Oracle® Identity Management

アプリケーション開発者ガイド

10*g*（10.1.4.0.1）

部品番号 : B31464-01

2006 年 11 月

ORACLE®

Oracle Identity Management アプリケーション開発者ガイド , 10*g*（10.1.4.0.1）

部品番号 : B31464-01

原本名 : Oracle Identity Management Application Developer's Guide, 10*g* (10.1.4.0.1)

原本部品番号 : B15997-01

原本著者 : Ellen Desmond

原本協力者 : Vasuki Ashok , Tridip Bhattacharya, Ramakrishna Bollu, Saheli Dey, Ajay Keni, Ganesh Kirti, Ashish Kolli, Stephen Lee, Samit Roy, David Lin, Saurabh Shrivastava, Arun Theebaprakasam, Andy Tian

目次

はじめに xxi

対象読者 xxii
ドキュメントのアクセシビリティについて xxii
関連ドキュメント xxii
表記規則 xxiii
サポートおよびサービス xxiii

SDK の新機能 xxv

10*g*（10.1.4.0.1）の SDK の新機能 xxvi
リリース 10.1.2 の SDK の新機能 xxvi
リリース 9.0.4 の SDK の新機能 xxvii

第 I 部　Oracle Identity Management 向けプログラミング

1　Oracle Identity Management アプリケーションの開発

Oracle Identity Management との統合の利点 1-2
アプリケーションに統合可能な Oracle Identity Management サービス 1-2
既存のアプリケーションと Oracle Identity Management の統合 1-3
新しいアプリケーションと Oracle Identity Management の統合 1-3
Oracle Internet Directory プログラミング : 概要 1-4
Oracle Internet Directory SDK でサポートされるプログラミング言語 1-4
Oracle Internet Directory SDK コンポーネント 1-4
Oracle Internet Directory 環境でのアプリケーションの開発 1-5
ディレクトリ対応アプリケーションのアーキテクチャ 1-5
アプリケーションのライフ・サイクルでの Oracle Internet Directory の相互作用 1-6
アプリケーションと Oracle Internet Directory の統合に使用するサービスおよび API 1-7
既存のアプリケーションと Oracle Internet Directory の統合 1-8
新しいアプリケーションと Oracle Internet Directory の統合 1-9
Oracle Internet Directory のその他のコンポーネント 1-10

2　標準的な LDAP API を使用したアプリケーションの開発

サンプル・コード 2-2
LDAP の歴史 2-2
LDAP モデル 2-2
ネーミング・モデル 2-2
情報モデル 2-3

機能モデル 2-4
セキュリティ・モデル 2-4
認証 2-5
アクセス制御と認可 2-6
データ整合性 2-6
データ・プライバシ 2-6
パスワード・ポリシー 2-6
標準的な LDAP API の概要 2-7
API の使用モデル 2-7
C API スタート・ガイド 2-8
DBMS_LDAP パッケージ・スタート・ガイド 2-8
Java API スタート・ガイド 2-8
LDAP セッションの初期化 2-8
C API を使用したセッションの初期化 2-9
DBMS_LDAP を使用したセッションの初期化 2-9
JNDI を使用したセッションの初期化 2-10
LDAP セッションの認証 2-10
C API を使用した LDAP セッションの認証 2-11
DBMS_LDAP を使用した LDAP セッションの認証 2-11
ディレクトリの検索 2-12
検索操作のプログラム・フロー 2-12
検索有効範囲 2-14
フィルタ 2-14
C API を使用したディレクトリの検索 2-15
DBMS_LDAP を使用したディレクトリの検索 2-16
セッションの終了 2-17
C API を使用したセッションの終了 2-17
DBMS_LDAP を使用したセッションの終了 2-17

3 LDAP プロトコルに対する拡張機能

SASL 認証 3-2
Digest-MD5 を使用した SASL 認証 3-2
Digest-MD5 を使用した SASL 認証に含まれる手順 3-3
外部メカニズムを使用した SASL 認証 3-3
コントロールの使用 3-4
エンド・ユーザーの代理となるプロキシ 3-6
動的パスワード・ベリファイアの作成 3-7
動的パスワード・ベリファイアのリクエスト・コントロール 3-7
DynamicVerifierRequestControl の構文 3-8
ハッシング・アルゴリズムに必要なパラメータ 3-8
認証 API の構成 3-9
ldap_search の使用時に渡されるパラメータ 3-9
ldap_compare の使用時に渡されるパラメータ 3-9
動的パスワード・ベリファイアのレスポンス・コントロール 3-9
動的ベリファイア・フレームワークに対する権限の取得 3-9
階層検索の実行 3-9
CONNECT_BY コントロールの新機能 3-10
CONNECT_BY コントロールの値フィールド 3-10

ソート済の LDAP 検索結果 3-11
ページング済の LDAP 検索結果 3-11

4 標準的な 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

サンプル・コード 4-2
標準的な API に対する Oracle の拡張機能の使用 4-2
ディレクトリへのアプリケーション ID の作成 4-3
アプリケーション ID の作成 4-3
アプリケーション ID への権限の割当て 4-3
ユーザーの管理 4-3
グループの管理 4-4
レルムの管理 4-4
ディレクトリ・サーバーの検出 4-4
Oracle Internet Directory の検出インタフェースの利点 4-5
検出インタフェースの使用モデル 4-5
DNS からのサーバー名およびポート番号の判断 4-6
ネーミング・コンテキストの識別名のマッピング 4-6
ローカル・マシンのドメイン・コンポーネントによる検索 4-7
DNS でのデフォルト SRV レコードの検索 4-7
DNS サーバー検出の環境変数 4-7
DNS サーバー検出のプログラミング・インタフェース 4-8

5 JNDI に対する Java API 拡張機能の使用

サンプル・コード 5-2
Java 拡張機能のインストール 5-2
oracle.java.util パッケージを使用した LDAP オブジェクトのモデル化 5-2
PropertySetCollection、PropertySet および Property クラス 5-2
ユーザーの管理 5-3
ユーザーの認証 5-3
ユーザーの作成 5-4
ユーザー・オブジェクトの取得 5-4
レルムからのオブジェクトの取得 5-5
例 : OracleAS Single Sign-On のログイン名の検索 5-5
ディレクトリ・サーバーの検出 5-6
例 : ディレクトリ・サーバーの検出 5-7
Digest-MD5 を使用した SASL 認証の実行 5-8
例 : SASL Digest-MD5 の auth-int モードおよび auth-conf モードの使用 5-8

6 PL/SQL での API 拡張機能の使用

サンプル・コード 6-2
PL/SQL 拡張機能のインストール 6-2
ハンドルを使用したディレクトリ・データへのアクセス 6-2
ユーザーの管理 6-2
ユーザーの認証 6-3
PL/SQL LDAP API の依存性と制限事項 6-3

7 プロビジョニング統合アプリケーションの開発

8 Oracle Delegated Administration Services との統合

Oracle Delegated Administration Services とは 8-2
アプリケーションに対する Oracle Delegated Administration Services の利点 8-2
アプリケーションと Delegated Administration Services の統合 8-3
統合プロファイル 8-3
統合方法および考慮事項 8-3
URL アクセスに使用する Java API 8-5

9 シングル・サインオン対応のアプリケーションの開発

mod_osso とは 9-2
mod_osso を使用したアプリケーションの保護 : 2 つの方法 9-3
URL の静的な保護 9-3
ダイナミック・ディレクティブを使用した URL の保護 9-3
mod_osso を使用したアプリケーションの開発 9-4
静的に保護された PL/SQL アプリケーションの開発 9-4
静的に保護された Java アプリケーションの開発 9-6
ダイナミック・ディレクティブを使用する Java アプリケーションの開発 9-7
Java の例 1: 簡易認証 9-7
Java の例 2: シングル・サインオフ 9-8
非 GET 認証について 9-9
グローバルな非アクティブ・タイムアウトおよびダイナミック・ディレクティブ 9-9
セキュリティに関する問題 9-9
シングル・サインオフとアプリケーションのログアウト 9-10
アプリケーションのログイン : コード例 9-10
アプリケーションのログアウト : 推奨コード 9-12
mod_osso Cookie のセキュアな送信 9-12
強制認証 9-12

10 J2EE アプリケーションと Oracle Internet Directory の統合

標準の J2EE セキュリティ API 10-2
OC4J セキュリティ API 10-2
JAAS ポリシー管理 API 10-5
JAAS ポリシー管理 10-5
標準の JAAS API を使用したユーザー・ポリシーおよび権限の取得 10-6

第 II 部 サーバー・プラグイン

11 Oracle Internet Directory サーバーのプラグインの開発

サーバー・プラグインとは 11-2
サーバー・プラグインでサポートされている言語 11-2
サーバー・プラグインの前提条件 11-2
サーバー・プラグインの利点 11-2
プラグイン設計のガイドライン 11-3
サーバー・プラグイン・フレームワークとは 11-3

ディレクトリがサポートする LDAP 操作およびタイミング 11-3
pre 操作サーバー・プラグイン 11-4
post 操作サーバー・プラグイン 11-4
when 操作サーバー・プラグイン 11-4
when_replace 操作サーバー・プラグイン 11-5
プラグインの登録 11-5
プラグイン構成エントリ 11-5
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 11-8
Oracle Directory Manager を使用したプラグインの管理 11-9
Oracle Directory Manager を使用したプラグインの登録 11-9
Oracle Directory Manager を使用したプラグインの編集 11-9
Oracle Directory Manager を使用したプラグインの削除 11-9

12 PL/SQL サーバー・プラグイン

PL/SQL サーバー・プラグインの設計、作成および使用 12-2
PL/SQL プラグインの注意事項 12-2
PL/SQL プラグイン操作のタイプ 12-2
PL/SQL プラグインの名前付け 12-2
PL/SQL プラグインの作成 12-3
プラグイン・モジュール・インタフェースのパッケージ仕様 12-3
PL/SQL プラグインのコンパイル 12-5
依存性 12-5
プラグインの再コンパイル 12-5
PL/SQL プラグインの管理 12-5
プラグインの変更 12-5
プラグインのデバッグ 12-5
PL/SQL プラグインの有効化と無効化 12-6
PL/SQL プラグインにおける例外処理 12-6
エラー処理 12-6
Oracle Internet Directory とプラグインの間を処理するプログラム制御 12-6
PL/SQL プラグイン LDAP API 12-7
PL/SQL プラグインおよびレプリケーション 12-7
PL/SQL プラグインおよびデータベース・ツール 12-8
PL/SQL プラグインのセキュリティ 12-8
PL/SQL プラグインのデバッグ 12-8
PL/SQL プラグイン LDAP API の仕様 12-9
データベースの制限事項 12-9
PL/SQL プラグインの例 12-10
例 1: 問合せロギングの検索 12-10
例 2: 2 つのディレクトリ情報ツリーの同期化 12-12
PL/SQL プラグイン・フレームワークでのバイナリ・サポート 12-14
ldapmodify でのバイナリ操作 12-14
ldapadd でのバイナリ操作 12-17
ldapcompare でのバイナリ操作 12-19
データベース・オブジェクト・タイプの定義 12-22
PL/SQL プラグイン・プロシージャの仕様 12-23

13 Java サーバー・プラグイン

Java プラグインの利点 13-2
Java プラグインの設定 13-2
Java プラグイン API 13-3
サーバーとプラグインの通信 13-3
Java プラグインの構造 13-4
PluginDetail 13-4
Server 13-4
LdapBaseEntry 13-5
LdapOperation 13-6
PluginFlexfield 13-11
PluginResult 13-11
ServerPlugin インタフェース 13-12
Ldapbind の ServerPlugin メソッド 13-12
Ldapcompare の ServerPlugin メソッド 13-12
Ldapadd の ServerPlugin メソッド 13-12
Ldapmodify の ServerPlugin メソッド 13-12
Ldapmoddn の ServerPlugin メソッド 13-12
Ldapsearch の ServerPlugin メソッド 13-13
Ldapdelete の ServerPlugin メソッド 13-13
Java プラグイン・エラーおよび例外処理 13-13
ランタイム例外の例 13-13
ランタイム・エラーの例 13-14
PluginException の例 13-14
Java プラグインのデバッグおよびロギング 13-14
Java プラグインの例 13-15
例 1: パスワード検証プラグイン 13-15
パスワード検証プラグインの構成エントリ 13-15
パスワード検証プラグインのコード例 13-15
例 2: Active Directory の外部認証プラグイン 13-17
外部認証プラグインの構成エントリ 13-17
外部認証プラグインのコード 13-17

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

14 C API リファレンス

Oracle Internet Directory C API の概要 14-2
Oracle Internet Directory SDK C API SSL 拡張機能 14-2
SSL インタフェース・コール 14-2
Wallet サポート 14-3
C API のファンクション 14-3
ファンクションの概要 14-4
LDAP セッションの初期化 14-6
ldap_init および ldap_open 14-6
LDAP セッション・ハンドル・オプション 14-7
ldap_get_option および ldap_set_option 14-7
ディレクトリに対する認証 14-11
ldap_sasl_bind、ldap_sasl_bind_s、ldap_simple_bind および ldap_simple_bind_s 14-11

Oracle の拡張機能を使用した SASL 認証 14-13
ora_ldap_init_SASL 14-13
ora_ldap_create_cred_hdl、ora_ldap_set_cred_props、ora_ldap_get_cred_props および ora_ldap_free_cred_hdl 14-14
コントロールの使用 14-15
セッションのクローズ 14-17
ldap_unbind、ldap_unbind_ext および ldap_unbind_s 14-17
LDAP 操作の実行 14-17
ldap_search_ext、ldap_search_ext_s、ldap_search および ldap_search_s 14-17
エントリの読取り 14-20
エントリの子のリスト表示 14-20
ldap_compare_ext、ldap_compare_ext_s、ldap_compare および ldap_compare_s 14-21
ldap_modify_ext、ldap_modify_ext_s、ldap_modify および ldap_modify_s 14-22
ldap_rename および ldap_rename_s 14-24
ldap_add_ext、ldap_add_ext_s、ldap_add および ldap_add_s 14-26
ldap_delete_ext、ldap_delete_ext_s、ldap_delete および ldap_delete_s 14-27
ldap_extended_operation および ldap_extended_operation_s 14-29
操作の中止 14-30
ldap_abandon_ext および ldap_abandon 14-30
結果の取得と LDAP メッセージの確認 14-31
ldap_result、ldap_msgtype および ldap_msgid 14-31
エラーの処理と結果の解析 14-33
ldap_parse_result、ldap_parse_sasl_bind_result、ldap_parse_extended_result および ldap_err2string 14-33
結果リストの参照 14-35
ldap_first_message および ldap_next_message 14-35
検索結果の解析 14-36
ldap_first_entry、ldap_next_entry、ldap_first_reference、ldap_next_reference、ldap_count_entries および ldap_count_references 14-36
ldap_first_attribute および ldap_next_attribute 14-37
ldap_get_values、ldap_get_values_len、ldap_count_values、ldap_count_values_len、ldap_value_free および ldap_value_free_len 14-38
ldap_get_dn、ldap_explode_dn、ldap_explode_rdn および ldap_dn2ufn 14-39
ldap_get_entry_controls 14-40
ldap_parse_reference 14-40
C API の使用例 14-41
SSL モードでの C API の使用方法 14-41
非 SSL モードでの C API の使用方法 14-42
SASL ベースの Digest-MD5 認証での C API の使用方法 14-43
C API に必要なヘッダー・ファイルおよびライブラリ 14-45
C API の依存性と制限事項 14-46

15 DBMS_LDAP PL/SQL リファレンス

サブプログラムの概要 15-2
例外の概要 15-4
データ型の概要 15-5
サブプログラム 15-6
init ファンクション 15-6
simple_bind_s ファンクション 15-7

bind_s ファンクション 15-8
unbind_s ファンクション 15-9
compare_s ファンクション 15-10
search_s ファンクション 15-11
search_st ファンクション 15-12
first_entry ファンクション 15-14
next_entry ファンクション 15-15
count_entries ファンクション 15-16
first_attribute ファンクション 15-17
next_attribute ファンクション 15-18
get_dn ファンクション 15-19
get_values ファンクション 15-20
get_values_len ファンクション 15-21
delete_s ファンクション 15-22
modrdn2_s ファンクション 15-23
err2string ファンクション 15-24
create_mod_array ファンクション 15-24
populate_mod_array プロシージャ（文字列バージョン） 15-25
populate_mod_array プロシージャ（バイナリ・バージョン） 15-26
populate_mod_array プロシージャ（バイナリ・バージョン。BLOB データ型を使用） 15-27
get_values_blob ファンクション 15-28
count_values_blob ファンクション 15-29
value_free_blob ファンクション 15-30
modify_s ファンクション 15-30
add_s ファンクション 15-31
free_mod_array プロシージャ 15-32
count_values ファンクション 15-33
count_values_len ファンクション 15-33
rename_s ファンクション 15-34
explode_dn ファンクション 15-35
open_ssl ファンクション 15-36
msgfree ファンクション 15-37
ber_free ファンクション 15-38
nls_convert_to_utf8 ファンクション 15-38
nls_convert_to_utf8 ファンクション 15-39
nls_convert_from_utf8 ファンクション 15-40
nls_convert_from_utf8 ファンクション 15-41
nls_get_dbcharset_name ファンクション 15-42

16 Java API リファレンス

17 DBMS_LDAP_UTL PL/SQL リファレンス

サブプログラムの概要 17-2
サブプログラム 17-3
ユーザー関連サブプログラム 17-3
authenticate_user ファンクション 17-4
create_user_handle ファンクション 17-5

set_user_handle_properties ファンクション 17-6
get_user_properties ファンクション 17-7
set_user_properties ファンクション 17-8
get_user_extended_properties ファンクション 17-10
get_user_dn ファンクション 17-11
check_group_membership ファンクション 17-12
locate_subscriber_for_user ファンクション 17-13
get_group_membership ファンクション 17-14
グループ関連サブプログラム 17-15
create_group_handle ファンクション 17-16
set_group_handle_properties ファンクション 17-16
get_group_properties ファンクション 17-17
get_group_dn ファンクション 17-19
サブスクライバ関連サブプログラム 17-20
create_subscriber_handle ファンクション 17-20
get_subscriber_properties ファンクション 17-21
get_subscriber_dn ファンクション 17-22
get_subscriber_ext_properties ファンクション 17-23
プロパティ関連サブプログラム 17-24
その他のサブプログラム 17-25
normalize_dn_with_case ファンクション 17-25
get_property_names ファンクション 17-26
get_property_values ファンクション 17-27
get_property_values_len ファンクション 17-28
free_propertyset_collection プロシージャ 17-29
create_mod_propertyset ファンクション 17-29
populate_mod_propertyset ファンクション 17-30
free_mod_propertyset プロシージャ 17-31
free_handle プロシージャ 17-31
check_interface_version ファンクション 17-32
get_property_values_blob ファンクション 17-32
property_value_free_blob プロシージャ 17-33
ファンクション・リターン・コードの概要 17-34
データ型の概要 17-36

18 DAS_URL インタフェース・リファレンス

サービス・ユニットのディレクトリ・エントリ 18-2
サービス・ユニットおよび対応する URL パラメータ 18-3
DAS URL API のパラメータの説明 18-6
ユーザーまたはグループの検索および選択サービス・ユニット 18-7
ユーザーまたはグループの検索および選択サービス・ユニットの起動 18-7
ユーザーまたはグループの検索および選択サービス・ユニットからのデータ受信 18-8

19 Oracle Directory Integration Platform ユーザー・プロビジョニング Java API リファレンス

アプリケーション構成 19-2
アプリケーション登録とプロビジョニング構成 19-2
アプリケーションの登録 19-2
プロビジョニングの構成 19-4

アプリケーション構成クラス 19-14
ユーザー管理 19-14
ユーザーの作成 19-15
ユーザーの変更 19-15
ユーザーの削除 19-15
ユーザーの検索 19-16
デバッグ 19-16
サンプル・コード 19-16

20 Oracle Directory Integration Platform PL/SQL API リファレンス

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 20-2
拡張可能なイベント定義の構成 20-2
着信イベントおよび発信イベント 20-4
PL/SQL 双方向インタフェース（バージョン 3.0） 20-5
PL/SQL 双方向インタフェース（バージョン 2.0） 20-8
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） 20-9
事前定義されるイベント型 20-11
属性の型 20-11
属性の変更型 20-11
イベント処理の定数 20-11
コールバック 20-12
GetAppEvent() 20-12
PutAppEventStatus() 20-12
PutOIDEvent() 20-12

第 IV 部　付録

A　ユーザー・プロビジョニング用の Java プラグイン

プロビジョニング・プラグイン・タイプとその用途 A-2
プロビジョニング・プラグインの要件 A-2
データ・エントリ・プロビジョニング・プラグイン A-3
プレデータ・エントリ・プロビジョニング・プラグイン A-5
ポストデータ・エントリ・プロビジョニング・プラグイン A-5
データ・アクセス・プロビジョニング・プラグイン A-6
イベント配信プロビジョニング・プラグイン A-7
プロビジョニング・プラグインのリターン・ステータス A-10
プロビジョニング・プラグインの構成テンプレート A-10
プロビジョニング・プラグインのサンプル・コード A-12

B　DSML 構文

DSML の機能 B-2
DSML の利点 B-2
DSML 構文 B-2
トップレベルの構造 B-2
ディレクトリ・エントリ B-3
スキーマ・エントリ B-3
DSML で使用可能なツール B-4

C Netscape LDAP SDK API から Oracle LDAP SDK API への移行

機能 C-2
ファンクション C-2
マクロ C-2

用語集

索引

図一覧

1-1 ディレクトリ対応アプリケーション 1-5
1-2 API およびサービスを利用するアプリケーション 1-7
2-1 ディレクトリ情報ツリー 2-2
2-2 Anne Smith に関するエントリの属性 2-3
2-3 DBMS_LDAP の一般的な使用手順 2-7
2-4 検索関連操作の流れ 2-13
2-5 3 つの有効範囲オプション 2-14
4-1 API の拡張機能のプログラム・フロー 4-2
8-1 Delegated Administration Service の概要 8-2
13-1 サーバーと Java プラグインの通信 13-3
19-1 プロビジョニング構成データのディレクトリ情報ツリー 19-5

表一覧

1-1 アプリケーションのライフサイクルでの相互作用 1-6
1-2 Oracle Internet Directory との統合に使用するサービスおよび API 1-7
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 1-8
1-4 アプリケーション統合に関するポイント 1-9
2-1 LDAP 機能 2-4
2-2 SSL 認証モード 2-5
2-3 ldap_init() のパラメータ 2-9
2-4 ldap_simple_bind_s() の引数 2-11
2-5 search_s() または search_st() ファンクションのオプション 2-14
2-6 検索フィルタ 2-14
2-7 ブール演算子 2-15
2-8 ldap_search_s() の引数 2-16
2-9 DBMS_LDAP.search_s() および DBMS_LDAP.search_st() の引数 2-16
3-1 Oracle Internet Directory でサポートされているコントロール 3-4
3-2 DynamicVerifierRequestControl のパラメータ 3-8
3-3 ハッシング・アルゴリズムに必要なパラメータ 3-8
4-1 DNS 検出の環境変数 4-7
5-1 ディレクトリ・サーバー検出のメソッド 5-6
8-1 統合の考慮事項 8-3
8-2 Oracle Delegated Administration Services の URL パラメータ 8-4
9-1 パートナ・アプリケーションに渡されるユーザー属性 9-2
9-2 一般的にリクエストされるダイナミック・ディレクティブ 9-3
11-1 プラグイン構成オブジェクトおよび属性 11-5
12-1 プラグイン・モジュール・インタフェース 12-3
12-2 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 12-3
12-3 プラグインのリターン・コードの有効な値 12-6
12-4 プラグイン例外発生時のプログラム制御処理 12-6
12-5 LDAP 操作障害時のプログラム制御処理 12-7
13-1 各 LDAP 操作の DN 情報の内容 13-5
13-2 Operation Result Code の動作 13-6
13-3 LdapOperation のサブクラスおよびクラス固有の情報 13-6
13-4 各プラグインのタイミングに対する LdapEntry 情報の動作 13-7
13-5 各プラグインのタイミングに対する AttributeName の動作 13-7
13-6 各プラグインのタイミングに対する Attribute Value の動作 13-8
13-7 各プラグインのタイミングに対する Delete DN の動作 13-8
13-8 各プラグインのタイミングに対する New Parent DN 情報の動作 13-8
13-9 各プラグインのタイミングに対する New Relative DN 情報の動作 13-9
13-10 各プラグインのタイミングに対する Delete Old RDN 情報の動作 13-9
13-11 各プラグインのタイミングに対する LdapModification 情報の動作 13-9
13-12 各プラグインのタイミングに対する Required Attributes の動作 13-10
13-13 各プラグインのタイミングに対する Scope の動作 13-10
13-14 各プラグインのタイミングに対する SearchResultSet の動作 13-10
13-15 Java プラグインのロギングのデバッグ・レベル 13-14
14-1 SSL インタフェース・コールの引数 14-2
14-2 C API のファンクションおよびプロシージャ 14-4
14-3 LDAP セッションを初期化するためのパラメータ 14-6
14-4 LDAP セッション・ハンドル・オプションのパラメータ 14-7
14-5 定数 14-8
14-6 ディレクトリに対する認証のパラメータ 14-12
14-7 ora_ldap_init_SASL() に渡されるパラメータ 14-13
14-8 SASL 資格証明の管理パラメータ 14-15
14-9 ldapcontrol 構造体のフィールド 14-15
14-10 セッションをクローズするためのパラメータ 14-17
14-11 検索操作のためのパラメータ 14-19
14-12 比較操作のためのパラメータ 14-22
14-13 変更操作のためのパラメータ 14-23
14-14 LDAPMod 構造体のフィールド 14-24
14-15 名前の変更操作のためのパラメータ 14-25

14-16 追加操作のためのパラメータ 14-27
14-17 削除操作のためのパラメータ 14-28
14-18 拡張操作のためのパラメータ 14-29
14-19 操作を中止するためのパラメータ 14-30
14-20 結果の取得と LDAP メッセージの確認のためのパラメータ 14-31
14-21 エラーの処理と結果の解析を行うためのパラメータ 14-34
14-22 結果リストを参照するためのパラメータ 14-35
14-23 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 14-36
14-24 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 14-37
14-25 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 14-38
14-26 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 14-39
14-27 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 14-40
14-28 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 14-41
15-1 DBMS_LDAP API のサブプログラム 15-2
15-2 DBMS_LDAP 例外の概要 15-4
15-3 DBMS_LDAP データ型の概要 15-5
15-4 INIT ファンクションのパラメータ 15-6
15-5 INIT ファンクションの戻り値 15-6
15-6 INIT ファンクションの例外 15-6
15-7 SIMPLE_BIND_S ファンクションのパラメータ 15-7
15-8 SIMPLE_BIND_S ファンクションの戻り値 15-7
15-9 SIMPLE_BIND_S ファンクションの例外 15-7
15-10 BIND_S ファンクションのパラメータ 15-8
15-11 BIND_S ファンクションの戻り値 15-8
15-12 BIND_S ファンクションの例外 15-8
15-13 UNBIND_S ファンクションのパラメータ 15-9
15-14 UNBIND_S ファンクションの戻り値 15-9
15-15 UNBIND_S ファンクションの例外 15-9
15-16 COMPARE_S ファンクションのパラメータ 15-10
15-17 COMPARE_S ファンクションの戻り値 15-10
15-18 COMPARE_S ファンクションの例外 15-10
15-19 SEARCH_S ファンクションのパラメータ 15-11
15-20 SEARCH_S ファンクションの戻り値 15-11
15-21 SEARCH_S ファンクションの例外 15-12
15-22 SEARCH_ST ファンクションのパラメータ 15-12
15-23 SEARCH_ST ファンクションの戻り値 15-13
15-24 SEARCH_ST ファンクションの例外 15-13
15-25 FIRST_ENTRY ファンクションのパラメータ 15-14
15-26 FIRST_ENTRY の戻り値 15-14
15-27 FIRST_ENTRY の例外 15-14
15-28 NEXT_ENTRY ファンクションのパラメータ 15-15
15-29 NEXT_ENTRY ファンクションの戻り値 15-15
15-30 NEXT_ENTRY ファンクションの例外 15-15
15-31 COUNT_ENTRY ファンクションのパラメータ 15-16
15-32 COUNT_ENTRY ファンクションの戻り値 15-16
15-33 COUNT_ENTRY ファンクションの例外 15-16
15-34 FIRST_ATTRIBUTE ファンクションのパラメータ 15-17
15-35 FIRST_ATTRIBUTE ファンクションの戻り値 15-17
15-36 FIRST_ATTRIBUTE ファンクションの例外 15-17
15-37 NEXT_ATTRIBUTE ファンクションのパラメータ 15-18
15-38 NEXT_ATTRIBUTE ファンクションの戻り値 15-18
15-39 NEXT_ATTRIBUTE ファンクションの例外 15-18
15-40 GET_DN ファンクションのパラメータ 15-19
15-41 GET_DN ファンクションの戻り値 15-19
15-42 GET_DN ファンクションの例外 15-19
15-43 GET_VALUES ファンクションのパラメータ 15-20
15-44 GET_VALUES ファンクションの戻り値 15-20
15-45 GET_VALUES ファンクションの例外 15-20

15-46 GET_VALUES_LEN ファンクションのパラメータ 15-21
15-47 GET_VALUES_LEN ファンクションの戻り値 15-21
15-48 GET_VALUES_LEN ファンクションの例外 15-21
15-49 DELETE_S ファンクションのパラメータ 15-22
15-50 DELETE_S ファンクションの戻り値 15-22
15-51 DELETE_S ファンクションの例外 15-22
15-52 MODRDN2_S ファンクションのパラメータ 15-23
15-53 MODRDN2_S ファンクションの戻り値 15-23
15-54 MODRDN2_S ファンクションの例外 15-23
15-55 ERR2STRING ファンクションのパラメータ 15-24
15-56 ERR2STRING ファンクションの戻り値 15-24
15-57 CREATE_MOD_ARRAY ファンクションのパラメータ 15-24
15-58 CREATE_MOD_ARRAY ファンクションの戻り値 15-25
15-59 POPULATE_MOD_ARRAY（文字列バージョン）プロシージャのパラメータ 15-25
15-60 POPULATE_MOD_ARRAY（文字列バージョン）プロシージャの例外 15-26
15-61 POPULATE_MOD_ARRAY（バイナリ・バージョン）プロシージャのパラメータ 15-26
15-62 POPULATE_MOD_ARRAY（バイナリ・バージョン）プロシージャの例外 15-27
15-63 POPULATE_MOD_ARRAY（バイナリ）のパラメータ 15-27
15-64 POPULATE_MOD_ARRAY（バイナリ）の例外 15-28
15-65 GET_VALUES_BLOB のパラメータ 15-28
15-66 get_values_blob の戻り値 15-28
15-67 get_values_blob の例外 15-29
15-68 COUNT_VALUES_BLOB のパラメータ 15-29
15-69 COUNT_VALUES_BLOB の戻り値 15-29
15-70 VALUE_FREE_BLOB のパラメータ 15-30
15-71 MODIFY_S ファンクションのパラメータ 15-30
15-72 MODIFY_S ファンクションの戻り値 15-31
15-73 MODIFY_S ファンクションの例外 15-31
15-74 ADD_S ファンクションのパラメータ 15-31
15-75 ADD_S ファンクションの戻り値 15-32
15-76 ADD_S ファンクションの例外 15-32
15-77 FREE_MOD_ARRAY プロシージャのパラメータ 15-32
15-78 COUNT_VALUES ファンクションのパラメータ 15-33
15-79 COUNT_VALUES ファンクションの戻り値 15-33
15-80 COUNT_VALUES_LEN ファンクションのパラメータ 15-33
15-81 COUNT_VALUES_LEN ファンクションの戻り値 15-33
15-82 RENAME_S ファンクションのパラメータ 15-34
15-83 RENAME_S ファンクションの戻り値 15-34
15-84 RENAME_S ファンクションの例外 15-35
15-85 EXPLODE_DN ファンクションのパラメータ 15-35
15-86 EXPLODE_DN ファンクションの戻り値 15-35
15-87 EXPLODE_DN ファンクションの例外 15-35
15-88 OPEN_SSL ファンクションのパラメータ 15-36
15-89 OPEN_SSL ファンクションの戻り値 15-36
15-90 OPEN_SSL ファンクションの例外 15-36
15-91 MSGFREE ファンクションのパラメータ 15-37
15-92 MSGFREE ファンクションの戻り値 15-37
15-93 BER_FREE ファンクションのパラメータ 15-38
15-94 nls_convert_to_utf8 のパラメータ 15-38
15-95 nls_convert_to_utf8 の戻り値 15-38
15-96 nls_convert_to_utf8 のパラメータ 15-39
15-97 nls_convert_to_utf8 の戻り値 15-39
15-98 nls_convert_from_utf8 のパラメータ 15-40
15-99 nls_convert_from_utf8 の戻り値 15-40
15-100 nls_convert_from_utf8 のパラメータ 15-41
15-101 nls_convert_from_utf8 の戻り値 15-41
15-102 nls_get_dbcharset_name の戻り値 15-42
17-1 DBMS_LDAP_UTL のユーザー関連サブプログラム 17-2
17-2 DBMS_LDAP_UTL のグループ関連サブプログラム 17-2
17-3 DBMS_LDAP_UTL のサブスクライバ関連サブプログラム 17-2

17-4 DBMS_LDAP_UTL のその他のサブプログラム 17-2
17-5 authenticate_user ファンクションのパラメータ 17-4
17-6 authenticate_user ファンクションの戻り値 17-5
17-7 CREATE_USER_HANDLE ファンクションのパラメータ 17-6
17-8 CREATE_USER_HANDLE ファンクションの戻り値 17-6
17-9 SET_USER_HANDLE_PROPERTIES ファンクションのパラメータ 17-6
17-10 SET_USER_HANDLE_PROPERTIES ファンクションの戻り値 17-7
17-11 GET_USER_PROPERTIES ファンクションのパラメータ 17-7
17-12 GET_USER_PROPERTIES ファンクションの戻り値 17-8
17-13 SET_USER_PROPERTIES ファンクションのパラメータ 17-9
17-14 SET_USER_PROPERTIES ファンクションの戻り値 17-9
17-15 GET_USER_EXTENDED_PROPERTIES ファンクションのパラメータ 17-10
17-16 GET_USER_EXTENDED_PROPERTIES ファンクションの戻り値 17-10
17-17 GET_USER_DN ファンクションのパラメータ 17-11
17-18 GET_USER_DN ファンクションの戻り値 17-11
17-19 CHECK_GROUP_MEMBERSHIP ファンクションのパラメータ 17-12
17-20 CHECK_GROUP_MEMBERSHIP ファンクションの戻り値 17-12
17-21 LOCATE_SUBSCRIBER_FOR_USER ファンクションのパラメータ 17-13
17-22 LOCATE SUBSCRIBER FOR USER ファンクションの戻り値 17-13
17-23 GET_GROUP_MEMBERSHIP ファンクションのパラメータ 17-14
17-24 GET_GROUP_MEMBERSHIP ファンクションの戻り値 17-14
17-25 CREATE_GROUP_HANDLE ファンクションのパラメータ 17-16
17-26 CREATE_GROUP_HANDLE ファンクションの戻り値 17-16
17-27 SET_GROUP_HANDLE_PROPERTIES ファンクションのパラメータ 17-17
17-28 SET_GROUP_HANDLE_PROPERTIES ファンクションの戻り値 17-17
17-29 GET_GROUP_PROPERTIES ファンクションのパラメータ 17-18
17-30 GET_GROUP_PROPERTIES ファンクションの戻り値 17-18
17-31 GET_GROUP_DN ファンクションのパラメータ 17-19
17-32 GET_GROUP_DN ファンクションの戻り値 17-19
17-33 CREATE_SUBSCRIBER_HANDLE ファンクションのパラメータ 17-20
17-34 CREATE_SUBSCRIBER_HANDLE ファンクションの戻り値 17-21
17-35 GET_SUBSCRIBER_PROPERTIES ファンクションのパラメータ 17-21
17-36 GET_SUBSCRIBER_PROPERTIES ファンクションの戻り値 17-22
17-37 GET_SUBSCRIBER_DN ファンクションのパラメータ 17-22
17-38 GET_SUBSCRIBER_DN ファンクションの戻り値 17-23
17-39 GET_SUBSCRIBER_EXT_PROPERTIES ファンクションのパラメータ 17-23
17-40 GET_USER_EXTENDED_PROPERTIES ファンクションの戻り値 17-24
17-41 NORMALIZE_DN_WITH_CASE ファンクションのパラメータ 17-25
17-42 NORMALIZE_DN_WITH_CASE ファンクションの戻り値 17-25
17-43 GET_PROPERTY_NAMES ファンクションのパラメータ 17-26
17-44 GET_PROPERTY_NAMES ファンクションの戻り値 17-26
17-45 GET_PROPERTY_VALUES ファンクションのパラメータ 17-27
17-46 GET_PROPERTY_VALUES ファンクションの戻り値 17-27
17-47 GET_PROPERTY_VALUES_LEN ファンクションのパラメータ 17-28
17-48 GET_PROPERTY_VALUES_LEN ファンクションの戻り値 17-28
17-49 FREE_PROPERTYSET_COLLECTION プロシージャのパラメータ 17-29
17-50 CREATE_MOD_PROPERTYSET ファンクションのパラメータ 17-29
17-51 CREATE_MOD_PROPERTYSET ファンクションの戻り値 17-30
17-52 POPULATE_MOD_PROPERTYSET ファンクションのパラメータ 17-30
17-53 POPULATE_MOD_PROPERTYSET ファンクションの戻り値 17-30
17-54 FREE_MOD_PROPERTYSET プロシージャのパラメータ 17-31
17-55 FREE_HANDLE プロシージャのパラメータ 17-31
17-56 CHECK_INTERFACE_VERSION ファンクションのパラメータ 17-32
17-57 CHECK_VERSION_INTERFACE ファンクションの戻り値 17-32
17-58 GET_PROPERTY_VALUES_BLOB ファンクションのパラメータ 17-33
17-59 GET_PROPERTY_VALUES_BLOB の戻り値 17-33
17-60 PROPERTY_VALUE_FREE_BLOB ファンクションのパラメータ 17-33
17-61 ファンクション・リターン・コード 17-34
17-62 DBMS_LDAP_UTL のデータ型 17-36
18-1 サービス・ユニットおよび対応するエントリ 18-2

18-2 サービス・ユニットおよび対応する URL パラメータ 18-3
18-3 DAS URL のパラメータの説明 18-6
18-4 ユーザーの検索と選択 18-8
18-5 グループの検索と選択 18-8
19-1 有益な権限グループの一部 19-3
19-2 インタフェースとその構成 19-8
19-3 PL/SQL インタフェースでサポートされる情報の書式 19-8
19-4 属性の構成エントリに属性として格納されるプロパティ 19-10
19-5 イベント伝播のパラメータ 19-11
20-1 事前定義済のイベント定義 20-2
20-2 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 20-4

はじめに

『Oracle Identity Management アプリケーション開発者ガイド』では、アプリケーションを変更して Oracle Identity Management インフラストラクチャと連携させる方法について説明します。このマニュアルでは、Oracle Application Server Single Sign-On、Oracle Internet Directory、Oracle Delegated Administration Services および Directory Integration Platform をまとめて Oracle Identity Management インフラストラクチャと呼んでいます。

「はじめに」の項目は次のとおりです。

- 対象読者
- ドキュメントのアクセシビリティについて
- 関連ドキュメント
- 表記規則
- サポートおよびサービス

対象読者

このマニュアルは次のような読者を対象としています。

- アプリケーションを Oracle Identity Management インフラストラクチャと統合する開発者。統合プロセスでは、Oracle Internet Directory サーバーに情報を格納し、格納した情報を更新する必要があります。アプリケーションを変更して、Oracle HTTP Server の認証モジュール mod_osso と連携させることも必要です。
- LDAP API と LDAP API に対する Oracle の拡張機能について学習するすべてのユーザー。

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様にもオラクル社の製品、サービスおよびサポート・ドキュメントを簡単にご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト `http://www.oracle.com/accessibility/` を参照してください。

ドキュメント内のサンプル・コードのアクセシビリティについて

スクリーン・リーダーは、ドキュメント内のサンプル・コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかし JAWS は括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

Oracle サポート・サービスへの TTY アクセス

アメリカ国内では、Oracle サポート・サービスへ 24 時間年中無休でテキスト電話（TTY）アクセスが提供されています。TTY サポートについては、(800)446-2398 にお電話ください。

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。

- 『Oracle Identity Management インフラストラクチャ管理者ガイド』
- 『Oracle Internet Directory 管理者ガイド』
- 『Oracle Identity Management 統合ガイド』
- 『Oracle Identity Management 委任管理ガイド』
- 『Oracle Application Server Single Sign-On 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』
- 『Oracle Database アプリケーション開発者ガイド - 基礎編』
- 『Oracle セキュリティ開発ツール・リファレンス』

その他の情報は、次を参照してください。

- Chadwick, David 著『Understanding X.500 - The Directory』Thomson Computer Press, 1996
- Howes, Tim および Mark Smith 著『LDAP: Programming Directory-enabled Applications with Lightweight Directory Access Protocol』Macmillan Technical Publishing, 1997

- Howes, Tim、Mark Smith および Gordon Good 著『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

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、また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。
イタリック	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

サポートおよびサービス

次の各項に、各サービスに接続するための URL を記載します。

Oracle サポート・サービス

オラクル製品サポートの購入方法、および Oracle サポート・サービスへの連絡方法の詳細は、次の URL を参照してください。

`http://www.oracle.co.jp/support/`

製品マニュアル

製品のマニュアルは、次の URL にあります。

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

研修およびトレーニング

研修に関する情報とスケジュールは、次の URL で入手できます。

`http://www.oracle.co.jp/education/`

その他の情報

オラクル製品やサービスに関するその他の情報については、次の URL から参照してください。

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

注意： ドキュメント内に記載されている URL や参照ドキュメントには、Oracle Corporation が提供する英語の情報も含まれています。日本語版の情報については、前述の URL を参照してください。

SDK の新機能

ここでは、Oracle Internet Directory Software Developer's Kit の今回および前回のリリースにおける新機能について説明します。各機能の詳細は、記載のリンクを使用して参照してください。

10*g*（10.1.4.0.1）の SDK の新機能

10*g*（10.1.4.0.1）の SDK には、次の機能が追加されています。

- Java プラグイン・サポート

  サーバー・プラグインは PL/SQL だけでなく、Java で記述することも可能です。詳細は、第 11 章「Oracle Internet Directory サーバーのプラグインの開発」および第 13 章「Java サーバー・プラグイン」を参照してください。

- LDAP 検索結果のページングおよびソート

  LDAP 検索からページングおよびソート済の結果を取得できるようになりました。詳細は、第 3 章「LDAP プロトコルに対する拡張機能」の「ソート済の LDAP 検索結果」および「ページング済の LDAP 検索結果」を参照してください。

- 階層検索への追加機能

  階層全体をどちらの方向にも検索できるようになりました。また、検索する階層のレベル数も指定できます。詳細は、第 3 章「LDAP プロトコルに対する拡張機能」の「階層検索の実行」を参照してください。

- SASL Digest-MD5 認証の 3 つのすべてのモードに対するサポート

  Oracle Internet Directory では、jdk1.4 API の Java Naming and Directory Interface（JNDI）、または OpenLDAP Java API で 3 つのモードがすべてサポートされています。詳細は、第 3 章「LDAP プロトコルに対する拡張機能」の「SASL 認証」および第 5 章「JNDI に対する Java API 拡張機能の使用」の「例 : SASL Digest-MD5 の auth-int モードおよび auth-conf モードの使用」を参照してください。

リリース 10.1.2 の SDK の新機能

リリース 10.1.2 の SDK には、次の機能が追加されています。

- 一元化されたユーザー・プロビジョニング

  この機能により、アプリケーション・ユーザーを Oracle Identity Management インフラストラクチャにプロビジョニングできます。詳細は、第 19 章「Oracle Directory Integration Platform ユーザー・プロビジョニング Java API リファレンス」を参照してください。

- 動的パスワード・ベリファイア

  この機能は、パスワード・ベリファイアのパラメータを実行時にのみ指定するアプリケーションの必要性に対応したものです。詳細は、第 3 章の「動的パスワード・ベリファイアの作成」を参照してください。

- `ldapmodify`、`ldapadd` および `ldapcompare` のプラグインに対するバイナリ・サポート

  ディレクトリ・プラグインがディレクトリ・データベースのバイナリ属性にアクセスできるようになりました。詳細は、第 12 章の「PL/SQL プラグイン・フレームワークでのバイナリ・サポート」を参照してください。

- Oracle Directory Integration Platform Server のプラグイン・サポート

  これらの Java フックにより、企業では、独自のビジネス・ルールを組み入れ、各要件に合せてフットプリント作成をカスタマイズできます。詳細は、付録 A を参照してください。

リリース 9.0.4 の SDK の新機能

次の機能は、リリース 9.0.4 の SDK で導入されました。

- Oracle Delegated Administration Services の URL API

  この API を使用すると、委任管理者がディレクトリ操作を行うために使用する管理セルフ・サービス・コンソールを構築できます。詳細は、第 8 章を参照してください。

- PL/SQL API の拡張機能

  - LDAP v3 規格の新規ファンクション。これまでは C API でのみ使用可能でしたが、PL/SQL でも使用できるようになりました。
  - 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 Platform のプロビジョニング・プロファイルを作成および管理するファンクション。

  詳細は、第 7 章を参照してください。

- 外部認証プラグインのサポート

  この機能により、管理者は Microsoft Active Directory を使用して Oracle コンポーネントのセキュリティ資格証明を格納および管理できます。詳細は、第 11 章を参照してください。

- DNS を使用したサーバー検出

  この機能により、ディレクトリ・クライアントはディレクトリ・サーバーのホスト名およびポート番号を検出できます。その結果、大規模な配置におけるディレクトリ・クライアントの維持コストが削減されます。詳細は、第 4 章の「ディレクトリ・サーバーの検出」を参照してください。

- ディレクトリ SDK とディレクトリ・ツールに対する XML サポート

  この機能により、LDAP ツールで XML および LDIF 表記法を処理できます。ディレクトリ API では DSML 1.0 形式のデータを操作できます。

- クライアント側の参照キャッシュ

  この機能により、クライアントは参照情報をキャッシュできるため、参照プロセスの処理時間が短縮されます。詳細は、第 8 章の「LDAP セッション・ハンドル・オプション」を参照してください。

第 I 部

Oracle Identity Management 向けプログラミング

第 I 部では、アプリケーションを変更して Oracle Identity Management の各種コンポーネントと連携させる方法について説明します。最初に、Oracle Internet Directory SDK と LDAP プログラミングの概念を紹介します。次に、3 種類の LDAP API とその拡張機能を使用してアプリケーションを Oracle Internet Directory 対応にする方法を説明します。

第 I 部は、次の章で構成されています。

- 第 1 章「Oracle Identity Management アプリケーションの開発」
- 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」
- 第 3 章「LDAP プロトコルに対する拡張機能」
- 第 4 章「標準的な 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」
- 第 5 章「JNDI に対する Java API 拡張機能の使用」
- 第 6 章「PL/SQL での API 拡張機能の使用」
- 第 7 章「プロビジョニング統合アプリケーションの開発」
- 第 8 章「Oracle Delegated Administration Services との統合」
- 第 9 章「シングル・サインオン対応のアプリケーションの開発」
- 第 10 章「J2EE アプリケーションと Oracle Internet Directory の統合」

1

Oracle Identity Management アプリケーションの開発

Oracle Identity Management は、すべての Oracle アプリケーションを対象とした共有インフラストラクチャを提供します。また、サード・パーティのエンタープライズ・アプリケーションの開発を容易にするサービスやインタフェースも提供します。このインタフェースは、アプリケーションに ID 管理を組み込む必要のあるアプリケーション開発者にとって有益です。

この章では、このインタフェースについて説明し、Oracle Identity Management 環境におけるアプリケーション開発の推奨ベスト・プラクティスを示します。

Oracle Identity Management に統合できるアプリケーションは 2 種類あります。

- 企業内で使用されている既存のアプリケーション。こうしたアプリケーションには企業がすでに投資している場合もあり、Oracle Identity Management インフラストラクチャと統合することで利益が得られます。
- Oracle テクノロジ・スタックに基づき、企業の IT 部門または ISV で新たに開発されるアプリケーション。

この章の項目は次のとおりです。

- Oracle Identity Management との統合の利点
- アプリケーションに統合可能な Oracle Identity Management サービス
- 既存のアプリケーションと Oracle Identity Management の統合
- 新しいアプリケーションと Oracle Identity Management の統合
- Oracle Internet Directory プログラミング : 概要

Oracle Identity Management との統合の利点

エンタープライズ・アプリケーションと Oracle Identity Management インフラストラクチャの統合には、次のような利点があります。

- **アプリケーション配置の迅速化とコストの削減を促進 :** 既存の Oracle Identity Management インフラストラクチャを使用している企業（主に Oracle の顧客）は、Oracle Delegated Administration Services のセルフ・サービス・コンソールを使用して新しいアプリケーションを配置できます。アプリケーション管理をユーザーに委任することで、アプリケーションの配置コストが削減されます。
- **Oracle アプリケーションとのシームレスな統合 :** Oracle アプリケーションはすべて Oracle Identity Management インフラストラクチャをベースにしているため、新しいエンタープライズ・アプリケーションは Oracle Identity Management のすべての機能を使用できます。
- **サード・パーティの ID 管理ソリューションとのシームレスな統合 :** Oracle Identity Management インフラストラクチャにはサード・パーティの ID 管理ソリューションとの統合機能が組み込まれているため、アプリケーション開発者は ID 管理機能を利用できます。

アプリケーションに統合可能な Oracle Identity Management サービス

カスタム・アプリケーションでは、文書化およびサポートされている一連のサービスや API を通じて Oracle Identity Management を使用できます。次に例を示します。

- Oracle Internet Directory: C、Java および PL/SQL の LDAP API を提供します。他の LDAP SDK と互換性があります。
- Oracle Delegated Administration Services: 主要なセルフ・サービス・コンソールを提供します。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。また、ディレクトリ・データを操作するカスタム管理インタフェース構築用のサービスもいくつか提供します。
- Oracle Directory Integration Services: Oracle Internet Directory をサード・パーティ・ディレクトリやその他のユーザー・リポジトリと同期化するためのカスタム・ソリューションの開発および配置を容易にします。
- Oracle Provisioning Integration Services: サード・パーティ・アプリケーションをプロビジョニングするメカニズムを提供します。また、Oracle 環境をその他のプロビジョニング・システムと統合する手段にもなります。
- OracleAS Single Sign-On: その他の Oracle Web アプリケーションとシングル・サインオン・セッションを共有するパートナ・アプリケーションの開発および配置を行う API を提供します。
- JAZN: Java Authentication and Authorization Service（JAAS）Support 標準の Oracle 実装。Oracle J2EE 環境を使用して Web 向けに開発されたアプリケーションが、ID 管理インフラストラクチャを使用して認証と認可を行えるようにします。

既存のアプリケーションと Oracle Identity Management 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があります。Oracle Identity Management インフラストラクチャの次のサービスを使用すると、既存のアプリケーションを変更できます。

- **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。このエージェントは、Oracle Provisioning Integration Service のインタフェースを使用して開発する必要があります。

  関連資料 : ユーザーの自動プロビジョニングの開発の詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

- **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、OracleAS Single Sign-On サービスを使用してすべての着信ユーザー・リクエストを認証できます。
- **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、OracleAS 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 Delegated Administration Services のセルフ・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。

新しいアプリケーションと Oracle Identity Management 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は、アプリケーション開発者は Oracle Identity Management インフラストラクチャで提供されるサービスをより広範囲に使用できます。アプリケーション開発者は、統合に関する次のポイントを考慮する必要があります。

- **ユーザー認証サービス :** 次の選択肢があります。
  - J2EE ベースのアプリケーションの場合、Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider インタフェースで提供されるサービスを使用できます。
  - Oracle Containers for J2EE（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 ヘッダー内の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、OracleAS Single Sign-On API を使用するパートナ・アプリケーションとして OracleAS Single Sign-On を使用できます。
  - Web ベース以外の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。
- **プロファイルの一元管理 :** 次の選択肢があります。
  - アプリケーション固有のプロファイルとユーザー・プリファレンスを Oracle Internet Directory の属性としてモデル化できます。
  - 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、OracleAS 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 Delegated Administration Services のセルフ・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。
  - Oracle Internet Directory LDAP API（C、PL/SQL および Java で使用可能）を使用して、実行時にユーザー・プロファイルを取得することもできます。

- **ユーザーの自動プロビジョニング :** 次の選択肢を検討する必要があります。
  - 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、OracleAS 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。
  - アプリケーションを Oracle Internet Directory Provisioning Integration Service と統合することもできます。これにより、Oracle Identity Management インフラストラクチャでの ID の追加、既存の ID のプロパティ変更、既存の ID の削除などの管理アクションに応じて、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解除が自動的に行えます。

    関連資料 :『Oracle Identity Management 統合ガイド』

Oracle Internet Directory プログラミング : 概要

この項では、Oracle Internet Directory Software Developer's Kit を紹介し、このキットを使用してアプリケーションをディレクトリと統合する方法について概説します。ディレクトリ製品スイートのその他のコンポーネントも示します。

この項では、次の項目について説明します。

- Oracle Internet Directory SDK でサポートされるプログラミング言語
- Oracle Internet Directory SDK コンポーネント
- Oracle Internet Directory 環境でのアプリケーションの開発
- Oracle Internet Directory のその他のコンポーネント

Oracle Internet Directory SDK でサポートされるプログラミング言語

SDK 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ています。Java 開発者がディレクトリとの統合を行う場合は、Sun 社の JNDI プロバイダを使用する必要があります。

Oracle Internet Directory SDK コンポーネント

Oracle Internet Directory Software Developer's Kit 10*g*（10.1.4.0.1）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 API
- 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 Identity Management アプリケーション開発者ガイド』（このドキュメント）
- 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 Oracle Internet Directory の相互作用
- アプリケーションと Oracle Internet Directory の統合に使用するサービスおよび 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リクエスト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、このようなアプリケーションによる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

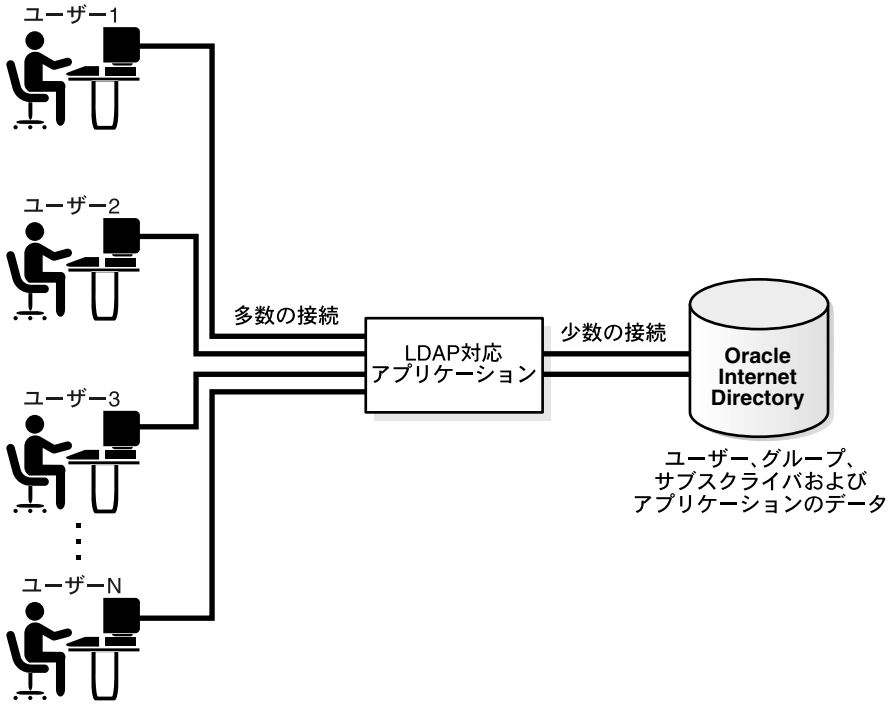


図 1-1 に示すように、ユーザー・リクエストが LDAP 対応の操作を必要とする場合、アプリケーションはあらかじめ確立されたディレクトリ接続の一部を使用してリクエストを処理します。

アプリケーションのライフ・サイクルでの Oracle Internet Directory の相互作用

1-6 ページの表 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操作を順に示します。

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ポイント	ロジック
アプリケーションのインストール	1. ディレクトリにアプリケーションの ID を作成します。アプリケーションはこの ID を使用して、ほとんどの LDAP 操作を実行します。 2. LDAP 認可を指定するために、アプリケーションの ID を正しい LDAP グループに含めます。この認可により、アプリケーションはユーザー資格証明を受け入れて、ディレクトリに対して認証することができます。また、ディレクトリは、そのユーザーにかわって LDAP 操作を実行する必要があるときに、アプリケーション認可を使用してユーザーの代理を務めることができます。
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、自身をディレクトリ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・リクエストを処理できます。
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・リクエストについて、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。 ■ 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。 ■ この章で説明する標準 API または API 拡張機能を使用して、LDAP 操作を実行します。 ■ LDAP 操作の完了後、有効なユーザーがアプリケーションの ID になってい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 ID とそのアプリケーション ID に指定されている LDAP 認可を削除します。

アプリケーションと Oracle Internet Directory の統合に使用するサービスおよび API

アプリケーション開発者は、1-7 ページの表 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 との統合に使用するサービスおよび 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 は、ID 管理関連の様々な概念をモデル化するプログラム・インタフェースを提供します。	第 4 章「標準的な 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、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ています。管理インタフェースは、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変更できます。	■ 第 8 章「Oracle Delegated Administration Services との統合」 ■ 『Oracle Identity Management 委任管理ガイド』の Delegated Administration Services フレームワークに関する章
Oracle Directory Provisioning Integration Service	Oracle Provisioning Integration System を使用して、サード・パーティ・アプリケーションのプロビジョニング、およびその他のプロビジョニング・システムの統合を行うことができます。	■ 第 7 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 統合ガイド』
Oracle Internet Directory プラグイン	プラグインは、特定の配置におけるディレクトリ動作のカスタマイズに使用できます。	■ 第 11 章「Oracle Internet Directory サーバーのプラグインの開発」 ■ 『Oracle Internet Directory 管理者ガイド』のプラグインに関する章 ■ 付録 A「ユーザー・プロビジョニング用の Java プラグイン」

図 1-2 に、1-7 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

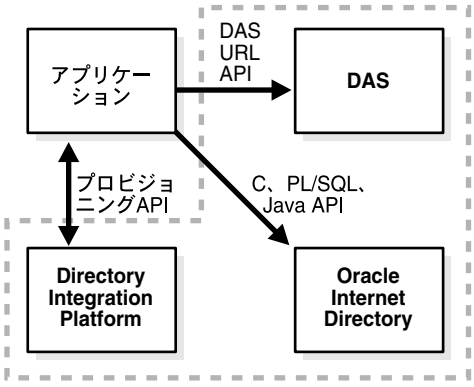


図 1-2 に示すとおり、アプリケーションは次のように Oracle Internet Directory と統合されています。

- PL/SQL、C または Java の API を使用して、ディレクトリに対して LDAP 操作を直接実行します。
- ユーザーを Oracle Delegated Administration Services のセルフ・サービス機能に送る場合もあります。
- Oracle Internet Directory 内のユーザーまたはグループのエントリに対する変更について通知を受けます。この通知は、Oracle Directory Provisioning Integration Service が実行します。

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

Oracle の ID 管理インフラストラクチャとの統合の対象となる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があります。表 1-3 に示すサービスを使用すれば、そのようなアプリケーションも統合できます。

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	Oracle の ID 管理インフラストラクチャでプロビジョニング・イベントが発生したときに、ユーザーのプロビジョニングを自動的に行うエージェントを開発できます。このエージェントを開発するには、Oracle Directory Provisioning Integration Service のインタフェースを使用します。	第 7 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	ユーザー・インタフェースが HTTP に基づいている場合、ユーザー・インタフェースを Oracle HTTP Server と統合できます。これにより、mod_osso と OracleAS Single Sign-On を使用してアプリケーションの URL を保護できるようになります。	『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	ユーザー・インタフェースが HTTP に基づいていて、OracleAS Single Sign-On と統合されている場合、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。コンソールは、アプリケーションの要件に応じてカスタマイズできます。	■ 第 8 章「Oracle Delegated Administration Services との統合」 ■ 『Oracle Identity Management 委任管理ガイド』の Delegated Administration Services フレームワークに関する章

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は、多くの統合オプションから自由に選択できます。1-9 ページの表 1-4 に、統合オプションを示します。

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 インタフェースを使用してユーザーを認証できます。OC4J ベースのアプリケーションの場合は、同じ目的で mod_osso を使用します。2 番目のオプションでは、アプリケーションは HTTP ヘッダーからユーザー情報を取得できます。 Web ベースのスタンドアロン・アプリケーションの場合でも、OracleAS Single Sign-On と統合できます。シングル・サインオン API を使用するパートナ・アプリケーションになることで、Oracle Application Server Single Sign-On も利用できます。 Web ベース以外のユーザー・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 を使用してユーザーを統合できます。	■ 『Oracle Containers for J2EE 開発者ガイド』 ■ 『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」。ここでは、様々な LDAP API について説明しています。
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 インタフェースを使用して、アプリケーション・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定義し、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。この目的で Oracle Internet Directory LDAP API を使用できます。	■ 『Oracle Containers for J2EE 開発者ガイド』 ■ 第 II 部「Oracle Internet Directory プログラミング・リファレンス」。ここでは、様々な LDAP API について説明しています。
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とユーザー・プリファレンスを Oracle Internet Directory の属性として定義できます。 ユーザー・インタフェースが HTTP に基づいていて、OracleAS Single Sign-On と統合されている場合、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。コンソールは、アプリケーションの要件に応じてカスタマイズできます。 また、Oracle Internet Directory LDAP API を使用して、実行時にユーザー・プロファイルを取得することもできます。	■ 『Oracle Internet Directory 管理者ガイド』の配置についての考慮事項に関する章 ■ 第 8 章「Oracle Delegated Administration Services との統合」 ■ 『Oracle Identity Management 委任管理ガイド』 ■ 第 II 部。ここでは様々な LDAP API について説明しています。
ユーザーの自動プロビジョニング	ユーザー・インタフェースが HTTP に基づいていて、OracleAS Single Sign-On と統合されている場合、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。 Oracle Directory Provisioning Integration Service を使用して、アプリケーションを Oracle の ID 管理インフラストラクチャと統合できます。統合後、管理者が ID を追加、変更または削除したときに、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解除を自動的に行うことができます。	第 7 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 のその他のコンポーネント

SDK はディレクトリ・スイートのコンポーネントの 1 つにすぎません。他にも次のようなコンポーネントがあります。

- Oracle Internet Directory サーバー、LDAP バージョン 3
- 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』

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 を使用して行うことのできる操作について概説します。また、アプリケーションを API と統合する方法も説明します。これらの説明を始める前に、**Lightweight Directory Access Protocol** について振り返ります。

この章では、次の項目について説明します。

- サンプル・コード
- LDAP の歴史
- LDAP モデル
- 標準的な LDAP API の概要
- LDAP セッションの初期化
- LDAP セッションの認証
- ディレクトリの検索
- セッションの終了

サンプル・コード

サンプル・コードは次の URL で入手できます。

`http://www.oracle.com/technology/sample_code/`

「Sample Applications-Fusion Middleware」の下の「Oracle Identity Management」リンクを探してください。

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。LDAP は X.500 Directory Access Protocol を次のように簡素化しています。

- 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。
- ネットワーク上のデータ・トランスポートに使用される X.500 エンコーディング規則の簡素化バージョンを使用します。

LDAP モデル

LDAP では、4 つの基本モデルを使用してその操作を定義します。

- ネーミング・モデル
- 情報モデル
- 機能モデル
- セキュリティ・モデル

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**を使用して表現されます。

図 2-1 に、識別名とディレクトリ情報ツリーの関係を示します。

図 2-1 ディレクトリ情報ツリー

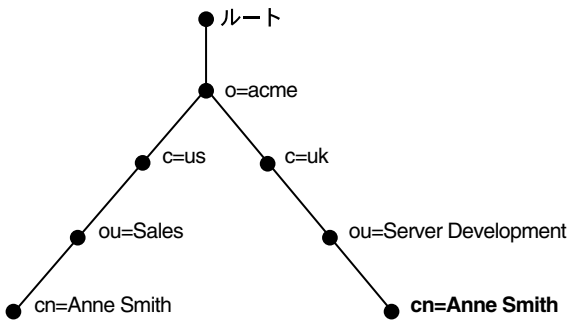


図 2-1 のディレクトリ情報ツリーは、Acme Corporation に所属する、Anne Smith という同じ名前を持つ 2 人の従業員のエントリを示しています。この図のディレクトリ情報ツリーは、地理的および組織的な線で構造化されています。左の分岐で表されている Anne Smith は、米国の販売部門に勤務しています。もう一方は、英国のサーバー開発部門に勤務しています。

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。前述の識別名では、相対識別名は `cn=Anne Smith` です。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` です。また、`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。図 2-1 に示すグローバル組織の中で 2 人の Anne Smith を混同しないためには、それぞれの完全な識別名を使用します。同じ組織単位内に同じ名前の従業員が 2 人いる場合は、他のメカニズムを使用できます。たとえば、一意の識別番号でこれらの従業員を識別します。

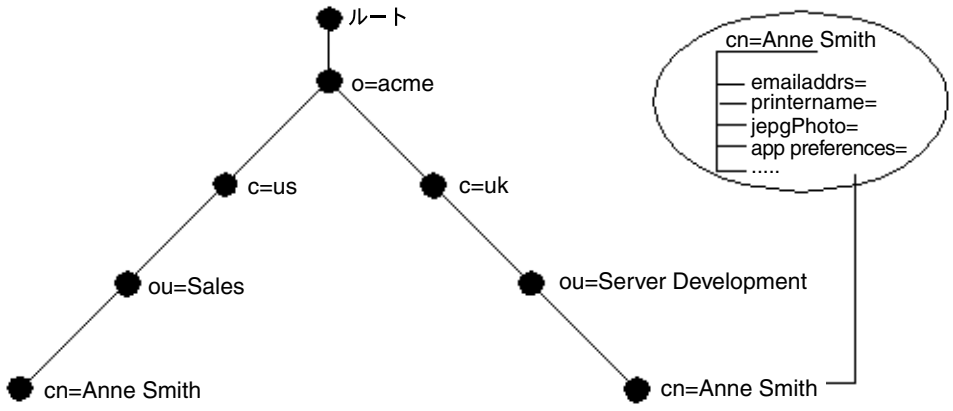
情報モデル

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。このモデルでは、定義特性としてエントリという概念を使用します。ディレクトリの**エントリ**とは、オブジェクトに関する情報の集合です。たとえば、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。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が含まれています。オンライン・ディレクトリでは、このような情報はそれぞれ**属性**と呼ばれます。一般的な従業員エントリには、役職名、電子メール・アドレス、電話番号などの属性が含まれています。

図 2-2 では、英国（uk）の Anne Smith に関するエントリにいくつかの属性があります。それぞれが Anne Smith についての固有の情報を提供します。ツリーの右側の円の中にリストされた属性は、`emailaddrs`、`printername`、`jpegPhoto` および `app preferences` です。図 2-2 のその他の黒丸も属性を持つエントリですが、それらの属性は示していません。

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**は実際の情報です。たとえば、`jobTitle` 属性に対する値には `manager` があります。

機能モデル

LDAP 機能モデルによって、ディレクトリ・エントリに対して実行できる操作が決まります。2-4 ページの表 2-1 に、3 種類の機能を示します。

表 2-1 LDAP 機能

機能	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、リクエスト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 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。値の追加と削除が可能です。 ■ 追加 : エントリをディレクトリに挿入します。 ■ 削除 : エントリをディレクトリから削除します。 ■ 相対識別名の変更 : エントリの名前を変更します。
認証	このカテゴリでは、バインド操作を定義します。バインドによって、クライアントはセッションを開始し、その ID をディレクトリに示すことができます。Oracle Internet Directory は、単純なクリアテキストのパスワードから公開鍵まで、様々な認証方式をサポートしています。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。

セキュリティ・モデル

LDAP セキュリティ・モデルによって、ディレクトリの情報を保護できます。このモデルはいくつかの部分から成ります。

- 認証

  ユーザー、ホストおよびクライアントの ID 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・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 ID を設定するプロセスです。ディレクトリ認証は、LDAP バインド操作で LDAP セッションを確立するときに行われます。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。

Oracle Internet Directory には、匿名、簡易および 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、ユーザーは匿名でログインできます。**匿名認証**では、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。そうすると、匿名ユーザーに対して指定されたすべての権限を使用できます。

簡易認証 **簡易認証**では、クライアントは暗号化されていない識別名とパスワードを使用し、サーバーに対して自己認証を行います。サーバーは、クライアントの識別名およびパスワードが、ディレクトリに格納されている識別名およびパスワードと一致していることを検証します。

Secure Sockets Layer（SSL）を使用した認証 **Secure Sockets Layer** 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。**証明書**の交換によりユーザーを認証します。交換された証明書は、信頼できる認証局によって検証されます。証明書は、エンティティの ID 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、表 2-2 に示す 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 暗号化および復号化のみが使用されます。
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、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 認証は次の 3 つの基本手順に従って行われます。

1. ユーザーは、SSL ポートで 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。

> **関連資料：** 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

認可プロセスにより、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が保証されます。ディレクトリ・サーバーは、特定のディレクトリ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。必要な権限がないと、操作は実行できません。

適切な認可が行われていることを保証するためにディレクトリ・サーバーが使用するメカニズムは、アクセス制御と呼ばれます。また、アクセス制御項目（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、ユーザーが変更できる操作属性として、Oracle Internet Directory に格納されています。通常、この ACI 属性値のリスト全体が 1 つのディレクトリ・オブジェクトに関連付けられています。このリストはアクセス制御リストと呼ばれます。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**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いことを保証します。この機能では、暗号化チェックサムを使用して、セキュアなメッセージ・ダイジェストを生成します。チェックサムは、**MD5** アルゴリズムまたは **Secure Hash Algorithm** を使用して作成されます。メッセージ・ダイジェストは各ネットワーク・パケットに組み込まれます。

データ・プライバシ

Oracle Internet Directory は、SSL による**公開鍵暗号化**を使用して、送信中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。ディレクトリは 2 つのレベルの暗号化をサポートしています。

- 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。

- RC4_40

  Oracle は、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・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しようとすると、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、指定された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 API を使用すると、「LDAP モデル」で説明した基本的な LDAP 操作を実行できます。この API には、C 版、PL/SQL 版および Java 版があります。最初の 2 つはディレクトリ SDK に含まれています。最後の API は、Sun 社提供の JNDI パッケージに含まれています。3 つとも TCP/IP 接続を使用します。これらは、LDAP バージョン 3 に基づいており、Oracle Internet Directory への SSL 接続をサポートしています。

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 スタート・ガイド
- DBMS_LDAP パッケージ・スタート・ガイド
- Java API スタート・ガイド

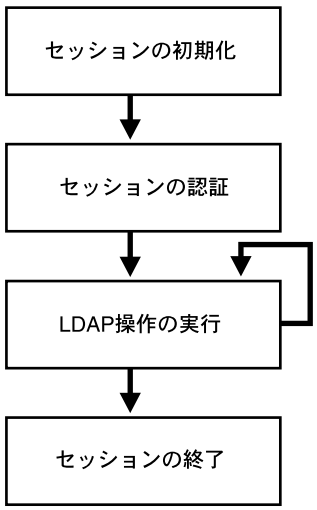
API の使用モデル

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



C API スタート・ガイド

C API を使用してアプリケーションを作成する場合、$ORACLE_HOME/ldap/public にあるヘッダー・ファイル ldap.h をインクルードする必要があります。また、$ORACLE_HOME/lib/libclntsh.so.10.1 にあるライブラリに動的にリンクすることも必要です。

> **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、14-41 ページの「C API の使用例」を参照してください。

DBMS_LDAP パッケージ・スタート・ガイド

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・コールの名前と構文は、C API の場合と同様です。このファンクションは、C API に関する **Internet Engineering Task Force** の最新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 の使用を開始するには、次のコマンド・シーケンスを使用して DBMS_LDAP をデータベースにロードします。

1. SQL*Plus を使用して、データベースにログインします。データベースが置かれている Oracle ホームにあるツールを実行します。SYSDBA として接続します。

   ```
   SQL> CONNECT / AS SYSDBA
   ```

2. 次のコマンドを使用して、API をデータベースにロードします。

   ```
   SQL> @?/rdbms/admin/catladap.sql
   ```

Java API スタート・ガイド

Java 開発者は、Sun 社の Java Naming and Directory Interface（JNDI）を使用して、Oracle Internet Directory の情報にアクセスできます。JNDI は次のリンクで提供されています。

```
http://java.sun.com/products/jndi
```

この章では Java API について説明しませんが、この後の項「JNDI を使用したセッションの初期化」で、Sun JNDI に対するラッパー・メソッドを使用して基本的な接続を確立する方法を示します。

LDAP セッションの初期化

C API に基づいたすべての LDAP 操作で、クライアントが LDAP サーバーとの LDAP セッションを確立することが求められます。PL/SQL API に基づいた 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。ほとんどの Java 操作では、Java Naming and Directory Interface（JNDI）接続が必要になります。ここで説明する oracle.ldap.util.jndi パッケージは、この接続の確立に伴う作業を簡素化します。

この項の項目は次のとおりです。

- C API 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
- JNDI 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

C ファンクション ldap_init() は、LDAP サーバーとのセッションを初期化します。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、初期化した後にオプションを設定できます。

ldap_init の構文は、次のとおりです。

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

表 2-3 に、このファンクションのパラメータを示します。

表 2-3 ldap_init() のパラメータ

パラメータ	説明
hostname	ディレクトリ・ホスト名またはドット表記文字列の IP アドレスを空白で区切ったリストが入ります。各ホスト名にポート番号を組み合せることもできます。その場合、ホスト名とポート番号をコロンで区切ります。 接続に成功するまで、ホストがリストの順序に従って試されます。 **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	接続先ディレクトリの 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() および ldap_open() の戻り値は、そのセッションへの後続のコールに渡す必要がある不透明な構造体へのセッション・ハンドル（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。オペレーティング・システムのエラー・レポート・メカニズムをチェックすると、コールに失敗した理由を確認できます。

DBMS_LDAP を使用したセッションの初期化

PL/SQL API では、ファンクション DBMS_LDAP.init() により LDAP セッションを開始します。この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、そのための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。コールで戻されたハンドルは、DBMS_LDAP でセッションに定義された後続のすべての LDAP 操作で使用する必要があります。API は、このセッション・ハンドルを使用して、オープン接続、未処理のリクエスト、その他の情報に関する状態をメンテナンスします。

1 つのデータベース・セッションで、必要な数の LDAP セッションを取得できます。ただし、同時にアクティブにできる接続の数は 64 までに制限されています。通常、1 つのデータベース・セッションが複数の LDAP セッションを持つのは、複数のサーバーから同時にデータベースを取得する必要がある場合、または複数の LDAP ID を使用するオープン・セッションが必要な場合です。

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。

JNDI を使用したセッションの初期化

oracle.ldap.util.jndi パッケージは、Sun 社の JNDI 実装に対するラッパー・メソッドを提供することで、基本的な接続をサポートします。JNDI を使用して接続を確立する場合は、次のリンクを参照してください。

http://java.sun.com/products/jndi

次に、非 SSL 接続を確立する oracle.ldap.util.jndi の実装を示します。

```
import oracle.ldap.util.jndi
import javax.naming.*;

public static void main(String args[])
{
  try{
       InitialDirContext ctx = ConnectionUtil.getDefaultDirCtx(args[0], // host
                                                               args[1],  // port
                                                               args[2],  // DN
                                                               args[3];  // password)
       // Do work
     }
  catch(NamingException ne)
  {
    // javax.naming.NamingException is thrown when an error occurs
  }
}
```

注意：

- *DN* と *password* は、バインド識別名とパスワードを示します。匿名バインドの場合、これらを "" に設定します。
- ConnectionUtil.getSSLDirCtx() を使用すると、認証なしの SSL 接続を確立できます。

LDAP セッションの認証

LDAP サーバーに対する操作を実行する個人またはアプリケーションは、最初に認証を受ける必要があります。これらのエンティティの 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 ID をユーザーに割り当てます。通常、匿名ユーザーは最小限の権限を与えられたディレクトリ・ユーザーです。

バインド操作の完了後、別のバインド操作が発生するか、LDAP セッションが終了する（unbind_s）まで、ディレクトリ・サーバーに新規の ID が保持されます。LDAP サーバーは、この ID を使用して、配置先の企業で規定されたセキュリティ・モデルを実施します。この ID は、識別された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ために役立ちます。

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ます。ネットワークがセキュアでない場合は、認証とデータ転送を伴うその他の LDAP 操作で SSL を使用すること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
- 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

C ファンクション ldap_simple_bind_s() を使用すると、ユーザーとアプリケーションは、識別名とパスワード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。

ファンクション ldap_simple_bind_s() 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 に、このファンクションのパラメータを示します。

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 ID
passwd	認証 ID 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 ID をユーザーまたはアプリケーションに割り当てます。

DBMS_LDAP を使用した LDAP セッションの認証

PL/SQL ファンクション simple_bind_s を使用すると、ユーザーとアプリケーションは、識別名とパスワードを使用してディレクトリへの認証を受けることができます。simple_bind_s 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;
```

このファンクションには、init 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あることに注意してください。

次の PL/SQL コード例は、初期化と認証を行う前述の PL/SQL ファンクションの実装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 LDAP サーバー yow.acme.com で初期化されます。このサーバーは、TCP/IP ポート番号 389 でリクエストをリスニングします。次に、ID cn=orcladmin（パスワードは welcome）が認証されます。認証が完了したら、通常の LDAP 操作を開始できます。

ディレクトリの検索

検索は、最も一般的な LDAP 操作です。アプリケーションは、複雑な検索条件を使用し、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。

- 検索操作のプログラム・フロー
- 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
- 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、LDAP サーバーが結果セット全体を戻すまで、検索ファンクションのコール元がブロックされることを意味します。

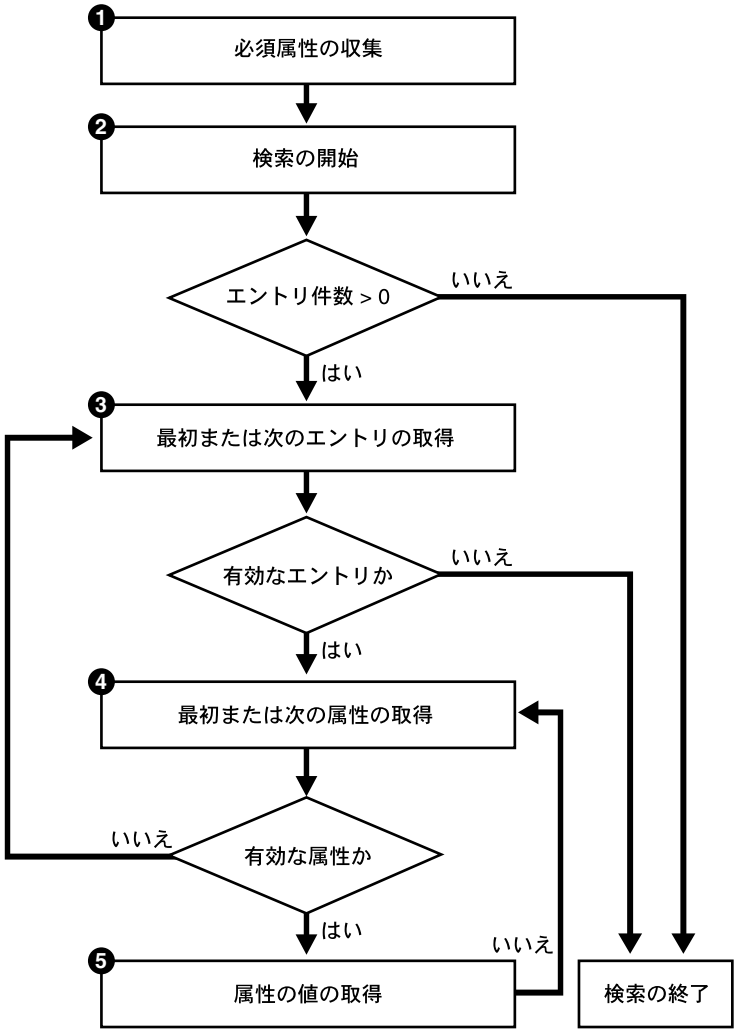
検索操作のプログラム・フロー

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、その属性を配列に入れます。
2. 選択した有効範囲オプションとフィルタを使用して、検索を開始します。
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から属性を取得します。
5. 手順 4 で取得した属性の値を取得し、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。

2-13 ページの図 2-4 は、これらの手順をフロー・チャートに表したものです。

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、ディレクトリ・サーバーが検索ベースと比較して検証するエントリの数を決定します。表 2-5 と 2-14 ページの図 2-5 に示す 3 つのオプションのいずれかを選択できます。

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、検索ベースに対応しているエントリのみを検索します。
SCOPE_ONELEVEL	ディレクトリ・サーバーは、検索対象を検索ベース・エントリの直接の子エントリに限定します。
SCOPE_SUBTREE	ディレクトリ・サーバーは、検索ベース・エントリとその下のサブツリー全体を対象に検索します。

図 2-5 3 つの有効範囲オプション

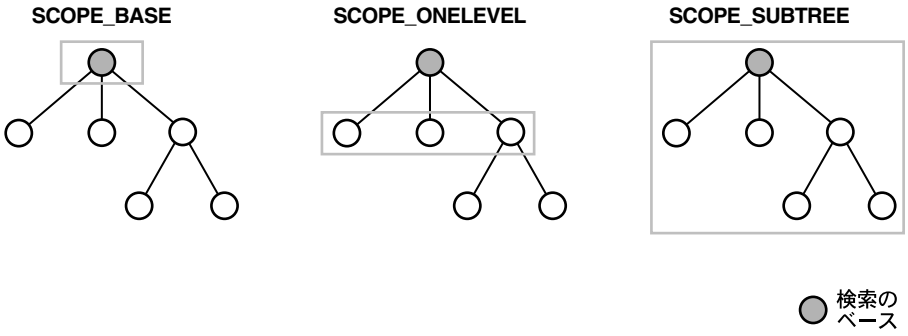


図 2-5 では、検索ベースはグレーの丸で表されています。検索対象のエントリは、薄い色の四角形で囲まれています。

フィルタ

検索フィルタは、検索対象を特定のタイプのエントリに限定するための式です。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。表 2-6 に示すように、6 種類の検索フィルタがあります。これらは、*attribute operator value* という書式で入力します。

表 2-6 検索フィルタ

フィルタ・タイプ	書式	例	一致
等価	(*att=value*)	(sn=Keaton)	Keaton と完全に等しい姓。
近似	(*att~=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 を含み、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。

ブール演算子や接頭辞表記法を使用してこれらのフィルタを組み合せると、より複雑なフィルタを構成できます。2-15 ページの表 2-7 に例を示します。この例では、& 文字は AND、| 文字は OR、！文字は NOT を表します。

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。

表 2-7 の複合フィルタを組み合せて、さらに複雑なネストした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

C ファンクション ldap_search_s() は、ディレクトリの同期検索を実行します。

ldap_search_s() の構文は、次のとおりです。

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

ldap_search_s は、いくつかのサポート・ファンクションと連携して、検索対象を絞り込みます。次の手順は、これらの C ファンクションが検索操作のプログラム・フローにどのように対応するかを示しています。これらの C ファンクションの詳細は、第 14 章「C API リファレンス」を参照してください。

1. 検索によって戻される属性を決定し、その属性を文字列の配列に入れます。配列はヌル文字で終了する必要があります。
2. ldap_search_s() を使用して、検索を開始します。有効範囲オプションとフィルタで検索対象を絞り込みます。
3. ldap_first_entry() ファンクションまたは 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. 手順 3 で取得したエントリから属性を取得します。これには、ldap_first_attribute() ファンクションまたは ldap_next_attribute() ファンクションを使用します。
5. 手順 4 で取得した属性のすべての値を取得し、その値をローカル変数にコピーします。これには、ldap_get_values() ファンクションまたは ldap_get_values_len() ファンクションを使用します。
6. エントリのすべての属性が処理されるまで、手順 4 を繰り返します。
7. すべてのエントリが処理されるまで、手順 3 を繰り返します。

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	検索のベースの識別名です。
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。
attrso	1 に設定すると、属性のみが戻されます。
res	この引数は検索結果を戻します。

DBMS_LDAP を使用したディレクトリの検索

PL/SQL API を使用する場合、ファンクション DBMS_LDAP.search_s() を使用してディレクトリ検索を実行します。

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;
```

このファンクションは、2-16 ページの表 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。

search_s は、いくつかのサポート・ファンクションと連携して、検索対象を絞り込みます。次の手順は、すべての PL/SQL ファンクションが検索操作のプログラム・フローにどのように対応するかを示しています。

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() を使用して、検索を実行します。有効範囲オプションとフィルタで検索対象を絞り込みます。
3. DBMS_LDAP.first_entry() または DBMS_LDAP.next_entry() を使用して、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から属性を取得します。これには、DBMS_LDAP.first_attribute() または DBMS_LDAP.next_attribute() を使用します。
5. 手順 4 で取得した属性のすべての値を取得し、その値をローカル変数にコピーします。これには、DBMS_LDAP.get_values() または DBMS_LDAP.get_values_len() を使用します。
6. エントリのすべての属性が処理されるまで、手順 4 を繰り返します。
7. すべてのエントリが処理されるまで、手順 3 を繰り返します。

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
- 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、ディレクトリ関連の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。C API では、そのために ldap_unbind_s() ファンクションを使用します。

ldap_unbind_s() の構文は、次のとおりです。

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

ldap_unbind_s() のコールが正常終了すると、ディレクトリへの TCP/IP 接続がクローズします。また、LDAP セッションで使用されたシステム・リソースの割当てが解除されます。最後に、整数 LDAP_SUCCESS がコール元に戻されます。ldap_unbind_s() を起動した後は、その他の LDAP 操作は実行できません。ldap_init() を使用して新しいセッションを開始する必要があります。

DBMS_LDAP を使用したセッションの終了

PL/SQL API が使用されている場合、DBMS_LDAP.unbind_s() ファンクションにより LDAP セッションを破棄します。unbind_s の構文は、次のとおりです。

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

unbind_s により、ディレクトリへの TCP/IP 接続がクローズします。また、LDAP セッションで使用されたシステム・リソースの割当てが解除されます。最後に、整数 DBMS_LDAP.SUCCESS がコール元に戻されます。unbind_s を起動した後は、その他の LDAP 操作は実行できません。init ファンクションを使用して新しいセッションを開始する必要があります。

3

LDAP プロトコルに対する拡張機能

この章では、Oracle Internet Directory10*g*（10.1.4.0.1）で使用可能な LDAP プロトコルに対する拡張機能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。

- SASL 認証
- コントロールの使用
- エンド・ユーザーの代理となるプロキシ
- 動的パスワード・ベリファイアの作成
- 階層検索の実行
- ソート済の LDAP 検索結果
- ページング済の LDAP 検索結果

SASL 認証

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
- 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

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 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。C API でサポートされているのは `auth` モードのみです。

関連項目：

- Java 固有の情報については、5-8 ページの「Digest-MD5 を使用した SASL 認証の実行」および 5-8 ページの「例 : SASL Digest-MD5 の auth-int モードおよび auth-conf モードの使用」を参照してください。
- C 固有の情報については、14-11 ページの「ディレクトリに対する認証」および 14-13 ページの「Oracle の拡張機能を使用した SASL 認証」を参照してください。

SASL Digest-MD5 メカニズムには 3 つのモードがあり、それぞれが異なるセキュリティ・レベルまたは Quality of Protection（QoP）を表しています。その内容は、次のとおりです。

- `auth`: 認証のみ。初期バインドにのみ認証が必要です。その後、情報はクリアテキストで渡されます。
- `auth-int`: 認証および整合性。初期バインドには認証が必要です。その後、チェックサムを使用してデータの整合性が保証されます。
- `auth-conf`: 認証および機密保護。初期バインドには認証が必要です。その後、暗号化を使用してデータが保護されます。次に示す 5 つの暗号方式を使用できます。
  - DES
  - 3DES
  - RC4
  - RC4-56
  - RC4-40

これらはすべて対称型暗号化アルゴリズムです。

10*g*（10.1.4.0.1）より前の Oracle Internet Directory では、Digest-MD5 メカニズムの `auth` モードのみがサポートされていました。10*g*（10.1.4.0.1）の Oracle Internet Directory では、jdk1.4 API の Java Naming and Directory Interface（JNDI）、または OpenLDAP Java API で 3 つのモードがすべてサポートされています。Oracle LDAP SDK でサポートされているのは `auth` モードのみです。

Oracle Internet Directory の SASL Digest-MD5 認証では、特別な設定をしなくても、レルムではなくユーザーまたはパスワードに基づいた静的な SASL Digest-MD5 ベリファイアの生成がサポートされています。レルムに基づいて SASL Digest-MD5 を使用する場合は、新しいユーザーをプロビジョニングする前に関連するパスワード・ポリシーで `orclpasswordencryptionenable` 属性の値を 1 に変更して、どちらからもパスワードを生成できるようにしておく必要があります。値を変更する LDIF ファイルは次のようになります。

```
dn: cn=default,cn=pwdPolicies,cn=Common,cn=Products,cn=OracleContext
changetype: modify
replace: orclpwdencryptionenable
orclpwdencryptionenable: 1
```

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。

関連資料：

- Internet Engineering Task Force の Web サイト（`http://www.ietf.org`）
- OpenLDAP のクラス・ライブラリ（`http://www.openldap.org`）

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レスポンスを送信して応答します。レスポンスは、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、クライアントのレスポンスを復号化し、検証します。

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 ID が含まれた初期レスポンスを送信します。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 ID 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 ID 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 ID をリクエストする）場合、認可 ID 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 ID（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。

コントロールの使用

RFC 2251 で定義されているように、LDAP v3 プロトコルではコントロールによる拡張機能が許可されています。Oracle Internet Directory では複数のコントロールをサポートしています。一部は標準で、RFC で説明されています。階層検索用の CONNECT_BY コントロールなど、それ以外のコントロールは Oracle 固有です。コントロールは、Java または C で使用できます。

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。

C における LDAP コントロールの使用の詳細は、14-15 ページの「コントロールの使用」を参照してください。

Java における LDAP コントロールの使用の詳細は、`http://java.sun.com/products/jndi` の JNDI パッケージ `javax.naming.ldap` に関するドキュメントを参照してください。

Oracle Internet Directory 10*g*（10.1.4.0.1）では、次のコントロールがサポートされています。

表 3-1 Oracle Internet Directory でサポートされているコントロール

オブジェクト識別子	名前	説明
2.16.840.1.113730.3.4.2	GSL_MANAGE_DSA_CONTROL	Oracle Internet Directory の参照、動的グループおよび別名オブジェクトの管理に使用されます。詳細は、`http://www.ietf.org` の RFC 3296「Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories」を参照してください。
2.16.840.1.113894.1.8.1	OID_RESET_PROXYCONTROL_IDENTITY	確立された LDAP 接続で ID のプロキシ・スイッチを実行するために使用されます。たとえば、アプリケーション A がディレクトリ・サーバーに接続していて、アプリケーション B に切り替える必要があるとします。アプリケーション B の資格証明を指定することで単純にリバインドできますが、資格証明がなくても、アプリケーションで ID を切り替えるためのプロキシ・メカニズムを使用できる場合があります。アプリケーション A に Oracle Internet Directory でアプリケーション B のかわりとなる権限がある場合には、このコントロールを使用して、アプリケーション A をアプリケーション B に切り替えることができます。
2.16.840.1.113894.1.8.2	OID_APPLYUSEPASSWORD_POLICY	アプリケーションにユーザーのベリファイアを送信する前に、Oracle Internet Directory によるアカウント・ロックアウトの確認が必要なアプリケーションによって送信されます。Oracle Internet Directory によりベリファイア検索リクエスト内にこのコントロールが検出され、ユーザー・アカウントがロックされている場合、Oracle Internet Directory はアプリケーションにベリファイアを送信しません。適切なパスワード・ポリシー・エラーが送信されます。
2.16.840.1.113894.1.8.3	CONNECT_BY	詳細は、3-9 ページの「階層検索の実行」を参照してください。
2.16.840.1.113894.1.8.4	OID_CLIENT_IP_ADDRESS	Oracle Internet Directory によって IP ロックアウトが実行される場合に、クライアントがエンド・ユーザー IP アドレスを送信するために使用されます。
2.16.840.1.113894.1.8.5	GSL_REQDATTR_CONTROL	動的グループとともに使用されます。ディレクトリ・サーバーに、メンバーシップ・リストではなくメンバーの特定の属性を読み取るよう指示します。
2.16.840.1.113894.1.8.6	OID_PASSWORD_REQUEST_CONTROL	パスワード・ポリシー・コントロール。クライアントがサーバーからのレスポンスを取得するために送信するリクエスト・コントロール。

表 3-1 Oracle Internet Directory でサポートされているコントロール（続き）

オブジェクト識別子	名前	説明
2.16.840.1.113894.1.8.7	OID_PASSWORD_EXPWARNING_CONTROL	パスワード・ポリシー・コントロール。pwdExpireWarning 属性が有効化されていて、クライアントからリクエスト・コントロールが送信された場合にサーバーが送信するレスポンス・コントロール。レスポンス・コントロール値には、パスワードの有効期限を表す秒単位の時間が含まれます。
2.16.840.1.113894.1.8.8	OID_PASSWORD_GRACELOGIN_CONTROL	パスワード・ポリシー・コントロール。猶予期間ログインが構成されていて、クライアントからリクエスト・コントロールが送信された場合にサーバーが送信するレスポンス・コントロール。レスポンス・コントロール値には、猶予期間ログインの残数が含まれます。
2.16.840.1.113894.1.8.9	OID_PASSWORD_MUSTCHANGE_CONTROL	パスワード・ポリシー・コントロール。強制パスワードのリセットが有効化されていて、クライアントからリクエスト・コントロールが送信された場合にサーバーが送信するレスポンス・コントロール。クライアントは、ユーザーがこのコントロールを受信する際にパスワードの変更を強制する必要があります。
2.16.840.1.113894.1.8.14	OID_DYNAMIC_VERIFIER_REQUEST_CONTROL	サーバーによる動的パスワード・ベリファイアの作成が必要な場合にクライアントが送信するリクエスト・コントロール。サーバーは、リクエスト・コントロールのパラメータを使用してベリファイアを作成します。
2.16.840.1.113894.1.8.15	OID_DYNAMIC_VERIFIER_RESPONSE_CONTROL	エラーが発生した場合にサーバーがクライアントに送信するレスポンス・コントロール。レスポンス・コントロールにはエラー・コードが含まれています。
2.16.840.1.113894.1.8.16	OID_APPLYALLPWDPOLICIES_CONTROL	このコントロールがベリファイア検索リクエストに含まれている場合は、ユーザーに適用可能なすべてのパスワード・ポリシーがベリファイア検索に適用されます。
2.16.840.1.113894.1.8.23	GSL_CERTIFICATE_CONTROL	証明書検索コントロール。ユーザー証明書の検索方法を指定するためにクライアントが送信するリクエスト・コントロール。
1.2.840.113556.1.4.473	OID_SEARCH_SORTING_REQUEST_CONTROL	詳細は、3-11 ページの「ソート済の LDAP 検索結果」を参照してください。
1.2.840.113556.1.4.319	OID_SEARCH_PAGING_CONTROL	詳細は、3-11 ページの「ページング済の LDAP 検索結果」を参照してください。

使用している Oracle Internet Directory インストールで使用可能なコントロールを確認するには、次のように入力します。

```
ldapsearch -p port -b "" -s base "objectclass=*"
```

`supportedcontrol=` で始まるエントリを探します。

エンド・ユーザーの代理となるプロキシ

多くの場合、アプリケーションはエンド・ユーザーの代理を務めることが必要になる操作を実行する必要があります。たとえば、エンド・ユーザーのリソース・アクセス記述子を取得する場合などです。（リソース・アクセス記述子の詳細は、『Oracle Internet Directory 管理者ガイド』の概念の章を参照してください。）

プロキシ・スイッチは、実行時に JNDI コンテキストで行われます。LDAP v3 の機能であるプロキシは、`InitialDirContext` のサブクラス `InitialLdapContext` を使用しないと実行できません。Oracle の拡張機能 `oracle.ldap.util.jndi.ConnectionUtil` を使用して接続を確立する場合（後述の例）、`InitialLdapContext` が常に戻されます。JNDI を使用して接続を確立する場合は、`InitialLdapContext` が戻されるようにしてください。

エンド・ユーザーへのプロキシ・スイッチを実行するには、ユーザー識別名が必要です。識別名を取得する方法については、次の URL で `oracle.ldap.util.User` クラスの実装例を参照してください。

`http://www.oracle.com/technology/sample_code/`

「Sample Applications-Fusion Middleware」の下の「Oracle Identity Management」リンクから、「Sample Application Demonstrating Proxy Switching using Oracle Internet Directory Java API」を探してください。

次のコードは、プロキシ・スイッチがどのように行われるかを示しています。

```
import oracle.ldap.util.jndi.*;
import javax.naming.directory.*;
import javax.naming.ldap.*;
import javax.naming.*;

public static void main(String args[])
{
  try{
       InitialLdapContext appCtx=ConnectionUtil.getDefaultDirCtx(args[0], // host
                                                                 args[1], // port
                                                                 args[2], // DN
                                                                 args[3]; // pass)
         // Do work as application
         // . . .
         String userDN=null;
         // assuming userDN has the end user DN value
         // Now switch to end user
         ctx.addToEnvironment(Context.SECURITY_PRINCIPAL, userDN);
         ctx.addToEnvironment("java.naming.security.credentials", "");
         Control ctls[] = {
           new ProxyControl()
         };
         ((LdapContext)ctx).reconnect(ctls);
         // Do work on behalf of end user
         // . . .
       }
    catch(NamingException ne)
      {
       // javax.naming.NamingException is thrown when an error occurs
      }
}
```

前述のコードの `ProxyControl` クラスは、`javax.naming.ldap.Control` を実装します。LDAP コントロールの詳細は、『Oracle Identity Management ユーザー・リファレンス』の LDAP コントロールに関する項を参照してください。次に、`ProxyControl` クラスの例を示します。

```
import javax.naming.*;
import javax.naming.ldap.Control;
import java.lang.*;
```

```
public class ProxyControl implements Control {

   public byte[] getEncodedValue() {
      return null;
   }

   public String getID() {
      return "2.16.840.1.113894.1.8.1";
   }

   public boolean isCritical() {
      return false;
   }
}
```

動的パスワード・ベリファイアの作成

LDAP 認証 API を変更して、アプリケーション・パスワードを動的に（ユーザーがアプリケーションにログインするときに）生成できます。この機能は、パスワード・ベリファイアのパラメータを実行時にのみ指定するアプリケーションの必要性を満たすことを目的としています。

この項の項目は次のとおりです。

- 動的パスワード・ベリファイアのリクエスト・コントロール
- DynamicVerifierRequestControl の構文
- ハッシング・アルゴリズムに必要なパラメータ
- 認証 API の構成
- 動的パスワード・ベリファイアのレスポンス・コントロール
- 動的ベリファイア・フレームワークに対する権限の取得

動的パスワード・ベリファイアのリクエスト・コントロール

パスワード・ベリファイアを動的に作成するには、LDAP 認証 API の `ldap_search` または `ldap_modify` を変更して、パスワード・ベリファイアのパラメータを含める必要があります。LDAP コントロール `DynamicVerifierRequestControl` は、このパラメータを転送するためのメカニズムです。これは、パスワード・ベリファイアの静的作成に使用されるパスワード・ベリファイア・プロファイルにかわるものです。ただし、動的ベリファイアの場合にも、静的ベリファイアと同様に、ディレクトリ属性 `orclrevpwd`（同期）および `orclunsyncrevpwd`（非同期）が存在し、これらの属性に値が移入されていることが必要になります。

`orclrevpwd` を生成する場合は、ユーザー・レルムにあるパスワード・ポリシー・エントリの `orclpwdencryptionenable` 属性を `1` に設定する必要があることに注意してください。この属性を設定しないと、ユーザーが認証を受けようとしたときに例外がスローされます。`orclunsyncrevpwd` を生成するには、暗号タイプ 3DES をエントリ `cn=defaultSharedPINProfileEntry,cn=common,cn=products,cn=oraclecontext` に追加する必要があります。

DynamicVerifierRequestControl の構文

リクエスト・コントロールは次のようになります。

```
DynamicVerifierRequestControl
controlOid:  2.16.840.1.113894.1.8.14
criticality:  FALSE
controlValue: an OCTET STRING whose value is the BER encoding of the following type:

ControlValue ::= SEQUENCE {

                 version [0]
                 crypto  [1] CHOICE OPTIONAL {
                    SASL/MD5  [0] LDAPString,
                    SyncML1.0 [1] LDAPString,
                    SyncML1.1 [2] LDAPString,
                    CRAM-MD5  [3] LDAPString },
                 username [1] OPTIONAL LDAPString,
                 realm    [2] OPTIONAL LDAPString,
                 nonce    [3] OPTIONAL LDAPString,
                }
```

コントロール構造のパラメータは出現順に渡す必要があることに注意してください。表 3-2 に、このパラメータを示します。

表 3-2 DynamicVerifierRequestControl のパラメータ

パラメータ	説明
controlOID	コントロール構造を一意に識別する文字列。
crypto	ハッシング・アルゴリズム。コントロール構造に指定された 4 種類のうちいずれかを選択します。
username	ユーザーの識別名（DN）。この値は常に含める必要があります。
realm	ランダムに選択されたレルム。ユーザーが属する ID 管理レルムの場合があります。アプリケーション・レルムの場合もあります。SASL/MD5 アルゴリズムにのみ必要です。
nonce	ランダムに選択された任意の値。SYNCML1.0 と SYNCML1.1 に必要です。

ハッシング・アルゴリズムに必要なパラメータ

表 3-3 に、動的パスワード・ベリファイアの作成に使用される 4 種類のハッシング・アルゴリズムを示します。この表には、各アルゴリズムの構成要素として使用されるパラメータも示します。ユーザー名とパスワードのパラメータはすべてのアルゴリズムで使用しますが、realm および nonce パラメータを使用するかどうかはアルゴリズムによって異なることに注意してください。

表 3-3 ハッシング・アルゴリズムに必要なパラメータ

アルゴリズム	必要なパラメータ
SASL/MD5	username、realm、password
SYNCML1.0	username、password、nonce
SYNCML1.1	username、password、nonce
CRAM-MD5	username、password

認証 API の構成

パスワード・ベリファイアを動的に生成する必要のあるアプリケーションの場合、認証 API に DynamicVerifierRequestControl を組み込む必要があります。ldap_search または ldap_compare に controlOID とコントロール値をパラメータとして含めることが必要です。これらの API は、「DynamicVerifierRequestControl の構文」に示したコントロール値を BER エンコードし、controlOID とコントロール値の両方をディレクトリ・サーバーに送信する必要があります。

ldap_search の使用時に渡されるパラメータ

アプリケーションでユーザーを認証するには、ldap_search を使用してコントロール構造を渡します。ldap_search を使用する場合、ディレクトリは作成したパスワード・ベリファイアをクライアントに渡します。

ldap_search には、ユーザーの識別名、controlOID およびコントロール値を含める必要があります。ユーザーのパスワードがシングル・サインオン・パスワードの場合、渡される属性は authpassword です。パスワードが数値 PIN やその他の非同期パスワードの場合、渡される属性は orclpasswordverifier;orclcommonpin です。

ldap_compare の使用時に渡されるパラメータ

Oracle Internet Directory でユーザーを認証するには、ldap_compare を使用してコントロール構造を渡します。この場合、ディレクトリはベリファイアを保持し、ユーザー自身を認証します。

ldap_search と同様に、ldap_compare には、ユーザーの識別名、controlOID、コントロール値およびユーザーのパスワード属性を含める必要があります。ldap_compare の場合、パスワード属性は orclpasswordverifier;orclcommonpin（非同期）です。

動的パスワード・ベリファイアのレスポンス・コントロール

エラーが発生すると、ディレクトリは LDAP コントロール DynamicVerifierResponseControl をクライアントに送信します。このレスポンス・コントロールにはエラー・コードが含まれています。レスポンス・コントロールが送信するエラー・コードの詳細は、『Oracle Internet Directory 管理者ガイド』のトラブルシューティングの章を参照してください。

動的ベリファイア・フレームワークに対する権限の取得

パスワード・ベリファイアをディレクトリで動的に作成するには、アプリケーション ID をディレクトリ管理者の VerifierServices グループに追加する必要があります。このタスクを実行しないと、ディレクトリは LDAP_INSUFFICIENT_ACCESS エラーを戻します。

階層検索の実行

LDAP 検索ファンクションに渡すことのできるサーバー・コントロールの 1 つは CONNECT_BY です。これは、Oracle 固有のコントロールで、階層全体を検索できます。たとえば、CONNECT_BY コントロールを使用せずに group1 のすべてのユーザーを検索すると、検索ファンクションは group1 の直接のメンバーであるユーザーのみを返します。CONNECT_BY コントロールを渡すと、検索ファンクションは階層全体を検索します。group2 が group1 のメンバーの場合には、検索で group2 のユーザーも返されます。group3 が group2 のメンバーの場合には、検索で group3 のユーザーも返されます。

CONNECT_BY コントロールの新機能

10*g*（10.1.4.0.1）では、CONNECT_BY コントロールは次の 2 点において拡張されています。

- 階層をどちらの方向にも検索できるようになりました。つまり、エントリ内に含まれているすべてのコンテナだけでなく、エントリが含まれているすべてのコンテナも検索できます。
- 検索する階層のレベル数を指定できるようになりました。

CONNECT_BY コントロールの値フィールド

以前のリリースでは、CONNECT_BY コントロールには値は必要ありませんでした。新機能により、次に示す値のいずれか、または両方を CONNECT_BY に渡すことができるようになりました。

- 階層構成属性 - 検索対象の属性を表す文字列。この値は、エントリが含まれるすべてのコンテナを検索する場合にのみ必要です。エントリ内に含まれるコンテナを検索する場合、この情報は検索フィルタによって提供されるため、この値を指定する必要はありません。
- レベル数 - 検索するレベル数を表す整数。値を 0 にすると、すべてのレベルが検索されます。デフォルト値は 0 のため、すべてのレベルを検索する場合にはこの値を渡す必要はありません。

例 1: ユーザーが属するすべてのグループの検索

(member=cn=jsmith) などのフィルタを使用した場合、階層構成属性メンバーは検索フィルタに含まれているため指定する必要はありません。デフォルトは 0 であるため、レベル数の値を渡す必要はありません。

例 2: ユーザーが直接が属するグループのみの検索

例 1 と同じフィルタを使用し、整数のコントロール値 1 を渡します。結果は、CONNECT_BY コントロールを使用しなかった場合と同じになります。

例 3: グループのすべてのメンバーの検索

この場合、検索フィルタには (objectclass=*) が指定されていますが、group1 のすべてのメンバーを検索する場合、階層全体を検索するための属性は member です。この検索では、文字列 "member" を階層構成属性として渡す必要があります。デフォルトは 0 であるため、レベル数の値を渡す必要はありません。

例 4: ユーザーのすべてのマネージャの検索

ユーザー jsmith のすべてのマネージャを検索する点を除き、例 3 に似ているため、階層全体を検索するための属性は manager です。この検索では、文字列 "manager" を渡します。デフォルトは 0 であるため、レベル数の値を渡す必要はありません。

関連項目：

- 14-17 ページの「ldap_search_ext、ldap_search_ext_s、ldap_search および ldap_search_s」
- 14-15 ページの「コントロールの使用」

ソート済の LDAP 検索結果

Oracle Internet Directory 10*g*（10.1.4.0.1）では、IETF RFC 2891 で説明されているように、LDAP 検索からソート済の結果を取得できるようになりました。タイプ 1.2.840.113556.1.4.473 のコントロールを検索ファンクションに渡して、ソート済の結果をリクエストします。サーバーにより戻されるレスポンス・コントロールは、タイプ 1.2.840.113556.1.4.474 です。エラー処理およびその他の詳細は、RFC 2891 を参照してください。

> **関連資料：** `http://www.ietf.org` の IETF RFC 2891「LDAP Control Extension for Server Side Sorting of Search Results」

ソートおよびページングを一緒に使用できます。

RFC 2891 の Oracle Internet Directory 実装には、次の制限事項があります。

- コントロール値でサポートされているのは、1 つの `attributeType` のみです。
- 各属性のスキーマに定義されているデフォルトの順序付けルールが使用されます。
- 言語ソートはサポートされていません。
- デフォルトのソート順序は昇順です。
- ソート・キーが複数の値を含む属性で、エントリにもその属性に対して複数の値があり、ソート順序に影響するその他のコントロールが存在しない場合、サーバーでは、その属性の順序付けルールに応じて最後の値が使用されます。
- ソート属性は検索可能である必要があります。つまり、Oracle Internet Directory でカタログ化されている属性である必要があります。

ページング済の LDAP 検索結果

Oracle Internet Directory 10*g*（10.1.4.0.1）では、IETF RFC 2696 で説明されているように、LDAP 検索からページング済の結果を取得できるようになりました。タイプ 1.2.840.113556.1.4.319 のコントロールを検索ファンクションに渡して、ソート済の結果をリクエストします。詳細は、RFC 2696 を参照してください。

> **関連資料：** `http://www.ietf.org` の IETF RFC 2696「LDAP Control Extension for Simple Paged Results Manipulation」

ソートおよびページングを一緒に使用できます。

RFC 2696 の Oracle Internet Directory 実装には、次の制限事項があります。

- ACI により検索結果から複数のエントリが部分的にブロックされた場合、ページ内のエントリ数はページ・サイズより少なくなる可能性があります。
- ページングのレスポンス・コントロールには、合計のエントリ件数の見積りは含まれません。戻り値は常に 0 です。

4

標準的な 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、Java および PL/SQL LDAP API に対する Oracle の拡張機能について説明します。第 4 章では Java の拡張機能について説明します。第 5 章では PL/SQL の拡張機能について説明します。C API に対する拡張機能は、Oracle ではサポートされません。

この章では、次の項目について説明します。

- サンプル・コード
- 標準的な API に対する Oracle の拡張機能の使用
- ディレクトリへのアプリケーション ID の作成
- ユーザーの管理
- グループの管理
- レルムの管理
- ディレクトリ・サーバーの検出

サンプル・コード

サンプル・コードは次の URL で入手できます。

`http://www.oracle.com/technology/sample_code/`

「Sample Applications-Fusion Middleware」の下の「Oracle Identity Management」リンクを探してください。

標準的な API に対する Oracle の拡張機能の使用

オラクル社が既存の API に追加した API は、次の機能を実現します。

- ユーザー管理

  アプリケーションで各種ユーザー・プロパティを設定または取得できます。

- グループ管理

  アプリケーションでグループ・プロパティの問合せができます。

- レルム管理

  アプリケーションで ID 管理レルムに関するプロパティを設定または取得できます。

- サーバー検出管理

  アプリケーションでドメイン・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。

この後の各項では、これらの機能を 1 つずつ詳細に解説します。一般的なタスク（接続の確立およびクローズ、API の拡張機能で検索できないディレクトリ・エントリの検索など）については、基礎となる API をアプリケーションで使用する必要があります。

図 4-1 に、API の拡張機能を使用した場合のプログラム・フローを示します。

図 4-1 API の拡張機能のプログラム・フロー

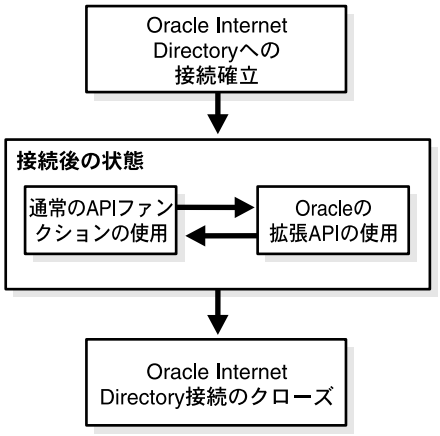


図 4-1 に示すように、アプリケーションはまず Oracle Internet Directory への接続を確立します。その後、標準的な API ファンクションと API の拡張機能を区別なく使用できます。

ディレクトリへのアプリケーション ID の作成

アプリケーションが LDAP API とその拡張機能を使用するためには、LDAP 接続を確立する必要があります。接続が確立されたら、操作を実行する権限を得る必要があります。しかし、アプリケーションの ID がディレクトリにない場合は、どちらのタスクも実行できません。

アプリケーション ID の作成

ディレクトリにアプリケーションの ID を作成するのは比較的簡単です。このようなエントリに必要なのは、`orclApplicationEntity` と `top` の 2 つのオブジェクト・クラスのみです。Oracle Directory Manager または LDIF ファイルを使用してエントリを作成できます。LDIF 表記法では、エントリは次のようになります。

```
dn: orclapplicationcommonname=application_name
changetype: add
objectclass:top
objectclass: orclApplicationEntity
userpassword: password
```

`userpassword` に指定する値は、アプリケーションがディレクトリへのバインドに使用する値です。

アプリケーション ID への権限の割当て

アプリケーションに割り当てることのできる権限の詳細は、『Oracle Internet Directory 管理者ガイド』の Oracle テクノロジ配置のための権限の委任に関する章を参照してください。正しい権限セットが特定できたら、アプリケーション・エンティティの識別名を適切なディレクトリ・グループに追加します。前述のドキュメントでは、Oracle Directory Manager または `ldapmodify` コマンドを使用してこのタスクを実行する方法が説明されています。

ユーザーの管理

この項では、LDAP API のユーザー管理機能について説明します。

ディレクトリ対応アプリケーションは、次の操作を実行する必要があります。

- ユーザー・エントリのプロパティの取得

  このプロパティは、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ます。

- 拡張ユーザー・プリファレンスの取得

  このプリファレンスはユーザーに適用されますが、ユーザー・エントリを含む DIT とは異なる DIT に格納されています。拡張ユーザー・プリファレンスは、すべてのアプリケーションに共通のユーザー・プロパティ、または 1 つのアプリケーションに固有のユーザー・プロパティです。前者のタイプのプロパティは、Oracle コンテキストの共通の位置に格納されています。後者のタイプのプロパティは、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

- 単純な名前と資格証明が与えられたユーザーの認証

  通常、アプリケーションは完全に修飾された識別名、GUID または単純なユーザー名を使用してユーザーを識別します。ホスト環境では、ユーザー名とレルム名の両方を識別に使用することがあります。

グループの管理

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、特定の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

レルムの管理

ID 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。ディレクトリ対応の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

ディレクトリ・サーバーの検出

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

関連資料：

- 「Discovering LDAP Services with DNS」Michael P. Armijo 著（次の URL を参照）

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

- 「A DNS RR for specifying the location of services (DNS SRV)」、Internet RFC 2782（同じ URL を参照）

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、ソースを指定します。この場合、ソースは DNS です。

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
- SSL モード

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（reshdl）で戻されます。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ソートされます。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。

> **関連資料：**『Oracle Internet Directory 管理者ガイド』のディレクトリ管理に関する章を参照してください。

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ソートされ、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

次の環境変数は、DNS サーバー検出のデフォルト動作を変更します。

表 4-1 DNS 検出の環境変数

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。

関連資料 : 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。

5

JNDI に対する Java API 拡張機能の使用

この章では、標準のディレクトリ API に Java の拡張機能を使用して、第 3 章で紹介した多くの操作を実行する方法について説明します。この章では使用例を示します。標準 API に対する Oracle の拡張機能の詳細は、『Oracle Internet Directory API Reference』を参照してください。

この章の項目は次のとおりです。

- サンプル・コード
- Java 拡張機能のインストール
- oracle.java.util パッケージを使用した LDAP オブジェクトのモデル化
- PropertySetCollection、PropertySet および Property クラス
- ユーザーの管理
- ユーザーの認証
- ユーザーの作成
- ユーザー・オブジェクトの取得
- レルムからのオブジェクトの取得
- 例 : OracleAS Single Sign-On のログイン名の検索
- ディレクトリ・サーバーの検出
- 例 : ディレクトリ・サーバーの検出
- Digest-MD5 を使用した SASL 認証の実行
- 例 : SASL Digest-MD5 の auth-int モードおよび auth-conf モードの使用

サンプル・コード

サンプル・コードは次の URL で入手できます。

http://www.oracle.com/technology/sample_code/

「Sample Applications-Oracle Application Server」の下の「Oracle Identity Management」リンクを探してください。

Java 拡張機能のインストール

Java 拡張機能は、LDAP クライアントのインストール時に標準の Java API とともにインストールされます。API とその拡張機能は、$ORACLE_HOME/jlib/ldapjclnt10.jar にあります。

oracle.java.util パッケージを使用した LDAP オブジェクトのモデル化

Java では、LDAP エンティティ（ユーザー、グループ、レルム、アプリケーション）は、ハンドルではなく Java オブジェクトとしてモデル化されます。このモデル化は、oracle.java.util パッケージで行われます。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 や PL/SQL のプログラマとは異なり、Java プログラマはオブジェクトを明示的に解放する必要がありません。このタスクは、Java のガベージ・コレクション・メカニズムが実行します。

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、1 つ以上の LDAP エントリの集合を表しています。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、各エントリのオブジェクト・クラス値をすべて出力します。

ユーザーの管理

ユーザー関連機能はすべて oracle.ldap.util.User という Java クラスで抽象化されます。このプロセスは、次のようになります。

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.authenticateUser(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。

ユーザーの認証

ユーザー認証は、ユーザーがログイン時に指定した資格証明とそのユーザーのディレクトリ内の資格証明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 userpassword 属性のみに適用されることに注意してください。

次に、API を使用してユーザーを認証するためのコード例を示します。

```
 // 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
    }
```

ユーザーの作成

subscriber クラスによって createUser() メソッドが使用され、プログラムにより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 just defined
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・オブジェクトの取得

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 directory 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 user's nickname
starts with "Ani"
User[] userArr = sub.getUsers(ctx, Util.IDTYPE_SIMPLE,
"Ani", null);
// Do work with the User array
```

レルムからのオブジェクトの取得

この項では、Java API を使用して ID 管理レルムのオブジェクトを取得する方法について説明します。

RootOracleContext クラスはルート Oracle コンテキストを表します。ID 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 ID 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 contains a valid directory
// 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
```

例 : OracleAS Single Sign-On のログイン名の検索

次の例では、単純な名前、GUID または DN がわかっている場合にユーザーのログイン名を検索する方法を示します。Oracle Application Server Single Sign-On のログイン名は、ニックネームとも呼ばれます。

この例は、2 つの部分から構成されています。

1. このレルムへのニックネームの保存に使用されている属性を確認します。
2. ユーザー・オブジェクトを取得し、ニックネーム属性の値を確認します。

```
import javax.naming.*;
import javax.naming.directory.*;
import javax.naming.ldap.*;
import oracle.ldap.util.jndi.*;
import oracle.ldap.util.*;
import java.io.*;

public class NickNameSearch {

   public static void main(String[] args)
      throws Exception
   {
      InitialLdapContext ctx = ConnectionUtil.getDefaultDirCtx( args[0],
                                    args[1], args[2],args[3]);

      RootOracleContext roc=new RootOracleContext(ctx);
      Subscriber sub = null;
```

```
      sub = roc.getSubscriber(ctx, Util.IDTYPE_DEFAULT, null, null) ;

      PropertySetCollection psc = sub.getProperties(ctx,
                                    Subscriber.USER_NAMING_PROPERTIES, null);

      String nickNameAttribute = null;
      try
      {
          nickNameAttribute = (String)
psc.getPropertySet(0).getProperty(Subscriber.USER_NAMING_ATTR_SIMPLE).getValue(0);
      }
      catch (Exception e)
      {
         // unable to retrieve the attribute name
         System.exit(0);
      }
      System.out.println("Nickname attribute: " + nickNameAttribute);

      // Retrieve user using simple name, guid or DN
      User user = sub.getUser(ctx, Util.IDTYPE_SIMPLE,"orcladmin", null);
      System.out.println("user DN: " + user.getDN(ctx));

      // Retrieve nickname value using User object
      psc = user.getProperties(ctx, new String[]{ nickNameAttribute });

      String nickName = null;
      try
      {
          nickName = (String)
psc.getPropertySet(0).getProperty(nickNameAttribute).getValue(0);
      }
      catch (Exception e)
      {
         // unable to retrieve the attribute value
         System.exit(0);
      }
      System.out.println("Nickname : " + nickName);
   }
}
```

ディレクトリ・サーバーの検出

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 5-1 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・サーバーの検出

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+"'");
```

```
    }
   }
  }
 }
```

Digest-MD5 を使用した SASL 認証の実行

JNDI を使用して SASL 接続を確立する場合、javax.naming.Context の次のプロパティを設定する必要があります。

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

後者にはプリンシパル名を設定します。この名前は、サーバー固有の形式です。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 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。

例 : SASL Digest-MD5 の auth-int モードおよび auth-conf モードの使用

次のコードでは、SASL Digest-MD5 を使用した Java LDAP/JNDI の例を示します。

```
/* $Header: LdapSasl.java 27-oct-2005.11:26:59 qdinh Exp $ */

/* Copyright (c) 2003, 2005, Oracle. 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, and so on.>

  MODIFIED    (MM/DD/YY)
   qdinh       04/23/03 - Creation
*/

/**
*  @version $Header: LdapSasl.java 27-oct-2005.11:26:59 qdinh Exp $
*  @author  qdinh   *  @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
   {

    int numofargs;

    numofargs = args.length;

    Hashtable hashtable = new Hashtable();

    // Look through System Properties for Context Factory if it is available
    // then 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");
    // possible valid arguments
    // args[0] - hostname
    // args[1] - port number
    // args[2] - Entry DN
    // args[3] - Entry Password
    // args[4] - QoP [ auth | auth-int | auth-conf ]
    // args[5] - SASL Realm
    // args[6] - Cipher Choice
    //    If QoP == "auth-conf" then args[6] cipher choice can be
    //     - des
    //     - 3des
    //     - rc4
    //     - rc4-56
    //     - rc4-40

    hashtable.put(Context.PROVIDER_URL, "ldap://"+args[0]+":"+args[1]);
        hashtable.put(Context.SECURITY_AUTHENTICATION, "DIGEST-MD5");
    System.out.println("hash put security dn: " + args[2]);
    hashtable.put(Context.SECURITY_PRINCIPAL, args[2] );
    hashtable.put(Context.SECURITY_CREDENTIALS, args[3] );

    // For Quality of Protection modes
    // 1. Authentication and Data Integrity Mode - "auth-int"
    // 2. Authentication and Data Confidentiality Mode  "auth-conf"

    //
    // hashtable.put("javax.security.sasl.qop",args[4]);
    hashtable.put("javax.naming.security.sasl.realm", args[5]);

    // Setup Quality of Protection
    //
    // System.out.println("hash sasl.qop: " + args[4]);

    hashtable.put("javax.security.sasl.qop",args[4]);


    if (numofargs > 4)
    {
    if (args[4].equalsIgnoreCase("AUTH-CONF"))
    {
```

```
        //  Setup a cipher choice only if QoP == "auth-conf"
        String strength = "high";
        String cipher = new String(args[6]);
                if (cipher.compareToIgnoreCase("rc4-40") == 0)
        strength = "low";
        else if (cipher.compareToIgnoreCase("rc4-56") == 0 ||
             cipher.compareToIgnoreCase("des")== 0 )
        strength = "medium";
        else if (cipher.compareToIgnoreCase("3des") == 0 ||
             cipher.compareToIgnoreCase("rc4") == 0)
        strength = "high";

        // setup cipher choice
        System.out.println("hash sasl.strength:"+strength);
        hashtable.put("javax.security.sasl.strength",strength);
    }

    // set maxbuffer length if necessary
    if (numofargs > 7 && !"".equals(args[6]))
        hashtable.put("javax.security.sasl.maxbuf", args[5].toString());
    }


    // Enable Debug --
    // hashtable.put("com.sun.jndi.ldap.trace.ber", System.err);

    LdapContext ctx = new InitialLdapContext(hashtable,null);


    // At this stage - SASL Digest -MD5 has been successfully

    System.out.println("sasl bind successful");

    // Ldap Search Scope Options
    //
    // - Search base - OBJECT_SCOPE
    // - One Level   - ONELEVEL_SCOPE
    // - Sub Tree    - SUBTREE_SCOPE
    //
        // Doing an LDAP Search
    PropertySetCollection psc =
Util.ldapSearch(ctx,"o=oracle,dc=com","objectclass=*",SearchControls.OBJECT_SCOPE,
                          new String[] {"*"});
      // Print out the serach result
    Util.printResults(psc);

    System.exit(0);

 }                                                          }
```

6

PL/SQL での API 拡張機能の使用

この章では、標準のディレクトリ API に PL/SQL の拡張機能を使用して、ユーザーを管理および認証する方法について説明します。Oracle の拡張機能には、ユーザーを作成する PL/SQL API は含まれないことに注意してください。標準 API に対する Oracle の拡張機能の詳細は、第 17 章を参照してください。

この章では、次の項目について説明します。

- サンプル・コード
- PL/SQL 拡張機能のインストール
- ハンドルを使用したディレクトリ・データへのアクセス
- ユーザーの管理
- ユーザーの認証
- PL/SQL LDAP API の依存性と制限事項

サンプル・コード

サンプル・コードは次の URL で入手できます。

http://www.oracle.com/technology/sample_code/

「Sample Applications-Oracle Application Server」の下の「Oracle Identity Management」リンクを探してください。

PL/SQL 拡張機能のインストール

PL/SQL 拡張機能は、Oracle データベースのインストール時に DBMS_LDAP パッケージとともにインストールされます。スクリプト $ORACLE_HOME/rdbms/admin/catldap.sql を実行する必要があります。

ハンドルを使用したディレクトリ・データへのアクセス

この章で説明する拡張機能のほとんどは、補助的なファンクションです。これらは、ユーザー、グループ、レルム、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します。多くの場合、これらのファンクションは、いずれかのエンティティに対する参照を標準的な API ファンクションに渡す必要があります。そのために、API 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。次の手順は、拡張機能がユーザー・ハンドルを作成する方法を示しています。

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. ユーザーの入力からユーザー・ハンドルを作成します。この入力は、識別名、GUID または単純なシングル・サインオン・ユーザー ID です。
3. LDAP 接続ハンドル、ユーザー・ハンドルまたは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、接続プールに接続を戻します。

ユーザーの管理

次の手順は、DBMS_LDAP_UTL パッケージを使用して、ディレクトリからユーザー・プロパティを取得するハンドルを作成および使用する方法を示しています。

1. DBMS_LDAP_UTL.create_user_handle(user_hd, user_type, user_id) を呼び出して、ユーザーの入力からユーザー・ハンドルを作成します。この入力は、識別名、GUID または単純なシングル・サインオン・ユーザー ID です。
2. DBMS_LDAP_UTL.set_user_handle_properties(user_hd, property_type, property) を呼び出して、レルムをユーザー・ハンドルと関連付けます。
3. DBMS_LDAP_UTL.get_user_properties(ld, user_handle, attrs, ptype, ret_pset_coll) を呼び出して、ユーザー・エントリの属性を結果ハンドルに入れます。
4. DBMS_LDAP_UTL.get_property_names(pset, property_names) と DBMS_LDAP_UTL.get_property_values(pset, property_name, property_values) を呼び出して、手順 3 で取得した結果ハンドルからユーザー属性を抽出します。

ユーザーの認証

ディレクトリに対してユーザーを認証するには、`DBMS_LDAP_UTL.authenticate_user(session, user_handle, auth_type, cred, binary_cred)` を使用します。このファンクションは、ユーザーが指定したパスワードとユーザーのディレクトリ・エントリのパスワード属性を比較します。

PL/SQL 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。

7

プロビジョニング統合アプリケーションの開発

10*g*（10.1.4.0.1）から、プロビジョニング統合アプリケーションの開発に新しい API を使用できます。詳細は、次のドキュメントを参照してください。

- 『Oracle Identity Management 統合ガイド』の「Oracle プロビジョニング・サービスの概要」の章
- 『Oracle Identity Management 統合ガイド』の「プロビジョニング統合アプリケーションの配置」の章

8

Oracle Delegated Administration Services との統合

この章では、アプリケーションと Oracle Delegated Administration Services との統合方法について説明します。この Web ツールを使用すると、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。

この章では、次の項目について説明します。

- Oracle Delegated Administration Services とは
- アプリケーションと Delegated Administration Services の統合
- URL アクセスに使用する Java API

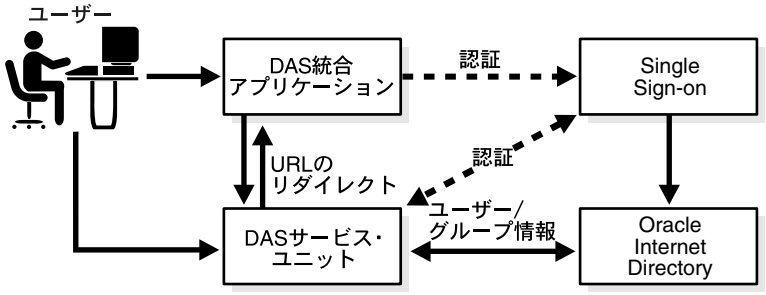
Oracle Delegated Administration Services とは

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・ユニットで構成されます。このユニットにより、ディレクトリ・ユーザーは各自の情報を更新できます。

Delegated Administration Services は、ディレクトリ対応アプリケーションに必要な機能のほとんどを備えています。サービス・ユニットを使用して、ユーザーおよびグループのエントリ作成、エントリの検索、およびユーザー・パスワードの変更が行えます。

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 ポータルを構築する場合は、サービス・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。実行時には、アプリケーションはディレクトリへの問合せを行うことで URL を検出できます。

図 8-1 Delegated Administration Service の概要



アプリケーションに対する Oracle Delegated Administration Services 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よりも優れています。第 1 に、サービス・ユニットは Web ベースであるため、サービス・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リクエスト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。第 2 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。第 3 に、Oracle Delegated Administration Services は Oracle Application Server Single Sign-On と統合されています。アプリケーションは、シングル・サインオン・サーバーで自動的に認証されます。つまり、アプリケーションがユーザーにかわってディレクトリに対する問合せを実行できます。

アプリケーションと Delegated Administration Services の統合

この項では、次の項目について説明します。

- 統合プロファイル
- 統合方法および考慮事項

統合プロファイル

Oracle Delegated Administration Services と統合されたアプリケーションには、次のような特徴があります。

- Web ベースの GUI です。
- mod_osso を介して Oracle Application Server Single Sign-On と統合されています。
- サインオン・ユーザーを介して実行する必要のある操作があります。この操作は、Oracle Delegated Administration Services を使用して実行できます。
- 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 を使用できます。
- Oracle Application Server インフラストラクチャまたは中間層で実行します。そうしないと、サービス URL の検出メカニズムにアクセスできません。

統合方法および考慮事項

8-3 ページの表 8-1 に、アプリケーションと Oracle Delegated Administration Services の統合に必要なタスクを示します。

表 8-1 統合の考慮事項

アプリケーションの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ます。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を処理するための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 Oracle Delegated Administration Services の位置を移動する場合に行います。

利用例 1: ユーザーの作成

この例では、Create User ユニットをカスタム・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 のベース URL を識別します。

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

   この API は、`http://host_name:port/` という形式でベース URL を戻します。

2. 次の文字列を使用して `Create User` ユニットの 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。

   8-4 ページの表 8-2 に示すパラメータをこのユニット用にカスタマイズします。

表 8-2 Oracle Delegated Administration Services の 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した場合、「**Home**」をクリックすると、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。
`cancelURL`	この URL は、Oracle Delegated Administration Services ユニットに表示されるすべての「**Cancel**」ボタンにリンクされます。ユーザーが「Cancel」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「**Assign Privileges**」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/
   AppCreateUserInfoAdmin?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

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 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、データがドメイン間を移動することはできません。この制限により、同じドメイン内のページのみが LOV ユニットにアクセスできます。

ベース URL と相対 URL は、Create User の場合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
```

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

Java API を使用して、Oracle Delegated Administration Services の URL を検出できます。この API の詳細は、第 4 章「標準的な 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 18 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。URL 検出に対応した API ファンクションは、`getDASUrl(DirContext ctx, String urlTypeDN)` および `getAllDASUrl(DirContext ctx)` です。

9

シングル・サインオン対応のアプリケーションの開発

この章では、mod_osso で動作するアプリケーションの開発方法を説明します。この章の項目は次のとおりです。

- mod_osso とは
- mod_osso を使用したアプリケーションの保護 : 2 つの方法
- mod_osso を使用したアプリケーションの開発
- セキュリティに関する問題
- 強制認証

mod_osso とは

OracleAS リリース 2（10.1.2）では、Oracle HTTP Server の認証モジュールである mod_osso を使用して、アプリケーションをシングル・サインオン対応にします。mod_osso は、以前のリリースでパートナ・アプリケーションとの統合に使用されていた Single Sign-On SDK にかわる簡単な方法です。mod_osso は、Single Sign-On Server の単独のパートナ・アプリケーションとして機能することで、認証プロセスを単純化します。これにより、OracleAS アプリケーションでの透過的な認証が実現します。

ユーザーを認証した後、mod_osso はアプリケーションでのユーザー検証に必要とされる単純なヘッダー値を転送します。転送される値は次のとおりです。

- ユーザー名
- ユーザーの GUID
- 言語と地域

表 9-1 は、mod_osso からアプリケーションに渡されるすべてのユーザー属性の一覧です。この表では、鍵やハンドルとして使用したり、Oracle Internet Directory から他のユーザー属性を取得したりする場合に、それぞれの属性が推奨されるかどうかも示しています。

表 9-1 パートナ・アプリケーションに渡されるユーザー属性

HTTP ヘッダー名	説明	ソース	鍵またはハンドルとしての使用
Osso-User-Guid	シングル・サインオン・ユーザーのグローバルに一意なユーザー ID（GUID）	シングル・サインオン・ユーザーのグローバルに一意なユーザー ID（GUID）	推奨。
Osso-Subscriber-Guid	レルムの GUID	Oracle Internet Directory のレルム・エントリ	推奨。
Remote-User	ログイン・ページでユーザーが入力したユーザー・ニックネーム	シングル・サインオン・ログイン・ページ	9.0.4 より前のアプリケーションの場合にのみ推奨。
Osso-Subscriber	レルムのわかりやすい名前	Oracle Internet Directory のレルム・エントリ	非推奨。Oracle Internet Directory でのユーザー検索には GUID ヘッダーを使用してください。
Accept-Language	ISO 形式の言語と地域	Single Sign-On Server	非適用。

mod_osso と同時に使用できるのは Oracle HTTP リスナーのみです。Sun One や IIS などのサード・パーティ・リスナーと連携するアプリケーションの保護には、OracleAS SSO Plug-in を使用します。OracleAS SSO Plug-in の使用方法については、『Oracle HTTP Server 管理者ガイド』の OracleAS SSO Plug-in に関する付録を参照してください。

mod_osso を使用したアプリケーションの保護 : 2 つの方法

mod_osso は、リクエストされた URL が保護されるように構成されている場合にかぎり、ユーザーを Single Sign-On Server にリダイレクトします。URL は、静的な方法または動的な方法で保護できます。スタティック・ディレクティブは、単純にユーザーの対話に対する制御を mod_osso に渡すことで、アプリケーションを保護します。ダイナミック・ディレクティブは、アプリケーションを保護するだけでなく、アプリケーションにおけるユーザー・アクセスの調整も可能にします。

この項の項目は次のとおりです。

- URL の静的な保護
- ダイナミック・ディレクティブを使用した URL の保護

URL の静的な保護

mod_osso.conf ファイルにディレクティブを適用すると、mod_osso を使用して URL を静的に保護できます。このファイルは、$ORACLE_HOME/Apache/Apache/conf にあります。次の例では、Oracle HTTP Server のドキュメント・ルートの真下にある /private ディレクトリを、このディレクティブによって保護する方法を示します。

```
<IfModule mod_osso.c>

  <Location /private>
    AuthType Basic
    require valid-user
  </Location>

</IfModule>
```

エントリを作成した後、Oracle HTTP Server を再起動します。

```
$ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
```

最後に、ディレクトリにページを追加し、それらをテストします。次に例を示します。

```
http://host:port/private/helloworld.html
```

ダイナミック・ディレクティブを使用した URL の保護

ダイナミック・ディレクティブは、特殊なエラー・コードを持つ HTTP レスポンス・ヘッダーで、これを使用すると、複雑なシングル・サインオン・プロトコルを実装しなくても、アプリケーションでシングル・サインオン・システムの細かい機能を利用できます。mod_osso は、単純な HTTP レスポンスの一部としてアプリケーションからディレクティブを受信すると、適切なシングル・サインオン・プロトコル・メッセージを作成して、Single Sign-On Server に送信します。

OracleAS でサポートされているダイナミック・ディレクティブは、Java サーブレットと JSP 用です。現在のところ、PL/SQL アプリケーションにはダイナミック・ディレクティブを使用できません。

表 9-2 は、一般的にリクエストされるダイナミック・ディレクティブの一覧です。

表 9-2 一般的にリクエストされるダイナミック・ディレクティブ

ディレクティブ	ステータス・コード	ヘッダー
認証リクエスト	401、499	-
強制認証リクエスト	499	Osso-Paranoid: true
シングル・サインオフ	470	Osso-Return-URL これは、シングル・サインオフの完了後に返す URL です。

mod_osso を使用したアプリケーションの開発

この項では、mod_osso を使用したアプリケーションの作成と有効化の方法について説明します。この項の項目は次のとおりです。

- 静的に保護された PL/SQL アプリケーションの開発
- 静的に保護された Java アプリケーションの開発
- ダイナミック・ディレクティブを使用する Java アプリケーションの開発
- 非 GET 認証について
- グローバルな非アクティブ・タイムアウトおよびダイナミック・ディレクティブ

静的に保護された PL/SQL アプリケーションの開発

次は、mod_osso で保護された単純なアプリケーションの例です。このアプリケーションは、Single Sign-On Server にユーザーがログインし、ユーザー情報を表示し、ユーザーがアプリケーションと Single Sign-On Server の両方からログアウトするようにします。

mod_osso を使用して PL/SQL アプリケーションを作成および有効化する手順は、次のとおりです。

1. アプリケーション・プロシージャがロードされるスキーマを作成します。

```
sqlplus sys/sys_password as sysdba
create user schema_name identified by schema_password;
grant connect,  resource to schema_name;
```

2. 次のプロシージャをスキーマにロードし、プロシージャにパブリック・アクセス権を付与します。

```
create or replace procedure show_user_info
 is
 begin
    begin
        htp.init;
     exception
         when others then null;
     end;
     htp.htmlOpen;
     htp.bodyOpen;
     htp.print('<h2>Welcome to Oracle Single Sign-On</h2>');
     htp.print('<pre>');
     htp.print('Remote user:'
        || owa_util.get_cgi_env('REMOTE_USER'));
     htp.print('User DN:'
        || owa_util.get_cgi_env('Osso-User-Dn'));
     htp.print('User Guid:'
        || owa_util.get_cgi_env('Osso-User-Guid'));
     htp.print('Subscriber:'
        || owa_util.get_cgi_env('Osso-Subscriber'));
     htp.print('Subscriber DN:'
        || owa_util.get_cgi_env('Osso-Subscriber-Dn'));
     htp.print('Subscriber Guid:'
        || owa_util.get_cgi_env('Osso-Subscriber-Guid'));
     htp.print('</pre>');
     htp.print('<a href=/osso_logout?'
        ||'p_done_url=http://my.oracle.com>Logout</a>');

     htp.bodyClose;
     htp.htmlClose;
end show_user_info;
/
show errors;
```

```
grant execute on show_user_info to public;
```

3. アプリケーションに対するデータベース・アクセス記述子（DAD）を、$ORACLE_HOME/Apache/modplsql/conf にある dads.conf ファイルに作成します。

```
<Location /pls/DAD_name>
       SetHandler pls_handler
       Order deny,allow
       AllowOverride None
       PlsqlDatabaseConnectString     hostname:port:SID
       PlsqlDatabasePassword          schema_password
       PlsqlDatabaseUsername          schema_name
       PlsqlDefaultPage               schema_name.show_user_info
       PlsqlDocumentTablename         schema_name.wwdoc_document
       PlsqlDocumentPath              docs
       PlsqlDocumentProcedure         schema_name.wwdoc_process.process_
                                      download
       PlsqlAuthenticationMode        Basic
       PlsqlPathAlias                 url
       PlsqlPathAliasProcedure        schema_name.wwpth_api_alias.process_
                                      download
       PlsqlSessionCookieName         schema_name
       PlsqlCGIEnvironmentList        OSSO-USER-DN
       PlsqlCGIEnvironmentList        OSSO-USER-GUID
       PlsqlCGIEnvironmentList        OSSO-SUBSCRIBER
       PlsqlCGIEnvironmentList        OSSO-SUBSCRIBER-DN
       PlsqlCGIEnvironmentList        OSSO-SUBSCRIBER-GUID
</Location>
```

4. 次の行を mod_osso.conf ファイルに入力して、アプリケーション DAD を保護します。

```
<Location /pls/DAD_name>
  require valid-user
  authType Basic
</Location>
```

> **注意：** ここでは、mod_osso がシングル・サインオン用にすでに構成されていると想定しています。この手順は、OracleAS のインストール時に実行されます。

5. Oracle HTTP Server を再起動します。

```
http://host:port/private/helloworld.html
```

6. 新しく作成されたファンクションとプロシージャが mod_osso によって保護されているかどうかをテストするには、次のようにブラウザからアクセスします。

```
http://host:port/pls/DAD/schema_name.show_user_info
```

mod_osso.conf が適切に構成されていて、mod_osso が Single Sign-On Server に登録されていれば、URL を選択したときにシングル・サインオン・ログイン・ページが表示されます。

静的に保護された Java アプリケーションの開発

mod_osso を使用してサーブレットや JSP アプリケーションを作成および有効化する手順は、次のとおりです。

1. JSP またはサーブレットを作成します。前述の PL/SQL アプリケーションの例と同様に、次の単純なサーブレットはユーザーのログイン、ユーザー情報の表示およびユーザーのログアウトを行います。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Example servlet showing how to get the SSO User information
 */

public class SSOProtected extends HttpServlet
{

    public void service(HttpServletRequest request,
         HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");

        // Show authenticated user informationsingle sign-on
        PrintWriter out = response.getWriter();
        out.println("<h2>Welcome to Oracle Single Sign-On</h2>");
        out.println("<pre>");
        out.println("Remote user: "
            + request.getRemoteUser());
        out.println("Osso-User-Dn: "
            +  request.getHeader("Osso-User-Dn"));
        out.println("Osso-User-Guid: "
            +  request.getHeader("Osso-User-Guid"));
        out.println("Osso-Subscriber: "
            +  request.getHeader("Osso-Subscriber"));
        out.println("Osso-User-Dn: "
            +  request.getHeader("Osso-User-Dn"));
        out.println("Osso-Subscriber-Dn: "
            +  request.getHeader("Osso-Subscriber-Dn"));
        out.println("Osso-Subscriber-Guid: "
            +  request.getHeader("Osso-Subscriber-Guid"));
        out.println("Lang/Territory: "
            + request.getHeader("Accept-Language"));
        out.println("</pre>");
        out.println("<a href=/osso_logout?"
            +"p_done_url=http://my.oracle.com>Logout</a>");
```

2. mod_osso.conf ファイルに次の行を入力して、サーブレットを保護します。

```
<Location /servlet>
   require valid-user
   authType Basic
</Location>
```

3. サーブレットを配置します。詳細は、『Oracle Containers for J2EE サーブレット開発者ガイド』の概要の章を参照してください。この章では、サーブレットの例とその配置方法を説明しています。

4. Oracle HTTP Server と OC4J を再起動します。

```
$ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
$ORACLE_HOME/opmn/bin/opmnctl stopproc type=oc4j
$ORACLE_HOME/opmn/bin/opmnctl startproc type=oc4j
```

5. ブラウザからサーブレットにアクセスを試みて、サーブレットをテストします。URL を選択すると、ログイン・ページが表示されます。

   このプロセスは、次のようになります。まず、ブラウザからサーブレットにアクセスしようとすると、認証を行うために Single Sign-On Server にリダイレクトされます。その後、もう一度サーブレットにリダイレクトされ、ユーザー情報が表示されます。ログアウト・リンクを選択すると、アプリケーションと Single Sign-On Server からログアウトできます。

ダイナミック・ディレクティブを使用する Java アプリケーションの開発

mod_osso による保護は、ダイナミック・ディレクティブとしてアプリケーションに直接書き込まれるため、ダイナミック・ディレクティブを使用するアプリケーションに mod_osso.conf ファイルのエントリは必要ありません。以下のサーブレットは、そのようなディレクティブがどのように組み込まれるかを示しています。静的なアプリケーションと同様に、これらのサンプルの動的なアプリケーションは、ユーザー情報を生成します。

この項の項目は次のとおりです。

- Java の例 1: 簡易認証
- Java の例 2: シングル・サインオフ

Java の例 1: 簡易認証

このサーブレットは、request.getRemoteUser() メソッドを使用して、mod_osso Cookie でユーザー名をチェックしています。ユーザー名がない場合、サーブレットはダイナミック・ディレクティブ 499 という簡易認証リクエストを発行します。重要な行は、太字で示しています。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Example servlet showing how to use
 * Dynamic Directive for login
 */

public class SSODynLogin extends HttpServlet
{

    public void service(HttpServletRequest request,
         HttpServletResponse response)
        throws IOException, ServletException
    {
        String l_user     = null;

        // Try to get the authenticate user name
        try
        {
            l_user = request.getRemoteUser();
        }
        catch(Exception e)
        {
            l_user = null;
        }

        // If user is not authenticated then generate
        // dynamic directive for authentication
        if((l_user == null) || (l_user.length() <= 0) )
        {
           response.sendError(499, "Oracle SSO");
        }
```

```
        else
        {
            // Show authenticated user information
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<h2>Welcome to Oracle Single Sign-On</h2>");
            out.println("<pre>");
            out.println("Remote user: "
                + request.getRemoteUser());
            out.println("Osso-User-Dn: "
                +  request.getHeader("Osso-User-Dn"));
            out.println("Osso-User-Guid: "
                +  request.getHeader("Osso-User-Guid"));
            out.println("Osso-Subscriber: "
                +  request.getHeader("Osso-Subscriber"));
            out.println("Osso-User-Dn: "
                +  request.getHeader("Osso-User-Dn"));
            out.println("Osso-Subscriber-Dn: "
                +  request.getHeader("Osso-Subscriber-Dn"));
            out.println("Osso-Subscriber-Guid: "
                +  request.getHeader("Osso-Subscriber-Guid"));
            out.println("Lang/Territory: "
                + request.getHeader("Accept-Language"));
            out.println("</pre>");
        }
    }
```

注意： Oracle JAAS Provider を使用している場合、`499` のかわりにディレクティブ・コード `401` を使用できます。

Java の例 2: シングル・サインオフ

このサーブレットは、ユーザーがアプリケーションにあるログイン・リンクを選択した際に起動します。アプリケーションはサインオフ完了後に戻る URL を設定します。そして、ユーザーがシングル・サインオフ・ページに移動するためのディレクティブを発行します。重要な行は、太字で示しています。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Example servlet showing how to use
 * Dynamic Directive for logout
 */

public class SSODynLogout extends HttpServlet
{
    public void service (HttpServletRequest request,
         HttpServletResponse response)
        throws ServletException, IOException
    {
        // Set the return URL
        response.setHeader("Osso-Return-Url",
               "http://my.oracle.com" );
        // Send Dynamic Directive for logout
        response.sendError(470, "Oracle SSO");
    }
}
```

注意： 別の方法として、同じコンピュータの osso_logout URL にリダイレクトすることもできます。

非 GET 認証について

mod_osso で保護されたアプリケーションの 1 ページ目は、GET 認証方式を使用する URL であることが必要です。POST メソッドを使用すると、Single Sign-On Server にリダイレクトしている間に、ユーザーがログイン時に入力したデータが失われます。グローバル・ユーザーの非アクティブ・タイムアウトを有効にするかどうかを決定する際は、ユーザーのリダイレクトが、タイムアウト後に再びログインしてから行われることに注意してください。

グローバルな非アクティブ・タイムアウトおよびダイナミック・ディレクティブ

アプリケーションでシングル・サインオンを有効にするためにグローバルな非アクティブ・タイムアウトおよびダイナミック・ディレクティブを使用している場合、アプリケーションの Osso-Idle-Timeout-Exceeded HTTP ヘッダーを使用してタイムアウト・ステータスを判別できます。このヘッダーの値は、タイムアウトが発生した場合は true に、発生していない場合は false に設定されます。

次の例は、Osso-Idle-Timeout-Exceeded HTTP ヘッダーの使用方法を示しています。

```
// Get the timeout status
String timeoutStatus = request.getHeader("Osso-Idle-Timeout-Exceeded")
// Check if user has timedout
if ( (timeoutStatus != null) && timeoutStatus.equalsIgnoreCase("true") )
{
   response.setHeader( "Osso-Paranoid", "true" );
   response.sendError(499, "Oracle SSO");
}
else
{
  // Display page content here
}
```

セキュリティに関する問題

この項では、OracleAS Single Sign-On 対応のアプリケーションを開発する際のセキュリティ上の考慮事項について説明します。次の項目について説明します。

- シングル・サインオフとアプリケーションのログアウト
- mod_osso Cookie のセキュアな送信

シングル・サインオフとアプリケーションのログアウト

OracleAS を使用してカスタム・アプリケーションを構築する場合は、グローバル・ログアウトまたはシングル・サインオフの実行時に、シングル・サインオン Cookie と mod_osso Cookie のみがクリアされることに注意してください。つまり、OracleAS アプリケーションをコーディングする際は、シングル・サインオン・ユーザー名とレルム名が OC4J セッションまたはアプリケーション・セッションに保存されるようにする必要があります。アプリケーションではその後、これらの値と、mod_osso から渡された値とを比較し、値が一致した場合は、パーソナライズされたコンテンツを表示する必要があります。値が一致しない場合、すなわち mod_osso Cookie が存在しない場合は、アプリケーション・セッションをクリアし、ユーザーにログインを強制する必要があります。

この項の項目は次のとおりです。

- アプリケーションのログイン : コード例
- アプリケーションのログアウト : 推奨コード

アプリケーションのログイン : コード例

最初の 2 つのコード例には、前述の項で説明したロジックが使用されていません。このロジックは、3 つ目の例で使用されています。これらは Java の例ですが、Perl、PL/SQL、CGI などの言語で記述することもできます。

不適切なコード例 1

```
// Get user name from application session. This session was
// established by the application cookie or OC4J session cookie
String username = request.getSession().getAttribute('USER_NAME');

// Get subscriber name from application session. This session was
// established by the application cookie or OC4J session cookie.
String subscriber = request.getSession().getAttribute('SUBSCRIBER_NAME');

// Get user security information from application session. This session was established
by the application cookie or OC4J session cookie
String user_sec_info = request.getSession().getAttribute('USER_APP_SEC');

if((username != null) && (subscriber!= null))
{
  // Show personalized user content
  show_personalized_page(username, subscriber, user_sec_info);
}
else
{
  // Send Dynamic Directive for login
  response.sendError( 499, "Oracle SSO" );
```

不適切なコード例 2

```
// Get SSO username from http header
String username = request.getRemoteUser();

// Get subscriber name from SSO http header
String subscriber = request.getHeader('OSSO-SUBSCRIBER');

// Get user security information from application session.
// This session was established by the application or OC4J session.
String user_sec_info =request.getSession().getAttribute('USER_APP_SEC');

if((ssousername != null)&&(subscriber!= null))
{
  // Show personalized user content
show_personalized_page(username, subscriber, user_sec_info);
}
```

```
else
{
  // Send Dynamic Directive for login
response.sendError( 499, "Oracle SSO" );
}
```

推奨コード

```
// Get user name from application session. This session was
// established by the application or OC4J session
String username = request.getSession().getAttribute('USER_NAME');

// Get subscriber name from application session. This session was
// established by the application or OC4J session
String subscriber = request.getSession().getAttribute('SUBSCRIBER_NAME');

// Get user security information from application session.
 // This session was established by the application or OC4J session.
String user_sec_info = request.getSession().getAttribute('USER_APP_SEC');

// Get username and subscriber name from JAZN API */
JAZNUserAdaptor jaznuser  = (JAZNUserAdaptor)requset.getUserPrincipal();
     String ssousername   = jaznuser.getName();
     String ssosubscriber = jaznuser.getRealm().getName();

// If you are not using JAZN api then you can also get the username and
// subscriber name from mod_osso headers
String ssousername   = request.getRemoteUser();
String ssosubscriber = request.getHeader('OSSO-SUBSCRIBER');

// Check for application session. Create it if necessary.
if((username == null) || (subscriber == null) )
  {
   ...Code to create application session. Get the user information from
   JAZN api (or mod_osso headers if you are not using JAZN api) and populate the
   application session with user, subscriber, and user security info.
  }

if((username != null)&&(subscriber != null)
  &&(ssousername != null)&&(ssosubscriber != null)
  &&(username.equalsIgnoreCase(ssousername) == 0 )
  &&(subscriber.equalsIgnoreCase(ssosubscriber) == 0))
{
  // Show personalized user content
show_personalized_page(username, subscriber, user_sec_info);
}
else
{
     ...Code to Wipe-out application session, followed by...

// Send Dynamic Directive for login
// If you are using JAZN then you should use following code
// response.sendError( 401);

// If you are not using JAZN api then you should use following code
// response.sendError( 499, "Oracle SSO" );
}
```

アプリケーションのログアウト : 推奨コード

ユーザーを認証するアプリケーションのほとんどには、ログアウト・リンクがあります。シングル・サインオン対応のアプリケーションでは、ログアウト・ハンドラ内の他のコードに加えて、ログアウト用のダイナミック・ディレクティブが起動します。ユーザーがログアウト・ディレクティブを起動すると、シングル・サインオフまたはグローバル・ログアウトが実行されます。次の例では、Java でシングル・サインオフを記述した場合のコードを示しています。

```
// Clear application session, if any
String l_return_url := return url to your application
response.setHeader( "Osso-Return-Url", l_return_url);
response.sendError( 470, "Oracle SSO" );
```

mod_osso Cookie のセキュアな送信

OssoSecureCookies ディレクティブを追加して、mod_osso により作成されるすべての Cookie に Secure フラグを設定できます。これにより、ブラウザでは、HTTPS で接続が保護されている場合にのみこれらの Cookie が送信されます。

$ORACLE_HOME/Apache/Apache/conf/mod_osso.conf にある mod_osso 構成ファイルでのこのディレクティブの使用例は、次のとおりです。

```
<IfModule mod_osso.c>
    OssoIpCheck off
    OssoIdleTimeout off
    OssoSecureCookies on
    OssoConfigFile osso/osso.conf

    <Location /j2ee/webapp>
        require valid-user
        AuthType Basic
    </Location>

</IfModule>
```

強制認証

Oracle Application Server Single Sign-On によって保護されているアプリケーションには、アプリケーションの実行時にユーザーを強制的に再認証するオプションがあります。強制認証プロセスでは、すでに有効なシングル・サインオン・セッションがある場合でも、ユーザーは Oracle Single Sign-On にログインする必要があります。これは、オンラインでの金銭の転送など、高レベルのセキュリティが必要な場合に適しています。強制認証には、Oracle HTTP Server リリース 10.1.3.1.0 以降および Oracle Application Server Single Sign-On リリース 9.0.4 以降が必要です。

この機能を使用するには、強制認証プロセスが成功したことを確認するために、保護されているアプリケーションでいくつかのユーザー・セッションの状態を記録しておく必要があります。これらのアプリケーションでは、Osso-Cookie-Timestamp リクエスト・ヘッダー値（time1）、およびユーザーに認証を強制する直前の現在の時間（time2）を記録する必要があります。ユーザーは、再認証後に再度アプリケーションにアクセスします。この時点で、アプリケーションにより、現在の Osso-Cookie-Timestamp リクエスト・ヘッダー値（time3）が time1 および time2 と比較されます。アプリケーションでは time3 が、time1 と time2 のどちらよりも遅いことを確認する必要があります。遅くない場合には、アプリケーションによりそのユーザー・セッションが拒否され、ユーザーはセキュリティに影響を与えるいかなる操作も実行できません。

Osso-Cookie-Timestamp の値は文字列で、OracleAS Single Sign-On セッションが開始されたカレンダ時間の 16 進エンコーディングを表します。この時間値は、(Epoch とも呼ばれる) 1970 年 1 月 1 日の 00:00:00 から経過した秒数を表します。次の手順でプロセスの概要を示します。

1. 既存の有効な OracleAS Single Sign-On セッションから Osso-Cookie-Timestamp 値を取得します。この値を time1 として記録します。
2. 現在の時間を取得します。この値を time2 として記録します。
3. Osso-Paranoid リクエスト・ヘッダーを true に設定して強制認証をトリガーし、Oracle HTTP Server に HTTP ステータス 499 を戻します。
4. Oracle HTTP Server によりユーザーが OracleAS Single Sign-On サーバーにリダイレクトされ、再認証が要求されます。
5. ユーザーが保護されているアプリケーションにアクセスすると、新しい Osso-Cookie-Timestamp 値が取得されます。これが time3 です。
6. アプリケーションにより、time3 が time1 および time2 のどちらよりも遅いことが確認されます。遅くない場合、ユーザー・ログイン・セッションはアプリケーションにより拒否されます。

次に、強制認証のサンプル・コードを示します。

```
//About to execute sensitive security operation.
//user should have already been forced to login.
//verify timestamps
if(!checkForcedAuthSuccess(l_session))
{
   //forced authentication was unsuccessful
   destroyUserSession();
}
else
{
   //successful forced authentication
}
public boolean checkForcedAuthSuccess(HttpSession session
{
   try
   {
   SessionStateObject state = session.getAttribute("SESSION_STATE")

   //get the current cookie timestamp (time3)
   l_currTimestampStr = (String) request.getHeader("Osso-Cookie-Timestamp");

   //convert hex to decimal & get date
   l_decValue = convertHexToDecimal(l_currTimestampStr);
   l_decValue *= 1000;
   l_currTimestampDate = new Date(decValue);

   //time when user was forced to authenticate (time2)
   l_forcedCheckDate = state.getForcedCheckTime();
   //previous mod_osso cookie timestamp (time1)
   l_previousAuthDate = state.getPreviousAuthTimestamp();
   // current auth timestamp needs to be AFTER prevAuthDate
   // current auth timestamp needs to be AFTER forcedCheckDate
   if((l_currTimestampDate.after(l_previousAuthDate)) &&
      (l_currTimestampDate.after(l_forcedCheckDate)))
   {
      l_ret = true;
   }
   else
   {
      l_ret = false;
   }
```

```
}
catch(Exception ex)
   {
   throw new RuntimeException("Unable to check forcedAuth status.", ex);
   }
return l_ret;
}
```

関連資料：『Oracle Application Server Single Sign-On 管理者ガイド』

10

J2EE アプリケーションと Oracle Internet Directory の統合

この章では、Oracle Internet Directory からユーザー権限、グループおよびポリシーに関する情報を取得するために J2EE アプリケーションで使用できる API の概要について説明します。

Oracle Containers for J2EE（OC4J）は、J2EE 認定のサーバー実装です。OC4J では、標準の J2EE セキュリティ API がサポートされます。

標準のセキュリティ API に加え、OC4J では、JAZN と総称される一連のセキュリティ機能が提供されます。JAZN には、Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider、JAZN ユーザー・マネージャ、JAAS ポリシー管理 API およびレルム API が含まれます。OC4J は、Oracle Application Server Single Sign-On および Oracle Internet Directory と完全に統合されています。JAZN のセキュリティ API により、標準の J2EE セキュリティ API には存在しない機能が提供されます。

OracleAS JAAS Provider は、XML ファイルまたは Oracle Internet Directory にセキュリティ・ポリシーを格納する Java Authentication and Authorization Services（JAAS）の実装です。OC4J アプリケーションでは、厳密な認可のために JAAS ポリシー管理 API を使用できます。

この章の項目は次のとおりです。

- 標準の J2EE セキュリティ API
- OC4J セキュリティ API
- JAAS ポリシー管理 API

標準の J2EE セキュリティ API

J2EE の標準実装には、ユーザーおよびロールに関する情報を取得するために Java サーブレットと Enterprise JavaBeans（EJB）で使用できるセキュリティ API が含まれます。これらの API は、Oracle Internet Directory とは独立して動作します。これらの API により、アプリケーションが Oracle Identity Management と統合されているかどうかにかかわらず、認証済のユーザーに関する情報を取得できます。

Java サーブレット仕様の一部である `javax.servlet.http` パッケージには、ユーザー情報を取得するための次のメソッドが含まれます。

- `javax.servlet.http.HttpServletRequest.getUserPrincipal()`
- `javax.servlet.http.HttpServletRequest.isUserInRole()`
- `javax.servlet.http.HttpServletRequest.getRemoteUser()`

`javax.servlet.http` パッケージの詳細は、次の URL を参照してください。

`http://java.sun.com/products/servlet/2.2/javadoc/index.html`

同様に、Enterprise JavaBeans 仕様の一部である `javax.ejb` パッケージには、ユーザー情報を取得するための次のメソッドが含まれます。

- `javax.ejb.EJBContext.getCallerPrincipal()`
- `javax.ejb.EJBContext.isCallerInRole()`

`javax.ejb` パッケージの詳細は、次の URL を参照してください。

`http://java.sun.com/j2ee/1.4/docs/api/javax/ejb/package-tree.html`

OC4J セキュリティ API

JAZN のセキュリティ API は、`com.evermind.security` パッケージに基づいています。このクラスは、J2EE アプリケーションにアクセスを試みるユーザーおよびグループの認証と認可を行うユーザー・マネージャを指定します。デフォルトの JAZN ユーザー・マネージャは、`JAZNUserManager` です。このユーザー・マネージャは、LDAP ベース・プロバイダに対応しており、Oracle Application Server Single Sign-On および Oracle Internet Directory と統合されています。

`JAZNUserManager` を使用して Oracle Internet Directory の情報にアクセスするには、JAZN を構成して LDAP ベース・プロバイダ（`jazn-ldap`）を使用する必要があります。詳細は、『Oracle Containers for J2EE セキュリティ・ガイド』を参照してください。

JAZN では、Oracle Internet Directory からユーザー属性を取得するために、`com.evermind.security.User` の次のメソッドがサポートされます。

- `getDescription()`: このユーザーの短い説明または NULL（説明が存在しない場合）を返します。
- `getGroups()`: このユーザーが属しているグループを返します（グループが既知でサポートされている場合）。
- `getName()`: このユーザーのユーザー名を返します。
- `hasPermission()`: このユーザーが名前付き権限を保持しているかどうかをチェックします。
- `isMemberOf()`: このユーザーが指定グループのメンバーであるかどうかをチェックします。

詳細は、『JAAS Provider API Reference』を参照してください。

電子メール・アドレスや Oracle Internet Directory 固有の属性など、追加のユーザー属性を必要とするアプリケーションでは、Oracle Internet Directory API を使用する必要があります。詳細は、『Oracle Internet Directory API Reference』と、第 2 章および第 5 章の説明を参照してください。

JAZN API では、ユーザー作成はサポートされません。ユーザーを作成するには、Oracle Internet Directory API または Oracle Delegated Administration Services を使用してください。

サンプル・コード

次のサンプル・コードは、認証の実施後にユーザー情報を取得するために使用される標準の J2EE および JAZN API を示しています。

```
package oracle.security.jazn.samples.http;

import java.io.IOException;
import java.util.Date;
import java.util.Properties;
import javax.naming.*;
import javax.servlet.*;
import javax.servlet.http.*;


/**
 * A simple demo that exercises the Servlet security APIs.
 *
 */
public class CallerInfo extends HttpServlet {

    public CallerInfo()
    {
                super();
    }

    public void init(ServletConfig config)
        throws ServletException
    {
                super.init(config);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse
             response)
        throws ServletException, IOException
    {
                ServletOutputStream out = response.getOutputStream();

                response.setContentType("text/html");
                out.println("<HTML><BODY bgcolor='#FFFFFF'>");

                //Standard J2EE APIs
                out.println("request.getRemoteUser = " +
                            request.getRemoteUser() + "<br>");
                out.println("request.isUserInRole('FOO') = " +
                            request.isUserInRole("FOO") + "<br>");
                out.println("request.isUserInRole('ar_manager') = " +
                            request.isUserInRole("ar_manager") + "<br>");
                out.println("request.isUserInRole('ar_developer') = " +
                            request.isUserInRole("ar_developer") + "<br>");
                out.println("request.getUserPrincipal = " +
                            request.getUserPrincipal() + "<br>");



                //JAZN-LDAP APIs
                //Get the User principal from request
                com.evermind.security.User user =
                  (com.evermind.security.User)request.getUserPrincipal();
                //getDescription API Test
                try {
```

```
                java.lang.String s = user.getDescription();
                out.println("<b>getDescription</b> API Result: ["
                            +s+ "]<br>");
            }catch(Throwable e) {
                out.println("<b>getDescription</b> API FAILED: " +
                             e.toString() + "<br>");
            }

            //getGroups API Test
            try {
                java.util.Set s = user.getGroups();
                out.println("<b>getGroups</b> API Result: [" +s+
                            "]<br>");
            }catch(Throwable e) {
                out.println("<b>getGroups</b> API FAILED: " +
                             e.toString() + "<br>");
            }

            //getName API Test
            try {
                java.lang.String s = user.getName();
                out.println("<b>getName</b> API Result: [" +s+
                            "]<br>");
            }catch(Throwable e) {
                out.println("<b>getName</b> API FAILED: " +
                             e.toString() + "<br>");
            }

            //hasPermission API Test
            try {
                com.evermind.server.rmi.RMIPermission p = new
                    com.evermind.server.rmi.RMIPermission("login");
                boolean b = user.hasPermission(p);
                out.println("<b>hasPermission</b> API Result: [" + b
                            + "]<br>");
            }catch(Throwable e) {
                out.println("<b>hasPermission</b> API FAILED: " +
                            e.toString() + "<br>");
            }

            //isMemberOf API Test
            try {
                java.util.Set s = user.getGroups();
                java.util.Iterator itr = s.iterator();
                boolean b = false;
                if(itr.hasNext())
                {
                        b =
            user.isMemberOf((com.evermind.security.Group)itr.next());
                }
                out.println("<b>isMemberOf</b> API Result: [" +b+
                            "]<br>");
            }catch(Throwable e) {
                out.println("<b>isMemberOf</b> API FAILED: " +
                            e.toString() + "<br>");
            }


            out.println("</BODY>");
            out.println("</HTML>");
        }
    }
```

JAAS ポリシー管理 API

OC4J には、高度にスケーラブルな Java Authentication and Authorization Service（JAAS）プロバイダである OracleAS JAAS Provider が含まれます。Oracle Internet Directory と統合された J2EE アプリケーションでは、保護されたリソースに対する厳密なアクセス制御を実現するために、JAAS プロバイダを利用できます。

OracleAS JAAS Provider では、JAAS の権限およびポリシーのリポジトリとして Oracle Internet Directory を使用できます。OracleAS JAAS Provider は、アプリケーション・セキュリティ強化のために Oracle Internet Directory および OracleAS Single Sign-On と統合されています。

この項の項目は次のとおりです。

- JAAS ポリシー管理
- 標準の JAAS API を使用したユーザー・ポリシーおよび権限の取得

JAAS ポリシー管理

権限の付与と取消しは、コマンドラインで JAZN Admintool を使用して実行するか、JAZN API によりプログラム的に実行します。

Admintool の `jazn.jar` は、インフラストラクチャ・インストール環境では `$ORACLE_HOME/j2ee/home` にあります。Admintool を使用する前に、`ORACLE_HOME` および `J2EE_HOME` 環境変数を設定してください。

次のコマンドラインでは、ユーザー `scott` にファイル `foo.txt` の読取り権限が付与されます。レルム名 `scottsRealm` は Oracle Internet Directory で定義されており、ユーザー名 `scott` は Oracle Internet Directory に存在します。

```
java -jar jazn.jar -grantperm scottsRealm -user scott java.io.FilePermission foo.txt,
read
```

ユーザー管理に Admintool を使用する方法の詳細は、『Oracle Containers for J2EE セキュリティ・ガイド』の付録 B「JAZN Admintool の使用」を参照してください。

ユーザーに権限をプログラム的に付与するには、JAZN の API を次のように使用します。

```
//get JAZNConfiguration related info
JAZNConfig jc = JAZNConfig.getJAZNConfig();

//create a Grantee for "scott"
RealmManager realmmgr = jc.getRealmManager();
Realm realm = realmMgr.getRealm("scottsRealm");
UserManager userMgr = realm.getUserManager();
final RealmUser user = userMgr.getUser("scott");

//grant scott file permission
JAZNPolicy policy = jc.getPolicy();

if ( policy != null) {
     Grantee gtee = new Grantee( (Principal) user);
     java.io.FilePermission fileperm = new java.io.FilePermission("foo.txt", "read");
     policy.grant( gtee, fileperm);
}
```

詳細は、『JAAS Provider API Reference』および『Oracle Containers for J2EE セキュリティ・ガイド』を参照してください。

標準の JAAS API を使用したユーザー・ポリシーおよび権限の取得

サーブレットは、doasprivileged または runas モードで実行できます。これにより、サーブレットは、それぞれ Subject.doAsPrivileged または Subject.doAs ブロックで実行されます。サーブレットがこれらのモードのいずれかで実行される場合、標準 API である Policy API または AccessController のいずれかを使用して権限をチェックできます。ポリシーを取得するには、doasprivileged モードを使用するようにサーブレットを構成します。doasprivileged または runas モードを構成する方法の詳細は、『Oracle Containers for J2EE セキュリティ・ガイド』の J2EE 認可の構成に関する項を参照してください。

次のコード例は、ユーザー scott が foo.txt の読取り権限を保持している場合に権限をチェックする方法を示しています。

javax.security.auth.Policy を使用した権限のチェックとリスト表示

この方法では、権限をチェックするだけでなく、ユーザーまたはグループに付与されているすべての権限をリスト表示できます。ユーザーまたはグループに付与されている権限をチェックするのみで、コードベースの権限をチェックする必要がない場合は、こちらの方が高速です。

```
//create Permission
FilePermission perm = new FilePermission("/home/scott/foo.txt","read");
{
javax.security.auth.Policy currPolicy =
    javax.security.auth.Policy.getPolicy();
// Query policy now
System.out.println("Policy permissions for this subject are " +
    currPolicy.getPermissions(Subject.getSubject(acc),null));

//Check Permissions
System.out.println("Policy.impiles permission: "+ perm +" ?  " +
    currPolicy.getPermissions(Subject.getSubject(acc),null).implies(perm));
}
```

AccessController を使用した権限のチェック

セキュリティ・マネージャが有効であるか無効であるかにかかわらず、このコードを使用すると、コードを実行するサブジェクトまたはユーザーに権限があるかどうかをチェックできます。

> **注意：** このコードが runas モードで構成されたサーブレットで実行される場合、コードベースにも権限が必要となる場合があります。

```
 //create Permission
FilePermission perm = new FilePermission("/home/scott/foo.txt","read");
{
  //get current AccessControlContext
  AccessControlContext acc = AccessController.getContext();
   AccessController.checkPermission(perm);
}
```

OracleAS JAAS Provider により提供されるポリシー API の詳細は、『Oracle Containers for J2EE セキュリティ・ガイド』の付録 A「OracleAS JAAS Provider のサンプル」と、『Oracle Containers for J2EE セキュリティ・ガイド』の付録 B「JAZN Admintool の使用」を参照してください。

Oracle Internet Directory Java API の詳細は、『Oracle Internet Directory API Reference』および第 5 章「JNDI に対する Java API 拡張機能の使用」を参照してください。

第 II 部

サーバー・プラグイン

第 II 部では、Oracle Internet Directory のサーバー・プラグインおよびプラグイン・フレームワークについて説明します。第 II 部は、次の章で構成されています。

- 第 11 章「Oracle Internet Directory サーバーのプラグインの開発」
- 第 12 章「PL/SQL サーバー・プラグイン」
- 第 13 章「Java サーバー・プラグイン」

11

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

この章では、Oracle Internet Directory のサーバー・プラグインおよび Oracle Internet Directory のプラグイン・フレームワークの概要について説明します。

この章では、次の項目について説明します。

- サーバー・プラグインとは
- サーバー・プラグインでサポートされている言語
- サーバー・プラグインの前提条件
- サーバー・プラグインの利点
- プラグイン設計のガイドライン
- サーバー・プラグイン・フレームワークとは
- ディレクトリがサポートする LDAP 操作およびタイミング
- プラグインの登録
- Oracle Directory Manager を使用したプラグインの管理

サーバー・プラグインとは

サーバー・プラグインは、Oracle Internet Directory サーバーの機能の拡張に使用できるカスタマイズされたプログラムです。サーバー・プラグインは、PL/SQL パッケージ、Java プログラムまたはパッケージ、共有オブジェクトまたはライブラリ、Windows 上の動的リンク・ライブラリのいずれかです。各プラグインには、Oracle Internet Directory サーバーに構成エントリがあります。構成エントリは、プラグインの起動条件を指定します。次に、プラグインの起動条件を示します。

- `ldapbind` または `ldapmodify` などの LDAP 操作
- pre_bind または post_modify などの LDAP 操作に関連するタイミング

サーバー・プラグインでサポートされている言語

10*g*（10.1.4.0.1）の Oracle Internet Directory では、PL/SQL だけでなく Java のプラグインもサポートされています。この章では、Java および PL/SQL プラグインに共通の情報を説明します。第 12 章では PL/SQL プラグインに固有の情報を、第 13 章では Java プラグインに固有の情報を説明します。

サーバー・プラグインの前提条件

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 Application Server と Oracle Internet Directory の統合

次のいずれかの分野でのプログラミング技術が必要です。

- SQL、PL/SQL およびデータベース RPC
- Java

サーバー・プラグインの利点

次の内容を含み、LDAP 操作はプラグインを使用して様々に拡張できます。

- 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ユーザーを認証できます。
- ユーザー独自のサーバー・モジュールで既存のサーバー・モジュールを置換できます。

ディレクトリ・サーバーは、起動時にユーザーのプラグイン構成とライブラリをロードします。また、各種 LDAP リクエストの処理中に、プラグイン・ファンクションをコールします。

> **関連資料：**『Oracle Internet Directory 管理者ガイド』のパスワード・ポリシー・プラグインに関する章を参照してください。この章には、ユーザー独自のパスワード値検査を実装し、Oracle Internet Directory サーバー内に配置する方法の例が示されています。

プラグイン設計のガイドライン

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 `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。

プラグインは注意して使用してください。関連する LDAP 操作が発生するたびに実行されます。

サーバー・プラグイン・フレームワークとは

プラグイン・フレームワークとは、ユーザーがプラグインを開発、構成および適用する環境です。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 API
  - PL/SQL パッケージ `ODS.LDAP_PLUGIN`
  - Java パッケージ `oracle.ldap.ospf`

どちらの言語でも、サーバーのプラグイン・フレームワークを使用するには、次の一般的な手順に従います。

1. ユーザー定義のプラグイン・プロシージャを、PL/SQL または Java で記述します。
2. プラグイン・モジュールをコンパイルします。
3. コマンドラインまたは Oracle Directory Manager を使用して、構成エントリ・インタフェース経由でプラグイン・モジュールを登録します。

ディレクトリがサポートする LDAP 操作およびタイミング

Oracle Internet Directory サーバーには、プラグインをサポートする次の LDAP 操作があります。

- `ldapadd`
- `ldapbind`
- `ldapcompare`
- `ldapdelete`
- `ldapmoddn`（Java のみ）
- `ldapmodify`
- `ldapsearch`

Oracle Internet Directory は、プラグイン用に次の 4 つの操作タイミングをサポートしています。

- pre
- post
- when
- when_replace

これらのタイミングは、次の 4 つの項で説明されています。

pre 操作サーバー・プラグイン

サーバーは、LDAP 操作を実行する前に、pre 操作プラグイン・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。

pre 操作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 リクエストは続行されます。
- リターン・コードが障害ステータスを示す場合。リクエストは続行されません。

関連する LDAP リクエストに後で障害が発生した場合、ディレクトリは、プラグイン・モジュールのコミット済コードをロールバックしません。

post 操作サーバー・プラグイン

Oracle Internet Directory サーバーは、LDAP 操作を実行した後に、post 操作プラグイン・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、post 操作プラグイン・ファンクションの例です。

post 操作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 リクエストに障害が発生した場合、post 操作プラグインはそのまま実行されます。

when 操作サーバー・プラグイン

ディレクトリは、標準の LDAP 操作の実行中に、when 操作プラグイン・モジュールをコールします。when 操作プラグインは、操作用のサーバー独自のコードの直前に実行されます。このタイプのプラグインの主な目的は、同じ LDAP トランザクション内の既存の操作を補強することです。when 操作プラグインが失敗すると、標準の LDAP 操作が実行されなくなります。when 操作プラグインが正常に完了し、標準の LDAP 操作が失敗した場合には、プラグインに対する変更内容はロールバックされません。

たとえば、when 操作プラグインを ldapcompare 操作とともに使用できます。ディレクトリは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。

PL/SQL の when 操作プラグインは、`ldapadd`、`ldapdelete` および `ldapmodify` でサポートされています。Java の when 操作プラグインは、`ldapadd`、`ldapdelete`、`ldapmoddn`、`ldapmodify` および `ldapsearch` でサポートされています。

when_replace 操作サーバー・プラグイン

when_replace 操作プラグインは、操作用のサーバーのコードのかわりに実行されます。たとえば、when_replace 操作プラグインを ldapcompare 操作とともに使用できます。ディレクトリは比較コードを実行しません。そのかわりに、プラグイン・モジュールを使用して比較を実行します。

PL/SQL の when_replace 操作プラグインは、ldapadd、ldapcompare、ldapdelete、ldapmodify および ldapbind でのみサポートされています。

Java の when_replace 操作プラグインは、ldapadd、ldapbind、ldapcompare、ldapdelete、ldapmoddn、ldapmodify および ldapsearch でサポートされています。

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 の下のディレクトリ・スキーマに作成します。

プラグイン構成エントリ

表 11-1 に、プラグイン構成で指定できるオブジェクト・クラスおよび属性を示します。

表 11-1 プラグイン構成オブジェクトおよび属性

名前	値	必須かどうか
objectclass	orclPluginConfig	はい
objectclass	top	いいえ
dn	プラグイン・エントリ DN。	はい
cn	プラグイン・エントリ名。	はい
orclPluginAttributeList	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。ldapcompare および ldapmodify のプラグイン専用です。	いいえ
orclPluginEnable	0 = 使用禁止（デフォルト） 1 = 使用可能	いいえ
orclPluginEntryProperties	ldapsearch のフィルタ・タイプの値。たとえば、orclPluginEntryProperties:(&(objectclass=inetorgperson)(sn=Cezanne)) と指定すると、ターゲット・エントリの objectclass が inetorgperson であり、sn が Cezanne である場合は、プラグインが起動されません。	いいえ
orclPluginIsReplace	0 = 使用禁止（デフォルト） 1 = 使用可能 when_replace タイミングの場合は、有効化して orclPluginTiming を when に設定します。	いいえ
orclPluginKind	PL/SQL または Java（デフォルトは PL/SQL）。	いいえ

表 11-1 プラグイン構成オブジェクトおよび属性（続き）

名前	値	必須かどうか
orclPluginLDAPOperation	次のいずれかの値です。 ldapcompare ldapmodify ldapbind ldapadd ldapdelete ldapsearch ldapmoddn（Java のみ）	はい
orclPluginName	プラグイン名。	はい
orclPluginFlexfield	カスタム・テキスト情報（Java のみ）。 サブタイプを特定するには、orclPluginFlexfield;minPwdLength: 8 のように、orclPluginFlexfield;subtypename を指定します。	いいえ
orclPluginBinaryFlexfield	カスタム・バイナリ情報（Java のみ）。	いいえ
orclPluginSecuredFlexfield	クリアテキストには表示しないカスタム・テキスト情報（Java のみ）。 サブタイプを特定するには、orclPluginSecuredFlexfield;telephonenumber1: 650.123.456 のように、orclPluginSecuredFlexfield;subtypename を指定します。値は暗号化形式で保存および表示されます。検索結果では、orclPluginSecuredFlexfield;telephonenumber1: 1291zjs8134 のように表示されます。 ユーザーがこの属性をクリアテキストで取得できないように、Oracle Internet Directory のプライバシ・モードが有効化されていることを確認してください。『Oracle Internet Directory 管理者ガイド』の「受信した機密の属性のプライバシ」を参照してください。	いいえ
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、プラグインの登録時に orclpluginrequestgroup:cn=security,cn=groups,dc=oracle,dc=com と指定すると、LDAP リクエストが 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ものではない場合は、プラグインが起動されません。	いいえ

表 11-1 プラグイン構成オブジェクトおよび属性（続き）

名前	値	必須かどうか
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、プラグインの登録時に orclpluginrequestgroup: cn=security,cn=groups,dc=oracle,dc=com と指定すると、LDAP リクエストが 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ものである場合は、プラグインが起動されません。	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。 これは post 操作タイプのプラグイン専用です。	いいえ
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 when_replace タイミングの場合は、when を指定し、orclPluginIsReplace を有効化します。	いいえ
orclPluginType	次のいずれかの値です。 operational attribute password_policy syntax matchingrule **関連項目 :** 11-3 ページの「ディレクトリがサポートする LDAP 操作およびタイミング」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
orclPluginClassReloadEnabled	値が 1 の場合、サーバーは、プラグインが起動されるたびにプラグイン・クラスをリロードします。値が 0 の場合、サーバーは、プラグインの最初の起動時にのみクラスをロードします。	

コマンドライン・ツールによるプラグイン構成エントリの追加

コマンドラインからプラグイン構成エントリを追加するには、プラグイン構成を含む LDIF ファイルを作成します。cn=plugin,cn=subconfigsubentry に DN を指定します。

次の 2 つの部分で構成されている LDIF ファイル my_ldif_file.ldif 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。

```
dn: 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

dn: 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
```

次のような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はレプリケートされません。レプリケートすると、一貫性のない状態になるためです。

Oracle Directory Manager を使用したプラグインの管理

プラグインは、Oracle Directory Manager を使用して登録、編集および削除できます。

Oracle Directory Manager を使用したプラグインの登録

プラグインを登録するには、次のようにします。

1. ナビゲータ・ペインで、「Oracle Internet Directory サーバー」→「< ディレクトリ・サーバー・インスタンス >」を開き、「プラグイン管理」を選択します。プラグイン管理ウィンドウが右側のペインに表示されます。
2. **「作成」**を選択します。「新規プラグイン」ダイアログ・ボックスが表示されます。
3. 「新規プラグイン」ダイアログ・ボックスに値を入力します。
4. 値を入力したら、「OK」を選択します。プラグイン管理ウィンドウに戻ります。作成したプラグインが「プラグイン・エントリ名」列に表示されています。
5. **「適用」**を選択します。

Oracle Directory Manager を使用したプラグインの編集

プラグイン・エントリを編集するには、次のようにします。

1. ナビゲータ・ペインで、「Oracle Internet Directory サーバー」→「< ディレクトリ・サーバー・インスタンス >」を開き、**「プラグイン管理」**を選択します。プラグイン管理ウィンドウが右側のペインに表示されます。
2. 右側のペインで、編集するプラグイン・エントリの名前を選択し、**「編集」**を選択します。「プラグイン」ダイアログ・ボックスが表示されます。
3. 「プラグイン」ダイアログ・ボックスで、適切なフィールドの値を編集します。
4. **「OK」**を選択します。

Oracle Directory Manager を使用したプラグインの削除

プラグインを削除するには、次のようにします。

1. ナビゲータ・ペインで、「Oracle Internet Directory サーバー」→「< ディレクトリ・サーバー・インスタンス >」を開き、**「プラグイン管理」**を選択します。プラグイン管理ウィンドウが右側のペインに表示されます。
2. 右側のペインで、削除するプラグインの名前を選択し、**「編集」**を選択します。「プラグイン」ダイアログ・ボックスが表示されます。
3. 「プラグイン」ダイアログ・ボックスで、**「削除」**を選択し、要求された場合には削除内容を確認します。プラグイン管理ウィンドウに戻ります。削除したプラグイン・エントリがリストから削除されています。

12

PL/SQL サーバー・プラグイン

この章では、PL/SQL におけるプラグイン・フレームワークの使用方法について説明します。

この章では、次の項目について説明します。

- PL/SQL サーバー・プラグインの設計、作成および使用
- PL/SQL プラグインの例
- PL/SQL プラグイン・フレームワークでのバイナリ・サポート
- データベース・オブジェクト・タイプの定義
- PL/SQL プラグイン・プロシージャの仕様

PL/SQL サーバー・プラグインの設計、作成および使用

この項では、次の項目について説明します。

- PL/SQL プラグインの注意事項
- PL/SQL プラグインの作成
- PL/SQL プラグインのコンパイル
- PL/SQL プラグインの管理
- PL/SQL プラグインの有効化と無効化
- PL/SQL プラグインにおける例外処理
- PL/SQL プラグイン LDAP API
- PL/SQL プラグインおよびレプリケーション
- PL/SQL プラグインおよびデータベース・ツール
- PL/SQL プラグインのセキュリティ
- PL/SQL プラグインのデバッグ
- PL/SQL プラグイン LDAP API の仕様
- データベースの制限事項

PL/SQL プラグインの注意事項

PL/SQL プラグインには、次の注意事項があります。

PL/SQL プラグイン操作のタイプ

PL/SQL プラグインは、`ldapbind`、`ldapadd`、`ldapmodify`、`ldapcompare`、`ldapsearch` および `ldapdelete` の各操作にのみ関連付けることができます。PL/SQL プラグインを `moddn` に関連付けることはできません。プラグインを `moddn` に関連付ける場合には、Java プラグインを使用する必要があります。

PL/SQL プラグインの名前付け

プラグインが他のプラグインやストアド・プロシージャと同じデータベース・スキーマを共有する場合、そのプラグインの名前（PL/SQL パッケージ名）は一意であることが必要です。しかし、他のデータベース・スキーマ・オブジェクト（表やビューなど）とは同じ名前を共有できます。ただし、この種の共有はお薦めしません。

PL/SQL プラグインの作成

PL/SQL プラグイン・モジュールの作成は、PL/SQL パッケージの作成に似ています。どちらも仕様部と本体部を持ちます。プラグイン仕様は Oracle Internet Directory とカスタム・プラグイン間のインタフェースの役割を果すため、プラグインではなくディレクトリに定義します。

セキュリティ上の理由と LDAP サーバーの整合性を維持する目的から、PL/SQL プラグインをコンパイルできるのは ODS データベース・スキーマ内のみです。プラグインのコンパイルは、Oracle Internet Directory のバックエンド・データベースとして機能するデータベースで行う必要があります。

プラグイン・モジュール・インタフェースのパッケージ仕様

パッケージ仕様はプラグインによって異なります。表 12-1 に示すように、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。詳細は、「PL/SQL プラグイン・プロシージャの仕様」を参照してください。

表 12-1 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		×

表 12-2 に、様々なプラグイン・プロシージャを示します。また、各プロシージャで使用されるパラメータについても説明します。

表 12-2 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

表 12-2 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
ldapcompare 後	POST_COMPARE	ldapcontext、compare result、dn、attribute、value	return code、error message
ldapadd 前	PRE_ADD	ldapcontext、dn、entry	return code、error message
ldapadd 時	WHEN_ADD	ldapcontext、dn、entry	return code、error message
ldapadd 時（ただし、デフォルト・サーバーの動作を置換）	WHEN_ADD_REPLACE	ldapcontext、dn、entry	return code、error message
ldapadd 後	POST_ADD	ldapcontext、add result、dn、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、12-6 ページの「エラー処理」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、12-23 ページの「PL/SQL プラグイン・プロシージャの仕様」を参照してください。

PL/SQL プラグインのコンパイル

Oracle Internet Directory のバックエンド・データベースと同じ役割を果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する必要があります。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を踏んで行われます。

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 SELECT 文を使用して USER_ERRORS ビューから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;
```

PL/SQL プラグインの管理

この項では、プラグインの変更およびデバッグ方法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。

PL/SQL 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。

PL/SQL プラグインにおける例外処理

PL/SQL プラグインの各プロシージャには、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。

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。

表 12-3 に、リターン・コードの有効な値を示します。

表 12-3 プラグインのリターン・コードの有効な値

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
-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 は、プラグイン・プログラムを実行するたびに、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を判断します。

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（リクエスト元が 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

表 12-4 に、プラグイン例外の発生場所とディレクトリによる処理方法を示します。

表 12-4 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。

表 12-5 に、LDAP 操作障害時のディレクトリの対応を示します。

表 12-5 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	pre 操作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	post 操作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	when 操作タイプのプラグインの変更はロールバックされます。
WHEN	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。

PL/SQL 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ることに注意してください。
- Oracle Internet Directory にはプラグイン LDAP API が用意されています。LDAP リクエストに関連付けられている構成済のプラグインがある場合、このプラグインは、ディレクトリ・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・モジュールで指定されたディレクトリ・サーバーに接続するための API が提供されます。プラグインを実行しているサーバーに接続するには、この API を使用する必要があります。外部サーバーに接続する場合は、DBMS_LDAP API を使用します。

各プラグイン・モジュールでは、ldapcontext が Oracle ディレクトリ・サーバーから渡されます。プラグイン LDAP API がコールされると、セキュリティとバインドの目的で ldapcontext が渡されます。この ldapcontext を使用してバインドすると、Oracle Internet Directory は、この LDAP リクエストがプラグイン・モジュールからのリクエスト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、ディレクトリは後続のプラグインをトリガーしません。ディレクトリは、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： 12-9 ページの「PL/SQL プラグイン LDAP API の仕様」

PL/SQL プラグインおよびレプリケーション

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
- プラグイン・プログラムやその他の LDAP 操作でディレクトリ・エントリが変更される
- 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

PL/SQL プラグインおよびデータベース・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。

PL/SQL プラグインのセキュリティ

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、ディレクトリの `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。

PL/SQL プラグインのデバッグ

プラグインの処理および内容を検証するには、Oracle Internet Directory のプラグイン・デバッグ・メカニズムを使用します。次のコマンドは、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、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');
  ```

  生成されたデバッグ・メッセージは、プラグイン・デバッグ表に格納されます。

- 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
  ```

PL/SQL プラグイン LDAP API の仕様

Oracle Internet Directory が提供する PL/SQL プラグイン LDAP API のパッケージ仕様は次のとおりです。

```
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 plug-in context
    FUNCTION get_requester (ldappluginctx IN ODS.plugincontext)
      RETURN VARCHAR2;
END LDAP_PLUGIN;
```

データベースの制限事項

Oracle Internet Directory 10*g*（10.1.4.0.1）では、複数の異なるリリースの Oracle Database を使用してディレクトリ・データを格納できます。これらのデータベースには、Oracle9*i* Database Server リリース 2（9.2.0.6）以上、および Oracle Database 10*g* リリース 1（10.1.0.4）以上が含まれます。

Oracle Application Server 10*g*（10.1.4.0.1）の場合、次のプラグイン機能は、Oracle9*i* Database Server リリース 2 を対象に実行されているディレクトリ・サーバーではサポートされません。

- Windows ドメインの外部認証プラグイン。
- プラグイン定義に含まれるディレクトリ・サーバーへの接続用として、Oracle Internet Directory PL/SQL PLUGIN API で提供されている LDAP_PLUGIN パッケージの `simple_bind_s()` ファンクション。

PL/SQL プラグインの例

この項では、2 つのサンプル・プラグインを示します。一方は、すべての ldapsearch コマンドを記録します。もう一方は、2 つのディレクトリ情報ツリー（DIT）を同期化します。

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。これには、ldapsearch の post 操作プラグインを使用します。リクエストをすべて記録することも、検索対象の識別名で発生したリクエストのみを記録することも可能です。

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. 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あります。
   - 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
(ldappluginсontext 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 plug-in error msg';
   COMMIT;
EXCEPTION
   WHEN others THEN
      rc := 1;
      errormsg := 'exception: post_search plug-in';
END;
END LDAP_PLUGIN_EXAMPLE1;
/
```

4. プラグイン・エントリを Oracle Internet Directory に登録します。

```
dn: 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 : 相互に依存する 2 つの製品が cn=Products, cn=oraclecontext にあります。この相互依存性は、これらの製品のコンテナ内のユーザーにも適用されます。最初のディレクトリ情報ツリー（製品 1）のユーザーが削除された場合、もう一方のディレクトリ情報ツリー（製品 2）の対応するユーザーを削除する必要があります。

トリガーを設定して、最初のディレクトリ情報ツリーのユーザーが削除された場合に 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する、または渡すことは可能でしょうか。

解答 : 可能です。ldapdelete の post 操作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 というネーミング・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 というネーミング・コンテキストがある場合、2 人のユーザーは同一の ID 属性を共有します。ldapdelete の post 操作プラグイン・モジュール内で、LDAP_PLUGIN と DBMS_LDAP の API を使用して、2 番目のディレクトリ情報ツリー内のユーザーを削除できます。

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 の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るようになります。

注意 : ldapmodify の post 操作プラグインを使用して 2 つの Oracle Internet Directory ノード間で変更を同期化する場合、一方のノードから他方のノードへすべての属性をプッシュすることはできません。これは、プラグイン・モジュールで取得された変更（変更構造体）に操作属性が含まれているためです。このような操作属性は各ノードで生成されるものであり、標準の LDAP メソッドを使用して変更することはできません。

プラグイン・プログラムを記述する際には、操作属性 authPassword、creatorsname、createtimestamp、modifiersname、modifytimestamp、pwdchangedtime、pwdfailuretime、pwdaccountlockedtime、pwdexpirationwarned、pwdreset、pwdhistory, pwdgraceusetime を同期化の対象から除外します。

pwdchangedtime、pwdfailuretime、authpassword、pwdaccountlockedtime は、配置環境で最もよく使用される属性です。最初に同期化の対象から除外する必要があります。

---

1. 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 plug-in error msg';
EXCEPTION
   WHEN others THEN
      rc := 1;
      errormsg := 'exception: post_delete plug-in';
END;
END LDAP_PLUGIN_EXAMPLE2;
/
(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 plug-in error msg';
EXCEPTION
   WHEN others THEN
      rc := 1;
      errormsg := 'exception: post_delete plug-in';
END;
END LDAP_PLUGIN_EXAMPLE2;
/
```

4. プラグイン・エントリを Oracle Internet Directory に登録します。

   LDIF ファイル register_post_delete.ldif を構成します。

```
dn: 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
```

PL/SQL プラグイン・フレームワークでのバイナリ・サポート

リリース 10.1.2 から、プラグイン LDAP API のオブジェクト定義によって、ldapmodify、ldapadd および ldapcompare のプラグインがディレクトリ・データベースのバイナリ属性にアクセスできるようになりました。これまでは VARCHAR2 型の属性にしかアクセスできませんでした。このオブジェクト定義によってリリース 10.1.2 より前のプラグイン・コードが無効になることはありませんので、このコードの変更は不要です。新しい定義は、「データベース・オブジェクト・タイプの定義」に示します。

この項では、3 種類のプラグインが関係するバイナリ操作について説明します。また、これらのプラグインの例も示します。新しいオブジェクト定義は、3 種類すべての pre、post および when プラグインに適用されます。

3 つの例では、LOB のかわりに RAW ファンクションおよび変数を使用していることに注意してください。

ldapmodify でのバイナリ操作

プラグイン・フレームワークが ldapmodify のプラグインに渡す modobj オブジェクトに、バイナリ属性の値が binvals として格納されるようになりました。この変数は、binvalobj オブジェクトの表です。

このプラグインは、modobj の operation フィールドを調べて、バイナリ操作が実行されているかどうかを判断します。値 DBMS_LDAP.MOD_ADD、DBMS_LDAP.MOD_DELETE、DBMS_LDAP.MOD_REPLACE のいずれかが DBMS_LDAP.MOD_BVALUES と組み合されていないかどうかがチェックされます。たとえば、DBMS_LDAP.MOD_ADD+DBMS_LDAP.MOD_BVALUES の組合せは、変更操作におけるバイナリの追加を示します。

次の例は、別のディレクトリのエントリを変更する、ldapmodify の post 操作プラグインを示しています。このプラグインは、ldapmodify がプラグイン・ディレクトリの同じエントリに同じ変更を適用した後、起動されます。もう一方のディレクトリのエントリは、ディレクトリ情報ツリー cn=users,dc=us,dc=acme,dc=com にあります。

```
create or replace package moduser as
   procedure post_modify(ldapplugincontext IN ODS.plugincontext,
                         result IN integer,
                         dn IN varchar2,
                         mods IN ODS.modlist,
                         rc OUT integer,
                         errormsg OUT varchar2);
end moduser;
/
```

```
show error

CREATE OR REPLACE PACKAGE BODY moduser AS
   procedure post_modify(ldapplugincontext IN ODS.plugincontext,
                         result IN integer,
                         dn IN varchar2,
                         mods IN ODS.modlist,
                         rc OUT integer,
                         errormsg OUT varchar2)
   is
      counter1 pls_integer;
      counter2 pls_integer;
      retval pls_integer := -1;
      user_session DBMS_LDAP.session;
      user_dn varchar(256);
      user_array DBMS_LDAP.mod_array;
      user_vals DBMS_LDAP.string_collection;
      user_binvals DBMS_LDAP.blob_collection;
      ldap_host varchar(256);
      ldap_port varchar(256);
      ldap_user varchar(256);
      ldap_passwd varchar(256);
   begin
      ldap_host :='backup.us.oracle.com';
      ldap_port :='4000';
      ldap_user :='cn=orcladmin';
      ldap_passwd :='welcome';

      plg_debug('START MODIFYING THE ENTRY');

      -- Get a session
         user_session := dbms_ldap.init(ldap_host, ldap_port);

      -- Bind to the directory
         retval := dbms_ldap.simple_bind_s(user_session, ldap_user,
         ldap_passwd);

      -- Create a mod_array
         user_array := dbms_ldap.create_mod_array(mods.count);

      -- Create a user_dn
         user_dn := substr(dn,1,instr(dn,',',1,1))||'cn=users,dc=us,dc=acme,
         dc=com';

        plg_debug('THE CREATED DN IS'||user_dn);

      -- Iterate through the modlist
         for counter1 in 1..mods.count loop

      -- Log the attribute name and operation
         if (mods(counter1).operation > DBMS_LDAP.MOD_BVALUES) then
           plg_debug('THE NAME OF THE BINARY ATTR. IS'||mods(counter1).type);
         else
           plg_debug('THE NAME OF THE NORMAL ATTR. IS'||mods(counter1).type);
         end if;
           plg_debug('THE OPERATION IS'||mods(counter1).operation);

      -- Add the attribute values to the collection
         for counter2 in 1..mods(counter1).vals.count loop
           user_vals(counter2) := mods(counter1).vals(counter2).val;
         end loop;
```

```
      -- Add the attribute values to the collection
         for counter2 in 1..mods(counter1).binvals.count loop
           plg_debug('THE NO. OF BYTES OF THE BINARY ATTR. VALUE IS'
           ||mods(counter1).binvals(counter2).length);
           user_binvals(counter2) := mods(counter1).binvals(counter2).binval;
         end loop;

      -- Populate the mod_array accordingly with binary/normal attributes
         if (mods(counter1).operation >= DBMS_LDAP.MOD_BVALUES) then
           dbms_ldap.populate_mod_array(user_array,mods(counter1).operation -
           DBMS_LDAP.MOD_BVALUES,mods(counter1).type,user_binvals);
           user_binvals.delete;
         else
           dbms_ldap.populate_mod_array(user_array,mods(counter1).operation,
           mods(counter1).type,user_vals);
           user_vals.delete;
         end if;

        end loop;

      -- Modify the entry
         retval := dbms_ldap.modify_s(user_session,user_dn,user_array);
         if retval = 0 then
           rc := 0;
           errormsg:='No error occured while modifying the entry';
         else
           rc := retval;
           errormsg :='Error code'||rc||' while modifying the entry';
         end if;

      -- Free the mod_array
         dbms_ldap.free_mod_array(user_array);

       plg_debug('FINISHED MODIFYING THE ENTRY');

   exception
 WHEN others THEN
   plg_debug (SQLERRM);
 end;
end moduser;
/
show error

exit;
```

ldapadd でのバイナリ操作

プラグイン・フレームワークが ldapadd のプラグインに渡す entryobj オブジェクトに、バイナリ属性が binattr として格納されるようになりました。この変数は、binattrobj オブジェクトの表です。次の例は、プラグイン・ディレクトリの変更（追加されたユーザー）を別のディレクトリに伝播する追加後プラグインを示しています。後者のディレクトリのエントリは、ディレクトリ情報ツリー cn=users,dc=us,dc=acme,dc=com にあります。

```
create or replace package adduser as
   procedure post_add(ldapplugincontext IN ODS.plugincontext,
                      result IN integer,
                      dn IN varchar2,
                      entry IN ODS.entryobj,
                      rc OUT integer,
                      errormsg OUT varchar2);
end adduser;
/
show error

CREATE OR REPLACE PACKAGE BODY adduser AS
   procedure post_add(ldapplugincontext IN ODS.plugincontext,
                      result IN integer,
                      dn IN varchar2,
                      entry IN ODS.entryobj,
                      rc OUT integer,
                      errormsg OUT varchar2)
   is
      counter1 pls_integer;
      counter2 pls_integer;
      retval pls_integer := -1;
      s integer;
      user_session DBMS_LDAP.session;
      user_dn varchar(256);
      user_array DBMS_LDAP.mod_array;
      user_vals DBMS_LDAP.string_collection;
      user_binvals DBMS_LDAP.blob_collection;
      ldap_host varchar(256);
      ldap_port varchar(256);
      ldap_user varchar(256);
      ldap_passwd varchar(256);
   begin
      ldap_host :='backup.us.oracle.com';
      ldap_port :='4000';
      ldap_user :='cn=orcladmin';
      ldap_passwd :='welcome';

      plg_debug('START ADDING THE ENTRY');

      -- Get a session
         user_session := dbms_ldap.init(ldap_host, ldap_port);

      -- Bind to the directory
         retval := dbms_ldap.simple_bind_s(user_session, ldap_user, ldap_passwd);

      -- Create a mod_array
         user_array := dbms_ldap.create_mod_array(entry.binattr.count +
         entry.attr.count);

      -- Create a user_dn
         user_dn := substr(dn,1,instr(dn,',',1,1))||'cn=users,dc=us,dc=acme,
         dc=com';
         plg_debug('THE CREATED DN IS'||user_dn);
```

```
      -- Populate the mod_array with binary attributes
         for counter1 in 1..entry.binattr.count loop
           for counter2 in 1..entry.binattr(counter1).binattrval.count loop
             plg_debug('THE NAME OF THE BINARY ATTR. IS'||
             entry.binattr(counter1).binattrname);
             s := dbms_lob.getlength(entry.binattr(counter1).
             binattrval(counter2));
             plg_debug('THE NO. OF BYTES OF THE BINARY ATTR. VALUE IS'||s);
             user_binvals(counter2) := entry.binattr(counter1).
             binattrval(counter2);
           end loop;
         dbms_ldap.populate_mod_array(user_array,DBMS_LDAP.MOD_ADD,
         entry.binattr(counter1).binattrname,user_binvals);
         user_binvals.delete;
        end loop;

      -- Populate the mod_array with attributes
         for counter1 in 1..entry.attr.count loop
           for counter2 in 1..entry.attr(counter1).attrval.count loop
             plg_debug('THE NORMAL ATTRIBUTE'||entry.attr(counter1).attrname||'
             HAS THE VALUE'||entry.attr(counter1).attrval(counter2));
             user_vals(counter2) := entry.attr(counter1).attrval(counter2);
           end loop;
         dbms_ldap.populate_mod_array(user_array,DBMS_LDAP.MOD_ADD,
         entry.attr(counter1).attrname,user_vals);
         user_vals.delete;
         end loop;

      -- Add the entry
         retval := dbms_ldap.add_s(user_session,user_dn,user_array);
         plg_debug('THE RETURN VALUE IS'||retval);
         if retval = 0 then
           rc := 0;
           errormsg:='No error occured while adding the entry';
         else
           rc := retval;
           errormsg :='Error code'||rc||' while adding the entry';
         end if;

      -- Free the mod_array
         dbms_ldap.free_mod_array(user_array);
         retval := dbms_ldap.unbind_s(user_session);

         plg_debug('FINISHED ADDING THE ENTRY');

   exception
 WHEN others THEN
   plg_debug (SQLERRM);
 end;
end adduser;
/
show error

exit;
```

ldapcompare でのバイナリ操作

ldapcompare のプラグインは、オーバーロードされた 3 つの新しいモジュール・インタフェースを使用してバイナリ属性を比較できます。これらのインタフェースを使用して、バイナリと非バイナリ両方の属性を処理するプラグイン・パッケージを開発する場合、パッケージに 2 つの別々のプロシージャを含める必要があります。orclPluginName はプラグイン・エントリに 1 つしか登録できないため、2 つのプロシージャのパッケージ名は同じです。

既存のプラグイン・パッケージを更新してバイナリ属性を比較するプロシージャを含めたら、パッケージを再インストールします。そのプラグイン・パッケージに依存するパッケージを再コンパイルします。

3 つの新しいインタフェースは次のようになります。

```
PROCEDURE pre_compare (ldapplugincontext IN ODS.plugincontext,
                       dn          IN VARCHAR2,
                       attrname    IN VARCHAR2,
                       attrval     IN BLOB,
                       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 BLOB,
                                rc          OUT INTEGER,
                                errormsg    OUT VARCHAR2 );

PROCEDURE post_compare (ldapplugincontext IN ODS.plugincontext,
                        result      IN INTEGER,
                        dn          IN VARCHAR2,
                        attrname    IN VARCHAR2,
                        attrval     IN BLOB,
                        rc          OUT INTEGER,
                        errormsg    OUT VARCHAR2 );
```

次の例では、プラグイン・ディレクトリのエントリのバイナリ属性と別のディレクトリのエントリのバイナリ属性を比較します。このパッケージは、サーバーの比較コードをプラグインの比較コードに置き換えます。また、バイナリと非バイナリ両方の属性を処理します。そのため、2 つの別々のプロシージャが含まれています。

```
create or replace package compareattr as
   procedure when_compare_replace(ldapplugincontext IN ODS.plugincontext,
                         result OUT integer,
                         dn IN varchar2,
                         attrname IN VARCHAR2,
                         attrval IN BLOB,
                         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);
end compareattr;
/
show error
```

```
CREATE OR REPLACE PACKAGE BODY compareattr AS
   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)
   is
   pos            INTEGER := 2147483647;
   begin
      plg_debug('START');
      plg_debug('THE ATTRNAME IS'||attrname||' AND THE VALUE IS'||attrval);
      plg_debug('END');
      rc := 0;
      errormsg :='No error!!!';
   exception
   WHEN others THEN
    plg_debug ('Unknown UTL_FILE Error');
   end;

   procedure when_compare_replace(ldapplugincontext IN ODS.plugincontext,
                         result OUT integer,
                         dn IN varchar2,
                         attrname IN VARCHAR2,
                         attrval IN BLOB,
                         rc OUT integer,
                         errormsg OUT varchar2)
   is
      counter pls_integer;
      retval pls_integer := -1;
      cmp_result integer;
      s integer;
      user_session DBMS_LDAP.session;
      user_entry DBMS_LDAP.message;
      user_message DBMS_LDAP.message;
      user_dn varchar(256);
      user_attrs DBMS_LDAP.string_collection;
      user_attr_name VARCHAR2(256);
      user_ber_elmt DBMS_LDAP.ber_element;
      user_vals DBMS_LDAP.blob_collection;
      ldap_host varchar(256);
      ldap_port varchar(256);
      ldap_user varchar(256);
      ldap_passwd varchar(256);
      ldap_base varchar(256);
   begin
      ldap_host :='backup.us.oracle.com';
      ldap_port :='4000';
      ldap_user :='cn=orcladmin';
      ldap_passwd :='welcome';
      ldap_base := dn;

      plg_debug('STARTING COMPARISON IN WHEN REPLACE PLUG-IN');

        s := dbms_lob.getlength(attrval);
        plg_debug('THE NUMBER OF BYTES OF ATTRVAL'||s);

      -- Get a session
         user_session := dbms_ldap.init(ldap_host, ldap_port);
```

```
-- Bind to the directory
   retval := dbms_ldap.simple_bind_s(user_session, ldap_user, ldap_passwd);

-- issue the search
   user_attrs(1) := attrname;
   retval := DBMS_LDAP.search_s(user_session, ldap_base,
                                DBMS_LDAP.SCOPE_BASE,
                               'objectclass=*',
                                user_attrs,
                                0,
                                user_message);

-- Get the entry in the other OID server
   user_entry := DBMS_LDAP.first_entry(user_session, user_message);

-- Log the DN and the Attribute name
   user_dn := DBMS_LDAP.get_dn(user_session, user_entry);
   plg_debug('THE DN IS'||user_dn);
   user_attr_name := DBMS_LDAP.first_attribute(user_session,user_entry,
   user_ber_elmt);

-- Get the values of the attribute
   user_vals := DBMS_LDAP.get_values_blob(user_session, user_entry,
   user_attr_name);

-- Start the binary comparison between the ATTRVAL and the attribute
-- values
   if user_vals.count > 0 then
    for counter in user_vals.first..user_vals.last loop
      cmp_result := dbms_lob.compare(user_vals(counter),attrval,
                       dbms_lob.getlength(user_vals(counter)),1,1);
      if  cmp_result = 0 then
           rc := 0;
           -- Return LDAP_COMPARE_TRUE
           result := 6;
           plg_debug('THE LENGTH OF THE ATTR.'||user_attr_name||' IN THE
           ENTRY IS'||dbms_lob.getlength(user_vals(counter)));
           errormsg :='NO ERROR. THE COMPARISON HAS SUCCEEDED.';
           plg_debug(errormsg);
           plg_debug('FINISHED COMPARISON');
           return;
      end if;
    end loop;
  end if;

rc := 1;
-- Return LDAP_COMPARE_FALSE
   result := 5;
   errormsg :='ERROR. THE COMPARISON HAS FAILED.';
   plg_debug('THE LENGTH OF THE ATTR.'||user_attr_name||' IN THE ENTRY IS'
   ||dbms_lob.getlength(user_vals(user_vals.last)));
   plg_debug(errormsg);
   plg_debug('FINISHED COMPARISON');

-- Free user_vals
   dbms_ldap.value_free_blob(user_vals);
   exception
   WHEN others THEN
     plg_debug (SQLERRM);
   end;
end compareattr;
/
```

```
show error

exit;
```

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 に導入されたオブジェクト型の定義を示します。これらの定義は、すべて Oracle Directory Server のデータベース・スキーマにあります。API にはプラグインがデータベースからバイナリ・データを抽出するためのオブジェクト型が含まれていることに注意してください。

```
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 binattrvalType as TABLE OF BLOB;
/
create or replace type binattrobj as object (
binattrname    varchar2(2000),
binattrval     binattrvalType
);
/
create or replace type binattrlist as table of binattrobj;
/
create or replace type entryobj as object (
entryname      varchar2(2000),
attr           attrlist,
binattr        binattrlist
);
/
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 binvalobj as object (
length         integer,
binval         blob
);
/
create or replace type binvallist as table of binvalobj;
/
create or replace type modobj as object (
operation      integer,
type           varchar2(256),
vals           bvallist,
binvals        binvallist
);
```

```
/
create or replace type modlist as table of modobj;
```

PL/SQL プラグイン・プロシージャの仕様

プラグインを使用する場合、各プラグインに定義されたシグネチャに従う必要があります。次に各シグネチャを示します。

```
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 pre_compare (ldapplugincontext IN ODS.plugincontext,
dn              IN VARCHAR2,
attrname        IN VARCHAR2,
attrval         IN BLOB,
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 when_compare_replace (ldapplugincontext IN ODS.plugincontext,
result          OUT INTEGER,
dn              IN VARCHAR2,
attrname        IN VARCHAR2,
attrval         IN BLOB,
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 post_compare (ldapplugincontext IN ODS.plugincontext,
result          IN INTEGER,
dn              IN VARCHAR2,
attrname        IN VARCHAR2,
attrval         IN BLOB,
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,
dn             IN  VARCHAR2,
passwd         IN  VARCHAR2,
rc             OUT INTEGER,
errormsg       OUT 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,
result         IN  INTEGER,
dn             IN  VARCHAR2,
passwd         IN  VARCHAR2,
rc             OUT INTEGER,
errormsg       OUT VARCHAR2
);
```

13

Java サーバー・プラグイン

顧客と社内の両方からの要望に応え、Oracle Internet Directory 10*g*（10.1.4.0.1）のサーバー・プラグイン・フレームワークに Java API を追加しました。サーバー・チェーンなど、Oracle Internet Directory の新機能のいくつかは、Java プラグイン API を使用して開発されました。

この章では、次の項目について説明します。

- Java プラグインの利点
- Java プラグインの設定
- Java プラグイン API
- Java プラグイン・エラーおよび例外処理
- Java プラグインのデバッグおよびロギング
- Java プラグインの例

Java プラグインの利点

Java 言語自体の利点に加え、Java サーバー・プラグインには、PL/SQL プラグインに優る次のような利点があります。

- サーバーとプラグインの間の双方向通信
- プラグインで検索結果を戻すことが可能
- moddn 操作のサポート
- より高いパフォーマンス
- データベースの知識が不要
- セキュリティの強化
- デバッグ機能の強化

Java プラグインの設定

Java プラグインは次のように設定します。

1. 事前定義済のクラス定義およびメソッドを使用して、スタンドアロンの Java プログラムを作成します。プラグインは、jar ファイルまたはパッケージとして実装できます。
2. プラグイン・ファイルまたはパッケージをコンパイルします。コンパイルする前に、クラスパスが `$ORACLE_HOME/ldap/jlib/ospf.jar` に設定されていることを確認してください。コンパイルでエラーが発生していないことを確認します。
3. クラス・ファイル、jar またはパッケージを、事前に定義したクラスの場所 `$ORACLE_HOME/ldap/server/plugin` に配置します。
4. プラグイン構成エントリを追加して Java プラグインを登録します。

   エントリは、コマンドラインまたは Oracle Directory Manager を使用して追加できます。詳細は、11-5 ページの「プラグインの登録」を参照してください。

jar ファイルには任意の名前を付けられます。マニフェスト・ファイルには、後ろに Java プラグインの名前が続く、属性 Main-Class が含まれている必要があります。次に例を示します。

```
Main-Class: myjavaplugin
```

プラグイン構成エントリの `orclPluginName` 属性の値は、次のいずれかと対応している必要があります。

- クラス・ファイル内のクラス名
- パッケージ内のクラスの完全修飾名
- jar ファイル名

`myjavaplugin` という名前を指定すると、サーバーは対応するクラス `$ORACLE_HOME/ldap/server/plugin/myjavaplugin.class` を検索します。`myjavaplugin.jar` という名前を指定すると、サーバーは対応する jar ファイル `$ORACLE_HOME/ldap/server/plugin/myjavaplugin.jar` を検索します。`my.package.myjavaplugin` という名前を指定すると、サーバーはクラスのパスは `$ORACLE_HOME/ldap/server/plugin/my/package/myjavaplugin` であると認識します。

これらの手順を実行すると、起動条件が満たされた場合にはサーバーによりプラグインが起動されます。

jar ファイルに含まれるクラスが環境内に存在しないようにしてください。存在すると、予期しないエラーが発生する可能性があります。この問題を修正するには、環境からクラスを削除し、Oracle Internet Directory サーバーを再起動します。JAR またはクラス・ファイルが別の JAR ファイルやクラス・ファイルに依存している場合は、依存している JAR ファイルまたはクラス・ファイルのパスを `CLASSPATH` に追加して、Oracle Internet Directory サーバーを再起動します。

プラグインが実行されるたびに、サーバーで Java プラグイン・クラスをリロードするかどうかを制御できます。属性 orclPluginClassReloadEnabled の値が 1 の場合、サーバーはプラグイン・クラスを毎回リロードします。値が 0 の場合、サーバーは、プラグインの最初の実行時にのみクラスをロードします。

Oracle Internet Directory サーバー・プラグイン・フレームワークの jar ファイルのパスは、$ORACLE_HOME/ldap/jlib/ospf.jar です。

Java プラグイン API

この項では、API の高度な概要、およびメイン・クラスやインタフェースの役割を説明します。すべての Java サーバー・プラグイン・クラスおよびインタフェースの詳細は、Javadoc の『Oracle Internet Directory API Reference』を参照してください。

この項の項目は次のとおりです。

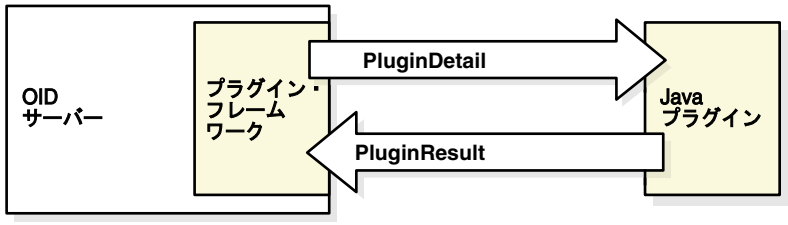
- サーバーとプラグインの通信
- Java プラグインの構造
- PluginDetail
- PluginResult
- ServerPlugin インタフェース

注意： Java プラグインでは System.exit() を使用しないでください。使用すると、Oracle ディレクトリ・サーバーが予期しない動作をする可能性があります。

サーバーとプラグインの通信

すべての Java プラグインでは、プラグインと Oracle Internet Directory サーバーとの通信に ServerPlugin インタフェースが使用されます。サーバーにより Java プラグインが起動される際に、PluginDetail オブジェクトが構成され、そのオブジェクトに含めた情報がプラグインに渡されます。プラグインにより、PluginResult オブジェクトが構成されます。タスクが完了すると、プラグインにより PluginResult オブジェクトがサーバーに戻されます。プラグインにより、PluginDetail から受信した情報に変更または追加が加えられ、PluginResult オブジェクトに含めたその情報がサーバーに戻される場合もあります。図 13-1 に、Oracle Internet Directory サーバーと Java プラグインの通信を示します。

図 13-1 サーバーと Java プラグインの通信



Java プラグインでは、ログ・ファイルのメッセージを監査用に記録するために、ServerLog クラスが使用されます。

Java プラグインの構造

次に、Java プラグインの一般的な構造を示します。

```
public class Java_Plug-in_Class_Name {extends
ServerPluginAdapter}  {
     public PluginResult
Name_of_ServerPlugin_Method(PluginDetail plgObj)
throws Exception {
      // Plug-in Code
      }
}
```

または

```
public class Java_Plug-in_Class_Name {implements
ServerPlugin}  {
     public PluginResult
Name_of_ServerPlugin_Method(PluginDetail plgObj)
throws Exception {
      // Plug-in Code
      }
}
```

PluginDetail

PluginDetail には、次の情報が含まれます。

- Server
- LdapBaseEntry
- LdapOperation
- PluginFlexfield

Server

このオブジェクトには、プラグインが実行される Oracle Internet Directory サーバーに関するメタデータ情報が含まれます。含まれる情報は次のとおりです。

- Hostname
- Port
- LdapContext

Hostname および Port は、サーバーが稼働しているホストとポートを示します。

LdapContext オブジェクトは、プラグインによるサーバーへの接続を可能にし、プラグインが接続を取得中であることをサーバーに通知します。それ自体が ldapbind を実行する ldapbind プラグインなどに必要な動作です。LdapContext オブジェクトを使用してサーバーに接続すると、サーバーにより同じプラグインが起動され、それが原因で無限ループが発生するのを防ぐことができます。

次のコード・フラグメントは、プラグインが PluginDetail から Server オブジェクトを取得し、サーバーに接続する様子を示しています。

```
// An LDAP Bind Plug-in
public class MyBindPlugin extends ServerPluginAdapter
{
  …..
  // Retrieve the Server Object from the PluginDetail
  Server srvObj = plgObj.getServer();
  ……
  // This bind will not result in the LDAP Bind Plug-in being called
  // in an infinite loop
```

```
  InitialLdapContext myConn =

(InitialLdapContext)srvObj.getLdapContextFromServerPlugin();
  myConn.bind(…);
  ….
}
```

この例で使用されているメソッドの詳細は、Javadoc の『Oracle Internet Directory API Reference』を参照してください。

LdapBaseEntry

`LdapBaseEntry` には、次の情報が含まれます。

- DN
- Attributes

ldapadd を除き、すべての操作の DN 情報をサーバーから送信する必要があります。表 13-1 に、各操作の DN の内容を示します。

表 13-1 各 LDAP 操作の DN 情報の内容

操作	DN の内容
ldapadd	DN は送信されません。
ldapbind	ディレクトリ・サーバーがバインドを試行するエントリ。
ldapcompare	比較が実行されるベース・エントリ。
ldapdelete	タイミングが pre および when の場合に削除されるエントリ。 タイミングが post の場合、DN は送信されません。
ldapmoddn	移動対象のベース・エントリ。
ldapmodify	タイミングが pre および when の場合に変更が行われるエントリ。 タイミングが post の場合に変更されるエントリ。
ldapsearch	検索のベース・エントリ。

属性は JNDI 属性です。

`LdapBaseEntry` には、DN および Attributes にアクセスするためのメソッドがあります。`LdapBaseEntry` がグループ・エントリで、エントリ・キャッシュ機能が無効化されている場合、パフォーマンス上の理由から、属性 `uniquemember` および `member` にはアクセスできません。

> **関連資料：** パフォーマンス・チューニングの詳細は、『Oracle Internet Directory 管理者ガイド』のチューニングに関する章を参照してください。

LdapOperation

すべてのプラグインは、add、bind、compare、delete、moddn、modify または search の 7 つの基本的な LDAP 操作のいずれかに関連付けられています。`LdapOperation` オブジェクトには、次の情報が含まれ、7 つのすべての操作に渡されます。

- Bind DN
- Server Controls
- Operation Result Code

Bind DN は、LDAP 操作をリクエストする ID の識別名です。Server Controls は、制御情報を含むベクターです。操作中に任意のサーバー・コントロールがサーバーに渡された場合、制御情報は Server Controls の Java プラグインに渡されます。表 13-2 に示すように、Operation Result Code の内容は操作のタイミングにより異なります。when_replace 操作の場合、プラグインによる Operation Result Code の情報の変更が可能で、その情報は `PluginResult` に含められてサーバーに渡されます。

表 13-2 Operation Result Code の動作

プラグインのタイミング	Operation Result Code の内容および動作
pre	使用されません。
when	
when_replace	プラグインによって実行された LDAP 操作のエラー・ステータス。プラグインからサーバーへの出力です。
post	サーバーによって実行された LDAP 操作のエラー・ステータス。サーバーからプラグインへの入力です。

`LdapOperation` には、コンテンツを取得および変更するためのメソッドもあります。

7 つの LDAP 操作を表す異なる 7 つのクラスにより、`LdapOperation` クラスが拡張されます。各サブクラスには、`LdapOperation` 情報の他に、クラス固有の情報も含まれます。表 13-3 に、クラスおよびクラス固有の情報を示します。表 13-3 の各クラス名は、そのクラスの詳細が説明されている項にリンクされています。

表 13-3 LdapOperation のサブクラスおよびクラス固有の情報

クラス	クラス固有の情報
AddLdapOperation	`LdapEntry`
BindLdapOperation	Bind Password
CompareLdapOperation	Attribute Name Attribute Value
DeleteLdapOperation	Delete DN
ModdnLdapOperation	New Parent DN New Relative DN Delete Old RDN New DN
ModifyLdapOperation	`LdapModification`
SearchLdapOperation	Filter Required Attributes Scope `SearchResultSet`（サーバーからは送信されません。データを戻すためにプラグインによって作成されます。）

各クラスには、情報を作成、変更および取得するためのメソッドもあります。クラス固有の情報は、プラグインへの入力またはプラグインからサーバーへの出力（あるいはその両方）を表します。

操作固有のクラスの詳細は、この項のこれ以降の部分で説明します。

AddLdapOperation ldapadd プラグインの起動時に、サーバーにより、追加するエントリの情報を渡すための `LdapEntry` オブジェクトを含む `AddLdapOperation` オブジェクトが構成されます。LdapEntry オブジェクトには、次の情報が含まれます。

- DN
- Attributes

DN は、追加するエントリの識別名を表します。Attributes は、エントリの JNDI 属性です。表 13-4 に示すように、post 操作を除くすべての操作で、プラグインによる `LdapEntry` の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-4 各プラグインのタイミングに対する LdapEntry 情報の動作

プラグインのタイミング	LdapEntry 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

BindLdapOperation サーバーにより、ldapbind プラグインに次の情報が渡されます。

- Bind Password
- Proxy Requester DN

Bind Password は、バインド用のパスワードです。Proxy Requester DN は、プロキシ・スイッチをリクエストする ID の識別名です。

CompareLdapOperation サーバーにより、ldapcompare プラグインに次の情報が渡されます。

- Attribute Name
- Attribute Value

Attribute Name は、ldapcompare 操作中に比較される名前です。表 13-5 に示すように、post 操作を除くすべての操作で、プラグインによる Attribute Name の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-5 各プラグインのタイミングに対する AttributeName の動作

プラグインのタイミング	Attribute Name 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

Attribute Value は、ldapcompare 操作中に比較される値です。表 13-6 に示すように、post 操作を除くすべての操作で、プラグインによる Attribute Value の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-6 各プラグインのタイミングに対する Attribute Value の動作

プラグインのタイミング	Attribute Value 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

DeleteLdapOperation サーバーにより、ldapdelete プラグインに Delete DN オブジェクトが渡されます。これは削除される識別名です。表 13-7 に示すように、post 操作を除くすべての操作で、プラグインによる Delete DN の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-7 各プラグインのタイミングに対する Delete DN の動作

プラグインのタイミング	Delete DN 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

ModdnLdapOperation サーバーにより、ldapmoddn プラグインに次の情報が渡されます。

- New Parent DN
- New Relative DN
- Delete Old RDN
- New DN

New Parent DN には、`PluginDetail` の `LdapBaseEntry` に指定された RDN の新規の親が含まれます。表 13-8 に示すように、post 操作を除くすべての操作で、プラグインによる New Parent DN の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-8 各プラグインのタイミングに対する New Parent DN 情報の動作

プラグインのタイミング	New Parent DN 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

New Relative DN は、PluginDetail の LdapBaseEntry に指定された RDN を置き換える新規の RDN です。表 13-9 に示すように、post 操作を除くすべての操作で、プラグインによる New Relative DN の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-9 各プラグインのタイミングに対する New Relative DN 情報の動作

プラグインのタイミング	New Relative DN 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

Delete Old RDN 値には、PluginDetail の LdapBaseEntry に指定されている古い RDN を、新規の相対識別名に置き換えた後も保存するかどうかを指定します。表 13-10 に示すように、post 操作を除くすべての操作で、プラグインによる Delete Old RDN の値の変更が可能で、その情報はサーバーに戻されます。

表 13-10 各プラグインのタイミングに対する Delete Old RDN 情報の動作

プラグインのタイミング	Delete Old RDN 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

New DN には、ldapmoddn 操作のターゲット識別名を指定します。この情報は、サーバーからプラグインへの入力のみです。プラグインがこの情報を変更してサーバーに戻すことはできません。

ModifyLdapOperation サーバーにより、ldapmodify プラグインに LdapModification オブジェクトが渡されます。LdapModification オブジェクトには、JNDI 変更アイテムである Modification Items が含まれます。表 13-11 に示すように、post 操作を除くすべての操作で、プラグインによる LdapModification の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-11 各プラグインのタイミングに対する LdapModification 情報の動作

プラグインのタイミング	LdapModification 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

SearchLdapOperation `SearchLdapOperation` オブジェクトには、次の情報が含まれます。

- Filter
- Required Attributes
- Scope
- `SearchResultSet`

Filter、Required Attributes および Scope はサーバーに渡されます。

Filter には、ldapsearch 操作に指定された LDAP 検索フィルタが含まれます。これは、プラグインへの入力のみです。プラグインがこの情報を変更してサーバーに戻すことはできません。

Required Attributes には、ldapsearch 操作に指定された必須属性が含まれます。表 13-12 に示すように、post 操作を除くすべての操作で、プラグインによる Required Attributes の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-12 各プラグインのタイミングに対する Required Attributes の動作

プラグインのタイミング	Required Attributes 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

Scope には、ldapsearch 操作により実行される検索の範囲が含まれます。表 13-13 に示すように、post 操作を除くすべての操作で、プラグインによる Scope の情報の変更が可能で、その情報はサーバーに戻されます。

表 13-13 各プラグインのタイミングに対する Scope の動作

プラグインのタイミング	Scope 情報の動作
pre when when_replace	入力および出力の両方。プラグインによる情報の変更が可能で、その情報はサーバーに戻されます。
post	入力のみ。

`SearchResultSet` には、Java プラグインからサーバーに戻される検索結果を定義します。ldapsearch 操作を実行するプラグインは、このオブジェクトを構成できます。表 13-14 に示すように、SearchResultSet をサーバーに戻すことができるのは when および when_replace プラグインのみです。

表 13-14 各プラグインのタイミングに対する SearchResultSet の動作

プラグインのタイミング	SearchResultSet 情報の動作
pre	このプラグインはオブジェクトを戻せません。
when when_replace	このプラグインは、サーバーにオブジェクトを戻すことができます。
post	このプラグインはオブジェクトを戻せません。

PluginFlexfield

プラグインを登録すると、プラグイン構成エントリにカスタム情報を保存できます。サーバーによりプラグインが起動される際に、PluginFlexfield に含めたこの情報がプラグインに渡されます。

構成エントリにカスタム情報を保存するためのスキーマ属性は 3 つあります。orclPluginFlexfield 属性にはテキスト情報を保存できます。保存するカスタム情報をさらに詳細に記録するには、サブタイプを使用できます。たとえば、サブタイプ orclPluginFlexfield;ad-host を使用して、プラグインが接続する必要のある Active Directory サーバーのホスト名を保存できます。

orclPluginBinaryFlexfield 属性には、バイナリ値を保存できます。サーバーではバイナリ属性の属性サブタイプがサポートされていないため、プラグインごとに orclPluginBinaryFlexfield に保存できるのは 1 つの値のみです。

クリアテキストには表示しないカスタム・テキスト情報を保存するには、orclPluginSecuredFlexfield を使用できます。値は暗号化形式で保存および表示されます。ユーザーがこの属性をクリアテキストで取得できないように、Oracle Internet Directory のプライバシ・モードが有効化されていることを確認してください。『Oracle Internet Directory 管理者ガイド』の「受信した機密の属性のプライバシ」を参照してください。保存するカスタム情報をさらに詳細に記録するには、サブタイプを使用できます。orclPluginFlexfield と同じサブタイプ形式を使用してください。

サーバーによりプラグインが起動される際に、orclPluginFlexfield、orclPluginBinaryFlexfield および orclPluginSecuredFlexfield から PluginFlexfield オブジェクトに含まれた情報がプラグインに渡されます。プラグインにより情報が解析されて使用されます。プラグインは、PluginFlexfield をサーバーに戻すことはできません。

次に示す構成エントリの例では、orclPluginFlexfield のサブタイプにより、パスワードの長さは 8 文字以上で、数値が含まれている必要があり、同じ文字は繰返し使用できないことが指定されています。

```
dn: cn=pre_add replace,cn=plugin,cn=subconfigsubentry
orclPluginFlexfield;minPwdLength: 8
orclPluginFlexfield;isDigitPwd: 1
orclPluginFlexfield;isRepeatCharsPwd: 0
objectclass: orclPluginConfig
objectclass: top
orclpluginname: MyJavaPwdCheckPlugin
orclplugintype: operational
orclplugintiming: pre
orclpluginldapoperation: ldapadd
orclpluginenable: 1
orclpluginsubscriberdnlist: cn=users,dc=us,dc=oracle,dc=com
orclpluginattributelist: userpassword
orclPluginKind: Java
```

PluginResult

実行結果をサーバーに戻すため、Java プラグインにより PluginResult オブジェクトが構成され、そのオブジェクトがサーバーに渡されます。PluginResult には、LdapOperation またはその操作固有のサブクラスの 1 つの、いずれかのオブジェクトが含まれます。これらのオブジェクトは、13-6 ページの「LdapOperation」の項で説明されています。その項で説明されているように、操作およびタイミングによっては、PluginDetail から受信した LdapOperation サブクラス・オブジェクトの情報をプラグインが変更することが可能で、そのオブジェクトは PluginResult に含まれてサーバーに戻されます。

ServerPlugin インタフェース

すべての Java プラグインでは、ServerPlugin インタフェースが使用されます。このインタフェースには、サーバーと通信するための事前定義済のメソッドがあります。各 LDAP 操作およびタイミングに対して 1 つのメソッドがあります。各メソッドでは、PluginDetail オブジェクトが入力として使用され、PluginResult オブジェクトが Oracle Internet Directory サーバーに戻されます。

ServerPluginAdapter クラスは、ServerPlugin インタフェースを実装します。ServerPluginAdapter クラスには、デフォルト（NULL）で、ServerPlugin メソッドが実装されています。このクラスを使用すると、すべてのメソッドを実装しなくても Java プラグインをコーディングできます。

この項のこれ以降では、各 LDAP 操作の ServerPlugin メソッドを示します。次に示すメソッドが示されます。

- Ldapbind の ServerPlugin メソッド
- Ldapcompare の ServerPlugin メソッド
- Ldapadd の ServerPlugin メソッド
- Ldapmodify の ServerPlugin メソッド
- Ldapmoddn の ServerPlugin メソッド
- Ldapsearch の ServerPlugin メソッド
- Ldapdelete の ServerPlugin メソッド

Ldapbind の ServerPlugin メソッド

```
public PluginResult pre_bind(PluginDetail pc) throws Exception;
public PluginResult when_bind_replace(PluginDetail pc) throws Exception;
public PluginResult post_bind(PluginDetail pc) throws Exception;
```

Ldapcompare の ServerPlugin メソッド

```
public PluginResult pre_compare(PluginDetail pc) throws Exception;
public PluginResult when_compare_replace(PluginDetail pc) throws Exception;
public PluginResult post_compare(PluginDetail pc) throws Exception;
```

Ldapadd の ServerPlugin メソッド

```
public PluginResult pre_add(PluginDetail pc) throws Exception;
public PluginResult when_add(PluginDetail pc) throws Exception;
public PluginResult when_add_replace(PluginDetail pc) throws Exception;
public PluginResult post_add(PluginDetail pc) throws Exception;
```

Ldapmodify の ServerPlugin メソッド

```
public PluginResult pre_modify(PluginDetail pc) throws Exception;
public PluginResult when_modify(PluginDetail pc) throws Exception;
public PluginResult when_modify_replace(PluginDetail pc) throws Exception;
public PluginResult post_modify(PluginDetail pc) throws Exception;
```

Ldapmoddn の ServerPlugin メソッド

```
public PluginResult pre_moddn(PluginDetail pc) throws Exception;
public PluginResult when_moddn(PluginDetail pc) throws Exception;
public PluginResult when_moddn_replace(PluginDetail pc) throws Exception;
public PluginResult post_moddn(PluginDetail pc) throws Exception;
```

Ldapsearch の ServerPlugin メソッド

```
public PluginResult pre_search(PluginDetail pc) throws Exception;
public PluginResult when_search(PluginDetail pc) throws Exception;
public PluginResult when_search_replace(PluginDetail pc) throws Exception;
public PluginResult post_search(PluginDetail pc) throws Exception;
```

Ldapdelete の ServerPlugin メソッド

```
public PluginResult pre_delete(PluginDetail pc) throws Exception;
public PluginResult when_delete(PluginDetail pc) throws Exception;
public PluginResult when_delete_replace(PluginDetail pc) throws Exception;
public PluginResult post_delete(PluginDetail pc) throws Exception;Java plug-in AP
```

Java プラグイン・エラーおよび例外処理

すべての未処理例外は、プラグインの実行中に Oracle Internet Directory サーバーによって捕捉されます。例外スタック・トレースおよび各例外に関するメッセージは、サーバー・ログ・ファイルに記録されます。これらの例外は 3 つのカテゴリに分類されます。

- ランタイム・エラーおよび例外は、不適切なプラグイン・コードまたはロジックが原因で発生します。`NullPointerException` を含め、すべてのランタイム・エラーおよび例外はサーバーにより捕捉され、Java プラグインの実行中に生成されます。これらのエラーおよび例外は、サーバー・ログ・ファイルに記録されます。
- プラグインによりスローされる予想される例外は、Oracle Internet Directory サーバーのログ・ファイルに記録されます。また、例外はプラグインによって捕捉され、サーバー・ログ・ファイルに記録するためにサーバーにスローされます。
- プラグインでは、`PluginException` クラスを使用してエラーを生成できます。`PluginException` オブジェクトを使用してサーバーに渡されるエラー・メッセージ、またはそのサブクラスは、LDAP クライアントに渡されます。このメッセージは、例外スタック・トレースおよびメッセージとともに、サーバーによりサーバー・ログ・ファイルに記録されます。

この項では、3 つの例を説明します。その内容は、次のとおりです。

- ランタイム例外の例
- ランタイム・エラーの例
- PluginException の例

ランタイム例外の例

次に、プラグインの実行中に生成される典型的な例外のログ・エントリを示します。

```
….
06:17:03 *
    ERROR * gslpg_exceptionHndlr * Exception Message : Error
    ERROR * gslpg_exceptionHndlr * Exception Stack Trace :
              MyCompareJavaPlugin.post_compare(Prog2.java:75)
END

BEGIN
2004/10/19:01:52:13 *
    ServerWorker (REG):4 * ConnID:0 * OpID:1 * OpName:compare
    ERROR * gslpg_exceptionHndlr * Exception Stack Trace :
               java.lang.NullPointerException
              java.util.Hashtable.put(Hashtable.java:393)
               oracle.ldap.ospf.PluginDetail.put(PluginDetail.java:41)
END
```

ランタイム・エラーの例

このエラーは、プラグイン MyJavaPlugin が $ORACLE_HOME/ldap/server/plugin ディレクトリに存在しなかったために発生しました。ログ・ファイル・エントリは次のようになります。

```
BEGIN
2004/10/19:01:52:13 *
    ServerWorker (REG):4 * ConnID:0 * OpID:1 * OpName:compare
    ERROR * gslpg_exceptionHndlr * Exception Stack Trace :
              java.lang.NoClassDefFoundError: MyJavaPlugin
END
```

PluginException の例

PluginException オブジェクトがサーバーにスローされると、Oracle Internet Directory サーバーにより、その他のエラー・メッセージとともに、標準のプラグイン・エラー・メッセージが LDAP クライアントに戻されます。次に、LDAP クライアントにより表示されるエラーを示します。

```
ldap_compare: UnKnown Error Encountered
ldap_compare: additional info: Error Message returned by the Java Plug-in
```

Java プラグインのデバッグおよびロギング

プラグインでは、独自のログ・ファイルを保持し、そのファイルにリアルタイムで記録することが可能です。また、プラグインでは、ServerLog クラスを使用して、実行中に Oracle Internet Directory サーバーのログ・ファイルにデバッグ・メッセージを記録できます。ServerLog クラスにメッセージを記録するためのメソッドは、次のようになります。

```
public static void log(String message);
```

ServerLog.log() メソッドにより記録されたメッセージは、先頭に次の文字列が付きます。

```
* Server Java Plug-in *
```

次に例を示します。

```
2006/05/11:01:11:28 * ServerWorker (REG):7
  ConnID:241 * mesgID:2 * OpID:1 * OpName:bind
01:11:28 * Server Java Plug-in * MESSAGE FROM PLUGIN
01:11:28 * Server Java Plug-in * Bind DN :
 cn=ad_user,cn=oiddvusers,cn=oraclecontext,dc=us,dc=oracle,dc=com
```

プラグインのデバッグ・メッセージをサーバー・ログに記録するには、次に示すデバッグ・レベルのいずれかを使用して Oracle Internet Directory サーバーを起動する必要があります。

表 13-15 Java プラグインのロギングのデバッグ・レベル

Oracle Internet Directory サーバーのデバッグ・レベル	デバッグ・レベルの内容
134217728	すべての Java プラグインのデバッグ・メッセージおよび Java プラグイン・フレームワークに関連するサーバー・メッセージ
268435456	ServerLog オブジェクトを使用して Java プラグインにより渡されるすべてのメッセージ
402653184	前述の内容の両方

ServerLog.log() メソッドはスレッド・セーフです。このメソッドを実行すると、パフォーマンスが低下する場合があります。

Java プラグインの例

この項では、2 つの例を説明します。その内容は、次のとおりです。

- 例 1: パスワード検証プラグイン
- 例 2: Active Directory の外部認証プラグイン

注意： Java プラグインでは System.exit() を使用しないでください。使用すると、Oracle ディレクトリ・サーバーが予期しない動作をする可能性があります。

例 1: パスワード検証プラグイン

この例では、ldapmodify 操作の前に `userPassword` を検証する Java プラグインを説明します。`pre` Java プラグインは、Oracle Internet Directory サーバーに登録されています。プラグイン構成には、そのプラグインで検証される最低限のパスワードの長さが含まれます。この情報は、`orclPluginFlexfield` 属性を使用して、プラグイン構成エントリに登録されます。サブタイプ `minPwdLength` には、最低限の長さが指定されます。この情報は、`PluginFlexfield` を使用してプラグインに渡されます。`orclPluginName` により、Oracle Internet Directory サーバーにより起動される Java プラグインの名前が指定されます。

プラグインへの入力は `PluginDetail` で、プラグインからの出力は `PluginResult` です。

パスワード検証プラグインの構成エントリ

```
dn: cn=checkuserpassword,cn=plugin,cn=subconfigsubentry
orclPluginFlexfield;minPwdLength: 8
objectclass: orclPluginConfig
objectclass: top
orclpluginname: CheckPassword
orclplugintype: operational
orclplugintiming: pre
orclpluginldapoperation: ldapmodify
orclpluginenable: 1
orclpluginsubscriberdnlist: cn=users,dc=us,dc=oracle,dc=com
orclpluginattributelist: userPassword
orclPluginKind: Java
```

パスワード検証プラグインのコード例

```
import java.io.*;
import java.lang.*;
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;
import oracle.ldap.ospf.*;
/**
 * This PRE modify plug-in will check whether the "userPassword"
 * is greater than 8 characters in length
 */
public class CheckPassword extends ServerPluginAdapter {

    // This PRE modify plug-in takes in a PluginDetail Object
    // and returns a PluginResult Object
    public PluginResult pre_modify(PluginDetail plgObj)
       throws Exception
    {
       try {
         // Retrieve the LdapOperation Object from the PluginDetail
         ModifyLdapOperation opObj = (ModifyLdapOperation)
```

```
plgObj.getLdapOperation();


         // Retrieve the LdapModification Object from the LdapOperation
            LdapModification modObj = opObj.getLdapModification();



         // Retrieve the PluginFlexfield Object from the PluginDetail
         PluginFlexfield flxFldObj = plgObj.getPluginFlexfield();

         // Retrieve the custom information from the PluginFlexfield
         // Get the minimum password length
 String passwdlength =
            flxFldObj.getFlexfield("minPwdLength");

        // Create a Result Object to return to the OID server
           PluginResult plgResObj = new PluginResult();

         // Check if the LdapModification Object is a NULL
         // set the appropriate error and error message
         if (modObj==null)
         {
            throw new PluginException("CheckPassword Plug-in Execution
Error");
         }

          // Retrieve the "userPassword" Attribute Value
         ModificationItem modItem = modObj.getModificationItemAt(0);
         BasicAttribute attr = (BasicAttribute)modItem.getAttribute();
         String attrval = null;
          if ((attr.getID()).equals("userpassword"))
     attrval = attr.get(0);

         // Check for the password length and set appropriate error
         // and error message
          if (attrval.length() < Integer.parseInt(passwdlength))
         {
             throw new PluginException("userPassword is less than 8
characters");
         }

        // Return the PluginResult Object to the OID Server
        return plgResObj;
    }
    // Catch any unexpected exception which may occur and throw
    // it back to the OID server to log it
    catch (Exception e)
    {
        throw e;
    }
  }
}
```

例 2: Active Directory の外部認証プラグイン

この例では、Active Directory の外部認証プラグインを説明します。クライアントが userPassword の ldapcompare 操作をリクエストすると、サーバーによりこの Java プラグインが起動され、Active Directory に対してユーザーが認証されます。

外部認証プラグインの構成エントリ

```
dn: cn=when_rep_comp,cn=plugin,cn=subconfigsubentry
orclpluginsubscriberdnlist: cn=users,dc=us,dc=oracle,dc=com;
orclpluginflexfield;ad-host: dlin-pc2.us.oracle.com
orclpluginflexfield;ad-port: 389
orclpluginflexfield;ad-su-dn: administrator@dlin.net
orclpluginflexfield;ad-su-passwd: welcome1
objectclass: orclPluginConfig
objectclass: top
orclpluginname: ExtAuthAD
orclplugintype: operational
orclplugintiming: when
orclpluginisreplace: 1
orclpluginldapoperation: ldapcompare
orclpluginversion: 1.0.1
cn: when_rep_comp
orclpluginkind: Java
orclpluginenable: 1
```

外部認証プラグインのコード

```
public class ExtAuthAD extends ServerPluginAdapter {

public PluginResult when_compare_replace(PluginDetail plgObj)
   throws Exception {
   try {

       // Retrieve the LdapOperation from the PluginDetail
      LdapOperation opObj = (CompareLdapOperation) plgObj.getLdapOperation();


       // Retrieve the Base DN, Attribute and Attribute Value
      String bdn = opObj.getBaseDN().substring(0,
             opObj.getBaseDN().lastIndexOf("cn=users,dc=us,dc=oracle,dc=com")-1)
             +",cn=users,dc=dlin,dc=net";
       String ban = opObj.getAttributeName();
       String bav = opObj.getAttributeValue();

       // Retrieve the AD Information from the PluginFlexfield
       PluginFlexfield flxObj = plgObj.getPluginFlexfield();
       String adhost = flxObj.getFlexfield("ad-host");
       String adport = flxObj.getFlexfield("ad-port");
       String adsudn = flxObj.getFlexfield("ad-su-dn");
       String adsupasswd = flxObj.getFlexfield("ad-su-passwd");


       // Create a PluginResult Object to return to the OID server
       PluginResult plgResObj = new PluginResult();


       // Create a Hashtable with values required to connect to AD
       Hashtable env = new Hashtable();

       env.put(Context.INITIAL_CONTEXT_FACTORY,
              "com.sun.jndi.ldap.LdapCtxFactory");
       env.put(Context.PROVIDER_URL, "ldap://"+adhost+":"+adport);
```

```
        env.put(Context.SECURITY_AUTHENTICATION, "simple");
        env.put(Context.SECURITY_PRINCIPAL, bdn);
        env.put(Context.SECURITY_CREDENTIALS, bav);

        // Try to connect to AD
        DirContext dirContext = null;
        try {
          dirContext = new InitialDirContext(env);
          if (dirContext != null) {
              // User has been successfully authenticated, add the appropriate
              // result code to the LdapOperation
              opObj.setOperationResultCode(6);
          }
        }
        catch(NamingException ne) {
         // Unable to connect to the AD directory server with the given
         // credentials, add the appropriate result code to the LdapOperation
          opObj.setOperationResultCode(5);
        }

        // Add the LdapOperation to the PluginResult
        plgResObj.addLdapOperation(opObj);

        // Return the PluginResult
        return plgResObj;

    } catch(Exception e) {
       // In case of any unexpected errors in the plug-in, throw the Exception
       // back to the OID server to log it
       throw e;
     }
    }
  }
```

第III部

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

第 III 部では、標準 API と標準 API に対する Oracle の拡張機能について説明します。第 III 部は、次の章で構成されています。

- 第 14 章「C API リファレンス」
- 第 15 章「DBMS_LDAP PL/SQL リファレンス」
- 第 16 章「Java API リファレンス」
- 第 17 章「DBMS_LDAP_UTL PL/SQL リファレンス」
- 第 18 章「DAS_URL インタフェース・リファレンス」
- 第 19 章「Oracle Directory Integration Platform ユーザー・プロビジョニング Java API リファレンス」
- 第 20 章「Oracle Directory Integration Platform PL/SQL API リファレンス」

14

C API リファレンス

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

この章では、次の項目について説明します。

- Oracle Internet Directory C API の概要
- C API のファンクション
- C API の使用例
- C API に必要なヘッダー・ファイルおよびライブラリ
- 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*（10.1.4.0.1）は、次のモードで使用できます。

- SSL モード : SSL を使用してすべての通信を保護します。
- 非 SSL モード : クライアント / サーバー間の通信を保護しません。

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、14-41 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
- ファンクションの概要

Oracle Internet Directory SDK C API SSL 拡張機能

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

認証には次の 3 つのモードがあります。

- 認証なし : クライアントとサーバーのどちらも認証せず、SSL による暗号化のみを使用します。
- サーバー認証 : クライアントによりサーバーのみが認証されます。
- 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が相互に認証します。

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

次のコールを行うのみで、SSL は使用可能になります。

```
int ldap_init_SSL(Sockbuf *sb, char *sslwallet, char *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 14-1 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。

表 14-1 SSL インタフェース・コールの引数（続き）

引数	説明
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： 14-41 ページの「C API の使用例」

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。10*g*（10.1.4.0.1）の API 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 のファンクション

この項では、C API のファンクションおよびプロシージャを 1 つずつ取り上げます。それぞれの目的および構文について説明し、使用方法のヒントも示します。

この項の項目は次のとおりです。

- ファンクションの概要
- LDAP セッションの初期化
- LDAP セッション・ハンドル・オプション
- ディレクトリに対する認証
- Oracle の拡張機能を使用した SASL 認証
- コントロールの使用
- セッションのクローズ
- LDAP 操作の実行
- 操作の中止
- 結果の取得と LDAP メッセージの確認
- エラーの処理と結果の解析
- 結果リストの参照
- 検索結果の解析

ファンクションの概要

表 14-2 に、C API のファンクションおよびプロシージャをすべて示し、それぞれの目的を簡単に説明します。

表 14-2 C API のファンクションおよびプロシージャ

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 ldap_abandon	非同期操作の取消し
ldap_add_ext ldap_add_ext_s ldap_add ldap_add_s	ディレクトリに新規エントリを追加
ldap_compare_ext ldap_compare_ext_s ldap_compare ldap_compare_s	ディレクトリ内のエントリの比較
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 ldap_delete_ext_s ldap_delete ldap_delete_s	ディレクトリからのエントリの削除
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 ldap_explode_rdn	識別名を構成要素に分割
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
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

表 14-2 C API のファンクションおよびプロシージャ（続き）

ファンクションまたはプロシージャ	説明
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_first_attribute 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
ldap_result ldap_msgfree ldap_msgtype ldap_msgid	非同期操作の結果のチェック
ldap_sasl_bind ldap_sasl_bind_s	LDAP サーバーへの一般認証
ldap_search_ext ldap_search_ext_s ldap_search ldap_search_s	ディレクトリの検索
ldap_search_st	タイムアウト値でのディレクトリの検索
ldap_get_option 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_simple_bind_s ldap_sasl_bind ldap_sasl_bind_s	LDAP サーバーへの簡易認証
ldap_unbind_ext ldap_unbind ldap_unbind_s	LDAP セッションの終了
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 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。

関連資料： 各コールの詳細は、次の URL を参照してください。

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

LDAP セッションの初期化

この項のコールは、LDAP サーバーとのセッションを初期化します。

ldap_init および ldap_open

ldap_init() は、LDAP サーバーとのセッションを初期化しますが、接続はオープンしません。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。ldap_open() は、セッションを初期化して接続をオープンします。この 2 つは目的も構文も同じですが、前者の使用をお薦めします。

構文

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

パラメータ

表 14-3 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	接続先の LDAP サーバーを実行しているホストの IP アドレスを示す、空白で区切られたホスト名またはドット表記の文字列のリストが入ります。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。ホスト名とポート番号はコロンで区切る必要があります。接続に成功するまで、ホストがリストの順序に従って試されます。 **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。hostname にポート番号が含まれている場合、portno 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。これは、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。セッションが初期化できない場合、オペレーティング・システムのエラー・レポート・メカニズムをチェックし、コールに失敗した理由を確認してください。

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 および 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リクエスト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 14-4 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 14-4 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。
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、14-8 ページの表 14-5 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 16 進値をカッコで囲んで記述します。

表 14-4 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
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

14-8 ページの表 14-5 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 14-5 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)	非適用。オプションは読取り専用です。	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。このオプションは読取り専用で、設定はできません。
ORA_LDAP_OPT_RFRL_CACHE	void* (LDAP_OPT_ON void* (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。
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。

表 14-5 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_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。この値は検索リクエスト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) void *(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 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）値はオフ、0（ゼロ）以外の値はオンを意味します。このオプションはデフォルトでオンになっています。
LDAP_OPT_RESTART(0X09)	void * (LDAP_OPT_ON) void * (LDAP_OPT_OFF)	int *	LDAP の入出力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。LDAP_OPT_ON または LDAP_OPT_OFF 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）値はオフ、0（ゼロ）以外の値はオンを意味します。このオプションは、タイマーの停止などによって、入出力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。このオプションはデフォルトでオフになっています。
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。LDAP_VERSION2（2）または LDAP_VERSION3（3）に設定します。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***	各リクエスト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：**14-15 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：**14-15 ページの「コントロールの使用」を参照してください。

表 14-5 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_API_FEATURE_INFO (0x15)	非適用。オプションは読取り専用です。	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。このオプションは読取り専用です。設定はできません。
LDAP_OPT_HOST_NAME(0x30)	char *	char **	プライマリ LDAP サーバーのホスト名（またはホストのリスト）です。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 *	-	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 */
```

ディレクトリに対する認証

この項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 および 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識別子（OID）のドット表記の文字列および資格証明を保持している struct berval 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リクエストできます。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 14-6 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 14-6 ディレクトリに対する認証のパラメータ

パラメータ	説明
ld	セッション・ハンドルです。
dn	バインドするエントリの名前です。
mechanism	LDAP_SASL_SIMPLE（NULL）を指定して簡易認証を取得するか、SASL メソッドを識別するテキスト文字列を指定します。
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リクエストのメッセージ ID に設定されます。
servercredp	相互認証が指定されている場合は、この結果パラメータにサーバーから戻された資格証明が入力されます。割り当てられた berval 構造体が戻されるため、ber_bvfree() をコールして解放する必要があります。このフィールドを無視する場合は、NULL を渡す必要があります。

使用方法

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

ldap_sasl_bind() ファンクションは、非同期のバインド操作を開始し、リクエスト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() によってリクエストのメッセージ ID が *msgidp に格納されます。続けて ldap_result() 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。続けて ldap_result() 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。

続けてバインド・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、「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

この項の項目は次のとおりです。

- ora_ldap_init_SASL
- ora_ldap_create_cred_hdl、ora_ldap_set_cred_props、ora_ldap_get_cred_props および ora_ldap_free_cred_hdl

ora_ldap_init_SASL

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、入力引数の 1 つに指定したメカニズムに基づいて認証を実行します。

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。

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 14-7 ora_ldap_init_SASL() に渡されるパラメータ

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	認証されるユーザー 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。

cred パラメータは、ユーザー用の SASL 証明書ハンドルです。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。

サポートされる SASL メカニズムは次のとおりです。

- DIGEST-MD5

  Oracle Internet Directory 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。これにより、`ora_ldap_init_SASL()` ファンクションは、`mechanism` 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。

ora_ldap_create_cred_hdl、ora_ldap_set_cred_props、ora_ldap_get_cred_props および ora_ldap_free_cred_hdl

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` 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()` ファンクションは、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                  String[],
        void             *   inProperty
);
OraLdapHandle ora_ldap_get_cred_props
(
        OraLdapClientCtx *   clientCtx,
        OraLdapHandle   cred,
        int                   String[],
        void                 *   outProperty
);

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

パラメータ

表 14-8 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。
String[]	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。

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 14-9 に、ldapcontrol 構造体のフィールドを示します。

表 14-9 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、サーバーからのレスポンスはありません。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。

関連項目： コントロールの詳細は、第 3 章「LDAP プロトコルに対する拡張機能」を参照してください。

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（14-7 ページの「ldap_get_option および 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。

この章では、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）。

クライアント・コントロールの参照プロセス　14-7 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、リクエスト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 のオブジェクト識別子（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、値 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。

関連資料： オブジェクト識別子の詳細は、『Oracle Internet Directory 管理者ガイド』の「ディレクトリ・スキーマの管理」を参照してください。

セッションのクローズ

この項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、ldap_unbind_ext および ldap_unbind_s

ldap_unbind_ext()、ldap_unbind() および ldap_unbind_s() は、サーバーにバインド解除リクエストを送信し、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レスポンス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（リクエストが 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、バインド解除リクエストに対するサーバーからのレスポンスがないため、バインド解除リクエストで送信されたサーバー・コントロールに対するレスポンスを受信する方法はないことに注意してください。

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 14-10 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。

LDAP 操作の実行

この項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リクエストされた属性セットを戻します。

ldap_search_ext、ldap_search_ext_s、ldap_search および ldap_search_s

ldap_search_ext() ファンクションは、非同期の検索操作を開始し、リクエスト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() によってリクエストのメッセージ ID が *msgidp に格納されます。続けて ldap_result() 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、続けて ldap_result() 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、「エラーの処理と結果の解析」を参照してください。

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 14-11 に、検索操作のパラメータを示します。

表 14-11 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。
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リクエスト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）がリクエスト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。

表 14-11 検索操作のためのパラメータ（続き）

パラメータ	説明
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。
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リクエストのメッセージ 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。その内容は、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。

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 パラメータには、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。パラメータ 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 および ldap_compare_s

これらのルーチンを使用して、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。

ldap_compare_ext() ファンクションは、非同期の比較操作を開始し、リクエスト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() によってリクエストのメッセージ ID が *msgidp に格納されます。続けて ldap_result() 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、続けて ldap_result() 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、「エラーの処理と結果の解析」を参照してください。

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 14-12 に、比較操作のパラメータを示します。

表 14-12 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、struct berval 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リクエストのメッセージ ID に設定されます。

ldap_modify_ext、ldap_modify_ext_s、ldap_modify および 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。

ldap_modify_ext() ファンクションは、非同期の変更操作を開始し、リクエスト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() によってリクエストのメッセージ ID が *msgidp に格納されます。続けて ldap_result() 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、続けて ldap_result() 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、「エラーの処理と結果の解析」を参照してください。

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 14-13 に、変更操作のパラメータを示します。

表 14-13 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リクエストのメッセージ ID に設定されます。

表 14-14 に、LDAPMod 構造体のフィールドを示します。

表 14-14 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_rename および 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。

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、リクエスト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() によってリクエストの識別名メッセージ ID が *msgidp に格納されます。続けて ldap_result() 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、「エラーの処理と結果の解析」を参照してください。

構文

```
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 14-15 に、名前の変更操作のパラメータを示します。

表 14-15 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。

表 14-15 名前の変更操作のためのパラメータ（続き）

パラメータ	説明
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リクエストのメッセージ ID に設定されます。

ldap_add_ext、ldap_add_ext_s、ldap_add および 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。

ldap_add_ext() ファンクションは、非同期の追加操作を開始し、リクエスト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() によってリクエストのメッセージ ID が *msgidp に格納されます。続けて ldap_result() 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、続けて ldap_result() 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、「エラーの処理と結果の解析」を参照してください。

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 14-16 に、追加操作のパラメータを示します。

表 14-16 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リクエストのメッセージ 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_delete_ext、ldap_delete_ext_s、ldap_delete および 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。

ldap_delete_ext() ファンクションは、非同期の削除操作を開始し、リクエスト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() によってリクエストのメッセージ ID が *msgidp に格納されます。続けて ldap_result() 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、続けて ldap_result() 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、「エラーの処理と結果の解析」を参照してください。

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 14-17 に、削除操作のパラメータを示します。

表 14-17 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リクエストのメッセージ 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_extended_operation および 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。

ldap_extended_operation() ファンクションは、非同期の拡張操作を開始し、リクエスト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() によってリクエストのメッセージ ID が *msgidp に格納されます。続けて ldap_result() 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オブジェクト識別子（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、「エラーの処理と結果の解析」を参照してください。

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 14-18 に、拡張操作のパラメータを示します。

表 14-18 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	リクエストを指定する、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。

表 14-18 拡張操作のためのパラメータ（続き）

パラメータ	説明
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リクエストのメッセージ 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、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コピーに設定されます。struct berval 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。

操作の中止

この項のファンクションを使用して、進行中の操作を中止します。

ldap_abandon_ext および ldap_abandon

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レスポンスはありません。

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 14-19 に、操作を中止するためのパラメータを示します。

表 14-19 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	中止するリクエストのメッセージ 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。

関連項目： エラーとその解析方法の詳細は、「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

この項のファンクションを使用して、非同期で開始した操作の結果を戻します。これらのファンクションは、メッセージを型および ID で識別します。

ldap_result、ldap_msgtype および 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、単一のリクエスト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 リクエスト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

14-31 ページの表 14-20 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 14-20 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。
`all`	1 回の `ldap_result()` コールで取得するメッセージの数を指定します。このパラメータは、検索結果を取得する場合にのみ使用されます。定数 `LDAP_MSG_ONE (0x00)` を渡して、一度に 1 つのメッセージを取得します。すべての結果が 1 つの結果セットとして戻される前に、すべての検索結果を取得するには、`LDAP_MSG_ALL (0x01)` を渡します。すでに取得したすべてのメッセージを結果セットに戻すには、`LDAP_MSG_RECEIVED (0x02)` を渡します。

表 14-20 結果の取得と LDAP メッセージの確認のためのパラメータ（続き）

パラメータ	説明
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

この項のファンクションを使用して、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_result、ldap_parse_sasl_bind_result、ldap_parse_extended_result および ldap_err2string

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。

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。

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 14-21 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 14-21 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、リクエスト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、リクエスト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リクエスト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。
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、拡張操作のレスポンス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。

表 14-21 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
retdatap	拡張操作の結果に関するこの結果パラメータには、拡張操作のレスポンス・データが含まれる struct berval 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固有のパラメータについては、RFC 1823 を参照してください。

結果リストの参照

この項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 および ldap_next_message

検索操作の結果セットには、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 14-22 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 14-22 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

この項のファンクションを使用して、ldap_search() 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 および 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 14-23 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 14-23 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_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。

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_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 および ldap_next_attribute

この項のファンクションを使用して、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。

構文

```
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 14-24 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 14-24 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、エラーが発生した場合に、NULL を戻します。後者の場合は、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 および 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 14-25 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 14-25 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 および 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 14-26 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 14-26 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。
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、識別名解析エラーが発生すると NULL を戻します。このファンクションは、エラーを示すためにセッション・ハンドル ld のエラー・パラメータを設定します。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 14-27 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 14-27 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() を使用して、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 14-28 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 14-28 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、このルーチンはリクエスト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 <ldap.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 <ldap.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 :    Currently 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 the directory 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 に必要なヘッダー・ファイルおよびライブラリ

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・ファイルをインクルードします。
- 次の場所にあるライブラリに動的にリンクします。
  - $ORACLE_HOME/lib/libclntsh.so.10.1（UNIX オペレーティング・システムの場合）
  - %ORACLE_HOME%¥bin¥oraldapclnt10.dll（Windows オペレーティング・システムの場合）

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。

> **関連資料：** 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。

C API に入力されるすべての文字列は UTF-8 形式の必要があります。文字列が UTF-8 形式でない場合は、OCI ファンクション `OCINlsCharSetConvert` を使用して変換できます。`http://www.oracle.com/technology/documentation` の Oracle Database ライブラリにある『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

15

DBMS_LDAP PL/SQL リファレンス

DBMS_LDAP には、PL/SQL プログラマが LDAP サーバーのデータにアクセスするためのファンクションとプロシージャが含まれています。この章では、すべての API ファンクションの詳細を説明します。

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
- 例外の概要
- データ型の概要
- サブプログラム

注意： DBMS_LDAP パッケージのサンプル・コードは、次の URL で入手できます。

http://www.oracle.com/technology/sample_code/

「Sample Applications-Fusion Middleware」の下の「Oracle Identity Management」リンクを探してください。

サブプログラムの概要

表 15-1 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、検索リクエスト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。
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、検索リクエスト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。
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。

表 15-1 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
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。
populate_mod_array プロシージャ（バイナリ・バージョン。BLOB データ型を使用）	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。
get_values_blob ファンクション	get_values_blob() 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より大きな値を取り出せます。
count_values_blob ファンクション	DBMS_LDAP.get_values_blob() によって戻された値の数をカウントします。
value_free_blob ファンクション	DBMS_LDAP.get_values_blob() によって戻された BLOB_COLLECTION に関連付けられているメモリーを解放します。
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、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。
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。
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。

表 15-1 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
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 UTF-8 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	UTF-8 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() の詳細は、第 2 章の「ディレクトリの検索」を参照してください。
- DBMS_LDAP.unbind_s() の詳細は、第 2 章の「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、15-4 ページの表 15-2 に示す例外が生成される場合があります。

表 15-2 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。エラー文字列では、ユーザーが使用している言語で問題が説明されます。
init_failed	31203	DBMS_LDAP.init() に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() でリクエスト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、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 15-2 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	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、無効な相対識別名を指定した場合に呼び出されます。
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、表 15-3 に示すデータ型を使用します。

表 15-3 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。

表 15-3 DBMS_LDAP データ型の概要（続き）

データ型	用途
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。
BLOB_COLLECTION	バイナリ・データを表す BLOB データのリストを保持するために使用します。

サブプログラム

この項では、DBMS_LDAP の各サブプログラムについてさらに詳しく説明します。

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 15-4 INIT ファンクションのパラメータ

パラメータ	説明
hostname	接続先の LDAP サーバーを実行しているホストの IP アドレスを示す、空白で区切られたホスト名またはドット表記の文字列のリストが入ります。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。ホスト名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 15-5 INIT ファンクションの戻り値

値	説明
SESSION	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 15-6 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、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。このルーチンは、セッションが初期化できない場合に NULL を戻し、INIT_FAILED 例外を呼び出します。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 15-7 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 15-8 SIMPLE_BIND_S ファンクションの戻り値

値	説明
PLS_INTEGER	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 15-9 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 15-10 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	ユーザーの識別名です。
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。有効な値は DBMS_LDAP_UTL.AUTH_SIMPLE のみです。

戻り値

表 15-11 BIND_S ファンクションの戻り値

値	説明
PLS_INTEGER	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 15-12 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	リクエスト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 OUT SESSION
)
RETURN PLS_INTEGER;
```

パラメータ

表 15-13 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 15-14 UNBIND_S ファンクションの戻り値

値	説明
PLS_INTEGER	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 15-15 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リクエストが送信され、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 が無効になります。

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 15-16 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 15-17 COMPARE_S ファンクションの戻り値

値	説明
PLS_INTEGER	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 15-18 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、ディレクトリ内の属性が特定の値を持つことを確認できます。この操作は、比較が可能な構文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、ディレクトリ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、検索リクエスト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 15-19 SEARCH_S ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
base	検索の開始点となるエントリの識別名です。
scope	SCOPE_BASE (0x00)、SCOPE_ONELEVEL (0x01) または SCOPE_SUBTREE (0x02) のいずれかで、検索範囲を指定します。
filter	検索フィルタを表す文字列です。この値を NULL にすると、すべてのエントリに一致するフィルタ "(objectclass=*)" を使用するように指定できます。
attrs	一致した各エントリのどの属性を戻すかを指定する文字列の集合です。このパラメータを NULL にすると、取得可能なユーザー属性がすべて取り出されます。文字列 NO_ATTRS ("1.1") を配列内の唯一の文字列として使用すると、サーバーが属性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 15-20 SEARCH_S ファンクションの戻り値

値	説明
PLS_INTEGER	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	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 15-21 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、属性および値は、この章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。

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、検索リクエスト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。

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 15-22 SEARCH_ST ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドルです。
base	検索の開始点となるエントリの識別名です。
scope	SCOPE_BASE (0x00)、SCOPE_ONELEVEL (0x01) または SCOPE_SUBTREE (0x02) のいずれかで、検索範囲を指定します。
filter	検索フィルタを表す文字列です。この値を NULL にすると、すべてのエントリに一致するフィルタ "(objectclass=*)" を使用するように指定できます。

表 15-22 SEARCH_ST ファンクションのパラメータ（続き）

パラメータ	説明
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 15-23 SEARCH_ST ファンクションの戻り値

値	説明
PLS_INTEGER	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	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 15-24 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 15-25 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 15-26 FIRST_ENTRY 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 15-27 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 15-28 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 15-29 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 15-30 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 15-31 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 15-32 COUNT_ENTRY ファンクションの戻り値

値	説明
PLS_INTEGER	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 15-33 COUNT_ENTRY ファンクションの例外

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

使用方法

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

関連項目

DBMS_LDAP.first_entry()、DBMS_LDAP.next_entry()。

first_attribute ファンクション

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

構文

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

パラメータ

表 15-34 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	エントリ内の読取り済の属性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 15-35 FIRST_ATTRIBUTE ファンクションの戻り値

値	説明
VARCHAR2	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 15-36 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取り出されます。

構文

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

パラメータ

表 15-37 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	エントリ内の読取り済の属性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 15-38 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 15-39 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,
ldapentrymsg IN MESSAGE
)
RETURN VARCHAR2;
```

パラメータ

表 15-40 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 15-41 GET_DN ファンクションの戻り値

値	説明
VARCHAR2	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 15-42 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 15-43 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 15-44 GET_VALUES ファンクションの戻り値

値	説明
STRING_COLLECTION	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 15-45 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 15-46 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 15-47 GET_VALUES_LEN ファンクションの戻り値

値	説明
BINVAL_COLLECTION	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 15-48 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、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 15-49 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 15-50 DELETE_S ファンクションの戻り値

値	説明
PLS_INTEGER	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 15-51 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、ディレクトリ情報ツリー内のリーフ・エントリのみを削除できます。リーフ・エントリとは、下位エントリを持たないエントリです。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 15-52 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 15-53 MODRDN2_S ファンクションの戻り値

値	説明
PLS_INTEGER	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 15-54 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。同じ目的を実現する 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 15-55 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 15-56 ERR2STRING ファンクションの戻り値

値	説明
VARCHAR2	各国語へ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

例外は呼び出されません。

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。

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 15-57 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 15-58 CREATE_MOD_ARRAY ファンクションの戻り値

値	説明
MOD_ARRAY	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

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、DBMS_LDAP.free_mod_array をコールしてメモリーを解放します。

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 15-59 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。

例外

表 15-60 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

このプロシージャは、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します。

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 15-61 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。

例外

表 15-62 POPULATE_MOD_ARRAY（バイナリ・バージョン）プロシージャの例外

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

使用方法

このプロシージャは、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します。

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。BLOB データ型を使用）

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します。

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.BLOB_COLLECTION
);
```

パラメータ

表 15-63 POPULATE_MOD_ARRAY（バイナリ）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バイナリ属性値を指定します。

例外

表 15-64 POPULATE_MOD_ARRAY（バイナリ）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

このプロシージャは、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します。

関連項目

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

get_values_blob ファンクション

get_values_blob() 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より大きな値を取り出せます。

構文

```
Syntax
FUNCTION get_values_blob
(
ld IN SESSION,
ldapentry IN MESSAGE,
attr IN VARCHAR2
)
RETURN BLOB_COLLECTION;
```

パラメータ

表 15-65 GET_VALUES_BLOB 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 15-66 get_values_blob の戻り値

値	説明
BLOB_COLLECTION	指定した属性のすべての値を含む PL/SQL BLOB 型データの集合です。
NULL	指定した属性に関連する値がありません。

例外

表 15-67 get_values_blob 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_blob() 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_blob()、DBMS_LDAP.get_values()。

count_values_blob ファンクション

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

構文

```
FUNCTION count_values_blob
(
values IN DBMS_LDAP.BLOB_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 15-68 COUNT_VALUES_BLOB のパラメータ

パラメータ	説明
values	ラージ・バイナリ値の集合です。

戻り値

表 15-69 COUNT_VALUES_BLOB 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

例外は呼び出されません。

関連項目

DBMS_LDAP.count_values()、DBMS_LDAP.get_values_blob()。

value_free_blob ファンクション

DBMS_LDAP.get_values_blob() によって戻された BLOB_COLLECTION に関連付けられているメモリーを解放します。

構文

```
PROCEDURE value_free_blob
(
vals IN OUT DBMS_LDAP.BLOB_COLLECTION
);
```

パラメータ

表 15-70 VALUE_FREE_BLOB のパラメータ

パラメータ	説明
vals	DBMS_LDAP.get_values_blob() によって戻されたラージ・バイナリ値の集合です。

例外

例外は呼び出されません。

関連項目

DBMS_LDAP.get_values_blob()。

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 15-71 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、DBMS_LDAP.init() のコールが正常終了した場合に戻される LDAP セッションのハンドルです。
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 15-72 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 15-73 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 15-74 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 15-75 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 15-76 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 15-77 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、DBMS_LDAP.create_mod_array() のコールが正常終了した場合に戻される LDAP 変更構造体のハンドルです。

例外

例外は呼び出されません。

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 15-78 COUNT_VALUES ファンクションのパラメータ

パラメータ	説明
values	文字列値の集合です。

戻り値

表 15-79 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

例外は呼び出されません。

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 15-80 COUNT_VALUES_LEN ファンクションのパラメータ

パラメータ	説明
values	バイナリ値の集合です。

戻り値

表 15-81 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

例外は呼び出されません。

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 15-82 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、DBMS_LDAP.init() のコールが正常終了した場合に戻される LDAP セッションのハンドルです。
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 15-83 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 15-84 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。

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 15-85 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）以外にすると、属性タグは戻されません。

戻り値

表 15-86 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 15-87 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。

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 15-88 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、DBMS_LDAP.init() のコールが正常終了した場合に戻される LDAP セッションのハンドルです。
sslwrl	このパラメータで、Wallet の位置を指定します。サーバー、またはクライアントとサーバーの SSL 接続の場合は必須です。
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、またはクライアントとサーバーの SSL 接続の場合は必須です。
sslauth	このパラメータで、SSL 認証モードを指定します。(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 です。)

戻り値

表 15-89 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 15-90 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 15-91 MSGFREE ファンクションのパラメータ

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

戻り値

表 15-92 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

例外は呼び出されません。

関連項目

DBMS_LDAP.search_s()、DBMS_LDAP.search_st()。

ber_free ファンクション

BER ELEMENT へのハンドルに対応付けられたメモリーを解放します。

構文

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

パラメータ

表 15-93 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

値は戻されません。

例外

例外は呼び出されません。

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 UTF-8 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 15-94 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 15-95 nls_convert_to_utf8 の戻り値

値	説明
VARCHAR2	UTF-8 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 UTF-8 キャラクタ・セットのデータであると想定します。nls_convert_to_utf8() ファンクションは、データベース・キャラクタ・セットのデータを UTF-8 キャラクタ・セットのデータに変換します。

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・セットのデータ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のコレクションを UTF-8 キャラクタ・セットのデータに変換します。その後、変換したデータを戻します。

構文

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

パラメータ

表 15-96 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 15-97 nls_convert_to_utf8 の戻り値

値	説明
STRING_COLLECTION	UTF-8 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 UTF-8 キャラクタ・セットであると想定します。nls_convert_to_utf8() ファンクションは、入力データをデータベース・キャラクタ・セットから UTF-8 キャラクタ・セットに変換します。

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

UTF-8 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ます。その後、このデータを戻します。

構文

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

パラメータ

表 15-98 nls_convert_from_utf8 のパラメータ

パラメータ	説明
data_utf8	UTF-8 キャラクタ・セットのデータを指定します。

戻り値

表 15-99 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、UTF-8 キャラクタ・セット・データを戻します。
nls_convert_from_utf8() ファンクションは、出力データを UTF-8 キャラクタ・セットからデータベース・キャラクタ・セットに変換します。

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

UTF-8 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ます。その後、このデータを戻します。

構文

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

パラメータ

表 15-100 nls_convert_from_utf8 のパラメータ

パラメータ	説明
data_utf8	UTF-8 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 15-101 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、UTF-8 キャラクタ・セット・データを戻します。
nls_convert_from_utf8() は、出力データを UTF-8 キャラクタ・セットからデータベース・キャラクタ・セットに変換します。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

ありません。

戻り値

表 15-102 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()。

16

Java API リファレンス

Oracle Internet Directory の標準的な Java API は、Sun 社の Java Naming and Directory Interface（JNDI）として入手できます。JNDI は次のリンクで提供されています。

```
http://java.sun.com/products/jndi
```

標準 API に対する Oracle の拡張機能については、『Oracle Internet Directory API Reference』で説明されています。

Java API のサンプル・コードは、次の URL で入手できます。

```
http://www.oracle.com/technology/sample_code/
```

「Sample Applications-Oracle Application Server」の下の「Oracle Identity Management」リンクを探してください。

17

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
- サブプログラム
- ファンクション・リターン・コードの概要
- データ型の概要

> **注意：** DBMS_LDAP_UTL パッケージのサンプル・コードは、次の URL で入手できます。
>
> http://www.oracle.com/technology/sample_code/
>
> 「Sample Applications-Fusion Middleware」の下の「Oracle Identity Management」リンクを探してください。

サブプログラムの概要

表 17-1 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 17-2 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 17-3 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 17-4 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。

表 17-4 DBMS_LDAP_UTL のその他のサブプログラム（続き）

ファンクションまたはプロシージャ	用途
get_property_values_blob ファンクション	プロパティ名のラージ・バイナリ値のリストを取得します。
property_value_free_blob プロシージャ	DBMS_LDAP_UTL.get_property_values_blob() によって戻された BLOB_COLLECTION に関連付けられているメモリーを解放します。
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。

サブプログラム

この項の項目は次のとおりです。

- ユーザー関連サブプログラム
- グループ関連サブプログラム
- サブスクライバ関連サブプログラム
- プロパティ関連サブプログラム
- その他のサブプログラム

ユーザー関連サブプログラム

ユーザーは DBMS_LDAP_UTL.HANDLE データ型で表現されます。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。次の例は、ユーザー・ハンドルの作成方法を示しています。

```
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、subscriber_handle が o=acme,dc=com 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
);
```

ユーザー・ハンドルの一般的な用途としては、ユーザー・プロパティの設定と取得、ユーザーの認証などがあります。次に、ユーザーを認証するハンドルを示します。

```
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ハンドルを示します。

```
--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
);
```

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 17-5 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 DBMS_LDAP_UTL.AUTH_SIMPLE のみです。
credentials	VARCHAR2	ユーザー資格証明。
binary_credentials	RAW	バイナリ資格証明。このパラメータはオプションです。デフォルトで NULL にできます。

戻り値

表 17-6 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	ユーザーが複数の識別名エントリを持ちます。
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	サブスクライバが複数の識別名エントリを持ちます。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.ACCT_TOTALLY_LOCKED_EXCP	ユーザー・アカウントがロックされています。
DBMS_LDAP_UTL.AUTH_PASSWD_CHANGE_WARN	この戻り値は使用できません。
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 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 17-7 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 17-8 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 17-9 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 DBMS_LDAP_UTL.SUBSCRIBER_HANDLE です。
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 17-10 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 のユーザー・タイプで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 17-11 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	取得するユーザー属性のリストです。
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_collection	PROPERTY_SET_COLLECTION	コール元がリクエストした属性に含まれているユーザーの詳細です。

戻り値

表 17-12 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	ユーザーが複数の識別名エントリを持ちます。
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 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 DBMS_LDAP_UTL.TYPE_GUID または DBMS_LDAP_UTL.TYPE_DN 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。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 17-13 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	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 ENTRY_PROPERTIES です。
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 17-14 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	ユーザーが複数の識別名エントリを持ちます。
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,
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 17-15 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 DBMS_LDAP_UTL.EXTPROPTYPE_RAD です。
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	コール元がリクエストした属性が含まれているユーザーの詳細です。

戻り値

表 17-16 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	ユーザーが複数の識別名エントリを持ちます。
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 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 17-17 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 17-18 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	ユーザーが複数の識別名エントリを持ちます。
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 error codes	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、ユーザーがグループに属しているかどうか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 17-19 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 17-20 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 17-21 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 17-22 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 17-23 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 17-24 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。次に例を示します。

```
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
);
```

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 17-25 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 17-26 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 17-27 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 DBMS_LDAP_UTL.GROUP_HANDLE です。
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 17-28 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 のグループ・タイプで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 17-29 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 DBMS_LDAP_UTL.ENTRY_PROPERTIES です。
ret_pset_coll	PROPERTY_SET_COLLECTION	コール元がリクエストした属性が含まれているグループの詳細です。

戻り値

表 17-30 GET_GROUP_PROPERTIES ファンクションの戻り値

値	説明
DBMS_LDAP_UTL.SUCCESS	正常終了した場合の戻り値です。
DBMS_LDAP_UTL.PARAM_ERROR	入力パラメータが無効です。
DBMS_LDAP_UTL.NO_SUCH_GROUP	グループが存在しません。
DBMS_LDAP_UTL.MULTIPLE_GROUP_ENTRIES	指定したグループに対して、複数のグループ識別名エントリがディレクトリに存在します。
DBMS_LDAP_UTL.INVALID_ROOT_ORCL_CTX	ルートの Oracle コンテキストが無効です。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	LDAP サーバーによる LDAP 操作中に発生した無条件の障害に対して、適切な DBMS_LDAP エラー・コードを戻します。

使用方法

このファンクションには、次の要件があります。

- DBMS_LDAP.init() ファンクションで取得した有効な LDAP セッション・ハンドルが必要です。
- グループのタイプが DBMS_LDAP_UTL.TYPE_NICKNAME の場合は、グループ・ハンドルのプロパティに有効なサブスクライバ・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 DBMS_LDAP_UTL.TYPE_GUID または DBMS_LDAP_UTL.TYPE_DN の場合は、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。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 17-31 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 17-32 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。
DBMS_LDAP_UTL.GENERAL_ERROR	その他のエラーです。
DBMS_LDAP エラー・コード	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。次の例は、サブスクライバ・ハンドルの作成方法を示しています。

```
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・プロパティの取得は、サブスクライバ・ハンドルの一般的な用途の 1 つです。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
);
```

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
(
subscriber_hd OUT HANDLE,
subscriber_type IN PLS_INTEGER,
subscriber_id IN VARCHAR2
)
RETURN PLS_INTEGER;
```

パラメータ

表 17-33 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 DBMS_LDAP_UTL.TYPE_DEFAULT の場合は、このパラメータを NULL にできます。その場合、デフォルト・サブスクライバはルートの Oracle コンテキストから取得されます。

戻り値

表 17-34 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 17-35 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取得する必要がある属性のリストです。
ptype	PLS_INTEGER	戻すサブスクライバの Oracle コンテキストのプロパティ。有効な値は次のとおりです。 ■ DBMS_LDAP_UTL.ENTRY_PROPERTIES ■ DBMS_LDAP_UTL.COMMON_PROPERTIES
ret_pset_coll	PROPERTY_SET_COLLECTION	コール元がリクエスト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 17-36 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	サブスクライバが複数の識別名エントリを持ちます。
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 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 17-37 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 17-38 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。
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 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 17-39 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	取得するサブスクライバ属性のリストです。
ptype	PLS_INTEGER	戻すプロパティのタイプ。有効な値は DBMS_LDAP_UTL.DEFAULT_RAD_PROPERTIES です。
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。

表 17-39 GET_SUBSCRIBER_EXT_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
ret_pset_collection	PROPERTY_SET_COLLECTION	コール元がリクエスト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 17-40 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 error codes	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。

その他のサブプログラム

DBMS_LDAP_UTL パッケージのその他のサブプログラムは、様々な機能を実行します。

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 17-41 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 17-42 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 17-43 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 17-44 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 17-45 GET_PROPERTY_VALUES ファンクションのパラメータ

パラメータ名	パラメータ・タイプ	パラメータの説明
property_name	VARCHAR2	プロパティ名です。
pset	PROPERTY_SET	次のいずれかのファンクションで戻される PropertySetCollection 内のプロパティ・セットです。 ■ DBMS_LDAP_UTL.get_group_membership() ■ DBMS_LDAP_UTL.get_subscriber_properties() ■ DBMS_LDAP_UTL.get_user_properties() ■ DBMS_LDAP_UTL.get_group_properties()
property_values	STRING_COLLECTION	プロパティ値（文字列）のリストです。

戻り値

表 17-46 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 17-47 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 17-48 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 17-49 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()

関連項目

DBMS_LDAP_UTL.get_group_membership()、
DBMS_LDAP_UTL.get_subscriber_properties()、
DBMS_LDAP_UTL.get_user_properties()、
DBMS_LDAP_UTL.get_group_properties()。

create_mod_propertyset ファンクション

create_mod_propertyset() は、MOD_PROPERTY_SET データ構造を作成するファンクションです。

構文

```
FUNCTION create_mod_propertyset
(
pset_type IN PLS_INTEGER,
pset_name IN VARCHAR2,
mod_pset OUT MOD_PROPERTY_SET
)
RETURN PLS_INTEGER;
```

パラメータ

表 17-50 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 ENTRY_PROPERTIES です。
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 17-51 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 17-52 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 17-53 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 17-54 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。

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 17-55 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。

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 17-56 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 17-57 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。

get_property_values_blob ファンクション

get_property_values_blob() は、指定したプロパティ名とプロパティに対するラージ・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_blob
(
pset IN PROPERTY_SET,
property_name IN VARCHAR2,
auth_type IN PLS_INTEGER,
property_values OUT BLOB_COLLECTION
)
RETURN PLS_INTEGER;
```

パラメータ

表 17-58 GET_PROPERTY_VALUES_BLOB ファンクションのパラメータ

パラメータ	パラメータ・タイプ	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	BLOB_COLLECTION	バイナリ・プロパティ値のリストです。

戻り値

表 17-59 GET_PROPERTY_VALUES_BLOB 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()。

property_value_free_blob プロシージャ

DBMS_LDAP.get_property_values_blob() によって戻された BLOB_COLLECTION に関連付けられているメモリーを解放します。

構文

```
Syntax
PROCEDURE property_value_free_blob
(
vals IN OUT DBMS_LDAP.BLOB_COLLECTION
);
```

パラメータ

表 17-60 PROPERTY_VALUE_FREE_BLOB ファンクションのパラメータ

パラメータ	説明
vals	DBMS_LDAP.get_property_values_blob() によって戻されたラージ・バイナリ値の集合です。

関連項目

DBMS_LDAP.get_property_values_blob()。

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 17-61 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	ユーザーがグループのメンバーでない場合は、ユーザー関連ファンクションとグループ・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。
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	このリターン・コードは使用できません。
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。

表 17-61 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_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。
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_user() ファンクションからこのコードが戻されます。これは、パスワード・ポリシー・エラーです。

表 17-61 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
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 17-62 DBMS_LDAP_UTL のデータ型

データ型	用途
HANDLE	エンティティ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。

18

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。

- サービス・ユニットのディレクトリ・エントリ
- サービス・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検索および選択サービス・ユニット

サービス・ユニットのディレクトリ・エントリ

表 18-1 に、Oracle Delegated Administration Services ユニットと、各ユニットの相対 URL が格納されるディレクトリ・エントリを示します。

表 18-1 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=Create User,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContext
ユーザーの編集	cn=Edit User,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=Edit UserGivenGUID,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=Create Group,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループの編集	cn=Edit Group,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=Edit GroupGivenGUID,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
ユーザーへの権限の割当て	cn=User Privilege,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=User Privilege Given GUID,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループへの権限の割当て	cn=Group Privilege,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=Group Privilege Given GUID,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=Account Info,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=Password Change,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザーの検索	cn=User Search,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループの検索	cn=Group Search,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
ユーザー LOV の検索	cn=User LOV,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
グループ LOV の検索	cn=Group LOV,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"

表 18-1 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
コンソールの委任	cn=Delegation Console,cn=OperationURLs,cn=DAS,cn=Products, cn=OracleContext
パスワードのリセット	cn=Reset Password,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContex
ユーザー・プロファイルの表示	cn=View User Profile,cn=OperationURLs,cn=DAS,cn=Products,cn=OracleContext

サービス・ユニットおよび対応する URL パラメータ

表 18-2 に、サービス・ユニットと各ユニットに指定可能な URL パラメータを示します。

表 18-2 サービス・ユニットおよび対応する URL パラメータ

サービス・ユニット	パラメータ	戻り値
ユーザーの作成	doneURL homeURL cancelURL enablePA parentDN enableHomeURL enableHelpURL	returnGUID
ユーザーの編集	homeURL doneURL cancelURL enablePA enableHomeURL enableHelpURL	-
UserGivenGUID の編集	homeURL doneURL cancelURL enablePA userGUID enableHomeURL enableHelpURL	-
プロファイルの編集	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
コンソールの委任	-	-
DeleteUser	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
DeleteUserGivenGUID	homeURL doneURL cancelURL userGUID enableHomeURL enableHelpURL	-

表 18-2 サービス・ユニットおよび対応する URL パラメータ（続き）

サービス・ユニット	パラメータ	戻り値
ユーザー権限	homeURL doneURL cancelURL enableHomeURL enableHelpURL	
ユーザー権限（指定された GUID）	homeURL doneURL cancelURL userGUID enableHomeURL enableHelpURL	-
グループの作成	homeURL doneURL cancelURL enablePA parentDN enableHomeURL enableHelpURL	returnGUID
グループの編集	homeURL doneURL cancelURL enablePA enableHomeURL enableHelpURL	-
GroupGivenGUID の編集	homeURL doneURL cancelURL enablePA groupGUID enableHomeURL enableHelpURL	-
DeleteGroup	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
DeleteGroupGivenGUID	homeURL doneURL cancelURL groupGUID enableHomeURL enableHelpURL	-
グループ権限	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
グループ権限（指定された GUID）	homeURL doneURL cancelURL groupGUID enableHomeURL enableHelpURL	-

表 18-2 サービス・ユニットおよび対応する URL パラメータ（続き）

サービス・ユニット	パラメータ	戻り値
アカウント情報	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
パスワードの変更	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
ユーザーの検索	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
グループの検索	homeURL doneURL cancelURL enableHomeURL enableHelpURL	-
パスワードのリセット	cancelURL doneURL enableHomeURL enableHelpURL	-
ユーザー・プロファイルの表示	userGuid doneURL homeURL nableHomeURL enableHelpURL	-
ユーザー LOV	base cfilter title dasdomain callbackURL	userDn userGuid userName nickName userEmail
グループ LOV	otype base cfilter title dasdomain callbackURL	groupDN groupGuid groupName groupDescription

DAS URL API のパラメータの説明

表 18-3 に示すパラメータは DAS ユニットで使用されます。

表 18-3 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した場合、「Home」をクリックすると、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。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。
callbackURL	DAS は、この URL を使用して、起動元のアプリケーションに戻り値を送信します。UserLOV および GroupLOV ユニットの場合、戻り値は HTML フォーム・パラメータとして HTTP POST メソッドで送信されます。
cancelURL	この URL は、DAS ユニットに表示されるすべての「Cancel」ボタンにリンクされます。ユーザーが「Cancel」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。true に設定すると、ユーザーまたはグループ操作での権限の割当てが使用可能になります。「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。GUID を指定すると、editUser または deleteUser ユニットのユーザー検索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。GUID を指定すると、editGroup または deleteGroup ユニットのグループ検索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、作成操作で終了 URL 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/
enableHomeUR	このパラメータが値 false とともに渡されると、サービス・ユニットは、「Home」ボタンまたは「Home」リンクなしでレンダリングされます。デフォルトでは、このパラメータは true に設定されます。

表 18-3 DAS URL のパラメータの説明（続き）

パラメータ	説明
enableHelpURL	このパラメータが値 false とともに渡されると、サービス・ユニットは、「Help」ボタンまたは「Help」リンクなしでレンダリングされます。デフォルトでは、このパラメータは true に設定されます。

ユーザーまたはグループの検索および選択サービス・ユニット

DAS には、ユーザーまたはグループの検索と選択を行うサービス・ユニットが用意されています。このサービス・ユニットは、ユーザーまたはグループの値リスト（LOV）と呼ばれることもあります。

ユーザーまたはグループの検索および選択サービス・ユニットの起動

ユーザーまたはグループの検索および選択用 URL を指定すると、カスタム・アプリケーションでポップアップ・ウィンドウを開き、そのポップアップ・ウィンドウに内容を移入できます。それぞれに次の形式の URL を使用します。

```
http://das_host:das_port/oiddas/ui/oracle/ldap/das/search/LOVUserSearch
?title=User&callbackurl=http://app_host:app_port/custapp/Callback
```

または

```
http://das_host:das_port/oiddas/ui/oracle/ldap/das/search/LOVGroupSearch
?title=User&callbackurl=http://app_host:app_port/custapp/Callback
```

次に例を示します。

```
http://server02.example.com:7777/oiddas/ui/oracle/ldap/das/search/LOVUserSearch?
Mary.Smith=User&callbackurl=http://server04.example.com:7778/custapp/Callback
```

この例で、`server02.example.com:7777` は、Oracle Internet Directory DAS アプリケーション・サーバーのホスト名とポートです。`server04.example.com:7778` は、カスタム・アプリケーション・サーバーのホスト名とポートです。`Mary.Smith` は、「Search and Select」ページのタイトルに表示される文字列です。
`http://server04.example.com:7778/custapp/Callback` は、ユーザーまたはグループに対して選択したパラメータを受信するカスタム・アプリケーション・サーバーの URL です。

注意： ポップアップがブロックされないように、ローカル・カスタム・アプリケーション・サーバーの URL でポップアップ・ウィンドウを開き、Oracle Internet Directory DAS のユーザーまたはグループの検索および選択用 URL へ即座にリダイレクトすることもできます。

ユーザーまたはグループの検索および選択サービス・ユニットからのデータ受信

Oracle Internet Directory DAS のユーザーまたはグループの検索および選択サービス・ユニットでユーザーまたはグループを選択すると、HTTP フォームが POST メソッドを使用して callbackurl ページに送信されます。callbackurl ページでは、表 18-4 と表 18-5 に示すパラメータを使用できます。

表 18-4 ユーザーの検索と選択

パラメータ	説明
userDn	ユーザーの識別名。
userGuid	ユーザーのグローバルに一意な ID。
userName	ユーザーの名前。
nickName	ユーザーのニックネーム
userEmail	ユーザーの電子メール・アドレス。

表 18-5 グループの検索と選択

パラメータ	説明
groupDn	グループの識別名。
groupGuid	グループのグローバルに一意な ID。
groupName	グループの名前。
groupDescription	グループの説明。

ポップアップ・ウィンドウの callbackurl ページは、JavaScript を使用して、元のウィンドウの起動元ページにフォーム・パラメータを転送します。その後、ポップアップ・ウィンドウを閉じます。

注意： JavaScript のセキュリティの問題を回避するために、起動元ページと同じサーバーで callbackurl ページを開いてもかまいません。この場合、ポップアップ・ウィンドウの callbackurl ページと元のウィンドウの起動元ページは JavaScript で直接通信できます。

19

Oracle Directory Integration Platform ユーザー・プロビジョニング Java API リファレンス

10*g*（10.1.4.0.1）には、様々な使用方法に対して最適化された、2 つの補完的な製品が用意されています。

- Oracle Identity Manager（旧 Oracle Xellerate IP）は、ディレクトリ、データベース、メインフレーム、独自のテクノロジ、フラット・ファイルなどを含む、非常に異なる技術で構成された複雑な環境を管理するために設計されたエンタープライズ・プロビジョニング・プラットフォームです。Oracle Identity Manager には、豊富な一連の監査およびコンプライアンス機能に加え、様々な機能を持つワークフローおよびポリシー機能が備わっています。
- Identity Management インフラストラクチャのコンポーネントである Oracle Directory Integration Platform は、ディレクトリ集中型の環境において、ディレクトリ同期やプロビジョニング・タスクを実行するために設計されたメタディレクトリ・テクノロジです。Oracle Directory Integration Platform は、ディレクトリおよび互換性のある Oracle 製品で構成されるより複雑な異機種間環境を管理するよう設計されています。Oracle Directory Integration Platform は、データ同期を使用してプロビジョニング・タスクを実行します。ワークフローおよびフル機能のポリシー・エンジンが不要な場合には、Oracle Directory Integration Platform により少量のデプロイメント・フットプリントが用意されます。

Oracle Internet Directory SDK には、Oracle Directory Integration Platform ユーザー・プロビジョニング API が含まれます。この API を使用すると、Oracle Identity Management インフラストラクチャでユーザーとそのアプリケーションのプロパティを管理できます。この章では、この API の主要機能と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。

- アプリケーション構成
- ユーザー管理
- デバッグ
- サンプル・コード

アプリケーション構成

アプリケーションをプロビジョニング可能な状態にするには、プロビジョニング・システムに登録する必要があります。コマンドライン・インタフェースを使用して Oracle Internet Directory に独自の構成を作成する必要もあります。アプリケーション構成を表示するための Java クラスが存在します。

この項の項目は次のとおりです。

- アプリケーション登録とプロビジョニング構成
- アプリケーション構成クラス

アプリケーション登録とプロビジョニング構成

プロビジョニング・システムに登録するためには、アプリケーションでプロビジョニング構成を作成する必要があります。プロビジョニング構成が存在すると、ディレクトリ対応であり、プロビジョニングが可能なアプリケーションとしてプロビジョニング・システムに認識されます。

アプリケーションでは、次の手順によりプロビジョニング構成を作成します。

1. アプリケーションの登録
2. プロビジョニングの構成

アプリケーションの登録

通常、Oracle アプリケーションは、`$ORACLE_HOME/jlib` にある `repository.jar` ファイルのリポジトリ API を使用して自己登録を行います。このファイルは、アプリケーション登録専用としてインストール時に配置されます。リポジトリ API は、Oracle Internet Directory にアプリケーション・エントリを作成する場合だけでなく、アプリケーションを権限グループに追加する場合にも使用できます。

ただし、顧客が作成したアプリケーションでは、アプリケーション登録に `repository.jar` API を使用することはできません。そのため、アプリケーション開発者は、LDIF テンプレートを使用し、LDAP コマンドで Oracle Internet Directory にアプリケーション・エントリを作成する必要があります。

次のいずれかのコンテナの下に、アプリケーション独自のコンテナを作成する必要があります。

- `cn=Products,cn=OracleContext`: 複数レルムのユーザーにサービスを提供するアプリケーション用
- `cn=Products,cn=OracleContext,`*`RealmDN`*: 特定レルムのユーザーにサービスを提供するアプリケーション用

特定レルム用としてアプリケーションを構成すると、そのアプリケーションでは、他のレルムのユーザーを管理できません。ほとんどの場合、Oracle Internet Directory の特定レルムに関連付けないように、アプリケーションを任意の ID 管理レルムの外部に作成する必要があります。

アプリケーションの新規インスタンスをインストールすると、そのアプリケーションのコンテナの下にアプリケーション・インスタンス用の個別エントリが作成されます。一部のプロビジョニング構成は特定タイプのインスタンスすべてに共通ですが、一部はそのインスタンスに固有です。企業でアプリケーションの複数のインスタンスを配置する場合、各インスタンスは相互に独立しています。各インスタンスは、それぞれ個別のプロビジョニング可能アプリケーションとして定義されます。このアプリケーションの 1 つ以上のインスタンスにユーザーをプロビジョニングできます。そのため、ユーザーは、このアプリケーションの 1 つ以上のインスタンスにアクセスできます。

この項の例では、Oracle Files に似たサンプル・アプリケーションを扱います。このアプリケーションの最初のインスタンスをインストールするときに、Oracle Internet Directory に特定のエントリを作成する必要があります。次の例において、実行時に選択されるこのアプリケーションの名前は `Files-App1` であり、アプリケーションのタイプは `FILES` です。このアプリケーションには、必要に応じてインスタンス化して Oracle Internet Directory にアップロードできる LDIF テンプレートを含めることができます。この例では、アプリケーション ID は任意のレル

ムの外部にあります。つまり、アプリケーション ID は、cn=Products,cn=OracleContext コンテナの下に存在します。

```
dn: cn=FILES,cn=Products,cn=OracleContext
changetype: add
objectclass: orclContainer

dn: orclApplicationCommonName=Files-App1,cn=FILES,cn=Products,cn=OracleContext
changetype: add
orclappfullname: Files Application Instance 1
userpassword: welcome123
description: This is a test Appliction instance.
protocolInformation: xxxxx
orclVersion: 1.0
orclaci: access to entry by group="cn=odisgroup,cn=DIPAdmins,
 cn=Directory Integration  Platform,cn=Products,
 cn=OracleContext" (browse,proxy) by group="cn=User Provisioning Admins,
 cn=Groups,cn=OracleContext" (browse,proxy)
orclaci: access to attr=(*) by group="cn=odisgroup,cn=DIPAdmins,
 cn=Directory Integration Platform,cn=Products,
 cn=OracleContext" (search,read,write,compare)
 by group="cn=User Provisioning Admins,
 cn=Groups,cn=OracleContext" (search,read,write,compare)
```

この例での ACL の詳細は、「アプリケーション・ユーザー・データの場所」を参照してください。

アプリケーションでは、プロビジョニング管理者以外にも、いくつかのプロビジョニング・サービスに一定の権限を付与する必要があります。

このアプリケーションの 2 番目のインスタンスをインストールする場合、Oracle Directory Integration Platform に次のエントリを作成する必要があります。この例では、実行時に決定されるこのアプリケーションの名前を Files-App2 とします。

```
dn: orclApplicationCommonName=Files-App2,cn=FILES,cn=Products,cn=OracleContext
changetype: add
orclappfullname: Files Application Instance 2
userpassword: welcome123
description: This is a test Appliction instance.
orclVersion: 1.0
orclaci: access to entry by group="cn=odisgroup,
 cn=DIPAdmins,cn=Directory Integration Platform,cn=Products,
 cn=OracleContext" (browse,proxy) by group="cn=User Provisioning Admins,
 cn=Groups,cn=OracleContext" (browse,proxy)
orclaci: access to attr=(*) by group="cn=odisgroup,cn=DIPAdmins,
 cn=Directory Integration Platform,cn=Products,
 cn=OracleContext" (search,read,write,compare) by
 group="cn=User Provisioning Admins,cn=Groups,cn=OracleContext"
 (search,read,write,compare)
```

アプリケーションのエントリが正常に作成されると、アプリケーション ID が Oracle Internet Directory に登録されます。この時点で、特定の権限が必要な場合には、アプリケーションを Oracle Internet Directory の特定の権限グループに追加できます。表 19-1「有益な権限グループの一部」に、アプリケーションを追加できる権限グループの一部を示します。これらの各グループは、すべてのレルムに存在し、また、RootOracleContext にも存在します。RootOracleContext グループは、すべてのレルムのグループ・メンバーです。

表 19-1 有益な権限グループの一部

グループ名	権限
OracleDASCreateUser	パブリック・ユーザーの作成
OracleDASEditUser	パブリック・ユーザーの編集
OracleDASDeleteUser	パブリック・ユーザーの削除

表 19-1 有益な権限グループの一部（続き）

グループ名	権限
OracleDASCreateGroup	新規パブリック・グループの作成
OracleDASEditGroup	パブリック・グループの編集
OracleDASDeleteGroup	パブリック・グループの削除

たとえば、次の LDIF ファイルを使用すると、すべてのレルムでユーザーを作成する権限を付与する cn=OracleCreateUser に、Files-App1 アプリケーションを追加できます。

```
dn:cn=OracleCreateUser,cn=Groups,cn=OracleContext
changetype: modify
add: uniquemember
uniquemember:
orclApplicationCommonName=Files-App1,cn=FILES,cn=Products,cn=OracleContext
```

プロビジョニングの構成

アプリケーションのプロビジョニング構成は、そのアプリケーションのプロビジョニング・プロファイルに保存されます。プロビジョニング・システムは、バージョン 1.1、2.0、3.0 という 3 つの異なるプロビジョニング・プロファイルに対応しています。プロビジョニング・サービスでは、プロファイル・バージョンごとに異なるサービスが提供されます。一部の一般構成の詳細は、バージョンに関係なくすべてのアプリケーションに共通です。

プロビジョニング構成バージョン間の差異

バージョン 3.0 のプロファイルとバージョン 2.0 および 1.1 のプロファイルとの違いは、次のとおりです。

- 新しいプロビジョニング・フレームワークでは、バージョン 3.0 のアプリケーションのみが認識されます。したがって、プロビジョニング対象のアプリケーションとして Oracle プロビジョニング・コンソールに表示されるのは、バージョン 3.0 のプロビジョニング・プロファイルを保持するアプリケーションのみです。バージョン 2.0 および 1.1 のプロファイルを保持するアプリケーションは、プロビジョニング対象のアプリケーションとしてプロビジョニング・コンソールに表示されません。ただし、これらのアプリケーションに構成されているイベントの情報は、アプリケーションに通知されます。
- バージョン 3.0 のプロファイルでは、アプリケーションのプロビジョニング構成の作成は複数ステップのプロセスです。旧バージョンのプロファイルでは、プロビジョニング登録には単一のステップのみが必要とされます（oidprovtool コマンドの実行）。
- アプリケーションは、異なるインタフェースを使用してプロビジョニング・イベントにサブスクライブできます。Java と OID-LDAP の 2 つのインタフェースは、バージョン 3.0 のプロビジョニング構成と併用されるバージョン 3.0 のインタフェースでのみ使用できます。詳細は、表 19-2「インタフェースとその構成」を参照してください。
- アプリケーションでは、LDIF ファイルで、そのアプリケーション固有のユーザー属性の構成を指定できます。これは、バージョン 3.0 のプロビジョニング構成と併用されるバージョン 3.0 のインタフェースでのみサポートされます。詳細は、19-9 ページの「アプリケーション・ユーザー属性とデフォルト値の構成」を参照してください。
- ユーザーのプロビジョニング・ステータス（『Oracle Identity Management 統合ガイド』を参照）は、バージョン 3.0 のアプリケーションのみを対象に保存されます。バージョン 3.0 より前のプロファイルを使用するアプリケーションのプロビジョニング・ステータスは、保存されません。
- イベント伝播の構成パラメータは、バージョンごとに変化します。詳細は、表 19-5「イベント伝播のパラメータ」を参照してください。

バージョン 3.0 固有のプロビジョニング構成

特に明記されていないかぎり、この項の残りの部分では、バージョン 3.0 に固有のプロビジョニング構成について説明します。図 19-1 に、プロビジョニング構成の格納に使用される Oracle Internet Directory のディレクトリ情報ツリー（DIT）を示します。すべてのプロビジョニング構成情報は、次のコンテナの下に配置されます。

```
cn=Provisioning,cn=Directory Integration Platform,cn=Products,cn=OracleContext
```

共通のプロビジョニング構成情報は、次のコンテナの下のエントリに格納されます。

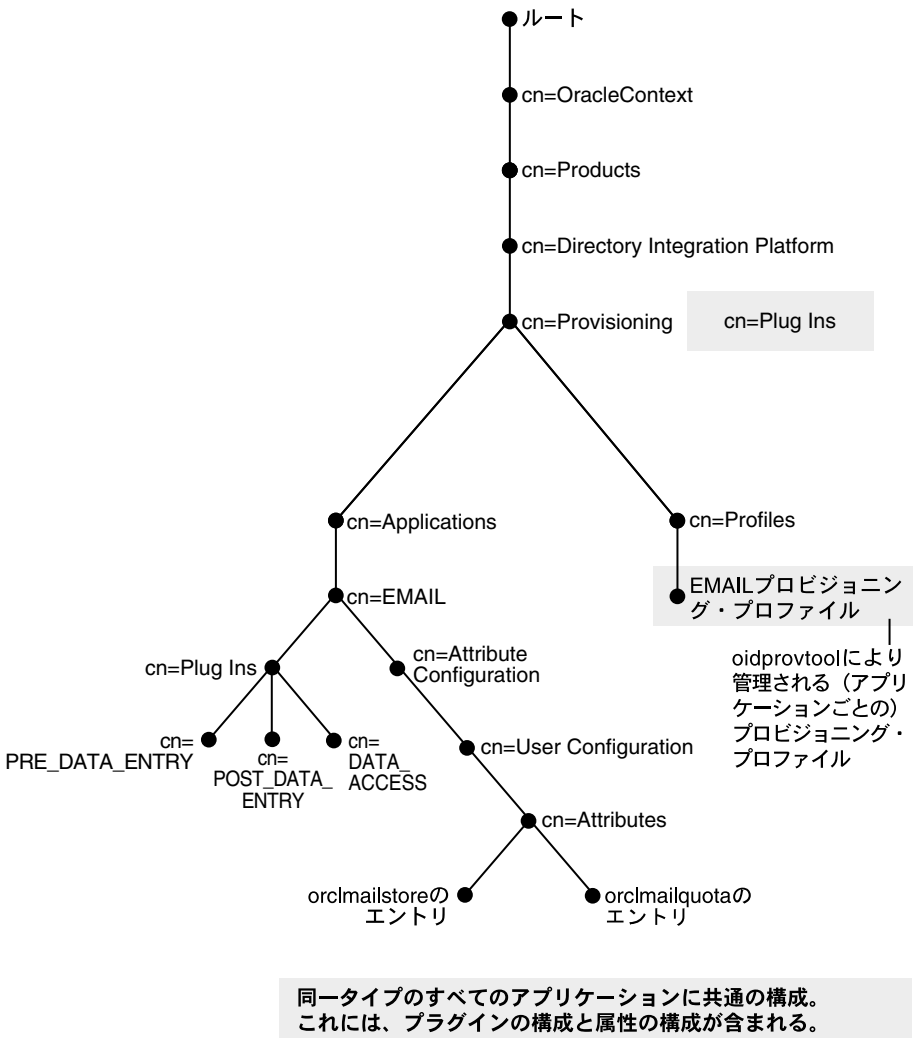
```
cn=Profiles,cn=Provisioning,cn=Directory Integration Platform,
 cn=Products,cn=OracleContext
```

残りのアプリケーション用のプロビジョニング構成は、次のコンテナの下に配置されます。

```
cn=ApplicationType,cn=Applications,cn=Provisioning,
 cn=Directory Integration Platform,cn=Products,cn=OracleContext
```

特定のアプリケーション・タイプのすべてのインスタンスは、このコンテナの下にある構成を共有します。つまり、既存のアプリケーション・タイプの 2 番目のインスタンスによってプロビジョニング・プロファイルが作成されると、*cn=ApplicationType* コンテナの下のすべての構成情報が共有されます。

図 19-1 プロビジョニング構成データのディレクトリ情報ツリー



Profiles コンテナには、次のタイプの構成情報が含まれます。

- アプリケーション ID の情報
- アプリケーション ID レルムの情報
- アプリケーション・プロビジョニングとデフォルト・ポリシー
- アプリケーション・ユーザー・データの場所
- イベント・インタフェースの構成
- アプリケーション・ユーザー属性とデフォルト値の構成
- アプリケーション・プロビジョニング・プラグインの構成
- アプリケーション伝播の構成
- アプリケーション・イベント伝播の実行時ステータス

アプリケーション・インスタンスによってプロファイルが作成されると、その新規プロファイルは、常に次の名前書式で Profiles コンテナの下の個別エントリに格納されます。

orclODIPProfileName=*GUID_of_the_Realm_Entry_GUID_of_the_Application_Identity*,….

プロビジョニング構成の作成時に、アプリケーションの次の情報を指定する必要があります。

アプリケーション ID の情報 アプリケーションの各インスタンスは、次のパラメータで一意に識別されます。

- アプリケーション識別名 : アプリケーションを示す Oracle Internet Directory での一意の識別名。これは必須パラメータです。
- アプリケーション・タイプ : 同一アプリケーションのすべてのインスタンスに共通のパラメータ。一部の構成は、特定タイプの複数のインスタンスで共有できます。これは必須パラメータです。
- アプリケーション名 : このパラメータは、個別に指定できます。指定しない場合、識別名から抽出されます。これはオプション・パラメータです。
- アプリケーション表示名 : ユーザーにわかりやすいアプリケーション名。この名前は、プロビジョニング可能なターゲット・アプリケーションとしてプロビジョニング・コンソールに表示されます。これはオプション・パラメータです。

プロビジョニング・プロファイルの作成時にこれらのアプリケーション ID のパラメータを指定するには、$ORACLE_HOME/bin/oidprovtool コマンドライン・ユーティリティにそれぞれ次の引数を使用します。

- application_type
- application_dn
- application_name
- application_display_name

> **関連資料 :**『Oracle Identity Management ユーザー・リファレンス』の oidprovtool コマンドライン・ツールのリファレンス

アプリケーション ID レルムの情報 特定レルムのユーザーのみにサービスを提供するには、そのレルムを対象にアプリケーションを登録します。アプリケーションでサービスを提供するレルムごとに、個別のプロビジョニング・プロファイルを作成する必要があります。複数レルムを使用する環境（ホスティングされた OracleAS Portal 環境など）では、個々のレルムを対象にアプリケーションを登録する必要があります。

レルムのプロビジョニング管理者がプロビジョニング・コンソールにアクセスすると、プロビジョニング可能なターゲット・アプリケーションとして、そのレルムに登録されているアプリケーションのみが表示されます。

プロビジョニング・プロファイルの作成時にレルム情報を指定するには、organization_dn 引数付きで $ORACLE_HOME/bin/oidprovtool コマンドライン・ユーティリティを使用します。

> **関連資料：**『Oracle Identity Management ユーザー・リファレンス』の oidprovtool コマンドライン・ツールのリファレンス

アプリケーション・プロビジョニングとデフォルト・ポリシー プロビジョニング・プロファイルの作成時に、プロビジョニング・コンソールでそのアプリケーションに対するプロビジョニングを管理するかどうかを指定できます。管理しない場合、そのアプリケーションは、プロビジョニング・コンソールにプロビジョニング対象として表示されません。ただし、Oracle Directory Integration Platform では、通常どおりこのプロファイルが処理され、イベントが伝播されます。

プロビジョニング・プロファイルの作成時にこの情報を指定するには、$ORACLE_HOME/bin/oidprovtool コマンドライン・ユーティリティに application_isdasvisible 引数を使用します。デフォルト値は TRUE です。

デフォルトで該当レルム内のすべてのユーザーをアプリケーションにプロビジョニングするか、どのユーザーもプロビジョニングしないかを決定するためのデフォルト・ポリシーを構成できます。有効な値は次のとおりです。

- PROVISIONING_REQUIRED: デフォルトですべてのユーザーをプロビジョニングします。
- PROVISIONING_NOT_REQUIRED: デフォルトでどのユーザーもプロビジョニングしません。

デフォルト値は PROVISIONING_REQUIRED です。

アプリケーション付属のポリシー・プラグインを使用して、デフォルト・ポリシーを実行時に上書きできます。また、管理者は、デフォルト・ポリシーとポリシー・プラグインの決定を両方とも上書きできます。

デフォルト・ポリシー情報を指定するには、$ORACLE_HOME/bin/oidprovtool コマンドライン・ユーティリティに default_provisioning_policy 引数を使用します。

アプリケーション・ユーザー・データの場所 アプリケーション固有のユーザー情報は、アプリケーション固有のコンテナに格納されます。このデータをプロビジョニング・システムで管理する場合、プロビジョニングの登録時にこれらのコンテナの場所を指定する必要があります。ユーザー・データの場所を指定するには、$ORACLE_HOME/bin/oidprovtool コマンドライン・ユーティリティに user_data_location 引数を使用します。アプリケーションでは、このコンテナに対する ACL で Oracle Delegated Administration Services および Oracle Directory Integration Platform を使用したコンテナの情報管理を許可する必要があります。

イベント・インタフェースの構成 アプリケーションは、PL/SQL、Java、OID-LDAP などの異なるインタフェースを使用してプロビジョニング・イベントにサブスクライブできます。表 19-2「インタフェースとその構成」に、サポートされるインタフェースとその関連構成を示します。INTERFACE_VERSION は、プロビジョニング・プロファイルのバージョンに合せてください。

表 19-2 インタフェースとその構成

構成パラメータ	PL/SQL	Java	OID-LDAP
INTERFACE_VERSION	1.1、2.0、3.0	3.0	3.0
INTERFACE_NAME	インタフェースを実装する PL/SQL パッケージの名前。	未使用。	未使用。
INTERFACE_CONNECT_INFO	データベース接続文字列。すべてのバージョンでサポートされる複数の書式です。	未使用。	未使用。
INTERFACE_ADDITIONAL_INFO	未使用。	未使用。	未使用。
プラグイン・タイプ	PRE_DATA_ENTRY、POST_DATA_ENTRY、DATA_ACCESS	PRE_DATA_ENTRY、POST_DATA_ENTRY、DATA_ACCESS、EVENT_DELIVERY (MUST)	PRE_DATA_ENTRY、POST_DATA_ENTRY、DATA_ACCESS
説明	主に Oracle データベースのバックエンドを基盤とするアプリケーションが対象です。DIP サーバーは、PL/SQL プロシージャを起動してリモート・データベースにイベントを送信します。	インタフェース型が Java の場合、イベント配信プラグインを構成する必要があります。構成しない場合、サーバーはエラーを返します。プラグイン構成により、残りの構成が決定されます。詳細は、「アプリケーション・プロビジョニング・プラグインの構成」を参照してください。	主に、アプリケーションが Oracle Internet Directory と緊密に統合されており、PL/SQL インタフェースまたは Java イベント配信プラグインによるイベント配信が不要な場合に使用されます。このインタフェースは、将来的に使用できなくなります。かわりに Java インタフェースを使用してください。

イベント・インタフェース構成を指定する場合、`$ORACLE_HOME/bin/oidprovtool` に次の引数を使用できます。

- `interface_type`（デフォルトは PLSQL）
- `interface_version`（デフォルトは 2.0）
- `interface_name`
- `interface_connect_info`
- `interface_additional_info`

表 19-3「PL/SQL インタフェースでサポートされる情報の書式」に、リモート・データベースへの接続時に PL/SQL インタフェースでサポートされるインタフェース接続情報の書式を示します。すべてのインタフェース・バージョンですべての書式がサポートされます。

表 19-3 PL/SQL インタフェースでサポートされる情報の書式

書式	説明
dbHost:*dbPort*:*dbSID*:*username*:*password*	古い書式であり、推奨されません。Oracle Directory Integration Platform では、この情報がシン JDBC ドライバに渡されます。

表 19-3 PL/SQL インタフェースでサポートされる情報の書式（続き）

書式	説明
dbHost:*dbPort*:*dbServiceName*:*username*:*password*	新しい書式です。高可用性実装では、データベース・ホストおよびポートが変わる可能性があるため、推奨されません。DIP では、この情報がシン JDBC ドライバに渡されます。
DBSVC=*DB_TNS_Connect_Sring_Alias*:*username*:*password*	JDBC シック OCI ドライバで使用されます。ローカルの `tnsnames.ora` ファイルには、DIP が稼働するノード上での別名を含める必要があります。
DBURL=ldap://*LDAP_host*:*LDAP_port*/*ServiceName*,cn=OracleContext	高可用性要件に対応しているため、推奨される書式です。DIP では、この情報がシン JDBC ドライバに渡されます。ドライバは、Oracle Internet Directory のデータベース登録エントリを検索し、実際のデータベース接続情報を取得します。

サポートされる書式の例は、次のとおりです。

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

localhost:1521:iasdbsvc:scott:tiger

DBSVC=TNSALIAS:scott:tiger

DBURL=ldap://acme.com:389/samplegdbname:scott:tiger
```

アプリケーション・ユーザー属性とデフォルト値の構成 アプリケーションでは、LDIF ファイルで、そのアプリケーション固有のユーザー属性の構成を指定できます。これは、バージョン 3.0 のインタフェースでのみサポートされます。

図 19-1「プロビジョニング構成データのディレクトリ情報ツリー」に示されているとおり、特定の属性の構成は、次のコンテナの下に個別エントリとして格納されます。

```
"cn=Attributes,cn=User Configuration,cn=Attribute configuration,
 cn=Application_Type,cn=Applications,cn=Provisioning,
 cn=Directory Integration Platform,cn=Products,cn=OracleContext"
```

この情報をアップロードするための `oidprovtool` の引数はありません。LDAP ファイルとコマンドライン・ツールを使用して、属性の構成情報を Oracle Internet Directory にアップロードする必要があります。

各アプリケーションに固有の属性は、個別のエントリとして示されます。次の例は、属性 `orclFilesDomain` を対象としています。

```
dn: cn=orclFilesDomain,cn=Attributes,cn=User configuration,cn=Attribute
configuration,.....
changetype: add
orcldasadminmodifiable: 1
orcldasviewable: 1
displayname: Files Domain
orcldasismandatory: 1
orcldasuitype: LOV
orcldaslov: us.oracle.com
orcldaslov: oraclecorp.com
orclDASAttrIsUIField: 1
orclDASAttrIsFieldForCreate: 1
orclDASAttrIsFieldForEdit: 1
orclDASAttrToDisplayByDefault: 1
orclDASSelfModifiable: 1
orclDASAttrDisplayOrder: 1
orclDASAttrDefaultValue: oraclecorp.com
orclDASAttrObjectClass: orclFILESUser
objectclass: orclDASConfigAttr
```

表 19-4「属性の構成エントリに属性として格納されるプロパティ」に、属性の構成エントリに属性として格納される各プロパティの意味を示します。

表 19-4 属性の構成エントリに属性として格納されるプロパティ

プロパティ名	説明	コメント
orclDASIsUIField	このプロパティが DAS コンソールに表示されるかどうかを示します。	10*g*（10.1.4.0.1）では使用されません。すべての属性が表示されます。
orclDASUIType	singletext、multitext、LOV、DATE、Number、password などの UI フィールドのタイプです。	Oracle Internet Directory セルフ・サービス・コンソールでのみ使用されます。
orclDASAdminModifiable	管理者がこのフィールドを変更できるかどうかを示します。	10*g*（10.1.4.0.1）では使用されません。管理者はすべての属性を変更できます。
orclDASViewAble	この属性が Oracle Internet Directory セルフ・サービス・コンソールで読取り専用属性となるかどうかを示します。	10*g*（10.1.4.0.1）では使用されません。
displayName	Oracle Internet Directory セルフ・サービス・コンソールに表示されるローカライズされた属性名です。	
orclDASIsMandatory	この属性が必須であるかどうかを示します。	必須属性が移入されない場合、Oracle Internet Directory セルフ・サービス・コンソールから警告を受けます。
orclDASAttrIsFieldForCreate	ユーザー作成時にかぎりこの属性が公開されるかどうかを示します。	10*g*（10.1.4.0.1）では使用されません。
orclDASAttrIsFieldForEdit	ユーザー編集時にかぎりこの属性が公開されるかどうかを示します。	10*g*（10.1.4.0.1）では使用されません。
orclDASAttrToDisplayByDefault	閉じられたセクションにおいて属性がデフォルトで非表示となるかどうかを示します。	10*g*（10.1.4.0.1）では使用されません。
orclDASSelfModifiable	ユーザーがこの属性を変更できるかどうかを示します。	10*g*（10.1.4.0.1）では使用されません。Oracle Internet Directory セルフ・サービス・コンソールは、アプリケーション固有の属性のみに対応しているためです。ユーザーは、Oracle Internet Directory セルフ・サービス・コンソールで各自のユーザー・プリファレンスを変更することはできません。
OrclDASAttrDisplayOrder	アプリケーション固有のセクションに属性が表示される順序です。	10*g*（10.1.4.0.1）では使用されません。
OrclDASAttrDefaultValue	プロビジョニング・コンポーネント（Oracle Internet Directory セルフ・サービス・コンソール、Oracle Directory Integration Platform およびバルク・プロビジョニング・ツール）で使用される属性の初期デフォルト値です。	Oracle Internet Directory セルフ・サービス・コンソールのアプリケーション管理ページを使用して変更できます。プラグインまたは管理者により、初期デフォルト値を上書きできます。

表 19-4 属性の構成エントリに属性として格納されるプロパティ（続き）

プロパティ名	説明	コメント
OrclDASAttrObjectClass	属性が属する LDAP オブジェクト・クラスです。	プロビジョニング・システムにより管理されるアプリケーション固有のユーザー・エントリを作成するために使用されます。

アプリケーションにアプリケーション固有の属性がある場合、プロビジョニング・システムでその属性のデフォルト値を管理するよう指定できます。これを行うには、`$ORACLE_HOME/bin/oidprovtool` に `manage_application_defaults` 引数を使用します。この引数は、デフォルトで `TRUE` です。

アプリケーション・プロビジョニング・プラグインの構成 アプリケーション・プロビジョニング・プラグインの詳細は、次を参照してください。

付録 A「ユーザー・プロビジョニング用の Java プラグイン」

アプリケーション伝播の構成 イベント伝播の構成パラメータは、プロファイルのバージョンごとに変わります。表 19-5「イベント伝播のパラメータ」に、イベント伝播の構成パラメータとその説明を示します。

表 19-5 イベント伝播のパラメータ

パラメータ	サポートされるプロビジョニング・プロファイルのバージョン	説明
profile_mode	2.0、3.0	アプリケーションで Oracle Internet Directory からの発信プロビジョニング・イベントの受信、着信イベントの送信、またはその両方を実行するかどうかを示します。値は、OUTBOUND（デフォルト）、INBOUND、BOTH です。
Schedule	1.1、2.0、3.0	保留イベントが伝播されるまでのスケジューリング間隔です。
enable_bootstrap	3.0	アプリケーション・ブートストラップ用のイベントを有効にします。これにより、アプリケーションでのプロビジョニング・プロファイルの作成前に、Oracle Internet Directory に存在するユーザーについてアプリケーションに通知するよう指定します。
enable_upgrade	3.0	アプリケーション・ユーザーのアップグレード用のイベントを有効にします。これにより、アップグレードの前に、Oracle Internet Directory に存在するユーザーについてアプリケーションに通知するよう指定します。アップグレードの前から存在するアプリケーションの場合、そのアプリケーションにユーザーがすでに存在している可能性があります。このようなユーザーについては、通常の新規ユーザーとは別に処理するように、Oracle Directory Integration Platform でアプリケーションにアップグレード・イベントを送信します。
lastchangenumber	3.0	アプリケーションへのイベントの送信元にする必要がある Oracle Internet Directory の変更番号です。
max_prov_failure_limit	3.0	アプリケーションにユーザーをプロビジョニングする際に、Oracle Directory Integration Platform Server が繰り返す最大試行回数です。

表 19-5 イベント伝播のパラメータ（続き）

パラメータ	サポートされるプロビジョニング・プロファイルのバージョン	説明
max_events_per_invocation	2.0、3.0	バルク・イベント伝播の場合、このパラメータにより、1 回のイベント・インタフェースの起動中にパッケージ化して送信できるイベントの最大数を指定します。
max_events_per_schedule	2.0	プロファイルの 1 回の実行で Oracle Directory Integration Platform がアプリケーションに送信するイベントの最大数です。デフォルトは 25 です。多くのプロファイルおよびアプリケーションが存在する配置の場合、このパラメータにより、マルチスレッド化された Oracle Directory Integration Platform を使用して複数のプロファイルのスレッドを実行できます。
event_subscription	1.1、2.0、3.0	アプリケーションがイベント伝播サービスから受信する OUTBOUND イベントのタイプを定義します。書式は次のとおりです。 *Object_Type*:*Domain*:*Operation*(*Attributes*,…) 次に例を示します。 USER:cn=users,dc=acme,dc=com:ADD(*) この例では、作成されたユーザーが指定ドメインに存在する場合、USER_ADD イベントを送信し、すべての属性も送信するよう指定しています。 USER:cn=users,dc=acme,dc=com:MODIFY(cn,sn.mail,telephonenumber) この例では、変更されたユーザーが指定ドメインに存在し、リストされた属性のいずれかが変更された場合、USER_MODIFY イベントを送信するよう指定しています。 USER:cn=users,dc=acme,dc=com:DELETE この例では、指定ドメインのユーザーが削除された場合、USER_DELETE イベントを送信するよう指定しています。

表 19-5 イベント伝播のパラメータ（続き）

パラメータ	サポートされるプロビジョニング・プロファイルのバージョン	説明
event_permitted_operations	2.0	アプリケーションが Oracle Directory Integration Platform Server への送信権限を持つ INBOUND イベントのタイプを定義します。書式は次のとおりです。 *Object_Type*:*Domain*:*Operation*(*Attributes*,…) 次に例を示します。 IDENTITY:cn=users,dc=acme,dc=com:ADD(*) この例では、指定ドメインに対して IDENTITY_ADD イベントを許可すると同時に、すべての属性も許可するよう指定しています。つまり、アプリケーションでは、Oracle Internet Directory にユーザーを作成できます。 IDENTITY:cn=users,dc=acme,dc=com:MODIFY(cn, sn.mail,telephonenumber) この例では、リストされている属性に対してのみ IDENTITY_MODIFY を許可するよう指定しています。他の属性は、暗黙的に無視されます。つまり、アプリケーションでは、リストされている Oracle Internet Directory のユーザー属性を変更できます。 IDENTITY:cn=users,dc=acme,dc=com:DELETE この例では、アプリケーションによる Oracle Internet Directory のユーザーの削除を許可するよう指定しています。
event_mapping_rules	2.0	INBOUND プロファイルの場合、このパラメータにより、アプリケーションおよび修飾フィルタ条件から取得するオブジェクトの型を指定して、このイベントの対象ドメインを決定します。複数の値を指定できます。書式は次のとおりです。 *Object_Type*: *Filter_condition*: *Domain_Of_Interest* 次に例を示します。 EMP::cn=users,dc=acme,dc=com この例では、取得されたオブジェクト型が EMP の場合、イベントの対象を cn=users,dc=acme,dc=com ドメインとするよう指定しています。 EMP:l=AMERICA:l=AMER,cn=users,dc=acme,dc=com この例では、取得されたオブジェクト型が EMP で、イベントに属性 l（地域）があり、その値が AMERICA の場合、イベントの対象を l=AMER,cn=users,dc=acme,dc=com ドメインとするよう指定しています。

アプリケーション・イベント伝播の実行時ステータス Oracle プロビジョニング・サービスでは、プロビジョニング統合アプリケーションごとに、ユーザーのプロビジョニング・ステータスが Oracle Internet Directory に記録されます。詳細は、『Oracle Identity Management 統合ガイド』の「プロビジョニングの配置および構成」の章を参照してください。

アプリケーション構成クラス

oracle.idm.user.provisioning.configuration.Configuration クラスを使用すると、プロビジョニング・スキーマの情報を取得できます。oracle.idm.user.provisioning.configuration.Application クラスを使用すると、登録アプリケーションのメタデータを取得できます。これらのクラスは、oracle.idm.provisioning.configuration パッケージに記述されています。

Configuration クラスでは、アプリケーション構成にアクセスできます。Configuration オブジェクトを構成するには、レルムを指定する必要があります。次に例を示します。

```
Configuration cfg = new Configuration ("us");
```

これで、Configuration クラスのメソッドを使用して、レルム内の一部またはすべてのアプリケーション構成を取得できます。レルムの LDAP コンテキストを指定する必要があります。

Configuration オブジェクトは、作成時に Oracle Internet Directory メタデータへのアクセスを必要とするため、パフォーマンスにかなりの影響を与えます。ベスト・プラクティスは、アプリケーションの初期化中に Configuration オブジェクトを一度作成し、後は必要とされるすべての操作でこのオブジェクトを再利用することです。

Application オブジェクトは、アプリケーション・インスタンスを示します。このオブジェクトのメソッドにより、インフラストラクチャの登録アプリケーションに関するメタデータを取得できます。

ユーザー管理

Oracle Directory Integration Platform または Oracle Delegated Administration Services でプロビジョニング・プラグインが起動すると、プロビジョニング対象のユーザーに関する情報が渡されます。配置されたアプリケーションでは、ユーザー・オブジェクトを使用してユーザーを変更できます。

ユーザー管理用のプロビジョニング・クラスにより、次の操作を実行できます。

- ベース・ユーザーの作成、変更および削除
- アプリケーション固有のユーザー情報の作成、変更および削除
- ベース・ユーザーの検索
- アプリケーションのユーザー・プロビジョニング・ステータスの取得

この項の項目は次のとおりです。

- ユーザーの作成
- ユーザーの変更
- ユーザーの削除
- ユーザーの検索

ユーザーの作成

Oracle Identity Management リポジトリにユーザーを作成する場合、次の 2 つの手順を実行します。

1. 指定したレルムに基本ユーザー情報を作成します。この情報は、ベース・ユーザーと呼ばれます。
2. アプリケーション固有のユーザー属性（フットプリント）を作成します。この情報は、アプリケーション・ユーザーと呼ばれます。

リポジトリ内のベース・ユーザーとアプリケーション・ユーザーの組合せは、Oracle Identity Management ユーザーと呼ばれます。メソッドには、ベース・ユーザーのみを作成するものと、Oracle Identity Management ユーザーの 2 つの構成要素を作成するものがあります。

ユーザーの作成に必要な最低限の情報は、ベース・ユーザーを示す属性のセットです。属性は、名前 / 値ペアの形式を取ります。これらのユーザー属性は、`oracle.ldap.util.ModPropertySet` クラスを使用する Java オブジェクトとして示されます。

一部のユーザー作成メソッドでは、Oracle Identity Management のユーザー・リポジトリに作成するエントリの識別名を指定する必要があります。それ以外のメソッドでは、識別名は必要ありません。かわりに、ユーザー作成先のレルムから取得されたメタデータ構成情報を使用して、Oracle Identity Management ユーザーが構成されます。

ベース・ユーザーとアプリケーション・ユーザーの作成に成功すると、作成メソッドによって `IdmUser` オブジェクトが戻されます。このオブジェクトを使用して、ベース・ユーザーとアプリケーション・ユーザーの属性を管理します。

ユーザーの変更

Oracle Identity Management リポジトリのベース・ユーザーの変更は、次の操作に関連します。

- ベース・ユーザー情報の変更
- アプリケーション・ユーザー情報の作成または変更

Oracle Identity Management ユーザーを変更するには、次の情報を指定する必要があります。

1. ユーザーの識別名、GUID または `IdmUser` オブジェクト参照
2. ベース・ユーザー属性に対する変更操作（`oracle.ldap.util.ModPropertySet` で指定）

一部のユーザー変更メソッドは、ベース・ユーザー属性のみを変更します。それ以外のメソッドは、アプリケーション・ユーザー属性も変更します。

ユーザーの削除

Oracle Identity Management リポジトリのベース・ユーザーの削除は、次の操作に関連します。

- ベース・ユーザー情報の削除
- アプリケーション・ユーザー情報の削除

Oracle Identity Management ユーザーを変更するには、識別名、GUID または IdmUser オブジェクト参照を指定する必要があります。

この操作の結果、ベース・ユーザーとアプリケーション・ユーザーの属性が削除されます。

ユーザーの検索

検索メソッドには、次の 2 つの検索オプションがあります。

- GUID または識別名を使用した特定の Oracle Identity Management ユーザーの検索
- 検索フィルタを使用した Oracle Identity Management ユーザーのセットの検索

Oracle Identity Management ユーザーを検索するには、識別名または GUID を指定する必要があります。

検索メソッドの出力は、次のいずれかになります。

- 単一の IdmUser オブジェクト
- IdmUser オブジェクトのリスト

デバッグ

デバッグとトレース情報を有効にするには、UtilDebug.MODE_PROVISIONING_API モードを設定します。ログ・メッセージの出力ストリームを指定しない場合、メッセージは標準出力に書き込まれます。

次のコード例は、UtilDebug.MODE_PROVISIONING_API モードを設定して出力ストリームを指定する方法を示しています。

```
Import oracle.ldap.util.UtilDebug;
FileOutputStream logStream = new FileOutputStream("ProvAPI.log")
…
UtilDebug.setDebugMode(UtilDebug.MODE_PROVISIONING_API);
UtilDebug.setPrintStream(logStream);
```

サンプル・コード

次のコード例は、ユーザーの作成、変更および検索方法と、アプリケーションのユーザー・プロビジョニング・ステータスの取得方法を示しています。

```
UtilDebug.setDebugMode(UtilDebug.MODE_PROVISIONING_API);
…
Configuration cfg = new Configuration(realm);
    try {
      debug("Connecting...");
      InitialLdapContext ctx =
          ConnectionUtil.getDefaultDirCtx(hostName, port, bindDn, passwd);
      debug("Connected...");
      UserFactory factory = UserFactoryBuilder.createUserFactory(ctx, cfg);

      // Create
      ModPropertySet mpSet = new ModPropertySet();
      mpSet.addProperty("cn","Heman");
      mpSet.addProperty("sn","The Master");
      mpSet.addProperty("uid","Heman");
      IdmUser idmUser = factory.createUser(mpSet);

      // Modify
      mpSet = new ModPropertySet();
      mpSet.addProperty(LDIF.ATTRIBUTE_CHANGE_TYPE_REPLACE,"sn",
            "Heman The Master");
      mpSet.addProperty("givenName","Master of the Universe");
      factory.modifyUser(idmUser, mpSet);

      // Lookup
      List users = factory.searchUsers(Util.IDTYPE_SIMPLE, "Hema*", null);
….
```

```
   // Get user provisioning status for an application.
   Application app = cfg.getApplication(lCtx, "Files", "FilesInstace");
   String  status = idmUser.getProvisioningStatus(app);


   // Another way to get user provisioning status
   String userDn = idmUser.getDNn();
   String  status = ProvUtil.getUserProvisioningStatus(dirctx,
         Util.IDTYPE_DN, userDn, app.getType(), app.getName());
 } catch (Exception ex) {
   ex.printStackTrace();
    //
}
```

20

Oracle Directory Integration Platform PL/SQL API リファレンス

この章では、Directory Integration Platform の登録 API 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。

- 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 3.0）
- 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

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 でした。リリース 9.0.4 および 10.1.2.0.0 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。リリース 10.1.2.0.1 では、さらに第 3 のバージョンが追加されました。管理者は、これらのいずれかのバージョンを使用できます。

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が 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、たとえば 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イベントを生成できることを示します。（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。

これら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 20-1 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 20-1 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 MODIFY DELETE	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、Oracle Internet Directory 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 または XYZ_DELETE イベントが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。20-4 ページの表 20-2 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 20-2 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）。 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。この文字列の形式は、[USER]GROUP]:[*domain_of_interest*]:[DELETE|ADD|MODIFY(*list_of_attributes_separated_by_comma*)] です。 この文字列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。パラメータを指定しない場合は、デフォルトとして、USER:*organization_DN*:DELETEGROUP:*organization_DN*: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。
MappingRules	着信イベントのみ（複数の値を取ります）。 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。マッピングの形式は次のとおりです。 *OBJECT_TYPE*: *Filter_condition*: *domain_of_interest* 複数の値を指定できます。マッピング EMP:cn=users,dc=acme,dc=com では、取得したオブジェクトの型は EMP です。イベントの対象はドメイン cn=users,dc=acme,dc=com です。マッピング EMP:l=AMERICA:l=AMER,cn=users,dc=acme,dc=com では、取得したオブジェクトの型は EMP です。イベントの対象はドメイン l=AMER,cn=users,dc=acme,dc=com です。
permittedOperations	着信イベントのみ（複数の値を取ります）。 アプリケーションがプロビジョニング統合サービスに送信する権限を持つイベントの型を定義します。マッピングの形式は次のとおりです。 *Event_Object*: *affected_domain*:*operation*(*attributes*, . . .) マッピング IDENTITY:cn=users,dc=acme,dc=com:ADD(*) では、IDENTITY_ADD イベントが指定したドメインおよびすべての属性に対して許可されます。マッピング IDENTITY:cn=users,dc=acme,dc=com:MODIFY(cn,sn.mail,telephonenumber) では、IDENTITY_MODIFY イベントがリスト内の属性に対してのみ許可されます。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 3.0）

バージョン 3.0 の PL/SQL インタフェースを使用する前に、次の関連資料を参照してください。

- 付録 A「ユーザー・プロビジョニング用の Java プラグイン」
- 『Oracle Identity Management 統合ガイド』の「Oracle プロビジョニング・サービスの概要」の章
- 『Oracle Identity Management 統合ガイド』の「プロビジョニング統合アプリケーションの配置」の章

PL/SQL コールバック・インタフェースを使用するには、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 パッケージ仕様を使用してパッケージを実装します。

```
DROP TYPE LDAP_EVENT_LIST_V3;
DROP TYPE LDAP_EVENT_V3;
DROP TYPE LDAP_EVENT_STATUS_LIST_V3;
DROP TYPE LDAP_ATTR_LIST_V3;
DROP TYPE LDAP_ATTR_V3;
DROP TYPE LDAP_ATTR_VALUE_LIST_V3;
DROP TYPE LDAP_ATTR_VALUE_V3;
-------------------------------------------------------------------------------------------
-----------
-- Name: LDAP_ATTR_VALUE_V3
-- Data Type: OBJECT
-- DESCRIPTION: This structure contains values of an attribute. A list of one or
more of this object is passed in any event.
-------------------------------------------------------------------------------------------
------------

CREATE TYPE LDAP_ATTR_VALUES_V3 AS OBJECT (
     attr_value       VARCHAR2(4000),
     attr_bvalue      RAW(2048),
     attr_value_len   INTEGER
);

GRANT EXECUTE ON LDAP_ATTR_VALUE_V3 to public;

CREATE TYPE LDAP_ATTR_VALUE_LIST_V3 AS TABLE OF LDAP_ATTR_VALUE_V3;
/
GRANT EXECUTE ON LDAP_ATTR_VALUE_LIST_V3 to public;
-------------------------------------------------------------------------------------------
-----------
-- Name: LDAP_ATTR_V3
-- 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_V3 AS OBJECT (
     attr_name        VARCHAR2(256),
     attr_type        INTEGER ,
     attr_mod_op      INTEGER,
     attr_values      LDAP_ATTR_VALUE_LIST_V3
);

GRANT EXECUTE ON LDAP_ATTR_V3 to public;

CREATE TYPE LDAP_ATTR_LIST_V3 AS TABLE OF LDAP_ATTR_V3;
/
GRANT EXECUTE ON LDAP_ATTR_LIST_V3 to public;
```

```
-------------------------------------------------------------------------------------
------------
-- Name: LDAP_EVENT_V3
-- Data Type: OBJECT
-- DESCRIPTION: This structure contains event information plus the attribute List.
-------------------------------------------------------------------------------------
------------

CREATE TYPE LDAP_EVENT_V3 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_V3 ) ;
/

GRANT EXECUTE ON LDAP_EVENT_V3 to public;
CREATE TYPE LDAP_EVENT_LIST_V3 AS TABLE OF LDAP_EVENT_V3;
/
GRANT EXECUTE ON LDAP_EVENT_LIST_V3 to public;
-------------------------------------------------------------------------------------
------------
-- Name: LDAP_EVENT_STATUS_V3
-- 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_V3 AS OBJECT (
          event_id     VARCHAR2(32),
          status       VARCHAR2(32),
          status_msg   VARCHAR2(2048),
          object_guid  VARCHAR(32)
) ;
/

GRANT EXECUTE ON LDAP_EVENT_STATUS_V3 to public;
CREATE TYPE LDAP_EVENT_STATUS_LIST_V3 AS TABLE OF LDAP_EVENT_STATUS_V3;
/
GRANT EXECUTE ON LDAP_EVENT_STATUS_LIST_V3 to public;
-------------------------------------------------------------------------------------
------------
-- Name: LDAP_NTFY
-- DESCRIPTION: This is the interface to be implemented by provisioning integrated
applications to send information to and receive information from the directory.
The name of the package can be customized as needed. The function and procedure
names within this package should not be changed.

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

CREATE OR REPLACE PACKAGE LDAP_NTFY AS

    -- The Predefined Event Types

    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';

-- The Attribute Type

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

-- The Attribute Modification Type

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

-- The Event dispostions constants

EVENT_SUCCESS            CONSTANT VARCHAR2(32)  :='EVENT_SUCCESS';
EVENT_IN_PROGRESS        CONSTANT VARCHAR2(32)  :='EVENT_IN_PROGRESS';
EVENT_USER_NOT_REQUIRED  CONSTANT VARCHAR2(32)  :='EVENT_USER_NOT_REQUIRED';
EVENT_ERROR              CONSTANT VARCHAR2(32)  :='EVENT_ERROR';
EVENT_ERROR_ALERT        CONSTANT VARCHAR2(32)  :='EVENT_ERROR_ALERT';
EVENT_ERROR_ABORT        CONSTANT VARCHAR2(32)  :='EVENT_ERROR_ABORT';

-- The Actual Callbacks

FUNCTION GetAppEvents (events OUT LDAP_EVENT_LIST_V3)
RETURN NUMBER;

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

着信イベントを処理できない場合、プロビジョニング・サーバーは EVENT_ERROR_ALERT ステータスを戻し、その結果 Oracle Enterprise Manager にトリガーが生成されます。

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の対象とな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 を戻し、不具合があることをアプリケーションに示します。ステータス・イベントは、再度アプリケーションによって処理されます。

EVENT_ERROR は、ディレクトリ操作ではエラーがなかったことを意味します。その他の理由により、イベントは処理されませんでした。

```
-- PutAppEventStatus() : DIP Server invokes this callback in the remote Data
base after processing an event it had received using the GetAppEvents()
callback.  For every event received, the DIP server sends the status event
back after processing the event.  This API will NOT be required by the
Oracle Collaboration Suite release 3.0 components.

PROCEDURE PutAppEventStatus (event_status IN LDAP_EVENT_STATUS_LIST_V3);

-- PutOIDEvents() : DIP Server invokes this API in the remote Database. DIP
server sends event to applications using this callback. It also expects a status
event object in response as an OUT parameter. This API needs to be implemented
by all the Oracle Collaboration Suite release 3.0 components.

PROCEDURE PutOIDEvents (event         IN  LDAP_EVENT_LIST_V3,
                        event_status  OUT LDAP_EVENT_STATUS_LIST_V3);

END LDAP_NTFY;
/
```

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・インタフェースを使用するには、プロビジョニング統合サービス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 パッケージ仕様を使用して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）

プロビジョニング統合サービス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。アプリケーションとプロビジョニング統合サービス間のインタフェースは、表ベースまたは PL/SQL コールバックを使用したものになります。

PL/SQL コールバック・インタフェースを使用するには、プロビジョニング統合サービス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 パッケージ仕様を使用して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 is a notifier interface implemented by Provisioning System
--                clients to receive information about changes in Oracle Internet
--                Directory. The name of package can be customized as needed.
--                The function 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 information to and receive information from
--              Oracle Internet Directory. The name of the package can be
--              customized as needed. The function 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_ERROR       CONSTANT VARCHAR2(32)     := 'EVENT_ERROR';
EVENT_RESEND      CONSTANT VARCHAR2(32)     := 'EVENT_RESEND';
```

コールバック

コールバックとは、通知イベントを送受信するために、プロビジョニング統合サービスによって起動される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。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、コレクションに 2 つの行がある状態で送信されます。

GetAppEvent()

Oracle Directory Integration Platform サーバーは、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 Platform は、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;
```

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 を戻します。GetAppEvent() が再度起動されたときに、アプリケーションはそのイベントを再送する必要があります。

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の対象とな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 を戻し、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 Platform サーバーは、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。受信した各イベントについて、Oracle Directory Integration Platform サーバー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 Platform サーバーは、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、イベント・ステータス・オブジェクトが戻されることを想定します。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、つまり RESEND の場合、Oracle Directory Integration Platform サーバーは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 IV 部

付録

第 IV 部では、Oracle Collaboration Suite でのプロビジョニングのカスタマイズに使用できるプラグインについて説明します。また、この部には、DSML の構文と使用方法に関する付録も含まれます。

- 付録 A「ユーザー・プロビジョニング用の Java プラグイン」
- 付録 B「DSML 構文」
- 付録 C「Netscape LDAP SDK API から Oracle LDAP SDK API への移行」

A

ユーザー・プロビジョニング用の Java プラグイン

この付録では、バージョン 3.0 の Oracle Directory Integration Platform プロビジョニング・サービスの典型的な配置において、プロビジョニング・ポリシー評価、データ検証、データ操作およびイベント配信をカスタマイズするプラグインの使用方法について説明します。

Oracle のプロビジョニング・サーバーは、配置におけるすべてのプロビジョニング要件には対応していません。そのため、ユーザーの作成、変更および削除の様々な段階でフックが提供されます。これらのフックにより、企業では、独自のビジネス・ルールを組み入れ、各要件に合せて情報作成をカスタマイズできます。フックは、Java プラグインの形式を取ります。

この付録では、次の項目について説明します。

- プロビジョニング・プラグイン・タイプとその用途
- プロビジョニング・プラグインの要件
- データ・エントリ・プロビジョニング・プラグイン
- データ・アクセス・プロビジョニング・プラグイン
- イベント配信プロビジョニング・プラグイン
- プロビジョニング・プラグインのリターン・ステータス
- プロビジョニング・プラグインの構成テンプレート
- プロビジョニング・プラグインのサンプル・コード

プロビジョニング・プラグイン・タイプとその用途

プロビジョニング・プラグインには、次の 3 つのタイプがあります。

- データ・エントリ・プラグイン
- データ操作およびデータ・アクセス・プラグイン
- イベント配信プラグイン

データ・エントリ・プラグインは、同期プロビジョニングまたは非同期プロビジョニングを利用するプロビジョニング・フレームワークと統合されるアプリケーションで使用されます。データ・アクセス・プラグインは、同期プロビジョニング用のプロビジョニング・フレームワークと統合されるアプリケーションでのみ使用されます。イベント配信プラグインは、非同期プロビジョニング用のプロビジョニング・フレームワークと統合されるアプリケーションでのみ使用されます。

ディレクトリのベース・ユーザー情報に影響を与える Oracle プロビジョニング・コンソール、Oracle Directory Integration Platform サーバー、およびその他のメカニズムでは、情報の作成時にこれらのプラグインが起動されます。データ・エントリ・プラグインを構成することで、配置では次の操作を実行できます。

- アプリケーション・ユーザーの属性値の検証
- ベース・ユーザーの属性値の検証
- アプリケーション・ユーザーの属性値の拡張
- ベース・ユーザーの属性値の拡張
- プロビジョニング・ポリシーの評価

配置済のアプリケーションでアプリケーション・ユーザー情報を管理するには、そのアプリケーション用にデータ・アクセス・プラグインを構成する必要があります。このプラグイン・タイプを使用すると、ディレクトリ外部で、またはディレクトリ内部の複数エントリとしてアプリケーション情報を管理できます。

データ・エントリ・プラグインとデータ・アクセス・プラグインは、通常、次のいずれかの環境から起動されます。

- Oracle Delegated Administration Services のユーザー・プロビジョニング・コンソール
- Oracle Directory Integration Platform Server
- プロビジョニング API
- バルク・プロビジョニング・ツール

イベント配信プラグインは、Java インタフェース型を備え、プロビジョニング・イベントにサブスクライブするアプリケーションに必要です。同期プロビジョニング対応のアプリケーションには、イベント配信プラグインを実装しないでください。

プロビジョニング・プラグインの要件

アプリケーションに指定するすべてのプラグインは、標準の LDIF テンプレートでディレクトリにアップロードできる JAR ファイルに含まれている必要があります。この例は、「プロビジョニング・プラグインの構成テンプレート」を参照してください。プラグイン・インタフェース定義は、`$ORACLE_HOME/jlib/ldapjclnt10.jar` にあります。詳細は、『Oracle Internet Directory API Reference』およびパブリック・インタフェースを参照してください。アプリケーションに追加の jar ファイルが必要な場合、それらのファイルもアップロードできます。

データ・エントリ・プロビジョニング・プラグイン

データ・エントリ・プラグインの形式には、次の 2 つがあります。

- プレデータ・エントリ・プラグイン
- ポストデータ・エントリ・プラグイン

これらのプラグインのいずれかを使用する場合、`oracle.idm.provisioning.plugin.IdataEntryPlugin` インタフェースを実装する必要があります。このインタフェースには、次の 3 つのメソッドがあります。

```
/**
   * The applications can perform a post data entry operation by
   * implementing this method.
   *
   * @param appCtx the application context
   * @param idmUser the IdmUser object
   * @param baseUserAttr Base user properties
   * @param appUserAttr App user properties
   * @throws PluginException when an exception occurs.
   */
  public PluginStatus process(ApplicationContext appCtx,
     IdmUser idmUser, ModPropertySet baseUserAttr,
       ModPropertySet appUserAttr)throws PluginException;
  /**
   * Returns the Modified Base User properties
   *
   * @return ModPropertySet modified base user properties.
   */
  public ModPropertySet getBaseAttrMods();

  /**
   * Returns the Modified App User properties
   *
   * @return ModPropertySet modified app user properties.
   */
  public ModPropertySet getAppAttrMods();
```

通常、プラグインを実装する開発者は、データ検証またはポリシー評価にこれらのメソッドを使用します。ポリシー評価の場合、ベース・ユーザー属性が評価の決定に使用されます。

アプリケーション・コンテキスト・オブジェクトには、次の情報が含まれます。

- LDAP ディレクトリ・コンテキスト

  アプリケーションでディレクトリ操作を実行する場合、アプリケーション・オブジェクトから LDAP コンテキストを取得できます。この LDAP コンテキストは、プラグインでクローズしないよう注意してください。

- プラグイン・コール・モード

  プラグインは、Oracle プロビジョニング・コンソール、Oracle Directory Integration Platform Server、またはプロビジョニング API を起動するその他の環境からコールされます。コール元の環境が Oracle Directory Integration Platform の場合、プロビジョニング・サービスがプラグインをコールします。モードとして設定可能な値は、`INTERACTIVE_MODE` と `AUTOMATIC_MODE` です。1 番目のモードは、Oracle Delegated Administration Services とクライアント・アプリケーション間の対話を通じてプラグインが起動されたことを示します。2 番目のモードは、ユーザー操作の発生しない Oracle Directory Integration Platform によってプラグインが起動されたことを示します。

- クライアント・ロケール

  プラグインでは、特に Oracle Delegated Administration Services から起動された場合などに、クライアント・ロケールの特定が必要になる可能性があります。

- プラグイン・コール操作

  ユーザーの作成操作と変更操作の両方で、データ・エントリ・プラグインを使用できます。これらのプラグインを同じクラスに実装することも可能です。このような場合、プラグインでどの操作を起動するかを決定する必要があります。アプリケーション・コンテキスト・オブジェクトでは、OP_CREATE および OP_MODIFY という値を使用して操作を識別します。

- プラグイン起動ポイント

  データ・エントリ・プラグインは、通常、ユーザーをアプリケーションにプロビジョニングする必要があるかどうかを決定する際に使用します。プレデータ・エントリ・プラグインまたはポストデータ・エントリ・プラグインにより、ポリシー評価とデータ検証を実行できます。2 つのプラグインのいずれか、または両方を選択できます。両方選択する場合、2 つのプラグインを同じクラスに実装することが可能です。アプリケーション・コンテキスト・オブジェクトを使用して、実際に起動するプラグインを指定します。これには、PRE_DATA_ENTRY および POST_DATA_ENTY という値を使用します。

- コールバック・コンテキスト

  操作にプレ・プラグインとポスト・プラグインの両方を使用し、プレ・プラグインでポスト・プラグインと情報を共有する場合、プレデータ・エントリ・プラグインのアプリケーション・コンテキスト・オブジェクトにコールバック・コンテキストを設定できます。これにより、ポストデータ・エントリ・プラグインでは、このコールバック・コンテキストを取得して使用できます。

- ロギング

  アプリケーション・コンテキスト・オブジェクトで提供されるログ・メソッドを使用して、プラグインの情報を記録できます。

  コール順序は、次のようになります。

  1. Oracle Internet Directory の構成情報オブジェクトに基づいて、プラグイン・オブジェクトをダウンロードしてインスタンス化します。
  2. プラグインに渡されるアプリケーション・コンテキスト・オブジェクトを構成します。
  3. process method() をコールします。
  4. getBaseAttrMods() をコールして、process() で変更されるベース・ユーザー属性を取得します。
  5. プラグインの実行ステータスに応じて、getBaseAttrMods() から戻されたベース・ユーザー属性と元のベース・ユーザー属性をマージします。実行ステータスは、success または failure です。プラグインを実装する開発者は、有効なプラグイン実行ステータス・オブジェクトを戻す必要があります。NULL が戻された場合、実行ステータスは失敗とみなされます。
  6. プラグイン実行ステータスが成功の場合にのみ、ベース・ユーザーのマージが実行されます。
  7. プラグインの getAppAttrMods() をコールします。このメソッドは、process() で変更されるアプリケーション・ユーザー属性を取得します。
  8. プラグインにより戻されるユーザー・プロビジョニング・ステータスに応じて、getAppAttrMods() から戻されたアプリケーション・ユーザー属性と元のアプリケーション・ユーザー属性をマージします。

プレデータ・エントリ・プロビジョニング・プラグイン

プレデータ・エントリ・プラグインは、アプリケーション属性の値を生成します。アプリケーション登録時に指定された属性のデフォルト値が、現在のベース・ユーザー属性とともにこのプラグインに渡されます。Oracle Delegated Administration Services などの対話的な起動環境の場合、戻り値は UI に表示されます。

プレデータ・エントリ・プラグインでは、ユーザーをアプリケーションにプロビジョニングするかどうかを決定できます。プラグインは、ベース・ユーザー属性を調査して決定を行います。プラグインは、作成操作および変更操作中に起動されます。2 つの操作を 1 つのプラグイン・クラスでサポートするか、操作ごとに 1 つのクラスを割り当てることが可能です。

アプリケーションで作成操作と変更操作にプレデータ・エントリ・プラグインを使用する場合、Oracle Internet Directory のアプリケーション・コンテナの下に 2 つの構成エントリを作成する必要があります。最初のエントリは、作成操作用です。

```
dn: cn=PRE_DATA_ENTRY_CREATE, cn=Plugins, cn=FILES, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: oracle.myapp.provisioning.UserCreatePlugin
orclODIPPluginAddInfo: Pre Data Entry Plugin for CREATE operation
```

2 番目のエントリは、変更操作用です。

```
dn: cn=PRE_DATA_ENTRY_MODIFY, cn=Plugins, cn=FILES, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: oracle.myapp.provisioning.UserModifyPlugin
orclODIPPluginAddInfo: Pre Data Entry Plugin for MODIFY operation
```

この例では、作成プラグインと変更プラグインで別々のクラスを使用しています。

ポストデータ・エントリ・プロビジョニング・プラグイン

ポストデータ・エントリ・プラグインは、ユーザーが UI に入力したデータを検証します。また、導出される属性値を生成します。いずれかのアプリケーションでプラグインが失敗すると、UI の処理は停止します。ユーザー・エントリをディレクトリに作成するには、すべてのアプリケーションでデータ検証に成功する必要があります。ただし、非 UI 環境や自動ルーティング処理の場合、プラグインを実装する開発者は、プラグインのコール・モード（INTERACTIVE_MODE または AUTOMATIC_MODE）に基づいてエラーを生成するか処理を継続するかを決定できます。

プレデータ・エントリ・プラグインと同様に、ポストデータ・エントリ・プラグインも、作成操作および変更操作中に起動されます。アプリケーションでは、2 つの操作を 1 つのプラグイン・クラスに実装するか、操作ごとに個別のクラスに実装することが可能です。

作成操作と変更操作にポストデータ・エントリ・プラグインを使用する場合、Oracle Internet Directory のアプリケーション・コンテナの下に 2 つの構成エントリを作成する必要があります。最初のエントリは、作成操作用です。

```
dn: cn=POST_DATA_ENTRY_CREATE, cn=Plugins, cn=FILES, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: oracle.myapp.provisioning.UserMgmtPlugin
orclODIPPluginAddInfo: Post Data Entry Plugin for CREATE and MODIFY
 operations
```

2 番目のエントリは、変更操作用です。

```
dn: cn=POST_DATA_ENTRY_MODIFY, cn=Plugins, cn=FILES, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: oracle.myapp.provisioning.UserMgmtPlugin
orclODIPPluginAddInfo: Post Data Entry Plugin for MODIFY and CREATE operation
```

この例でも、作成プラグインと変更プラグインで別々のクラスを使用しています。

データ・アクセス・プロビジョニング・プラグイン

データ・アクセス・プラグインの主な目的は、ディレクトリにあるユーザーのアプリケーション固有の情報を管理することです。このプラグインを使用して、情報を作成および取得できます。

データ・アクセス・プラグインは、ユーザーが作成されると、Oracle Delegated Administration Services、Oracle Directory Integration Platform またはバルク・プロビジョニング・ツールのいずれかによって起動され、アプリケーションに対するプロビジョニングをリクエストします。

データ・アクセス・プラグインは、変更操作および削除操作中にも起動されます。このプラグインで、アプリケーション情報を更新または削除できます。

データ・アクセス・プラグインを使用する場合、`oracle.idm.provisioning.plugin.IDataAccessPlugin` インタフェースを実装します。このインタフェースは、次のとおりです。

```
/**
 * The applications can create/modify/delete the user footprint by
 * implementing this method.
 *
 * @param appCtx the application context
 * @param idmUser IdmUser object
 * @param baseUserAttr Base user properties
 * @param appUserAttr App user properties
 *
 * @return PluginStatus a plugin status object, which must contain
 * the either <codE>IdmUser.PROVISION_SUCCESS</CODE> or
 * <codE>IdmUser.PROVISION_FAILURE</CODE> provisioning status
 *
 * @throws PluginException when an exception occurs.
 */
public PluginStatus process(ApplicationContext appCtx,
 IdmUser idmUser, ModPropertySet baseUserAttr,
 ModPropertySet ppUserAttr) throws PluginException;

/**
 * The applications can return their user footprint by
 * implementing this method. Use <CODE>
 * oracle.ldap.util.VarPropertySet </CODE>
 * as the return object
 *
 * <PRE>
 *  For Ex.
 *   PropertySet retPropertySet =  null;
 *   retPropertySet = new VarPropertySet();
 *
 *   //Fetch the App data and add it to retPropertySet
 *   retPropertySet.addProperty("name", "value");
 *   ..
```

```
    *   return retPropertySet;
    * </PRE>
    *
    * @throws PluginException when an exception occurs.
    */
   public PropertySet getAppUserData(ApplicationContext appCtx,
         IdmUser user, String reqAttrs[]) throws PluginException;
```

アプリケーションのユーザー情報を管理する場合、ディレクトリのアプリケーション・コンテナの下にプラグイン構成エントリを作成する必要があります。このエントリの例は、次のとおりです。

```
dn: cn=DATA_ACCESS, cn=Plugins, cn=FILES, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: oracle.myapp.provisioning.UserDataAccPlugin
orclODIPPluginAddInfo: Data Access Plugin
```

イベント配信プロビジョニング・プラグイン

イベント配信プラグインの主な目的は、Oracle Directory Integration Platform Server によって通知されるイベントを使用することです。イベントは、Oracle Directory Integration Platform Server によってプラグインに送信されます。イベント型と、アプリケーション・リポジトリで実行されるアクションに基づいて、必要な操作をプラグインで実行します。このプラグインのインタフェース定義は、次のとおりです。

```
/* $Header: IEventPlugin.java 09-jun-2005.12:45:53  *
/* Copyright (c) 2004, 2005, Oracle. All rights reserved.  */
/*
   DESCRIPTION
    All of the plug-in interfaces must extend this common interface.
   PRIVATE CLASSES
    None
   NOTES
    None
*/
package oracle.idm.provisioning.plugin;
/**
 * This is the base interface
 */
public interface IEventPlugin
{
   /**
    * The applications can perform the initialization logic in this method.
    *
    * @param Object For now it is the provisioning Profile that will be passed.
    *        look at oracle.ldap.odip.engine.ProvProfile for more details.
    *
    *
    * @throws PluginException when an exception occurs.
    */
   public void initialize(Object profile) throws PluginException;
   /**
    * The applications can perform the termination logic in this method.
    *
    * @param void Provisioning Profile Object will be sent.
    *             refer to oracle.ldap.odip.engine.ProvProfile for more details
    * @throws PluginException when an exception occurs.
    */
   public void terminate(Object profile) throws PluginException;
```

```
   /**
    * Set Additional Info.
    * Since we pass on the complete profile, there is no requirement to set
    * the additiona
    * @param addInfo Plugin additional info
    */
   //public void setAddInfo(Object addInfo);
}

/* $Header: IEventsFromOID.java 09-jun-2005.12:45:53  */
/* Copyright (c) 2004, 2005, Oracle. All rights reserved.  */
/*
   DESCRIPTION
   Applications interested in receiving changes from OID should
implement this
   interface.
   PRIVATE CLASSES
    <None>
   NOTES
*/
package oracle.idm.provisioning.plugin;
import oracle.idm.provisioning.event.Event;
import oracle.idm.provisioning.event.EventStatus;

/**
 * Applications interested in receiving changes from OID should implement this
 * interface.  The applications register with the OID for the changes occurring
 * at OID. The DIP engine would instantiate an object of this class and invoke
 * the initialize(), sendEventsToApp(), and truncate() method in the same
 * sequence. The initialize method would provide the appropriate information
 * from the profile in the form of a java.util.Hashtable object.
 * The property names, that is, the hash table key that could be used by the
 * interface implementer will be defined as constants in this interface.
 *
 * @version $Header: IEventsFromOID.java 09-jun-2005.12:45:53  $
 */
public interface IEventsFromOID extends IEventPlugin
{

   /**
    * Initialize. The application would provide any initialization logic
    * through method. The DIP engine after instantiating a class that
    * implements this interface will first invoke this method.
    *
    * @param prop A HashMap that would contain necessary information exposed
    * to the applications
    * @throws EventInitializationException the applications must throw this
    * exception in case of error.
    */
   public void initialize(Object provProfile)
                          throws EventPluginInitException;

   /**
    * OID Events are deliverd to the application through this method.
    *
    * @param evts an array of LDAPEvent objects returned by the DIP engine
    * @return the application logc must process these events and return the
    * status of the processed events
    * @throws EventDeliveryException the applications must throw this exception
    * in case of any error.
    */
   public EventStatus[] sendEventsToApp(Event [] evts)
                        throws EventDeliveryException;
```

```
}

/* $Header: IEventsToOID.java 09-jun-2005.12:45:53 $ */
/* Copyright (c) 2004, 2005, Oracle. All rights reserved.  */

/*
   DESCRIPTION
   Applications interested in sending changes to OID should implement this
   interface.
*/
package oracle.idm.provisioning.plugin;
import oracle.idm.provisioning.event.Event;
import oracle.idm.provisioning.event.EventStatus;

/**
 * Applications interested in sending changes to OID should implement this
 * interface.  The applications must register with the OID for the sending
 * changes at their  end to DIP. The DIP engine would instantiate an object
 * of this class and invoke the initialize(), sendEventsFromApp(), and
 * truncate() method in the same sequence. The initialize method would
 * provide the appropriate information from the profile in the form of
 * a java.util.Hashtable object. The property names, that is, the hash table key
 * that could be used by the interface implementer will be defined as
 * constants in this interface.
 *
 */
public interface IEventsToOID extends IEventPlugin
{
   /**
    * Initialize. The application would provide any initialization logic
    * through method. The DIP engine after instantiating a class that
    * implements this interface will first invoke this method.
    *
    * @param  prop ProvProfile
    *              oracle.ldap.odip.engine.ProvProfile
    * @throws EventPluginInitException the applications must throw this
    * exception in case of error.
    */
   public void initialize(Object profile) throws EventPluginInitException;

   /**
    * Application Events are deliverd to OID through this method.
    *
    * @return an array of Event objects returned to be processed by the
    * DIP engine.
    * @throws EventDeliveryException the applications must throw this exception
    * in case of any error.
    */
   public Event[] receiveEventsFromApp()
                       throws EventDeliveryException;

   /**
    * Application can let the DIP engine know whether there are more event to
    * follow through this method
    *
    * @return ture if there are more events to be returned and false otherwise
    * @throws PluginException the applications must throw this exception
    * in case of any error.
    */
   public boolean hasMore() throws PluginException;

   /**
    * The status of the application events are intimated through this method.
```

```
     * i.e the DIP engine after processing the events calls this method to set
     * the event status.
     *
     * @param an array of Event status objects describing the processed event
     * status by the DIP engine.
     * @throws EventDeliveryException the applications must throw this exception
     * in case of any error.
     */
    public void setAppEventStatus(EventStatus[] evtStatus)
                        throws EventDeliveryException;
}
```

プラグインからディレクトリ操作を実行するには、アプリケーション・コンテキストが必要です。イベント配信プラグインの `initialize()` メソッドで `ProvProfile.getApplicationContext()` を使用して、`oracle.idm.provisioning.plugin.ApplicationContext` のインスタンスを取得できます。この `applicationContext` を使用すると、任意のプラグイン・メソッドでディレクトリ操作を実行できます。

プロビジョニング・プラグインのリターン・ステータス

各プロビジョニング・プラグインでは、`oracle.idm.provisioning.plugin.PluginStatus` クラスのオブジェクトを戻す必要があります。このオブジェクトは、実行ステータス（`success` または `failure`）を示します。オブジェクトでは、ユーザー・プロビジョニング・ステータスも戻すことができます。

プロビジョニング・プラグインの構成テンプレート

次に示す LDIF テンプレートは、Oracle Internet Directory 10*g*（10.1.4.0.1）でアプリケーション・プラグインを指定するために使用されます。アプリケーションのディレクトリ・エントリを作成し、プラグインを実装するクラスを含む JAR ファイルをアップロードする必要があります。

```
dn: cn=Plugins, cn=APPTYPE, cn=Applications, cn=Provisioning,
 cn=Directory Integration Platform,cn=Products,cn=OracleContext
changetype: add
add: orclODIPPluginExecData
orclODIPPluginExecData: full_path_name_of_the_JAR_file
objectclass: orclODIPPluginContainer

dn: cn=PRE_DATA_ENTRY_CREATE, cn=Plugins, cn=APPTYPE, cn=Applications,
 cn=Provisioning, cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: Name_of_the_class_that_implements_the_plug-in
orclODIPPluginAddInfo: Pre Data Entry Plugin for CREATE operation

dn: cn=PRE_DATA_ENTRY_MODIFY, cn=Plugins, cn=APPTYPE, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: Name_of_the_class_that_implements_the_plug-in
orclODIPPluginAddInfo: Pre Data Entry Plugin for MODIFY operation

dn: cn=POST_DATA_ENTRY_CREATE, cn=Plugins, cn=APPTYPE, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
```

```
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: Name_of_the_class_that_implements_the_plug-in
orclODIPPluginAddInfo: Post Data Entry Plugin for CREATE and modify operations

dn: cn=POST_DATA_ENTRY_MODIFY, cn=Plugins, cn=APPTYPE, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: Name_of_the_class_that_implements_the_plug-in
orclODIPPluginAddInfo: Post Data Entry Plugin for MODIFY and CREATE operation

dn: cn=DATA_ACCESS, cn=Plugins, cn=APPTYPE, cn=Applications,
 cn=Provisioning,  cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: Name_of_the_class_that_implements_the_plug-in
orclODIPPluginAddInfo: Data Access Plugin

dn: cn=EVENT_DELIVERY_OUT, cn=Plugins, cn=APPTYPE, cn=Applications,
 cn=Provisioning, cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: Name_of_the_class_that_implements_the_plug-in
orclODIPPluginAddInfo: Event Delivery Plugin for Outbound

dn: cn=EVENT_DELIVERY_IN, cn=Plugins, cn=APPTYPE, cn=Applications,
 cn=Provisioning, cn=Directory Integration Platform, cn=Products,
 cn=OracleContext
changetype: add
objectClass: orclODIPPlugin
orclStatus: ENABLE
orclODIPPluginExecName: Name_of_the_class_that_implements_the_plug-in
orclODIPPluginAddInfo: Event Delivery Plugin for Inbound
```

プロビジョニング・プラグインのサンプル・コード

```
/* Copyright (c) 2004, Oracle. All rights reserved. */
/**
 DESCRIPTION
 Sample PRE DATA Entry Plugin for CREATE operation that
 validates the attribute.
 PRIVATE CLASSES
 None.
 NOTES
 This class implements the PRE_DATA_ENTRY_CREATE plugin ONLY
 MODIFIED (MM/DD/YY)
 12/15/04 \226 Creation
*/
package oracle.ldap.idm;

import java.util.*;
import javax.naming.*;
import javax.naming.ldap.*;
import javax.naming.directory.*;
import oracle.ldap.util.*;
import oracle.idm.provisioning.plugin.*;
/**
 * This class implements the PRE_DATA_ENTRY_CREATE plugin ONLY
 *
 */
public class SamplePreDataEntryCreatePlugin implements IDataEntryPlugin
{
   public ModPropertySet mpBaseUser = null;
   public ModPropertySet mpAppUser = null;

   public PluginStatus process(ApplicationContext appCtx,IdmUser idmuser,
    ModPropertySet baseUserAttr, ModPropertySet appUserAttr)
          throws PluginException
   {
     PluginStatus retPluginStatus = null;
     String retProvStatus = null;
     String retProvStatusMsg = null;

     LDIFRecord lRec = null;
     LDIFAttribute lAttr = null;
     String val = null;
     if(null == baseUserAttr.getModPropertyValue(\223departmentNumber\224))
     {
        mpBaseUser = new ModPropertySet();
        mpBaseUser.addProperty("departmentNumber","ST");
        appCtx.log(\223Base user attribute \226 departmentNumber missing\224 +
               \223Setting default - ST\224);
     }
     else if ( baseUserAttr.getModPropertyValue(\223departmentNumber\224)
                .notIn(\223ST\224, \223APPS\224, \224CRM\224) )
     {
        throw new PluginException(\223Invalid department Number\224);
     }
     if((null == appUserAttr) ||
               null == appUserAttr.getModPropertyValue(\223emailQouta\224))
     {
        mpAppUser = new ModPropertySet();
        mpAppUser.addProperty("emailQouta","50M");
        appCtx.log(\223Application user attribute - email Qouta missing \224 +
            \223Setting default - 50M\224);
     }
     return new PluginStatus(PluginStatus.SUCCESS, null, null);
```

```
  }

  public ModPropertySet getBaseAttrMods()
  {
    return mpBaseUser;
  }

  public ModPropertySet getAppAttrMods()
  {
    return mpAppUser;
  }
}

/* Copyright (c) 2004, Oracle. All rights reserved. */
/**
 DESCRIPTION
 Sample POST DATA Entry Plugin for CREATE operation. Implementing a
 policy check to provision only those users who belong to \223SALES\224.
 PRIVATE CLASSES
 None.
 NOTES
 This class implements the POST_DATA_ENTRY_CREATE plugin ONLY
 MODIFIED (MM/DD/YY)
 12/15/04 \226 Creation
*/
package oracle.ldap.idm;

import java.util.*;
import javax.naming.*;
import javax.naming.ldap.*;
import javax.naming.directory.*;
import oracle.ldap.util.*;
import oracle.idm.provisioning.plugin.*;
/**
 * This class implements the POST_DATA_ENTRY_CREATE plugin ONLY
 *
 */
public class SamplePostDataEntryCreatePlugin
{
   public ModPropertySet mpBaseUser = null;
   public ModPropertySet mpAppUser = null;

   public PluginStatus process(ApplicationContext appCtx,IdmUser idmuser,
    ModPropertySet baseUserAttr, ModPropertySet appUserAttr)
          throws PluginException
   {
     PluginStatus retPluginStatus = null;
     String retProvStatus = null;
     String retProvStatusMsg = null;

     if(null == baseUserAttr.getModPropertyValue(\223deptartmentNumber\224))
     {
        mpBaseUser = new ModPropertySet();
        mpBaseUser.addProperty("deptartmentNumber ","SALES");
        appCtx.log("Base user attribute \221c\222 is missing");

        retProvStatus = IdmUser.PROVISION_ REQUIRED;
        retProvStatusMsg =  "Provision policy: Only \221SALES\222\224.
     }
     else if (baseUserAttr.getModPropertyValue(\223deptartmentNumber\224)
             .equals(\223SALES\224))
     {
        retProvStatus = IdmUser.PROVISION_ REQUIRED;
```

```
        retProvStatusMsg =  "Provision policy: Only \221SALES\222\224.
      }
      else
      {
         // do not provision those users who do not belong to SALES.
         retProvStatus = IdmUser.PROVISION_NOT_REQUIRED;
         retProvStatusMsg =
                 "Do not provision the person who is not from \221SALES\222";
       }

       return new PluginStatus(PluginStatus. SUCCESS, retProvStatusMsg,
                                    retProvStatus);
  }

  public ModPropertySet getBaseAttrMods()
  {
    return mpBaseUser;
  }

  public ModPropertySet getAppAttrMods()
  {
    return mpAppUser;
  }
}

/* Copyright (c) 2004, Oracle. All rights reserved. */
/**
 DESCRIPTION
 Sample DATA Access Plugin.
 NOTES
 This class implements the DATA_ACCESS plugin
 MODIFIED (MM/DD/YY)
 12/15/04 \226 Creation
*/
package oracle.ldap.idm;

import javax.naming.*;
import javax.naming.ldap.*;
import javax.naming.directory.*;
import oracle.ldap.util.*;
import oracle.idm.provisioning.plugin.*;
/**
 * This class implements the DATA_ACCESS plugin ONLY
 *
 */
public class SampleDataAccessPlugin
{
   public PluginStatus process(ApplicationContext appCtx,IdmUser idmuser,
        ModPropertySet baseUserAttr,ModPropertySet appUserAttr)
            throws PluginException
   {
      try {
          DirContext dirCtx = appCtx.getDirCtx();
          if ( appCtx.getCallOp().equals(ApplicationContext.OP_CREATE )
          {
            // Use the directory context and create the entry.
          }
          elseif ( appCtx.getCallOp().equals(ApplicationContext.OP_MODIFY)
          {
            // Use the directory context and modify the entry.
          }
      } catch (Exception e) {
         throw new PluginException(e);
```

```
        }
        return new PluginStatus(PluginStatus.SUCCESS, null, null);
    }

    public PropertySet getAppUserData(ApplicationContext appCtx,
           IdmUser idmuser, String [] reqAttrs) throws PluginException
    {
        VarPropertySet vpSet = null;
        DirContext dirCtx = appCtx.getDirCtx();

        try {
          Attributes attrs= dirCtx.getAttributes(\223myAppContainer\224);
          vpSet = new VarPropertySet(); // Populate the VarPropertySet from attrs
        } catch(Exception ne) {
          throw new PluginException(e);
        }
        return vpSet;   }
}
```

B

DSML 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 の利点
- 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。

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。

```
<!- 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。

次に、ディレクトリ・エントリを記述する XML エントリを示します。

```
<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:attr name="givenname">
<dsml:value>Siva V. Kumar</dsml:value></dsml:attr>
<dsml:attr name="cn">
<dsml:value>SVK@oracle.com</dsml:value></dsml:attr>
<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>
```

ディレクトリ属性も同様に記述されます。たとえば、cn 属性の属性定義は次のようになります。

```
<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`

> **関連資料：** 構文と使用方法の詳細は、『Oracle Identity Management ユーザー・リファレンス』の Oracle Internet Directory Server 管理ツールに関する項を参照してください。

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 に変換すると、XML ファイルをアプリケーション・サーバーに格納し、この XML ファイルに対して問合せを行うことができます。LDAP サーバーに対して LDAP 操作を実行する場合と比べ、問合せとレスポンスにかかる時間が短縮されます。

C

Netscape LDAP SDK API から Oracle LDAP SDK API への移行

Oracle Internet Directory SDK C API の詳細は、第 14 章「C API リファレンス」で説明されています。この付録では、コードを移行する際に重要な、Netscape LDAP SDK および Oracle Internet Directory LDAP SDK の違いを説明します。

機能

次に示す Oracle Internet Directory LDAP SDK の機能は、Netscape の SDK 機能と異なります。

- Netscape SDK では、参照を処理するために、クライアントは LDAP リバインド・コールバックを登録する必要があります。Oracle LDAP SDK では自動的に処理されます。
- LDAP 構造へのアクセス方法が異なります。Netscape LDAP SDK の LDAP ハンドルは不透明型です。このハンドル内の個々のフィールドにアクセスするには、アクセサリ機能が必要です。Oracle Internet Directory LDAP SDK では、LDAP 構造が公開されており、クライアントは構造内の個々のフィールドを変更できます。
- Oracle LDAP SDK では、ldap_init() ではなく ldap_open() を使用します。
- Oracle LDAP SDK では、SSL 接続の初期化には異なるファンクション・コールおよびプロシージャが必要です。Oracle Internet Directory の SSL 用のファンクション・コールの詳細は、第 14 章「C API リファレンス」を参照してください。
- Oracle Internet Directory C API は、ライブラリおよびその他のファイルなどの Oracle 環境に依存します。Oracle Application Server または Oracle Database をインストールし、アプリケーションを作成する前に、環境変数 $ORACLE_HOME を適切な場所に設定する必要があります。
- LDAP SDK ユーザーは、メモリーをクリアする calloc() などの割当て機能を使用して、LDAPMod structure() を割り当てる必要があります。
- Oracle Internet Directory API はスレッド・セーフではありません。

ファンクション

次に、Netscape LDAP SDK では使用可能で、Oracle LDAP SDK では使用できない機能を示します。

- Oracle LDAP SDK には、ldap_ber_free() ファンクションはありません。かわりに、ber_free() を使用します。
- Oracle LDAP SDK には、ID エラーおよび一致した文字列を取得するための ldap_get_lderrno() ファンクションはありません。LDAP.ld_matched および LDAP.ld_error フィールドにアクセスすることで、この情報を直接取得できます。ユーザーがアクセスする必要のある LDAP 構造のフィールドは、この 2 つのみです。

マクロ

- Oracle LDAP SDK には LDAPS_PORT は定義されていません。かわりに、LDAP_SSL_PORT を使用します。
- Oracle LDAP SDK には LDAP_AFFECT_MULTIPLE_DSA は定義されていません。これは Nestcape 固有のマクロです。

用語集

3DES

「トリプル・データ暗号化規格」を参照。

ACI

「アクセス制御項目」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。

AES

「高度暗号化規格」を参照。

API

「Application Program Interface」を参照。

Application Program Interface（API）

コンピュータ・アプリケーションと低位レベルのサービスおよび機能（オペレーティング・システム、デバイス・ドライバ、その他のソフトウェア・アプリケーションなど）の間のインタフェースとなる一連のソフトウェア・ルーチンおよび開発ツール。API は、プログラマがソフトウェア・アプリケーションを構築するためのビルディング・ブロックとして機能する。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・コールを通じて Oracle Internet Directory の情報にアクセスする。

ASN.1

Abstract Syntax Notation One（ASN.1）は、情報データの構文定義に使用される国際電気通信連合（ITU）の表記法である。ASN.1 は、構造型の情報、特になんらかの通信メディアを介して伝達される情報を記述するために使用される。インターネット・プロトコルの仕様では、ASN.1 が広く利用されている。

ASR

「Oracle データベース・アドバンスト・レプリケーション」を参照。

Basic Encoding Rules（BER）

ASN.1 で規定されているデータ単位をエンコードするための標準規則。BER は、誤って ASN.1 と対比されることがあるが、ASN.1 は抽象構文の記述言語であり、エンコーディング方式とは異なる。

Basic 認証（basic authentication）

ほとんどのブラウザでサポートされる**認証**プロトコル。Web サーバーは、データ転送で渡されるエンコードされたユーザー名とパスワードを使用してエンティティを認証する。Basic 認証は、平文認証と呼ばれることもある。BASE64 エンコーディングは、一般的に使用できるデコーディング・ユーティリティを使用することで、誰でもデコードできるためである。エンコーディングは、**暗号化**とは異なることに注意。

BER

「**Basic Encoding Rules**」を参照。

Blowfish

DES のより高速な代替方式として 1993 年に Bruce Schneier が開発した**対称型暗号**アルゴリズム。Blowfish は、64 ビットのブロックと最大 448 ビットの鍵を使用した**ブロック暗号**である。

CA

「**認証局**」を参照。

CA 証明書（CA certificate）

認証局は、発行するすべての証明書に自身の**秘密鍵**を使用して署名を行う。対応する認証局（CA）の**公開鍵**は、CA 証明書（ルート証明書）と呼ばれる証明書内にそれ自体が格納される。認証局の秘密鍵で署名されたメッセージを信頼するためには、ブラウザの信頼できるルート証明書のリストに CA 証明書が含まれている必要がある。

CBC

「**暗号ブロック連鎖**」を参照。

CMP

「**証明書管理プロトコル**」を参照。

CMS

「**暗号化メッセージ構文**」を参照。

configset

「**構成設定エントリ**」を参照。

CRL

「**証明書失効リスト**」を参照。

CRMF

「**証明書リクエスト・メッセージ・フォーマット**」を参照。

dads.conf

データベース・アクセス記述子の構成に使用される Oracle HTTP Server 用の構成ファイル。

DAS

「**Oracle Delegated Administration Services**」を参照。

Delegated Administration Services

「**Oracle Delegated Administration Services**」を参照。

DER

「**Distinguished Encoding Rules**」を参照。

DES

「**データ暗号化規格**」を参照。

DIB

「**ディレクトリ情報ベース**」を参照。

Diffie-Hellman

送信者と受信者がセキュアでない通信チャネルを通じて共有秘密を確立できるようにする公開鍵暗号プロトコル。1976 年に初めて公開され、実用的な初の公開鍵暗号システムとなった。

「**対称型アルゴリズム**」も参照。

Directory Manager

「**Oracle Directory Manager**」を参照。

DIS

「**ディレクトリ統合プラットフォーム・サーバー**」を参照。

Distinguished Encoding Rules（DER）

ASN.1 オブジェクトをバイト列でエンコードするための規則セット。DER は、**Basic Encoding Rules** の特殊な例である。

DIT

「**ディレクトリ情報ツリー**」を参照。

DN

「**識別名**」を参照。

Document Type Definition（DTD）

特定の **XML** 文書にとって妥当なタグおよびタグ順序の制約を指定するドキュメント。DTD は、XML の親言語である Simple Generalized Markup Language（SGML）の規則に準拠している。

DRG

「**ディレクトリ・レプリケーション・グループ**」を参照。

DSA

「**デジタル署名アルゴリズム**」または「**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。

DTD

「**Document Type Definition**」を参照。

ECC

「**楕円曲線暗号**」を参照。

ECDSA

「**楕円曲線デジタル署名アルゴリズム**」を参照。

EJB

「**Enterprise JavaBeans**」を参照。

Enterprise JavaBeans（EJB）

Sun 社によって開発された、複数層のクライアント / サーバー・システム用のコンポーネント・アーキテクチャを定義する Java API。EJB システムは Java で記述されるため、プラットフォームに依存しない。また、オブジェクト指向であるため、再コンパイルや構成作業をほとんど必要とせずに既存システムに実装できる。

Enterprise Manager

「**Oracle Enterprise Manager**」を参照。

米国連邦情報処理標準（Federal Information Processing Standards: FIPS）

米国商務省の標準技術研究所（NIST）によって公開された情報処理の標準規格。

FIM

「**連携型 ID 管理**」を参照。

FIPS

「**米国連邦情報処理標準**」を参照。

GET

ログイン URL の一部としてログイン資格証明が送信される認証方式。

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**」を参照。

Hashed Message Authentication Code（HMAC）

秘密のハッシュ関数出力を生成するために使用されるハッシュ関数技術。これにより、MD5 や SHA などの既存のハッシュ関数が強化される。Transport Layer Security（TLS）でも使用される。

HMAC

「**Hashed Message Authentication Code**」を参照。

HTTP

Hyper Text Transfer Protocol（HTTP）は、ドキュメントのリクエストとそのコンテンツの転送のために Web ブラウザとサーバー間で使用されるプロトコルである。その仕様は、World Wide Web Consortium で管理および開発される。

HTTP Server

「**Oracle HTTP Server**」を参照。

httpd.conf

Oracle HTTP Server の構成に使用されるファイル。

iASAdmins

Oracle Application Server のユーザーおよびグループ管理機能を制御する管理グループ。OracleAS Single Sign-On 管理者は、iASAdmins グループのメンバーである。

ID 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。ID 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。

ID 管理インフラストラクチャ・データベース（identity management infrastructure database）

OracleAS Single Sign-On および Oracle Internet Directory のデータを格納するデータベース。

ID 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 ID 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。ID 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。

ID 管理レルム固有の Oracle コンテキスト（identity management realm-specific Oracle Context）

各 ID 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。

- ID 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
- ID 管理レルム内のグループの位置
- ID 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）

ID プロバイダ（identity provider）

ユーザーを認証し、**連携**内でユーザーのデジタル ID 情報を他の関係者に提供する役割を果すエンティティとして、**トラスト・サークル**のメンバーから認識されている組織。ID プロバイダはサービス・プロバイダと提携関係を結び、連携内のすべての関係者が同意した承認済の手順に従ってサービスを提供する。

Internet Directory

「**Oracle Internet Directory**」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。

J2EE

「**Java 2 Platform, Enterprise Edition**」を参照。

Java 2 Platform, Enterprise Edition（J2EE）

Sun 社によって定義された、エンタープライズ・アプリケーションを開発および配置するための環境。J2EE プラットフォームは、サービスのセット、複数の Application Program Interface（API）、および複数層対応の Web ベース・アプリケーションの開発機能を提供する各種プロトコルで構成される。

JavaServer Pages（JSP）

Sun 社によって開発された Java サーブレット・テクノロジの拡張であるサーバー・サイド・テクノロジ。JSP は HTML コードと連携動作する動的スクリプト機能を備えており、ページ・ロジックを静的要素（ページの設計と表示）から分離する。HTML ページに埋め込まれた Java ソース・コードとその拡張機能により、HTML はより効果的に動作する（動的データベース問合せでの使用など）。

JSP

「**JavaServer Pages**」を参照。

LDAP

「**Lightweight Directory Access Protocol**」を参照。

LDAP Data Interchange Format（LDIF）

システム間でディレクトリ・データを交換するためのテキスト・ベースの共通形式。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。

LDAP 接続キャッシュ（LDAP connection cache）

スループットの向上のため、OracleAS Single Sign-On Server では Oracle Internet Directory との接続をキャッシュして再利用する。

LDIF

「**LDAP Data Interchange Format**」を参照。

Liberty Alliance

Liberty Alliance Project は、世界中の 150 を超える企業、非営利団体および政府機関の連合体である。この共同体は、現在と将来のすべてのネットワーク・デバイスをサポートする連携型ネットワーク ID を実現するため、オープンな標準仕様の開発に尽力している。Liberty Alliance は、**連携型 ID 管理**のオープン・テクノロジ標準、プライバシおよびビジネス・ガイドラインを定義および推進するために活動している唯一の国際団体である。

Lightweight Directory Access Protocol（LDAP）

ディレクトリの情報にアクセスするためのプロトコルのセット。LDAP では、任意のタイプのインターネット・アクセスに必要となる TCP/IP がサポートされる。その設計規則のフレームワークは、業界標準のディレクトリ製品（Oracle Internet Directory など）に対応している。LDAP は **X.500** 標準の簡易バージョンであるため、X.500 Light と呼ばれることもある。

MAC

「**メッセージ認証コード**」を参照。

MD2

Message Digest Two（MD2）は、メッセージ・ダイジェスト・**ハッシュ関数**である。このアルゴリズムでは、入力テキストが処理され、メッセージ固有でデータ整合性の検証に使用できる 128 ビットの**メッセージ・ダイジェスト**が生成される。MD2 は RSA Security 社の Ron Rivest によって開発されたもので、メモリーが限られているシステム（スマート・カードなど）で使用されることを意図している。

MD4

Message Digest Four（MD4）は **MD2** に似ているが、特にソフトウェアでの高速処理を目的として設計されている。

MD5

Message Digest Five（MD5）は、メッセージ・ダイジェスト・**ハッシュ関数**である。このアルゴリズムでは、入力テキストが処理され、メッセージ固有でデータ整合性の検証に使用できる 128 ビットの**メッセージ・ダイジェスト**が生成される。MD5 は **MD4** の潜在的な脆弱性が報告された後に、Ron Rivest によって開発された。MD5 は MD4 に似ているが、元のデータに対する操作がより多いため、MD4 より処理が遅い。

MDS

「**マスター定義サイト**」を参照。

mod_osso

一度ユーザーが OracleAS Single Sign-On Server にログインすると、OracleAS Single Sign-On によって保護されるアプリケーションが、ユーザー名とパスワードのかわりに HTTP ヘッダーを受信できるようにする Oracle HTTP Server のモジュール。これらのヘッダーの値は、**mod_osso Cookie** に格納される。

mod_osso Cookie

HTTP サーバーに格納されるユーザー・データ。Cookie は、ユーザーの認証時に作成される。同じユーザーが別のアプリケーションをリクエストした場合、Web サーバーでは、mod_osso Cookie の情報を使用してユーザーがアプリケーションにログインできる。この機能により、サーバーのレスポンス時間が短縮される。

mod_proxy

mod_osso を使用したレガシー・アプリケーション（**外部アプリケーション**）へのシングル・サインオンを可能にする Oracle HTTP Server のモジュール。

MTS

「**共有サーバー**」を参照。

Net Services

「**Oracle Net Services**」を参照。

OASIS

Organization for the Advancement of Structured Information Standards。OASIS は、E-Business 標準の開発、集約および採用を推進する非営利の国際共同団体である。

OC4J

「**Oracle Containers for J2EE**」を参照。

OCA

「**Oracle 認証局**」を参照。

OCI

「**Oracle Call Interface**」を参照。

OCSP

「**Online Certificate Status Protocol**」を参照。

OEM

「**Oracle Enterprise Manager**」を参照。

OID

「**Oracle Internet Directory**」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 Database 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 Internet Directory 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 Platform Server の制御も行う。

Online Certificate Status Protocol（OCSP）

デジタル証明書の有効性をチェックするための 2 つの共通スキームのうちの 1 つ。もう 1 つは、古い方式の**証明書失効リスト**であり、OCSP はいくつかの使用例でその後継となっている。OCSP は、**RFC** 2560 で規定されている。

Oracle Application Server Single Sign-On

OracleAS Single Sign-On は、経費レポート、メール、給付金などのアプリケーションへのセキュアなログインを可能にするプログラム・ロジックで構成される。これらのアプリケーションは、**パートナ・アプリケーション**および**外部アプリケーション**という 2 つの形式を取る。どちらの場合も、一度のみの認証で複数のアプリケーションにアクセスできる。

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 Database 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 CMS

Oracle CMS は、IETF の**暗号化メッセージ構文**プロトコルを実装する。CMS は、セキュアなメッセージ・エンベロープに対応するデータ保護スキームを定義する。

Oracle Containers for J2EE（OC4J）

Java 2 Platform, Enterprise Edition 用の軽量でスケーラブルなコンテナ。

Oracle Crypto

Oracle Crypto は、中核となる暗号化アルゴリズムを提供する Pure Java ライブラリである。

Oracle Delegated Administration Services

Oracle Delegated Administration Services ユニットと呼ばれる個別の事前定義済サービスのセット。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・コンソールを使用することで、Oracle Internet Directory を使用する Oracle アプリケーションおよびサード・パーティ・アプリケーション用の管理ソリューションを簡単に開発して配置できる。

Oracle Directory Integration Platform

Oracle Internet Directory といくつかの関連プラグインおよびコネクタを使用して複数のディレクトリを統合するためのインタフェースとサービスの集合。企業が外部ユーザー・リポジトリを使用して Oracle 製品の認証を受けることを可能にする Oracle Internet Directory の機能。

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 Platform 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実行するデーモン・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 HTTP Server

Hypertext Transfer Protocol（HTTP）を使用する Web トランザクションを処理するソフトウェア。Oracle では、Apache グループにより開発された HTTP ソフトウェアを使用している。

Oracle Identity Management

すべての企業 ID 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。**Lightweight Directory Access Protocol** バージョン 3 と Oracle Database 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 Liberty SDK

Oracle Liberty SDK は、サード・パーティの Liberty 準拠アプリケーション間で連携シングル・サインオンを実現する **Liberty Alliance** Project の仕様を実装する。

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 SDK

Oracle PKI SDK は、**公開鍵インフラストラクチャ**実装内で必要なセキュリティ・プロトコルを実装する。

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 S/MIME

Oracle S/MIME は、セキュアな電子メール環境を実現するため、**Internet Engineering Task Force** の **Secure/Multipurpose Internet Mail Extension** 仕様を実装する。

Oracle SAML

Oracle SAML では、**Security Assertions Markup Language** に関する **OASIS** 仕様の概要のとおり、XML ベースの形式で異種環境システムおよびアプリケーション間でセキュリティ資格証明を交換するためのフレームワークが提供される。

Oracle Security Engine

Oracle Security Engine は、X.509 ベースの証明書管理機能を提供することで Oracle Crypto を拡張する。Oracle Security Engine は Oracle Crypto のスーパーセットである。

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。

『Oracle Advanced Security 管理者ガイド』も参照。

Oracle Web Services Security

Oracle Web Services Security では、Web サービス・セキュリティに関する **OASIS** 仕様の概要のとおり、既存のセキュリティ・テクノロジを使用して認証と認可を行うフレームワークが提供される。

Oracle XML Security

Oracle XML Security は、XML 暗号化および XML 署名に関する W3C 仕様を実装する。

OracleAS Portal

ファイル、イメージ、アプリケーションおよび Web サイトを統合するためのメカニズムを提供する OracleAS Single Sign-On の**パートナ・アプリケーション**。外部アプリケーション・ポートレットにより、外部アプリケーションにアクセスできる。

Oracle コンテキスト（Oracle Context）

「**ID 管理レルム固有の Oracle コンテキスト**」および「**ルート Oracle コンテキスト**」を参照。

Oracle データベース・アドバンスト・レプリケーション（Oracle Database 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 Oracle Database の機能。

Oracle 認証局（Oracle Certificate Authority）

Oracle Application Server Certificate Authority は、Oracle Application Server 環境内で使用される**認証局**である。OracleAS Certificate Authority は、証明書の格納リポジトリとして Oracle Internet Directory を使用する。OracleAS Certificate Authority と OracleAS Single Sign-On および Oracle Internet Directory との統合により、この環境に依存するアプリケーションに対してシームレスな証明書プロビジョニング・メカニズムを提供する。Oracle Internet Directory にプロビジョニングされ、OracleAS Single Sign-On で認証されたユーザーは、OracleAS Certificate Authority のデジタル署名をリクエストできる。

OWM

「**Oracle Wallet Manager**」を参照。

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#1

Public Key Cryptography Standards（PKCS）は、RSA Laboratories によって開発された仕様である。PKCS#1 では、RSA アルゴリズムに基づく公開鍵暗号の実装上の推奨事項が提供される。これには、暗号プリミティブ、暗号化スキーム、署名スキーム、および鍵の表現とスキームの識別を行うための ASN.1 構文が含まれる。

PKCS#10

Public Key Cryptography Standards（PKCS）は、RSA Laboratories によって開発された仕様である。PKCS#10 では、公開鍵、名前および（場合により）属性セットの証明をリクエストするための構文が記述されている。

PKCS#12

Public Key Cryptography Standards（PKCS）は、RSA Laboratories によって開発された仕様である。PKCS#12 では、秘密鍵、証明書、各種の秘密情報、拡張などを含む個人 ID 情報の転送構文が記述されている。この標準をサポートするシステム（ブラウザやオペレーティング・システムなど）により、ユーザーは、通常 **Wallet** と呼ばれる形式で個人 ID 情報の単一セットをインポート、エクスポートおよび使用できる。

PKCS#5

Public Key Cryptography Standards（PKCS）は、RSA Laboratories によって開発された仕様である。PKCS#5 では、パスワード・ベース暗号の実装上の推奨事項が提供される。

PKCS#7

Public Key Cryptography Standards（PKCS）は、RSA Laboratories によって開発された仕様である。PKCS#7 では、デジタル署名やデジタル・エンベロープなど、暗号化を適用できる可能性のあるデータの一般的な構文が記述されている。

PKCS#8

Public Key Cryptography Standards（PKCS）は、RSA Laboratories によって開発された仕様である。PKCS#8 では、複数の公開鍵アルゴリズム用の秘密鍵と属性のセットを含む秘密鍵情報の構文が記述されている。また、暗号化された秘密鍵の構文についても記述されている。

PKI

「**公開鍵インフラストラクチャ**」を参照。

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。

policy.properties

シングル・サインオン・サーバーに必要な基本パラメータを含む Oracle Application Server Single Sign-On 用の多目的構成ファイル。マルチレベル認証など、OracleAS Single Sign-On の拡張機能の構成にも使用される。

POSIX

Portable Operating System Interface for UNIX。オペレーティング・システム間でアプリケーションが移植可能になるようにアプリケーション・ソース・コードを記述する方法を規定したプログラミング・インタフェース標準のセット。一連の標準は、**Internet Engineering Task Force** によって開発されている。

POST

ログイン資格証明がログイン・フォームのボディ内で送信される認証方式。

RC2

Rivest Cipher Two（RC2）は、RSA Security 社の Ronald Rivest により開発された 64 ビットの**ブロック暗号**であり、**データ暗号化規格**の後継方式として設計されている。

RC4

Rivest Cipher Four（RC4）は、RSA Security 社の Ronald Rivest により開発された**ストリーム暗号**である。RC4 では、最大 1024 ビットの可変長の鍵を使用できる。RC4 は、**Secure Sockets Layer** プロトコルを使用する Web サイト間のトラフィックを暗号化してデータ通信を保護するために最もよく使用される。

RDN

「**相対識別名**」を参照。

RFC

Internet Request For Comments（RFC）のドキュメントには、インターネットのプロトコルとポリシーに関する定義が記述されている。Internet Engineering Task Force（IETF）は、新しい標準の議論、開発および確立を推進している。標準は、RFC という頭字語と参照番号を使用して公表される。たとえば、電子メールの公式標準は、RFC 822 である。

RSA

発明者（Rivest、Shamir、Adelman）にちなんで名付けられた**公開鍵暗号**アルゴリズム。RSA アルゴリズムは、最も普及している暗号化および認証アルゴリズムであり、Netscape 社や Microsoft 社の Web ブラウザや、多くの製品の一部として組み込まれている。

RSAES-OAEP

RSA Encryption Scheme - Optimal Asymmetric Encryption Padding（RSAES-OAEP）は、**RSA** アルゴリズムと OAEP 方式を結合した公開鍵暗号化スキームである。Optimal Asymmetric Encryption Padding（OAEP）は、Mihir Bellare と Phil Rogaway によって開発されたメッセージ・エンコードのための方式である。

S/MIME

「**Secure/Multipurpose Internet Mail Extension**」を参照。

SAML

「**Security Assertions Markup Language**」を参照。

SASL

「**Simple Authentication and Security Layer**」を参照。

Secure Hash Algorithm（SHA）

入力に基づいて 160 ビットの**メッセージ・ダイジェスト**を生成する**ハッシュ関数**アルゴリズム。このアルゴリズムは、デジタル署名規格（DSS）で使用される。128、192、256 ビットという 3 つの鍵サイズを提供する高度暗号化規格（AES）の導入により、同レベルのセキュリティ強度を備えた付随ハッシュ・アルゴリズムの必要性が生じた。新しい SHA-256、SHA-284 および SHA-512 ハッシュ・アルゴリズムは、これらの高度な要件に準拠している。

Secure Sockets Layer（SSL）

Netscape 社によって設計されたプロトコルであり、インターネットなどのネットワーク環境において認証された暗号化通信を可能にする。SSL では、RSA の**公開鍵暗号化**システムを使用している（デジタル証明書の使用も含まれる）。SSL は、**機密保護**、**認証**および**整合性**というセキュアな通信の 3 要素を備えている。

SSL は **Transport Layer Security** に改良されている。TLS と SSL は相互運用できないが、TLS で送信されたメッセージは、SSL 対応のクライアントで処理できる。

Secure/Multipurpose Internet Mail Extension（S/MIME）

デジタル署名と**暗号化**を使用して MIME データを保護するための Internet Engineering Task Force（IETF）の標準。

Security Assertions Markup Language（SAML）

インターネットを通じてセキュリティ情報を交換するための **XML** ベースのフレームワーク。SAML により、通常であれば相互運用できない様々なセキュリティ・サービス・システム間で**認証**および**認可**情報を交換できる。SAML 1.0 仕様は、2002 年に **OASIS** によって採用された。

SGA

「**システム・グローバル領域**」を参照。

SHA

「**Secure Hash Algorithm**」を参照。

Signed Public Key And Challenge（SPKAC）

Netscape Navigator ブラウザによる証明書リクエストで使用される独自仕様のプロトコル。

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。

Single Sign-On SDK

OracleAS Single Sign-On のパートナ・アプリケーションのシングル・サインオンを可能にするレガシー API。SDK は、PL/SQL と Java API に加え、各 API の実装方法を示すサンプル・コードで構成される。SDK は現在使用不可となっており、かわりに **mod_osso** が使用される。

Single Sign-On Server

経費レポート、メール、給付金などのシングル・サインオン・アプリケーションへのセキュアなログインを可能にするプログラム・ロジック。

SLAPD

スタンドアロンの LDAP デーモン。レプリケーション以外のほとんどのディレクトリ機能を担当する LDAP ディレクトリ・サーバー・サービス。

SOAP

Simple Object Access Protocol（SOAP）は、**XML** ベースのプロトコルであり、HTTP を通じてインターネット上のシステム間でメッセージをやり取りするためのフレームワークを定義する。SOAP メッセージは、メッセージとその処理方法を記述したエンベロープ、アプリケーション定義によるデータ型のインスタンスを表現するためのエンコード規則セット、およびリモート・プロシージャ・コールとレスポンスを表現するための表記規則という 3 つの部分で構成される。

SPKAC

「**Signed Public Key And Challenge**」を参照。

SSL

「**Secure Sockets Layer**」を参照。

subACLSubentry

アクセス制御リストの情報を含む特定のタイプの**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**。

Time Stamp Protocol（TSP）

Time Stamp Protocol（TSP）は、RFC 3161 に規定されているとおり、デジタル・メッセージのタイムスタンプに関連する参加エンティティ、メッセージ形式および転送プロトコルを定義する。TSP システムでは、信頼できる第三者である時刻認証局（TSA）がメッセージのタイムスタンプを発行する。

TLS

「**Transport Layer Security**」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。

TSP

「**Time Stamp Protocol**」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。

URI

Uniform Resource Identifier（URI）。Web 上の任意の場所にあるコンテンツを識別するための方法。コンテンツは、テキスト・ページ、ビデオ・クリップ、サウンド・クリップ、静止画、動画またはプログラムのいずれでもよい。最も一般的な URI の形式は、**URL** と呼ばれる URI の特定の形式またはサブセットである Web ページ・アドレスである。

URL

Uniform Resource Locator（URL）。インターネット上でアクセス可能なファイルのアドレス。このファイルは、テキスト・ファイル、HTML ページ、イメージ・ファイル、プログラム、または HTTP でサポートされているその他のファイルである。URL には、リソースにアクセスするために必要なプロトコル名、インターネット上の特定のコンピュータを識別するドメイン名、およびコンピュータ上でのファイルの場所を示す階層的な記述が含まれる。

URLC トークン（URLC token）

認証済のユーザー情報を**パートナ・アプリケーション**に渡す OracleAS Single Sign-On コード。パートナ・アプリケーションでは、この情報を使用してセッション Cookie を構成する。

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。

Wallet Manager

「**Oracle Wallet Manager**」を参照。

Web Services Description Language（WSDL）

XML を使用して Web サービスを記述するための標準形式。WSDL 定義では、Web サービスへのアクセス方法と、サービスによって実行される操作の内容が記述されている。

Web サービス（Web service）

HTTP、**XML**、**SOAP** などの標準的なインターネット・プロトコルを使用してアクセスできるアプリケーションまたはビジネス・ロジック。Web サービスでは、コンポーネント・ベース開発の最良の部分と World Wide Web が結合される。Web サービスは、コンポーネントと同様に、サービスの実装方法を考慮することなく使用および再利用できるブラックボックスとして機能する。

WS-Federation

Web Services Federation Language（WS-Federation）は、Microsoft、IBM、BEA、VeriSign および RSA Security の各社によって開発された仕様である。WS-Federation では、関与する **Web サービス**間で ID、属性、認証の信頼情報を承認および仲介することにより、異種または同種のメカニズムを使用するエンティティ間で**連携**を実現するメカニズムを定義している。

「**Liberty Alliance**」も参照。

WSDL

「**Web Services Description Language**」を参照。

X.500

グローバル・ディレクトリの構成方法を定義する、国際電気通信連合（ITU）の標準。X.500 ディレクトリは、各情報カテゴリ（国、都道府県、市区など）に対応する異なるレベルを備えた階層である。

X.509

デジタル証明書を定義するための最も一般的な標準。認証サービスを備えた階層型ディレクトリのための国際電気通信連合（ITU）の標準であり、多くの**公開鍵インフラストラクチャ**実装で使用される。

XML

Extensible Markup Language（XML）は、World Wide Web Consortium（W3C）によって開発された仕様である。XML は、Standard Generalized Mark-Up Language（SGML）の軽量バージョンであり、特に Web ドキュメント用として設計されている。XML は、メタ言語（タグ・セットの定義方法）であり、開発者は多くのドキュメント・クラスに対して独自のカスタム・マークアップ言語を定義できる。

XML 正規化（XML canonicalization: C14N）

2 つの論理的に等価な XML 文書を同一の物理表現に変換するプロセス。デジタル署名では、このプロセスが重要になる。署名は、最初の計算時に基準としたデータと同じ物理表現を基準とした場合にのみ検証できるためである。詳細は、W3C の XML 正規化に関する仕様を参照。

アカウント・ロックアウト（account lockout）

セキュリティ・ポリシー設定に基づいて、一定の時間内にログインを繰返し試行して失敗した場合、ユーザー・アカウントをロックするセキュリティ機能。OracleAS Single Sign-On では、ユーザーが、Oracle Internet Directory により許可されている回数を超えて 1 台以上のワークステーションからアカウントとパスワードの組合せを送信すると、アカウント・ロックアウトが発生する。デフォルトのロックアウト期間は、24 時間である。

アクセス制御項目（Access Control Item: ACI）

アクセス制御情報は、様々なエンティティまたはサブジェクトがディレクトリ内の特定オブジェクトに対して操作を実行するために保持している権限を示す。この情報は、ユーザーが変更可能な操作**属性**として Oracle Internet Directory に格納される。これらの各属性は、アクセス制御項目（ACI）と呼ばれる。ディレクトリ・データへのユーザーのアクセス権限が ACI により決定される。これには、エントリ（構造型アクセス項目）および属性（コンテンツ・アクセス項目）へのアクセスを制御する規則セットが含まれる。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）

ディレクトリ情報ツリーにおいて、そこから下位にあるすべてのエントリにまで適用されるアクセス制御ポリシー情報を含むディレクトリ・エントリ。この情報は、このエントリ自体と、下位にあるすべてのエントリに影響する。Oracle Internet Directory では、ACP を作成して、ディレクトリの**サブツリー**全体にアクセス制御ポリシーを適用できる。

アクセス制御リスト（Access Control List: ACL）

リソースと、コンピュータ・システム内にあるそれらのリソースへのアクセスを許可されているユーザー名のリスト。Oracle Internet Directory では、ACL は、ディレクトリ・オブジェクトに関連付けられた**アクセス制御項目**の**属性値**のリストである。このリストの属性値は、様々なディレクトリ・ユーザー・エンティティ（またはサブジェクト）が特定のオブジェクトに対して保持する権限を示す。

アドバンスト対称型レプリケーション（advanced symmetric replication: ASR）

「**Oracle データベース・アドバンスト・レプリケーション**」を参照。

アドバンスト・レプリケーション（advanced replication）

「**Oracle データベース・アドバンスト・レプリケーション**」を参照。

アプリケーション・サービス・プロバイダ（application service provider）

アプリケーション・サービス・プロバイダ（ASP）は、ソフトウェア・ベースのサービスおよびソリューションを管理して、それらを中央データ・センターから Wide Area Network を通じて顧客に配信するサード・パーティ・エンティティである。つまり、ASP は、企業が情報技術のニーズの一部またはほぼすべてをアウトソーシングするための手段である。

暗号（cipher）

「**暗号化アルゴリズム**」を参照。

暗号化（cryptography）

情報を読取り不可能な形式に変換することで、その情報を保護するプロセス。情報は、データを読取り不可能にする**鍵**を使用して暗号化され、その後、その情報を再度利用する必要が生じたときに復号化される。「**公開鍵暗号**」および「**対称型暗号**」も参照。

暗号化（encryption）

暗号化アルゴリズムの適用により、平文を暗号文に変換するプロセス。

暗号化アルゴリズム（cryptographic algorithm）

可読データ（平文）と可読不能データ（暗号文）を相互に変換するための定義されたプロセス順序。これらの変換には、なんらかの秘密の情報（通常は**鍵**に格納される）が必要とされる。暗号化アルゴリズムの例には、**DES**、**AES**、**Blowfish**、**RSA** などがある。

暗号化証明書（encryption certificate）

電子メッセージ、ファイル、ドキュメント、データ転送を暗号化するため、または同じ目的でセッション鍵を確立するか交換するために使用する**公開鍵**を含む**証明書**。

暗号化メッセージ構文（Cryptographic Message Syntax: CMS）

デジタル・メッセージの署名、ダイジェスト作成、認証および暗号化を行うために **RFC** 3369 で定義されている構文。

暗号スイート（cipher suite）

Secure Sockets Layer 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暗号スイートを確認する。

暗号ブロック連鎖（cipher block chaining: CBC）

ブロック暗号の操作モードの 1 つ。CBC では、一定長の初期化ベクター（IV）として知られる要素を使用する。この方式の主要な特徴の 1 つは、ある暗号文ブロックの復号化が、先行するすべての暗号文ブロックに依存するという連鎖メカニズムの使用である。つまり、先行するすべてのブロックの全妥当性は、直前の暗号文ブロックに含まれている。

暗号文（ciphertext）

暗号文は、適切な**鍵**を保持するエンティティ以外のすべてのエンティティによるデータの読取りを不可能にするため、可読データ（平文）に**暗号化アルゴリズム**を適用した結果、生じるものである。

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**。

「**ハッシュ関数**」も参照。

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 ID 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。

インフラストラクチャ層（infrastructure tier）

ID 管理を担当する Oracle Application Server の複数のコンポーネント。これらのコンポーネントは、OracleAS Single Sign-On、Oracle Delegated Administration Services および Oracle Internet Directory である。

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。

エンドツーエンド・セキュリティ（end-to-end security）

メッセージがビジネス・エンティティ内またはエンティティ間の複数のアプリケーションを横断する際に確立されるもので、そのときのビジネス・エンティティにおける経路全体がセキュアであるというメッセージ・レベル・セキュリティの特性。

エントリ（entry）

個人などのオブジェクトを説明するディレクトリ内の一意のレコード。エントリは、そのエントリ・オブジェクトを記述する**オブジェクト・クラス**の規定に従い、**属性**とそれに関連する**属性値**で構成される。LDAP ディレクトリ構造のすべてのエントリは、**識別名**を介して一意に識別される。

オブジェクト・クラス（object class）

LDAP では、オブジェクト・クラスは情報のグループ化に使用される。通常、オブジェクト・クラスは、個人やサーバーなど、実際の対象物をモデル化したものである。各ディレクトリ・エントリは、1 つ以上のオブジェクト・クラスに属している。オブジェクト・クラスにより、エンティティを構成する属性が決定される。あるオブジェクト・クラスは別のオブジェクト・クラスから導出でき、それによって他のクラスの一部の特性が継承される。

下位 CA（subordinate CA）

階層型の**公開鍵インフラストラクチャ**において、下位**認証局**とは、管理する証明書の署名鍵が他の CA によって保証されており、その活動が他の CA によって制約されている CA である。

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。

外部アプリケーション（external application）

認証を OracleAS Single Sign-On Server に委任しないアプリケーション。かわりに、これらのアプリケーションでは HTML ログイン・フォームが表示され、アプリケーション・ユーザー名とパスワードが要求される。ユーザーは最初のログイン時に、OracleAS Single Sign-On Server が各自の資格証明を取得するように選択できる。その後は、これらのアプリケーションに透過的にログインできる。

外部エージェント（external agent）

Oracle Directory Integration Platform Server に依存しないディレクトリ統合エージェント。Oracle Directory Integration Platform 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）

任意のデータ・ブロックを適切に暗号化および復号化するために必要な秘密情報を含んだデータ構造。鍵のサイズが大きくなると、暗号化されたデータ・ブロックの解読がより困難になる。たとえば、256 ビットの鍵は、128 ビットの鍵よりもセキュリティ強度が高い。

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**」も参照。

仮想 IP アドレス（virtual IP address）

Oracle Application Server Cold Failover Cluster（Identity Management）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（virtual host）

1 つ以上の Web サイトまたはドメインをホスティングする単一の物理 Web サーバー、または他のマシンのプロキシとして機能している（着信リクエストを受信し、それらを適切なサーバーに再ルーティングしている）サーバー。

OracleAS Single Sign-On の場合、仮想ホストは、2 つ以上の OracleAS Single Sign-On Server 間でのロード・バランシングに使用される。また、仮想ホストは、追加のセキュリティ・レイヤーとして機能する。

仮想ホスト名（virtual host name）

Oracle Application Server Cold Failover Cluster（Identity Management）で、特定の仮想 IP アドレスに対応するホスト名。

可読データ（readable data）

暗号化によって暗号文に変換される前のデータ、または復号化によって暗号文から変換された後のデータ。

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で制御される。指定された管理者が、ディレクトリ・**スキーマ**、**アクセス制御リスト**、およびエントリの**属性**とともに、管理領域の各**エントリ**を制御する。

機密保護（confidentiality）

暗号技術では、機密保護（プライバシ保護）とは、権限のないエンティティによるデータの読取りを防止する機能。通常は、**暗号化**を通じて実現される。

キャッシュ（cache）

一般的には、コンピュータ内で迅速にアクセス可能なメモリー容量を示す。ただし、Web 関連では、ブラウザがダウンロード・ファイルや画像を格納するユーザー・コンピュータ上の場所を示すのが普通である。

競合（contention）

リソースの競合。

強制認証（forced authentication）

ユーザーが事前構成された期間中アクティブでない状態が続いた場合に、そのユーザーに再認証を強制する動作。Oracle Application Server Single Sign-On では、グローバル・ユーザーの非アクティブ・タイムアウトを指定できる。これは、機密情報を扱うアプリケーションを含むインストール環境向けの機能である。

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・リクエスト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リクエストを取り出す。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。

クライアント SSL 証明書（client SSL certificates）

Secure Sockets Layer（クライアント認証）を通じてサーバーにクライアント・マシンを保証する際に使用される**証明書**の一種。

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 ID 管理レルムのノード。

グローバリゼーション・サポート（globalization support）

Graphical User Interface（GUI）の複数言語サポート。Oracle Application Server Single Sign-On では、29 の言語がサポートされる。

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。

グローバルに一意なユーザー ID（globally unique user ID）

ユーザーを一意に識別する数値文字列。ユーザー名、パスワード、識別名は変更または追加することが可能であるが、そのユーザーのグローバルに一意なユーザー ID は常に同一である。

グローバル・ユーザーの非アクティブ・タイムアウト（global user inactivity timeout）

ユーザーが事前構成された期間中アクティブでない状態が続いた場合に、そのユーザーに再認証を強制する Oracle Application Server Single Sign-On のオプション機能。グローバル・ユーザーの非アクティブ・タイムアウトは、シングル・サインアウトのセッション・タイムアウトよりもさらに短時間である。

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。

検証（verification）

署名を作成したとみなされる**秘密鍵**に対応する**公開鍵**と、署名の適用先とみなされるデータ・ブロックが存在する場合に、任意の**デジタル署名**が有効であることを確認するプロセス。

コード署名証明書（code signing certificates）

Java プログラム、JavaScript、またはその他の署名済ファイルに署名したエンティティを識別するために使用される**証明書**の一種。

コールド・バックアップ（cold backup）

Oracle Internet Directory では、データベース・コピー・プロシージャを使用して、新規**ディレクトリ・システム・エージェント**・ノードを既存のレプリケート・システムに追加する手順を示す。

公開鍵（public key）

公開鍵暗号で使用される**公開鍵と秘密鍵のペア**のうち、秘密ではない鍵。エンティティは公開鍵を使用してデータを暗号化できるが、このデータを対応する**秘密鍵**で復号化できるのは、公開鍵の所有者のみである。公開鍵は、対応する秘密鍵で作成されたデジタル署名を検証する目的にも使用できる。

公開鍵暗号（public key cryptography）

公開鍵暗号（非対称型暗号とも呼ばれる）では、公開鍵と秘密鍵という 2 つの鍵を使用する。これらの鍵は、鍵のペアと呼ばれる。秘密鍵は非公開のまま維持されるが、公開鍵は任意の関係者に送信できる。秘密鍵と公開鍵には、数学的な関連性がある。秘密鍵で署名されたメッセージは、対応する公開鍵で検証できる。同様に、公開鍵で暗号化されたメッセージは、秘密鍵で復号化できる。この方式では、秘密鍵の所有者のみがメッセージを復号化できるため、プライバシが確保される。

公開鍵暗号化（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。

公開鍵インフラストラクチャ（public key infrastructure: PKI）

公開鍵と**秘密鍵**の発行、配布および認証を管理するシステム。PKI は、一般的に次の要素で構成される。

- **認証局**: デジタル証明書の生成、発行、公開および取消しを担当する。
- **登録局**: 認証局に対する証明書リクエストで提出される情報の検証を担当する。
- ディレクトリ・サービス: 認証局が**証明書**または**証明書失効リスト**を公開し、信頼できる第三者がそれを取得する場所。
- 信頼できる第三者: 認証局から発行された証明書とその中に含まれる**公開鍵**を使用して、**デジタル署名**の検証とデータの暗号化を行う。

公開鍵証明書（public key certificate）

「**証明書**」を参照。

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 Oracle Internet Directory のエントリ。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、**ディレクトリ固有のエントリ**の `subConfigsubEntry` 属性で指定されているサブツリー内で管理される。ディレクトリ固有のエントリ自体は、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**に存在する。

高度暗号化規格（Advanced Encryption Standard: AES）

データ暗号化規格の後継となるよう設計された**対称型暗号**のアルゴリズムの 1 つ。AES は、民間および政府データの暗号化に対応した米国連邦情報処理標準（FIPS）である。

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**識別名**。

サード・パーティ・アクセス管理システム（third-party access management system）

Oracle Application Server アプリケーションへのアクセスに OracleAS Single Sign-On を使用するように変更できる Oracle 以外のシングル・サインオン・システム。

サーバー証明書（server certificate）

セキュアな Web サーバーを使用してデータ提供を行う組織の ID を保証する**証明書**。サーバー証明書は、相互に信頼関係を結んでいる**認証局**によって発行された**公開鍵と秘密鍵のペア**に関連付けられている必要がある。サーバー証明書は、ブラウザと Web サーバー間のセキュアな通信に必要とされる。

サービス時間（service time）

リクエストの開始から、そのリクエストに対するレスポンスの完了までの時間。

サービス・プロバイダ（service provider）

Web ベース・サービスをユーザーに提供するエンティティとして**トラスト・サークル**のメンバーから認められている組織。サービス・プロバイダは、共通のユーザーが**連携**内のすべての関連システムに安全にシングル・サインオンできるようにするため、他のサービス・プロバイダおよび ID プロバイダと提携関係を結ぶ。

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。

サブツリー（subtree）

ディレクトリ階層（**ディレクトリ情報ツリー**とも呼ばれる）の一部分。サブツリーは、通常、特定のディレクトリ・ノードを起点とし、ディレクトリ階層にあるそのノード以下のすべてのサブディレクトリとオブジェクトを含む。

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。リクエスト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）

X.500 識別名（DN）は、ディレクトリ・ツリーにおけるノードの一意名である。識別名は、人物または他のディレクトリ・エントリの一意名を示すために使用される。識別名は、ルート・ノードから名前付きエントリ・ノードまでのパスに沿って、ツリー内の各ノードから選択した属性を連結したものである。たとえば、LDAP の表記法では、オラクル社の米国オフィスで働く John Smith という人物の識別名は、cn=John Smith, ou=People, o=Oracle, c=us のようになる。

思考時間（think time）

ユーザーが実際にプロセッサを使用していない時間。

システム・グローバル領域（System Global Area: SGA）

共有メモリー構造の 1 グループ。1 つの Oracle データベース・インスタンスに関するデータと制御情報が含まれている。複数のユーザーが同じインスタンスに同時に接続した場合、そのインスタンスの SGA 内のデータはユーザー間で共有される。したがって、SGA は共有グローバル領域と呼ばれることもある。バックグラウンド・プロセスとメモリー・バッファの組合せは、Oracle インスタンスと呼ばれる。

システム固有のエージェント（native agent）

Oracle Directory Integration Platform 環境で、ディレクトリ統合プラットフォーム・サーバー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、参照先のディレクトリ・システム・エージェント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。

条件（predicates）

Oracle Application Server Certificate Authority（OCA）において、ポリシー条件とは、ポリシーに適用することで着信証明書リクエストまたは失効に対するポリシーの効果を制限できる論理式である。たとえば、次の条件式では、ou=sales,o=acme,c=us を含む識別名を持つクライアントからのリクエストまたは失効に対し、ポリシーが異なる効果を発揮するよう指定している。

```
Type=="client" AND DN=="ou=sales,o=acme,c=us"
```

証明書（certificate）

公開鍵とその所有者の ID を関連付けるために特別に形式化されたデータ構造。証明書は、認証局により発行される。証明書には、名前、シリアル番号、有効期限、および特定のエンティティの公開鍵が含まれる。証明書は、本物であることを受信者が検証できるように、発行元の認証局によってデジタル署名される。ほとんどのデジタル証明書は、X.509 規格に準拠している。

証明書管理プロトコル（certificate management protocol: CMP）

証明書管理プロトコル（CMP）は、証明書の作成と管理に関連するすべての処理を扱う。CMPは、**公開鍵インフラストラクチャ**の構成要素（**認証局**、**登録局**、証明書の発行を受けるユーザーまたはアプリケーションなど）の間で発生する対話をサポートする。

証明書失効リスト（certificate revocation list: CRL）

発行元の**認証局**によって取り消されたデジタル**証明書**のリスト。

証明書リクエスト・メッセージ・フォーマット（certificate request message format: CRMF）

RFC 2511 仕様に規定されている、**X.509** 証明書のライフ・サイクル管理に関連するメッセージで使用される形式。

証明連鎖（certificate chain）

ユーザー**証明書**とその関連 **CA 証明書**の 1 つ以上のペアを含む順序付けられた証明書のリスト。

シングル・サインオフ（single sign-off）

OracleAS Single Sign-On セッションを終了し、すべてのアクティブなパートナ・アプリケーションから同時にログアウトするプロセス。作業中のアプリケーションからログアウトすることで実行できる。

シングル・サインオン（single sign-on: SSO）

ユーザーが、一度の認証で複数のコンピュータ・プラットフォームまたはアプリケーション・システムにアクセスできるようにするプロセスまたはシステム。

申告（claim）

エンティティによって行われる宣言（名前、ID、鍵、グループなど）。

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 ID。信頼できる証明書は、ID 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。通常、信頼できる証明書は、ユーザー証明書の発行について信頼できる**認証局**から取得する。

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。

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。

ストリーム暗号（stream cipher）

対称型アルゴリズムの一種。ストリーム暗号では、通常はビットやバイトなどの小さな単位にデータを同時分割することで暗号化を行う。また、鍵が常に変化するようなある種のフィードバック・メカニズムの形式を実装する。**RC4** は、ストリーム暗号の一例である。

「**ブロック暗号**」も参照。

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**。リクエスト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リクエストの数。通常、「操作 / 秒」（1 秒当たりの操作件数）で表される。

スレーブ（slave）

「コンシューマ」を参照。

成功 URL（success URL）

Oracle Application Server Single Sign-On を使用している場合、アプリケーションのセッションおよびセッション Cookie の確立を担当するルーチンへの URL を示す。

整合性（integrity）

暗号化において、整合性とは、変更権限を持たないエンティティによってデータが変更されていないかどうかを検出する機能である。

セカンダリ・ノード（secondary node）

Oracle Application Server Cold Failover Cluster（Identity Management）で、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。

「**プライマリ・ノード**」も参照。

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**秘密鍵**。

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 Oracle Database に対応するサービス名、または 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）

ディレクトリ属性は、名前、電話番号、役職名などの特定のデータ要素を保持する。各ディレクトリ・**エントリ**は属性のセットで構成され、各属性はそれぞれ**オブジェクト・クラス**に属する。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**に同じ値が含まれないことを保証する 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）

特定の**エントリ**の**属性**内に格納されている実際のデータ。たとえば、属性の型が `email` の場合、属性値は `sally.jones@oracle.com` などになる。

属性の型（attribute type）

属性の型により、データ要素に関する情報（データ型、最大長、単一値と複数値のいずれを取るかなど）が指定される。属性の型では、値の実際の意味が示され、名前や電子メール・アドレスなどの特定のデータ断片を作成および格納するための規則が指定される。

その他の情報リポジトリ（other information repository）

Oracle Internet Directory 以外のすべての情報リポジトリ。Oracle Directory Integration Platform 環境では、Oracle Internet Directory が**中央ディレクトリ**として機能する。

待機時間（latency）

指定したディレクトリ操作が完了するまでのクライアントの待機時間。待機時間は、空費時間として定義される場合がある。ネットワーク通信では、待機時間は、ソースから宛先へパケットが移動する時間として定義される。

待機時間（wait time）

リクエストの発行からレスポンスの開始までの時間。

ダイジェスト（digest）

「**メッセージ・ダイジェスト**」を参照。

対称鍵（symmetric key）

「**秘密鍵**」を参照。

対称型アルゴリズム（symmetric algorithm）

暗号化と復号化に同じ鍵を使用する暗号化アルゴリズム。対称型（または秘密鍵）アルゴリズムには、基本的に**ストリーム暗号**と**ブロック暗号**という 2 つのタイプがある。

対称型暗号（symmetric cryptography）

対称型暗号（共有秘密暗号）システムでは、データの暗号化と復号化に同じ鍵を使用する。対称型暗号の問題点は、送信者と受信者で秘密鍵を一致させるセキュアな方法を確保する必要があることである。送信中に秘密鍵が第三者によって不正に入手された場合、その鍵を使用して暗号化されたデータは、すべて復号化されてしまう。対称型暗号は、通常、非対称型暗号より処理が高速なため、大量のデータを交換する場合によく使用される。**DES**, **RC2** および **RC4** は、対称型暗号アルゴリズムの例である。

楕円曲線暗号（Elliptic Curve Cryptography: ECC）

RSA 暗号化システムの代替方式であり、大きな数値の因数分解ではなく、楕円曲線上の離散対数問題を解決する場合の困難さに基づいている。Certicom 社によって開発および市販されている ECC は、ワイヤレス・デバイスや PC カードなど、計算能力に限界があり、高速処理が必要とされる環境に特に適している。任意の鍵サイズ（ビット単位）のいずれにおいても、ECC は RSA よりもセキュリティに優れている（鍵なしでの復号化がより困難である）。

楕円曲線デジタル署名アルゴリズム（Elliptic Curve Digital Signature Algorithm: ECDSA）

デジタル署名アルゴリズム標準の楕円曲線版。RSA などのスキームと比較した場合の ECDSA の利点は、鍵長が短く、高速な署名および復号化に対応している点にある。たとえば、160（210）ビットの ECC 鍵は、1024（2048）ビットの RSA 鍵とセキュリティ強度が同じであると考えられており、この利点はセキュリティ・レベルが上がるにつれて大きくなる。

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 Platform 環境では、Oracle Internet Directory が中央ディレクトリになる。

中間層（middle tier）

Oracle HTTP Server および OC4J で構成される OracleAS Single Sign-On インスタンスの一部分。OracleAS Single Sign-On 中間層は、ID 管理インフラストラクチャ・データベースとクライアントの間に位置する。

データ暗号化規格（Data Encryption Standard: DES）

1974 年に IBM 社によって開発され、広く使用されている**対称型暗号**アルゴリズム。DES では、56 ビットの鍵が 64 ビットの各データ・ブロックに適用される。DES と 3DES は、通常、**S/MIME** により暗号化アルゴリズムとして使用される。

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。

「**整合性**」も参照。

データベース・アクセス記述子（database access descriptor: DAD）

OracleAS Single Sign-On スキーマなど、特定の Oracle Application Server コンポーネントのデータベース接続情報。

ディレクトリ（directory）

「**Oracle Internet Directory**」、「**Lightweight Directory Access Protocol**」および「**X.500**」を参照。

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**ディレクトリ情報ツリー**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（**DN**）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ラットフォーム・サーバー（directory integration platform 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。

ディレクトリ統合プロファイル（directory integration profile）

Oracle Directory Integration Platform 環境で、Oracle Directory Integration Platform が外部システムとどのように通信し、何を通信するかを示す Oracle Internet Directory のエントリ。

ディレクトリ・ネーミング・コンテキスト（directory naming context）

「**ネーミング・コンテキスト**」を参照。

ディレクトリ・プロビジョニング・プロファイル（Directory Provisioning Profile）

Oracle Directory Integration Platform がディレクトリ対応アプリケーションに送信するプロビジョニング関連通知の性質を記述した特殊な**ディレクトリ統合プロファイル**。

ディレクトリ・ユーザー・エージェント（directory user agent: DUA）

ディレクトリ・ユーザーのかわりにディレクトリ・サービスにアクセスするソフトウェア。ディレクトリ・ユーザーは、個人または他のソフトウェア要素である。

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。

デジタル証明書（digital certificate）

「**証明書**」を参照。

デジタル署名（digital signature）

デジタル署名は、任意のデータ・ブロックに 2 ステップのプロセスを適用した結果として生じる。まず、データに**ハッシュ関数**が適用されて結果が取得される。次に、その結果が署名者の**秘密鍵**を使用して暗号化される。デジタル署名は、整合性、メッセージ認証、およびデータの否認防止を保証するために使用できる。デジタル署名アルゴリズムの例には、**DSA**、**RSA**、**ECDSA** などが含まれる。

デジタル署名アルゴリズム（Digital Signature Algorithm: DSA）

デジタル署名規格（DSS）の一部として使用される**非対称型アルゴリズム**。デジタル署名アルゴリズムは、暗号化には使用できず、デジタル署名専用である。このアルゴリズムにより、署名者の認証を可能とし、結果的に添付データの整合性も保証する大きな数値のペアが生成される。DSA は、デジタル署名の生成と検証の両方に使用される。

「**楕円曲線デジタル署名アルゴリズム**」も参照。

デフォルト ID 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 ID 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 ID 管理レルムに関連付けられる。

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・レルム位置（default realm location）

デフォルト ID 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）

複数のリクエスト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 Oracle Internet Directory 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。

登録局（Registration Authority: RA）

認証局による証明書の発行前に、ユーザーの検証と登録を行う機関。登録局は、各申請者に、申請された新規証明書の相対識別値または相対識別名を割り当てる。ただし、証明書の署名や発行は行わない。

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。

ドメイン・コンポーネント属性（domain component attribute）

ドメイン・コンポーネント（dc）属性は、ドメイン名から**識別名**を構成する際に使用できる。たとえば、oracle.com というドメイン名を使用すると、dc=oracle, dc=com で始まる識別名を構成して、その識別名をディレクトリ情報のサブツリーのルートとして使用できる。

トラスト・サークル（circle of trust）

Liberty Alliance アーキテクチャおよび運用契約を基礎とする業務関係を確立した**サービス・プロバイダ**と **ID プロバイダ**の**連携**。これにより、ユーザーは、セキュアかつシームレスな環境で業務を遂行できる。

トリプル・データ暗号化規格（Triple Data Encryption Standard: 3DES）

トリプル・データ暗号化規格（3DES）は、1974 年に IBM 社によって開発され、1977 年に米国標準規格として採用された**データ暗号化規格**のアルゴリズムに基づいている。3DES では、3 つの 64 ビット長の鍵（鍵長全体は 192 ビットだが、実際の鍵長は 56 ビット）を使用する。データは、1 番目の鍵で暗号化され、2 番目の鍵で復号化された後、3 番目の鍵で再度暗号化される。このため、3DES の処理は標準の DES と比較して 3 倍遅くなるが、同時にセキュリティ強度も 3 倍になる。

ナレッジ参照（knowledge reference）

リモートの**ディレクトリ・システム・エージェント**（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**ディレクトリ情報ツリー**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）

サービスまたはネットワーク・リソースへのアクセス権を付与または禁止するプロセス。ほとんどのセキュリティ・システムは、2 ステップのプロセスに基づく。第 1 段階は認証であり、ユーザーが自分の身元（ID）を証明する。第 2 段階は認可であり、ユーザー ID と定義された**認可ポリシー**に基づいて、ユーザーが様々なリソースへのアクセスを許可される。

認可ポリシー（authorization policy）

認可ポリシーは、保護されたリソースに対するアクセスの制御方法を示す。ポリシーは、なんらかのシステム・モデルに従って ID およびオブジェクトを権限の集合にマップする。たとえば、ある特定の認可ポリシーでは、ユーザーが販売グループに属する場合のみ販売レポートへのアクセスを許可するよう指定できる。

認証（authentication）

エンティティの申告する ID をその資格証明に基づいて検証するプロセス。ユーザーの認証は、通常、ユーザーの知識または所有物（パスワードや証明書など）に基づいて行われる。

電子メッセージの認証では、ある種のシステム（**公開鍵暗号**など）を使用して、ファイルまたはメッセージがその送信元であると申告する特定の個人または企業から実際に送信されていることを確認し、メッセージのコンテンツに基づいてその内容が送信中に改ざんされていないことをチェックする。

認証局（Certificate Authority: CA）

デジタル**証明書**の発行、更新および取消しを行う信頼できる第三者。認証局は、基本的にエンティティの ID を保証する。また、申請者の検証を**登録局**に委任することもある。著名な認証局としては、Digital Signature Trust 社、Thawte 社、VeriSign 社などがある。

認証プラグイン（authentication plugin）

特定の認証方式の実装。OracleAS Single Sign-On には、パスワード認証、デジタル証明書、Windows システム固有の認証およびサード・パーティ・アクセス管理のための Java プラグインが付属する。

認証レベル（authentication level）

アプリケーションの特定の認証動作を指定できるようにする OracleAS Single Sign-On のパラメータ。このパラメータは、特定の**認証プラグイン**とリンクできる。

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、接続リクエストを開始する。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。

パートナ・アプリケーション（partner application）

認証機能を OracleAS Single Sign-On Server に委任する Oracle Application Server アプリケーションまたは Oracle 以外のアプリケーション。このタイプのアプリケーションは、**mod_osso** ヘッダーを受信することでユーザーの再認証を省略する。

バインド（binding）

ネットワーク環境では、バインドは通信エンティティ間での論理的接続の確立を示す。

Oracle Internet Directory では、バインドはディレクトリの認証を受けるプロセスを示す。

交換のために別のプロトコル（基底プロトコル）の内部または上部で **SOAP** メッセージを転送するための正式な規則セットもバインドと呼ばれる。

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。

「**ハッシュ関数**」も参照。

ハッシュ関数（hash function）

暗号化において、ハッシュ関数または一方向ハッシュ関数は、任意のデータ・ブロックに適用すると一定の値を生成するアルゴリズムである。ハッシュ関数の結果は、任意のデータ・ブロックの整合性を保証するために使用できる。ハッシュ関数がセキュアであるとみなすには、既知のデータ・ブロックおよび既知の結果が与えられたときに、同じ結果を生成する別のデータ・ブロックの作成が非常に困難である必要がある。

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。

非対称型アルゴリズム（asymmetric algorithm）

暗号化と**復号化**で異なる**鍵**を使用する**暗号化アルゴリズム**。

「**公開鍵暗号**」も参照。

非対称型暗号（asymmetric cryptography）

「**公開鍵暗号**」を参照。

否認防止（non-repudiation）

暗号化において、特定の**デジタル署名**が特定のエンティティの**秘密鍵**で生成されたことと、メッセージが任意の時点で改ざんされずに送信されたことを証明する機能。

秘密鍵（private key）

公開鍵暗号で使用される**公開鍵と秘密鍵のペア**に含まれる非公開の鍵。任意のエンティティでは、この秘密鍵を使用することで、対応する**公開鍵**で暗号化されたデータを復号化できる。また、エンティティでは、秘密鍵を使用して**デジタル署名**を作成することもできる。エンティティの公開鍵で暗号化されたデータと、秘密鍵で作成された署名のセキュリティは、非公開の秘密鍵に依存する。

秘密鍵（secret key）

対称型アルゴリズムで使用される**鍵**。秘密鍵は、暗号化と復号化の両方に使用されるため、相互に暗号文をやり取りする関係者間で共有されるが、その他の権限のないエンティティには秘密にしておく必要がある。

秘密鍵暗号（private key cryptography）

「**対称型暗号**」を参照。

秘密鍵暗号（secret key cryptography）

「**対称型暗号**」を参照。

平文（plaintext）

暗号化を使用して暗号文に変換される前の可読データ、または復号化を使用して暗号文から変換された後の可読データ。

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。

フィルタ（filter）

ディレクトリに対するリクエストまたは検索から戻されるエントリを定義する式。フィルタは、通常、`cn=susie smith,o=acme,c=us` などの識別名で表現される。

フェイルオーバー（failover）

障害の認識とリカバリのプロセス。Oracle Application Server Cold Failover Cluster（Identity Management）では、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）

Oracle Application Server Cold Failover Cluster（Identity Management）で、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。

「**セカンダリ・ノード**」も参照。

プロキシ・サーバー（proxy server）

Web ブラウザなどのクライアント・アプリケーションと実際の宛先サーバー間に存在するサーバー。プロキシ・サーバーは、実際の宛先サーバーに対するすべてのリクエストに割り込み、プロキシ・サーバーでそれらのリクエストに対応できるかどうかを判断する。対応できない場合、リクエストは実際の宛先サーバーに転送される。OracleAS Single Sign-On では、ロード・バランシングと追加のセキュリティ・レイヤーのためにプロキシが使用される。

「**ロード・バランサ**」も参照。

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。

ブロック暗号（block cipher）

対称型アルゴリズムの一種。ブロック暗号では、固定サイズ（通常 64 ビット）のブロックにメッセージを分割し、各ブロックを鍵で暗号化することによりメッセージを暗号化する。よく知られているブロック暗号には、**Blowfish**、**DES**、**AES** などがある。

「**ストリーム暗号**」も参照。

プロビジョニング（provisioning）

ユーザーに対し、エンタープライズ環境で使用する可能性のあるアプリケーションや他のリソースへのアクセスを提供するプロセス。

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。

プロビジョニング統合プロファイル（provisioning integration profile）

Oracle Directory Integration Platform がディレクトリ対応アプリケーションに送信するプロビジョニング関連通知の性質を記述した特殊な**ディレクトリ統合プロファイル**。

プロファイル（profile）

「**ディレクトリ統合プロファイル**」を参照。

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。

ポリシー優先順位（policy precedence）

Oracle Application Server Certificate Authority（OCA）では、メインのポリシー・ページに表示されている順序で着信リクエストにポリシーが適用される。OCA のポリシー・プロセッサ・モジュールでポリシーを解析する場合、ポリシー・リストの上位に表示されているポリシーが最初にリクエストに適用される。リストの下位に表示されているポリシーは最後に適用され、他のポリシーを上書きする。有効なポリシーのみが着信リクエストに適用される。

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。

メッセージ・ダイジェスト（message digest）

ハッシュ関数の結果。

「**ハッシュ**」も参照。

メッセージ認証（message authentication）

特定のメッセージが特定のエンティティから送信されたことを検証するプロセス。

「**認証**」も参照。

メッセージ認証コード（message authentication code: MAC）

メッセージ認証コード（MAC）は、任意のデータ・ブロックに 2 ステップのプロセスを適用した結果として生じる。まず、**ハッシュ関数**の結果が取得される。次に、その結果が**秘密鍵**を使用して暗号化される。MAC は任意のデータ・ブロックのソースの認証に使用できる。

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 ID 管理レルムのノード。

ユーザー名マッピング・モジュール（user name mapping module）

ユーザー**証明書**をユーザーのニックネームにマップする OracleAS Single Sign-On の Java モジュール。このニックネームが認証モジュールに渡されると、認証モジュールは、ニックネームを使用してディレクトリからユーザーの証明書を取得する。

猶予期間ログイン（grace login）

パスワード期限切れ前の指定された期間内に行われるログイン。

リモート・マスター・サイト（remote master site: RMS）

レプリケート環境における**マスター定義サイト**以外のサイトで、**Oracle データベース・アドバンスト・レプリケーション**のメンバーであるサイト。

リレーショナル・データベース（relational database）

構造化されたデータの集合。同一の列のセットを持つ 1 つ以上の行で構成される表にデータが格納される。Oracle では、複数の表のデータを容易にリンクできる。このため、Oracle はリレーショナル・データベース管理システム、つまり RDBMS と呼ばれる。Oracle はデータを複数の表に格納し、さらに表間の関係を定義できる。このリンクは両方の表に共通の、1 つ以上のフィールドに基づいて行われる。

ルート CA（root CA）

階層型の**公開鍵インフラストラクチャ**において、ルート**認証局**とは、管理する**公開鍵**がセキュリティ・ドメインで最も信頼できるデータとなる認証局である。

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 ID 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 ID 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: DSE）

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。

レガシー・アプリケーション（legacy application）

認証を OracleAS Single Sign-On Server に委任するように変更できない古いアプリケーション。**外部アプリケーション**とも呼ばれる。

レジストリ・エントリ（registry entry）

Oracle Internet Directory サーバー（**ディレクトリ・サーバー・インスタンス**と呼ばれる）の起動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。

レスポンス時間（response time）

リクエストの発行からレスポンスの完了までの時間。

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（realm）

「**ID 管理レルム**」を参照。

レルム検索ベース（realm search base）

すべての **ID 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プする際に使用される。

連携（federation）

ユーザー・ベースを共有しており、ユーザーが一度ログオンすることで**トラスト・サークル**内のすべてのサービスにアクセスできるような ID と認証トークンの提供に合意しているエンティティ（会社および組織）のグループ。連携内では、最低 1 つのエンティティが、ユーザーの認証を担当する **ID プロバイダ**の役割を果す。ユーザーにサービスを提供するエンティティは、**サービス・プロバイダ**と呼ばれる。

連携型 ID 管理（federated identity management: FIM）

自律型ドメイン全体で ID および資格の移行を可能にする契約、標準および技術。FIM により、認証済のユーザーは複数のドメイン全体で認識され、パーソナライズされたサービスを利用できる。このため、個人情報が中央記憶域に集中することなく、ID 情報を異なるアカウント間でリンクできる。連携型 ID には、信頼と標準という 2 つの主要な構成要素が必要とされる。連携型 ID 管理の信頼モデルは、**トラスト・サークル**に基づく。標準は、**Liberty Alliance** Project によって定義される。

ロード・バランサ（load balancer）

高負荷の解消またはフェイルオーバーを理由として、2 つ以上のサーバー間で接続リクエストを分散するハードウェア・デバイスおよびソフトウェア。よく知られているハードウェア・デバイスには、BigIP、Alteon、Local Director などがある。Oracle Application Server Web Cache は、ロード・バランシング・ソフトウェアの一例である。

論理ホスト（logical host）

Oracle Application Server Cold Failover Cluster（Identity Management）における、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

A

ACI,「アクセス制御情報項目（ACI)」を参照
ACL,「アクセス制御リスト（ACL)」を参照

C

C API
  SSL モードでの使用方法，14-41
  概要，14-4
  使用例，14-41
  非 SSL モードでの使用方法，14-42
  ファンクション
    abandon，14-30
    abandon_ext，14-30
    add，14-26
    add_ext_s，14-26
    add_s，14-26
    compare，14-21
    compare_ext，14-21
    compare_ext_s，14-21
    compare_s，14-21
    count_entries，14-36
    count_references，14-36
    count_values，14-38
    count_values_len，14-38
    delete，14-27
    delete_ext，14-27
    delete_ext_s，14-27
    delete_s，14-27
    dn2ufn，14-39
    err2string，14-33
    explode_dn，14-39
    explode_rdn，14-39
    extended_operation，14-29
    extended_operation_s，14-29
    first_attribute，14-37
    first_entry，14-36
    first_message，14-35
    first_reference，14-36
    get_dn，14-39
    get_entry_controls，14-40
    get_option，14-7
    get_values，14-38
    get_values_len，14-38
    init_ssl コール，14-2
    modify，14-22
    modify_ext，14-22
    modify_ext_s，14-22
    modify_s，14-22
    msgid，14-31
    msgtype，14-31
    next_attribute，14-37
    next_entry，14-36
    next_message，14-35
    next_reference，14-36
    parse_extended_result，14-33
    parse_reference，14-40
    parse_result，14-33
    parse_sasl_bind_result，14-33
    rename，14-24
    rename_s，14-24
    result，14-31
    sasl_bind，14-11
    sasl_bind_s，14-11
    search_st，14-17
    set_option，14-7
    simple_bind，14-11
    simple_bind_s，14-11
    unbind_ext，14-17
    unbind_s，14-17
    value_free，14-38
    value_free_len，14-38
C API の使用例，14-41
CONNECT_BY コントロール，3-10

D

DAS URL パラメータ，8-4，18-3
DAS URL パラメータの説明，18-6
DAS ユニット，8-2
DBMS_LDAP_UTL
  グループ関連サブプログラム
    create_group_handle ファンクション，17-16
    get_group_dn ファンクション，17-19
    get_group_properties ファンクション，17-17
    set_group_handle_properties ファンクション，17-16
    概要，17-2
  サブスクライバ関連サブプログラム
    create_subscriber_handle ファンクション，17-20
    get_subscriber_dn ファンクション，17-22
    get_subscriber_properties ファンクション，17-21
    概要，17-2
  その他のサブプログラム
    check_interface_version ファンクション，17-32

create_mod_propertyset ファンクション，17-29
free_handle プロシージャ，17-31
free_mod_propertyset プロシージャ，17-31
free_propertyset_collection プロシージャ，17-29
get_property_names ファンクション，17-26
get_property_values_len ファンクション，17-28
get_property_values ファンクション，17-27
normalize_dn_with_case ファンクション，17-25
populate_mod_propertyset ファンクション，17-30
データ型，17-36
ファンクション・リターン・コード，17-34
ユーザー関連サブプログラム
authenticate_user ファンクション，17-4
check_group_membership ファンクション，17-12
create_user_handle ファンクション，17-5
get_group_membership ファンクション，17-14
get_user_dn ファンクション，17-11
get_user_extended_properties ファンクション，17-10
get_user_properties ファンクション，17-7
locate_subscriber_for_user ファンクション，17-13
set_user_handle_properties ファンクション，17-6
set_user_properties ファンクション，17-8
概要，17-2
DBMS_LDAP_UTL PL/SQL リファレンス，17-1
DBMS_LDAP パッケージ
使用した検索，2-12
DES40 暗号化，2-6
DN，「識別名」を参照

G

GET 認証方式，9-9

H

HTTP ヘッダー，9-2

J

J2EE セキュリティ API，10-2
JAAS ポリシー管理 API，10-5
Java，1-4，2-8
Java API リファレンス
クラスの説明
PropertySetCollection クラス，5-2
PropertySet クラス，5-2
Property クラス，5-2
Java パートナ・アプリケーション
静的な保護，9-7
動的な保護，9-7
Java パートナ・アプリケーション , 静的な保護，9-6
Java プラグイン
設定，13-2
Java プラグイン API，13-3 〜 13-13
JAZN
「Oracle Application Server Java Authentication and Authorization Service」を参照
JNDI，1-4，2-8
JNDI の場所，16-1

L

LDAP
機能モデル，2-4
情報モデル，2-3
セキュリティ・モデル，2-4
セッション
初期化，2-8
セッション・ハンドル・オプション，14-7
C API，2-10
操作 , 実行，14-17
ネーミング・モデル，2-2
バージョン 2 C API，14-2
メッセージ , 結果の取得と確認，14-31
歴史，2-2
LDAP API，1-7
ldapadd
プラグイン・サポート，12-17 〜 12-18
ldap-bind 操作，2-5
ldapcompare
プラグイン・サポート，12-19 〜 12-22
ldapmodify
プラグイン・サポート，12-14 〜 12-16
LDAP 機能モデル，2-4
LDAP 情報モデル，2-3
LDAP セキュリティ・モデル，2-4
LDAP の歴史，2-2
LDAP モデル，2-2
LDAP ネーミング・モデル，2-2
LDAP モデルの概要，2-2

M

mod_osso，9-12
Single Sign-On SDK との比較，9-2
サンプル・アプリケーション，9-4
定義，9-2
統合方法，9-3
利点，9-2
mod_osso Cookie，9-10

O

OC4J セキュリティ API，10-2
OpenLDAP Community，xxiii
Oracle Application Server Java Authentication and Authorization Service
定義，1-2
Oracle Directory Manager，1-10
Oracle Directory Replication Server，1-10
Oracle Identity Management
アプリケーションの統合，1-1
サポートされているサービス，1-2
利点，1-2
インフラストラクチャ
既存のアプリケーションの変更，1-3
統合
新しいアプリケーション，1-3
Oracle Internet Directory, コンポーネント，1-10
Oracle Internet Directory サーバー，1-10
Oracle Internet Directory の Java API，16-1
Oracle SSL 関連ライブラリ，14-46
Oracle SSL コール・インタフェース，14-2

Oracle SSL の拡張機能，14-2
Oracle Wallet，14-3
Oracle Wallet Manager，14-3
　Wallet を作成するために必要，14-46
OracleAS Single Sign-On
　ユーザー属性，9-2
Oracle システム・ライブラリ，14-46
Oracle の拡張機能
　Java 言語のプログラム抽象化，5-1，6-1
　LDAP 対応アプリケーションの外観，1-5
　アプリケーション
　　起動とブートストラップに関するロジック，1-6
　　削除ロジック，1-6
　　実行時のロジック，1-6
　　停止ロジック，1-6
　グループ管理機能，4-4
　プログラム抽象化
　　Java 言語，5-1，6-1
　　PL/SQL 言語，6-2
　ユーザー管理機能，5-1，6-1

P

PL/SQL API，15-1
　C API の一部を含む，2-8
　概要，15-2
　サブプログラム，15-6
　データ型の概要，15-5
　データベースへのロード，2-8
　ファンクション
　　add_s，15-31
　　ber_free，15-38
　　bind_s，15-8
　　compare_s，15-10
　　count_entries，15-16
　　count_values，15-33
　　count_values_len，15-33
　　create_mod_array，15-24
　　dbms_ldap.init，15-7
　　delete_s，15-22
　　err2string，15-24
　　explode_dn，15-35
　　first_attribute，15-17
　　first_entry，15-14
　　get_dn，15-19
　　get_values，15-20
　　get_values_len，15-21
　　init，15-6
　　modify_s，15-30
　　modrdn2_s，15-23
　　msgfree，15-37
　　next_attribute，15-18
　　next_entry，15-15
　　open_ssl，15-36，15-37，15-38
　　rename_s，15-34
　　search_s，15-11
　　search_st，15-12
　　simple_bind_s，15-7
　　unbind_s，15-9
　プロシージャ
　　free_mod_array，15-32
　　populate_mod_array（バイナリ・バージョン），15-26
　　populate_mod_array（文字列バージョン），15-25
　例外の概要，15-4
POST 認証方式，9-9

R

RC4_40 暗号化，2-6
RDN，「相対識別名」を参照
RFC 1823，14-46

S

SDK コンポーネント，1-4
Secure Sockets Layer（SSL）をサポートする Oracle の拡張機能，14-2
Single Sign-On SDK
　mod_osso との比較，9-2
Smith, Mark，xxii，xxiii
SSL
　Oracle の拡張機能，14-2
　　暗号化と復号化，14-2
　Wallet，14-3
　インタフェース・コール，14-2
　クライアントとサーバーの認証，2-5
　サーバー認証，2-5
　デフォルト・ポート，2-5
　認証なし，2-5
　認証モード，14-2
　ハンドシェイク，14-2
SSO ログイン名
　検索，5-5

T

TCP/IP ソケット・ライブラリ，14-46

U

URL, 保護，9-3

W

Wallet
　SSL，14-3
　サポート，14-3

あ

アクセス制御，2-4，2-6
　認可，2-6
アクセス制御情報項目（ACI），2-6
　属性，2-6
　ディレクティブ
　　書式，2-6
アクセス制御リスト（ACL），2-6
アプリケーション , 作成
　C API を使用，14-45
アプリケーション・コンテキスト
　プロビジョニング・プラグイン，A-10
アプリケーション・セッション Cookie
　クリア，9-10
　コーディング，9-10
アプリケーションのログアウト，9-12

アプリケーションのログイン，9-11
暗号化
　DES40，2-6
　Oracle Internet Directory で使用可能なレベル，2-6
　RC4_40，2-6

い

依存性と制限事項，14-46
　C API，14-46
インタフェース・コール , SSL，14-2

え

エラー
　処理と結果の解析，14-32
エントリ
　識別名，2-2
　識別名を使用して位置を識別，2-3
　ネーミング，2-2
　読取り，14-20
エントリの子 , リスト表示，14-20

か

階層検索，3-9
簡易認証，2-5
関連ドキュメント，xxii

き

強制認証，9-11

く

クライアントとサーバーの認証 , SSL，14-2
グローバル・ユーザーの非アクティブ・タイムアウト，9-9

け

結果 , リストの参照，14-35
権限，2-4，2-6
検索
　階層，3-9
　結果
　　解析，14-36
　有効範囲，2-14
検索関連操作 , 流れ，2-12
厳密認証，2-5

こ

コード例
　アプリケーションのログイン，9-11
　強制認証，9-11
　シングル・サインオフ，9-8
　認証，9-7，9-8
コントロール , 使用，3-8，3-9，14-15
コンポーネント
　Oracle Internet Directory SDK，1-4

さ

サーバー認証の SSL，2-5，14-2
サービス位置レコード，4-4
サーブレット
　静的な保護，9-6，9-7
　動的な保護，9-7

し

識別名，2-2
　コンポーネント，2-3
　書式，2-3
証明書，2-5
証明書ベースの認証，2-5
書式 , 識別名，2-3
シングル・サインオフ，9-8

す

スタティック・ディレクティブ
　記述，9-3
　定義，9-3

せ

整合性 , データ，2-6
セキュリティ , Oracle Internet Directory 環境，2-4
セキュリティ API，10-2
セッション
　DBMS_LDAP を使用した終了の有効化，2-17
　クローズ，14-17
　初期化
　　C API を使用，2-9
　　DBMS_LDAP を使用，2-9
セッション固有のユーザー ID，2-5
セルフ・サービス・コンソール，8-2

そ

操作属性
　ACI，2-6
操作の中止，14-30
相対識別名，2-3
属性
　値，2-3
　型，2-3
属性の型，2-3

た

ダイナミック・ディレクティブ
　一般的なタイプ，9-3
　サポートするプログラミング言語，9-3
　定義，9-3

て

ディレクティブ，2-6
ディレクトリ・サーバー検出，4-4
ディレクトリ情報ツリー，2-2
ディレクトリ情報ツリー（DIT），2-2

ディレクトリ操作
　プロビジョニング・プラグイン，A-10
データ
　整合性，2-4，2-6
　プライバシ，2-4，2-6
データ型の概要，15-5

と

動的パスワード・ベリファイア
　コントロール，3-8，3-9
　作成，3-7 ～ 3-9
　パラメータ，3-8
ドキュメント , 関連，xxii
匿名認証，2-5

に

認可，2-4，2-6
認可 ID，2-5
認証，2-4，2-5
　SSL，2-5，14-2
　　クライアントとサーバー，14-2
　　サーバー，14-2
　　認証なし，14-2
　SSL クライアントとサーバー，2-5
　SSL サーバー，2-5
　オプション，2-5
　厳密，2-5
　証明書ベース，2-5
　ディレクトリ，14-11
　ディレクトリ・サーバー
　　有効化，2-10
　　有効化 , C API を使用，2-11
　　有効化 , DBMS_LDAP を使用，2-11
　匿名，2-5
　パスワード・ベース，2-5
　モード , SSL，14-2
認証 , 簡易，9-7
認証局，2-5

ね

ネーミング・エントリ，2-2

は

パスワード
　ポリシー，2-6
パスワード・ベースの認証，2-5
バルク・ツール，1-10

ふ

フィルタ，2-14
プライバシ , データ，2-4，2-6
プラグイン
　PL/SQL
　　バイナリ・サポート，12-14 ～ 12-22
　プロビジョニング・インタフェース，A-1
プロシージャ , PL/SQL
　free_mod_array，15-32
　populate_mod_array（バイナリ・バージョン），15-26
　populate_mod_array（文字列バージョン），15-25
プロビジョニング・インタフェース・プラグイン，A-1
プロビジョニング・プラグイン
　アプリケーション・コンテキストの取得，A-10
　ディレクトリ操作，A-10

へ

ヘッダー・ファイルとライブラリ , 必要，14-45

ほ

ポリシー管理 API，10-5

も

モジュール
　mod_osso，9-12

ゆ

ユーザー属性，9-2

れ

例外の概要，15-4

ろ

ログイン名
　検索，5-5