Oracle® Internet Directory

アプリケーション開発者ガイド

10*g*（9.0.4）

部品番号 : B12378-01

2004 年 2 月

ORACLE®

Oracle Internet Directory アプリケーション開発者ガイド , 10*g*（9.0.4）

部品番号 : B12378-01

原本名 : Oracle Internet Directory Application Developer's Guide, 10*g* (9.0.4)

原本部品番号 : B10461-01

原本著者 : Richard Smith

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

目次

3　標準的な LDAP API に対する Oracle の拡張機能を使用したアプリケーションの開発

4 プロビジョニング統合アプリケーションの開発

5 Oracle Internet Directory サーバー・プラグインの開発

6 Oracle Delegated Administration Services に統合されたアプリケーションの開発

第 II 部 Oracle Internet Directory プログラミング・リファレンス

7 Oracle Internet Directory の C API

8 DBMS_LDAP PL/SQL リファレンス

9 DBMS_LDAP_UTL PL/SQL リファレンス

10 DAS_URL インタフェース・リファレンス

11 プロビジョニング統合 API リファレンス

第 III 部　付録

A　LDIF およびコマンドライン・ツールの構文

B 使用例

C DSML の構文

用語集

索引

図目次

表目次

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

『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、PL/SQL API、Java API および Oracle の拡張機能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 I 部 : Oracle Internet Directory および LDAP プログラミングの概念

第 1 章「概要」

Oracle Internet Directory Software Developer's Kit 10*g*（9.0.4）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「標準的な LDAP API を使用したアプリケーションの開発」

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「標準的な LDAP API に対する Oracle の拡張機能を使用したアプリケーションの開発」

LDAP API に対する Oracle の拡張機能の概念について説明します。また、拡張機能によってモデル化された抽象エンティティ、および Oracle の拡張機能の使用モデルについても説明します。

第 4 章「プロビジョニング統合アプリケーションの開発」

Oracle Directory Integration and Provisioning Platform の Oracle Directory Provisioning Integration Service を使用でき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。

第 5 章「Oracle Internet Directory サーバー・プラグインの開発」

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。

第 6 章「Oracle Delegated Administration Services に統合されたアプリケーションの開発」
DAS URL API を使用して DAS との統合を実現する方法について説明します。

第 II 部 : Oracle Internet Directory API リファレンス

第 7 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 8 章「DBMS_LDAP PL/SQL リファレンス」
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ために使用する 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。

第 9 章「DBMS_LDAP_UTL PL/SQL リファレンス」
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 パッケージに関するリファレンス情報を示します。

第 10 章「DAS_URL インタフェース・リファレンス」
DAS_URL API に対する Oracle の拡張機能について説明します。

第 11 章「プロビジョニング統合 API リファレンス」
Oracle Directory Integration and Provisioning Platform API に関するリファレンス情報を示します。

第 III 部 : 付録

付録 A「LDIF およびコマンドライン・ツールの構文」
LDAP Data Interchange Format（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。

付録 B「使用例」
サンプル・コードを提供します。

付録 C「DSML の構文」
DSML（XML）統合の構文および使用方法を紹介します。

用語集

関連ドキュメント

詳細は、次の Oracle ドキュメントを参照してください。

- Oracle9*i* データベース・サーバーおよび Oracle Application Server のマニュアル。特に、次の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』
  - 『Oracle9*i* アプリケーション開発者ガイド - 基礎編』
  - 『Oracle Application Server 10*g* セキュリティ・ガイド』

リリース・ノート、インストール関連ドキュメント、ホワイト・ペーパーまたはその他の関連ドキュメントは、OTN-J（Oracle Technology Network Japan）から、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。登録は、次の Web サイトから無償で行えます。

`http://otn.oracle.co.jp/membership/`

すでに OTN-J のユーザー名およびパスワードを取得している場合は、次の URL で OTN-J Web サイトのドキュメントの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）（`http://www.ietf.org`）の次の Web サイト
  - LDAPEXT の Charter と LDAP Draft
  - LDUP の Charter と Draft
  - RFC 2254、「The String Representation of LDAP Search Filters」
  - RFC 1823、「The LDAP Application Program Interface」
- `http://www.openldap.org`（OpenLDAP Community）

表記規則

この項では、このマニュアルの本文およびコード例で使用される表記規則について説明します。この項の内容は、次のとおりです。

本文の表記規則

本文では、特定の項目が一目でわかるように、次の表記規則を使用します。次の表に、その規則と使用例を示します。

規則	意味	例
太字	太字は、本文中で定義されている用語および用語集に記載されている用語を示します。	この句を指定すると、**索引構成表**が作成されます。
固定幅フォントの大文字	固定幅フォントの大文字は、システム指定の要素を示します。このような要素には、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、パッケージおよびメソッドがあります。また、システム指定の列名、データベース・オブジェクト、データベース構造、ユーザー名およびロールも含まれます。	NUMBER 列に対してのみ、この句を指定できます。 BACKUP コマンドを使用して、データベースのバックアップを作成できます。 USER_TABLES データ・ディクショナリ・ビュー内の TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	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ユーザーが指定する要素のサンプルを示します。このような要素には、コンピュータ名およびデータベース名、ネット・サービス名および接続識別子があります。また、ユーザーが指定するデータベース・オブジェクトとデータベース構造、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も含まれます。 **注意：**プログラム要素には、大文字と小文字を組み合せて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	sqlplus と入力して、SQL*Plus をオープンします。 パスワードは、orapwd ファイルで指定します。 /disk1/oracle/dbs ディレクトリ内のデータ・ファイルおよび制御ファイルのバックアップを作成します。 hr.departments 表には、department_id、department_name および location_id 列があります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 oe ユーザーとして接続します。 JRepUtil クラスが次のメソッドを実装します。

規則	意味	例
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、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。ここで、*old_release* とはアップグレード前にインストールしたリリースを示します。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus または他のコマンドライン文の例です。次のように固定幅フォントで表示され、通常のテキストと区別されます。

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

次の表に、コード例で使用される表記規則とその使用例を示します。

規則	意味	例
[ ]	大カッコは、カッコ内の項目を任意に選択することを表します。大カッコは、入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは、カッコ内の項目のうち、1 つが必須であることを表します。中カッコは、入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択項目の区切りに使用します。項目のうちの 1 つを入力します。縦線は、入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。	
	■ 例に直接関連しないコードの一部が省略されている。	CREATE TABLE ... AS *subquery*;
	■ コードの一部を繰り返すことができる。	SELECT *col1*, *col2*, ... , *coln* FROM employees;

規則	意味	例
. . .	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
その他の記号	大カッコ、中カッコ、縦線および省略記号以外の記号は、記載されているとおりに入力する必要があります。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
大文字	大文字は、システム指定の要素を示します。これらの要素は、ユーザー定義の要素と区別するために大文字で示されます。大カッコ内にないかぎり、表示されているとおりの順序および綴りで入力します。ただし、大 / 小文字が区別されないため、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などです。 **注意：**プログラム要素には、大文字と小文字を組み合せて使用するものもあります。これらの要素は、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;

Windows オペレーティング・システムでの表記規則

次の表に、Windows オペレーティング・システムでの表記規則とその使用例を示します。

規則	意味	例
「スタート」→	プログラムを起動する方法を示します。	Database Configuration Assistant を起動するには、「スタート」→「プログラム」→「Oracle - *HOME_NAME*」→「Configuration and Migration Tools」→「Database Configuration Assistant」を選択します。
ファイル名およびディレクトリ名	ファイル名およびディレクトリ名は大 / 小文字が区別されません。特殊文字の左山カッコ（<）、右山カッコ（>）、コロン（:）、二重引用符（"）、スラッシュ（/）、縦線（|）およびハイフン（-）は使用できません。円記号（¥）は、引用符で囲まれている場合でも、要素のセパレータとして処理されます。Windows では、ファイル名が ¥¥ で始まる場合、汎用ネーミング規則が使用されていると解釈されます。	`c:¥winnt"¥"system32` は `C:¥WINNT¥SYSTEM32` と同じです。
Windows コマンド・プロンプト	Windows コマンド・プロンプトには、カレント・ディレクトリが表示されます。コマンド・プロンプトのエスケープ文字はカレット（^）です。プロンプトには、作業中のサブディレクトリが表示されます。このマニュアルでは、コマンド・プロンプトと呼びます。	`C:¥oracle¥oradata>`
特殊文字	Windows コマンド・プロンプトで二重引用符（"）のエスケープ文字として円記号（¥）が必要な場合があります。丸カッコおよび一重引用符（'）にはエスケープ文字は必要ありません。エスケープ文字および特殊文字の詳細は、Windows オペレーティング・システムのドキュメントを参照してください。	`C:¥>exp scott/tiger TABLES=emp QUERY=¥"WHERE job='SALESMAN' and sal<1600¥"` `C:¥>imp SYSTEM/`*`password`* `FROMUSER=scott TABLES=(emp, dept)`
HOME_NAME	Oracle ホームの名前を表します。ホーム名には、英数字で 16 文字まで使用できます。ホーム名に使用可能な特殊文字は、アンダースコアのみです。	`C:¥> net start Oracle`*`HOME_NAME`*`TNSListener`

規則	意味	例
ORACLE_HOME および *ORACLE_BASE*	Oracle8*i* より前のリリースでは、Oracle コンポーネントをインストールすると、すべてのサブディレクトリが最上位の *ORACLE_HOME* の直下に置かれました。Windows NT の場合、デフォルトの位置は C:¥orant です。 このリリースは、Optimal Flexible Architecture（OFA）のガイドラインに準拠しています。*ORACLE_HOME* ディレクトリ下に配置されないサブディレクトリもあります最上位のディレクトリは *ORACLE_BASE* と呼ばれ、デフォルトでは C:¥oracle です。他の Oracle ソフトウェアがインストールされていないコンピュータに最新リリースの Oracle をインストールした場合、Oracle ホーム・ディレクトリは、デフォルトで C:¥oracle¥ora*nn* に設定されます。*nn* は最新のリリースの番号です。Oracle ホーム・ディレクトリは、*ORACLE_BASE* の直下に配置されます。 このマニュアルに示すディレクトリ・パスの例は、すべて OFA の表記規則に準拠しています。	*ORACLE_BASE*¥*ORACLE_HOME*¥rdbms¥admin ディレクトリへ移動します。

Oracle Internet Directory Software Developer's Kit の新機能

ここでは、Oracle Internet Directory Software Developer's Kit の最新リリースで導入された新機能について説明します。各項目には、関連項目が記載されています。

Oracle Internet Directory リリース 9.0.4 の新機能

- Oracle Delegated Administration Services URL API

  この API を使用すると、特定のディレクトリ操作を行うために委任管理者およびユーザーが使用できる管理セルフ・サービス・コンソールを構築できます。

  関連項目： 第 6 章「Oracle Delegated Administration Services に統合されたアプリケーションの開発」

- PL/SQL API の拡張機能。これらの拡張機能には次のものが含まれます。
  - LDAP v3 規格で導入された新規ファンクション。これまでは、コア C-API で使用可能でしたが、現在、PL/SQL でも使用可能になりました。
  - 中間層アプリケーションへのプロキシ・アクセスを可能にするファンクション。
  - Oracle Directory Integration and Provisioning Platform でプロビジョニング・プロファイルを作成および管理するファンクション。

  関連項目： 第 4 章「プロビジョニング統合アプリケーションの開発」

- 外部認証プラグインのサポート。この機能によって、管理者は、Microsoft Active Directory を使用して、Oracle コンポーネントで使用するセキュリティ資格証明を格納および管理できます。

  関連項目： 第 5 章「Oracle Internet Directory サーバー・プラグインの開発」

- DNS を使用したサーバー検出。この機能によって、Oracle Internet Directory のクライアントで、指定した企業で実行している Oracle ディレクトリ・サーバーのホスト名およびポート番号を検出できます。これによって、大規模な配置で Oracle Internet Directory のクライアントを維持する管理コストが削減されます。

  関連項目： 3-13 ページの「サーバー検出機能」

- XML インタフェース（DSML 1.0）の OID SDK とツールのサポート。この機能によって、LDAP ツールを使用して XML および LDIF を処理できます。Oracle Internet Directory の API は、DSML 形式での結果および操作をプログラムによって処理できます。

  関連項目： 付録 C「DSML の構文」

- クライアント側の参照キャッシュ。この新機能によって、クライアントで参照情報がキャッシュされ、その情報を使用して参照プロセスの処理時間が短縮されます。

  関連項目： 7-9 ページの「LDAP セッション・ハンドル・オプション」

第 I 部

Oracle Internet Directory プログラミングの概念

第 I 部では、Oracle Internet Directory の概要、基本的な LDAP プログラミングの概念および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。また、言語固有の拡張機能の概要についても簡単に説明します。

第 I 部は、次の章で構成されています。

1

概要

この章では、Oracle Internet Directory Software Developer's Kit 10*g*（9.0.4）の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 10*g*（9.0.4）の概要

Oracle Internet Directory SDK 10*g*（9.0.4）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 10*g*（9.0.4）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 環境でのアプリケーションの開発

この項では、次の項目について説明します。

- ディレクトリ対応アプリケーションのアーキテクチャ
- アプリケーションのライフサイクルでのディレクトリの相互作用
- アプリケーションと Oracle Internet Directory 統合のサービスおよび API
- 既存のアプリケーションと Oracle Internet Directory の統合
- 新しいアプリケーションと Oracle Internet Directory の統合

ディレクトリ対応アプリケーションのアーキテクチャ

ほとんどのディレクトリ対応アプリケーションは、複数のユーザーからの複数の要求を同時に処理するバックエンド・プログラムです。図 1-1 に、このような環境でのディレクトリの使用方法を示します。

図 1-1 ディレクトリ対応アプリケーション

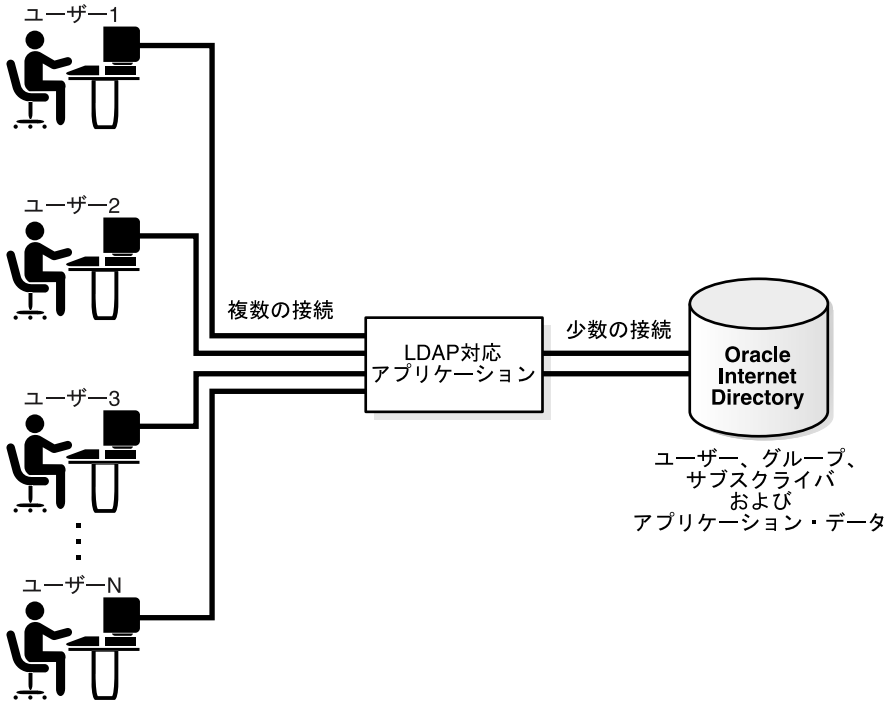


図 1-1 に示すように、ユーザー要求に LDAP 操作の実行が必要な場合、ディレクトリ対応アプリケーションは、あらかじめ確立されている Oracle Internet Directory への小規模な一連の接続を使用して、要求された操作を実行します。

アプリケーションのライフサイクルでのディレクトリの相互作用

表 1-1 に、アプリケーションがライフサイクルで行う通常のディレクトリの相互作用の概要を示します。

表 1-1 アプリケーションのライフサイクルでの相互作用

アプリケーションのライフサイクルでの操作	ロジック
アプリケーションのインストール	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	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できます。 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できます。

表 1-1 アプリケーションのライフサイクルでの相互作用（続き）

アプリケーションのライフサイクルでの操作	ロジック
アプリケーションの実行	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ことができます。 ■ LDAP 接続のプールから接続を選択します。 ■ Oracle Application Server 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す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	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	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。

アプリケーションと Oracle Internet Directory 統合のサービスおよび API

アプリケーション開発者は、表 1-2 で説明するサービスおよび API を使用して Oracle Internet Directory を統合することができます。

表 1-2 Oracle Internet Directory との統合のサービスおよび API

サービス /API	説明	詳細
C、PL/SQL および Java での標準的な LDAP API	これらの API によって、基本的な LDAP 操作が可能になります。Java で使用する標準的な LDAP API は、Sun 社の LDAP サービス・プロバイダで使用可能な JNDI API です。	第 2 章「標準的な LDAP API を使用したアプリケーションの開発」
標準的な C、PL/SQL および Java API に対する Oracle の拡張機能	これらの API によって、認証管理に関連する様々な概念をモデル化する、追加のプログラム・インタフェースを使用できます。	第 3 章「標準的な LDAP API に対する Oracle の拡張機能を使用したアプリケーションの開発」
Oracle Delegated Administration Services	Oracle Delegated Administration Services は、主要なセルフ・サービス・コンソールと管理インタフェースで構成され、サード・パーティ・アプリケーションをサポートするようにカスタマイズできます。	第 6 章「Oracle Delegated Administration Services に統合されたアプリケーションの開発」 『Oracle Internet Directory 管理者ガイド』の第 28 章
Oracle Directory Provisioning Integration Service	Oracle Provisioning Integration System は、サード・パーティのアプリケーションをプロビジョニングするために使用できます。また、その他のプロビジョニング・システムを統合する場合にも使用できます。	第 4 章「プロビジョニング統合アプリケーションの開発」 『Oracle Internet Directory 管理者ガイド』の第 33 章
Oracle Internet Directory プラグイン	Oracle Internet Directory プラグインを使用して、特定の配置例におけるディレクトリ・サーバーの動作をカスタマイズできます。	第 5 章「Oracle Internet Directory サーバー・プラグインの開発」 『Oracle Internet Directory 管理者ガイド』の第 44 章

図 1-2 に、1-6 ページの表 1-2 に示すサービスを利用するアプリケーションを示します。

図 1-2 API およびサービスを利用するアプリケーション

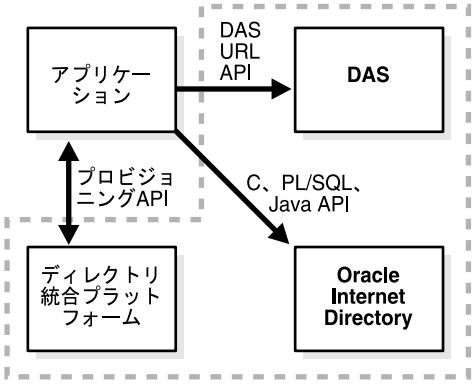


図 1-2 に示すとおり、アプリケーションは次のように Oracle Internet Directory と統合されています。

- Oracle Internet Directory の PL/SQL、C または Java API を使用して、Oracle Internet Directory に対して LDAP 操作を直接実行します。
- 特定の操作では、Oracle Delegated Administration Services のセルフ・サービス機能の一部をユーザーに送ります。
- Oracle Directory Provisioning Integration Service を使用して、Oracle Internet Directory 内の特定のユーザーまたはグループ・エントリに対する変更を通知します。

既存のアプリケーションと Oracle Internet Directory の統合

重要なビジネス・アプリケーションを実行するために特定のアプリケーションが企業内にすでに配置されている場合、表 1-3 に示す Oracle Internet Directory インフラストラクチャのサービスを使用して、既存のアプリケーションを変更できます。

表 1-3 既存のアプリケーションを変更するためのサービス

サービス	説明	詳細
ユーザーの自動プロビジョニング	カスタム・プロビジョニング・エージェントを開発し、Oracle Identity Management インフラストラクチャでのプロビジョニング・イベントに対応する、既存のアプリケーションでのユーザーのプロビジョニングを自動化できます。このエージェントを開発する場合は、Oracle Directory Provisioning Integration Service のインタフェースを使用する必要があります。	第 4 章「プロビジョニング統合アプリケーションの開発」
ユーザー認証サービス	既存のアプリケーションのユーザー・インタフェースが HTTP に基づいている場合は、Oracle HTTP Server と統合し、`mod_osso` を使用して URL を保護することで、Oracle Application Server Single Sign-On を使用してすべての着信ユーザー要求を認証することができます。	『Oracle Application Server Single Sign-On 管理者ガイド』
ユーザー・プロファイルの一元管理	既存のアプリケーションのユーザー・インタフェースが HTTP に基づいていて、Oracle Application Server Single Sign-On と統合して認証を行っている場合、アプリケーションで Oracle Internet Directory セルフ・サービス・コンソールを使用してユーザー・プロファイルを一元管理できます。セルフ・サービス・コンソールは、アプリケーションの特定の要件に対応できるように配置内でカスタマイズできます。	第 6 章「Oracle Delegated Administration Services に統合されたアプリケーションの開発」 『Oracle Internet Directory 管理者ガイド』の第 28 章

新しいアプリケーションと Oracle Internet Directory の統合

新しいアプリケーションを開発する場合、または既存のアプリケーションの新しいリリースを計画する場合に、Oracle Internet Directory インフラストラクチャで提供されるサービスを広範囲に使用できます。1-9 ページの表 1-4 に、統合に関する考慮事項を示します。

表 1-4 アプリケーション統合に関するポイント

統合に関するポイント	使用可能なオプション	詳細
ユーザー認証サービス	J2EE ベースのアプリケーションの場合、JAZN インタフェースで提供されるサービスを使用できます。OC4J に依存するアプリケーションの場合、`mod_osso` で提供されるサービスを使用してユーザーを認証し、HTTP ヘッダーに含まれるユーザーに関する重要な情報を取得できます。スタンドアロンの Web ベースのアプリケーションの場合、Oracle Application Server Single Sign-On API を使用してパートナ・アプリケーションになり、Oracle Application Server Single Sign-On を使用できます。Web ベース以外のアクセス・インタフェースを使用するアプリケーションの場合、Oracle Internet Directory LDAP API（C、PL/SQL および Java で使用可能）を使用してユーザーを認証できます。	『Oracle Application Server Containers for J2EE ユーザーズ・ガイド』 『Oracle Application Server Single Sign-On 管理者ガイド』 第 II 部「Oracle Internet Directory プログラミング・リファレンス」に示す様々な LDAP API についてのリファレンス情報
ユーザー認可サービス	J2EE ベースのアプリケーションの場合、JAZN インタフェースで提供されるサービスを使用して、アプリケーション定義のリソースに対してユーザー認可を実装および適用できます。Oracle Internet Directory で認可をグループとしてモデル化し、グループ・メンバーシップの確認によるユーザー認可を行うことができます。Oracle Internet Directory LDAP API（C、PL/SQL および Java で使用可能）を使用して、これを実行できます。	『Oracle Application Server Containers for J2EE ユーザーズ・ガイド』 第 II 部「Oracle Internet Directory プログラミング・リファレンス」に示す様々な LDAP API についてのリファレンス情報

表 1-4 アプリケーション統合に関するポイント（続き）

統合に関するポイント	使用可能なオプション	詳細
プロファイルの一元管理	アプリケーション固有のプロファイルおよびユーザー設定項目を Oracle Internet Directory の属性としてモデル化できます。	『Oracle Internet Directory 管理者ガイド』の第 IV 部を参照してください。
	アプリケーションのユーザー・インタフェースが HTTP に基づいていて、Oracle Application Server Single Sign-On と統合して認証を行っている場合、アプリケーションで Oracle Internet Directory セルフ・サービス・コンソールを使用してユーザー・プロファイルを一元管理できます。セルフ・サービス・コンソールは、アプリケーションの特定の要件に対応できるように配置内でカスタマイズできます。	第 6 章「Oracle Delegated Administration Services に統合されたアプリケーションの開発」 『Oracle Internet Directory 管理者ガイド』の第 28 章
	Oracle Internet Directory LDAP API（C、PL/SQL および Java で使用可能）を使用して、実行時にプロファイルを取得できます。	第 II 部に示す様々な LDAP API についてのリファレンス情報
ユーザーの自動プロビジョニング	アプリケーションのユーザー・インタフェースが HTTP に基づいていて、Oracle Application Server Single Sign-On と統合して認証を行っている場合、ユーザーが初めてアプリケーションにアクセスする際にユーザーの自動プロビジョニングを実装できます。	第 4 章「プロビジョニング統合アプリケーションの開発」
	Oracle Identity Management Infrastructure 内のアプリケーションを Oracle Directory Provisioning Integration Service と統合できます。管理アクション（識別情報の追加、変更、削除など）に応じて、アプリケーションで自動的にユーザー・アカウントのプロビジョニングまたはプロビジョニングの解除を行うことができます。	

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

次に示す Oracle Internet Directory 10*g*（9.0.4）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 サーバーおよびクライアントは、次のオペレーティング・システムをサポートしています。

- HP-UX（64 ビット）- 11.0 および 11i
- Linux（32 ビット）- Red Hat AS 2.1 および United Linux 1.0
- AIX 5L（64 ビット）- 5.1 および 5.2
- HP Tru64 - 5.1b

2

標準的な LDAP API を使用したアプリケーションの開発

この章では、標準的な LDAP 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、**Lightweight Directory Access Protocol（LDAP）**の概要を理解し、標準的な API と統合するための基礎知識を習得できます。

この章では、次の項目について説明します。

LDAP の歴史

LDAP 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**（DN）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**（DIT）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

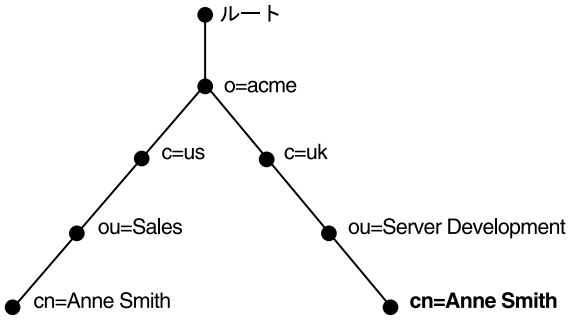


図 2-1 のディレクトリ情報ツリーは、Acme Corporation に所属する、Anne Smith という同じ名前を持つ 2 人の従業員のエントリを図示しています。この図のディレクトリ情報ツリーは、地理的および組織的な線で構造化されています。左の分岐で表されている Anne Smith は米国の販売部門に勤務し、もう一方の Anne Smith は英国のサーバー開発部門に勤務しています。

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。)

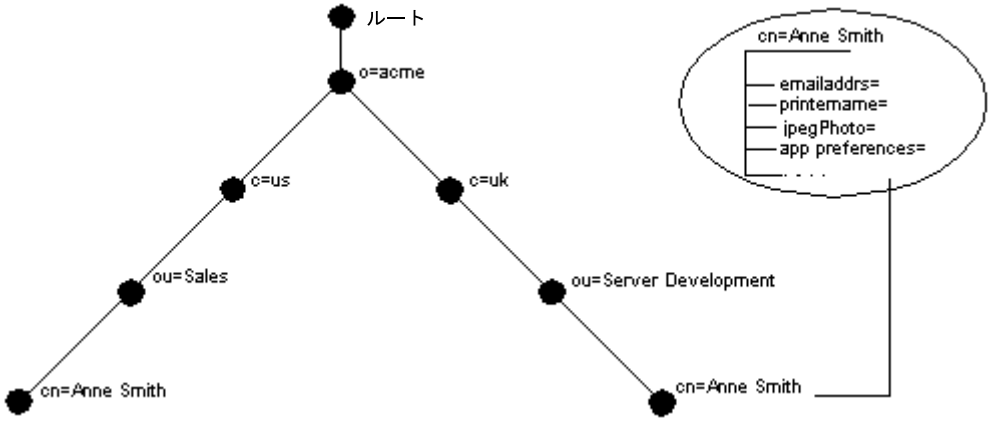
LDAP 情報モデル

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。`emailaddrs`、`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。

表 2-1 LDAP ファンクション

ファンクション	説明
検索および読取り	読取り操作では、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、サーバーの境界を超えている場合でも、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	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。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。

表 2-1 LDAP ファンクション（続き）

ファンクション	説明
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取り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 : 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、信頼でき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。

表 2-2 SSL 認証モード

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 つの基本手順が含まれます。

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取り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。

アクセス制御情報項目（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、アクセス制御リスト（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

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** の改良型で、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がです。

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、総当り攻撃や反転攻撃に対応できます。
- **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

標準的な LDAP API の概要

標準的な LDAP を使用すると、前の項で説明した基本的な LDAP 操作を実行できます。標準的な LDAP API は、次の言語で使用可能です。

- C—Oracle Internet Directory Software Developer's Kit の一部
- PL/SQL—DBMS_LDAP としての Oracle Internet Directory Software Developer's Kit の一部
- Java—Sun 社の JNDI パッケージの一部

これらのすべての API は、TCP/IP 接続を使用します。LDAP バージョン 3 に基づき、また、Oracle Internet Directory への SSL 接続をサポートします。

この項では、次の項目について説明します。

- API の使用モデル
- C API の概要
- Java API の概要
- DBMS_LDAP パッケージの概要

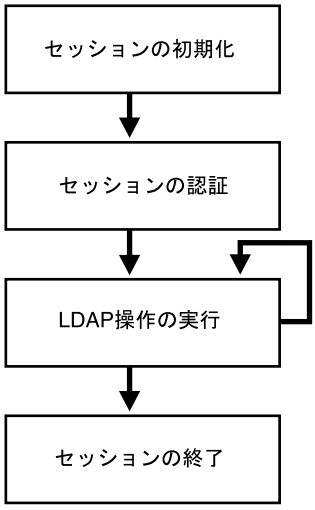
API の使用モデル

一般的に、アプリケーションでは、次の 4 つの手順に従って 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 2-3 に、これらの手順を示します。

図 2-3 DBMS_LDAP の一般的な使用手順



この章の後半で、各手順での API の重要な機能について説明します。

C API の概要

C API を使用してアプリケーションを作成するには、次のことが必要です。

- *$ORACLE_HOME*/ldap/public/ldap.h にあるヘッダー・ファイルを含める。
- *$ORACLE_HOME*/lib/libclntsh.so.9.0 にあるライブラリに動的にリンクする。

> **関連項目：** SSL モードおよび非 SSL モードの使用方法については、7-61 ページの「C API の使用例」を参照してください。

Java API の概要

Java 開発者は、Sun 社の JNDI LDAP サービス・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。

> **関連項目：** Sun 社の JNDI プロバイダの詳細は、http://java.sun.com を参照してください。

DBMS_LDAP パッケージの概要

DBMS_LDAP 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。

PL/SQL LDAP API を使用するには、データベースにロードします。`$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。その場合、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。また、データベースが存在する ORACLE ホームで SQL*Plus を実行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・シーケンスの例を次に示します。

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

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。

この項では、次の項目について説明します。

- C API を使用したセッションの初期化
- JNDI を使用したセッションの初期化
- DBMS_LDAP を使用したセッションの初期化

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 2-3 ldap_init() のパラメータ

パラメータ	説明
hostname	空白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。 **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。

JNDI を使用したセッションの初期化

関連項目： Sun 社の JNDI プロバイダの詳細は、http://java.sun.com を参照してください。

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。同時にアクティブにできる LDAP 接続の上限は 64 です。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。

LDAP セッションの認証

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。`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。

この項では、次の項目について説明します。

- C API を使用した LDAP セッションの認証
- JNDI を使用した LDAP セッションの認証
- DBMS_LDAP を使用した LDAP セッションの認証

C API を使用した LDAP セッションの認証

ldap_simple_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。

ldap_simple_bind_s() ファンクションの構文は、次のとおりです。

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

表 2-4 ldap_simple_bind_s() の引数

引数	説明
ld	有効な LDAP セッション・ハンドル
dn	アプリケーションが認証で使用する識別情報
passwd	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。

JNDI を使用した LDAP セッションの認証

認証を実行するための特別なファンクションはありません。必要な認証パラメータは、初期化時に設定されます。

> **関連項目：** Sun 社の JNDI プロバイダの詳細は、http://java.sun.com を参照してください。

DBMS_LDAP を使用した LDAP セッションの認証

simple_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、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を示します。

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
    retvalPLS_INTEGER;
    my_sessionDBMS_LDAP.session;

BEGIN
    retval:= -1;
    -- Initialize the LDAP session
    my_session:= DBMS_LDAP.init('yow.acme.com',389);
    --Authenticate 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。

ディレクトリの検索

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。

この項では、次の項目について説明します。

- 検索関連操作の流れ
- 検索有効範囲
- フィルタ
- C API を使用したディレクトリの検索
- JNDI を使用したディレクトリの検索
- DBMS_LDAP を使用したディレクトリの検索

注意： 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。

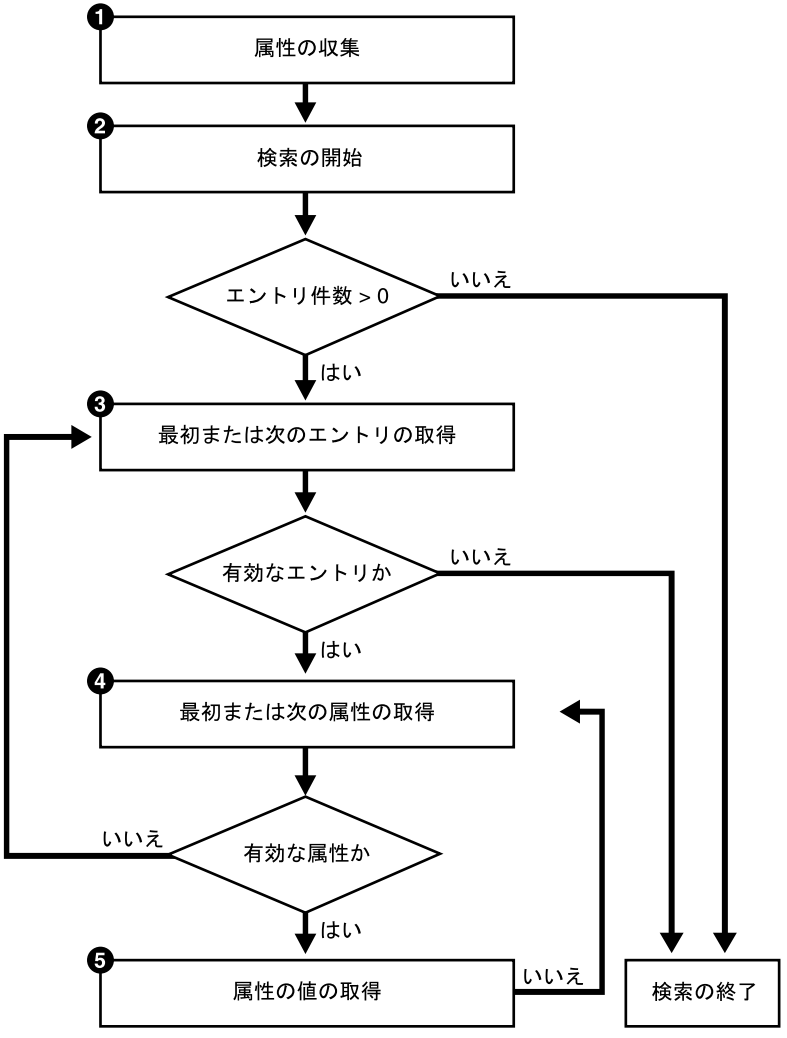
検索関連操作の流れ

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ミングは、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、その属性を配列に設定します。
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。
3. 結果セットからエントリを取得します。
4. 手順 3 で取得したエントリの属性を取得します。
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。
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 2-4 に、前述の手順の詳細を示します。

図 2-4 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。

表 2-5 search_s() または search_st() ファンクションのオプション

オプション	説明
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 2-5 に、この 3 つの有効範囲オプションの違いを示します。

図 2-5 3 つの有効範囲オプション

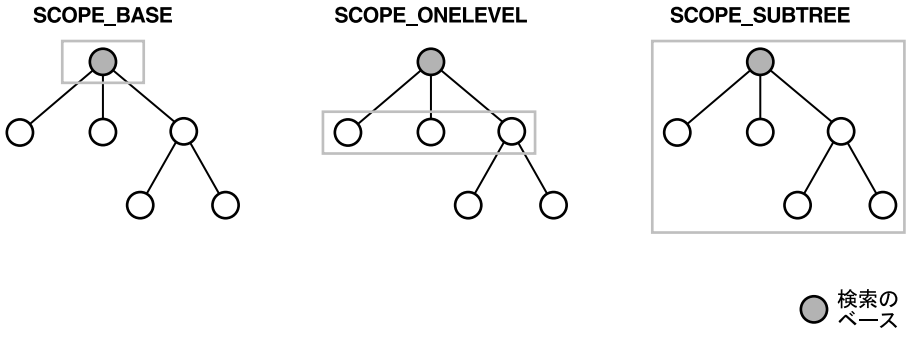


図 2-5 では、検索のベースはグレーの丸で表されています。検索対象のエントリは、薄い色の四角形で囲まれています。

フィルタ

search_s() および search_st() ファンクション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 2-6 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 2-6 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 2-7 に、基本的なブール演算子を示します。

表 2-7 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。

C API を使用したディレクトリの検索

ldap_search-s() ファンクションを使用すると、ディレクトリ内の同期検索操作の要求を開始できます。

ldap_search_s() の構文は、次のとおりです。

```
int ldap_search_s
(
LDAP*ld,
char*base,
intscope,
char*filter,
intattrsonly,
LDAPMessage**res,
);
```

検索操作の流れ :

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ミングは、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、NULL 文字で終了する配列を持つ文字列の配列にその属性を設定します。
2. ldap_search_s() ファンクションを使用して、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。
3. ldap_first_entry() または ldap_next_entry() ファンクションを使用して、結果セットからエントリを取得します。
4. ldap_first_attribute() または ldap_next_attribute() ファンクションを使用して、手順 3 で取得したエントリの属性を取得します。
5. ldap_get_values() または ldap_get_values_len() ファンクションを使用して、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。
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。

表 2-8 ldap_search_s() の引数

引数	説明
ld	有効な LDAP 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。
attrsonly	1 に設定すると、属性のみが戻されます。
res	この引数で検索結果が戻されます。

JNDI を使用したディレクトリの検索

関連項目 : Sun 社の JNDI プロバイダの詳細は、`http://java.sun.com` を参照してください。

DBMS_LDAP を使用したディレクトリの検索

DBMS_LDAP API で検索を開始するためのファンクションは DBMS_LDAP.search_s() 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、表 2-9 に示す引数を取ります。

表 2-9 DBMS_LDAP.search_s() および DBMS_LDAP.search_st() の引数

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 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ミングは、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. DBMS_LDAP.search_s() または DBMS_LDAP.search_st() ファンクションを使用して、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。
3. DBMS_LDAP.first_entry() または DBMS_LDAP.next_entry() ファンクションを使用して、結果セットからエントリを取得します。
4. DBMS_LDAP.first_attribute() または DBMS_LDAP.next_attribute() ファンクションを使用して、手順 3 で取得したエントリの属性を取得します。
5. DBMS_LDAP.get_values() または DBMS_LDAP.get_values_len() ファンクションを使用して、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。
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。

セッションの終了

この項では、次の項目について説明します。

- C API を使用したセッションの終了
- JNDI を使用したセッションの終了
- DBMS_LDAP を使用したセッションの終了

C API を使用したセッションの終了

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、ldap_unbind_s() をコールして行います。

ldap_unbind_s() ファンクションの構文は、次のとおりです。

```
int ldap_unbind_s
(
LDAP* ld
);
```

ldap_unbind_s() ファンクションのコールが正常終了すると、ディレクトリ・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・リソースの割当てが解除されて、整数 LDAP_SUCCESS がコール元に戻されます。特定のセッションで ldap_unbind_s() ファンクションを起動した後は、ldap_init() をコールして新しい LDAP セッションを初期化しないかぎり、そのセッションで他の LDAP 操作を実行することはできません。

JNDI を使用したセッションの終了

関連項目： Sun 社の JNDI プロバイダの詳細は、http://java.sun.com を参照してください。

DBMS_LDAP を使用したセッションの終了

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。

3

標準的な LDAP API に対する Oracle の拡張機能を使用したアプリケーションの開発

この章では、LDAP API に対する Oracle の拡張機能の概念について説明します。また、拡張機能によってモデル化された抽象エンティティ、およびこれらの拡張機能の使用モデルについても説明します。

この章では、次の項目について説明します。

標準的な LDAP API に対する Oracle の拡張機能の概要

API の拡張機能によって提供される機能は、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せることができます。
- レルム管理－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認証管理レルム関連のプロパティを取得または設定できます。
- サーバー検出管理－この機能によって、アプリケーションは、ドメイン・ネーム・システム（DNS）のディレクトリ・サーバーの位置を特定できます。
- SASL 管理－この機能によって、アプリケーションは、SASL Digest-MD5 認証を使用してディレクトリに対する認証を受けることができます。

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ホスト企業に対する LDAP 検索が必要な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。次の項目について説明します。

- PL/SQL での API 拡張機能の使用
- Java での API 拡張機能の使用
- 標準的な API に対する Oracle の拡張機能のインストールと初回の使用

> **関連項目：** 使用モデルの概要については、1-3 ページの「ディレクトリ対応アプリケーションのアーキテクチャ」を参照してください。

図 3-1 に、既存の API に関連した API の拡張機能の配置を示します。

図 3-1 Oracle API 拡張機能

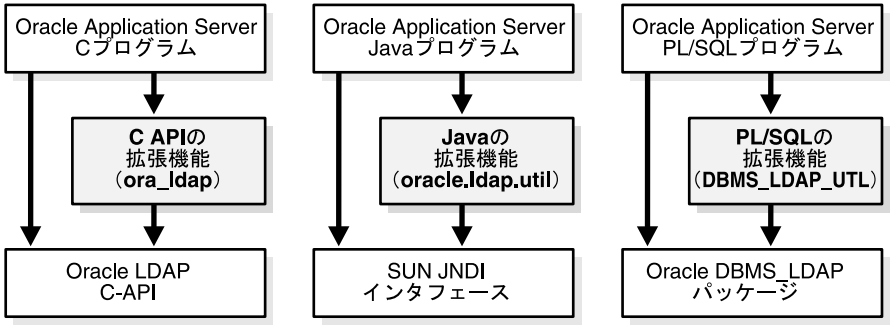


図 3-1 に示すように、PL/SQL および Java 言語では、既存の API 拡張機能は、既存の API の層の上に位置します。

- 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
- C プログラムについては、Oracle 社の LDAP C API

アプリケーションは、接続の確立およびクローズ、API 拡張機能ではカバーされないディレクトリ・エントリの検索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。

図 3-2 に、この章で説明する 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ます。

図 3-2 API の拡張機能のプログラム・フロー

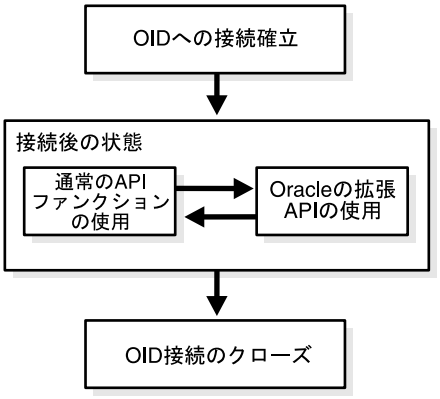


図 3-2 のように、アプリケーションは、最初に Oracle Internet Directory への接続を確立します。その後、既存の API ファンクションと API の拡張機能を交互に使用できます。

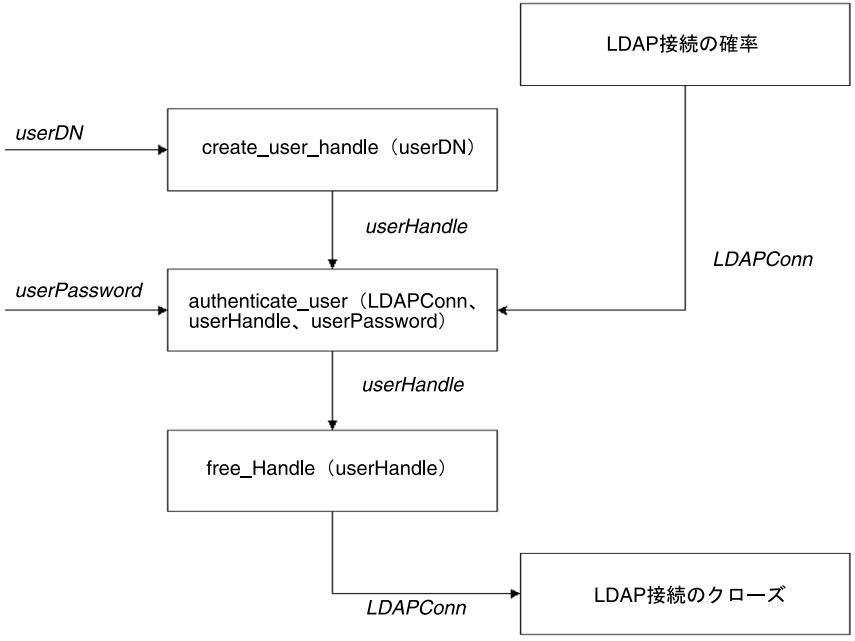
PL/SQL での API 拡張機能の使用

この章で説明するほとんどの拡張機能は、ユーザー、グループ、レルム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 Oracle Application Server 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 3-3 に、この使用モデルを示します。

図 3-3 PL/SQL 言語のプログラム抽象化



Java での API 拡張機能の使用

この項の内容は次のとおりです。

- `oracle.java.util` パッケージ
- PropertySetCollection、PropertySet および Property クラス

oracle.java.util パッケージ

LDAP エンティティ（つまり、ユーザー、グループ、レルム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、ユーザーを認証するには、アプリケーションは次の手順に従います。

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。

PropertySetCollection、PropertySet 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 オブジェクトで表され、識別名で識別されます。PropertySet には、Property として表される属性が含まれる場合があります。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));
    }
}
```

エンティティ myuser は User オブジェクトです。psc オブジェクトには、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、B-30 ページの「Java サンプル・コード」を参照してください。

標準的な API に対する Oracle の拡張機能のインストールと初回の使用

表 3-1 に、各 API のインストールと初回の使用に関する情報を示します。

表 3-1 インストールと初回の使用に関する情報

言語	インストールと初回の使用に関する情報
Java API	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	Oracle9*i* データベース・サーバーの一部としてインストールされます。$*ORACLE_HOME*/rdbms/admin に格納されているスクリプト catldap.sql を使用して、ロードする必要があります。
C API	C API を使用してアプリケーションを作成するには、次のことが必要です。 $*ORACLE_HOME*/ldap/public/ldap.h にあるヘッダー・ファイルを含める。 $*ORACLE_HOME*/lib/libclntsh.so.9.0 にあるライブラリに動的にリンクする。

ユーザー管理機能

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

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

この項では、次の項目について説明します。

ユーザー管理 API

この項では、各 API のユーザー管理機能について説明します。

ユーザー管理機能の Java API

前述の例で説明するように、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。この機能の高度な使用モデルは、次のとおりです。

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. 必要な場合は、`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)` 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。

ユーザー管理機能の C API

Oracle Internet Directory 10*g*（9.0.4）では、ユーザー管理機能に対して C API はサポートされていません。

ユーザー認証

この項では、Java、PL/SQL および C 言語の LDAP API に関するユーザー認証機能について説明します。

ユーザー認証の Java API

ユーザー認証は、基本的に特定の属性とその属性値を比較する一般的な LDAP 操作です。Oracle Internet Directory では、次のものがサポートされています。

- 認証時に使用可能な任意の属性。
- 認証メソッドによって戻される任意のパスワード・ポリシー例外。ただし、パスワード・ポリシーは、10*g*（9.0.4）では `userpassword` 属性のみに適用されることに注意してください。

次に、使用方法を示します。

```
        // User user1 - is a valid User Object
        try
        {
                user1.authenticateUser(ctx,
User.CREDTYPE_PASSWD, ?welcome?);

                // or
                // user1.authenticateUser(ctx, <any
attribute>, <attribute value>);
        }
        catch (UtilException ue)
        {
                // Handle the password policy error
accordingly
                if (ue instanceof PasswordExpiredException)
                        // do something
                else if (ue instanceof GraceLoginException)
                        // do something
        }
```

ユーザー認証の C API

Oracle Internet Directory 10*g*（9.0.4）では、ユーザー認証機能に対して C API はサポートされていません。

ユーザーの作成

この項では、Java、PL/SQL および C 言語の LDAP API に関するユーザー作成機能について説明します。

ユーザー作成の Java API

`subscriber` クラスによって `createUser()` メソッドが提供され、プログラムによってユーザーが作成されます。ユーザー・エントリに必要なオブジェクト・クラスは、Oracle Delegated Administration Services を介して設定可能です。`createUser()` メソッドは、ユーザーの作成時に、クライアントが要件を理解し必須の属性に対して値を提供することを前提にしています。プログラマによって必須情報が提供されない場合、サーバーはエラーを戻します。

次に、使用方法を示します。

```
// Subscriber sub is a valid Subscriber object
// DirContext ctx is a valid DirContext

// Create ModPropertySet object to define all the attributes and their values.
ModPropertySet mps = new ModPropertySet();
mps.addProperty(LDIF.ATTRIBUTE_CHANGE_TYPE_ADD,?cn?, ?Anika?);
mps.addProperty(LDIF.ATTRIBUTE_CHANGE_TYPE_ADD,?sn?, ?Anika?);
mps.addProperty(LDIF.ATTRIBUTE_CHANGE_TYPE_ADD,?mail?,
?Anika@oracle.com?);

// Create user by specifying the nickname and the ModPropertySet defined above
User newUser = sub.createUser( ctx, mps);

// Print the newly created user DN
System.out.println( newUser.getDN(ctx) );

// ? perform other operations with this new user
```

ユーザー作成の PL/SQL API

Oracle Internet Directory 10*g*（9.0.4）では、ユーザー作成機能に対して PL/SQL API はサポートされていません。

ユーザー作成の C API

Oracle Internet Directory 10*g*（9.0.4）では、ユーザー作成機能に対して C API はサポートされていません。

User オブジェクトの取得

この項では、Java、PL/SQL および C 言語の LDAP API に関する User オブジェクト取得機能について説明します。

User オブジェクト取得の Java API

subscriber クラスによって getUser() メソッドが提供され、User クラスのパブリック・コンストラクタと置き換えられます。このメソッドは、指定された情報に基づいて User オブジェクトを戻します。

次に、使用方法を示します。

```
// DirContext ctx is contains a valid OID connection with
sufficient privilege to perform the operations

// Creating RootOracleContext object
RootOracleContext roc = new RootOracleContext(ctx);

// Obtain a Subscriber object representing the default
subscriber
Subscriber sub = roc.getSubscriber(ctx,
Util.IDTYPE_DEFAULT, null, null);

// Obtain a User object representing the user whose
nickname is ?Anika?
User user1 = sub.getUser(ctx, Util.IDTYPE_SIMPLE, ?Anika?,
null);
// ? do work with this user

The getUser() method can retrieve users based on DN, GUID
and simple name.  A getUsers() method is also available to
perform a filtered search to return more than one user at a
time.  The returned object is an array of User objects.
For example,

// Obtain an array of User object where the users? nickname
starts with ?Ani?
User[] userArr = sub.getUsers(ctx, Util.IDTYPE_SIMPLE,
?Ani*?, null);
// ? do work with the User array
```

User オブジェクト取得の PL/SQL API

Oracle Internet Directory 10*g*（9.0.4）では、User オブジェクト取得機能に対して PL/SQL API はサポートされていません。

User オブジェクト取得の C API

Oracle Internet Directory 10*g*（9.0.4）では、User オブジェクト取得機能に対して C API はサポートされていません。

グループ管理機能

この項では、Java、PL/SQL および C 言語の LDAP API に関するグループ管理機能について説明します。

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、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

認証管理レルム機能

この項では、Java、PL/SQL および C 言語の LDAP API に関する認証管理レルム機能について説明します。

認証管理レルムは、多数の Oracle 製品によって提供される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、レルム・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）を取得するために、Oracle Internet Directory にアクセスする必要があります。

通常、レルム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
- 単純な企業名

Realm オブジェクト取得の Java API

`RootOracleContext` クラスはルート Oracle コンテキストを表します。認証管理レルムの作成に必要な情報のほとんどは、ルート Oracle コンテキストに格納されています。`RootOracleContext` クラスでは `getSubscriber()` メソッドが提供されます。このメソッドは、`subscriber` クラスのパブリック・コンストラクタと置き換えられ、指定された情報に基づいて認証管理レルム・オブジェクトを戻します。

次に、使用方法を示します。

```
        // DirContext ctx is contains a valid OID
connection with sufficient privilege to perform the
operations

        // Creating RootOracleContext object
        RootOracleContext roc = new RootOracleContext(ctx);

        // Obtain a Subscriber object representing the
Subscriber with simple name ?Oracle?
        Subscriber sub = roc.getSubscriber(ctx,
Util.IDTYPE_SIMPLE, ?Oracle?, null);

     // ? do work with the Subscriber object
```

サーバー検出機能

ディレクトリ・サーバー検出（DSD）を使用すると、ディレクトリ・クライアントによってOracle ディレクトリ・サーバーを自動検出できます。これにより、配置内で、ディレクトリのホスト名およびポート番号情報を中央の DNS サーバーで管理できます。すべてのディレクトリ・クライアントが実行時に DNS 問合せを行い、ディレクトリ・サーバーに接続します。ディレクトリ・サーバーの位置情報は、DNS サービス位置レコード（SRV）を使用して格納されます。

SRV には次のものが含まれます。

- LDAP サービスを提供するサーバーの DNS 名
- 対応するポートのポート番号
- クライアントが複数のサーバーから適切なサーバーを選択できるようにする任意のパラメータ

また、DSD によって、クライアントが `ldap.ora` ファイル自体からディレクトリのホスト名情報を検出できます。

この項では、次の項目について説明します。

- Oracle Internet Directory の検出インタフェースの利点
- 検出インタフェースの使用モデル
- DNS からのサーバー名およびポート番号の判断
- DNS サーバー検出の環境変数
- DNS サーバー検出のプログラミング・インタフェース

- サーバー検出の Java API
- 例 : ディレクトリ・サーバー検出の Java API

関連項目：

- 'Discovering LDAP Services with DNS' by Michael P. Armijo（http://www.ietf.org/）
- 'A DNS RR for specifying the location of services (DNS SRV)', Internet RFC 2782（http://www.ietf.org/）

Oracle Internet Directory の検出インタフェースの利点

通常、LDAP ホスト名およびポート情報は、`$ORACLE_HOME/network/admin` のクライアント上に格納されている `ldap.ora` という名前のファイルで静的に提供されます。多数のクライアントを持つ大規模な配置の場合、この情報の管理は非常に煩雑になります。たとえば、ディレクトリ・サーバーのホスト名またはポート番号が変更されるたびに、各クライアントの `ldap.ora` ファイルを変更する必要があります。

ディレクトリ・サーバー検出を使用すると、`ldap.ora` のホスト名およびポート番号を管理する必要がなくなります。ホスト名情報は中央の DNS サーバーに格納されているため、更新するのは 1 回のみです。すべてのクライアントで、接続時に DNS から新しいホスト名情報を動的に検出できます。

DSD では、取得する際に使用するメカニズムや規格にかかわらず、ディレクトリ・サーバー情報の取得に単一のインタフェースが提供されます。現在、Oracle ディレクトリ・サーバーの情報は、DNS または `ldap.ora` のいずれかから単一のインタフェースを使用して取得できます。

検出インタフェースの使用モデル

ホスト名情報を検出するには、まず検出ハンドルを作成します。検出ハンドルは、ホスト名情報の検出元を指定します。Java API の場合、検出ハンドルを作成するには `oracle.ldap.util.discovery.DiscoveryHelper` クラスのインスタンスを作成します。

```
DiscoveryHelper disco = new DiscoveryHelper(DiscoveryHelper.DNS_DISCOVER);
```

引数 `DiscoveryHelper.DNS_DISCOVER` は、ソース（この場合、DNS）を指定します。

各ソースには、ホスト名情報を検出するために指定するいくつかの入力項目があります。DNS の場合、入力項目は次のとおりです。

- ドメイン名
- 検出メソッド
- sslmode

これらのオプションの詳細は、DNS からのサーバー名およびポート番号の判断を参照してください。

```
// Set the property for the DNS_DN
disco.setProperty(DiscoveryHelper.DNS_DN,"dc=us,dc=fiction,dc=com");
// Set the property for the DNS_DISCOVER_METHOD
disco.setProperty(DiscoveryHelper.DNS_DISCOVER_METHOD
                                  ,DiscoveryHelper.USE_INPUT_DN_METHOD);
// Set the property for the SSLMODE
disco.setProperty(DiscoveryHelper.SSLMODE,"0");
```

これで、情報を検出できます。

```
// Call the discover method
disco.discover(reshdl);
```

検出された情報は、結果ハンドルで戻されます（前述の場合は、reshdl オブジェクトです）。結果は、結果ハンドルから抽出されます。

```
ArrayList result =
(ArrayList)reshdl.get(DiscoveryHelper.DIR_SERVERS);

if (result != null)
{
   if (result.size() == 0) return;
   System.out.println("The hostnames are :-");
   for (int i = 0; i< result.size();i++)
   {
      String host = (String)result.get(i);
System.out.println((i+1)+".'"+host+"'");
   }
}
```

DNS からのサーバー名およびポート番号の判断

DNS の検索からホスト名およびポート番号を判断するには、ドメインを取得した後、そのドメインに基づく SRV リソース・レコードを検索します。SRV リソース・レコードが複数存在する場合は、重み付けおよび優先順位に基づいて格納されます。SRV リソース・レコードには、接続に必要なホスト名およびポート番号が含まれます。この情報は、リソース・レコードから取得され、ユーザーに戻されます。

検索に必要なドメイン名の判断には、次の 3 つの方法があります。

- ネーミング・コンテキストの識別名（DN）のマッピング
- ローカル・マシンのドメイン・コンポーネントの使用
- DNS でのデフォルト SRV レコードの検索

ネーミング・コンテキストの識別名のマッピング

1 つ目は、ネーミング・コンテキストの識別名（DN）を、次のアルゴリズムを使用してドメイン名にマップする方法です。

最初は、出力されるドメイン名は空です。識別名は、右から順に処理されます。相対識別名（RDN）は、次の条件を満たす場合に変換できます。

- 単一の属性型および属性値で構成される。
- 属性型が DC である。
- 属性値が NULL 以外である。

RDN を変換できる場合、属性値がドメイン名コンポーネント（ラベル）として使用されます。

最初の値が、右側の最も重要なドメイン名コンポーネントになり、変換された後続の RDN 値が左側に追加されます。RDN を変換できない場合、処理は停止します。処理の停止時に出力されたドメイン名が空であった場合、DN はドメイン名に変換されません。

DN が cn=John Doe,ou=accounting,dc=example,dc=net の場合、クライアントによって、dc コンポーネントが DNS 名 example.net に変換されます。

ローカル・マシンのドメイン・コンポーネントによる検索

DN をドメイン名にマップできない場合もあります。たとえば、DN が o=Oracle IDC, Bangalore の場合、ドメイン名にはマップできません。この場合、2 つ目の方法として、クライアントが実行されているローカル・マシンのドメイン・コンポーネントを使用します。たとえば、クライアント・マシンのドメイン名が mc1.acme.com の場合、検索に使用するドメイン名は acme.com になります。

DNS でのデフォルト SRV レコードの検索

3 つ目は、DNS でデフォルト SRV レコードを検索する方法です。このレコードは、配置内のデフォルト・サーバーを指しています。このデフォルト・レコードのドメイン・コンポーネントは、_default です。

ドメイン名が判断されると、DNS への問合せの送信に使用されます。DNS に対して、Oracle Internet Directory 固有の形式で指定されている SRV レコードが問い合されます。たとえば、取得されたドメイン名が example.net の場合、SSL LDAP 以外のサーバーに対して、所有者名が _ldap._tcp._oid.example.net の SRV リソース・レコードが問い合されます。

DNS から SRV リソース・レコードが戻されない場合もあります。このような場合、DNS での検索は、標準形式で指定された SRV リソース・レコードに対して実行されます。たとえば、所有者名は _ldap._tcp.example.net となります。

関連項目：『Oracle Internet Directory 管理者ガイド』の第 5 章

問合せの結果は、SRV レコードのセットです。これらのレコードは格納され、レコードからホスト情報が抽出されます。抽出された情報はユーザーに戻されます。

注意： 前述の方法は、DNS の問合せ検索が成功して停止するまで、連続して試行することができます。これらの方法は、この項で説明した順序で試行されます。DNS に対して、Oracle Internet Directory 固有の形式の SRV レコードのみが問い合されます。前述のいずれの方法も失敗する場合、DNS に対して標準形式の SRV レコードのみの問合せを指定して、すべての方法が再度試行されます。

DNS サーバー検出の環境変数

デフォルトの DSD 動作を変更するために、次の環境変数が提供されています。

表 3-2 DSD 環境変数の動作

環境変数	説明
ORA_LDAP_DNS	SRV レコードを含む DNS サーバーの IP アドレス。この変数を定義しない場合、ホスト・マシンから DNS サーバー・アドレスが取得されます。
ORA_LDAP_DNSPORT	DNS サーバーが問合せをリスニングするポート番号。この変数を定義しない場合、DNS サーバーが標準のポート番号（53）でリスニングしていると判断されます。
ORA_LDAP_DOMAIN	ホスト・マシンのドメイン。この変数を定義しない場合、ホスト・マシン自身からドメインが取得されます。

DNS サーバー検出のプログラミング・インタフェース

ディレクトリ・サーバー情報の検出には、取得する際に使用するメカニズムや規格にかかわらず、単一のプログラミング・インタフェースが提供されます。情報は、様々なソースから検出される場合があります。それぞれのソース独自のメカニズムを使用して情報を検出できます。たとえば、LDAP ホストおよびポート番号情報は、ソースとして機能する DNS から検出される場合があります。この場合は、DNS からのホスト名の検出に DSD が使用されます。

関連項目： リファレンス情報およびクラスの説明については、プロダクト CD に格納されている Javadoc を参照してください。

サーバー検出の Java API

次の新しい Java クラス（パブリック・クラス）が導入されました。

```
public     class                  oracle.ldap.util.discovery.DiscoveryHelper
```

このクラスでは、指定されたソースから特定の情報を検出するメソッドが提供されます。

表 3-3 ディレクトリ・サーバー検出のメソッド

メソッド	説明
discover	指定されたソースから特定の情報を検出します。
setProperty	検出に必要なプロパティを設定します。
getProperty	プロパティの値にアクセスします。

既存の Java クラス oracle.ldap.util.jndi.ConnectionUtil に、次の 2 つの新しいメソッドが追加されます。

- getDefaultDirCtx: オーバーロードされたこのファンクションでは、oracle.ldap.util.discovery.DiscoveryHelper.discover() に対して内部コールを実行して、SSL 以外の LDAP サーバーのホスト名およびポート情報が判断されます。
- getSSLDirCtx: オーバーロードされたこのファンクションでは、oracle.ldap.util.discovery.DiscoveryHelper.discover() に対して内部コールを実行して、SSL の LDAP サーバーのホスト名およびポート情報が判断されます。

例 : ディレクトリ・サーバー検出の Java API

次に、ディレクトリ・サーバー検出に使用する Java プログラムの例を示します。

```
import java.util.*;
import java.lang.*;
import oracle.ldap.util.discovery.*;
import oracle.ldap.util.jndi.*;

public class dsdtest
{
  public static void main(String s[]) throws Exception
  {
    HashMap reshdl = new HashMap();
    String result = new String();
    Object resultObj = new Object();
    DiscoveryHelper disco = new
DiscoveryHelper(DiscoveryHelper.DNS_DISCOVER);
```

```
// Set the property for the DNS_DN
disco.setProperty(DiscoveryHelper.DNS_DN,"dc=us,dc=fiction,dc=com")
;

// Set the property for the DNS_DISCOVER_METHOD
disco.setProperty(DiscoveryHelper.DNS_DISCOVER_METHOD
                                ,DiscoveryHelper.USE_INPUT_DN_METHOD);

// Set the property for the SSLMODE
disco.setProperty(DiscoveryHelper.SSLMODE,"0");

// Call the discover method
int res=disco.discover(reshdl);
if (res!=0)
   System.out.println("Error Code returned by the discover method is :"+res) ;

// Print the results
printReshdl(reshdl);
}

public static void printReshdl(HashMap reshdl)
{
  ArrayList result = (ArrayList)reshdl.get(DiscoveryHelper.DIR_SERVERS);

if (result != null)
{
  if (result.size() == 0) return;
  System.out.println("The hostnames are :-");
  for (int i = 0; i< result.size();i++)
  {
      String host = (String)result.get(i);
      System.out.println((i+1)+".'"+host+"'");
   }
  }
 }
}
```

リソース情報管理機能

Oracle コンポーネントの中には、ユーザーの要求を実行するために、様々なリポジトリおよびサービスからデータを収集するものがあります。データを収集するために、これらのコンポーネントでは次の情報が必要です。

- データの収集元となるリソースのタイプを識別する情報。たとえば、Oracle データベースなどです。これは、リソース・タイプ情報と呼ばれます。
- リソースに対するユーザーの接続および認証に関する情報。これは、リソース・アクセス情報と呼ばれます。

この項では、次の項目について説明します。

リソース・タイプ情報

ユーザーの要求を実行するためにアプリケーションが使用するリソースの情報をリソース・タイプ情報と呼びます。リソース・タイプには、Oracle9*i* データベース・サーバーや交換可能な Java Database Connectivity データ・ソースなどがあります。リソース・タイプ情報には、ユーザーの認証に使用するクラス、ユーザー識別子、パスワードなどの項目が含まれます。

Oracle Internet Directory セルフ・サービス・コンソールを使用して、リソース・タイプ情報を指定します。

リソース・アクセス情報

データベースに対するユーザーの接続および認証に関する情報を、リソース・アクセス情報と呼びます。この情報は、様々な Oracle コンポーネントで取得および共有できるリソース・アクセス記述子（RAD）と呼ばれるエントリに格納されます。

たとえば、販売レポートに関するユーザーの要求を実行するために、Oracle Application Server Reports Services は複数のデータベースを問い合せます。データベースへの問合せでは、次の処理が実行されます。

1. RAD からの必要な接続情報の取得
2. 取得した情報を使用した、データベースへの接続およびデータを要求しているユーザーの認証

この処理が終了すると、レポートがコンパイルされます。

Oracle Internet Directory セルフ・サービス・コンソールを使用して、リソース・アクセス情報を指定します。各ユーザーに対してリソース・アクセス情報を個別に指定したり、すべてのユーザーに対して共通に指定することもできます。後者の場合、指定されたアプリケーションに接続するすべてのユーザーは、デフォルトで同じ情報を使用して必要なデータベースに接続します。たとえば、各ユーザーが一意のシングル・サインオン・ユーザー名でアプリケーション内に定義されている場合など、アプリケーションに独自の統合アカウント管理がある場合は、デフォルトのリソース・アクセス情報を定義することをお薦めします。

DIT 内のリソース情報の位置

図 3-4 に、DIT 内のリソース情報の位置を示します。

図 3-4 DIT 内のリソース・アクセス情報およびリソース・タイプ情報の配置

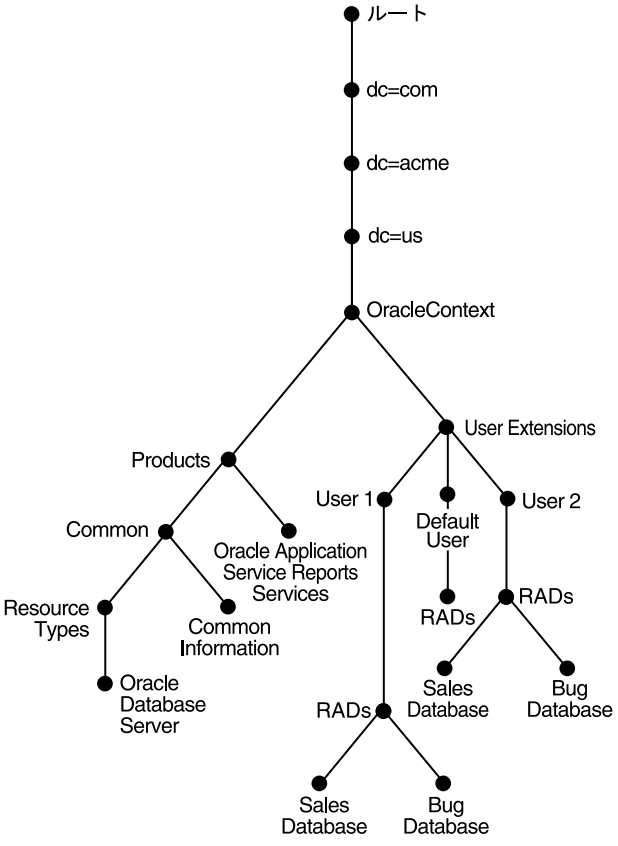


図 3-4 に示すとおり、リソース・アクセス情報およびリソース・タイプ情報は、Oracle コンテキストに格納されます。

各ユーザーのリソース・アクセス情報は、Oracle コンテキスト内の cn=User Extensions ノードに格納されます。この例では、cn=User Extensions ノードには、デフォルトのユーザーおよび特定のユーザーの両方のリソース・アクセス情報が含まれています。後者の場合、リソース・アクセス情報には、Sales データベースおよび Bug データベースの両方へのアクセスで必要な情報が含まれます。

各アプリケーションのリソース・アクセス情報は、アプリケーション名で識別されるオブジェクトに格納されます。たとえば、cn=Oracle Application Server Reports Services, cn=Products,cn=Oracle Context,dc=us,dc=acme,dc=com などです。これは、その製品に固有のユーザー情報です。

リソース・タイプ情報は、コンテナ cn=resource types、cn=common、cn=products、cn=Oracle Context に格納されます。

関連項目：

- RAD の設定および配置の詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。
- 『Oracle Application Server Reports Services レポート Web 公開ガイド』
- 『Oracle Application Server Forms Services 利用ガイド』
- 9-17 ページの「get_user_extended_properties ファンクション」
- oracle.ldap.util.get_extended_properties、oracle.ldap.util.set_extended_properties、および oracle.ldap.util.create_extended_properties については、『Oracle Internet Directory API Reference』を参照してください。

SASL 認証機能

Oracle Internet Directory では、SASL ベース認証の 2 つのメカニズムをサポートします。この項では、それらのメカニズムについて説明します。次の項目について説明します。

Digest-MD5 メカニズムを使用した SASL 認証

SASL Digest-MD5 認証は、LDAP バージョン 3 サーバー（RFC 2829）で必要な認証メカニズムです。LDAP バージョン 2 では、Digest-MD5 はサポートされません。

Digest-MD5 メカニズムについては、Internet Engineering Task Force の RFC 2831 を参照してください。このメカニズムは、HTTP Digest 認証（RFC 2617）に基づいています。

> **関連項目：** RFCs 2829、2831 および 2617 については、Internet Engineering Task Force の Web サイト（http://www.ietf.org）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。

- Digest-MD5 を使用した SASL 認証に含まれる手順
- Digest-MD5 を使用した SASL 認証の JAVA API
- Digest-MD5 を使用した SASL 認証の C API
- 外部メカニズムを使用した SASL 認証

Digest-MD5 を使用した SASL 認証に含まれる手順

SASL Digest-MD5 は、次のようにユーザーを認証します。

1. ディレクトリ・サーバーは、サポートする各種認証オプションと特別なトークンを含むデータを LDAP クライアントに送信します。
2. クライアントは、選択した認証オプションを示す暗号化された応答を送信して応答します。応答は、クライアントがそのパスワードを知ることがないように暗号化されます。
3. ディレクトリ・サーバーは、クライアントの応答を復号化し、検証します。

Digest-MD5 認証メカニズムを使用するには、Java API または C API のいずれかを使用して認証を設定できます。

Digest-MD5 を使用した SASL 認証の JAVA API

Context.SECURITY_AUTHENTICATION に「DIGEST-MD5」を設定します。

Context.SECURITY_PRINCIPAL には、プリンシパル名を設定します。

プリンシパル名は、サーバー固有の形式になります。次のいずれかのようになります。

- 認証されているエンティティの完全に修飾された識別名が続く識別名 `dn:`
- ユーザー識別子が続く文字列 `u:`

Oracle ディレクトリ・サーバーは、完全に修飾された識別名（`cn=user,ou=my department,o=my company` など）のみを受け入れます。

注意： SASL 識別名は、SASL バインドをコールする C または Java API に渡される前に正規化される必要があります。SASL ベリファイアを生成するために、Oracle Internet Directory では正規化された識別名のみがサポートされます。

関連項目： B-39 ページの「JNDI のサンプル・コード」

Digest-MD5 を使用した SASL 認証の C API

LDAP クライアントは、提供された C API を使用して、SASL Digest-MD5 を設定し、ディレクトリ・サーバーに接続することができます。

関連項目：

- 7-17 ページの「ディレクトリに対する認証」
- 7-63 ページの「SASL ベースの Digest-MD5 認証での C API の使用方法」

外部メカニズムを使用した SASL 認証

次の文は、Internet Engineering Task Force の RFC 2222 のセクション 7.4 を翻訳したものです。

> 外部認証に関連付けられたメカニズム名は、EXTERNAL です。クライアントは、認可識別情報が含まれた初期応答を送信します。サーバーは SASL の外部にある情報を使用して、クライアントが認可識別情報として認可されるかどうかを判断します。クライアントがこのようにして認可された場合、サーバーは認証通信が成功して完了したことを示します。そうでない場合、サーバーは失敗を示します。
>
> IPsec や SSL/TLS などのシステムによりこの外部情報が提供されます。
>
> クライアントが認可識別情報として空の文字列を送信した（クライアントの認証資格証明から作成された認可識別情報を要求する）場合、認可識別情報は外部認証を提供するシステムの認証資格証明から作成されます。

Oracle Internet Directory は、SSL 相互接続を介して SASL 外部メカニズムを提供します。認可識別情報（識別名）は、SSL ネットワーク協定時のクライアント証明書から作成されます。

PL/SQ LDAP API の依存性と制限事項

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 Application Server 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。

4

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 and Provisioning Platform のコンポーネントである Oracle Directory Provisioning Integration Service を使用でき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。

関連項目：『Oracle Internet Directory 管理者ガイド』の第 34 章を参照してください。

Oracle Directory Provisioning Integration Service の概要

ディレクトリ管理での大きな課題として、ユーザーごとに必要な多数のアカウントやアプリケーションのためのプロビジョニング情報の管理があります。たとえば、通常、ユーザーを情報システムに追加すると、大量のアプリケーション・プロビジョニングが必要になります。この場合、電子メール・アカウントを設定でき、そのメール・アカウントには、メール割当て制限、デフォルト・フォルダ、配布リストなど、固有の情報が設定されます。ユーザーがその他の接続アプリケーションを必要とする場合、ユーザー・アカウントおよび個人のプロファイルの管理は、大企業では非常に煩雑です。この問題を解決するために、Oracle Directory Provisioning Integration Service はアプリケーション統合のためのプラットフォームを提供します。このプラットフォームによって、多くの重要なシステムにワンステップでシームレスにユーザーを追加できます。

Oracle Directory Provisioning Integration Service は、ユーザー・アカウント情報のパススルーとして機能します。ユーザーを個々のアプリケーションにプロビジョニングするのではなく、単にアプリケーションをプロビジョニング・サービスに登録します。これによって、プロビジョニング情報を Oracle Internet Directory との間で直接送受信することができます。その後、ユーザーを、統合されたアプリケーションのデフォルト・セットに一度にプロビジョニングできます。このようにして、Oracle Directory Provisioning Integration Service は個々のアプリケーションに対する冗長な処理を削除します。

インストール中に定義されたプロビジョニング・イベントのデフォルト・セットに加え、Oracle Internet Directory は、新しいイベントを定義したり、定義したイベントを、それらのイベントにサブスクライブするアプリケーションへ適切に伝播させることができます。これらのプロビジョニング・イベントを送受信する機能によって、ユーザー・アカウントのシームレスな管理が実現します。

プロビジョニング統合アプリケーションの開発

Oracle Directory Provisioning Integration Service に統合された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。アプリケーションは Oracle Internet Directory に登録された後、Oracle Internet Directory との間でプロビジョニング情報を送受信できます。

アプリケーションをディレクトリ・プロビジョニング・サービスに統合するには、次の一般的な手順に従います。各手順については、この章の後半で説明します。

- アプリケーションを Oracle Internet Directory に登録します。
- イベントが伝播または適用される認証管理レルムを識別します。
- アプリケーションでイベントの受信または送信、あるいはその両方を実行する必要があるかどうかを判断します。
- 送信または受信の必要があるイベントをリストします。
- イベントに含める必要がある関連属性をリストします。

- 様々なイベントを Oracle Internet Directory から読取りおよび伝播可能にし、変更イベントを Oracle Internet Directory に適用するために、認証管理レルムにあるアプリケーションの識別情報に適切な権限を割り当てます。
- インタフェース名、インタフェース型、インタフェース接続情報を決定します。これは、プロビジョニング・サーバーがアプリケーションにイベントを伝播し、アプリケーションからイベントをコンシュームするために必要です。
- その他のプロビジョニングのスケジューリング間隔、スケジュールごとの最大イベント数などを決定します。
- アプリケーション内部のインタフェース仕様を実装します。
- イベント伝播を開始できるように Oracle Internet Directory にプロビジョニング・プロファイルを作成します。このプロファイルは、プロビジョニングのサブスクリプション・ツール（oidprovtool）を使用して作成します。

これらの一般的な手順についてわかりやすく説明するために、サンプル・アプリケーションを示します。

プロビジョニング統合アプリケーションの例

このプロビジョニング統合アプリケーションの例を、Employee Self Service Application（ESSA）と呼びます。ここでは、「ユーザー」および「識別情報」という用語は、同じ意味で使用します。

Employee Self Service Application の要件

このアプリケーションでは、すべてのユーザーのベースが Oracle Internet Directory から管理される必要があります。アプリケーション管理者は、Oracle Internet Directory で識別情報を作成、変更および削除します。識別情報は、イベント `IDENTITY_ADD` としてアプリケーションに伝播されます。

アプリケーションはユーザー・データとして識別情報を作成しますが、これだけでは、従業員はアプリケーションへのアクセスを許可されません。Oracle Internet Directory に識別情報が存在することによって、グローバル・ログインが簡素化されるのみです。アプリケーションは、特定の識別情報がアプリケーションへのアクセスを許可されているかどうかを識別する必要があります。これは、そのアプリケーションに識別情報をサブスクライブすることによって識別されます。このタスクは、アプリケーション管理者が実行できます。このサブスクライブによって、そのアプリケーションを使用するために、識別情報が Oracle Internet Directory でサブスクライブされたことを示す別のイベント（`SUBSCRIPTION_ADD`）が、Oracle Internet Directory からアプリケーションへトリガーされます。アプリケーションは、ユーザーにアプリケーションへのアクセスを許可する前に、そのユーザーがアプリケーションのサブスクリプション・リストに存在するかどうかをチェックするためにディレクトリを問い合せることができます。

この例では、このアプリケーションに対するイベントは、Oracle Internet Directory から受信されています。アプリケーション自身は、ディレクトリにイベントを送信していません。ただし、Oracle Internet Directory にイベントを送信することは可能です。イベントを送信するには、ディレクトリで実行する様々な操作のための別のディレクトリ権限がアプリケーション識別情報に必要です。詳細は、4-6 ページの「Employee Self Service Application のプロビジョニング・モードの決定」を参照してください。

手順は、次のとおりです。

1. Oracle Internet Directory セルフ・サービス・コンソール、またはその他の方法（サード・サーティのソースからの同期化またはコマンドライン・ツールの使用）を介して、Oracle Internet Directory にユーザーを追加します。ユーザー情報は、適切な認証管理レルムに格納される必要があります。
2. `IDENTITY_ADD` イベントは、Oracle Internet Directory からアプリケーションに伝播されます。ここでは、プロビジョニング・サブスクリプション・プロファイルの作成中に、アプリケーションが `IDENTITY_ADD` イベントに伝播されることを前提としています。
3. イベントを受信すると、アプリケーションはこの識別情報を自身のデータベースに追加します。ただし、この例では、ユーザーがアプリケーションへのアクセスを許可されたことを意味しません。そのアプリケーションの認可ユーザーとしてサブクスライブするためには、追加イベントが必要です。
4. Oracle Internet Directory では、Oracle Delegated Administration Services を使用してユーザーがアプリケーションにサブスクライブされます。
5. `SUBSCRIPTION_ADD` イベントは、Oracle Internet Directory からアプリケーションに伝播されます。ここでは、プロビジョニング・サブスクリプション・プロファイルの作成中に、アプリケーションが `SUBSCRIPTION_ADD` イベントに伝播されることを前提としています。
6. このイベントを受信すると、アプリケーションは、自身のデータベース内でユーザーが認可ユーザーでもあることを示すように、識別情報のレコードを更新できます。

Oracle Internet Directory での Employee Self Service Application の登録

アプリケーションは、自身をアプリケーション・エンティティとして、Oracle Internet Directory 内の独自の識別情報エントリに登録する必要があります。レルムが DIT 内の既知の位置に作成されている場合、どのレルムにアプリケーション識別情報を作成するかを決定できます。Oracle Internet Directory に必要な DIT 要素を作成する場合、この章で説明するテンプレートに従って作成する必要があります。

認証管理レルムの Oracle コンテキストには、次に示す、様々なアプリケーションのフットプリント用のコンテナが含まれます。

```
cn=products,cn=oraclecontext,identity management realm DN
```

アプリケーションが単一のレルムである場合、アプリケーションの識別情報 DN は、次の形式で記述することをお薦めします。

```
orclApplicationName=application name,cn=application
type,cn=products,cn=oraclecontext,identity management realm DN
```

cn=*application type* 要素は、アプリケーション・コンテナと呼ばれます。

アプリケーションが複数のレルムである場合、アプリケーション識別情報をルート Oracle コンテキスト cn=products,cn=oraclecontext に作成できます。

この例では、エントリの位置および内容は次のとおりです。

```
dn: ¥
orclApplicationCommonName=ESSA,cn=demoApps,cn=Products,cn=OracleContext,o=ACME,
dc=com
orclapplicationcommonname: ESSA
orclappfullname: Employee Self Service Application
userpassword: welcome123
description: This is an sample application for demonstration.
orclaci: access to entry by group="cn=odisgroup,cn=odi,cn=oracle internet direct
ory" (proxy)
objectclass: orclApplicationEntity
```

この例では、アプリケーション・タイプまたはアプリケーション・コンテナは demoApps です。このアプリケーション名は ESSA です。

すべてのディレクトリ操作が、アプリケーションのかわりにプロビジョニング・サーバーによって実行される必要があります。サーバーは、ドメイン下でイベントを送信およびコンシュームするために必要な権限を持っていないため、アプリケーションの識別情報の代理としてイベントを処理する必要があります。つまり、サーバーにプロキシ権限を付与する必要があります。この例では、アプリケーションの識別情報に必要な権限がすでに付与されていることを前提とします。

Employee Self Service Application の管理コンテキストの識別

通常、すべての認証管理レルムは、ルート Oracle コンテキストの認証管理レルムのベース下に存在します。アプリケーションは、適切なレルムにプロビジョニングする必要があります。つまり、アプリケーションがこのレルム下で自身の情報を管理できるように、アプリケーション識別情報に適切な権限が割り当てられる必要があります。この例では、適切なレルムを o=ACME,dc=com としています。

Employee Self Service Application のプロビジョニング・モードの決定

アプリケーションがイベントの受信のみを行うか、または Oracle Internet Directory へのイベントの送信も行うかどうかを決定する必要があります。モードは次のとおりです。

- INBOUND: アプリケーションから Oracle Internet Directory へ
- OUTBOUND: Oracle Internet Directory からアプリケーションへ（デフォルト）
- BOTH

デフォルト・モードは OUTBOUND です。

この例では、アプリケーションは Oracle Internet Directory からのイベントの受信のみを対象としているため、イベントを OUTBOUND のみに指定しています。

Employee Self Service Application のイベントの決定

インストール中に、特定のイベントが事前に定義されます。実行時に新しくイベントを定義できますが、それらのイベントは OUTBOUND モードの場合のみ伝播できます。Oracle Directory Provisioning Integration Service は、事前定義された特定のイベントのみ、INBOUND モードで処理できます。

この例では、新しいイベントを定義する必要はありません。次に示す Oracle Internet Directory のイベントをサンプル・アプリケーションに伝播する必要があります。

- 識別情報の作成（IDENTITY_ADD）
- 識別情報の変更（IDENTITY_MODIFY）
- 識別情報の従業員の削除（IDENTITY_DELETE）
- 識別情報のサブスクリプションの追加（SUBSCRIPTION_ADD）
- 惜別情報のサブスクリプションの変更（SUBSCRIPTION_MODIFY）
- 惜別情報のサブスクリプションの削除（SUBSCRIPTION_DELETE）

認証管理レルムに対する Employee Self Service Application のプロビジョニング

この手順は、最も重要な手順です。認証管理レルムにあるアプリケーションの識別情報に適切な権限を割り当てる手順が含まれます。これらの権限によって、アプリケーションで様々なイベントを Oracle Internet Directory から読取りおよび適用でき、また変更イベントを Oracle Internet Directory に送信できます。INBOUND イベントでは、Oracle Internet Directory を変更することになるため、別の権限が必要です。

通常、認証管理レルムの作成時に事前定義グループが作成されます。この項で説明するとおり、このグループは異なる権限を持ちます。

次のテンプレートには、アプリケーションがプロビジョニング・イベントを送信または受信するために必要な適切な ACL がすべて記述されています。

アプリケーションの識別情報は、必要な権限によって、適切なグループに追加される必要があります。たとえば、アプリケーションが Oracle Internet Directory からのイベントの受信のみを対象とする場合、このレルムでエントリを作成または変更できるグループに追加する必要はありません。

テンプレートの一部には変数を使用します。変数がインスタンス化されると、テンプレートは Oracle Internet Directory に対して実行可能な正規の LDIF ファイルになります。変数は、配置要件に応じて調整できます。

この例では、認証管理レルムは o=ACME,dc=com です。LDIF ファイルのテンプレートは、次のようになります。

```
# This creates The Application Identity subtree
#
# The following variables are used :
# (Some of them are OPTIONAL where the values oidprov tool can get  default
# values if not supplied.)
#
#  %s_IdentityRealm% : Identity Realm DN:
#                      (MANDATORY: This is the domain in which all the related users
and groups are present.
#                                  If Default Identity Realm needs to be used then
in an OID install it can be queried.
#                                  This value is stored in Root Oracle Context in
OID. This value is stored in
#                                  'orcldefaultsubscriber' attribute in
#                                  'dn: cn=Common,cn=Products,cn=OracleContext'
entry.)
#  %s_AppType% : Application Type (e.g EBusiness)
#                       (MANDATORY : Name of the suite  )
#  %s_AppName% : Application Name (e.g HRMS,Financials,Manufacturing)
#                       (MANDATORY: Name of the Application in the suite.)
#  %s_SvcType% : Service Type (e.g Ebusiness)
#                       (MANDATORY : Alias for name of suite.
#                                    This value can be be same as %s_AppType%)
#  %s_SvcName% : Service Name (e.g HRMS,Financials,Manufacturing)
#                       (MANDATORY : Alias for name of Application.
#                                    This value can be same as %s_AppName%)
#  %s_AppURL% : Application URL if any. (set it to 'NULL' if there is nothing.)
#
#  Apart from these variables this LDIF templates would also need the following
information to load this
# data to Oracle Internet Directory:
#
#  LDAP_HOST :  OID server hostname
```

```
#  LDAP_PORT :  OID server port number
#  BINDDN    :  cn=orcladmin
#  BINDPASSWD:  Password for orcladmin
#
#  After replacing the variables in the template this data can be loaded in OID by
running the following
# command:
#  ldapmodify -h %LDAP_HOST% -p %LDAP_PORT% -D %BINDDN% ¥
#                  -w %BINDPWD% -f <this_template_file_name>
#
#

# First we create the Application container. This needs to be created just once
#initially. If this container is
# existing  b'cos some application was already created using this template, #please
remove this entry from the template/LDIF file.

dn: cn=%s_AppType%,cn=Products,cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: %s_AppType%
objectclass: orclContainer

#  The application identity needs to created next. This is under the Applications
container. This object is of # type "orclApplicationEntity"

dn:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%
changetype: add
orclapplicationcommonname: %s_AppName%
orclaci: access to entry by group="cn=odisgroup,cn=odi,cn=oracle internet directory"
 (add,browse,delete,proxy)
objectclass: orclApplicationEntity



# The following ACLs are for giving privileges to the application entities for
adding/modifying/deleting
#  users in the relevant realm.

# All members of the group below are allowed to create users in the relevant realm.

dn: cn=OracleDASCreateUser,cn=Groups,cn=OracleContext,%s_IdentityRealm%
changetype: modify
add: uniquemember
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
```

```
 %s_IdentityRealm%

# All members of the group below are allowed to delete users in the relevant realm.

dn: cn=OracleDASDeleteUser,cn=Groups,cn=OracleContext,%s_IdentityRealm%
changetype: modify
add: uniquemember
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%

# All members of the group below are allowed to edit users in the relevant realm.

dn: cn=OracleDASEditUser,cn=Groups,cn=OracleContext,%s_IdentityRealm%
changetype: modify
add: uniquemember
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%

# All members of the group below are allowed to create groups in the relevant realm.

dn: cn=OracleDASCreateGroup,cn=Groups,cn=OracleContext,%s_IdentityRealm%
changetype: modify
add: uniquemember
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%



# All members of the group below are allowed to delete groups in the relevant realm.

dn: cn=OracleDASDeleteGroup,cn=Groups,cn=OracleContext,%s_IdentityRealm%
changetype: modify
add: uniquemember
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%

# All members of the group below are allowed to edit groups in the relevant realm.

dn: cn=OracleDASEditGroup,cn=Groups,cn=OracleContext,%s_IdentityRealm%
changetype: modify
add: uniquemember
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
```

```
 %s_IdentityRealm%

# The container is being created to hold the various subscription lists of the
application
# for this realm. This container will hold lots of subscription information and
resides just # under the application identity.

dn:
cn=subscriptions,orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,
 cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: subscriptions
objectclass: orclContainer

# The following is the group that will hold administrators DNs for managing
# subscription lists for this application. The application identity should also be
in this list and # will be added here.

dn: cn=Subscription_Admins,cn=Subscriptions,orclApplicationCommonName=%s_AppName%,
 cn=%s_AppType%,cn=products,cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: Subscription_Admins
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%
objectclass: groupOfUniqueNames
objectclass: orclACPGroup
objectclass: orclprivilegegroup



# The following is the group that will hold  DNs of users who can just view the
# subscription lists for this application. The application identity should also be
in this list and # will be added here.

dn: cn=Subscription_Viewers,cn=Subscriptions,orclApplicationCommonName=%s_AppName%,
 cn=%s_AppType%,cn=products,cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: Subscription_Viewers
uniquemember:
orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%
objectclass: groupOfUniqueNames
objectclass: orclACPGroup
objectclass: orclprivilegegroup

# The following is just a container for the actual subscription lists.
```

```
dn: cn=subscription_data,cn=subscriptions,orclApplicationCommonName=%s_AppName%,
 cn=%s_AppType%,cn=Products,cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: subscription_data
objectclass: orclContainer

# The following is a sample subscription list. We are calling it "cn=ACCOUNTS" since
it # signifies accounts in the application.

dn:
cn=ACCOUNTS,cn=subscription_data,cn=subscriptions,orclApplicationCommonName=%s_AppNa
me%,cn=%s_AppType%,cn=Products,cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: cn=ACCOUNTS
uniquemember: cn=orcladmin
objectclass: groupOfUniqueNames
objectclass: orclGroup

# The following is a container for the service instance entries in the Root Oracle
Context. An application
# publishes itself as a service by creating a service instance  entriy under this
container.  These service
# instance entries are created outside any realm and in the root #Oracle Context.

dn: cn=%s_SvcType%,cn=Services,cn=OracleContext
changetype: add
cn: %s_SvcType%
objectclass: orclContainer



# The following is a container for the service instance entries in the Root Oracle
Context for that service
# type

dn: cn=ServiceInstances,cn=%s_SvcType%,cn=Services,cn=OracleContext
changetype: add
cn: ServiceInstances
objectclass: orclContainer

# The following is a service instance entry. An application publishes itself as a
service by
# creating this service instance

dn: cn=%s_SvcName,cn=ServiceInstances,%,cn=%s_SvcType%,cn=Services,cn=OracleContext
changetype: add
```

```
cn: %s_SvcName%
orclServiceType: %s_SvcType%
presentationAddress: %s_AppURL%
objectclass: orclServiceInstance

# The following is a container for service instance reference entry that resides in
the relevant realm.

dn: cn=%s_SvcType%,cn=Services,cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: %s_SvcType%
objectclass: orclContainer

# It is a  reference entry which actually points to the actual service instance
entry as well as to the
# subscription list container  for the application.

dn: cn=%s_SvcName%,cn=%s_SvcType%,cn=Services,cn=OracleContext,%s_IdentityRealm%
changetype: add
cn: %s_SvcName%
description: Link To the Actual Subscription Location for the Application and the
actual Service instance.
orclServiceInstanceLocation:
cn=%s_SvcName%,cn=%s_SvcType%,cn=Services,cn=OracleContext
orclServiceSubscriptionLocation: cn=subscription_data,cn=subscriptions,
 orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,cn=OracleContext,
 %s_IdentityRealm%
objectclass: orclServiceInstanceReference


# This LDIF operation gives appropriate privileges to the subscription admin and
subscription viewers
# group. The groups have already been created earlier.

dn:
cn=subscriptions,orclApplicationCommonName=%s_AppName%,cn=%s_AppType%,cn=Products,
 cn=OracleContext,%s_IdentityRealm%
changetype: modify
replace: orclaci
orclaci: access to entry by
group="cn=Subscription_Admins,cn=Subscriptions,orclApplicationCommonName=%s_AppName%
,
 cn=%s_AppType%,cn=products,cn=OracleContext,%s_IdentityRealm%" (browse,add,delete)
by
group="cn=Subscription_Viewers,cn=Subscriptions,orclApplicationCommonName=%s_AppName
%,
 cn=%s_AppType%,cn=products,cn=OracleContext,%s_IdentityRealm%" (browse)
```

```
orclaci: access to attr=(*) by
group="cn=Subscription_Admins,cn=Subscriptions,orclApplicationCommonName=%s_AppName%
,
 cn=%s_AppType%,cn=products,cn=OracleContext,%s_IdentityRealm%"
(search,read,write,compare) by
group="cn=Subscription_Viewers,cn=Subscriptions,orclApplicationCommonName=%s_AppName
%,
 cn=%s_AppType%,cn=products,cn=OracleContext,%s_IdentityRealm%"
(search,read,compare)
```

Employee Self Service Application のスケジューリング・パラメータの決定

スケジューリング間隔によって、プロビジョニング・サーバーがイベントを送信または受信する頻度が決定します。サーバーは、イベントを（イベントがなくなるまで）送信または受信し、それらがすべて完了した後、スケジューリング間隔に指定した時間（秒単位）を停止します。一度に送信または受信可能なイベント数は、スケジュールごとの最大イベント数のパラメータで指定します。

2 分ごとに、毎回、最大で 100 のイベントを伝播する必要がある場合を考えてみます。

Employee Self Service Application のインタフェース接続情報の決定

インタフェース接続情報を決定するには、次の情報を使用します。

- インタフェース型 : これはイベント伝播の手段です。現在は、PL/SQL のみサポートされています。
- インタフェース名 : これは、アプリケーションが実装する必要があり、プロビジョニング・サーバーがイベントを送信および受信するために起動する PL/SQL パッケージ名です。このサンプル・アプリケーションでは、インタフェース名を ESSA_INTF としています。
- インタフェース接続情報 : これは、PL/SQL インタフェースを起動するために、サーバーがアプリケーション・データベースへの接続に使用します。

接続情報は、次の形式で指定します。

```
Database Host: Listener Port: Database SID: DB Account: Password
```

RAC 対応の高可用性データベースでは、接続情報は次の形式で指定します。

```
Database Host: Listener Port: Service Name: DB Account: Password; Database Host:
Listener Port: Service Name: DB Account: Password; Database Host: Listener Port:
Service Name: DB Account: Password
```

文字列全体を、1 行に単一の値として指定する必要があります。

このサンプル・アプリケーションでは、接続情報は次のようになります。

```
localhost: 1521: iasdb : scott : tiger
```

Oracle Directory Integration Server は、指定した接続情報を使用してアプリケーション・データベースに接続するために JDBC を使用し、次にイベントを伝播または受信するために PL/SQL API を起動します。

Employee Self Service Application のインタフェース仕様の実装

インタフェースの詳細は、第 11 章「プロビジョニング統合 API リファレンス」を参照してください。

OUTBOUND イベント（Oracle Internet Directory からアプリケーションへのイベント）では、次のインタフェースを実装する必要があります。

```
PROCEDURE PutOIDEvent (event         IN  LDAP_EVENT,
                       event_status  OUT LDAP_EVENT_STATUS);
```

INBOUND イベント（アプリケーションから Oracle Internet Directory へのイベント）では、次のインタフェースを実装する必要があります。

```
-- FUNCTION GetAppEvent(event OUT LDAP_EVENT)  RETURNING NUMBER;
-- PROCEDURE PutAppEventStatus(event_status IN LDAP_EVENT_STATUS)
```

ここでは、OUTBOUND イベントのみを処理しているため、OUTBOUND イベントに関連するすべてのインタフェースを実装します。

Employee Self Service Application のプロビジョニング・サブスクリプション・プロファイルの作成

プロビジョニング・サブスクリプション・プロファイルを作成するには、次の設定を使用します。

```
$ORACLE_HOME/bin/oidprovtool operation=create ldap_host=localhost ¥
ldap_port=389 ldap_user_dn=cn=orcladmin ldap_user_password=welcome ¥
organization_dn="o=ACME,dc=com" ¥
application_dn="orclApplicationCommonName=ESSA,cn=demoApps,cn=Products,¥
cn=OracleContext,o=ACME,dc=com" ¥
interface_name=ESSA_INTF  interface_type=PLSQL ¥
interface_connect_info="localhost:1521:iasdb:scott:tiger" ¥
event_subscription="IDENTITY:o=oracle,dc=com:ADD(cn,sn,mail,description,
telephonenumber)" ¥
event_subscription="IDENTITY:o=oracle,dc=com:MODIFY(cn,sn,mail,description,telephone
number)" ¥
event_subscription="IDENTITY:o=oracle,dc=com:DELETE " ¥
event_subscription="SUBSCRIPTION:cn=ESSA,cn=prducts,cn=oraclecontext,o=oracle,
dc=com:ADD(orclactivestartdate,orclactiveenddate,cn) ¥
event_subscription="SUBSCRIPTION:cn=ESSA,cn=prducts,cn=oraclecontext,o=oracle,dc=com
```

```
:MODIFY(orclactivestartdate,orclactiveenddate,cn) ¥
event_subscription="SUBSCRIPTION:cn=ESSA,cn=prducts,cn=oraclecontext,o=oracle,dc=com
:
DELETE"
```

プロビジョニング統合の前提条件

Oracle Directory Provisioning Integration Service で使用するアプリケーションは、Oracle RDBMS ベースで Oracle Application Server Single Sign-On 対応である必要があります。

アプリケーション開発者には、次の知識が必要です。

- 一般的な LDAP の概念
- Oracle Internet Directory
- Oracle Internet Directory と Oracle Application Server の統合
- Oracle Delegated Administration Services
- Oracle Application Server ドキュメントの『Oracle Internet Directory 管理者ガイド』の第 34 章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 and Provisioning Platform
- SQL、PL/SQL およびデータベース RPC に関する知識

さらに、Oracle Application Server Single Sign-On の概念を理解しておくことをお薦めします。

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。次の項目について説明します。

- プロビジョニング統合の開始
- プロビジョニング情報をディレクトリに戻す

プロビジョニング統合の開始

アプリケーションのインストール時に、次の情報が Oracle Directory 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 Directory 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）です。図 4-1 に、プロビジョニングの第 1 段階を示します。ここでは、Oracle Internet Directory から Oracle Directory Integration and Provisioning Platform プロビジョニング・フィルタを介してアプリケーションにユーザー・イベントを渡します。

図 4-1 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 Provisioning Integration Service を使用）

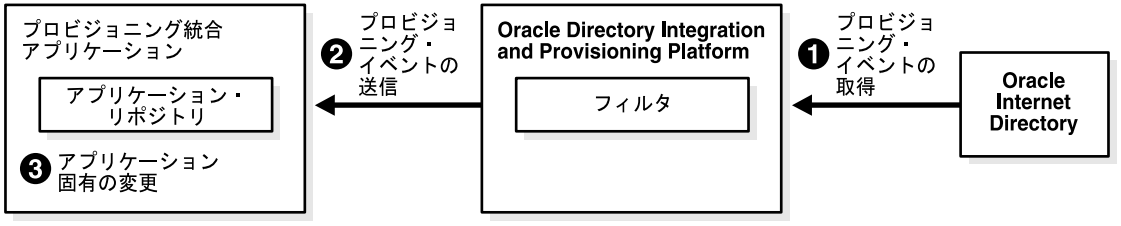


図 4-1 の内容は次のとおりです。

1. Oracle Directory Provisioning Integration Service は、ユーザーおよび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。
2. Oracle Directory 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。
3. 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。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。

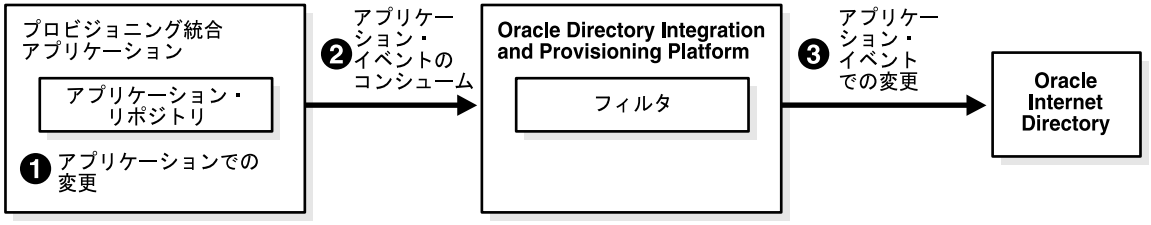
プロビジョニング情報をディレクトリに戻す

ここでは、プロビジョニング情報を Oracle Internet Directory に戻すことができます。図 4-2 に、この処理に含まれる手順を示します。基本的にはプロビジョニング処理の逆です。

1. アプリケーション・リポジトリがアプリケーションのイベント・データを生成し、そのデータを Oracle Directory Integration and Provisioning Platform に送信します。
2. Oracle Directory Integration and Provisioning Platform は、イベント・データを選別し、変更情報をディレクトリ・サーバーに戻します。
3. 変更が Oracle Internet Directory に適用されます。

更新された情報は、Oracle Internet Directory に格納され、他のアプリケーションからアクセスできるようになります。

図 4-2 アプリケーションがプロビジョニング情報を Oracle Internet Directory Provisioning Service に戻す方法



4-18 ページの図 4-3 に、プロビジョニング統合配置での、サービスおよびサブスクライブされたアプリケーションの関係を示します。

図 4-3 一般的な配置での、プロビジョニング・サービスとサブスクライブされたアプリケーション

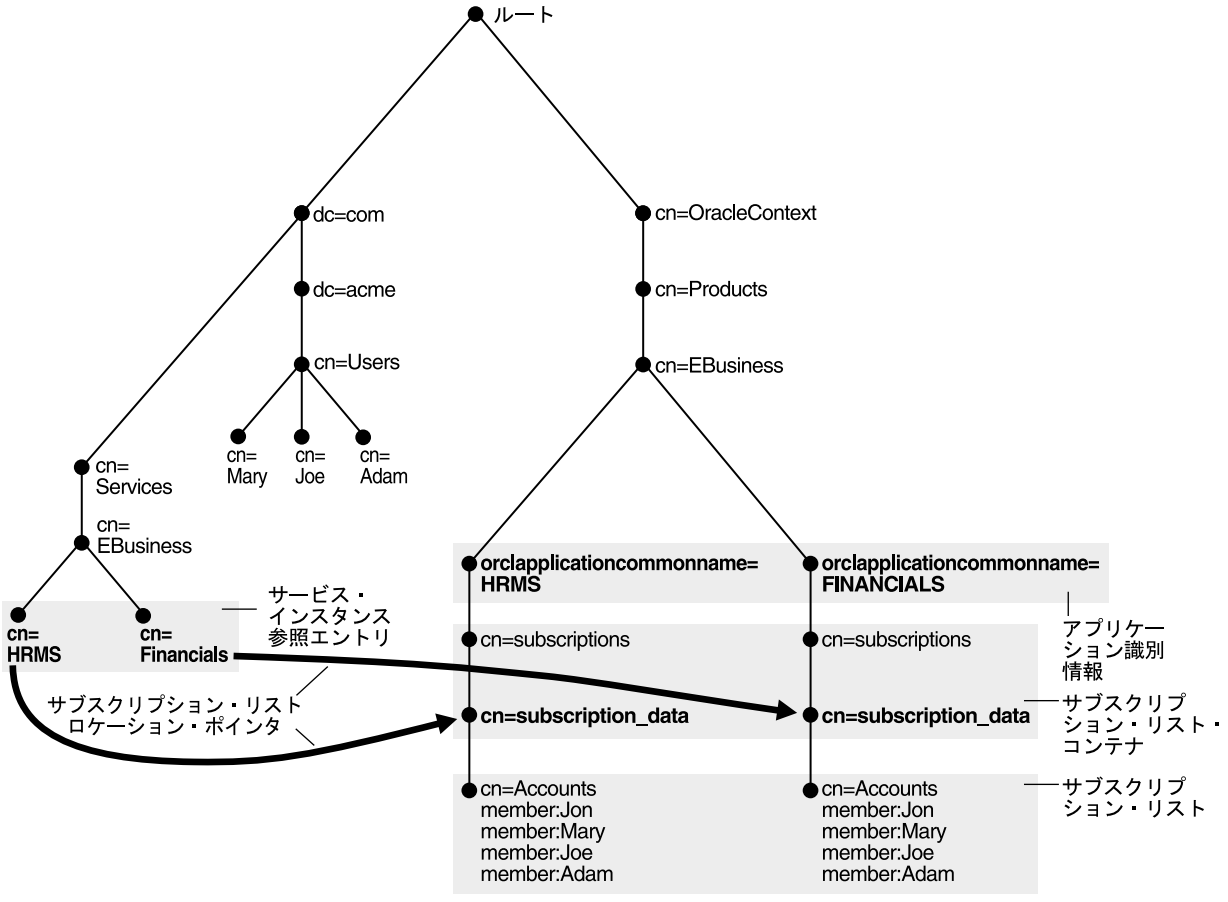


図 4-3 に、2 つのサービス（Oracle Human Resources および Oracle Financials）のエントリが対応するサブスクリプション・リスト・コンテナを指す DIT を示します。

- Oracle Human Resources は `cn=HRMS,cn=EBusiness,cn=Services,dc=com` として表されます。

  これは、サブスクリプション・リスト `cn=Accounts,cn=subscription_data, cn=subscriptions,orclapplicationcommonname=HRMS, cn=EBusiness,cn=Products,cn=OracleContext` を指します。

- Oracle Human Resources は cn=HRMS,cn=EBusiness,cn=Services,dc=com として表されます。

  これは、サブスクリプション・リスト cn=Accounts,cn=subscription_data,cn=subscriptions,orclapplication commonname=FINANCIALS, cn=EBusiness,cn=Products,cn=OracleContext を指します。

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
- 拡張可能なイベント定義
- 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：

- 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、4-15 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 Directory 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 Directory 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 Directory 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 Directory 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 4-4 に、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ます。

図 4-4 PL/SQL コールバック・インタフェース

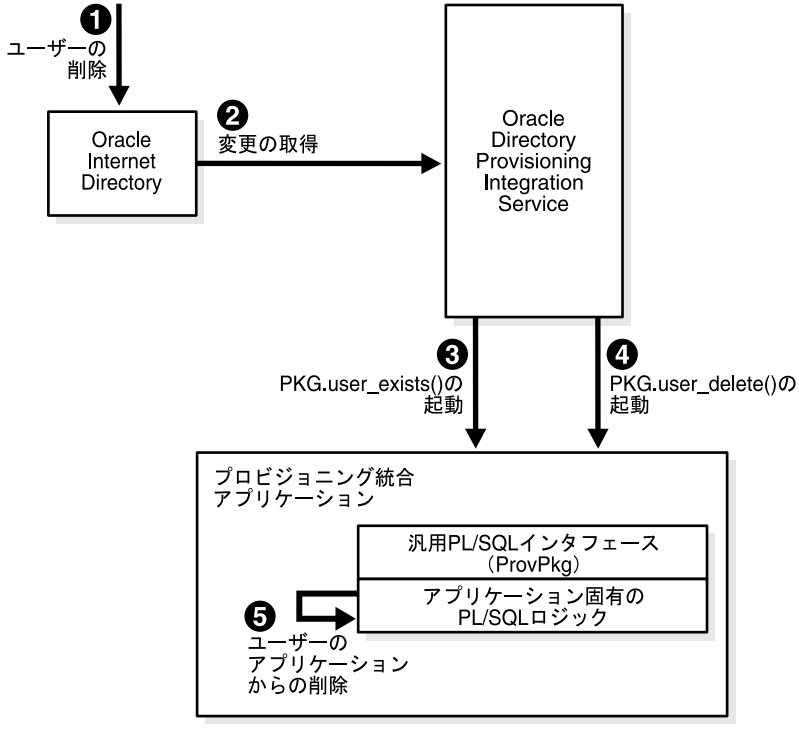


図 4-4 のように、ユーザーのアプリケーションからの削除は、次の手順で構成されています。

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 Directory 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 Directory 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 Directory 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。

拡張可能なイベント定義

この機能によって、Oracle Directory Provisioning Integration Service の機能を拡張し、事前定義されたプロビジョニング情報のセットをアプリケーションに戻すことができます。インストール時に次のイベントを構成し、適切なアプリケーションへ伝播します。

表 4-1 拡張可能なイベント定義

イベント定義	属性
イベント・オブジェクト型（orclODIPProvEventObjectType）	イベントが関連付けられているオブジェクトの型を指定します（例：USER、GROUP、IDENTITY）。
LDAP 変更型（orclODIPProvEventChangeType）	LDAP 操作でこの型のオブジェクトに対して生成できるイベントを示します（例：ADD、MODIFY、DELETE）。
イベント基準（orclODIPProvEventCriteria）	特定のオブジェクト型になる LDAP エントリを指定する追加の選択基準です。たとえば、Objectclass=orclUserV2 の場合、この基準を満たす LDAP エントリはこのオブジェクト型として識別され、このエントリを変更すると適切なイベントが生成されます。

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 Directory 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。

LDAP_NTFY ファンクションの定義

user_exists ファンクション

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 Directory 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 4-2 user_exists ファン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）正数を戻します。

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 Directory 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 4-3 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戻します。グループが存在しない場合は 0（ゼロ）を戻します。

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 Directory 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 4-4 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。

5

Oracle Internet Directory サーバー・プラグインの開発

この章では、カスタム開発を容易にする Oracle Internet Directory サーバーのプラグイン・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。

Oracle Internet Directory サーバー・プラグインの概要

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` 操作または `ldapmodify` 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。

Oracle Internet Directory サーバー・プラグインの開発の前提知識

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 と Oracle Application Server の統合
- SQL、PL/SQL およびデータベース RPC

Oracle Internet Directory サーバー・プラグインの概要

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

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。
- 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値検査を実装し、Oracle Internet Directory サーバー内に配置できます。

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。

> **関連項目：** ユーザー独自のパスワード値検査を実装し、Oracle Internet Directory サーバーに配置する方法については、『Oracle Internet Directory 管理者ガイド』の第 46 章を参照してください。

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。

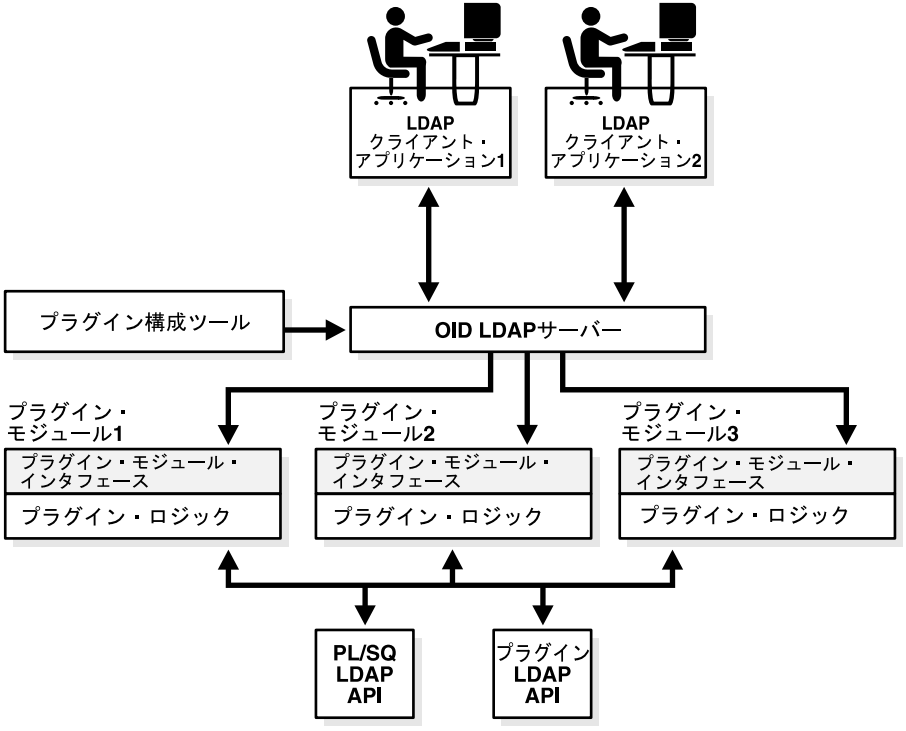
1. ユーザー定義のプラグイン・プロシージャを記述します。このプラグイン・モジュールは、PL/SQL で記述する必要があります。

   > **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. 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。

図 5-1 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 LDAP 操作の実行後にファンクションを起動することです。ロギングや通知などが、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

OID サーバーは、標準の LDAP 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。置換時のプラグインの場合、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、ldapadd、ldapcompare、ldapdelete、ldapmodify および ldapbind でのみサポートされます。追加時プラグインは、ldapadd、ldapdelete および ldapmodify でサポートされます。

Oracle Internet Directory プラグインの要件

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（DBMS_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 5-1 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	×	
プラグイン・プロシージャ名		×
プラグイン・プロシージャのシグネチャ		×

関連項目： サンプル・コードは、5-26 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 5-20 ページの「使用モデルと例」を参照してください。

次の表に、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ます。

表 5-2 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	return code、error message
ldapbind 時（ただし、デフォルト・サーバーの動作を置換）	WHEN_BIND_REPLACE	ldapcontext、bind result、DN、userpassword	bind result、return code、error message
ldapbind 後	POST_BIND	ldapcontext、bind result、bind dn、password	return code、error message
ldapmodify 前	PRE_MODIFY	ldapcontext、dn、mod structure	return code、error message
ldapmodify 時	WHEN_MODIFY	ldapcontext、dn、mod structure	return 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	return 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（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

表 5-2 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
ldapadd 時（ただし、デフォルト・サーバーの動作を置換）	WHEN_ADD_REPLACE	ldapcontext、entry	return code、error message
ldapadd 後	POST_ADD	ldapcontext、add result、entry	return code、error message
ldapdelete 前	PRE_DELETE	ldapcontext、dn	return code、error message
ldapdelete 時	WHEN_DELETE	ldapcontext、dn	return code、error message
ldapdelete 時（ただし、デフォルト・サーバーの動作を置換）	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 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、5-15 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、5-30 ページの「ディレクトリ・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、5-26 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。無名 PL/SQL 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 5-3 に、プラグインの属性のリストおよび説明を示します。

表 5-3 プラグインの属性名と属性値

属性名	属性値	必須かどうか
cn	プラグイン・エントリ名。	はい
orclPluginAttributeList （ldapcompare および ldapmodify プラグインのみ。）	セミコロンで区切られた属性名のリストで、プラグインの実行を制御します。ターゲット属性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。	いいえ
orclPluginEnable	0 = 使用禁止（デフォルト） 1 = 使用可能	いいえ
orclPluginEntryProperties	LDAP 検索フィルタ・タイプの値を指定する必要があります。たとえば、次のように指定します。 orclPluginEntryProperties:(&(objectclass=inetorgperson)(sn=Cezanne)) このとき、ターゲット・エントリの objectclass が inetorgperson であり、sn が Cezanne である場合、プラグインは起動されません。	いいえ
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。	いいえ
orclPluginKind	PL/SQL	いいえ

表 5-3 プラグインの属性名と属性値（続き）

属性名	属性値	必須かどうか
orclPluginLDAPOperation	次のいずれかの値です。 ldapcompare ldapmodify ldapbind ldapadd ldapdelete ldapsearch	はい
orclPluginName	プラグイン・パッケージ名。	はい
orclPluginRequestGroup	セミコロンで区切られたグループのリストで、プラグインの実行を制御します。このグループを使用して実際にプラグインを起動できるユーザーを指定できます。 たとえば、次のように指定します。 orclpluginrequestgroup:cn=security,cn=groups,dc=oracle,dc=com このとき、プラグインを登録すると、cn=security,cn=groups,dc=oracle,dc=com グループに属しているユーザーからの LDAP 要求ではない場合、プラグインは起動しません。	いいえ
orclPluginRequestNegGroup	セミコロンで区切られたグループのリストで、プラグインの実行を制御します。このグループを使用して実際にプラグインを起動できないユーザーを指定できます。たとえば、次のように指定します。 orclpluginrequestgroup:cn=security,cn=groups,dc=oracle,dc=com このとき、プラグインを登録すると、cn=security,cn=groups,dc=oracle,dc=com グループに属しているユーザーからの LDAP 要求の場合、プラグインは起動しません。	いいえ
orclPluginResultCode	LDAP 結果コードを指定する整数値です。この値が指定されると、LDAP 操作がその結果コードの使用例に含まれる場合のみプラグインが起動されます。 このプラグインのみが操作後プラグイン・タイプです。	いいえ

表 5-3 プラグインの属性名と属性値（続き）

属性名	属性値	必須かどうか
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。	いいえ
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。	いいえ
orclPluginTiming	次のいずれかの値です。 pre when post	いいえ
orclPluginType	次のいずれかの値です。 operational attribute password_policy syntax matchingrule **関連項目：**5-4 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」を参照してください。	はい
orclPluginVersion	サポート対象のプラグイン・バージョン番号。	いいえ

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の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
orclPluginAttributeList=userpassword
```

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
```

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、PL/SQL 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。

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（5-30 ページの「ディレクトリ・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が判断されます。

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

次の表に、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ます。

表 5-4 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_DELETE	LDAP 操作を完了します。ロールバックは行われません。
WHEN_MODIFY、WHEN_ADD、WHEN_DELETE	LDAP 操作をロールバックします。

次の表に、LDAP 操作に失敗した場合の、LDAP 操作の障害および障害に対する Oracle Internet Directory サーバーの処理を示します。

表 5-5 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_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_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_MODIFY、WHEN_ADD、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。
- 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ディレクトリ・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 ディレクトリ・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、5-19 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、次のような操作によって、一貫性のない状態になる場合があります。

- 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

プラグインとデータベース・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。
- アクセス制御リスト（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。

プラグインのデバッグ

Oracle Internet Directory のプラグインのデバッグでは、プラグインの処理および内容を検証することができます。次のコマンドを使用して、サーバーのデバッグ処理の操作を制御します。

- プラグインのデバッグを設定するには、次のコマンドを実行します。

```
% sqlplus ods/password @$ORACLE/ldap/admin/oidspdsu.pls
```

- プラグインのデバッグを可能にするには、次のコマンドを実行します。

```
% sqlplus ods/password @$ORACLE/ldap/admin/oidspdon.pls
```

- プラグインのデバッグを可能にした後、デバッグ・メッセージをプラグインのデバッグ表に格納するには、プラグイン・モジュール・コードで次のコマンドを実行します。

```
plg_debug('debuggingmessage');
```

- デバッグを使用不可にするには、次のコマンドを実行します。

```
% sqlplus ods/password @$ORACLE/ldap/admin/oidspdof.pls
```

- プラグイン・モジュールに設定したデバッグ・メッセージを表示するには、次のコマンドを実行します。

```
% sqlplus ods/password @$ORACLE/ldap/admin/oidspdsh.pls
```

- デバッグ表からすべてのデバッグ・メッセージを削除するには、次のコマンドを実行します。

```
% sqlplus ods/password @$ORACLE/ldap/admin/oidspdde.pls
```

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;
```

```
    -- Get requester info from the plugin context
    FUNCTION get_requester (ldappluginctx IN ODS.plugincontext)
      RETURN VARCHAR2;
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,
    errormsg     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,
    errormsg     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;
   errormsg := 'no post_search plugin error msg';
   COMMIT;
```

```
EXCEPTION
   WHEN others THEN
      rc := 1;
      errormsg := '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
```

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,
    errormsg 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;
   errormsg := 'no post_delete plguin error msg';
EXCEPTION
   WHEN others THEN
      rc := 1;
      errormsg := 'exception: post_delete plguin';
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
```

データベース・タイプの定義およびプラグイン・モジュール・インタフェースの仕様

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

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型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 when_add_replace (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,
    errormsg OUT VARCHAR2
    );

PROCEDURE pre_modify (ldapplugincontext IN  ODS.plugincontext,
    dn       IN  VARCHAR2,
    mods     IN  ODS.modlist,
    rc       OUT INTEGER,
    errormsg OUT VARCHAR2
    );

PROCEDURE when_modify (ldapplugincontext IN  ODS.plugincontext,
    dn       IN  VARCHAR2,
    mods     IN  ODS.modlist,
    rc       OUT INTEGER,
    errormsg OUT VARCHAR2
    );

PROCEDURE when_modify_replace (ldapplugincontext IN  ODS.plugincontext,
    dn       IN  VARCHAR2,
    mods     IN  ODS.modlist,
    rc       OUT INTEGER,
    errormsg OUT VARCHAR2
    );

PROCEDURE post_modify (ldapplugincontext IN  ODS.plugincontext,
    result   IN  INTEGER,
    dn       IN  VARCHAR2,
    mods     IN  ODS.modlist,
    rc       OUT INTEGER,
    errormsg OUT VARCHAR2
    );

PROCEDURE pre_compare (ldapplugincontext IN  ODS.plugincontext,
    dn       IN  VARCHAR2,
    attrname IN  VARCHAR2,
    attrval  IN  VARCHAR2,
    rc       OUT INTEGER,
    errormsg 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 when_delete_replace (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 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_replace (ldapplugincontext IN  ODS.plugincontext,
                      result     OUT INTEGER,
                         dn          IN  VARCHAR2,
                         passwd      IN  VARCHAR2,
                         rc          OUT INTEGER,
                         errormsg    OUT VARCHAR2
                         );

PROCEDURE post_bind (ldapplugincontext IN ODS.plugincontext,
    resultIN INTEGER,
    dnIN VARCHAR2,
    passwdIN VARCHAR2,
    rcOUT INTEGER,
    errormsg OUT VARCHAR2
    );

END plugin_test1;
/
```

ディレクトリ・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;
```

6

Oracle Delegated Administration Services に統合されたアプリケーションの開発

この章では、Oracle Delegated Administration Services URL サービス・ユニットを使用して Oracle Delegated Administration Services との統合を実現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。

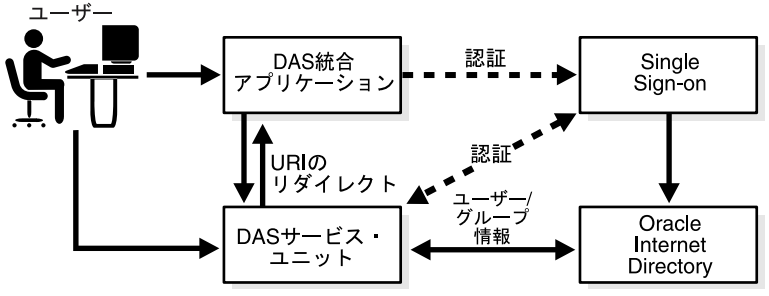
Delegated Administration Services の概要

Oracle Delegated Administration Services は、ユーザーのかわりにディレクトリ操作を実行するために事前定義された Web ベースの一連のサービスです。Oracle Delegated Administration Services ユニットによって、Oracle Internet Directory では、ディレクトリ・ユーザーが従業員ディレクトリにある自分の情報を更新するような、セルフ・サービス・モデルを使用できます。

Delegated Administration Services を使用すると、ディレクトリ内のアプリケーション・データを管理するツールをより簡単に開発できます。このサービスでは、ユーザー・エントリの作成、グループ・エントリの作成、エントリの検索、ユーザー・パスワードの変更など、ディレクトリ対応アプリケーションに必要な大部分の機能を提供します。

Delegated Administration Services ユニットは、アプリケーションに埋め込むことができます。たとえば、Web ポータルを構築している場合は、Oracle Delegated Administration Services ユニットを追加して、ディレクトリに格納されているアプリケーション・パスワードをユーザーが変更できるようにします。それぞれのサービス・ユニットは、ディレクトリに格納された URL に対応しています。アプリケーションは、ディレクトリへの問合せによる実行時の URL 検出によって、Oracle Delegated Administration Services ユニットを起動できます。

図 6-1 Delegated Administration Service の概要



Oracle Delegated Administration Services ベースのアプリケーションの利点

Oracle Delegated Administration Services ベースのアプリケーションは、従来の API をベースにしたアプリケーションと比較して、主に次の 3 つの点で優れています。

1 番目に、Oracle Delegated Administration Services ユニットは Web ベースであるため、DAS ユニットを使用して開発されたアプリケーションは言語に依存しません。これは、アプリケーションがすべてのタイプのユーザーまたはアプリケーションからの入力および要求を処理できるということであり、コストのかかるカスタム・ソリューションやカスタム・コンフィギュレーションの必要性がないということです。

2 番目に、Oracle Delegated Administration Services は、ディレクトリ指向アプリケーションの要件（作成、編集、削除など）の多くを自動化する GUI 開発ツールである、Oracle Internet Directory セルフ・サービス・コンソールを備えています。このツールによって、このような基本的な機能の設計時間および開発時間が削減されます。

3 番目に、Oracle Delegated Administration Services は Oracle Application Server Single Sign-On に統合されているため、Oracle Delegated Administration Services ベースのアプリケーションは、自動的に Oracle Application Server Single Sign-On に認証されます。これは、Oracle Delegated Administration Services を使用したアプリケーションをユーザーとしてプロキシ設定可能であり、より確実なセキュリティのもと、ユーザーのかわりにディレクトリを問い合せることができるということです。

Oracle Delegated Administration Services に統合されたアプリケーションの開発

この項では、次の項目について説明します。

Oracle Delegated Administration Services との統合の前提条件

アプリケーションを Oracle Delegated Administration Services ユニットに統合するには、次の条件を満たしている必要があります。

- アプリケーションは Web ベースの GUI である。
- アプリケーションは、`mod_osso` またはパートナ・アプリケーションを介して、Oracle Application Server Single Sign-On に統合されている。
- アプリケーションに、現在サインオンしているユーザーで実行される必要があり、Oracle Delegated Administration Services から利用できる一定の操作がある。
- アプリケーションに Oracle Internet Directory に格納されたユーザーまたはグループがあり、ユーザーおよびグループの管理に Oracle Delegated Administration Services が利用可能である。
- Oracle Delegated Administration Services URL 検出メカニズムが使用可能な Oracle Application Server Infrastructure または Middle-Tier 環境下で、アプリケーションが実行される。

Oracle Delegated Administration Services の統合方法および考慮事項

表 6-1 に、アプリケーションと Oracle Delegated Administration Services との統合の考慮事項を示します。

表 6-1 アプリケーションと Oracle Delegated Administration Services との統合の考慮事項

アプリケーション ライフサイクルでのポイント	考慮事項
アプリケーションの設計時	Oracle Delegated Administration Services が提供する各種サービスを検証し、アプリケーション GUI の統合ポイントを識別します。 パラメータを Oracle Delegated Administration Services セルフ・サービス・ユニットに渡し、Oracle Delegated Administration Services からの戻りパラメータも処理するために必要なコード変更を行います。 Oracle Internet Directory の構成情報から Oracle Delegated Administration Services ユニットの位置を動的に検出するために、ブートストラップおよびインストールのロジックにコードを導入します。導入するには、Oracle Internet Directory サービス検出 API を使用します。
アプリケーションのインストール時	Oracle Delegated Administration Services ユニットの位置を決定し、ローカル・リポジトリに格納します。
アプリケーションの実行	ユーザーに表示されるアプリケーション GUI に Oracle Delegated Administration Services の URL を表示します。 URL エンコーディングを使用して、Oracle Delegated Administration Services に適切なパラメータを渡します。 URL リターンを介して、Oracle Delegated Administration Services からのリターン・コードを処理します。
管理アクティビティの進行時	管理者の画面で Oracle Delegated Administration Services の位置および URL をリフレッシュする機能を提供します。これは、アプリケーションのインストール後に、配置によって Oracle Delegated Administration Services の位置が移動する場合に提供されます。

利用例 1: ユーザーの作成

この例では、Oracle Delegated Administration Services ユニット Create User をカスタム・アプリケーションに統合する方法を示します。カスタム・アプリケーションのページで、Create User がリンクとして表示されます。

1. 次の Java API を使用して Oracle Delegated Administration Services URL ベースを識別します。

   ```
   baseUrl = Util.getDASUrl(ctx,DASURL_BASE)
   ```

   この API は、次の形式で Oracle Delegated Administration Services ベース URL を戻します。

   ```
   http://host_name:port/
   ```

2. 次の文字列を使用して、Create User Oracle Delegated Administration Services ユニットに固有の URL を取得します。

   ```
   relUrl = Util.getDASUrl ( ctx , DASURL_CREATE_USER )
   ```

   戻り値は、Create User ユニットにアクセスするための相対 URL になります。

   固有の URL はアプリケーションに対するリンクを動的に生成するために必要な情報です。

   次に、このユニットに対してカスタマイズできるパラメータを示します。このユニットでは、次のパラメータを指定できます。

表 6-2 Oracle Delegated Administration Services URL パラメータ

パラメータ	説明
homeURL	Oracle Delegated Administration Services ユニットの Home グローバル・ボタンにリンクされた URL です。コール元のアプリケーションがこの値を指定すると、Home ボタンをクリックして Oracle Delegated Administration Services ユニットをこのパラメータで指定された URL へリダイレクトできます。
doneURL	この URL は、各操作の最後に Oracle Delegated Administration Services ページをリダイレクトするために、Oracle Delegated Administration Services が使用します。Create User の場合、ユーザーが作成された後、「OK」をクリックすると URL がこの位置にリダイレクトされます。このため、ユーザーのナビゲーション操作がスムーズになります。

表 6-2 Oracle Delegated Administration Services URL パラメータ（続き）

パラメータ	説明
cancelURL	この URL は、Oracle Delegated Administration Services ユニットに表示されるすべての「取消」ボタンにリンクされます。ユーザーが「取消」をクリックすると、常にそのページがこのパラメータで指定した URL にリダイレクトされます。
enablePA	このパラメータは、true または false のブール値をとります。これは、ユーザーまたはグループ操作での権限の割当てセクションを使用可能にします。Create User ページで enablePA に true が渡されると、Assign Privileges to User セクションも Create User ページに表示されます。

3. パラメータを次の値に設定して、リンクを作成します。

```
baseUrl = http://acme.mydomain.com:7777/
relUrl =  oiddas/ui/oracle/ldap/das/admin/AppCreateUserInfoAdmin
homeURL = http://acme.mydomain.com/myapp
cancelURL = http://acme.mydomain.com/myapp
doneURL = http://acme.mydomain.com/myapp
enablePA = true
```

URL 全体では、次のようになります。

```
http://acme.mydomain.com:7777/oiddas/ui/oracle/ldap/das/admin/AppCreateUserInfoA
dmin? homeURL=http://acme.mydomain.com/myapp&
cancelURL=http://acme.mydomain.com/myapp
& doneURL=http://acme.mydomain.com/myapp& enablePA=true
```

4. これでアプリケーションにこの URL を埋め込むことができます。

利用例 2: ユーザー LOV

Oracle Delegated Administration Services の値リスト（LOV）は、JavaScript を使用して実装されて起動し、LOV のコール元ウィンドウと Oracle Delegated Administration Services の LOV ページ間で値を渡します。LOV を起動するアプリケーションは、JavaScript を使用してポップアップ・ウィンドウを開く必要があります。JavaScript にはセキュリティ上の制限があるため、ドメイン間でデータを渡すことはできません。この制限により、同じドメイン内のページのみが Oracle Delegated Administration Services LOV ユニットにアクセスできます。

ベースおよび相対 URL は、Create User と同じ方法で起動できます。サンプル・ファイルは、次のディレクトリにあります。

```
$ORACLE_HOME/ldap/das/samples/lov
```

この例では、LOV が起動され、コール元のアプリケーションと Oracle Delegated Administration Services ユニット間でデータが渡される方法を示しています。LOV の起動の詳細については、この章では説明していません。

URL アクセスに使用する Java API

Oracle Delegated Administration Services URL を検出するには、Java API を使用します。Java API の詳細は、第 3 章「標準的な LDAP API に対する Oracle の拡張機能を使用したアプリケーションの開発」および第 10 章「DAS_URL インタフェース・リファレンス」を参照してください。Oracle Delegated Administration Services URL を検出する API ファンクションには、次のものがあります。

- `getDASUrl`（DirContext ctx、String urlTypeDN）
- `getAllDASUrl`（DirContext ctx）

第 II 部

Oracle Internet Directory プログラミング・リファレンス

ここでは、標準 API に対する Oracle 固有の拡張機能について説明します。C、PL/SQL、Oracle Delegated Administration Services およびプロビジョニング統合 API のクラス、例外および使用例を示すリファレンスの章があります。API の詳細なリファレンス情報については、プロダクト CD から参照できます。

第 II 部は、次の章で構成されています。

7

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。

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
- C API の依存性と制限事項

Oracle Internet Directory C API の概要

Oracle Internet Directory SDK C API は、LDAP バージョン 3 C API、および SSL をサポートする Oracle の拡張機能をベースにしています。

Oracle Internet Directory API 10*g*（9.0.4）は、次のモードで使用できます。

- SSL モードー SSL を使用してすべての通信を保護する
- 非 SSL モードークライアント / サーバー間の通信を保護しない

API は、TCP/IP を使用してディレクトリ・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、7-61 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。

- 認証なしー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
- サーバー認証ー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ー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。

表 7-1 SSL インタフェース・コールの引数

引数	説明
sb	LDAP ハンドルの一部として、*ldap_open* コールによって戻されたソケット・バッファ・ハンドル。
sslwallet	ユーザー Wallet の位置。
sslwalletpasswd	Wallet を使用するために必要なパスワード。
sslauthmode	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。 ■ GSLC_SSL_NO_AUTH －認証の必要なし ■ GSLC_SSL_ONEWAY_AUTH －サーバー認証のみ必要 ■ GSLC_SSL_TWOWAY_AUTH －クライアントとサーバーの両方の認証が必要 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、ldap_err2string ファンクションを使用してエラー・メッセージに変換できます。

関連項目： 7-61 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 10*g*（9.0.4）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

表 7-2 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 7-2 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
ora_ldap_create_clientctx	クライアントのコンテキストの作成およびハンドルを戻す
ora_ldap_create_cred_hdl	資格証明ハンドルの作成
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	
ora_ldap_destroy_clientctx	クライアントのコンテキストの破棄
ora_ldap_free_cred_hdl	資格証明ハンドルの破棄
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
ora_ldap_get_cred_props	資格証明ハンドルに関連付けられたプロパティの取得

表 7-2 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_get_dn	エントリの識別名の取得
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
ldap_init ldap_open	LDAP サーバーへの接続のオープン
ora_ldap_init_SASL	SASL 認証の実行
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 ldap_modify_ext_s ldap_modify ldap_modify_s	ディレクトリ内のエントリの変更
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 使用不可	エラー・メッセージの中から指定したメッセージを出力
ldap_rename ldap_rename_s	ディレクトリ内のエントリの相対識別名を変更
ldap_result2error 使用不可	結果メッセージからエラー・コードを取得

表 7-2 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_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	
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
ora_ldap_set_clientctx	クライアントのコンテキスト・ハンドルへのプロパティの追加
ora_ldap_set_cred_props	資格証明ハンドルへのプロパティの追加
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ーの解放
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ーの解放

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： 各コールの詳細は、次の URL を参照してください。

```
http://www.ietf.org/
```

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_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 7-3 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	空白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。 **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、後述する 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。

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、この項で説明する 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。

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 7-4 に、LDAP セッション・ハンドル・オプションのパラメータを示します。

表 7-4 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。

表 7-4 LDAP セッション・ハンドル・オプションのパラメータ（続き）

パラメータ	説明
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 7-5 で説明する定数のいずれかであることが必要です。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。

定数

表 7-5 に、LDAP セッション・ハンドル・オプションの定数を示します。

表 7-5 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できることが必要です。このオプションは READ-ONLY で、設定はできません。
ORA_LDAP_OPT_RFRL_CACHE	void *（LDAP_OPT_ON または LDAP_OPT_OFF）	int *	このオプションは、参照キャッシュが使用可能かどうかを指定します。このオプションを LDAP_OPT_ON に設定するとキャッシュは使用可能、LDAP_OPT_OFF に設定すると使用不可です。
ORA_LDAP_OPT_RFRL_CACHE_SZ	int *	int *	このオプションは、参照キャッシュのサイズを指定します。サイズは、キャッシュを大きくできるバイト数の最大サイズです。デフォルトで 1MB が設定されます。

表 7-5 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_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。
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。

表 7-5 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_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停止したとき、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。 **関連項目：**7-15 ページの「コントロールの使用」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。 **関連項目：**7-15 ページの「コントロールの使用」を参照してください。
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。

表 7-5 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_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、ldap_init() に対する hostname 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 を戻します。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;
```

表 7-6 に、ldapcontrol 構造体のフィールドを示します。

表 7-6 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（7-10 ページの「ldap_set_option」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。

クライアント・コントロールの参照プロセス　7-9 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 7-7 に、ディレクトリに対する認証のパラメータを示します。

表 7-7 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。
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。

表 7-7 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
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。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()（7-46 ページの「ldap_result」を参照）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()（7-46 ページの「ldap_result」を参照）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_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。

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。

関連項目： エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

Oracle の拡張機能を使用した SASL 認証

ora_ldap_init_SASL() ファンクションを使用すると、SASL 認証を実行できます。

このファンクションは、他の引数とともに指定すると、次のものを受け入れます。

- 認証されるエンティティの識別名。
- エンティティに対する SASL 資格証明ハンドル （このハンドルは、ora_ldap_create_cred_hdl()、ora_ldap_set_cred_props() および ora_ldap_free_cred_hdl() ファンクションを使用して管理できます)。
- 使用する SASL メカニズム。

このファンクションは、様々な標準 SASL メカニズムのクライアントとディレクトリ・サーバー間の SASL ハンドシェイクをカプセル化するため、ディレクトリ・サーバーへの SASL ベース接続の確立でのコードディングの負荷が軽減されます。

サポートされる SASL メカニズムは次のとおりです。

- DIGEST-MD5

  SASL API では、Digest-MD5 の認証専用モードをサポートしています。データ・プライバシおよびデータ整合性に対応する他の 2 つの認証モードもサポートされています。

  Oracle Internet Directory に対して認証する際は、ユーザーの識別名はサーバーに送信される前に正規化される必要があります。これは、DN を SASL API に渡す前に `ora_ldap_normalize_dn()` ファンクションを使用して SASL API の外部で行うか、または `ora_ldap_set_cred_handle()` を使用して、SASL 資格証明ハンドルに `ORA_LDAP_CRED_SASL_NORM_AUTHDN` オプションを設定して SASL API で行うことができます。

- EXTERNAL

  Oracle Internet Directory での SASL API および SASL の実装には、外部認証メカニズムの 1 つとして SSL 認証を使用します。

  このメカニズムを使用するには、`ora_ldap_init_SSL()` ファンクションを使用して、ディレクトリ・サーバーに対して SSL 接続（相互認証モード）を確立する必要があります。これによって、`ora_ldap_init_SASL()` ファンクションは、メカニズムの引数に EXTERNAL を指定して起動できます。ディレクトリ・サーバーは、SSL 接続のユーザー資格証明に基づいてユーザーを認証します。

次のファンクションを使用して、SASL 資格証明ハンドルを作成および管理します。

ora_ldap_create_cred_hdl

ora_ldap_set_cred_props

ora_ldap_get_cred_props

ora_ldap_free_cred_hdl

ora_ldap_create_cred_hdl ファンクションは、SASL 資格証明に使用するメカニズムのタイプをベースにした、特定のタイプの SASL 資格証明ハンドルの作成に使用します。ora_ldap_set_cred_props() は、SASL 認証に必要なハンドルに関連する資格証明の追加に使用できます。ora_ldap_get_cred_props() ファンクションは、資格証明ハンドルに格納されたプロパティの取得に、ora_ldap_free_cred_hdl () ファンクションは、使用後のハンドルの破棄に使用できます。

構文

```
OraLdapHandle ora_ldap_create_cred_hdl
(
        OraLdapClientCtx * clientCtx,
        int                credType
);

OraLdapHandle ora_ldap_set_cred_props
(
        OraLdapClientCtx *   clientCtx,
        OraLdapHandle        cred,
        int                  propType,
        void             *   inProperty
);
OraLdapHandle ora_ldap_get_cred_props
(
        OraLdapClientCtx *   clientCtx,
        OraLdapHandle   cred,
        int                   propType,
        void                *   outProperty
);
```

```
OraLdapHandle ora_ldap_free_cred_hdl
(
        OraLdapClientCtx *   clientCtx,
        OraLdapHandle   cred
);
```

パラメータ

表 7-8 SASL 資格証明の管理パラメータ

パラメータ	説明
clientCtx	C API のクライアント・コンテキストです。これは、`ora_ldap_init_clientctx()` および `ora_ldap_free_clientctx()` ファンクションを使用して管理できます。
credType	SASL メカニズム固有の資格証明ハンドルのタイプです。
cred	SASL 認証のための固有の SASL メカニズムに必要な SASL 資格証明を含む資格証明ハンドルです。
propType	資格証明ハンドルに追加する必要がある資格証明のタイプです。
inProperty	資格証明ハンドルに格納される SASL 資格証明の 1 つです。
outProperty	資格証明ハンドルに格納された SASL 資格証明の 1 つです。

SASL 認証

次のファンクションを使用して、SASL 認証を実行します。

ora_ldap_init_SASL

このファンクションは、入力引数の 1 つに指定したメカニズムに基づいて SASL 認証を実行します。

構文

```
int ora_ldap_init_SASL
(
OraLdapClientCtx * clientCtx,
LDAP*ld,
char* dn,
char* mechanism,
OraLdapHandle cred,
```

```
LDAPControl**serverctrls,
LDAPControl**clientctrls
);
```

パラメータ

表 7-9 SASL 資格証明の管理パラメータ

パラメータ	説明
clientCtx	C API のクライアント・コンテキストです。これは、`ora_ldap_init_clientctx()` および `ora_ldap_free_clientctx()` ファンクションを使用して管理できます。
ld	LDAP セッション・ハンドルです。
dn	認証が必要なユーザーの DN です。
mechanism	SASL メカニズムです。
cred	SASL 認証に必要な資格証明です。
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。

セッションのクローズ

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。

ldap_unbind_ext

ldap_unbind

ldap_unbind_s

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 7-10 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

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。

ldap_search_ext

ldap_search_ext_s

ldap_search

ldap_search_s

ldap_search_st

構文

```
int ldap_search_ext
(
    LDAP            *ld,
    const char      *base,
    int             scope,
    const char      *filter,
    char            **attrs,
    int             attrsonly,
    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 7-11 に、検索操作のパラメータを示します。

表 7-11 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=*）を使用するように指定できます。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、サーバーから属性の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 7-11 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 7-11 検索操作のためのパラメータ（続き）

パラメータ	説明
msgidp	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。 ■ 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。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()（7-46 ページの「ldap_result」を参照）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()（7-46 ページの「ldap_result」を参照）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_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。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、この項で説明する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停止します。

> **関連項目：** エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

エントリの読取り

LDAP では、読取り操作を直接サポートしていません。かわりに、この読取り操作は、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 7-12 に、比較操作のパラメータを示します。

表 7-12 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。
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。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()（7-46 ページの「ldap_result」を参照）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()（7-46 ページの「ldap_result」を参照）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_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。

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。

関連項目： エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

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 7-13 に、変更操作のパラメータを示します。

表 7-13 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。

表 7-14 に、LDAPMod 構造体のフィールドを示します。

表 7-14 LDAPMod 構造体のフィールド

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。
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。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()（7-46 ページの「ldap_result」を参照）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()（7-46 ページの「ldap_result」を参照）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。

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。

> **関連項目：** エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

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 7-15 に、名前の変更操作のパラメータを示します。

表 7-15 名前の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	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。ルートの識別名は、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、このパラメータは名前の変更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。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() (7-46 ページの「ldap_result」を参照) をコールすると、名前の変更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。

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。

> **関連項目：** エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

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 7-16 に、追加操作のパラメータを示します。

表 7-16 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・コードを戻します。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()（7-46 ページの「ldap_result」を参照）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 を戻します。ldap_add_ext() の場合は、後で ldap_result()（7-46 ページの「ldap_result」を参照）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_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。

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。

> **関連項目：** エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

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 7-17 に、削除操作のパラメータを示します。

表 7-17 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。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() (7-46 ページの「ldap_result」を参照) 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() (7-46 ページの「ldap_result」を参照) 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。

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。

> **関連項目：** エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

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 7-18 に、拡張操作のパラメータを示します。

表 7-18 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。
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。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() (7-46 ページの「ldap_result」を参照) 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。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。

> **関連項目：** エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

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 7-19 に、操作を中止するためのパラメータを示します。

表 7-19 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()` ファンクションは、`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。

`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。

> **関連項目：** エラーとその解析方法の詳細は、7-48 ページの「エラーの処理と結果の解析」を参照してください。

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 7-20 に、結果の取得と LDAP メッセージの確認のためのパラメータを示します。

表 7-20 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。
すべて	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 を戻します。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戻します。res が NULL の場合は何も処理されず、値 0（ゼロ）が戻されます。

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戻します。前述のタイプのいずれかを戻します。エラーの場合は -1 を戻します。

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 を戻します。エラーの場合は -1 を戻します。

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            **matcheddnp,
    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 7-21 に、エラーの処理と結果の解析を行うためのパラメータを示します。

表 7-21 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。
matcheddnp	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 7-21 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 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。
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、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。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 berval 構造体へのポインタが入力されます。このパラメータは、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。
err	ldap_err2string() に関する LDAP エラー・コードで、ldap_parse_result() 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

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。

ldap_first_message

ldap_next_message

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。前述の ldap_msgtype() ファンク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 7-22 に、結果リストを参照するためのパラメータを示します。

表 7-22 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、この項で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 7-23 に、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を示します。

表 7-23 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 7-24 に、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を示します。

表 7-24 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され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 タイプ自体は、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() や関連するファン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 7-25 に、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を示します。

表 7-25 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 を戻します。

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 を戻します。

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 7-26 に、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を示します。

表 7-26 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。

表 7-26 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（続き）

パラメータ	説明
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。このファンクションは、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 7-27 に、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を示します。

表 7-27 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・コードを戻します（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 7-28 に、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を示します。

表 7-28 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 C API の使用例、および SASL 認証の C 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・ツールのサンプル・コードも、SSL モードおよび非 SSL モードでの API の使用方法を示します。

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
- SASL ベースの Digest-MD5 認証での 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_ONEWAY_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 をコールしているので、この例でのクライアント / サーバー間の通信は、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 / サーバー間の通信は保護されていません。

SASL ベースの Digest-MD5 認証での C API の使用方法

この例では、ディレクトリ・サーバーに対する SASL ベースの Digest-MD5 認証における LDAP SASL C-API の使用方法を示します。

```
EXPORT FUNCTION(S)
  NONE

INTERNAL FUNCTION(S)
```

```
    NONE

  STATIC FUNCTION(S)
    NONE

  NOTES
    Usage:
     saslbind -h <LDAP host> -p < LDAP port> -D < Authentication identity DN> ¥
              -w <password >
     options
      -h     LDAP host
      -p     LDAP port
      -D     DN of the identity for authentication
      -p     Password

      Default SASL authentication parameters used by the demo program
      SASL Security Property :    Currenty only "auth" security property
                                  is supported by the C-API. This demo
                                  program uses this security property.
      SASL Mechanism         :    Supported mechanisms by OID
                                  "DIGEST-MD5" - This demo program
                                                 illustrates it's usage.
                                  "EXTERNAL" - SSL authentication is used.
                                               (This demo program does
                                                not illustrate it's usage.)
      Authorization identity :    This demo program does not use any
                                   authorization identity.

  MODIFIED   (MM/DD/YY)
  ******      06/12/03 - Creation

*/

/*-----------------------------------------------------------------------
                    PRIVATE TYPES AND CONSTANTS
  -----------------------------------------------------------------------*/

/*-----------------------------------------------------------------------
                    STATIC FUNCTION DECLARATIONS
  -----------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <ldap.h>


static int ldap_version = LDAP_VERSION3;
```

```
main (int argc, char **argv)
{
  LDAP*           ld;
  extern char*    optarg;
  char*           ldap_host = NULL;
  char*           ldap_bind_dn = NULL;
  char*           ldap_bind_pw = NULL;
  int             authmethod = 0;
  char            ldap_local_host[256] = "localhost";
  int             ldap_port = 389;
  char*           authcid = (char *)NULL;
  char*           mech = "DIGEST-MD5"; /* SASL mechanism */
  char*           authzid = (char *)NULL;
  char*           sasl_secprops = "auth";
  char*           realm = (char *)NULL;
  int             status = LDAP_SUCCESS;
  OraLdapHandle   sasl_cred = (OraLdapHandle )NULL;
  OraLdapClientCtx *cctx = (OraLdapClientCtx *)NULL;
  int              i = 0;


    while (( i = getopt( argc, argv,
    "D:h:p:w:E:P:U:V:W:O:R:X:Y:Z"
    )) != EOF ) {
switch( i ) {

case 'h':/* ldap host */
    ldap_host = (char *)strdup( optarg );
    break;
case 'D':/* bind DN */
    authcid = (char *)strdup( optarg );
    break;

case 'p':/* ldap port */
    ldap_port = atoi( optarg );
    break;
case 'w':/* Password */
    ldap_bind_pw = (char *)strdup( optarg );
    break;

    default:
    printf("Invalid Arguments passed¥n" );
}
    }
```

```
  /* Get the connection to the LDAP server */
  if (ldap_host == NULL)
    ldap_host = ldap_local_host;

  if ((ld = ldap_open (ldap_host, ldap_port)) == NULL)
  {
    ldap_perror (ld, "ldap_init");
    exit (1);
  }

  /* Create the client context needed by LDAP C-API Oracle Extension functions*/
  status = ora_ldap_init_clientctx(&cctx);

  if(LDAP_SUCCESS != status) {
     printf("Failed during creation of client context ¥n");
     exit(1);
  }

  /* Create SASL credentials */
  sasl_cred = ora_ldap_create_cred_hdl(cctx, ORA_LDAP_CRED_HANDLE_SASL_MD5);

  ora_ldap_set_cred_props(cctx, sasl_cred, ORA_LDAP_CRED_SASL_REALM, (void *)realm);
  ora_ldap_set_cred_props(cctx, sasl_cred, ORA_LDAP_CRED_SASL_AUTH_PASSWORD, (void
*)ldap_bind_pw);
  ora_ldap_set_cred_props(cctx, sasl_cred, ORA_LDAP_CRED_SASL_AUTHORIZATION_ID,(void
*)authzid);
  ora_ldap_set_cred_props(cctx, sasl_cred, ORA_LDAP_CRED_SASL_SECURITY_PROPERTIES,
(void *)sasl_secprops);

  /* If connecting to OID using SASL DIGEST-MD5, the Authentication ID
     has to be normalized before it's sent to the server,
     the LDAP C-API does this normalization based on the following flag set in
     SASL credential properties */
  ora_ldap_set_cred_props(cctx, sasl_cred, ORA_LDAP_CRED_SASL_NORM_AUTHDN, (void
*)NULL);

  /* SASL Authetication to LDAP Server */
  status = (int)ora_ldap_init_SASL(cctx, ld, (char *)authcid, (char
*)ORA_LDAP_SASL_MECH_DIGEST_MD5,
     sasl_cred, NULL, NULL);

  if(LDAP_SUCCESS == status) {
     printf("SASL bind successful ¥n" );
  }else {
     printf("SASL bind failed with status :  %d¥n", status);
  }
```

```
  /* Free SASL Credentials */
  ora_ldap_free_cred_hdl(cctx, sasl_cred);

  status = ora_ldap_free_clientctx(cctx);

  /* Unbind from LDAP server */
  ldap_unbind (ld);

  return (0);
}


/* end of file saslbind.c */
```

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、次のことが必要です。

- $*ORACLE_HOME*/ldap/public/ldap.h にあるヘッダー・ファイルを含める。
- $*ORACLE_HOME*/lib/libclntsh.so.9.0 にあるライブラリに動的にリンクする。

サンプル検索ツールの作成

Oracle Internet Directory SDK 10*g*（9.0.4）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 $*ORACLE_HOME*/ldap/demo ディレクトリにあります。

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, and so on>
*/
#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¥trun 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, includeufn, 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*sslpasswd = NULL;
int sslauth=0,err=0;

    infile = NULL;
    deref = verbose = allow_binary = not = kerberos = vals2tmp =
    attrsonly = 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
    "KknuvtRABLD: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 -DLDAP_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 Authenication 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 );
}
```

C API の依存性と制限事項

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グローバリゼーション・サポート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ディレクトリ・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。

8

DBMS_LDAP PL/SQL 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。前述の DBMS_LDAP PL/SQL パッケージの情報を読んでから、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

表 8-1 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。
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。
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、PL/SQL 環境に制御が戻されません。
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、PL/SQL 環境に制御が戻されません。
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。
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。
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。

表 8-1 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
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。
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。
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。
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。
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。
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。
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。
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。
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。
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。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。
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。
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。add_s をコールする前に、DBMS_LDAP.creat_mod_array() と DBMS_LDAP.populate_mod_array() をコールする必要があります。

表 8-1 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
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、DBMS_LDAP.creat_mod_array() と DBMS_LDAP.populate_mod_array() をコールする必要があります。
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。
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。
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。
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。
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。
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。
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。
nls_convert_to_utf8 ファンクション	データベース・キャラクタ・セットのデータを含む入力文字列を UTF8 キャラクタ・セットのデータに変換して戻します。
nls_convert_from_utf8 ファンクション	UTF8 キャラクタ・セットのデータを含む入力文字列をデータベース・キャラクタ・セットのデータに変換して戻します。
nls_get_dbcharset_name ファンクション	データベース・キャラクタ・セット名を含む文字列を戻します。

関連項目：

- DBMS_LDAP.search_s() および DBMS_LDAP.search_st() ファンクションの詳細は、「ディレクトリの検索」を参照してください。
- DBMS_LDAP.unbind_s() ファンクションの詳細は、「DBMS_LDAP を使用したセッションの終了」を参照してください。

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 8-2 DBMS_LDAP 例外の概要

例外名	Oracle エラー番号	例外の原因
general_error	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に呼び出されます。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。
init_failed	31203	DBMS_LDAP.init() でなんらかの問題がある場合に呼び出されます。
invalid_session	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。
invalid_auth_method	31205	DBMS_LDAP.bind_s() 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。
invalid_search_scope	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。
invalid_search_time_val	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 DBMS_LDAP.search_st() によって呼び出されます。
invalid_message	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。
count_entry_error	31209	DBMS_LDAP.count_entries 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。
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、追加または名前を変更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。

表 8-2 DBMS_LDAP 例外の概要（続き）

例外名	Oracle エラー番号	例外の原因
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 によって名前を変更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 8-3 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 BERVAL 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 8-4 INIT ファンクションのパラメータ

パラメータ	説明
hostname	空白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 8-5 INIT ファンクションの戻り値

値	説明
SESSION（ファンクションの戻り値）	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 8-6 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 8-7 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 8-8 SIMPLE_BIND_S ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-9 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 8-10 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 8-11 BIND_S ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-12 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 ld 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 8-13 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 8-14 UNBIND_S ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-15 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 8-16 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 8-17 COMPARE_S ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-18 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 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

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 8-19 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。

表 8-19 SEARCH_S ファンクションのパラメータ（続き）

パラメータ	説明
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、サーバーが属性の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。
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。
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 8-20 SEARCH_S ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-21 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,
    attronly IN  PLS_INTEGER,
    tv       IN  TIMEVAL,
    res      OUT MESSAGE
)
    RETURN PLS_INTEGER;
```

パラメータ

表 8-22 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。

表 8-22 SEARCH_ST ファンクションのパラメータ（続き）

パラメータ	説明
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、サーバーが属性の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。
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 8-23 SEARCH_ST ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-24 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()、DBML_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 8-25 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 8-26 FIRST_ENTRY の戻り値

値	説明
MESSAGE（ファンクションの戻り値）	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 8-27 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 8-28 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 8-29 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 8-30 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 `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

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 8-31 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 8-32 COUNT_ENTRY ファンクションの戻り値

値	説明
PLS INTEGER（ファンクションの戻り値）	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 8-33 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 8-34 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 8-35 FIRST_ATTRIBUTE ファンクションの戻り値

値	説明
VARCHAR2（ファンクションの戻り値）	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 8-36 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 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、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 8-37 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 8-38 NEXT_ATTRIBUTE ファンクションの戻り値

値	説明
VARCHAR2（ファンクションの戻り値）	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 8-39 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 8-40 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 8-41 GET_DN ファンクションの戻り値

値	説明
VARCHAR2（ファンクションの戻り値）	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 8-42 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 8-43 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 8-44 GET_VALUES ファンクションの戻り値

値	説明
STRING_COLLECTION（ファンクションの戻り値）	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 8-45 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もあります。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,
    ldapentry IN MESSAGE,
    attr IN VARCHAR2
)
    RETURN BINVAL_COLLECTION;
```

パラメータ

表 8-46 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 8-47 GET_VALUES_LEN ファンクションの戻り値

値	説明
BINVAL_COLLECTION（ファンクションの戻り値）	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 8-48 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 8-49 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 8-50 DELETE_S ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-51 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 8-52 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 8-53 MODRDN2_S ファンクションの戻り値

値	説明
PLS_INTEGER（ファンクションの戻り値）	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 8-54 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 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。
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 8-55 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 8-56 ERR2STRING ファンクションの戻り値

値	説明
VARCHAR2（ファンクションの戻り値）	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 8-57 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 8-58 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 8-59 CREATE_MOD_ARRAY ファンクションの戻り値

値	説明
MOD_ARRAY（ファンクションの戻り値）	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 8-60 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 8-61 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の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 8-62 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 8-63 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 8-64 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の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 8-65 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 8-66 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。

表 8-66 POPULATE_MOD_ARRAY（バイナリ・バージョン）プロシージャの例外（続き）

例外	説明
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 8-67 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 8-68 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 8-69 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 8-70 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 8-71 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 8-72 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している必要があります。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 8-73 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 8-74 FREE_MOD_ARRAY プロシージャの戻り値

値	説明
該当なし	

例外

表 8-75 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 8-76 COUNT_VALUES ファンクションのパラメータ

パラメータ	説明
値	文字列値の集合です。

戻り値

表 8-77 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 8-78 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 8-79 COUNT_VALUES_LEN ファンクションのパラメータ

パラメータ	説明
値	バイナリ値の集合です。

戻り値

表 8-80 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 8-81 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 8-82 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、名前を変更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 8-83 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 8-84 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 8-85 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 8-86 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 8-87 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 8-88 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 8-89 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 8-90 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 8-91 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 8-92 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。 ■ DBMS_LDAP.LDAP_RES_BIND ■ DBMS_LDAP.LDAP_RES_SEARCH_ENTRY ■ DBMS_LDAP.LDAP_RES_SEARCH_REFERENCE ■ DBMS_LDAP.LDAP_RES_SEARCH_RESULT ■ DBMS_LDAP.LDAP_RES_MODIFY ■ DBMS_LDAP.LDAP_RES_ADD ■ DBMS_LDAP.LDAP_RES_DELETE ■ DBMS_LDAP.LDAP_RES_MODDN ■ DBMS_LDAP.LDAP_RES_COMPARE ■ DBMS_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
```
PROCEDURE ber_free
(
    ber_elem IN BER_ELEMENT,
    freebuf  IN PLS_INTEGER
)
```

パラメータ

表 8-93 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()

nls_convert_to_utf8 ファンクション
データベース・キャラクタ・セットのデータを含む入力文字列を UTF8 キャラクタ・セットのデータに変換して戻します。

構文
```
Function nls_convert_to_utf8
(
data_local IN VARCHAR2
)
RETURN VARCHAR2;
```

パラメータ

表 8-94 nls_convert_to_utf8 のパラメータ

パラメータ	説明
data_local	データベース・キャラクタ・セットのデータを指定します。

戻り値

表 8-95 nls_convert_to_utf8 の戻り値

値	説明
VARCHAR2	UTF8 キャラクタ・セットのデータ文字列です。

使用方法

DBMS_LDAP パッケージのファンクションは、UTF8_CONVERSION パッケージ変数に FALSE が設定されている場合、入力データが UTF8 キャラクタ・セットであると想定します。この場合、nls_convert_to_utf8() ファンクションを使用すると、入力データをデータベース・キャラクタ・セットから UTF8 キャラクタ・セットに変換できます。

DBMS_LDAP パッケージの UTF8_CONVERSION パッケージ変数に TRUE が設定されている場合、DBMS_LDAP パッケージのファンクションは、入力データがデータベース・キャラクタ・セットであると想定します。

関連項目

DBMS_LDAP.nls_convert_from_utf8()、DBMS_LDAP.nls_get_dbcharset_name().

nls_convert_to_utf8 ファンクション

データベース・キャラクタ・セットのデータを含む入力文字列コレクションを UTF8 キャラクタ・セットのデータに変換して戻します。

構文

```
Function nls_convert_to_utf8
(
data_local IN STRING_COLLECTION
)
RETURN STRING_COLLECTION;
```

パラメータ

表 8-96 nls_convert_to_utf8 のパラメータ

パラメータ	説明
data_local	データベース・キャラクタ・セットのデータを含む文字列のコレクションです。

戻り値

表 8-97 nls_convert_to_utf8 の戻り値

値	説明
STRING_COLLECTION	UTF8 キャラクタ・セットのデータを含む文字列のコレクションです。

使用方法

DBMS_LDAP パッケージのファンクションは、UTF8_CONVERSION パッケージ変数に FALSE が設定されている場合、入力データが UTF8 キャラクタ・セットであると想定します。この場合、nls_convert_to_utf8() ファンクションを使用すると、入力データをデータベース・キャラクタ・セットから UTF8 キャラクタ・セットに変換できます。

DBMS_LDAP パッケージの UTF8_CONVERSION パッケージ変数に TRUE が設定されている場合、DBMS_LDAP パッケージのファンクションは、入力データがデータベース・キャラクタ・セットであると想定します。

関連項目

DBMS_LDAP.nls_convert_from_utf8()、DBMS_LDAP.nls_get_dbcharset_name().

nls_convert_from_utf8 ファンクション

UTF8 キャラクタ・セットのデータを含む入力文字列をデータベース・キャラクタ・セットのデータに変換して戻します。

構文

```
Function nls_convert_from_utf8
(
data_utf8 IN VARCHAR2
)
RETURN VARCHAR2;
```

パラメータ

表 8-98 nls_convert_from_utf8 のパラメータ

パラメータ	説明
data_utf8	UTF8 キャラクタ・セットのデータを指定します。

戻り値

表 8-99 nls_convert_from_utf8 の戻り値

値	説明
VARCHAR2	データベース・キャラクタ・セットのデータ文字列です。

使用方法

DBMS_LDAP パッケージのファンクションは、UTF8_CONVERSION パッケージ変数に FALSE が設定されている場合、UTF8 キャラクタ・セット・データを戻します。この場合、nls_convert_from_utf8() ファンクションを使用すると、出力データを UTF8 キャラクタ・セットからデータベース・キャラクタ・セットに変換できます。

DBMS_LDAP パッケージの UTF8_CONVERSION パッケージ変数に TRUE が設定されている場合、DBMS_LDAP パッケージのファンクションは、データベース・キャラクタ・セット・データを戻します。

関連項目

DBMS_LDAP.nls_convert_to_utf8()、DBMS_LDAP.nls_get_dbcharset_name()

nls_convert_from_utf8 ファンクション

UTF8 キャラクタ・セットのデータを含む入力文字列のコレクションをデータベース・キャラクタ・セットのデータに変換して戻します。

構文

```
Function nls_convert_from_utf8
(
data_utf8 IN STRING_COLLECTION
)
RETURN STRING_COLLECTION;
```

パラメータ

表 8-100 nls_convert_from_utf8 のパラメータ

パラメータ	説明
data_utf8	UTF8 キャラクタ・セットのデータを含む文字列のコレクションです。

戻り値

表 8-101 nls_convert_from_utf8 の戻り値

値	説明
VARCHAR2	データベース・キャラクタ・セットのデータを含む文字列のコレクションです。

使用方法

DBMS_LDAP パッケージのファンクションは、UTF8_CONVERSION パッケージ変数に FALSE が設定されている場合、UTF8 キャラクタ・セット・データを戻します。この場合、nls_convert_from_utf8() ファンクションを使用すると、出力データを UTF8 キャラクタ・セットからデータベース・キャラクタ・セットに変換できます。

DBMS_LDAP パッケージの UTF8_CONVERSION パッケージ変数に TRUE が設定されている場合、DBMS_LDAP パッケージのファンクションは、データベース・キャラクタ・セット・データを戻します。

関連項目

DBMS_LDAP.nls_convert_to_utf8()、DBMS_LDAP.nls_get_dbcharset_name()

nls_get_dbcharset_name ファンクション

データベース・キャラクタ・セット名を含む文字列を戻します。

構文

```
Function nls_get_dbcharset_name

RETURN VARCHAR2;
```

パラメータ

なし

戻り値

表 8-102 nls_get_dbcharset_name の戻り値

値	説明
VARCHAR2	データベース・キャラクタ・セット名を含む文字列です。

関連項目

DBMS_LDAP.nls_convert_to_utf8()、DBMS_LDAP.nls_convert_from_utf8()

9

DBMS_LDAP_UTL PL/SQL リファレンス

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 パッケージに関するリファレンス情報を示します。この章では、次の項目について説明します。

サブプログラムの概要

表 9-1 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ライバを取得します。
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 9-2 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 9-3 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 9-4 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。
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。

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 9-5 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。
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、ルートの 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。

表 9-5 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
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。
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。

表 9-5 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_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。
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_userr() 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 9-6 DBMS_LDAP_UTL のデータ型

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。

ユーザー関連サブプログラム

ユーザーは DBMS_LDAP_UTL.HANDLE データ型を使用して表現されます。適切なサブスクライバ・ハンドルの作成に加え、識別名、GUID または単純な名前を使用してユーザー・ハンドルを作成できます。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・ハンドルの作成例を示します。

```
  retval := DBMS_LDAP_UTL.create_user_handle(
user_handle,
  DBMS_LDAP_UTL.TYPE_DN,
          "cn=user1,cn=users,o=acme,dc=com"
          );
```

ユーザー・ハンドルは、適切なサブスクライバ・ハンドルに関連付ける必要があります。たとえば、o=acme,dc=com を表すサブスクライバ・ハンドル *subscriber_handle* の場合は、次の方法で関連付けることができます。

```
  retval := DBMS_LDAP_UTL.set_user_handle_properties(
          user_handle,
          DBMS_LDAP_UTL.SUBSCRIBER_HANDLE,
          subscriber_handle
          );
```

一部の共通ユーザー・ハンドルの使用には、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。

```
retval := DBMS_LDAP_UTL.authenticate_user(
      my_session,
      user_handle,
      DBMS_LDAP_UTL.AUTH_SIMPLE,
        "welcome",
      NULL
      );
```

この例では、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。

```
--    my_attrs is of type DBMS_LDAP.STRING_COLLECTION
      my_attrs(1) := 'telephonenumber';
      retval := DBMS_LDAP_UTL.get_user_properties(
my_session,
my_attrs,
DBMS_LDAP_UTL.ENTRY_PROPERTIES,
my_pset_coll
);
```

関連項目： ユーザー・ハンドルのサンプルは、B-12 ページの「DBMS_LDAP_UTL サンプル・コード」を参照してください。

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 9-7 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
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 9-8 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。

表 9-8 AUTHENTICATE_USER ファンクションの戻り値（続き）

値	説明
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。
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 9-9 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。 - 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 9-10 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 9-11 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 9-12 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、サブスクライバ・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 9-13 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 9-14 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。

表 9-14 GET_USER_PROPERTIES ファンクションの戻り値（続き）

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ライバ・ハンドルの設定が必要です。

このファンクションは、NULL のサブスクライバ・ハンドルをデフォルト・サブスクライバとして識別しません。デフォルト・サブスクライバは、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、サブスクライバ・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。

- DBMS_LDAP_UTL.TYPE_GUID

- DBMS_LDAP_UTL.TYPE_DN

サブスクライバ・ハンドルが設定されている場合、サブスクライバ・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 9-15 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 9-16 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。
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,
ptype IN PLS_INTEGER,
filter IN VARCHAR2,
rep_pset_coll OUT PROPERTY_SET_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 9-17 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_RAD
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 9-18 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。
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 9-19 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 9-20 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 9-21 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 9-22 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 9-23 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 9-24 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
(
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 9-25 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 9-26 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

グループは、DBMS_LDAP_UTL.HANDLE データ型を使用して表されます。グループ・ハンドルは、有効なグループ・エントリを表します。適切なサブスクライバ・ハンドルの作成に加え、識別名、GUID または単純な名前を使用してグループ・ハンドルを作成できます。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グループの識別に使用されます。次に、グループ・ハンドルの作成例を示します。

```
retval := DBMS_LDAP_UTL.create_group_handle(
group_handle,
DBMS_LDAP_UTL.TYPE_DN,
"cn=group1,cn=Groups,o=acme,dc=com"
);
```

このグループ・ハンドルを、適切なサブスクライバ・ハンドルに関連付ける必要があります。たとえば、o=acme,dc=com を表すサブスクライバ・ハンドル *subscriber_handle* の場合は、次の方法で関連付けることができます。

```
retval := DBMS_LDAP_UTL.set_group_handle_properties(
group_handle,
DBMS_LDAP_UTL.SUBSCRIBER_HANDLE,
subscriber_handle
);
```

グループ・ハンドルの使用例としては、グループ・プロパティの取得があります。次に例を示します。

```
my_attrs is of type DBMS_LDAP.STRING_COLLECTION
my_attrs(1) := 'uniquemember';
retval := DBMS_LDAP_UTL.get_group_properties(
my_session,
my_attrs,
DBMS_LDAP_UTL.ENTRY_PROPERTIES,
my_pset_coll
);
```

グループ関連サブプログラムは、メンバーシップに関連する機能もサポートしています。DBMS_LDAP_UTL.check_group_membership() ファンクションを使用すると、あるユーザー・ハンドルがグループのダイレクト・メンバーであるか、ネストされたメンバーであるかを確認できます。次に例を示します。

```
retval := DBMS_LDAP_UTL.check_group_membership(
session,
user_handle,
group_handle,
DBMS_LDAP_UTL.DIRECT_MEMBERSHIP
```

また、DBMS_LDAP_UTL.get_group_membership() ファンクションを使用して、特定のグループ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。

```
my_attrs is of type DBMS_LDAP.STRING_COLLECTION
my_attrs(1) := 'cn';
retval := DBMS_LDAP_UTL.get_group_membership(
my_session,
user_handle,
DBMS_LDAP_UTL.DIRECT_MEMBERSHIP,
my_attrs
my_pset_coll
);
```

関連項目： グループ・ハンドルの詳細な使用例は、B-25 ページの「例：グループ関連ファンクション」を参照してください。

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 9-27 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 9-28 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 9-29 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 9-30 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、サブスクライバ・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 9-31 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 9-32 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。

表 9-32 GET_GROUP_PROPERTIES ファンクションの戻り値（続き）

値	説明
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ライバ・ハンドルの設定が必要です。

このファンクションは、NULL のサブスクライバ・ハンドルをデフォルト・サブスクライバとして識別しません。デフォルト・サブスクライバは、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。

- DBMS_LDAP_UTL.TYPE_GUID

- DBMS_LDAP_UTL.TYPE_DN

サブスクライバ・ハンドルが設定されている場合、サブスクライバ・ハンドルは無視されます。

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 9-33 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 9-34 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。

表 9-34 GET_GROUP_DN ファンクションの戻り値（続き）

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()

サブスクライバ関連サブプログラム

サブスクライバは、dbms_ldap_utl.handle データ型を使用して表されます。サブスクライバ・ハンドルは、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、ルートの Oracle コンテキストの追加情報がサブスクライバの識別に使用されます。次に、サブスクライバ・ハンドルの作成例を示します。

```
retval := DBMS_LDAP_UTL.create_subscriber_handle(
        subscriber_handle,
        DBMS_LDAP_UTL.TYPE_DN,
        "o=acme,dc=com"
        );
```

subscriber_handle は、その識別名である o=oracle,dc=com によって作成されます。

サブスクライバ・ハンドルの共通の使用例としては、サブスクライバ・プロパティの取得があります。次に例を示します。

```
my_attrs is of type DBMS_LDAP.STRING_COLLECTION
       my_attrs(1) := 'orclguid';
       retval := DBMS_LDAP_UTL.get_subscriber_properties(
my_session,
my_attrs,
DBMS_LDAP_UTL.ENTRY_PROPERTIES,
my_pset_coll
);
```

関連項目： サブスクライバ・ハンドルのサンプルは、B-12 ページの「DBMS_LDAP_UTL サンプル・コード」を参照してください。

create_subscriber_handle ファンクション

create_subscriber_handle() は、サブスクライバ・ハンドルを作成するファンクションです。

構文

```
FUNCTION create_subscriber_handle
(
ld IN SESSION,
subscriber_hd OUT HANDLE,
subscriber_type IN PLS_INTEGER,
subscriber_id IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 9-35 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 9-36 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 9-37 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。

表 9-37 GET_SUBSCRIBER_PROPERTIES ファンクションのパラメータ（続き）

パラメータ名	パラメータ・タイプ	パラメータの説明
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_COLLECTION	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 9-38 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 9-39 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 9-40 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。

表 9-40 GET_SUBSCRIBER_DN ファンクションの戻り値（続き）

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()

get_subscriber_ext_properties ファンクション

get_subscriber_ext_properties() は、拡張されたサブスクライバのプロパティを取得するファンクションです。現在、このファンクションは、サブスクライバ全体のデフォルトのリソース・アクセス記述子を取得するために使用されます。

構文

```
FUNCTION get_subscriber_ext_properties
(
ld IN SESSION,
subscriber_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 9-41 GET_SUBSCRIBER_EXT_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.DEFAULT_RAD_PROPERTIES
filter	VARCHAR2	ファンクションで戻されたサブスクライバ・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サブスクライバの詳細です。

戻り値

表 9-42 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.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.get_subscriber_properties()

プロパティ関連サブプログラム

ユーザー関連、サブスクライバ関連およびグループ関連のサブプログラムの多くは、結果を表す 1 つ以上の LDAP エントリのコレクションである DBMS_LDAP_UTL.PROPERTY_SET_COLLECTION を戻します。これらの各エントリは、DBMS_LDAP_UTL.PROPERTY_SET で表されます。PROPERTY_SET には、属性（つまりプロパティ）とその値が含まれている場合があります。次に、DBMS_LDAP_UTL.PROPERTY_SET_COLLECTION からプロパティを取得する例を示します。

```
my_attrs is of type DBMS_LDAP.STRING_COLLECTION
my_attrs(1) := 'cn';

retval := DBMS_LDAP_UTL.get_group_membership(
my_session,
user_handle,
DBMS_LDAP_UTL.DIRECT_MEMBERSHIP,
my_attrs,
my_pset_coll
);

IF my_pset_coll.count > 0 THEN
      FOR i in my_pset_coll.first .. my_pset_coll.last LOOP
--    my_property_names is of type DBMS_LDAP.STRING_COLLECTION
       retval := DBMS_LDAP_UTL.get_property_names(
pset_coll(i),
property_names
       IF my_property_names.count > 0 THEN
           FOR j in my_property_names.first .. my_property_names.last LOOP
             retval := DBMS_LDAP_UTL.get_property_values(
pset_coll(i),
property_names(j),
property_values
             if my_property_values.COUNT > 0 then
              FOR k in my_property_values.FIRST..my_property_values.LAST LOOP
                    DBMS_OUTPUT.PUT_LINE(my_property_names(j)  ||  ': '

||my_property_values(k));
                    END LOOP; -- For each value
             else
```

```
                DBMS_OUTPUT.PUT_LINE('NO VALUES FOR ' || my_property_names(j));
              end if;
            END LOOP; -- For each property name
          END IF; -- IF my_property_names.count > 0
        END LOOP; -- For each propertyset
      END IF; -- If my_pset_coll.count > 0
```

use_handle はユーザー・ハンドルです。my_pset_coll には、user_handle 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 cn を出力します。

> **関連項目：** プロパティ関連サブプログラムの詳細な使用例は、B-18 ページの「例 : プロパティ関連サブプログラム」を参照してください。

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 9-43 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 9-44 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 9-45 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 9-46 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 9-47 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()

表 9-47 GET_PROPERTY_VALUES ファンクションのパラメータ（続き）

パラメータ名	パラメータ・タイプ	パラメータの説明
property_values	STRING_ COLLECTION	プロパティ値（文字列）のリストです。

戻り値

表 9-48 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,
auth_type IN PLS_INTEGER,
property_values OUT BINVAL_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 9-49 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 9-50 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ー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 9-51 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,
)
RETURN PLS_INTEGER;
```

パラメータ

表 9-52 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 9-53 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。
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 9-54 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 9-55 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 9-56 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 9-57 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 9-58 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 9-59 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 9-60 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。
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、ルートの 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。

表 9-60 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
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。
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。

表 9-60 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_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。
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_userr() 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 9-61 DBMS_LDAP_UTL のデータ型

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。

10

DAS_URL インタフェース・リファレンス

DAS_URL サービス・インタフェースに対する Oracle の拡張機能について説明します。次の項目について説明します。

- Oracle Delegated Administration Services ユニットおよび対応するディレクトリ・エントリ
- DAS ユニットおよび対応する URL パラメータ
- DAS URL API のパラメータの説明
- ユーザーまたはグループの LOV アクセス

Oracle Delegated Administration Services ユニットおよび対応するディレクトリ・エントリ

表 10-1 に、各 Oracle Delegated Administration Services ユニットおよび相対 URL を格納する Oracle Internet Directory 内の対応するエントリのリストを示します。

表 10-1 サービス・ユニットおよび対応するエントリ

サービス・ユニット	エントリ
ユーザーの作成	cn=CreateUser,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContext
ユーザーの編集	cn=EditUser,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContext
ユーザーの編集（GUID がパラメータとして渡される場合）	cn=EditUserGivenGUID,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザーの削除	cn=DeleteUser,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContext
ユーザーの削除（削除されるユーザーの GUID がパラメータとして渡される場合）	cn=DeleteUserGivenGUID,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループの作成	cn=CreateGroup,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループの編集	cn=EditGroup,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContext
グループの編集（GUID がパラメータから渡される場合）	cn=EditGroupGivenGUID,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループの削除	cn=DeleteGroup,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループの削除（GUID がパラメータから渡される場合）	cn=DeleteGroupGivenGUID,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext

表 10-1 サービス・ユニットおよび対応するエントリ（続き）

サービス・ユニット	エントリ
ユーザーへの権限の割当て	cn=UserPrivilege,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザーへの権限の割当て（GUID がパラメータから渡される場合）	cn=UserPrivilegeGivenGUID,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループへの権限の割当て	cn=GroupPrivilege,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループへの権限の割当て（指定された GUID を使用する場合）	cn=GroupPrivilegeGivenGUID,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザーのアカウント情報およびプロファイルの表示	cn=AccountInfo,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザーのアカウント情報およびプロファイルの編集	cn=Edit My Profile,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
パスワードの変更	cn=PasswordChange,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザーの検索	cn=UserSearch,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループの検索	cn=GroupSearch,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザー LOV の検索	cn=UserLOV,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループ LOV の検索	cn=GroupLOV,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
EUS コンソール	cn=EUS Console,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContext"
コンソールの委任	cn=DelegationConsole,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext

DAS ユニットおよび対応する URL パラメータ

次の表に、使用可能なすべての DAS ユニットおよび DAS ユニットに指定可能な URL パラメータのリストを示します。

表 10-2 DAS ユニットおよび対応する URL パラメータ

DAS ユニット	パラメータ	戻り値
ユーザーの作成	homeURL, doneURL, cancelURL, enablePA	returnGUID
ユーザーの編集	homeURL, doneURL	
	cancelURL, enablePA	
EditUserGivenGUID	homeURL, doneURL, cancelURL, enablePA, userGUID	
EditMyProfile	homeURL, doneURL, cancelURL	
コンソールの委任		
DeleteUser	homeURL, doneURL, cancelURL	
DeleteUserGivenGUID	homeURL, doneURL, cancelURL, userGUID	
UserPrivilege	homeURL, doneURL, cancelURL	
UserPrivilegeGivenGUID	homeURL, doneURL, cancelURL, userGUID	
CreateGroup	homeURL, doneURL, cancelURL, enablePA, userGUID	returnGUID
EditGroup	homeURL, doneURL, cancelURL, enablePA	
EditGroupGivenGUID	homeURL, doneURL, cancelURLenablePA, groupGUID	
DeleteGroup	homeURL, doneURL, cancelURL	

表 10-2 DAS ユニットおよび対応する URL パラメータ（続き）

DAS ユニット	パラメータ	戻り値
DeleteGroupGivenGUID	homeURL, doneURL, cancelURL, groupGUID	
GroupPrivilege	homeURL, doneURL, cancelURL	
GroupPrivilegeGivenGUID	homeURL, doneURL, cancelURL, groupGUID	
AccountInfo	homeURL, doneURL, cancelURL	
PasswordChange	homeURL, doneURL, cancelURL	
UserSearch	homeURL, doneURLm, cancelURL	
GroupSearch	homeURL, doneURL, cancelURL	
UserLOV	base, cfilter, title, dasdomain	
GroupLOV	otype, base, cfilter, title, dasdomain	

DAS URL API のパラメータの説明

次の表に、DAS ユニットで使用するパラメータを示します。

表 10-3 DAS URL のパラメータの説明

パラメータ	説明
homeURL	Home グローバル・ボタンにリンクされる URL です。コール元のアプリケーションがこの値を指定すると、Home ボタンをクリックして、このパラメータで指定された URL へ、DAS ユニットをリダイレクトできます。
doneURL	この URL は、各操作の最後に DAS ページをリダイレクトするために、DAS が使用します。ユーザーの作成の場合、ユーザーが作成された後、「OK」をクリックすると URL がこの位置にリダイレクトされます。このため、ユーザーのナビゲーション操作がスムーズになります。
cancelURL	この URL は、DAS ユニットに表示されるすべての「取消」ボタンにリンクされます。ユーザーが「取消」をクリックすると、常にそのページがこのパラメータで指定した URL にリダイレクトされます。

表 10-3 DAS URL のパラメータの説明（続き）

パラメータ	説明
enablePA	このパラメータは、true または false のブール値をとります。これは、ユーザーまたはグループ操作での権限の割当てセクションを使用可能にします。Create User ページで enablePA に true が渡されると、Assign Privileges to User セクションも Create User ページに表示されます。
userGUID	これは、編集または削除されるユーザーの GUID です。orclguid 属性に対応します。このパラメータを指定すると、editUser または deleteUser ユニットのユーザーの検索の手順がスキップされます。
GroupGUID	これは、編集または削除されるグループの GUID です。orclguid 属性に対応します。このパラメータを指定すると、editGroup または deleteGroup ユニットのグループの検索の手順がスキップされます。
parentDN	CreateGroup にこのパラメータを指定すると、このコンテナ下にグループが作成されます。このパラメータを指定しない場合、グループはデフォルトでグループの検索ベースに作成されます。
base	このパラメータは、検索操作での検索ベースを指定します。
cfilter	このパラメータは、検索に使用するフィルタを指定します。このフィルタは LDAP 準拠です。
title	このパラメータは、Search および Select LOV ページに表示されるタイトルを指定します。
otype	このパラメータは、検索に使用するオブジェクト・タイプを指定します。サポートされている値は、Select、Edit および Assign です。
returnGUID	このパラメータは、作成操作で doneURL に追加されます。値は、新しいオブジェクトの orclguid です。
dasdomain	このパラメータは、ブラウザが Internet Explorer で、コール元の URL と DAS URL が異なるホストで同じドメインである場合にのみ必要です。たとえば、us.oracle.com です。コール元のアプリケーションでも、formload に `document.domain` パラメータを設定する必要があることに注意してください。詳細は、Microsoft の次のサポート・サイトを参照してください。 `http://support.microsoft.com/`

ユーザーまたはグループの LOV アクセス

DAS では、ユーザーまたはグループの検索ページを値リスト（LOV）と呼びます。

DAS は JavaScript を使用して LOV にアクセスし、LOV のコール元ウィンドウと DAS の LOV ページ間で値を渡します。JavaScript にはセキュリティ上の制限があるため、ドメイン間でデータを渡すことはできません。この制限により、同じドメイン内のページのみが DAS LOV ユニットにアクセスできます。

次の例では、DAS ユーザーの LOV を起動する単純な HTML ファイルを示します。JavaScript ファンクションは、UIX が公開している `MarlinCore.js` ファイルから採用したものです。アプリケーションは、独自の JavaScript プロシージャまたは UIX ライブラリのユーティリティを使用します。

```
<html>
 <head>
  <title>test</title>
 </head>
 <script src="/cabo/jsLibs/MarlinCore.js">
 </script>
 <body>
  <form name="form1" method="GET" action="test.html">
    <script language="javascript">
       function doSelect(lovWindow, event)
       {
            var value = lovWindow.userName;
             if (value!=(void 0)){
                 document.form1.lov1.value = lovWindow.userName;
                 document.form1.lov2.value = lovWindow.userGuid
             }
        }
    </script>
 <table border="0" cellspacing="0" cellpadding="0">
<tr>
  <td align="right" nowrap>
   <span> 
     <label for="M__Id2500">Manager</label>
    </span>
  </td>
<td nowrap>
  <span>
    <input id="M__Id2500" name="lov1" size="25" type="text">
     <a onclick="openWindow(top,
'/oiddas/cabo/jsps/frameRedirect.jsp?redirect=http://sneaker.us.oracle.com:7777/oidd
as/ui/oracle/ldap/das/search/LOVUserSearch?title=User',
'lovWindow',{width:600,height:600},
false, 'dialog',doSelect);
return false;"
```

```
href="">
                          <img src="/oiddas/oracle/ldap/das/Images/c-lov.gif"
   alt="Testing lov" border="0" align="absmiddle">
     </a>
   </span>
  </td>
 </tr>
 <tr>
     <td align="right" nowrap>
       <span> 
            <label for="M__Id2501">Guid</label>
       </span>
     </td>
     <td>
        <input id="M__Id2501" name="lov2" size="25" type="text">
      </td>
  </tr>
 </table>
 </form>
 </body>
</html>
```

UIX openWindow ファンクションの詳細は、次のサイトを参照してください。

http://cabo/cabo/marlin/docs/windowJS.html

11

プロビジョニング統合 API リファレンス

この章では、Oracle Directory Provisioning Integration Service の登録 API に関するリファレンス情報を示します。この章では、次の項目について説明します。

- プロビジョニング・ファイルおよびインタフェースのバージョニング
- 拡張可能なイベント定義の構成
- 着信イベントおよび発信イベント
- PL/SQL 双方向インタフェース（バージョン 2.0）
- プロビジョニング・イベント・インタフェース（バージョン 1.1）

プロビジョニング・ファイルおよびインタフェースのバージョニング

Oracle Internet Directory リリース 9.0.2 では、デフォルトのインタフェースはバージョン 1.1 です。リリース 9.0.4 では、デフォルトのインタフェースはバージョン 2.0 です。ただし、管理者は以前のインタフェースを維持するために、バージョン 1.1 に戻すことができます。

拡張可能なイベント定義の構成

この機能は、発信イベントでのみ使用可能です。プロビジョニング統合サービスは Oracle Internet Directory での変更を解析し、アプリケーションに対して適切なイベントが生成および伝播されるかどうかを判断できるように、この機能によって、実行時に新しいイベントを定義できます。次に示すイベントは、インストール時に構成のみが行われるイベントです。

イベント定義（エントリ）は、次の属性で構成されます。

- イベント・オブジェクト型（`orclODIPProvEventObjectType`）: これはイベントが関連付けられているオブジェクトの型を指定します。オブジェクトは、USER、GROUP、IDENTITY などになります。
- LDAP 変更型（`orclODIPProvEventChangeType`）: これは、すべての LDAP 操作でこの型のオブジェクトに対して生成できるイベントを示します。イベントは、ADD、MODIFY、DELETE などになります。
- イベント基準（`orclODIPProvEventCriteria`）: 特定のオブジェクト型になる LDAP エントリを指定する追加の選択基準です。たとえば、`Objectclass=orclUserV2` の場合、この基準を満たす LDAP エントリはこのオブジェクト型として識別され、このエントリを変更すると適切なイベントが生成されます。

前述の属性を保持するオブジェクト・クラスは、`orclODIPProvEventTypeConfig` です。すべてのイベント型構成を格納するために、コンテナ `cn=ProvisioningEventTypeConfig,cn=odi,cn=oracle internet directory` が使用されます。

表 11-1 のイベント定義は、インストールの一部として事前定義されています。

表 11-1 事前定義済のイベント定義

イベント・オブジェクト型	LDAP 変更型	イベント基準
ENTRY	ADD、MODIFY、DELETE	OBJECTCLASS=*
USER	ADD、MODIFY、DELETE	OBJECTCLASS=interorgperson OBJECTCLASS=orcluserv2
IDENTITY	ADD、MODIFY、DELETE	OBJECTCLASS=interorgperson OBJECTCLASS=orcluserv2
GROUP	ADD、MODIFY、DELETE	OBJECTCLASS=orclgroup OBJECTCLASS=groupofuniquenames
SUBSCRPTION	ADD、MODIFY、DELETE	OBJECTCLASS=orclservicerecepient
SUBSCRIBER	ADD、DELETE、MODIFY	OBJECTCLASS=orclsubscriber

すべてのイベント定義構成を格納するために、コンテナ cn=ProvisioningEventTypeConfig,cn=odi,cn=oracle internet directory が使用されます。事前定義されたイベント定義の LDAP 構成は次のとおりです。

```
dn: orclODIPProvEventObjectType=ENTRY,cn=ProvisioningEventTypeConfig,cn=odi,
cn=oracle internet directory
orclODIPProvEventObjectType: ENTRY
orclODIPProvEventLDAPChangeType: Add
orclODIPProvEventLDAPChangeType: Modify
orclODIPProvEventLDAPChangeType: Delete
orclODIPProvEventCriteria: objectclass=*
objectclass: orclODIPProvEventTypeConfig

dn: orclODIPProvEventObjectType=USER,cn=ProvisioningEventTypeConfig,cn=odi,cn=oracle
internet directory
orclODIPProvEventObjectType: USER
orclODIPProvEventLDAPChangeType: Add
orclODIPProvEventLDAPChangeType: Modify
orclODIPProvEventLDAPChangeType: Delete
orclODIPProvEventCriteria: objectclass=InetOrgPerson
orclODIPProvEventCriteria: objectclass=orcluserv2
objectclass: orclODIPProvEventTypeConfig
```

```
dn: orclODIPProvEventObjectType=IDENTITY,cn=ProvisioningEventTypeConfig,cn=odi,
cn=oracle internet directory
orclODIPProvEventObjectType: IDENTITY
orclODIPProvEventLDAPChangeType: Add
orclODIPProvEventLDAPChangeType: Modify
orclODIPProvEventLDAPChangeType: Delete
orclODIPProvEventCriteria: objectclass=inetorgperson
orclODIPProvEventCriteria: objectclass=orcluserv2
objectclass: orclODIPProvEventTypeConfig

dn: orclODIPProvEventObjectType=GROUP,cn=ProvisioningEventTypeConfig,cn=odi,
cn=oracle internet directory
orclODIPProvEventObjectType: GROUP
orclODIPProvEventLDAPChangeType: Add
orclODIPProvEventLDAPChangeType: Modify
orclODIPProvEventLDAPChangeType: Delete
orclODIPProvEventCriteria: objectclass=orclgroup
orclODIPProvEventCriteria: objectclass=groupofuniquenames
objectclass: orclODIPProvEventTypeConfig

dn: orclODIPProvEventObjectType=SUBSCRIPTION,cn=ProvisioningEventTypeConfig,cn=odi,
cn=oracle internet directory
orclODIPProvEventObjectType: SUBSCRIPTION
orclODIPProvEventLDAPChangeType: Add
orclODIPProvEventLDAPChangeType: Modify
orclODIPProvEventLDAPChangeType: Delete
orclODIPProvEventCriteria: objectclass=orclservicerecepient
objectclass: orclODIPProvEventTypeConfig

dn: orclODIPProvEventObjectType=SUBSCRIBER,cn=ProvisioningEventTypeConfig,cn=odi,
cn=oracle internet directory
orclODIPProvEventObjectType: SUBSCRIBER
orclODIPProvEventLDAPChangeType: Add
orclODIPProvEventLDAPChangeType: Modify
orclODIPProvEventLDAPChangeType: Delete
orclODIPProvEventCriteria: objectclass=orclsubscriber
objectclass: orclODIPProvEventTypeConfig
```

オブジェクト型 XYZ（オブジェクト・クラス objXYZ で修飾）の新しいイベントを定義するには、OID に次のエントリを作成します。DIP サーバーはこの新しいイベント定義を認識し、このイベントにサブスクライブするアプリケーションに対し、必要に応じてイベントを伝播します。

```
dn: orclODIPProvEventObjectType=XYZ,cn=ProvisioningEventTypeConfig,cn=odi, cn=oracle
internet directory
orclODIPProvEventObjectType: XYZ
orclODIPProvEventLDAPChangeType: Add
orclODIPProvEventLDAPChangeType: Modify
orclODIPProvEventLDAPChangeType: Delete
orclODIPProvEventCriteria: objectclass=objXYZ
objectclass: orclODIPProvEventTypeConfig
```

これは、オブジェクト・クラス objXYZ を持つ LDAP エントリが追加、変更または削除された場合、DIP によって XYZ_ADD/XYZ_MODIFY/XYZ_DELETE イベントがそれぞれ関連するアプリケーションに伝播されることを意味します。

着信イベントおよび発信イベント

アプリケーションは、イベントのサプライヤおよびコンシューマとして登録できます。11-6 ページの表 11-2 に、プロビジョニング・サブスクリプション・プロファイルの属性を示します。

表 11-2 プロビジョニング・サブスクリプション・プロファイルの属性

属性	説明
EventSubscriptions	発信イベントのみ。（複数の値を取ります） これは以前のリリースと同じです。DIP がこのアプリケーションに通知を送信する必要があるイベントです。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 >:DELETEGROUP:< 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）。
MappingRules	着信イベントのみ。（複数の値を取ります。）このリリースの新機能です。アプリケーションおよび修飾フィルタ条件から取得したオブジェクトの型をマップし、このイベントの対象のドメインを判断します。 OBJECT_TYPE: フィルタ条件 : 対象のドメイン 複数の値を指定できます。 たとえば、次のようにします。 ■ EMP::cn=users,dc=acme,dc=com これは、取得したオブジェクト型が EMP の場合、イベントの対象のドメインは cn=users,dc=acme,dc=com であることを示します。 ■ EMP:l=AMERICA:l=AMER,cn=users,dc=acme,dc=com これは、取得したオブジェクト型が EMP、イベントの属性が l（Locality）、およびイベント値が AMERICA の場合、イベントの対象のドメインは l=AMER,cn=users,dc=acme,dc=com であることを示します。
permittedOperations	着信イベントのみ。（複数の値を取ります。） このリリースの新機能です。 アプリケーションに権限が付与されたイベントの型を定義し、プロビジョニング統合サービスに送信します。 この文字列の書式は次のとおりです。 Event_Object: 影響を受けるドメイン : 操作（属性 ,…） たとえば、次のようにします。 ■ IDENTITY:cn=users,dc=acme,dc=com:ADD(*) これは、IDENTITY_ADD イベントおよびすべての属性が特定のドメインに許可されることを示します。 ■ IDENTITY:cn=users,dc=acme,dc=com:MODIFY(cn,sn.mail,telephonenumber) これは、IDENTITY_MODIFY がこのリスト内の属性にのみ許可されることを示します。その他の属性は暗黙的に無視されます。

PL/SQL 双方向インタフェース（バージョン 2.0）

PL/SQL コールバック・インタフェースでは、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 パッケージ仕様部でパッケージを実装します。

```
DROP TYPE LDAP_EVENT;
DROP TYPE LDAP_EVENT_STATUS;
DROP TYPE LDAP_ATTR_LIST;
DROP TYPE LDAP_ATTR;
--------------------------------------------------------------------------------
-- Name: LDAP_ATTR
-- Data Type: OBJECT

DESCRIPTION: This structure contains details regarding an attribute. A list of one
or more of this object is passed in any event.
--------------------------------------------------------------------------------
---------------
CREATE TYPE LDAP_ATTR AS OBJECT (
     attr_name        VARCHAR2(256),
     attr_value       VARCHAR2(4000),
     attr_bvalue      RAW(2048),
     attr_value_len   INTEGER,
     attr_type        INTEGER ,
     attr_mod_op      INTEGER
);

GRANT EXECUTE ON LDAP_ATTR to public;

CREATE TYPE LDAP_ATTR_LIST AS TABLE OF LDAP_ATTR;
/
GRANT EXECUTE ON LDAP_ATTR_LIST to public;

--------------------------------------------------------------------------------
---------------
-- Name: LDAP_EVENT
-- Data Type: OBJECT
-- DESCRIPTION: This structure contains event information plus the attribute
-- list
--------------------------------------------------------------------------------
---------------
```

```
CREATE TYPE LDAP_EVENT AS OBJECT (
          event_type  VARCHAR2(32),
          event_id    VARCHAR2(32),
          event_src   VARCHAR2(1024),
          event_time  VARCHAR2(32),
          object_name VARCHAR2(1024),
          object_type VARCHAR2(32),
          object_guid VARCHAR2(32),
          object_dn   VARCHAR2(1024),
          profile_id  VARCHAR2(1024),
          attr_list   LDAP_ATTR_LIST ) ;
/

GRANT EXECUTE ON LDAP_EVENT to public;

--------------------------------------------------------------------------------
---------------
-- Name: LDAP_EVENT_STATUS
-- Data Type: OBJECT
-- DESCRIPTION: This structure contains information that is sent by the consumer of
an
                                event to the supplier in response to the actual
event.

--------------------------------------------------------------------------------
---------------

CREATE TYPE LDAP_EVENT_STATUS AS OBJECT (
          event_id    VARCHAR2(32),
          orclguid  VARCHAR(32),
          error_code  INTEGER,
          error_String VARCHAR2(1024),
          error_disposition VARCHAR2(32)) ;
/

GRANT EXECUTE ON LDAP_EVENT_STATUS to public;
```

プロビジョニング・イベント・インタフェース（バージョン 1.1）

4-19 ページの「プロビジョニング統合に関するタスクの開発」で説明したように、Oracle Directory Provisioning Integration Service によって生成されたイベントをコンシュームするためのロジックを開発する必要があります。PL/SQL コールバック・インタフェースの場合は、Oracle Directory 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 パッケージ仕様部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_NTFY
--  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_NTFY AS

--
-- LDAP_NTFY 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;
```

```
-------------------------------------------------------------------------------
---------------
-- Name: LDAP_NTFY
-- DESCRIPTION: This is the interface to be implemented by Provisioning System
-- clients to send/receive information to/from 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_NTFY AS
```

事前定義されるイベント型

```
ENTRY_ADD            CONSTANT VARCHAR2 (32)  := 'ENTRY_ADD';
ENTRY_DELETE      CONSTANT VARCHAR2 (32)  := 'ENTRY_DELETE';
ENTRY_MODIFY     CONSTANT VARCHAR2 (32) := 'ENTRY_MODIFY';

USER_ADD               CONSTANT VARCHAR2 (32)  := 'USER_ADD';
USER_DELETE          CONSTANT VARCHAR2 (32) := 'USER_DELETE';
USER_MODIFY CONSTANT VARCHAR2(32) := 'USER_MODIFY';

IDENTITY_ADD          CONSTANT VARCHAR2 (32) := 'IDENTITY_ADD';
IDENTITY_DELETE   CONSTANT VARCHAR2 (32) := 'IDENTITY_DELETE';
IDENTITY_MODIFY  CONSTANT VARCHAR2 (32) := 'IDENTITY_MODIFY';

GROUP_ADD               CONSTANT VARCHAR2 (32) := 'GROUP_ADD';
GROUP_DELETE         CONSTANT VARCHAR2 (32) := 'GROUP_DELETE';
GROUP_MODIFY         CONSTANT VARCHAR2 (32) := 'GROUP_MODIFY';

SUBSCRIPTION_ADD         CONSTANT VARCHAR2(32) := 'SUBSCRIPTION_ADD';
SUBSCRIPTION_DELETE  CONSTANT VARCHAR2(32) := 'SUBSCRIPTION_DELETE';
SUBSCRIPTION_MODI       CONSTANT VARCHAR2(32) := 'SUBSCRIPTION_MODIFY';

SUBSCRIBER_ADD          CONSTANT VARCHAR2(32) := 'SUBSCRIBER_ADD';
SUBSCRIBER_DELETE    CONSTANT VARCHAR2(32) := 'SUBSCRIBER_DELETE';
SUBSCRIBER_MODIFY   CONSTANT VARCHAR2(32) := 'SUBSCRIBER_MODIFY';
```

属性の型

```
ATTR_TYPE_STRING             CONSTANT NUMBER  := 0;
ATTR_TYPE_BINARY             CONSTANT NUMBER  := 1;
ATTR_TYPE_ENCRYPTED_STRING   CONSTANT NUMBER  := 2;
```

属性の変更型

```
MOD_ADD              CONSTANT NUMBER  := 0;
MOD_DELETE           CONSTANT NUMBER  := 1;
MOD_REPLACE          CONSTANT NUMBER  := 2;
```

イベント処理の定数

```
EVENT_SUCCESS             CONSTANT VARCHAR2(32)  := 'EVENT_SUCCESS';
EVENT_FAILURE             CONSTANT VARCHAR2(32)  := 'EVENT_FAILURE';
EVENT_RESEND              CONSTANT VARCHAR2(32)  := 'EVENT_RESEND';
```

コールバック

通知イベントを送受信するために、Oracle Directory 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。オブジェクトのイベントを転送する際は、関連する属性が他の詳細とともに転送されます。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。

GetAppEvent()

Oracle Directory Integration Server は、リモート・データベースでこの API を起動します。イベントに応答するのはアプリケーションです。Oracle Directory Integration and Provisioning Platform は、イベントを取得した後、そのイベントを処理し、`PutAppEventStatus()` コールバックを使用して、ステータスを戻します。`GetAppEvent()` の戻り値は、イベントが戻されたかどうかを示します。

```
FUNCTION GetAppEvent (event OUT LDAP_EVENT)
RETURN NUMBER;

-- Return CONSTANTS
EVENT_FOUND          CONSTANT NUMBER  := 0;
EVENT_NOT_FOUND      CONSTANT NUMBER  := 1403;
```

プロビジョニング・サーバーがイベントを処理できない（LDAP エラーが発生した）場合、プロビジョニング・サーバーは `EVENT_RESEND` に応答し、後で `GetAppEvent()` が再度起動されたときにアプリケーションがそのイベントを再送します。

プロビジョニング・サーバーはイベントを処理できるが、そのイベントが処理できないイベントである（たとえば、変更、サブスクライブまたは削除されるユーザーが存在しない）場合、プロビジョニング・サーバーは `EVENT_ERROR` に応答し、不具合があることをアプリケーションに示します。イベントの再送は不要です。このイベントはアプリケーションによって処理されます。

前述の EVENT_RESEND および EVENT_ERROR に相違はありません。EVENT_RESEND の場合、イベントを適用することはできたがサーバーではできなかったことを意味します。そのため、再度イベントを取得すると正常に処理されます。

EVENT_ERROR はディレクトリ操作の実行中にエラーがなかったことを意味します。ただし、その他の理由でイベントは処理されませんでした。

PutAppEventStatus()

Oracle Directory Integration Server は、GetAppEvent() コールバックを使用して受信したイベントを処理した後、リモート・データベースでこのコールバックを起動します。各イベントを受信すると、Oracle Directory Integration Server はイベントを処理した後、イベントのステータスを戻します。

```
PROCEDURE PutAppEventStatus (event_status IN LDAP_EVENT_STATUS);
```

PutOIDEvent()

Oracle Directory Integration Server は、リモート・データベースでこの API を起動します。Oracle Directory Integration Server はこのコールバックを使用して、アプリケーションにイベントを送信します。また、OUT パラメータとしての応答に、ステータス n のイベント・オブジェクトを想定します。有効なイベント・ステータス・オブジェクトが戻されない場合、または RESEND であることが示される場合、Oracle Directory Integration Server はこのイベントを再送します。EVENT_ERROR の場合、サーバーはイベントを再送しません。

```
PROCEDURE PutOIDEvent (event IN LDAP_EVENT, event_status OUT
LDAP_EVENT_STATUS);
END LDAP_NTFY;
/
```

第 III 部

付録

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、次の付録で構成されています。

- 付録 A「LDIF およびコマンドライン・ツールの構文」
- 付録 B「使用例」

A

LDIF およびコマンドライン・ツールの構文

この付録では、**LDAP Data Interchange Format（LDIF）**と LDAP コマンドライン・ツールに関する構文、使用方法および例を紹介します。この項では、次の項目について説明します。

- LDAP Data Interchange Format（LDIF）の構文
- Oracle Internet Directory サーバーの起動、停止、再起動および監視
- エントリおよび属性の管理コマンドライン・ツール構文
- Oracle Directory Integration and Provisioning Platform コマンドライン・ツールの構文

LDAP Data Interchange Format（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_ type:	*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、ある従業員のファイル・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
mail: ssmith@us.Acme.com
telephoneNumber: 69332
photo: /ORACLE_HOME/empdir/photog/ssmith.jpg
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

次の例は、ある組織のファイル・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

次に示すのは、形式化規則のリストです。このリストは、全規則ではありません。

- 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。詳細は、『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。
- 個々のエントリの間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。入力ファイルで使用さ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、入力ファイルで使用しないでください。
- 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、その子エントリの前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 形式化規則の完全なリストについては、xviii ページの「関連ドキュメント」を参照してください。
- 『Oracle Internet Directory 管理者ガイド』の付録 G を参照してください。

Oracle Internet Directory サーバーの起動、停止、再起動および監視

この項では、コマンドライン・ツールを使用して Oracle Internet Directory サーバーを起動、停止、再起動および監視する方法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。

- OID モニター（oidmon）構文
- OID 制御ユーティリティ（oidctl）の構文

OID モニター（oidmon）構文

OID モニターを使用して、ディレクトリ・サーバー・プロセスを開始、監視および終了します。レプリケーション・サーバーをインストールするように選択した場合、レプリケーション・サーバーは OID モニターによって制御されます。ディレクトリ・サーバー・インスタンスを起動または停止するために OID 制御ユーティリティ（OIDCTL）を介してコマンドを発行すると、そのコマンドはこのプロセスによって解析されます。

OID モニターの起動

OID モニターを起動すると、以前停止したすべての Oracle Internet Directory プロセスが再起動されます。

OID モニターを起動する手順は、次のとおりです。

1. 次の環境変数を設定します。
   - *ORACLE_HOME*
   - ORACLE_*SID* または適切な TNS CONNECT 文字列
   - NLS_LANG (*APPROPRIATE_LANGUAGE*.AL32UTF8) インストール時のデフォルトの言語設定は、AMERICAN_AMERICA です。
   - PATH。環境変数 PATH では、UNIX バイナリ・ディレクトリの前に Oracle LDAP バイナリ（*ORACLE_HOME*/bin）を指定します。
2. コマンド・プロンプトで、次のコマンドを入力します。

```
oidmon [connect=connect_string] [host=virtual/host_name][sleep=seconds] start
```

表 A-1 OID モニターを起動するための引数

引数	説明
connect=*connect_string*	接続先データベースの接続文字列を指定します。tnsnames.ora ファイルに設定されているネットワーク・サービス名です。この引数はオプションです。
host=*virtual/host_name*	OID モニターを起動する仮想ホストまたはラック・ノードを指定します。
sleep=*seconds*	OID モニターが、OID 制御ユーティリティからの新規要求、および停止している可能性があるサーバーの再起動要求をチェックするまでの秒数を指定します。デフォルトのスリープ・タイムは 10 秒です。この引数はオプションです。
start	OID モニター・プロセスを開始します。

次に例を示します。

```
oidmon connect=dbs1 sleep=15 start
```

仮想ホスト上で OID モニターを起動する手順は、次のとおりです。

```
oidmon connect=dbsl host=virtual_host start
```

OID モニターの停止

OID モニターを停止すると、他のすべての Oracle Internet Directory プロセスも停止されます。

OID モニター・デーモンを停止するには、コマンド・プロンプトで次のコマンドを入力します。

```
oidmon [connect=connect_string] [host=virtual/host_name] stop
```

表 A-2 OID モニターを停止するための引数

引数	説明
connect=*connect_string*	接続先データベースの接続文字列を指定します。これは、tnsnames.ora ファイルに設定されている接続文字列のセットです。
host=*virtual/host name*	OID モニターを起動する仮想ホストまたはラック・ノードを指定します。
stop	OID モニターのプロセスを停止します。

次に例を示します。

```
oidmon connect=dbs1 stop
```

コールド・フェイルオーバー・クラスタ構成での OID モニターの起動と停止

OID モニターを起動および停止する場合は、host パラメータを使用して仮想ホスト名を指定します。構文は次のとおりです。

```
oidmon [connect=connect_string] host=virtual_host start|stop
```

> **注意：** OIDMON および OIDCTL の両方を使用して、仮想ホストで Oracle Internet Directory サーバーを起動する場合は、仮想ホストとして必ず host 引数を指定してください。
>
> host=*host name* 引数を指定して OID モニターを起動すると、ホスト名が物理ホストの名前と一致しない場合は、OID モニターによって、対象ホストが論理ホストであるとみなされます。OIDCTL を使用してサーバーを停止または起動する場合は、同じホスト名を使用する必要があります。同じホスト名を使用しない場合、OID モニターはサーバーを起動または停止しません。
>
> 物理ホスト名を判断するには、uname コマンドを実行します。

OID 制御ユーティリティ（oidctl）の構文

OID 制御ユーティリティは、ディレクトリ・サーバーの起動と停止に使用するコマンドライン・ツールです。コマンドは、OID モニター・プロセスによって解析され、実行されます。

> **注意：** OID モニターおよび OID 制御ユーティリティを使用せずにディレクトリ・サーバーを起動することも可能ですが、オラクル社ではこれらを使用することをお薦めします。これによって、ディレクトリ・サーバーが予期せずに停止しても、OID モニターが自動的にディレクトリ・サーバーを起動します。

この項では、次の項目について説明します。

- Oracle ディレクトリ・サーバー・インスタンスの起動と停止
- ディレクトリ・サーバー・インスタンスの起動に関するトラブルシューティング
- Oracle ディレクトリ・レプリケーション・サーバー・インスタンスの起動と停止
- Oracle Directory Integration Server の起動
- Oracle Directory Integration Server の停止

- Oracle Internet Directory サーバー・インスタンスの再起動
- 仮想ホストまたはラック・ノードでの Oracle Internet Directory サーバーの起動と停止

Oracle ディレクトリ・サーバー・インスタンスの起動と停止

OID 制御ユーティリティを使用して、Oracle ディレクトリ・サーバー・インスタンスの起動と停止を行います。

Oracle ディレクトリ・サーバー・インスタンスの起動 Oracle ディレクトリ・サーバー・インスタンスを起動する構文は、次のとおりです。

```
oidctl connect=connect_string server=oidldapd instance=server_instance_number
[configset=configset_number] [host=virtual/host_name] [flags=' -p port_number -work
maximum_number_of_worker_threads_per_server -debug debug_level -l change_logging'
-server number_of_server_processes] start
```

表 A-3 OIDCTL を使用してディレクトリ・サーバーを起動するための引数

引数	説明
`-debug debug_level`	Oracle ディレクトリ・サーバー・インスタンス起動中のデバッグ・レベルを指定します。
`-l change_logging`	レプリケーションの変更ログを記録するかどうかを設定します。設定をオフにする場合は、`-l false` を入力します。設定をオンにするには、次のいずれかを実行します。 ■ `-l` フラグを省略します。 ■ `-l` を入力します。 ■ `-l true` を入力します。 `-l false` で、指定したノードに対する変更ログの記録をオフにすると、2 つの問題が発生します。指定したノードから DRG のその他のノードへの更新のレプリケーションが阻止され、アプリケーション・プロビジョニングおよび接続ディレクトリの同期が阻止されます。これは、この 2 つのサービスには、アクティブな変更ログが必要なためです。デフォルトは TRUE で、レプリケーション、プロビジョニングおよび同期を許可します。
`-p port_number`	サーバー・インスタンス起動中のポート番号を指定します。デフォルトのポート番号は 389 です。
`-server number_of_server_processes`	このポートで起動するサーバー・プロセスの数を指定します。

表 A-3 OIDCTL を使用してディレクトリ・サーバーを起動するための引数（続き）

引数	説明
-sport	サーバー・インスタンス起動中に SSL ポート番号を指定します。未設定の場合、デフォルトのポート番号は 636 です。 **関連項目：** ■ 『Oracle Internet Directory 管理者ガイド』の付録 B で orclsslenable 属性の情報を参照してください。 ■ 『Oracle Internet Directory 管理者ガイド』の第 13 章を参照してください。
-work *maximum_number_of_worker_threads_per_server*	このサーバーのワーカー・スレッドの最大数を指定します。
configset=*configset_number*	サーバーの起動に使用される configset の番号。未設定の場合は、デフォルトで configset0 に設定されます。0 ～ 1000 の間の数値を設定してください。
connect=*connect_string*	すでに tnsnames.ora ファイルを構成している場合は、*ORACLE_HOME*/network/admin にある、そのファイルに指定されているネット・サービス名です。
host=*virtual/host_name*	ディレクトリ・サーバーを起動する仮想ホストまたはラック・ノードを指定します。
instance=*server_instance_number*	起動するサーバーのインスタンス番号。1 ～ 1000 の間の数値を設定してください。
server=oidldapd	起動するサーバーの種類（有効な値は OIDLDAPD と OIDREPLD です）。大文字と小文字は区別されません。
start	server 引数で指定したサーバーを起動します。

たとえば、ネット・サービス名が dbs1 で、configset5 を使用し、ポート 12000、デバッグ・レベル 1024、インスタンス番号 3、変更ログ記録なしでディレクトリ・サーバー・インスタンスを起動するには、コマンド・プロンプトで次のように入力します。

```
oidctl connect=dbs1 server=oidldapd instance=3 configset=5 flags='-p 12000
-debug 1024 -l ' start
```

Oracle ディレクトリ・サーバー・インスタンスの起動と停止では、コマンド start または stop 同様に、サーバー名とインスタンス番号が必須です。その他の引数はすべてオプションです。

フラグ引数内のペアのキーワード値はすべて、その間を 1 つの空白で区切る必要があります。

フラグは引用符で囲む必要があります。

configset 識別子が未設定の場合は、デフォルトで 0（configset0）に設定されます。

注意： デフォルト・ポート（無保護使用の場合は 389、保護使用の場合は 636）以外のポートを使用する場合は、Oracle Internet Directory の配置に使用するポートをクライアントに通知する必要があります。デフォルト・ポートを使用する場合、クライアントは、接続要求でポートを参照せずに Oracle Internet Directory に接続できます。

Oracle ディレクトリ・サーバー・インスタンスの停止 コマンド・プロンプトで、次のコマンドを入力します。

```
oidctl connect=connect_string server=oidldapd instance=server_instance_number stop
```

次に例を示します。

```
oidctl connect=dbs1 server=oidldapd instance=3 stop
```

ディレクトリ・サーバー・インスタンスの起動に関するトラブルシューティング

ディレクトリ・サーバーが起動に失敗した場合は、ディレクトリ・サーバーを起動するためにユーザーが指定した構成パラメータをすべてオーバーライドし、サーバー起動後に ldapmodify 操作で、構成設定を使用可能な状態に戻すことができます。

ディレクトリに格納されている構成パラメータのかわりに、ハードコードされたデフォルト・パラメータを使用してディレクトリ・サーバーを起動するには、コマンド・プロンプトで次のコマンドを入力します。

```
oidctl connect=connect_string flags='-p port_number -f'
```

フラグ内に -f オプションを指定すると、定義済の構成設定が configset0 内の値を除いてすべてオーバーライドされ、ハードコードされた構成値でサーバーが起動されます。

OID 制御ユーティリティによって生成されたデバッグ・ログ・ファイルを見るには、$*ORACLE_HOME*/ldap/log にナビゲートします。

Oracle ディレクトリ・レプリケーション・サーバー・インスタンスの起動と停止

OID 制御ユーティリティを使用して、Oracle ディレクトリ・レプリケーション・サーバー・インスタンスの起動と停止を行います。

Oracle ディレクトリ・レプリケーション・サーバー・インスタンスの起動 Oracle ディレクトリ・レプリケーション・サーバーを起動する構文は、次のとおりです。

```
oidctl connect=connect_string server=oidrepld instance=server_instance_number
[configset=configset_number] flags=' -p directory_server_port_number -d debug_level
-h directory_server_host_name -m [true | false]-z transaction_size ' start
```

表 A-4 OIDCTL を使用してディレクトリ・レプリケーション・サーバーを起動するための引数

引数	説明
connect=*connect_string*	すでに tnsnames.ora ファイルを構成している場合は、ORACLE_HOME/network/admin にある、そのファイルに指定されている名前です。
server=oidrepld	起動するサーバーの種類（有効な値は OIDLDAPD と OIDREPLD です）。大文字と小文字は区別されません。
instance=*server_instance_number*	起動するサーバーのインスタンス番号。1 ～ 1000 の間の数値を設定してください。
configset=*configset_number*	サーバーの起動に使用される configset の番号。デフォルトの設定は、configset0 です。0 ～ 1000 の間の数値を設定してください。
-p *directory_server_port_number*	TCP ポート *directory_server_port_number* 上のディレクトリへの接続でレプリケーション・サーバーが使用するポート番号。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。
-d *debug_level*	レプリケーション・サーバー・インスタンス起動中のデバッグ・レベルを指定します。
-h *directory_server_host_name*	レプリケーション・サーバーを、デフォルトのホスト以外のホスト（つまり、ローカル・コンピュータ）に接続する場合、*directory_server_host_name* で指定します。*directory_server_host_name* には、コンピュータ名または IP アドレスを指定します。（レプリケーション・サーバーのみ）
-m [*true*|*false*]	競合解消を行うかどうかを設定します。TRUE および FALSE が有効な値です。デフォルトは TRUE です。（レプリケーション・サーバーのみ）
-z *transaction_size*	各レプリケーション更新サイクルで適用される変更の数を指定します。指定しない場合は、Oracle ディレクトリ・サーバーの sizelimit パラメータの値で決まります。sizelimit パラメータのデフォルト設定は 1024 です。この設定は変更できます。
start	*server* 引数で指定したサーバーを起動します。

たとえば、インスタンスが 1、ポート 12000、デバッグ・レベル 1024 でレプリケーション・サーバーを起動するには、コマンド・プロンプトで次のように入力します。

```
oidctl connect=dbs1 server=oidrepld instance=1 flags='-p 12000 -h eastsun11 -d 1024'
start
```

Oracle ディレクトリ・レプリケーション・サーバーの起動と停止では、-h フラグ（ホスト名を指定する引数）が必須です。その他のフラグはすべてオプションです。

フラグ引数内のペアのキーワード値はすべて、その間を 1 つの空白で区切る必要があります。

フラグは引用符で囲む必要があります。

configset 識別子が未設定の場合は、デフォルトで 0（configset0）に設定されます。

> **注意：** デフォルト・ポート（無保護使用の場合は 389、保護使用の場合は 636）以外のポートを使用する場合は、Oracle Internet Directory の配置に使用するポートをクライアントに通知する必要があります。デフォルト・ポートを使用する場合、クライアントは、接続要求でポートを参照せずに Oracle Internet Directory に接続できます。

Oracle ディレクトリ・レプリケーション・サーバー・インスタンスの停止 コマンド・プロンプトで、次のコマンドを入力します。

```
oidctl connect=connect_string server=OIDREPLD instance=server_instance_number stop
```

次に例を示します。

```
oidctl connect=dbs1 server=oidrepld instance=1 stop
```

Oracle Directory Integration Server の起動

Oracle Directory Integration Server の実行可能ファイル odisrv は、*$ORACLE_HOME*/bin ディレクトリに存在します。

Directory Integration and Provisioning Server の起動方法は、インストール環境によって異なります。

- 一般的な Oracle Internet Directory のインストール

  この場合、インストール環境には、サーバーおよびクライアント・コンポーネントのうち、特に OID モニターと OID 制御ユーティリティが含まれます。この場合は、これらのツールを使用して Directory Integration and Provisioning Server を起動および停止します。

> **注意：** Directory Integration and Provisioning Server は OID モニターと OID 制御ユーティリティを使用せずに起動することもできますが、これらを使用して起動することをお薦めします。これによって、Directory Integration and Provisioning Server が予期せずに終了した場合、OID モニターを使用して再起動できます。

- Oracle Directory Integration and Provisioning Platform のみのインストール

  この場合、Directory Integration and Provisioning Server の起動方法は、高可用性を目的として Oracle Directory Integration and Provisioning Platform を使用しているかどうかによって異なります。

  - 高可用性を目的として Oracle Directory Integration and Provisioning Platform を使用している場合は、OID モニターと OID 制御ユーティリティを使用して Directory Integration and Provisioning Server を起動することをお薦めします。この場合は、正しいホストおよび OID モニターの接続先の SID で、`tnsnames.ora` ファイルを構成する必要があります。
  - 高可用性を目的として Oracle Directory Integration and Provisioning Platform を使用していない場合は、OID モニターを使用せずに Directory Integration and Provisioning Server を起動することをお薦めします。

Directory Integration and Provisioning Server は、SSL モード（厳しいセキュリティ）または非 SSL モードのいずれでも起動することができます。データベースへの接続には、接続文字列を使用する必要があります。

> **注意：** Oracle Directory Integration Server をデフォルト・モードで起動すると、Oracle Directory Provisioning Integration Service のみがサポートされ、Oracle Directory Synchronization Service はサポートされません。

OID モニターと OID 制御ユーティリティを使用した Oracle Directory Integration Server の起動

Directory Integration and Provisioning Server を非 SSL モードで起動する手順は、次のとおりです。

1. OID モニターが実行されていることを確認します。このことを UNIX で確認するには、コマンドラインで次のように入力します。

   ```
   ps -ef | grep oidmon
   ```

   OID モニターが実行されていない場合は、A-4 ページの「OID モニター（oidmon）構文」の説明に従って OID モニターを起動します。

2. OID 制御ユーティリティを使用して Directory Integration and Provisioning Server を起動します。起動するには、次のように入力します。

```
oidctl [connect=connect_string] server=odisrv [instance=instance_number]
[config=configuration_set_number] [flags="[host=hostname] [port=port_number]
[debug=debug_level] [refresh=interval_between_refresh]
[grpID=group_identifier_of_provisioning_profile]
[maxprofiles=number_of_profiles]
[ sslauth=ssl_mode ]"] start
```

表 A-5 に、このコマンドの引数を示します。

表 A-5 Oracle Directory Integration Server を起動するための引数の説明

引数	説明
connect=*connect_string*	すでに tnsnames.ora ファイルを構成している場合は、$*ORACLE_HOME*/network/admin にある、そのファイルに指定されているネット・サービス名です。
server=odisrv	起動するサーバーの型。このケースでは、起動されるサーバーは odisrv です。大文字と小文字は区別されません。この引数は必須です。
instance=*instance_number*	Directory Integration and Provisioning Server に割り当てるインスタンス番号を指定します。このインスタンス番号は一意である必要があります。OID モニターは、インスタンス番号が、このサーバーの現在実行中のインスタンスに対応付けられていないことを検証します。インスタンス番号が現在実行中のインスタンスに対応付けられている場合、OID モニターはエラー・メッセージを戻します。
config=*configuration_set_number*	Directory Integration and Provisioning Server が実行する構成設定の番号を指定します。この引数は必須です。
host=*hostname*	Oracle ディレクトリ・サーバーのホスト名。
port=*port_number*	Oracle ディレクトリ・サーバーのポート番号。
debug=*debug_level*	Directory Integration and Provisioning Server に必要なデバッグ・レベル。 **関連項目：**様々なデバッグ・レベルの詳細は、『Oracle Internet Directory 管理者ガイド』の第 10 章を参照してください。
refresh=*interval_between_refreshes*	サーバーが統合プロファイルの変更を更新する間隔を分単位で指定します。デフォルトは 2 分（Refresh=2）です。
maxprofiles=*number_of_profiles*	このサーバー・インスタンスに対して同時に実行できるプロファイルの最大数を指定します。

表 A-5 Oracle Directory Integration Server を起動するための引数の説明（続き）

引数	説明

sslauth=*ssl_mode*	SSL モード。
	■ 0: SSL が使用されない非 SSL モード。
	■ 1: 暗号化用にのみ使用される SSL モード（PKI 認証なし）。この場合、Wallet は使用されません。
	■ 2: サーバー認証でのみ使用される SSL モード。このモードでは、Wallet のロケーションのみを必要とする他の Oracle Internet Directory ツールとは異なり、Oracle Wallet の完全なパス名（ファイル名自体を含む）を指定する必要があります。たとえば、サーバーのみのインストールまたは完全インストールの場合は、次のように入力します。
	`oidctl server=odisrv [instance=instance_number] [configset=configset_number] [grpID=group_identifier_of_provisioning_profile] flags="host=myhost port=myport sslauth=2`
	
	クライアントのみのインストールの場合は、次のように入力します。
	`odisrv [host=host_name] [port=port_number] config=configuration_set_number [instance=instance_number] [debug=debug_level] [refresh=interval_between_refresh] [maxprofiles=number_of_profiles] [refresh=interval_between_refresh] [maxprofiles=number_of_profiles] [sslauth=ssl_mode]`

OID モニターと OID 制御ユーティリティを使用しない Oracle Directory Integration Server の起動

OID モニターおよび OID 制御ツールを使用できないクライアントのみのインストール環境では、OID モニターまたは OID 制御ユーティリティを使用せずに、Oracle Directory Integration Server を非 SSL モードまたは SSL モード（高セキュリティ用）のいずれかで起動できます。各タイプで起動するためのパラメータについては、A-13 ページの表 A-5 を参照してください。

Directory Integration and Provisioning Server を起動するには、コマンドラインで次のように入力します。

```
odisrv [host=host_name] [port=port_number]
config=configuration_set_number [instance=instance_number] [debug=debug_level]
[refresh=interval_between_refresh] [maxprofiles=number_of_profiles]
[sslauth=ssl_mode]
```

Oracle Directory Integration Server の停止

Directory Integration and Provisioning Server を停止する方法は、起動に使用したツールによって異なります。

OID モニターと OID 制御ユーティリティを使用した Oracle Directory Integration Server の停止

OID モニターと OID 制御ユーティリティを使用して Directory Integration and Provisioning Server を起動した場合は、これらを使用して停止する必要があります。

1. Directory Integration and Provisioning Server を停止する前に、OID モニターが実行されていることを確認します。このことを確認するには、コマンドラインで次のように入力します。

   ```
   ps -ef | grep oidmon
   ```

   OID モニターが実行されていない場合は、A-4 ページの「OID モニター（oidmon）構文」の説明に従って OID モニターを起動します。

2. 次のように入力して、Directory Integration and Provisioning Server を停止します。

   ```
   oidctl [connect=connect_string] server=odisrv instance=instance stop
   ```

OID モニターと OID 制御ユーティリティを使用しない Oracle Directory Integration Server の停止

OID モニターと OID 制御ツールを使用できないクライアントのみのインストール環境では、OID 制御ツールを使用せずに Oracle Directory Integration Server を起動できます。これらのツールを使用せずにサーバーを停止するには、*$ORACLE_HOME*/ldap/admin ディレクトリに格納されている stopodiserver.sh ツールを使用します。

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 1.3.2.2-1 以上

  サイト：http://sources.redhat.com

- MKS Toolkit 6.1

  サイト：http://www.datafocus.com/

関連項目： stopodiserver.sh ツールの使用方法は、A-61 ページの「StopOdiServer.sh ツールの構文」を参照してください。

注意： 前述以外の方法で Oracle Directory Integration Server が停止した場合、そのサーバーは同じホストから起動できません。この場合は、次のコマンドを使用して、ディレクトリ内にある前回実行した際のフットプリントを削除する必要があります。

```
$ORACLE_HOME/ldap/admin/stopodiserver.sh [-host
directory_server_host] [-port directory_server_port]
[-binddn super_user_dN (default is cn=orcladmin)]
[-bindpass super_user_password (default is welcome)]
-instance number_of_the_instance_to_stop -clean
```

Oracle Internet Directory サーバー・インスタンスの再起動

予定の更新時刻を待たず、サーバーのキャッシュを即時に更新する場合は、RESTART コマンドを使用します。再起動した Oracle Internet Directory サーバーは、停止前と同じパラメータを保持しています。

Oracle Internet Directory サーバー・インスタンスを再起動するには、コマンド・プロンプトで次のコマンドを入力します。

```
oidctl connect=connect_string server={oidldapd|oidrepld|odisrv}
instance=server_instance_number  restart
```

ディレクトリ・サーバー・インスタンスを再起動する場合は、常に OID モニターが実行中であることが必要です。

ダウンしているサーバーに接続しようとすると、SDK からエラー・メッセージ「81:LDAP サーバーと通信できません。」を受け取ります。

アクティブなサーバー・インスタンスが参照している構成設定エントリを変更する場合、構成設定エントリの変更値をそのサーバー・インスタンスで有効にするには、そのインスタンスを停止してから再起動してください。STOP コマンドの後に START コマンドを発行するか、RESTART コマンドを使用します。RESTART は、サーバー・インスタンスを停止してから再起動します。

たとえば、Oracle ディレクトリ・サーバーの instance1 が、configset3 を使用してネット・サービス名 dbs1 で起動されたとします。その後、instance1 の稼働中に、configset3 内の属性の 1 つを変更したとします。configset3 の変更内容を instance1 で有効にするには、次のコマンドを入力します。

```
oidctl connect=dbs1 server=oidldapd instance=1 restart
```

configset3 を使用する複数の Oracle ディレクトリ・サーバーのインスタンスが、そのノードで実行中の場合は、次のコマンド構文を使用して、すべてのインスタンスを一度に再起動できます。

```
oidctl connect=dbs1 server=oidldapd restart
```

このコマンドは、configset3 を使用しているかどうかに関係なく、そのノードで実行中のインスタンスをすべて再起動することに注意してください。

重要： 再起動を実行中、クライアントは Oracle ディレクトリ・サーバー・インスタンスにアクセスできません。ただし、再起動にかかる時間は数秒です。

仮想ホストまたはラック・ノードでの Oracle Internet Directory サーバーの起動と停止

ディレクトリ・サーバー、ディレクトリ・レプリケーション・サーバーまたは Directory Integration and Provisioning Server を起動する場合は、host パラメータを使用して、仮想ホスト名を指定します。

仮想ホストまたはラック・ノードでのディレクトリ・サーバーの起動と停止

仮想ホスト上でディレクトリ・サーバーを起動するには、次のように入力します。

```
oidctl [connect=connect_string] host=virtual_host_name server=oidldapd
instance=instance_number configset=configset_number flags= "..." start
```

仮想ホスト上でディレクトリ・サーバーを停止するには、次のように入力します。

```
oidctl host=virtual_host_name server=oidldapd instance=instance_number stop
```

仮想ホストまたはラック・ノードでのディレクトリ・レプリケーション・サーバーの起動と停止

仮想ホスト上でディレクトリ・レプリケーション・サーバーを起動するには、次のように入力します。

```
oidctl [connect=connect_string] host=virtual_host_name server=oidrepld
instance=instance_number flags= "..." start
```

仮想ホスト上でディレクトリ・レプリケーション・サーバーを停止するには、次のように入力します。

```
oidctl host=virtual_host_name server=oidrepld instance=instance_number stop
```

仮想ホストまたはラック・ノードでの Oracle Directory Integration Server の起動と停止

仮想ホスト上で Directory Integration and Provisioning Server を起動するには、次のように入力します。

```
oidctl [connect=connect_string] host=virtual_host_name server=odisrv
instance=instance_number configset=configset_number flags= "..." start
```

仮想ホスト上で Directory Integration and Provisioning Server を停止するには、次のように入力します。

```
oidctl host=virtual/host_name server=odisrv instance=instance_number stop
```

ディレクトリ・サーバーは、仮想ホスト上での実行のために起動された場合、その仮想ホストのみに対応する IP アドレスに指定された LDAP ポートで要求をバインドおよびリスニングします。

ディレクトリ・サーバーとの通信時、ディレクトリ・レプリケーション・サーバーでは仮想ホスト名が使用されます。また、Oracle Internet Directory ノードに対する一意のレプリケーション識別子を表す `replicaID` 属性が 1 回生成されます。この属性は、ホスト名に依存しないため、コールド・フェイルオーバー構成での特別な処理は必要ありません。

ディレクトリ・サーバーとの通信時、Directory Integration and Provisioning Server では仮想ホスト名が使用されます。

エントリおよび属性の管理コマンドライン・ツール構文

この項では、次のツールの使用方法を説明します。

- カタログ管理ツール（catalog.sh）構文
- ldapadd の構文
- ldapaddmt の構文
- ldapbind の構文
- ldapcompare の構文
- ldapdelete の構文
- ldapmoddn の構文
- ldapmodify の構文
- ldapmodifymt の構文
- ldapsearch の構文

注意： UNIX シェルでは、アスタリスク（*）などの一部の文字が特殊文字として解釈される場合があります。使用しているシェルに応じて、これらの文字をエスケープする必要があります。

カタログ管理ツール（catalog.sh）構文

Oracle Internet Directory では、索引を使用して属性を検索できます。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。次の条件を満たす属性のみ索引を付けることができます。

- 等価の一致規則
- Oracle Internet Directory でサポートする一致規則

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。この操作は、Oracle Directory Manager を使用して属性を作成するときに実行できます。ただし、すでに存在している属性への索引付けに使用できるのは、カタログ管理ツールのみです。

catalog.sh を実行する前に、ディレクトリ・サーバーが停止または読取り専用モードのいずれかの状態にあることを確認してください。これらの状態でない場合は、データの一貫性が維持されなくなります。

注意： Oracle Internet Directory ベースのスキーマで作成された索引に対して、catalog.sh `-delete` オプションは使用しないでください。ベース・スキーマ属性から索引を削除すると、Oracle Internet Directory の操作に悪影響を及ぼす場合があります。

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 1.3.2.2-1 以上

  サイト：`http://sources.redhat.com`

- MKS Toolkit 6.1

  サイト：`http://www.datafocus.com/`

カタログ管理ツールは次の構文を使用します。

```
catalog.sh -connect connect_string {-add|-delete} {-attr attr_name|-file file_name}
```

表 A-6 カタログ管理ツール（catalog.sh）の引数

引数	説明
`-connect connect_string`	ディレクトリ・データベースに接続するための接続文字列を指定します。この引数は必須です。 **関連項目：**『Oracle9*i* Net Services 管理者ガイド』を参照してください。
`-add -attr attr_name`	指定した属性を索引付けします。
`-delete -attr attr_name`	指定した属性から索引を削除します。
`-add -file file_name`	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 file_name`	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後にその変更内容を有効にするには、Oracle ディレクトリ・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・サーバーの起動と再起動の方法は、A-6 ページの「OID 制御ユーティリティ（oidctl）の構文」を参照してください。ディレクトリ・サーバーを起動する場合は、あらかじめ OID モニターが実行されている必要があります。
- OID モニターの開始については、A-4 ページの「OID モニター（oidmon）構文」を参照してください。
- Oracle Internet Directory でサポートする一致規則については、『Oracle Internet Directory 管理者ガイド』を参照してください。

ldapadd の構文

ldapadd コマンドライン・ツールを使用すると、エントリ、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、A-32 ページの「ldapmodify の構文」を参照してください。

> **関連項目：** ldapadd を使用して、入力ファイルでサーバーを構成する方法については、『Oracle Internet Directory 管理者ガイド』の第 5 章を参照してください。

ldapadd は次の構文を使用します。

```
ldapadd [arguments] -f file_name
```

file_name は、A-2 ページの「LDAP Data Interchange Format（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
```

表 A-7 ldapadd の引数

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 ldapadd は停止します。）
-D "*binddn*"	ディレクトリに対して認証を行う場合に、*binddn* に指定されているエントリ（認証を必要とする識別名）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』の付録 G を参照してください。
-f *file_name*	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、A-2 ページの「LDAP Data Interchange Format（LDIF）の構文」を参照してください。

表 A-7 ldapadd の引数（続き）

オプションの引数	説明
`-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、*ldaphost* に接続します。*ldaphost* 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 バインドの最初の手順のみ実行します。
`-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`	`ManageDSAIT` 制御をサーバーに送信するようにツールに指示します。`ManageDSAIT` 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常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 ref_hop_limit`	クライアントが処理する参照ホップの数を指定します。デフォルト値は 5 です。
`-p directory_server_port_number`	TCP ポート *directory_server_port_number* 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`	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
`-U SSLAuth`	SSL 認証モードを指定します。 ■ 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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。
`-w password`	接続に必要なパスワードを指定します。

表 A-7 ldapadd の引数（続き）

オプションの引数	説明
`-W wallet_location`	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。 たとえば、このパラメータは、UNIX では `-W "file:/home/my_dir/my_wallet"` と設定します。 また、Windows NT では `-W "file:C:¥my_dir¥my_wallet"` と設定します。
`-X dsml_file`	DSML 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。

ldapaddmt の構文

ldapaddmt は ldapadd に類似しています。これを使用すると、エントリ、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、現行のディレクトリ内の `add.log` ファイルにエラー・ログを記録します。

ldapaddmt は次の構文を使用します。

```
ldapaddmt -T number_of_threads -h host -p port -f file_name
```

file_name は、A-2 ページの「LDAP Data Interchange Format（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。

表 A-8 ldapadd の引数

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 "`*`binddn`*`"`	ディレクトリに対して認証を行う場合に、*binddn* に指定されているエントリ（認証を必要とするユーザーの識別名）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』の付録 G を参照してください。
`-h` *ldap_host*	デフォルトのホスト（ローカル・コンピュータ）ではなく、*ldaphost* に接続します。*ldaphost* 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 バインドの最初の手順のみ実行します。
`-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、定義済の Kerberos でコンパイルする必要があります。証明書を付与する有効なチケットをすでに所有している必要があります。
`-M`	`ManageDSAIT` 制御をサーバーに送信するようにツールに指示します。`ManageDSAIT` 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常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` *ref_hop_limit*	クライアントが処理する参照ホップの数を指定します。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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
`-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。

表 A-8 ldapadd の引数（続き）

オプションの引数	説明
-U *SSLAuth*	SSL 認証モードを指定します。 ■ 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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。
-w *password*	接続に必要なパスワードを指定します。
-W *wallet_location*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば、このパラメータは、UNIX では -W "file:/home/my_dir/my_wallet" と設定し、Windows NT では -W "file:C:¥my_dir¥my_wallet" と設定します。
-X *dsml_file*	DSML 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。

ldapbind の構文

ldapbind コマンドライン・ツールを使用すると、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 は次の構文を使用します。

```
ldapbind [arguments]
```

表 A-9 ldapbind の引数

オプションの引数	説明
-D "*binddn*"	ディレクトリに対して認証を行う場合に、*binddn* に指定されているエントリ（認証を必要とする識別名）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』の付録 G を参照してください。

表 A-9 ldapbind の引数（続き）

オプションの引数	説明
-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、*ldaphost* に接続します。*ldaphost* 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 *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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
-U *SSLAuth*	SSL 認証モードを指定します。 ■ 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 *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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。
-w *password*	接続に必要なパスワードを指定します。
-W *wallet_location*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば、このパラメータは、UNIX では -W "file:/home/my_dir/my_wallet" と設定し、Windows NT では -W "file:C:¥my_dir¥my_wallet" と設定します。
-O *sasl_security_properties*	SASL セキュリティ・プロパティを指定します。サポートされるセキュリティ・プロパティは、-O "auth" です。このセキュリティ・プロパティは、Digest-MD5 SASL メカニズム用です。データ整合性またはデータ・プライバシのない認証を実行できます。
-Y *sasl_mechanism*	SASL メカニズムを指定します。次のメカニズムがサポートされます。 ■ Y "DIGEST-MD5" ■ Y "EXTERNAL": このメカニズムでの SASL 認証は、クライアントとサーバーの SSL 認証とともに実行されます。この場合、SSL Wallet に格納されているユーザーの識別情報が SASL 認証に使用されます。
-R *sasl_realm*	SASL レルムを指定します。

ldapcompare の構文

ldapcompare コマンドライン・ツールを使用すると、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 は次の構文を使用します。

```
ldapcompare [arguments]
```

次の例は、Person Nine の title 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
```

表 A-10 ldapcompare の引数

オプションの引数	説明
-a *attribute name*	比較を実行する属性を指定します。この引数は必須です。
-b "*basedn*"	比較を実行するエントリの識別名を指定します。この引数は必須です。
-v *attribute value*	比較する属性値を指定します。この引数は必須です。
-D *binddn*	ディレクトリに対して認証を行う場合に、*binddn* に指定されているエントリ（認証を必要とするユーザーの識別名）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。
-d *debug-level*	デバッグ・レベルを設定します。『Oracle Internet Directory 管理者ガイド』の第 10 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』の付録 G を参照してください。
-f *file_name*	入力ファイル名を指定します。
-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、*ldaphost* に接続します。*ldaphost* には、コンピュータ名または IP アドレスを指定します。
-M	ManageDSAIT 制御をサーバーに送信するようにツールに指示します。ManageDSAIT 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常のエントリとして戻されます。
-O *ref_hop_limit*	クライアントが処理する参照ホップの数を指定します。デフォルト値は 5 です。

表 A-10 ldapcompare の引数（続き）

オプションの引数	説明
-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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
-U *SSLAuth*	SSL 認証モードを指定します。 ■ 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 *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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。
-w *password*	接続に必要なパスワードを指定します。
-W *wallet_location*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば、このパラメータは、UNIX では `-W "file:/home/my_dir/my_wallet"` と設定します。 また、Windows NT では `-W "file:C:¥my_dir¥my_wallet"` と設定します。

ldapdelete の構文

ldapdelete コマンドライン・ツールを使用すると、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 は次の構文を使用します。

```
ldapdelete [arguments] ["entry_DN" | -f input_file_name]
```

注意： エントリ識別名を指定する場合は、`-f` オプションは使用できません。

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"
```

表 A-11 ldapdelete の引数

オプションの引数	説明
-D "*binddn*"	ディレクトリに対して認証を行う場合に、*binddn* パラメータに完全識別名（認証を必要とするユーザーの識別名）を使用します。通常、`-w` *password* オプションとともに使用します。
-d *debug-level*	デバッグ・レベルを設定します。『Oracle Internet Directory 管理者ガイド』の第 10 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』の付録 G を参照してください。
-f *input_file_name*	入力ファイル名を指定します。
-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、*ldaphost* に接続します。*ldaphost* 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、定義済の Kerberos でコンパイルする必要があります。証明書を付与する有効なチケットをすでに所有している必要があります。
`-M`	`ManageDSAIT` 制御をサーバーに送信するようにツールに指示します。`ManageDSAIT` 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常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 *ref_hop_limit*	クライアントが処理する参照ホップの数を指定します。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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
-U *SSLAuth*	SSL 認証モードを指定します。 ■ 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。

表 A-11 ldapdelete の引数（続き）

オプションの引数	説明
-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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。
-w *password*	接続に必要なパスワードを指定します。
-W *wallet_location*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば、このパラメータは、UNIX では -W "file:/home/my_dir/my_wallet" と設定し、Windows NT では -W "file:C:¥my_dir¥my_wallet" と設定します。

ldapmoddn の構文

ldapmoddn コマンドライン・ツールを使用すると、エントリの識別名または相対識別名を変更できます。

ldapmoddn は次の構文を使用します。

```
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=mary smith` から `cn=mary jones` 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=mary smith,dc=Americas,dc=imc,dc=com" -R "cn=mary jones"
```

表 A-12 ldapmoddn の引数

引数	説明
-b "*basedn*"	変更されるエントリの識別名を指定します。この引数は必須です。
-D "*binddn*"	ディレクトリに対して認証を行う場合に、*binddn* に指定されているエントリ（認証を必要とするユーザーの識別名）として認証します。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』の付録 G を参照してください。
-f *file_name*	入力ファイル名を指定します。
-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、*ldaphost* に接続します。*ldaphost* には、コンピュータ名または IP アドレスを指定します。

表 A-12 ldapmoddn の引数（続き）

引数	説明
-M	ManageDSAIT 制御をサーバーに送信するようにツールに指示します。ManageDSAIT 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常のエントリとして戻されます。
-N *newparent*	相対識別名の新しい親を指定します。この引数または引数 -R 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参照ホップの数を指定します。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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
-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 *newrdn*	新規相対識別名を指定します。この引数または引数 -N を指定する必要があります。
-U *SSLAuth*	SSL 認証モードを指定します。 ■ 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 *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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。
-w *password*	接続に必要なパスワードを指定します。
-W *wallet_location*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば UNIX では、このパラメータは `-W "file:/home/my_dir/my_wallet"` と設定します。 また、Windows NT では `-W "file:C:¥my_dir¥my_wallet"` と設定します。

ldapmodify の構文

ldapmodify ツールは、属性で作用します。

ldapmodify は次の構文を使用します。

```
ldapmodify [arguments] -f file_name
```

file_name は、A-2 ページの「LDAP Data Interchange Format（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、すべての引数ではありません。これらの引数はすべてオプションです。

表 A-13 ldapmodify の引数

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。
-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を行う場合に、*binddn* に指定されているエントリ（認証を必要とするユーザーの識別名）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』の付録 G を参照してください。
-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、*ldaphost* に接続します。*ldaphost* には、コンピュータ名または IP アドレスを指定します。
`-M`	`ManageDSAIT` 制御をサーバーに送信するようにツールに指示します。`ManageDSAIT` 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常のエントリとして戻されます。

表 A-13 ldapmodify の引数（続き）

引数	説明
-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、ログ・ファイル内の誤った LDIF エントリの書込みに使用できます。ログ・ファイル名には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参照ホップの数を指定します。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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
-U *SSLAuth*	SSL 認証モードを指定します。 ■ 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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。
-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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば、このパラメータは、UNIX では `-W "file:/home/my_dir/my_wallet"` と設定します。 また、Windows NT では `-W "file:C:¥my_dir¥my_wallet"` と設定します。

`-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（A-2 ページの「LDAP Data Interchange Format（LDIF）の構文」を参照）。仕様はこの項に示すとおりです。

いくつかの変更を行う場合は、入力する各変更の間に、ハイフン（-）のみを含む行を追加します。次に例を示します。

```
dn: cn=Barbara Fritchy,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。次に例を示します。

```
dn:cn=Barbara Fritchy,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。次に例を示します。

```
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（A-2 ページの「LDAP Data Interchange Format（LDIF）の構文」を参照）。

- `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...
    ```

次に例を示します。

```
dn:cn=Barbara Fritchy,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]
```

次に例を示します。

```
dn: cn=Barbara Fritchy,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、存在するかどうかわからない属性を削除する場合に便利です。

次に例を示します。

```
dn: cn=Barbara Fritchy,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。

次に例を示します。

```
dn: cn=Barbara Fritchy,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`

  次に例を示します。

```
dn: cn=Barbara Fritchy,ou=Sales,o=Oracle,c=US
changetype: modrdn
newrdn: cn=Barbara Fritchy-Blomberg
```

属性を単一値として指定するには、LDIF ファイルの属性定義エントリに、空白で囲んだキーワード SINGLE-VALUE を含めます。

例 : ldapmodify を使用した属性の追加

この例では、myAttr と呼ばれる新規属性を追加します。この操作の LDIF ファイルは次のようになります。

```
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、この新規属性の位置を指定する識別名を入力します。すべての属性およびオブジェクト・クラスが cn=subschemasubentry に格納されます。

2 行目と 3 行目は、新規属性を追加するための正しい形式を示します。

最後の行は属性定義です。この最初の部分は、オブジェクト識別子番号 1.2.3.4.5.6.7 です。これは、他のすべてのオブジェクト・クラスおよび属性の中で一意であることが必要です。次の部分は属性の NAME です。このケースでは、属性の NAME は myAttr です。これは引用符で囲む必要があります。次の部分は属性の説明です。引用符の間に任意の説明を入力します。この例の属性定義の最後の部分は、属性に対するオプションの形式化規則です。このケースでは、EQUALITY caseIgnoreMatch の一致規則と Directory String の SYNTAX を追加します。この例では、SYNTAXES の名前 Directory String のかわりにオブジェクト ID 番号 1.3.6.1.4.1.1466.115.121.1.15 が使用されています。

属性情報は、この例のような形式のファイルに格納します。次に、次のコマンドを実行して、Oracle ディレクトリ・サーバーのスキーマに属性を追加します。

```
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・サーバーがポート 389 で実行されており、スーパー・ユーザーのアカウント名が orcladmin で、スーパー・ユーザーのパスワードが welcome です。また、LDIF ファイルが newattr.ldif であることが仮定されています。*yourhostname* と表示されるコンピュータのホスト名を置換します。

LDIF ファイルがあるディレクトリを現在使用中でない場合は、コマンドの最後でファイルにフル・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 ファイルが /tmp ディレクトリにあることが仮定されています。

ldapmodifymt の構文

ldapmodifymt コマンドライン・ツールを使用すると、複数のエントリを同時に変更できます。

ldapmodifymt は次の構文を使用します。

```
ldapmodifymt -T number_of_threads [arguments] -f file_name
```

file_name は、A-2 ページの「LDAP Data Interchange Format（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形式設定の仕様については、A-32 ページの「ldapmodify の構文」を参照してください。

次の例は、5 つの同時スレッドを使用して、ファイル myentries.ldif 内のエントリを変更しています。

```
ldapmodifymt -T 5 -h node1 -p 3000 -f myentries.ldif
```

> **注意：** ldapmodifymt ツールは、エラー・メッセージを、このコマンドを実行しているディレクトリにあるファイル add.log にログします。

次の表の引数はすべてオプションです。

表 A-14 ldapmodifymt の引数

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。

表 A-14 ldapmodifymt の引数（続き）

引数	説明
`-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を行う場合に、*binddn* に指定されているエントリ（認証を必要とするユーザーの識別名）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』の付録 G を参照してください。
`-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、*ldaphost* に接続します。*ldaphost* には、コンピュータ名または IP アドレスを指定します。
`-M`	`ManageDSAIT` 制御をサーバーに送信するようにツールに指示します。`ManageDSAIT` 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常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 ref_hop_limit`	クライアントが処理する参照ホップの数を指定します。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`	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
`-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 SSLAuth`	SSL 認証モードを指定します。 ■ 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、この場合ツールは LDAP バージョン 3 のプロトコルを使用します。値が 2 の場合、ツールは LDAP バージョン 2 のプロトコルを使用します。

表 A-14 ldapmodifymt の引数（続き）

引数	説明
`-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`	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば、このパラメータは、UNIX では -W "file:/home/my_dir/my_wallet" と設定します。 また、Windows NT では `-W "file:C:¥my_dir¥my_wallet"` と設定します。

ldapsearch の構文

ldapsearch コマンドライン・ツールを使用すると、ディレクトリ内の特定のエントリを検索および取得できます。

ldapsearch ツールは次の構文を使用します。

```
ldapsearch [arguments] filter [attributes]
```

filter の形式は RFC-2254 に準拠している必要があります。

> **関連項目：** フィルタの形式の標準の詳細は、`http://www.ietf.org` で RFC-2254 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。

注意：

- ldapsearch ツールは、デフォルトでは LDIF 出力を生成しません。ldapsearch コマンドライン・ツールから LDIF 出力を生成するには、`-L` フラグを使用します。
- UNIX シェルでは、アスタリスク（*）などの一部の文字が特殊文字として解釈される場合があります。使用しているシェルに応じて、これらの文字をエスケープする必要があります。

表 A-15 ldapsearch の引数

引数	説明
-b "*basedn*"	検索のためのベース識別名を指定します。この引数は必須です。
-s *scope*	この引数は必須です。検索有効範囲（base、one または sub Base）を指定します。特定のディレクトリ・エントリを取り出します。この検索範囲の指定とともに、検索基準バーを使用して、属性 `objectClass` とフィルタ `Present` を選択します。1 レベル : 検索のルートの 1 レベル下のすべてのエントリに検索を制限します。サブツリー : 検索のルートを含め、サブツリー全体のエントリを検索します。
-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（never、always、search または find）を指定します。
-B	非 ASCII 値を出力します。
-D "*binddn*"	ディレクトリに対して認証を行う場合に、*binddn* に指定されているエントリ（認証を必要とする識別名）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。
-d *debug level*	指定したレベルにデバッグ・レベルを設定します。（『Oracle Internet Directory 管理者ガイド』の第 10 章を参照。）
-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』の付録 G を参照してください。
-f *file*	*file* 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「`sep`」を印刷します。
-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、*ldaphost* に接続します。*ldaphost* には、コンピュータ名または IP アドレスを指定します。
-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 *timelimit*	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。

表 A-15 ldapsearch の引数（続き）

引数	説明
-M	ManageDSAIT 制御をサーバーに送信するようにツールに指示します。ManageDSAIT 制御は、参照をクライアントに送信しないようにサーバーに指示します。この指示がない場合、参照エントリが通常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 *ref_hop_limit*	クライアントが処理する参照ホップの数を指定します。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*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet のパスワードを指定します。
-S *attr*	検索結果を属性 *attr* 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 *SSLAuth*	SSL 認証モードを指定します。 ■ 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。
-w *passwd*	簡易認証の場合にバインド・パスワードを指定します。
-W *wallet_location*	サーバー、またはクライアントとサーバーの SSL 接続の場合は必須の、Wallet の位置を指定します。たとえば、このパラメータは、UNIX では -W "file:/home/my_dir/*my_wallet*" と設定します。 また、Windows NT では -W "file:C:¥my_dir¥*my_wallet*" と設定します。
-z *sizelimit*	エントリの最大検索数を指定します。
-X	エントリを DSML バージョン 1 の形式で出力します。

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、ディレクトリ上でルートからベース・レベルの検索を実行します。

```
ldapsearch -p 389 -h myhost -b "" -s base -v "objectclass=*"
```

- -b で、検索のためのベース識別名（この場合はルート）を指定します。
- -s で、ベース検索（base）、1 レベルの検索（one）またはサブツリー検索（sub）のうちの、いずれの検索かを指定します。
- "objectclass=*" で、検索のフィルタを指定します。

例 2: 1 レベルの検索 次の例は、"ou=HR, ou=Americas, o=IMC, c=US" で開始される 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て、"cn=us" 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: サイズ制限を使用する検索 次の例では、一致するエントリが 3 つ以上あっても、実際に取り出すエントリは 2 つのみです。

```
ldapsearch -h myhost -p 389 -z 2 -b "ou=Benefits,ou=HR,ou=Americas,o=IMC,c=US" -s
one "objectclass=*"
```

例 5: 必須の属性での検索 次の例は、一致したエントリの DN 属性値のみを戻します。

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "objectclass=*" dn
```

次の例は、姓（sn）および説明（description）属性値を使用して、識別名のみを取り出します。

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "cn=Person*" dn sn description
```

例 6: 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のある一般名（cn）属性を使用して、エントリを取り出します。この例の場合には、一般名がフランス語で、R で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 のエントリで、cn;lang-it 言語コード属性オプションに値が設定されていないと想定します。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: 全ユーザー属性および指定した操作属性の検索 次の例は、全ユーザー属性と、createtimestamp 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
```

次の例は、Anne Smith によって変更されたエントリを取り出します。

```
ldapsearch -h sun1 -b "" "(&(objectclass=*)(modifiersname=cn=Anne
Smith))"
```

次の例は、2001 年 4 月 1 日から 2001 年 4 月 6 日までの間に変更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 と 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 つの属性に索引を付けてください。前述の 2 つの ldapsearch コマンドを発行する前に、Oracle ディレクトリ・サーバーを再起動してください。

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 属性の値を持つすべて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 "objectclass=*"
```

次の例は、objectclass 属性の値が orcl 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=orcl*"
```

次の例は、objectclass 属性が orcl 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=orcl*)(cn=foo*))"
```

次の例は、一般名（cn）が foo ではない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、あるいは 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 10000 より小か等しい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"
```

Oracle Directory Integration and Provisioning Platform コマンドライン・ツールの構文

この項では、次の項目について説明します。

- Directory Integration and Provisioning Assistant
- ldapUploadAgentFile.sh ツールの構文
- ldapCreateConn.sh ツール構文
- ldapDeleteConn.sh ツール構文
- StopOdiServer.sh ツールの構文
- schemasync ツールの構文
- Oracle Directory Integration Server の登録ツール（odisrvreg）
- プロビジョニング・サブスクリプション・ツール （oidprovtool）の構文

Directory Integration and Provisioning Assistant

表 A-16 に、Directory Integration and Provisioning Assistant および対応するコマンドを使用して実行可能なタスクを示します。各タスクの詳細情報の参照先も示します。

表 A-16 Directory Integration and Provisioning Assistant の機能の概要

タスク	コマンド	詳細情報の参照先

同期プロファイルの作成、変更または削除	`createprofile`
	`modifyprofile`
	`deleteprofile`
		A-46 ページの「同期プロファイルの作成、変更および削除」

Oracle Internet Directory 内のすべてのプロファイル名の表示	`listprofiles`	A-53 ページの「Oracle Internet Directory 内のすべての同期プロファイルの表示」
特定のプロファイルの詳細の表示	`showprofile`	A-54 ページの「特定の同期プロファイルの詳細の表示」
同期開始前の Oracle Internet Directory と接続ディレクトリの統一	`bootstrap`	A-48 ページの「Directory Integration and Provisioning Assistant を使用したディレクトリのブートストラップ」
Oracle Directory Integration and Provisioning Server が Oracle Internet Directory への接続時に使用する Wallet パスワードの設定	`wpasswd`	A-54 ページの「Oracle Directory Integration Server に対する Wallet パスワードの設定」
Oracle Directory Integration Platform 管理者のパスワードの再設定	`chgpasswd`	A-53 ページの「Oracle Directory Integration and Provisioning Platform 管理者のパスワードの変更」
認証管理ノード間での統合プロファイルの移動	`reassociate`	A-54 ページの「認証管理ノード間での統合プロファイルの移動」

Directory Integration and Provisioning Assistant のコマンドライン・インタフェースは次のとおりです。

dipassistant *command* [-help]

command := *Directory Integration and Provisioning Assistant command*

```
Directory Integration and Provisioning Assistant command :=
        createprofile [cp]
        | modifyprofile [mp]
        | deleteprofile [dp]
        | listprofiles[lsprof]
        | showprofile[sp]
        | bootstrap [bs]
        | wpasswd [wp]
```

```
        | chgpasswd [cpw]
        | reassociate [rs]
```

特定のコマンドのヘルプを参照するには、次のとおり入力します。

```
dipassistant command -help
```

同期プロファイルの作成、変更および削除

Directory Integration and Provisioning Assistant を使用して同期プロファイルを作成、変更または削除するための構文は、次のとおりです。

```
dipassistant createprofile | modifyprofile | deleteprofile
[-host host name] [-port port number] [-dn bind_DN] [-passwd password]
{-file file name | -profile profile name } [propName1=value]
[propName2=value]... [-configset configset_number]
```

次に例を示します。

```
dipassistant createprofile -host myhost -port 3060 -passwd xxxx
-file import.profile -configset 1

dipassistant modifyprofile -host myhost -port 3060 -passwd xxxx
-file import.profile -dn xxxx -passwd xxxx -profile myprofile
[propName1=value]
[propName2=value]...

dipassistant deleteprofile -profile myprofile [-host myhost] [-port 3060] [-dn xxxx]
[-passwd xxxx] [-configset 1]
```

A-46 ページの表 A-17 に Directory Integration and Provisioning Assistant を使用して同期プロファイルを作成、変更または削除するためのパラメータを示します。

表 A-17 Directory Integration and Provisioning Assistant を使用して同期プロファイルを作成、変更または削除するためのパラメータ

パラメータ	説明
-host	Oracle Internet Directory が実行されているホスト。デフォルト値はローカル・ホストの名前です。
-port	Oracle Internet Directory が起動されたポート。デフォルトは 389 です。
-dn	ディレクトリに対しての識別に使用されるバインド識別名。デフォルト値は Oracle Directory Integration and Provisioning Platform 管理者の識別名です。
-passwd	ディレクトリに対してのバインド中に使用されるバインド識別名のパスワード。

表 A-17 Directory Integration and Provisioning Assistant を使用して同期プロファイルを作成、変更または削除するためのパラメータ（続き）

パラメータ	説明
-file	すべてのプロファイル・パラメータを含むファイル。 **関連項目：**A-47 ページの表 A-18 にパラメータ・リストおよびその説明を示します。
-configset	プロファイルの関連付けが必要な構成設定エントリの番号。
-profile	変更が必要なプロファイル。

表 A-18 に、createprofile コマンドと modifyprofile コマンドによって想定されるプロパティが示します。既存のプロファイルの変更時に、デフォルトは想定されません。ファイル内に指定された属性のみが変更されます。

表 A-18 CreateProfile コマンドと ModifyProfile コマンドによって想定されるプロパティ

パラメータ	説明	デフォルト
odip.profile.name	プロファイルの名前。	-
odip.profile.password	プロファイルにアクセスするためのパスワード	-
odip.profile.status	DISABLE または ENABLE。	DISABLE
odip.profile.syncmode	同期の方向。変更がサード・パーティから Oracle Internet Directory に伝播される場合、同期のモードは IMPORT です。変更がサード・パーティのディレクトリに伝播される場合、同期のモードは EXPORT です。	IMPORT
odip.profile.retry	エラーが発生して統合サーバーが終了する前に、プロファイルを実行可能な最大回数。	4
odip.profile.schedinterval	統合サーバーによるプロファイルの連続実行の間隔。前回の実行が完了していない場合、その実行が完了するまで次の実行は再開しません。	1 分
odip.profile.agentexecommand	NON-LDAP インタフェースの場合、LDIF 形式で情報を生成するためのコマンド。	-
odip.profile.condirurl	サード・パーティ・ディレクトリの位置 [hostname:port]。	-
odip.profile.condiraccount	サード・パーティ・ディレクトリへの接続に使用されるバインド識別名またはユーザー名。	-
odip.profile.condirpassword	サード・パーティ・ディレクトリに対しての識別に使用されるパスワード。	-

表 A-18 CreateProfile コマンドと ModifyProfile コマンドによって想定されるプロパティ（続き）

パラメータ	説明	デフォルト
odip.profile.interface	データ交換に LDAP、LDIF、DB、TAGGED のいずれの形式を使用するかを示すインジケータ。	LDAP
odip.profile.configfile	実行に使用される追加のプロファイル固有の情報を含むファイルの名前。	-
odip.profile.mapfile	マッピング・ルールを含むファイルの名前。	-
odip.profile.condirfilter	Oracle Internet Directory へのインポート前に、接続ディレクトリから読み込まれた変更に対して適用する必要があるフィルタ。	-
odip.profile.oidfilter	接続ディレクトリへのエクスポート前に、Oracle Internet Directory から読み込まれた変更に対して適用する必要があるフィルタ。	-
odip.profile.lastchgnum	最後に適用された変更番号。エクスポート・プロファイルの場合、この番号は Oracle Internet Directory で最後に適用された番号を指しますが、インポート・プロファイルの場合、この番号は接続ディレクトリで最後に適用された変更番号を指します。	-

Directory Integration and Provisioning Assistant を使用したディレクトリのブートストラップ

bootstrap コマンドのコマンドライン・インタフェースは次のとおりです。

```
dipassistant bootstrap { -profile profile_name [-host host_name] [-port port_number]
-dn bind_DN [-passwd password] [-log log_file] [-logseverity severity] [-trace
trace_file] [-tracelevel trace_level] [-loadparallelism <#nThrs>] [-loadretry
<retryCnt>] | -cfg file_name }
```

例 :

```
dipassistant bs –cfg bootstrap cfg
```

または

```
dipassistant bs -host myhost –port 3060 –dn cn=orcladmin –password xxxx –profile
iPlanetProfile
```

表 A-19 deleteprofile コマンドのパラメータ

パラメータ	説明
-cfg	ブートストラップの実行に必要なすべてのパラメータを含む構成ファイル。 **関連項目：**A-50 ページの表 A-20 にパラメータ・リストおよびその説明を示します。
-host	Oracle Internet Directory が実行されていたホスト。
-port	Oracle Internet Directory が起動されたポート。
-dn	ディレクトリに対しての識別に使用されるバインド識別名。
-password	ディレクトリに対してのバインド中に使用されるバインド識別名のパスワード。
-profile	プロファイル名
-log	ログ・ファイル。このパラメータを指定しない場合、デフォルトでは OH/ldap/odi/bootstrap.log にログ情報が書き込まれます。
-logseverity	ログの重要度 1 ～ 15。1 は INFO、2 は WARNING、3 は DEBUG、4 は ERROR、またはこれらの任意組合せ。指定しない場合は、INFO および ERROR メッセージのみが記録されます。
-trace	デバッグのためのトレース・ファイル。
-trace level	トレース・レベル。
-loadRetry	宛先へのロードが失敗した場合、エントリに「不良エントリ」のマークが付く前に実行可能な再試行の回数。
-loadparallelism	Oracle Internet Directory へのロードが複数のスレッドを使用してパラレルで実行されることを示すインジケータ。たとえば、-loadparallelism 5 は、5 つのスレッドが作成され、それぞれが Oracle Internet Directory へのエントリのロードをパラレルで試行することを示します。

ブートストラップ・コマンドによって想定されるプロパティ

表 A-20 ブートストラップ・プロパティ

プロパティ	説明	必須	デフォルト
odip.bootstrap.srctype	ブートストラップのソースが LDAP と LDIF のいずれかを示すインジケータ。有効な値は LDAP または LDIF です。	はい	-
odip.bootstrap.desttype	ブートストラップの宛先が LDAP と LDIF のいずれかを示すインジケータ。有効な値は LDAP または LDIF です。	はい	-
odip.bootstrap.srcurl	LDAP ソース・タイプの場合はソース・ディレクトリの位置。LDIF の場合は、LDIF ファイルの位置。 **注意 :** LDAP の場合、想定される形式は host[:port] です。LDIF の場合、想定される形式はファイルの絶対パスです。	はい	-
odip.bootstrap.desturl	LDAP の場合は、宛先ディレクトリの位置。LDIF の場合は、LDIF ファイルの位置。 **注意 :** LDAP の場合、想定される形式は host[:port] です。LDIF の場合、想定される形式はファイルの絶対パスです。	はい	-
odip.bootstrap.srcsslmode	ブートストラップのソースへの接続に SSL ベースの認証を使用する必要があるかどうかを示すインジケータ。TRUE の値は、SSL ベースの認証を使用する必要があることを示します。	いいえ	FALSE

表 A-20 ブートストラップ・プロパティ（続き）

プロパティ	説明	必須	デフォルト
odip.bootstrap.destsslmode	ブートストラップの宛先への接続に SSL ベースの認証を使用する必要があるかどうかを示すインジケータ。TRUE は、SSL ベースの認証を使用する必要があることを示します。 **注意：** LDIF の場合、このパラメータは無効です。	いいえ	FALSE
odip.bootstrap.srcdn	ソース URL への補足。LDIF バインドの場合、このパラメータは無効です。ただし、LDAP の場合、このパラメータはバインド識別名を指定します。	LDAP の場合のみ	-
odip.bootstrap.destdn	接続先 URL への補足。LDIF バインドの場合、このパラメータは無効です。ただし、LDAP の場合、このパラメータはバインド識別名を指定します。	LDAP の場合のみ	-
odip.bootstrap.srcpasswd	ソースへのバインド・パスワード。LDAP バインドの場合、このパスワードはセキュリティとして使用されます。このファイルではパスワードを指定しないことをお薦めします。	いいえ	-
odip.bootstrap.destpasswd	バインド・パスワード。LDAP バインドの場合、このパスワードはセキュリティ資格証明として使用されます。 このファイルではパスワードを指定しないことをお薦めします。	いいえ	-
odip.bootstrap.mapfile	属性とドメインのマッピングを含むマップ・ファイルの位置。	いいえ	-

表 A-20 ブートストラップ・プロパティ（続き）

プロパティ	説明	必須	デフォルト
odip.bootstrap.logfile	ログ・ファイルの位置。このファイルがすでに存在する場合は、追加されます。デフォルトのログ・ファイルは、$*ORACLE_HOME*/ldap/odi/log ディレクトリに作成された bootstrap.log です。	いいえ	ディレクトリ $*ORACLE_HOME*/ldap/odi/ に作成されたファイル bootstrap.log
odip.bootstrap.logseverity	記録する必要があるログ・メッセージのタイプ。 INFO - 1 WARNING - 2 DEBUG – 4 ERROR - 8 **注意：**これらのタイプの組合せも使用されます。たとえば、WARNING および ERROR メッセージのみを記録する場合、8+2（つまり 10）と指定します。同様に、すべてのタイプのメッセージを記録する場合は、1 + 2 + 4 + 8 = 15 と指定します。	いいえ	1 + 8 = 9
odip.bootstrap.loadparallelism	処理されたデータを宛先にロードするために使用する Writer スレッドの数を示す数値。	いいえ	1-
odip.bootstrap.loadretry	エントリのロードに失敗した場合の再試行回数を示すインジケータ。	いいえ	5
odip.bootstrap.trcfile	トレース・ファイルの位置。このファイルがすでに存在する場合は、上書きされます。	いいえ	$*ORACLE_HOME*/ldap/odi/log/bootstrap.trc
odip.bootstrap.trclevel	トレース・レベル	いいえ	3

Oracle Directory Integration and Provisioning Platform 管理者のパスワードの変更

dipadmin アカウントのデフォルトのパスワードは、インストール中に選択した ias_admin のパスワードと同じです。このコマンドを使用すると、dipadmin アカウントのパスワードを再設定できます。このパスワードを再設定するには、orcladmin アカウントのセキュリティ資格証明が必要です。

次に例を示します。

```
$ dipassistant chgpasswd -passwd orcladmin password -host oid.heman.com
-port 3060
```

Assistant によって、新しいパスワードの入力を求める次のようなプロンプトが表示されます。

```
New Password:
Confirm Password:
```

Oracle Internet Directory 内のすべての同期プロファイルの表示

listprofiles コマンドによって、Oracle Internet Directory 内のすべての同期プロファイルのリストが表示されます。次に例を示します。

```
$ dipassistant listprofiles -passwd dipadmin password -host oid.heman.com
-port 3060
```

このコマンドによって、次のサンプル・リストが表示されます。

```
IplanetExport
IplanetImport
ActiveImport
ActiveExport
LdifExport
LdifImport
TaggedExport
TaggedImport
OracleHRAgent
ActiveChgImp
```

注意： ここに示すリストは、インストール中に作成されたデフォルトのプロファイルのセットです。

特定の同期プロファイルの詳細の表示

showprofile コマンドによって、特定の同期プロファイルの詳細が表示されます。次に例を示します。

```
$ dipassistant showprofile -passwd dipadmin password -host oid.heman.com
-port 3060 -profile ActiveImport
```

このコマンドによって、次のサンプル出力が表示されます。

```
odip.profile.version = 1.0
odip.profile.lastchgnum = 0
odip.profile.interface = LDAP
odip.profile.oidfilter = orclObjectGUID
odip.profile.schedinterval = 60
odip.profile.name = ActiveImport
odip.profile.syncmode = IMPORT
odip.profile.retry = 5
odip.profile.debuglevel = 0
odip.profile.status = DISABLE
```

Oracle Directory Integration Server に対する Wallet パスワードの設定

WPasswd コマンドを使用すると、Oracle Directory Integration Server が Oracle Internet Directory への接続時に使用する Wallet パスワードを指定できます。このコマンドを使用するには、次のように入力します。

```
dipassistant wp
```

Directory Integration and Provisioning Assistant によって、パスワードの入力および確認を求められます。

認証管理ノード間での統合プロファイルの移動

Directory Integration and Provisioning Assistant を使用して、ディレクトリ統合プロファイルを別のノードに移動し、それらを相互に再度関連付けることができます。たとえば、中間層コンポーネントが特定の Oracle Identity Management インフラストラクチャに関連付けられている場合、そのインフラストラクチャのノード内に存在するすべての統合プロファイルを新しいインフラストラクチャのノードに移動できます。

表 A-21 に、再度関連付ける場合の規則を示します。

表 A-21 ディレクトリ統合プロファイルを再度関連付ける場合の規則

状況	処置
統合プロファイルが 2 つ目の Oracle Internet Directory ノードに存在しない	統合プロファイルは、2 つ目の Oracle Internet Directory ノードにコピーされ、コピー完了後無効になります。これは、アプリケーションで有効にする必要があります。統合プロファイルの `lastchangenumber` 属性は、2 つ目の Oracle Internet Directory ノードの現行の最終変更番号に変更されます。
統合プロファイルが 2 つ目の Oracle Internet Directory ノードに存在する	両方の統合プロファイルが、次の方法で調停されます。 ■ ノード 1 のプロファイルの新しい属性がノード 2 のプロファイルに追加されます。 ■ 既存の同じ属性の場合は、ノード 1 のプロファイルの値によってノード 2 のプロファイルの属性が上書きされます。 ■ プロファイルは、コピー完了後無効になります。これは、アプリケーションで有効にする必要があります。 ■ 統合プロファイルの `lastchangenumber` 属性が 2 つ目の Oracle Internet Directory ノードの現行の最終変更番号に変更されます。

再度関連付けるには、次のとおり入力します。

```
dipassistant reassociate [-src_ldap_host <hostName>]
[-src_ldap_port <portNo>] [-src_ldap_dn <bindDn>] [-src_ldap_passwd
<password>] -dst_ldap_host <hostName> [-dst_ldap_port <portNo>]
[-dst_ldap_dn <bindDn>] [-dst_ldap_passwd <password>] [-log <logfile>]
Options:
-src_ldap_host <hostName> : Host where OID-1 runs
-src_ldap_port <portNo> : Port at which OID-1 runs
-src_ldap_dn <bindDn> : Bind Dn to connect to OID-1
-src_ldap_passwd <password> : Bind Dn password to connect to OID-1
-dst_ldap_host <hostName> : Host where OID-2 runs
-dst_ldap_port <portNo> : Port at which OID-2 runs
-dst_ldap_dn <bindDn> : Bind Dn to connect to OID-2
-dst_ldap_passwd <password> : Bind Dn password to connect to OID-2
-log <logFile> : Log file
```

デフォルト:

```
src_ldap_host - localhost, src_ldap_port & dst_ldap_port - 389
src_ldap_dn & dst_ldap_dn - cn=orcladmin account
```

例:

```
dipassistant reassociate -src_ldap_host oid1.mycorp.com ¥
-dst_ldap_host oid2.mycorp.com -src_ldap_passwd xxxx ¥
-dst_ldap_passwd xxxx

dipassistant rs -help
```

ログ・ファイルの位置を指定していない場合は、デフォルトで `$ORACLE_HOME/ldap/odi/log/reassociate.log` が作成されます。

Oracle Internet Directory 10*g*（9.0.4）での Directory Integration and Provisioning Assistant の制限

このリリースの Directory Integration and Provisioning Assistant では、次の機能はサポートされません。

- Oracle Internet Directory に対する SSL ベースの認証
- スキーマの同期化
- -cfg オプションが指定されているブートストラップ・プロセスの終了時に自動的に行われるプロファイルの作成
- マッピング・ファイルの検証
- 不具合があったエントリ・ファイルの作成

Directory Integration and Provisioning Assistant の次の要素はテストされていません。

- SSL 接続を介した接続ディレクトリのブートストラップ
- プロファイルに対して同期が行われている場合の modifyprofile コマンドの使用

表 A-22 に、Directory Integration and Provisioning Assistant のブートストラップ・コマンドの制限を示します。

表 A-22 Directory Integration and Provisioning Assistant でのブートストラップの制限

ブートストラップのタイプ	制限
LDIF-to-LDIF	なし
LDAP-to-LDIF	エントリが多数の場合、ブートストラップは、サイズ制限超過のエラーを返して正常に実行されない場合があります。この問題を解決するには、ブートストラップの実行元であるサーバーで次のことが必要です。 ■ ページ化された結果の制御（OID 1.2.840.113556.1.4.319）をサポート。現在、Microsoft Active Directory のみがこの制御をサポートする LDAP ディレクトリです。 ■ サーバー側のサイズ制限パラメータに適切な値を設定する。 ■ 独自のインポート / エクスポート・ツールを使用し、データのダンプを取得して、LDIF-to-LDIF または LDIF-to-LDAP を使用してブートストラップを実行する。
LDIF -to-LDAP	なし
LDAP-to-LDAP	LDAP-to-LDIF と同様

ldapUploadAgentFile.sh ツールの構文

ディレクトリの同期時に、LdapUploadAgentFile.sh を使用してマッピング情報と構成情報をロードします。

```
ldapUploadAgentFile.sh -name  profile_name
-config configset_the_profile_is_associated_with
-LDAPhost  directory_server_host
-LDAPport  directory_server_port
-binddn  DN_that_can_modify_the_profile   >
-bindpass password_for_the_bind_DN
-attrtype  "MAP" | "ATTR"
-filename complete_path_of_file_to_be_uploaded
```

表 A-23 ldapUploadAgentFile.sh の引数

引数	説明
Name	情報のロードが必要な統合プロファイルの名前。
Config	プロファイルが属している configset。
LDAPhost	ディレクトリ・サーバーのホスト。

表 A-23 ldapUploadAgentFile.sh の引数（続き）

引数	説明
LDAPport	ディレクトリ・サーバーのポート。
Binddn	プロファイル・エントリを変更するためのアクセス権限を持つディレクトリ・ユーザーのバインド識別名。デフォルトは cn=orcladmin です。
Bindpass	バインド識別名に対応するパスワード。デフォルトは welcome です。
AttrType	ロードするファイルのタイプ。マッピング・ファイルをロードする場合は、「MAP」を指定します。構成情報ファイルをロードする場合は、「ATTR」を指定します。
Filename	アップロードするファイルの完全パス名。

注意： Directory Integration and Provisioning Assistant を使用して、次の操作を実行することもできます。次のいずれかの値を入力します。

```
dipassistant mp [options] odip.profile.mapfile=your map
file


dipassistant mp [options] odip.profile.configfile= your
configuration file
```

関連項目： ldapUploadAgentFile.sh を使用する場合の詳細は、『Oracle Internet Directory 管理者ガイド』の第 33 章を参照してください。

ldapCreateConn.sh ツール構文

このコマンドライン・ツール ldapcreateConn.sh を使用すると、統合プロファイルを作成できます。このツールは、次のディレクトリにあります。

$*ORACLE_HOME*/ldap/admin/

次の例では、「HRMS」という名前の統合プロファイルを構成設定 2 で作成します。

```
ldapcreateConn.sh
        -name agent_name>
        [ -type  <IMPORT | EXPORT > ] ¥
        [ -agentpwd agent_password ] ¥
        [ -config configset_to_associate_with ] ¥
        [ -LDAPhost directory_server_host ]
        [ -LDAPport directory_server_port ] ¥
```

```
[ -binddn DN_of_super_user] ¥
[ -bindpass   Bind_password ] ¥
[ [-retry   maximum_retry_count_on_synchronization_errors ] ¥
[ -poll polling_interval_for_synchronization ] ¥
[ -host host_on_which_to_run_agent ] ¥
[ -conndirurl  connected_directory_URL ] ¥
[ -conndiracct connected_directory_account_information ] ¥
[ -conndirpwd connected_directory_account_password ] ¥
[ -execmd command_line_for_the_agent ]  ¥
[ -iftype interface_type ]      ¥
[ -condirfilter connected_directory_matching_filter ]¥
[ -oidfilter OID_matching_filter ] ¥
[ -U SSL_authentication_mode ]
[ -W wallet_location ]¥
[ -P wallet_password ]
```

表 A-24 ldapcreateConn.sh を使用して登録するための引数

引数	説明
Name	統合プロファイルの名前。一意である必要があります。
Type	IMPORT または EXPORT。デフォルトは IMPORT です。
Agentpwd	プロファイルを保護するためのパスワード。デフォルトは welcome です。
Config	構成設定番号。デフォルトは 1 です。
LDAPhost	ディレクトリ・サーバーのホスト。デフォルトは現行のホストです。
LDAPport	ディレクトリ・サーバー・ポート。デフォルトはポート 389 です。
Binddn	統合プロファイルの作成権限を持つディレクトリ・ユーザーのバインド識別名。デフォルトは cn=orcladmin です。
Bindpass	バインド・パスワード。デフォルトは welcome です。
Retry	サーバーによる同期エラーの検出によって実行される再試行の最大回数。デフォルトは 5 です。
Poll	プロファイルのスケジューリング間隔。デフォルトは 60 秒です。
Host	現在使用されています。暫定的に、DIP サーバーが実行されているマシンの名前に設定する必要があります。
Conndirurl	接続ディレクトリのアクセス情報。
Conndiracct	接続ディレクトリ・アカウント。
Conndirpwd	接続ディレクトリ・アカウントのパスワード。

表 A-24 ldapcreateConn.sh を使用して登録するための引数（続き）

引数	説明
Execmd	パートナ・エージェントを実行するための OS コマンドライン。
Iftype	インタフェース・タイプ。デフォルトは TAGGED です。
Condirfilter	接続ディレクトリの照合フィルタ。
Oidfilter	OID 照合フィルタ。

注意： Directory Integration and Provisioning Assistant の createprofile オプションを使用してもこの操作を実行できます。

ldapDeleteConn.sh ツール構文

同期プロファイルは、コマンドライン・ツール ldapDeleteConn.sh を使用して登録解除できます。このツールは、ディレクトリ *$ORACLE_HOME*/ldap/admin/ にあります。

構文は次のとおりです。

```
ldapdeleteConn.sh [ -name Profile_Name ]
 -LDAPhost <LDAP server host> (default is local host)]
            [ -LDAPport directory_server_port> (default 389)]
            [ -binddn SuperUserDN (default cn=orcladmin ) ]
            [ -bindpass   password (default=welcome) ]
            [ -config configset_associated_with_agent ]
            [ -U <SSL_authentication_mode> ]
            [ -W Wallet_location ]
            [ -P Wallet_password ]
            [ -help | -usage ]
```

次の例では、プロファイル・エントリを登録解除し、構成設定 2（config 2）のエントリから分離します。

```
ldapDeleteConn.sh name HRMS config 2
```

注意： Directory Integration and Provisioning Assistant の deleteprofile オプションを使用してもこの操作を実行できます。

StopOdiServer.sh ツールの構文

OID モニターおよび OIDCTL ツールを使用できないクライアントのみのインストール環境では、OIDCTL ツールを使用せずに Directory Integration and Provisioning Server を起動できます。サーバーを停止するには、stopOdiServer.sh ツールを使用します。

このツールのパス名は次のとおりです。
$*ORACLE_HOME*/ldap/admin/stopodiserver.sh

使用方法は次のとおりです。

```
$ORACLE_HOME/ldap/admin/stopodiserver.sh
        [ -LDAPhost LDAP_server_host ]
        [ -LDAPport LDAP_server_port ]
        [ -binddn super_user_dn (default cn=orcladmin ) ]
        [ -bindpass   bind_password (default=welcome) ]
        -instance instance_number_to_stop
```

表 A-25 Oracle Directory Integration Server を停止するための引数

引数	説明
LDAPhost	ディレクトリ・サーバーのホスト。デフォルトは現行のホストです。
LDAPport	ディレクトリ・サーバーのポート。デフォルトはポート 389 です。
Binddn	統合プロファイルの作成権限を持つディレクトリ・ユーザーのバインド識別名。デフォルトは cn=orcladmin です。
Bindpass	バインド・パスワード。デフォルトは welcome です。
Instance	停止する Oracle Directory Integration Server のインスタンス番号。

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 1.3.2.2-1 以上

  サイト : http://sources.redhat.com
- MKS Toolkit 6.1

  サイト : http://www.datafocus.com/

schemasync ツールの構文

schemasync ツールを使用すると、Oracle ディレクトリ・サーバーとサード・パーティの LDAP ディレクトリとの間で、スキーマ要素（つまり、属性とオブジェクト・クラス）を同期化できます。

schemasync の使用方法は次のとおりです。

```
$ORACLE_HOME/bin/schemasync
        -srchost source_LDAP_directory
        -srcport source_LDAP_port_numbert
        -srcdn privileged_DN_in_source_directory_to_access_schema
        -srcpwd password
        -dsthost destination_LDAP_directory
        -dstport destination_LDAP_port
        -dstdn privileged_dn_in_destination_directory_to_access_schema
        -dstpwd password
        [-ldap]
```

注意： -ldap パラメータはオプションです。このパラメータを指定した場合、スキーマの変更は、ソース LDAP ディレクトリから接続先 LDAP ディレクトリに直接適用されます。また、このパラメータを指定しない場合、スキーマの変更は、次の LDIF ファイルに格納されます。

- $*ORACLE_HOME*/ldap/odi/data/attributetypes.ldif
  このファイルには、新規属性の定義が格納されます。
- $*ORACLE_HOME*/ldap/odi/data/objectclasses.ldif
  このファイルには、新規オブジェクト・クラスの定義が格納されます。

-ldap を指定しない場合は、ldapmodify を使用して、これらの 2 つのファイルから、属性の型、オブジェクト・クラスの順に定義をアップロードする必要があります。

スキーマの同期中に発生したエラーは、次のログ・ファイルに記録されます。

- $*ORACLE_HOME*/ldap/odi/log/attributetypes.log
- $*ORACLE_HOME*/ldap/odi/log/objectclasses.log

Oracle Directory Integration Server の登録ツール（odisrvreg）

Oracle Directory Integration Server をディレクトリに登録する場合は、このツールで、ディレクトリにエントリが作成され、Directory Integration and Provisioning Server 用のパスワードが設定されます。登録エントリがすでに存在する場合は、このツールを使用して既存のパスワードを再設定できます。また、odisrvreg ツールは、$*ORACLE_HOME*/ldap/odi/conf に odisrvwallet_*hostname* と呼ばれるローカル・ファイルも作成します。このファイルは、Directory Integration and Provisioning Server のプライベート Wallet として機能し、Directory Integration and Provisioning Server はこのファイルを起動時に使用して、ディレクトリにバインドします。

表 A-26 に、Oracle Directory Integration Server 登録ツールで使用するパラメータを示します。odisrvreg を SSL モードで実行し、-U、-W および -P パラメータを使用して、ツールとディレクトリ間の通信を完全に保護することもできます。この 3 つのパラメータについても、表 A-26 に示します。

Directory Integration and Provisioning Server を登録するには、次のコマンドを入力します。

```
odisrvreg -h host_name -p port -D binddn -w bindpasswd –I passwd [-U ssl_mode -W
wallet –P wallet_password]
```

表 A-26 ODISRVREG の引数の説明

引数	説明
-h *host_name*	Oracle ディレクトリ・サーバーのホスト名。
-p *port_number*	ディレクトリ・サーバーが実行されているポート番号。
-D *binddn*	バインド識別名。バインド識別名には、Directory Integration and Provisioning Server の登録エントリを作成する認可が必要です。
-lhost	コールド・フェイルオーバー・クラスタ構成の仮想ホスト名。
-w *bindpasswd*	バインド・パスワード。
-U *SSL mode*	認可なしの場合は 0（ゼロ）を指定します。認可する場合は、1 を指定します。
-W *Wallet location*	SSL 証明書が格納される Oracle Wallet の位置。
-P *Wallet password*	Oracle Wallet をオープンするための Wallet パスワード。

プロビジョニング・サブスクリプション・ツール （oidprovtool）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。具体的には、次の操作の実行に使用します。

- 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、Oracle Directory Integration and Provisioning Platform で処理することができます。
- 既存のプロビジョニング・プロファイルの無効化。
- 無効なプロビジョニング・プロファイルの有効化。
- 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。コール元からは、アプリケーションとサブスクライバの組合せによって、プロビジョニング・プロファイルを一意に識別します。システムには、サブスクライバごとに、1 つのアプリケーションに 1 つのプロビジョニング・プロファイルのみ存在できるという制約があります。

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 1.3.2.2-1 以上

  サイト : `http://sources.redhat.com`

- MKS Toolkit 6.1

  サイト : `http://www.datafocus.com/`

実行可能ファイルの名前は 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・ツールが受け入れるパラメータは次のとおりです。

表 A-27 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

Name	説明	操作	必須 / オプション
operation	実行するサブスクリプション操作。このパラメータに指定できる値は、「create」、「enable」、「disable」、「delete」、「status」および「reset」です。ツールを起動するたびに 1 つの操作のみ実行できます。	すべて	必須
ldap_host	サブスクリプション操作を実行するディレクトリ・サーバーのホスト名。指定しない場合は、デフォルト値の localhost が使用されます。	すべて	オプション
profile_status	プロファイルのステータス（ENABLED/ DISABLED）。デフォルトは ENABLED です。	作成	オプション
profile_mode	IBOUND/OUTBOUND/BOTH。デフォルトは OUTBOUND です。	作成	オプション
profile_debug	プロファイルが Oracle Directory Integration Server によって実行されるデバッグ・レベル。	すべて	オプション
sslmode	プロビジョニング・サブスクリプション・ツールを SSL モードで実行するかどうかを示します。値 0 は非 SSL モードを、値 1 は SSL モードを示します。	すべて	オプション
ldap_port	LDAP サーバーが要求をリスニングする TCP/IP ポート。指定しない場合は、デフォルト値の 389 が使用されます。	すべて	オプション

表 A-27 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）

Name	説明	操作	必須 / オプション
ldap_user_dn	ユーザーのかわりに操作が実行される場合、そのユーザーの 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権限があるわけではありません。LDAP 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権限を付与または制限する方法については、管理ガイドを参照してください。	すべて	必須
ldap_user_password	ユーザーのかわりに操作が実行される場合、そのユーザーのパスワード。	すべて	必須
application_dn	プロビジョニング・サブスクリプション操作が実行され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します。	すべて	必須
organization_dn	プロビジョニング・サブスクリプション操作が実行される組織の 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します。	すべて	必須
interface_name	PL/SQL パッケージのデータベース・スキーマ名。値の形式は、[Schema].[PACKAGE_NAME] です。	作成のみ	必須
interface_type	イベントを伝播する必要があるインタフェースのタイプ。有効な値は PLSQL です（指定しない場合は、これがデフォルトとして使用されます）。	作成のみ	オプション
interface_connect_info	データベース接続文字列。この文字列の形式は、[HOST]:[PORT]:[SID]:[USER_ID]:[PASSWORD] です。	作成のみ	必須

表 A-27 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）

Name	説明	操作	必須 / オプション
interface_version	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 は古いインタフェースです。指定しない場合は、これがデフォルトとして使用されます。	作成のみ	オプション
interface_additional_info	インタフェースに関する追加情報。現在は使用されていません。	作成のみ	オプション
schedule	このプロファイルに関するスケジューリング情報。この値は、DIP がこのプロファイルを処理するまでの間隔の秒数です。指定しない場合は、デフォルト値の 3600 が使用されます。	作成のみ	オプション
max_retries	プロビジョニング・サービスが、失敗したイベント送信を再試行する回数。指定しない場合は、デフォルト値の 5 が使用されます。	作成のみ	オプション
event_subscription	DIP がこのアプリケーションに通知を送信する必要があるイベント。この文字列の形式は、「[USER]GROUP]:[対象のドメイン >]:[DELETE]ADD]MODIFY(< カンマで区切られた属性名のリスト >)]」です。異なる値を持つパラメータを複数回リストに含めると、複数の値を指定できます。指定しない場合は、デフォルトとして、USER:< 組織識別名 >:DELETE GROUP:< 組織識別名 >:DELETE が使用されます。つまり、組織識別名に属するユーザーとグループの削除通知が送信されます。	作成のみ	オプション

B

使用例

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。

- DBMS_LDAP サンプル・コード
- DBMS_LDAP_UTL サンプル・コード
- Java サンプル・コード

DBMS_LDAP サンプル・コード

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

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 `plsql` 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=F*IRST_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) constitue 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_elmt);
            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;
/
```

DBMS_LDAP_UTL サンプル・コード

この項では、次の項目について説明します。

- 例 : ユーザー関連ファンクション
- 例 : プロパティ関連サブプログラム
- 例 : サブスクライバ関連ファンクション
- 例 : グループ関連ファンクション

例 : ユーザー関連ファンクション

これは、DBMS_LDAP_UTL パッケージ内のユーザー関連ファンクションの使用例です。識別名、GUID またはユーザーを表す単純な名前を使用して、ユーザー・ハンドルを作成できます。

このサンプル・プログラムは、次のユーザー関連ファンクションを示すものです。

- DBMS_LDAP_UTL.create_user_handle()
- DBMS_LDAP_UTL.set_user_handle_properties()
- DBMS_LDAP_UTL.authenticate_user()
- DBMS_LDAP_UTL.get_user_properties()
- DBMS_LDAP_UTL.set_user_properties()

```
set serveroutput on size 30000

DECLARE

 ldap_host      VARCHAR2(256);
 ldap_port      PLS_INTEGER;
 ldap_user      VARCHAR2(256);
 ldap_passwd    VARCHAR2(256);
 ldap_base      VARCHAR2(256);

 retval              PLS_INTEGER;
 my_session          DBMS_LDAP.session;

 subscriber_handle   DBMS_LDAP_UTL.HANDLE;
 sub_type            PLS_INTEGER;
 subscriber_id       VARCHAR2(2000);

 my_pset_coll        DBMS_LDAP_UTL.PROPERTY_SET_COLLECTION;
 my_property_names   DBMS_LDAP.STRING_COLLECTION;
 my_property_values  DBMS_LDAP.STRING_COLLECTION;

 user_handle         DBMS_LDAP_UTL.HANDLE;
 user_id             VARCHAR2(2000);
 user_type           PLS_INTEGER;
 user_password       VARCHAR2(2000);

 my_mod_pset         DBMS_LDAP_UTL.MOD_PROPERTY_SET;


 my_attrs            DBMS_LDAP.STRING_COLLECTION;
```

```
BEGIN


 -- Please customize the following variables as needed

 ldap_host     :=   NULL ;
 ldap_port     :=   389;
 ldap_user     :=   'cn=orcladmin';
 ldap_passwd   :=   'welcome';

 sub_type      :=   DBMS_LDAP_UTL.TYPE_DN;
 subscriber_id :=   'o=acme,dc=com';
 user_type     :=   DBMS_LDAP_UTL.TYPE_DN;
 user_id       :=   'cn=user1,cn=users,o=acme,dc=com';
 user_password :=   'welcome';


 -- Choosing exceptions to be raised by DBMS_LDAP library.
 DBMS_LDAP.USE_EXCEPTION := TRUE;

 ------------------------------------------------
 -- Connect to the LDAP server
 -- and obtain and ld session.
 ------------------------------------------------

 my_session := DBMS_LDAP.init(ldap_host,ldap_port);

 ------------------------------------------------
 -- Bind to the directory
 --
 ------------------------------------------------

 retval := DBMS_LDAP.simple_bind_s(my_session,
                                   ldap_user,
                                   ldap_passwd);

 ------------------------------------------------------------------------
 -- Create Subscriber Handle
 --
 ------------------------------------------------------------------------

 retval := DBMS_LDAP_UTL.create_subscriber_handle(subscriber_handle,
                                                  sub_type,
                                                  subscriber_id);

 IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
```

```
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_subscriber_handle returns : ' || to_char(retval));
END IF;

-------------------------------------------------------------------
-- Create User Handle
--
-------------------------------------------------------------------

retval := DBMS_LDAP_UTL.create_user_handle(user_handle,user_type,user_id);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_user_handle returns : ' || to_char(retval));
END IF;

-------------------------------------------------------------------
-- Set user handle properties
-- (link subscriber to user )
-------------------------------------------------------------------

retval := DBMS_LDAP_UTL.set_user_handle_properties(user_handle,
                                        DBMS_LDAP_UTL.SUBSCRIBER_HANDLE,
                                        subscriber_handle);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('set_user_handle_properties returns : ' ||
to_char(retval));
END IF;

-------------------------------------------------------------------
-- Authenticate User
--
-------------------------------------------------------------------

retval := DBMS_LDAP_UTL.authenticate_user(my_session,
                                         user_handle,
                                         DBMS_LDAP_UTL.AUTH_SIMPLE,
                                         user_password,
                                         NULL);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('authenticate_user returns : ' || to_char(retval));
END IF;
```

```
-------------------------------------------------------------------
-- Retrieve User Properties
--
-------------------------------------------------------------------
-- like .. telephone number

my_attrs(1) := 'telephonenumber';

retval := DBMS_LDAP_UTL.get_user_properties(my_session,
                                        user_handle,
                                        my_attrs,
                                        DBMS_LDAP_UTL.ENTRY_PROPERTIES,
                                        my_pset_coll);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('get_user_properties returns : ' || to_char(retval));
END IF;

-------------------------------------------------------------------
-- Modifying User Properties
--
-------------------------------------------------------------------


retval := DBMS_LDAP_UTL.create_mod_propertyset(DBMS_LDAP_UTL.ENTRY_PROPERTIES,
                                           NULL,my_mod_pset);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_mod_propertyset returns : ' || to_char(retval));
END IF;


my_property_values.delete;
my_property_values(1) := '444-6789';
retval := DBMS_LDAP_UTL.populate_mod_propertyset(my_mod_pset,
                                           DBMS_LDAP_UTL.REPLACE_PROPERTY,
                                           'telephonenumber',my_property_values);
my_property_values.delete;

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('populate_mod_propertyset returns : ' || to_char(retval));
END IF;
```

```
retval := DBMS_LDAP_UTL.set_user_properties(my_session,user_handle,
                                            DBMS_LDAP_UTL.ENTRY_PROPERTIES,
                                            my_mod_pset,
                                            DBMS_LDAP_UTL.MODIFY_PROPERTY_SET);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('set_user_properties returns : ' || to_char(retval));
END IF;


----------------------------------------
-- Free Mod Propertyset
--
----------------------------------------

DBMS_LDAP_UTL.free_mod_propertyset(my_mod_pset);


------------------------------------------------------------------------
-- Free handles
--
------------------------------------------------------------------------

DBMS_LDAP_UTL.free_handle(subscriber_handle);
DBMS_LDAP_UTL.free_handle(user_handle);


 -- unbind from the directory
retval := DBMS_LDAP.unbind_s(my_session);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('unbind_s returns : ' || to_char(retval));
END IF;


-- 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;
/
```

例 : プロパティ関連サブプログラム

次のサンプル・コードは、DBMS_LDAP_UTL パッケージのプロパティ関連サブプログラムの使用方法を示しています。ユーザー・ハンドル、サブスクライバ・ハンドルおよびグループ・ハンドルに関連するサブプログラムのほとんどは、DBMS_LDAP_UTL.PROPERTY_SET_COLLECTION を戻します。

PROPERTY_SET_COLLECTION には、一連の PROPERTY_SET が含まれます。PROPERTY_SET は、識別名で識別される LDAP エントリに類似しています。各 PropertySet には、ゼロ個以上の Property のセットが含まれます。Property は LDAP エントリの特定の属性に類似しており、1 つ以上の値を含む可能性があります。

```
set serveroutput on size 30000

DECLARE

 ldap_host      VARCHAR2(256);
 ldap_port      PLS_INTEGER;
 ldap_user      VARCHAR2(256);
 ldap_passwd    VARCHAR2(256);
 ldap_base      VARCHAR2(256);

 retval              PLS_INTEGER;
 my_session          DBMS_LDAP.session;

 subscriber_handle   DBMS_LDAP_UTL.HANDLE;
 sub_type            PLS_INTEGER;
 subscriber_id       VARCHAR2(2000);

 my_pset_coll        DBMS_LDAP_UTL.PROPERTY_SET_COLLECTION;
 my_property_names   DBMS_LDAP.STRING_COLLECTION;
 my_property_values  DBMS_LDAP.STRING_COLLECTION;

 user_handle         DBMS_LDAP_UTL.HANDLE;
 user_id             VARCHAR2(2000);
 user_type           PLS_INTEGER;
 user_password       VARCHAR2(2000);

 my_mod_pset         DBMS_LDAP_UTL.MOD_PROPERTY_SET;


 my_attrs            DBMS_LDAP.STRING_COLLECTION;

BEGIN


 -- Please customize the following variables as needed
```

```
ldap_host     := NULL ;
ldap_port     := 389;
ldap_user     := 'cn=orcladmin';
ldap_passwd   := 'welcome';

sub_type      := DBMS_LDAP_UTL.TYPE_DN;
subscriber_id := 'o=acme,dc=com';
user_type     := DBMS_LDAP_UTL.TYPE_DN;
user_id       := 'cn=user1,cn=users,o=acme,dc=com';
user_password := 'welcome';


-- Choosing exceptions to be raised by DBMS_LDAP library.
DBMS_LDAP.USE_EXCEPTION := TRUE;

-----------------------------------------------
-- Connect to the LDAP server
-- and obtain and ld session.
-----------------------------------------------

my_session := DBMS_LDAP.init(ldap_host,ldap_port);

-----------------------------------------------
-- Bind to the directory
--
-----------------------------------------------

retval := DBMS_LDAP.simple_bind_s(my_session,
                                  ldap_user,
                                  ldap_passwd);

-------------------------------------------------------------------------
-- Create Subscriber Handle
--
-------------------------------------------------------------------------

retval := DBMS_LDAP_UTL.create_subscriber_handle(subscriber_handle,
                                                 sub_type,
                                                 subscriber_id);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_subscriber_handle returns : ' || to_char(retval));
END IF;

-------------------------------------------------------------------------
-- Create User Handle
```

```
--
-----------------------------------------------------------------------

retval := DBMS_LDAP_UTL.create_user_handle(user_handle,user_type,user_id);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_user_handle returns : ' || to_char(retval));
END IF;

-----------------------------------------------------------------------
-- Set user handle properties
-- (link subscriber to user )
-----------------------------------------------------------------------

retval := DBMS_LDAP_UTL.set_user_handle_properties(user_handle,
                                        DBMS_LDAP_UTL.SUBSCRIBER_HANDLE,
                                        subscriber_handle);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('set_user_handle_properties returns : ' ||
to_char(retval));
END IF;

-----------------------------------------------------------------------
-- Retrieve User Properties
--
-----------------------------------------------------------------------
-- like .. telephone number

my_attrs(1) := 'telephonenumber';

retval := DBMS_LDAP_UTL.get_user_properties(my_session,
                                      user_handle,
                                      my_attrs,
                                      DBMS_LDAP_UTL.ENTRY_PROPERTIES,
                                      my_pset_coll);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('get_user_properties returns : ' || to_char(retval));
END IF;

-----------------------------------------------------------------------
-- Print properties obtained for the user.
--
```

```
-------------------------------------------------------------------
 IF my_pset_coll.count > 0 THEN


     FOR i in my_pset_coll.first .. my_pset_coll.last LOOP

      retval := DBMS_LDAP_UTL.get_property_names(my_pset_coll(i),
                                                 my_property_names);
        IF my_property_names.count > 0 THEN

          FOR j in my_property_names.first .. my_property_names.last LOOP
            retval := DBMS_LDAP_UTL.get_property_values(my_pset_coll(i),
                                                        my_property_names(j),
                                                        my_property_values);
            IF my_property_values.COUNT > 0 THEN
               FOR k in my_property_values.FIRST..my_property_values.LAST LOOP

                 DBMS_OUTPUT.PUT_LINE( my_property_names(j) || ' : ' ||
                        my_property_values(k));

               END LOOP;
            END IF;

          END LOOP;

        END IF; -- IF my_property_names.count > 0

     END LOOP;

 END IF; -- If my_pset_coll.count > 0


  -- Free my_properties
  IF my_pset_coll.count > 0 then
    DBMS_LDAP_UTL.free_propertyset_collection(my_pset_coll);
  end if;

-------------------------------------------------------------------
-- Free handles
--
-------------------------------------------------------------------

DBMS_LDAP_UTL.free_handle(subscriber_handle);
DBMS_LDAP_UTL.free_handle(user_handle);


 -- unbind from the directory
```

```
 retval := DBMS_LDAP.unbind_s(my_session);

 IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
    -- Handle Errors
    DBMS_OUTPUT.PUT_LINE('unbind_s returns : ' || to_char(retval));
 END IF;


-- 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;
/
```

例 : サブスクライバ関連ファンクション

これは、DBMS_LDAP_UTL パッケージ内のサブスクライバ関連ファンクションの使用例です。識別名、GUID またはサブスクライバを表す単純な名前を使用して、サブスクライバ・ハンドルを作成できます。

このサンプル・プログラムは、次のサブスクライバ関連ファンクションを示すものです。

- DBMS_LDAP_UTL.create_subscriber_handle()
- DBMS_LDAP_UTL.get_subscriber_properties()

```
set serveroutput on size 30000

DECLARE

 ldap_host      VARCHAR2(256);
 ldap_port      PLS_INTEGER;
 ldap_user      VARCHAR2(256);
 ldap_passwd    VARCHAR2(256);
 ldap_base      VARCHAR2(256);

 retval              PLS_INTEGER;
 my_session          DBMS_LDAP.session;

 subscriber_handle   DBMS_LDAP_UTL.HANDLE;
 sub_type            PLS_INTEGER;
 subscriber_id       VARCHAR2(2000);

 my_pset_coll        DBMS_LDAP_UTL.PROPERTY_SET_COLLECTION;
```

```
 my_property_names   DBMS_LDAP.STRING_COLLECTION;
 my_property_values  DBMS_LDAP.STRING_COLLECTION;

 user_handle         DBMS_LDAP_UTL.HANDLE;
 user_id             VARCHAR2(2000);
 user_type           PLS_INTEGER;
 user_password       VARCHAR2(2000);

 my_mod_pset         DBMS_LDAP_UTL.MOD_PROPERTY_SET;


 my_attrs            DBMS_LDAP.STRING_COLLECTION;

BEGIN


 -- Please customize the following variables as needed

 ldap_host     := NULL ;
 ldap_port     := 389;
 ldap_user     := 'cn=orcladmin';
 ldap_passwd   := 'welcome';

 sub_type      := DBMS_LDAP_UTL.TYPE_DN;
 subscriber_id := 'o=acme,dc=com';
 user_type     := DBMS_LDAP_UTL.TYPE_DN;
 user_id       := 'cn=user1,cn=users,o=acme,dc=com';
 user_password := 'welcome';


 -- Choosing exceptions to be raised by DBMS_LDAP library.
 DBMS_LDAP.USE_EXCEPTION := TRUE;

 ------------------------------------------------
 -- Connect to the LDAP server
 -- and obtain and ld session.
 ------------------------------------------------

 my_session := DBMS_LDAP.init(ldap_host,ldap_port);

 ------------------------------------------------
 -- Bind to the directory
 --
 ------------------------------------------------

 retval := DBMS_LDAP.simple_bind_s(my_session,
                                   ldap_user,
```

```
                                ldap_passwd);

-------------------------------------------------------------------
-- Create Subscriber Handle
--
-------------------------------------------------------------------

retval := DBMS_LDAP_UTL.create_subscriber_handle(subscriber_handle,
                                          sub_type,
                                          subscriber_id);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_subscriber_handle returns : ' || to_char(retval));
END IF;

-------------------------------------------------------------------
-- Retrieve Subscriber Properties
--
-------------------------------------------------------------------
-- like .. telephone number

my_attrs(1) := 'orclguid';

retval := DBMS_LDAP_UTL.get_subscriber_properties(my_session,
                                     subscriber_handle,
                                     my_attrs,
                                     DBMS_LDAP_UTL.ENTRY_PROPERTIES,
                                     my_pset_coll);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('get_subscriber_properties returns : ' || to_char(retval));
END IF;

-------------------------------------------------------------------
-- Free handle
--
-------------------------------------------------------------------

DBMS_LDAP_UTL.free_handle(subscriber_handle);


 -- unbind from the directory
retval := DBMS_LDAP.unbind_s(my_session);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
```

```
    -- Handle Errors
    DBMS_OUTPUT.PUT_LINE('unbind_s returns : ' || to_char(retval));
 END IF;


-- 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;
/
```

例 : グループ関連ファンクション

これは、DBMS_LDAP_UTL パッケージ内のグループ関連ファンクションの使用例です。識別名、GUID またはグループを表す単純な名前を使用して、グループ・ハンドルを作成できます。

このサンプル・プログラムは、次のグループ関連ファンクションを示すものです。

- DBMS_LDAP_UTL.create_group_handle()
- DBMS_LDAP_UTL.set_group_handle_properties()
- DBMS_LDAP_UTL.check_group_membership()
- DBMS_LDAP_UTL.get_group_membership()
- DBMS_LDAP_UTL.get_group_properties()

```
set serveroutput on size 30000

DECLARE

 ldap_host      VARCHAR2(256);
 ldap_port      PLS_INTEGER;
 ldap_user      VARCHAR2(256);
 ldap_passwd    VARCHAR2(256);
 ldap_base      VARCHAR2(256);

 retval              PLS_INTEGER;
 my_session          DBMS_LDAP.session;

 subscriber_handle   DBMS_LDAP_UTL.HANDLE;
 sub_type            PLS_INTEGER;
 subscriber_id       VARCHAR2(2000);
```

```
 my_pset_coll        DBMS_LDAP_UTL.PROPERTY_SET_COLLECTION;
 my_property_names   DBMS_LDAP.STRING_COLLECTION;
 my_property_values  DBMS_LDAP.STRING_COLLECTION;

 group_handle         DBMS_LDAP_UTL.HANDLE;
 group_id             VARCHAR2(2000);
 group_type           PLS_INTEGER;

 user_handle         DBMS_LDAP_UTL.HANDLE;
 user_id             VARCHAR2(2000);
 user_type           PLS_INTEGER;

 my_mod_pset         DBMS_LDAP_UTL.MOD_PROPERTY_SET;


 my_attrs            DBMS_LDAP.STRING_COLLECTION;

BEGIN


 -- Please customize the following variables as needed

 ldap_host     := NULL ;
 ldap_port     := 389;
 ldap_user     := 'cn=orcladmin';
 ldap_passwd   := 'welcome';

 sub_type      := DBMS_LDAP_UTL.TYPE_DN;
 subscriber_id := 'o=acme,dc=com';
 user_type     := DBMS_LDAP_UTL.TYPE_DN;
 user_id       := 'cn=user1,cn=users,o=acme,dc=com';
 group_type     := DBMS_LDAP_UTL.TYPE_DN;
 group_id       := 'cn=group1,cn=groups,o=acme,dc=com';


 -- Choosing exceptions to be raised by DBMS_LDAP library.
 DBMS_LDAP.USE_EXCEPTION := TRUE;

 ------------------------------------------------
 -- Connect to the LDAP server
 -- and obtain and ld session.
 ------------------------------------------------

 my_session := DBMS_LDAP.init(ldap_host,ldap_port);

 ------------------------------------------------
```

```
-- Bind to the directory
--
-----------------------------------------------

retval := DBMS_LDAP.simple_bind_s(my_session,
                                  ldap_user,
                                  ldap_passwd);

-----------------------------------------------------------------------
-- Create Subscriber Handle
--
-----------------------------------------------------------------------

retval := DBMS_LDAP_UTL.create_subscriber_handle(subscriber_handle,
                                                 sub_type,
                                                 subscriber_id);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_subscriber_handle returns : ' || to_char(retval));
END IF;

-----------------------------------------------------------------------
-- Create User Handle
--
-----------------------------------------------------------------------

retval := DBMS_LDAP_UTL.create_user_handle(user_handle,user_type,user_id);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('create_user_handle returns : ' || to_char(retval));
END IF;

-----------------------------------------------------------------------
-- Set User handle properties
-- (link subscriber to user )
-----------------------------------------------------------------------

retval := DBMS_LDAP_UTL.set_user_handle_properties(user_handle,
                                                   DBMS_LDAP_UTL.SUBSCRIBER_HANDLE,
                                                   subscriber_handle);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('set_user_handle_properties returns : ' ||
to_char(retval));
```

```
 END IF;

 ----------------------------------------------------------------------
 -- Create Group Handle
 --
 ----------------------------------------------------------------------

 retval := DBMS_LDAP_UTL.create_group_handle(group_handle,group_type,group_id);

 IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
    -- Handle Errors
    DBMS_OUTPUT.PUT_LINE('create_group_handle returns : ' || to_char(retval));
 END IF;

 ----------------------------------------------------------------------
 -- Set Group handle properties
 -- (link subscriber to group )
 ----------------------------------------------------------------------

 retval := DBMS_LDAP_UTL.set_group_handle_properties(group_handle,
                                          DBMS_LDAP_UTL.SUBSCRIBER_HANDLE,
                                          subscriber_handle);

 IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
    -- Handle Errors
    DBMS_OUTPUT.PUT_LINE('set_group_handle_properties returns : ' ||
to_char(retval));
 END IF;

 ----------------------------------------------------------------------
 -- Retrieve Group Properties
 --
 ----------------------------------------------------------------------
 -- like .. telephone number

 my_attrs(1) := 'uniquemember';

 retval := DBMS_LDAP_UTL.get_group_properties(my_session,
                                       group_handle,
                                       my_attrs,
                                       DBMS_LDAP_UTL.ENTRY_PROPERTIES,
                                       my_pset_coll);

 IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
    -- Handle Errors
    DBMS_OUTPUT.PUT_LINE('get_group_properties returns : ' || to_char(retval));
 END IF;
```

```
--------------------------------------
-- Check Group Membership
--
--------------------------------------

retval :=  DBMS_LDAP_UTL.check_group_membership( my_session,
                                 user_handle,
                                 group_handle,
                                 DBMS_LDAP_UTL.DIRECT_MEMBERSHIP);

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('check_group_membership returns : ' || to_char(retval));
END IF;

---------------------------------------
-- Get Group Membership
--
---------------------------------------

my_attrs.delete();
my_attrs(1) := 'cn';

retval := DBMS_LDAP_UTL.get_group_membership ( my_session,
                               user_handle,
                               DBMS_LDAP_UTL.DIRECT_MEMBERSHIP,
                               my_attrs,
                               my_pset_coll );

IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
   -- Handle Errors
   DBMS_OUTPUT.PUT_LINE('get_group_membership returns : ' || to_char(retval));
END IF;


-----------------------------------------------------------------------
-- Free handle
--
-----------------------------------------------------------------------

DBMS_LDAP_UTL.free_handle(subscriber_handle);
DBMS_LDAP_UTL.free_handle(user_handle);
DBMS_LDAP_UTL.free_handle(group_handle);


 -- unbind from the directory
```

```
 retval := DBMS_LDAP.unbind_s(my_session);

 IF retval != DBMS_LDAP_UTL.SUCCESS  THEN
    -- Handle Errors
    DBMS_OUTPUT.PUT_LINE('unbind_s returns : ' || to_char(retval));
 END IF;


-- 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;
/
```

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
- JNDI のサンプル・コード
- SASL ベース認証のサンプル・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 ProperSetCollection 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 systemmatic 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
```

JNDI のサンプル・コード

```
import javax.naming.*;
import javax.naming.directory.*;
import javax.naming.ldap.*;
import oracle.ldap.util.jndi.*;
import oracle.ldap.util.*;
import java.lang.*;
import java.util.*;

/*
* JNDI SASL Digest MD5 is available in JDK 1.4 and later
*/
public class LdapSaslDigestMD5
{
public static void main( String[] args)
throws Exception
{


System.out.println("port : " + args[1]);
System.out.println("bindDN : " + args[2]);
System.out.println("bindPwd: " + args[3]);

// Important note:
// The bindDN must be normalized before passing it to JNDI context
```

```
// For example: cn=smith,ou=oid,o=oracle,c=us
// (capital and space will not be accepted as a normalized dn)
// Right now we only support dn in only.
// uid form will be supported in the next release.

// The noralize dn call is a static method in Util.java.

String normDN = Util.normalizeDN(args[2]);

Hashtable hashtable = new Hashtable();

// Look through System Properties for Context Factory if available
// set the CONTEXT factory only if it has not been set
// in the environment - set default to com.sun.jndi.ldap.LdapCtxFactory
hashtable.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
hashtable.put(Context.PROVIDER_URL, "ldap://"+args[0]+":"+args[1]);
// Set security authentication context to Digest MD5
hashtable.put(Context.SECURITY_AUTHENTICATION, "DIGEST-MD5");
hashtable.put(Context.SECURITY_PRINCIPAL, normDN );
hashtable.put(Context.SECURITY_CREDENTIALS, args[3] );
hashtable.put("java.naming.security.sasl.realm", "");
LdapContext ctx = new InitialLdapContext(hashtable,null);
System.out.println("sasl bind successful");

// Some search after the SASL bind has been done
PropertySetCollection psc = Util.ldapSearch(ctx,"","objectclass=*",
SearchControls.OBJECT_SCOPE,
new String[] {"supportedSASLmechanism"});

Util.printResults(psc);

System.exit(0);

}
}


/*
* Sample code Using JNDI/SASL EXTERNAL to connect to OID
* This code will work only with OID SSL setup in mutual authentication mode only.
* JNDI client needs to provide a client certificate that can be recognized by
* server side.
*/

import java.util.*;
import javax.naming.*;
```

```
import javax.naming.directory.*;
import oracle.security.jazn.spi.ldap.*;

public class LdapSaslExternal
{
public static void main (String[] args)
{
try {

Hashtable env = new Hashtable();

// Specify host and port to use for directory service
env.put("javax.net.debug", "all");
env.put("com.sun.jndi.ldap.trace.ber", System.out);
env.put("com.sun.naming.ldap.trace.ber", System.out);
env.put(Context.PROVIDER_URL, "ldap://some_url:5055/");

env.put("java.naming.security.protocol", "ssl");

System.setProperty("oracle.security.jazn.ldap.walletloc","<wallet_url>/ewallet.txt")
;

System.setProperty("oracle.security.jazn.ldap.walletpwd","welcome01");

// You can use any SSL Socket Factory of your implementation or toolkit

env.put("java.naming.ldap.factory.socket","oracle.security.jazn.spi.ldap.JAZNSSLSock
etFactoryImpl");

// specify authentication information
// Note: you can also set security authentication context to "SIMPLE" to
// connect to OID; however, this functionality supports for backward
// compatibility with LDAP version 2.
env.put(Context.SECURITY_AUTHENTICATION, "EXTERNAL"); // TO-DO: add secure
hannes
env.put(Context.SECURITY_PRINCIPAL, "cn=test,ou=security,o=oracle,c=us");
nv.put(Context.SECURITY_CREDENTIALS, "welcome"); // TO-DO: add SSL

env.put("java.naming.factory.initial", "com.sun.jndi.ldap.LdapCtxFactory");

// Set your own SSL Socket factory Impl class here.
System.getProperties().put("SSLSocketFactoryImplClass","oracle.security.jazn.spi.lda
p.JAZNSSLSocketFactoryImpl");

DirContext dirCtx = new InitialDirContext(env);
System.out.println("return from InitialDirContext");
Object obj = dirCtx.lookup("");
```

```
System.out.println("Looked up obj : " + obj);
} catch (Exception exp) {
exp.printStackTrace();
System.exit(-1);
}
}
}
```

SASL ベース認証のサンプル・コード

```
/* $Header: LdapSasl.java 05-may-2003.15:14:22 qdinh Exp $ */

/* Copyright (c) 2003, Oracle Corporation.  All rights reserved.  */

/*
   DESCRIPTION
    <short description of component this file declares/defines>

   PRIVATE CLASSES
    <list of private classes defined - with one-line descriptions>

   NOTES
    <other useful comments, qualifications, etc.>

   MODIFIED    (MM/DD/YY)
   *****       04/23/03 - Creation
 */

/**
 *  @version $Header: LdapSasl.java 05-may-2003.15:14:22 ***** Exp $
 *  @author  *****
 *  @since   release specific (what release of product did this appear in)
 */

package oracle.ldap.util.jndi;

import javax.naming.*;
import javax.naming.directory.*;
import javax.naming.ldap.*;
import oracle.ldap.util.jndi.*;
import oracle.ldap.util.*;
import java.lang.*;
import java.util.*;

public class LdapSasl
{
    public static void main( String[] args)
```

```
        throws Exception
    {


     System.out.println("port   : " + args[1]);
     System.out.println("bindDN : " + args[2]);
     System.out.println("bindPwd: " + args[3]);

     Hashtable hashtable = new Hashtable();

     // Look through System Properties for Context Factory if available
     // set the CONTEXT factory only if it has not been set
     // in the environment - set default to com.sun.jndi.ldap.LdapCtxFactory
     hashtable.put(Context.INITIAL_CONTEXT_FACTORY,
  "com.sun.jndi.ldap.LdapCtxFactory");

     hashtable.put(Context.PROVIDER_URL, "ldap://"+args[0]+":"+args[1]);

     //hashtable.put(Context.SECURITY_AUTHENTICATION, "simple");
     hashtable.put(Context.SECURITY_AUTHENTICATION, "DIGEST-MD5");
     hashtable.put(Context.SECURITY_PRINCIPAL, args[2] );
     hashtable.put(Context.SECURITY_CREDENTIALS, args[3] );
     hashtable.put("java.naming.security.sasl.realm", "");
     LdapContext ctx = new InitialLdapContext(hashtable,null);
     System.out.println("sasl bind successful");
     //PropertySetCollection psc =
Util.ldapSearch(ctx,"","objectclass=*",SearchControls.OBJECT_SCOPE,
     //new String[] {"supportedSASLmechanism"});

     //Util.printResults(psc);

     System.exit(0);

  }
}
```

C

DSML の構文

この付録では、次の項目について説明します。

- DSML の機能
- DSML 構文
- DSML で使用可能なツール

DSML の機能

ディレクトリ・サービスは、分散コンピューティングの主要部分を形成します。XML は、インターネット・アプリケーションの標準的なマークアップ言語になりつつあります。ディレクトリ・サービスがインターネットで普及するにつれ、ディレクトリ情報を XML データで表現することの必要性が急速に高まっています。この機能を使用すると、LDAP ディレクトリ・サーバーとの情報交換が必要な、LDAP を認識しないアプリケーションの種類の増加に対応できます。

Directory Services Markup Language（DSML）は、LDAP 情報および操作の XML 表現を定義します。LDAP Data Interchange Format（LDIF）は、ディレクトリ情報、またはディレクトリ・エントリに適用する一連の変更の伝達に使用します。前者は属性値レコード、後者は変更レコードと呼ばれます。

DSML の使用の利点

Oracle Internet Directory およびインターネット・アプリケーションで DSML を使用すると、異なるソースのデータを柔軟に統合できるようになります。また、DSML を使用すると、LDAP を使用しないアプリケーションで LDAP ベースのアプリケーションと通信でき、Oracle Internet Directory クライアント・ツールによって生成されたデータの操作やファイアウォールを経由したディレクトリへのアクセスが容易になります。

DSML は XML に基づき、Web での配信のために最適化されています。XML 形式の構造化データは均一で、アプリケーションまたはベンダーから独立しています。そのため、できるだけ多くの新しいフラット・ファイル・タイプの同期コネクタが作成されます。XML 形式にすると、ディレクトリ・データが中間層で使用可能になり、中間層でより有効な検索を実行できます。

DSML 構文

DSML バージョン 1 の文書は、ディレクトリ・エントリまたはディレクトリ・スキーマ（あるいはその両方）を定義します。各ディレクトリ・エントリには、識別名（DN）と呼ばれるディレクトリ全体で一意の名前があります。ディレクトリ・エントリには、ディレクトリ属性と呼ばれるプロパティ値のペアの番号があります。各ディレクトリ・エントリは、複数のオブジェクト・クラスのメンバーです。エントリのオブジェクト・クラスは、エントリが取得できるディレクトリ属性を制約します。このような制約は、ディレクトリ・スキーマで定義されます。ディレクトリ・スキーマは、同一の DSML 文書または別の文書に含まれます。

DSML バージョン 1 の `namespace` URI [9] は、`http://www.dsml.org/DSML` です。すべての XML 要素タグに、`dsml` という文字列の接頭辞（`namespace` 接頭辞）を付ける必要があります。

次の項では、DSML のトップレベルの構造およびディレクトリ・エントリとスキーマ・エントリの表現方法の概要を説明します。

トップレベルの構造

DSML のトップレベルでの文書要素の型は dsml です。次の型の子要素を持ちます。

```
directory-entries
directory-schema
```

また、子要素 *directory-entries* は、型 *entry* の子要素を持ちます。同様に、子要素 *directory-schema* は型 *class* および *attribute-type* の子要素を持ちます。

トップレベルでの DSML 文書の構造は次のとおりです。

```
<dsml:dsml xmlns:dsml=http://www.dsml.org/DSML>
<!- a document with directory & schema entries -->
  <dsml:directory-entries>
       <dsml:entry dn="...">...</dsml:entry>
         …..
  </dsml:directory-entries>
    ……
  <dsml:directory-schema>
   <dsml:class id="..." ...>...</dsml:class>
   <dsml:attribute-type id="..." ...>...</dsml:attribute-type>
   ……….
  </dsml:directory-schema>
</dsml:dsml>
```

ディレクトリ・エントリ

要素型 entry は、DSML 文書のディレクトリ・エントリを表します。entry 要素には、エントリのディレクトリ属性を表す要素が含まれます。エントリの識別名は、XML 属性 dn で指定します。

ディレクトリ・エントリを記述する XML エントリは次のとおりです。

```
<dsml:entry dn="uid=Heman, c=in, dc=oracle, dc=com">
  <dsml:objectclass>
    <dsml:oc-value>top</dsml:oc-value>
    <dsml:oc-value ref="#person">person</dsml:oc-value>
    <dsml:oc-value>organizationalPerson</dsml:oc-value>
    <dsml:oc-value>inetOrgPerson</dsml:oc-value>
  </dsml:objectclass>
  <dsml:attr name="sn">
       <dsml:value>Siva</dsml:value></dsml:attr>
  <dsml:attr name="uid">
       <dsml:value>Heman</dsml:value></dsml:attr>
```

```
  <dsml:attr name="mail">
       <dsml:value>Svenugop@Oracle.com</dsml:value></dsml:attr>
  <dsml:attr name="givenname">
       <dsml:value>Siva V. Kumar</dsml:value></dsml:attr>
  <dsml:attr name="cn">
       <dsml:value>Siva Kumar</dsml:value></dsml:attr>
```

oc-value's ref は、オブジェクト・クラスを定義するクラス要素への URI 参照です。この例の場合は、person オブジェクト・クラスを定義する要素への URI [9] 参照です。子要素 objectclass および attr は、ディレクトリ・エントリのオブジェクト・クラスおよび属性を指定するために使用します。

スキーマ・エントリ

要素型 class は、DSML 文書のスキーマ・エントリを表します。class 要素は、参照を容易にするために XML 属性 id を取ります。

たとえば、person オブジェクト・クラスのオブジェクト・クラス定義は次のようになります。

```
<dsml:class id="person" superior="#top"  type="structural">
    <dsml:name>person</dsml:name>
    <dsml:description>...</dsml:description>
    <dsml:object-identifier>2.5.6.6</object-identifier>
    <dsml:attribute ref="#sn" required="true"/>
    <dsml:attribute ref="#cn" required="true"/>
    <dsml:attribute ref="#userPassword" required="false"/>
    <dsml:attribute ref="#telephoneNumber" required="false"/>
    <dsml:attribute ref="#seeAlso" required="false"/>
    <dsml:attribute ref="#description" required="false"/>
</dsml:class>

In a similar way the directory attributes are also described. For example the
attribute definition for the cn attribute may look like the following:

<dsml:attribute-type id="cn">
    <dsml:name>cn</dsml:name>
    <dsml:description>...</dsml:description>
    <dsml:object-identifier>2.5.4.3</object-identifier>
    <dsml:syntax>1.3.6.1.4.1.1466.115.121.1.44</dsml:syntax>
</dsml:attribute-type>
```

DSML で使用可能なツール

XML フレームワークを使用すると、LDAP 以外のアプリケーションを使用してディレクトリ・データにアクセスできます。XML フレームワークはアクセス・ポイントを広範囲に定義し、次のツールを提供します。

- ldapadd
- ldapaddmt
- ldapsearch

関連項目： これらのツールの完全な構文と使用方法については、付録 A「エントリ管理コマンドライン・ツール構文」を参照してください。

Oracle Internet Directory クライアント・ツール `ldifwrite` は、ディレクトリ・データおよびスキーマの LDIF ファイルを生成します。これらの LDIF ファイルを XML ファイルに変換すると、XML ファイルはアプリケーション・サーバーに格納され、この XML ファイルに対して問合せが行われます。このように使用した場合、クライアントへの応答時間は、LDAP サーバーに対する LDAP 操作の実行時と比較すると大幅に短縮されます。

用語集

ACI

「**アクセス制御情報アイテム**」を参照。

ACL

「**アクセス制御リスト**」を参照。

ACP

「**アクセス制御ポリシー・ポイント**」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**」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**」を参照。

Directory Integration and Provisioning Server

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。

DIS

「**Directory Integration and Provisioning Server**」を参照。

DIT

「**ディレクトリ情報ツリー**」を参照。

DN

「**識別名**」を参照。

DRG

「**ディレクトリ・レプリケーション・グループ**」を参照。

DSA

「**ディレクトリ・システム・エージェント**」を参照。

DSE

「**ディレクトリ固有のエントリ**」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）

エントリがディレクトリに追加された場合に、システムによって生成され、エントリに挿入される識別子。マルチマイスターでレプリケートされた環境では、識別名ではなく GUID がエントリを一意に識別する。ユーザーは、エントリの GUID を変更できない。

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 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 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**」を参照。

MTS

「**共有サーバー**」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、第三世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 Delegated Administration Services

Oracle Delegated Administration Services ユニットと呼ばれる個々の事前定義済みサービスのセットで、ユーザーのかわりにディレクトリ操作を実行する。Oracle Internet Directory セルフ・サービス・コンソールによって、Oracle Internet Directory を使用する Oracle およびサード・パーティのアプリケーションの管理ソリューションを容易に開発および配置できる。

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

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 Identity Management

すべての企業識別情報および企業内の様々なアプリケーションへのアクセスを集中的かつ安全に管理するための配置を可能にするインフラストラクチャ。

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 Oracle9*i* の高度のパフォーマンス、スケーラビリティ、耐久性および可用性を組み合せたもの。

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。

関連項目：『Oracle Advanced Security 管理者ガイド』

Oracle9*i* Advanced 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 Oracle9*i* の機能。

peer-to-peer レプリケーション（peer-to-peer replication）

マルチマスター・レプリケーションまたは n-way レプリケーションとも呼ばれる。同等に機能する複数サイトがレプリケートされたデータのグループを管理できるようにするレプリケーションのタイプ。このようなレプリケーション環境では、各ノードはサプライヤ・ノードであると同時にコンシューマ・ノードであり、各ノードでディレクトリ全体がレプリケートされる。

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。

point-to-point レプリケーション（point-to-point replication）

ファンアウト・レプリケーション（fan-out replication）とも呼ばれる。サプライヤがコンシューマに直接レプリケートするレプリケーションのタイプ。コンシューマは 1 つ以上の他のコンシューマにレプリケートできる。レプリケーションには、完全レプリケーションと部分レプリケーションがある。

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整合性を実現している。

SGA

「**システム・グローバル領域**」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。

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-16

Unicode の 16 ビット・エンコーディング。Latin-1 文字は、この規格の最初の 256 コード・ポイントである。

UTF-8

文字ごとに連続した 1、2、3 または 4 バイトを使用する **Unicode** の可変幅 8 ビット・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、65536 以上の文字では 4 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 AL32UTF8（Unicode 3.1 規格用）となる。

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。

ファンアウト・レプリケーション（fan-out replication）

point-to-point レプリケーションとも呼ばれる。サプライヤがコンシューマに直接レプリケートするレプリケーションのタイプ。コンシューマは 1 つ以上の他のコンシューマにレプリケートできる。レプリケーションには、完全レプリケーションと部分レプリケーションがある。

アクセス制御情報アイテム（Access Control 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限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、属性に関係するコンテンツ・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。

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。

アクセス制御リスト（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。

アドバンスト・レプリケーション（Advanced Replication: AR）

「**Oracle9i Advanced Replication**」を参照。

アドバンスト・レプリケーション（ASR）

「**Oracle9i Advanced 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。

委任管理者（delegated administrator）

ホスティングされた環境では、アプリケーション・サービス・プロバイダなどの 1 企業が、他の複数の企業に Oracle コンポーネントを使用可能にして、その情報を格納する。この種の環境では、グローバル管理者はディレクトリ全体にまたがるアクティビティを実行する。委任管理者と呼ばれる他の管理者は、特定の認証管理レルムで、または特定のアプリケーションについてのロールを持つ。

インスタンス（instance）

「**ディレクトリ・サーバー・インスタンス**」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**」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 and Provisioning 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・パーテ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。

仮想 IP アドレス（virtual IP address）

コールド・フェイルオーバー・クラスタ構成では、各物理ノードに独自の物理 IP アドレスと物理ホスト名がある。外部に単一のシステム画像を公開する場合、クラスタはクラスタの物理ノードに移動できる動的 IP アドレスを使用する。これを仮想 IP アドレスと呼ぶ。

仮想ホスト名（virtual host name）

コールド・フェイルオーバーを行うクラスタ構成の場合、仮想 IP アドレスに対応するホスト名。

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。

クラスタ（cluster）

単一のコンピューティング・リソースとして使用される、相互接続された有効なコンピュータの集合。ハードウェア・クラスタによって、高可用性とスケーラビリティが得られる。

グループ検索ベース（group search base）

Oracle Internet Directory のデフォルトのディレクトリ情報ツリーで、すべてのグループを検索できる認証管理レルムのノード。

グローバル管理者（global administrator）

ホスティングされた環境では、アプリケーション・サービス・プロバイダなどの 1 企業が、他の複数の企業に Oracle コンポーネントを使用可能にして、その情報を格納する。この種の環境では、グローバル管理者はディレクトリ全体にまたがるアクティビティを実行する。

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鍵。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の **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 DN（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**」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、**Directory Integration and Provisioning Server** の制御下で実行されるエージェント。「**外部エージェント**」と対比。

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、信頼できる機関（**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ライ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。信頼できる証明書は、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。

スケーラビリティ（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。

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**」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。

セカンダリ・ノード（secondary node）

コールド・フェイルオーバーを行うクラスタ構成の場合、フェイルオーバー中にアプリケーションの移動先となるクラスタ・ノード。

関連項目： 用語集 -22 ページの「プライマリ・ノード」

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 Oracle9*i* リリース 2（9.2）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 Resources 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。

属性一意性（attribute uniqueness）

指定した 2 つの属性に同じ値が含まれていないようにする Oracle Internet Directory の機能。この機能によって、アプリケーションと企業のディレクトリを同期化し、属性を一意キーとして使用できる。

属性構成ファイル（attribute configuration file）

Oracle Directory Integration Platform 環境で、接続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 and Provisioning 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。

単一鍵ペア 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。

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 and Provisioning 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**。

ディレクトリ統合プロファイル（directory integration profile）

Oracle Directory Integration Platform 環境で、Oracle Directory Integration and Provisioning 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**」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 and Provisioning 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。

デフォルト認証管理レルム（default identity management realm）

ホスティングされた環境では、アプリケーション・サービス・プロバイダなどの 1 企業が、他の複数の企業に Oracle コンポーネントを使用可能にして、その情報を格納する。このようなホスティングされた環境では、ホスティングしている企業はデフォルト認証管理レルムと呼ばれ、ホスティングされている企業はそれぞれディレクトリ情報ツリー内のその企業独自の認証管理レルムに関連付けられる。

デフォルト・レルム位置（default realm location）

デフォルトの認証管理レルムのルートを識別するルート Oracle コンテキストでの属性。

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。

同時実行性（concurrency）

複数の要求を同時に処理できる機能。同時実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。

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ため、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。

ニックネーム属性（nickname attribute）

ディレクトリ全体のユーザーを一意に識別するために使用する属性。この属性のデフォルト値は uid。アプリケーションでは、この属性を使用して単純なユーザー名が完全な識別名に変換される。ユーザー・ニックネーム属性を複数値にはできない。つまり、ユーザーは同じ属性名で格納される複数のニックネームを所有できない。

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。

認証管理（identity management）

組織でネットワーク・エンティティのセキュリティ・ライフ・サイクル全体を管理するプロセス。通常、組織のアプリケーション・ユーザーの管理を指す。セキュリティ・ライフ・サイクルの手順には、アカウント作成、一時停止、権限変更およびアカウント削除が含まれる。管理されるネットワーク・エンティティには、デバイス、プロセス、アプリケーション、またはネットワーク環境で対話する必要があるその他のすべてのものが含まれる。認証管理プロセスで管理されるエンティティには、組織外のユーザー（顧客、取引先、Web サービスなど）も含まれる。

認証管理レルム（identity management realm）

すべてが同じ管理ポリシーによって管理されている識別情報の集合。企業では、イントラネットへのアクセス権限を所有しているすべての従業員は 1 つのレルムに属し、企業の公開アプリケーションにアクセスするすべての外部ユーザーは別のレルムに属する。認証管理レルムは、特別なオブジェクト・クラスが関連付けられた特定のエントリでディレクトリ内に表される。

認証管理レルム固有の Oracle コンテキスト（identity management realm-specific Oracle Context）

各認証管理レルムに含まれた Oracle コンテキスト。これには、次の情報が格納されている。

- 認証管理レルムのユーザー・ネーミング・ポリシー（ユーザーに名前を付け、配置する方法）
- 必須認証属性
- 認証管理レルム内のグループの位置
- 認証管理レルムに対する権限の割当て（レルムにユーザーを追加する権限の割当てなど）
- レルムに関するアプリケーション固有のデータ（認可など）

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）

Oracle Delegated Administration Services または Oracle Internet Directory Java API を使用して作成した新規ユーザー・エントリの相対識別名を構成するために使用する属性。この属性のデフォルト値は cn。

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 Server
- 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。

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。コールド・フェイルオーバーを行うクラスタ構成の場合、単一のクラスタ・ノード上で実行しているアプリケーションは透過的に他のクラスタ・ノードに移行される。移行中、クラスタ上のサービスにアクセスしているクライアントは一時停止され、フェイルオーバーの完了後、再接続する必要がある。

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。

プライマリ・ノード（primary node）

コールド・フェイルオーバーを行うクラスタ構成の場合、指定された時間にアプリケーションが実行されるクラスタ・ノード。

関連項目： 用語集 -16 ページの「セカンダリ・ノード」

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」を参照。

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。

マルチマスター・レプリケーション（multimaster replication）

peer-to-peer または n-way レプリケーションとも呼ばれる。同等に機能する複数のサイトがレプリケートされたデータのグループを管理できるようにするレプリケーションのタイプ。マルチマスター・レプリケーション環境では、各ノードはサプライヤ・ノードであると同時にコンシューマ・ノードであり、各ノードでディレクトリ全体がレプリケートされる。

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。

ユーザー検索ベース（user search base）

Oracle Internet Directory のデフォルトのディレクトリ情報ツリーで、すべてのユーザーが配置される認証管理レルムのノード。

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、Oracle9*i* Advanced Replication 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**」を参照。

ルート Oracle コンテキスト（Root Oracle Context）

Oracle Identity Management インフラストラクチャでは、ルート Oracle コンテキストは、インフラストラクチャのデフォルト認証管理レルムへのポインタを含む Oracle Internet Directory のエントリである。単純な名前を指定して認証管理レルムの位置を特定する方法の詳細も含まれる。

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。

レルム検索ベース（realm search base）

すべての認証管理レルムを含むディレクトリ情報ツリー内のエントリを識別するルート Oracle コンテキストでの属性。この属性は、単純なレルム名をディレクトリ内の対応するエントリにマッピングする際に使用される。

論理ホスト（logical host）

コールド・フェイルオーバーを行うクラスタ構成の場合、1 つ以上のディスク・グループおよびホスト名と IP アドレスのペア。クラスタの物理ホストにマップされる。この物理ホストは、論理ホストのホスト名と IP アドレスとして使用される。

索引

数字

A

B

C

D

J

K

L

M

O

P

R

S

T

U

W

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

そ

て

と

に

ね

は

ふ

へ

ほ

ま

れ

ろ