57
 PL/SQL LDAP API を使用, 4-2
暗号化
 DES40, 2-8
 Oracle Internet Directory で使用可能なレベル, 2-8
 RC4_40, 2-8
 パスワード, 2-8
 MD4, 2-8
 MD5, 2-9
 SHA, 2-9
 UNIX Crypt, 2-9
 デフォルト, 2-8
 パスワード用オプション, 2-8

い

依存性と制限事項, 3-70, 4-2
 C API, 3-70
 PL/SQL API, 4-2
インタフェース・コール、SSL, 3-2

え

エラー
 処理と結果の解析, 3-42
エントリ
 削除
 ldapdelete を使用, 9-11
 ldapmodify を使用, 9-18
 識別名, 2-2
 識別名を使用して位置を識別, 2-3
 追加
 ldapaddmt を使用, 9-6
 ldapadd を使用, 9-4
 同時, 9-6
 ネーミング, 2-2
 変更
 ldapmodifymt を使用して同時に, 9-20
 ldapmodify を使用, 9-15
 読込み, 3-25
エントリの子、リスト表示, 3-25

お

オブジェクト・クラス

 ldapaddmt を使用して同時に追加, 9-6

 LDIF ファイル, 9-2

オブジェクト、削除, 9-11, 9-15

か

カタログ管理ツール

 構文, 9-26

簡易認証, 2-6

管理ツール

 ldapadd, 9-4

 ldapaddmt, 9-6

 ldapbind, 9-8

 ldapcompare, 9-10

 ldapdelete, 9-11

 ldapmoddn, 9-13

 ldapmodify, 9-15

 ldapmodifymt, 9-20

関連文書, xi

き

規則、LDIF, 9-3

く

クライアントとサーバーの認証、SSL, 3-2

グループ・エントリ、ldapmodify を使用して作成,
9-17

け

結果、リストの参照, 3-46

権限, 2-6, 2-7

検索

 結果

 解析, 3-47

 フィルタ

 IETF 準拠, 9-22

 ldapsearch, 9-24

 有効範囲, 2-20

検索関連操作、流れ, 2-18

厳密認証, 2-6

こ

公開鍵

 インフラストラクチャ, 2-8

構文

 ldapadd, 9-4

 ldapaddmt, 9-6

 ldapbind, 9-8

 ldapcompare, 9-10

 ldapdelete, 9-11

 ldapmoddn, 9-13

 ldapmodify, 9-15

 ldapmodifymt, 9-20

 ldapsearch, 9-22

 LDIF, 9-2

 LDIF とコマンドライン・ツール, A-1

 カタログ管理ツール, 9-27

 コマンドライン・ツール, 9-4

 プロビジョニング・ツール, 9-28

コマンドライン・ツール

 ldapadd, 9-4

 ldapaddmt, 9-6

 ldapbind, 9-8

 ldapcompare, 9-10

 ldapdelete, 9-11

 ldapmoddn, 9-13

 ldapmodify, 9-15

 ldapmodifymt, 9-20

 ldapsearch, 9-22

 構文, 9-4

コントロール、使用, 3-14

コンポーネント

 Oracle Internet Directory SDK, 1-2

さ

サーバー認証の SSL, 2-7, 3-2

サンプル検索ツール、C API を使用して作成, 3-57

し

識別名, 2-2

 LDIF ファイル, 9-2

 コンポーネント, 2-3

 書式, 2-3

証明書, 2-6

証明書ベースの認証, 2-6
書式、識別名, 2-3

せ

整合性、データ, 2-8
セキュリティ、Oracle Internet Directory 環境, 2-6
セッション
 DBMS_LDAP を使用した終了の有効化, 2-22
 クローズ, 3-19
 初期化
 C API を使用, 2-12
 DBMS_LDAP を使用, 2-13
セッション固有のユーザー ID, 2-6

そ

操作属性
 ACI, 2-7
操作の中止, 3-38
相対識別名, 2-3
 ldapmodify を使用して変更, 9-19
属性
 LDIF ファイル, 9-2
 値, 2-5
 置換、ldapmodify を使用, 9-18
 型, 2-5
 削除
 ldapmodify を使用, 9-18
 値、ldapmodify を使用, 9-18
属性オプション
 ldapsearch を使用した検索, 9-25
追加
 ldapadd を使用, 9-4
 既存のエントリ, 9-4
 同時、ldapaddmt を使用, 9-6
属性オプション
 ldapsearch を使用した検索, 9-25
属性から値を削除, 9-18
属性の型, 2-5

て

ディレクティブ, 2-8
ディレクトリ情報ツリー, 2-2
データ
 整合性, 2-6, 2-8
 プライバシー, 2-6, 2-8
データ型の概要, 4-8

と

匿名認証, 2-6

に

認可, 2-6, 2-7
認可 ID, 2-6
認証, 2-6
 Kerberos, 9-5, 9-7, 9-12
 PKI, 2-8
 SSL, 2-6, 2-7, 3-2, 9-6, 9-8, 9-9, 9-16, 9-21
 クライアントとサーバー, 3-2
 サーバー, 3-2
 認証なし, 3-2
 SSL クライアントとサーバー, 2-7
 SSL サーバー, 2-7
 オプション, 2-6
 厳密, 2-6
 証明書ベース, 2-6
 ディレクトリ, 3-16
 ディレクトリ・サーバー
 有効化, 2-15
 有効化、C API を使用, 2-15
 有効化、DBMS_LDAP を使用, 2-15
 匿名, 2-6
 パスワード・ベース, 2-6
 モード、SSL, 3-2
認証局, 2-6

ね

ネーミング・エントリ, 2-2

は

パスワード

暗号化, 2-6, 2-8

MD4, 2-8

MD5, 2-9

SHA, 2-9

UNIX Crypt, 2-9

デフォルト, 2-8

暗号化オプション, 2-8

ポリシー, 2-9

パスワード・ベースの認証, 2-6

パフォーマンス

増加、複数のスレッドを使用, 9-7

バルク・ツール, 1-2

ふ

フィルタ, 2-21

IETF 準拠, 9-22

ldapsearch, 9-24

複数値の属性、値の追加, 9-17

複数のスレッド, 9-21

ldapaddmt, 9-6

数の増加, 9-7

プライバシー、データ, 2-6, 2-8

プロシージャ、PL/SQL

free_mod_array, 4-52

populate_mod_array (バイナリ・バージョン),
4-46

populate_mod_array (文字列バージョン), 4-44

プロビジョニング・ツール

構文, 9-28

文書、関連, xi

へ

ヘッダー・ファイルとライブラリ、必要, 3-57

変更の種類、ldapmodify 入力ファイル, 9-17

ま

マルチスレッド・コマンドライン・ツール

ldapaddmt, 9-6

ldapmodifymt, 9-21

れ

例外の概要, 4-6