Oracle® Virtual Directory

製品マニュアル

10*g*（10.1.4.0.1）

部品番号 : B31396-01

2006 年 11 月

ORACLE®

Oracle Virtual Directory 製品マニュアル , 10*g*（10.1.4.0.1）

部品番号 : B31396-01

原本名 : Oracle Virtual Directory Product Manual, 10*g* (10.1.4.0.1)

原本部品番号 : B28833-01

制限付権利の説明

このプログラム（ソフトウェアおよびドキュメントを含む）には、オラクル社およびその関連会社に所有権のある情報が含まれています。このプログラムの使用または開示は、オラクル社およびその関連会社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権と工業所有権に関する法律により保護されています。

独立して作成された他のソフトウェアとの互換性を得るために必要な場合、もしくは法律によって規定される場合を除き、このプログラムのリバース・エンジニアリング、逆アセンブル、逆コンパイル等は禁止されています。

このドキュメントの情報は、予告なしに変更される場合があります。オラクル社およびその関連会社は、このドキュメントに誤りが無いことの保証は致し兼ねます。これらのプログラムのライセンス契約で許諾されている場合を除き、プログラムを形式、手段（電子的または機械的）、目的に関係なく、複製または転用することはできません。

このプログラムが米国政府機関、もしくは米国政府機関に代わってこのプログラムをライセンスまたは使用する者に提供される場合は、次の注意が適用されます。

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

このプログラムは、核、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションへの用途を目的としておりません。このプログラムをかかる目的で使用する際、上述のアプリケーションを安全に使用するために、適切な安全装置、バックアップ、冗長性（redundancy）、その他の対策を講じることは使用者の責任となります。万一かかるプログラムの使用に起因して損害が発生いたしましても、オラクル社およびその関連会社は一切責任を負いかねます。

Oracle、JD Edwards、PeopleSoft、Octet String、Octet String ロゴ、VDE、DFE、DSE は米国 Oracle Corporation およびその子会社、関連会社の登録商標です。その他の名称は、他社の商標の可能性があります。

All statements regarding Oracle’s future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only. THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

このプログラムは、第三者の Web サイトへリンクし、第三者のコンテンツ、製品、サービスへアクセスすることがあります。オラクル社およびその関連会社は第三者の Web サイトで提供されるコンテンツについては、一切の責任を負いかねます。当該コンテンツの利用は、お客様の責任になります。第三者の製品またはサービスを購入する場合は、第三者と直接の取引となります。オラクル社およびその関連会社は、第三者の製品およびサービスの品質、契約の履行（製品またはサービスの提供、保証義務を含む）に関しては責任を負いかねます。また、第三者との取引により損失や損害が発生いたしましても、オラクル社およびその関連会社は一切の責任を負いかねます。

Included 3rd Party Software Credits. The Oracle Virtual Directoryincludes the following 3rd party software:

Code From The Apache Software Foundation: Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

RSA Data Security: “This product includes code licensed from RSA Data Security”.

JCUP/Jflex: The copyright notice “Copyright 1996-1999 by Scott Hudson, Frank Flannery, C. Scott Ananian” appears in all copies, and both this notice and the warranty disclaimer below appear in supporting documentation

目次

1 概要

2 仮想ディレクトリの計画

6 構成および設定

7 ルーティング

8 マッピング・システム

9 Java プラグインの開発

10 プラグインおよびマッピングの構成

11 アクセス制御

A Oracle Virtual Directory 2.0 および 10.1.4 のプロパティ表

B バンドル・プラグイン

C Web ゲートウェイ

1
概要

この章では、Oracle Virtual Directory と Virtual Directory Manager の概要を説明します。

暫定リリースについて

これは、オラクル社による OctetString 社の買収に伴う暫定リリースです。Oracle Virtual Directory（旧称 OctectString Virtual Directory Engine）、Oracle Virtual Directory Manager（旧称 Directory Manager Environment）および関連ドキュメントの一部では、依然として旧社名（OctetString 社）および旧製品名が記述されていることがあります。これらの記述は今後のリリースで変更される予定です。

概要

Oracle Virtual Directory は、1 つ以上のエンタープライズ・データ・ソースを 1 つのディレクトリ・ビューに仮想的に抽出する LDAPv3 対応サービスです。Oracle Virtual Directory を使用すると、LDAP 対応アプリケーションを様々なディレクトリ環境に統合することができます。このとき、インフラストラクチャまたはアプリケーションを変更する必要性は最小限に抑えられます。まったく変更の必要がない場合もあります。

Oracle Virtual Directory には 3 種類のアダプタが用意されており、管理者が 1 つ以上のアプリケーションのニーズに合せて仮想ディレクトリを作成するために使用できます。これらのアダプタの種類は次のとおりです。

プロキシ・アダプタ：

- LDAP アダプタ
- データベース・アダプタ
- Windows NTLM アダプタ

記憶域アダプタ：

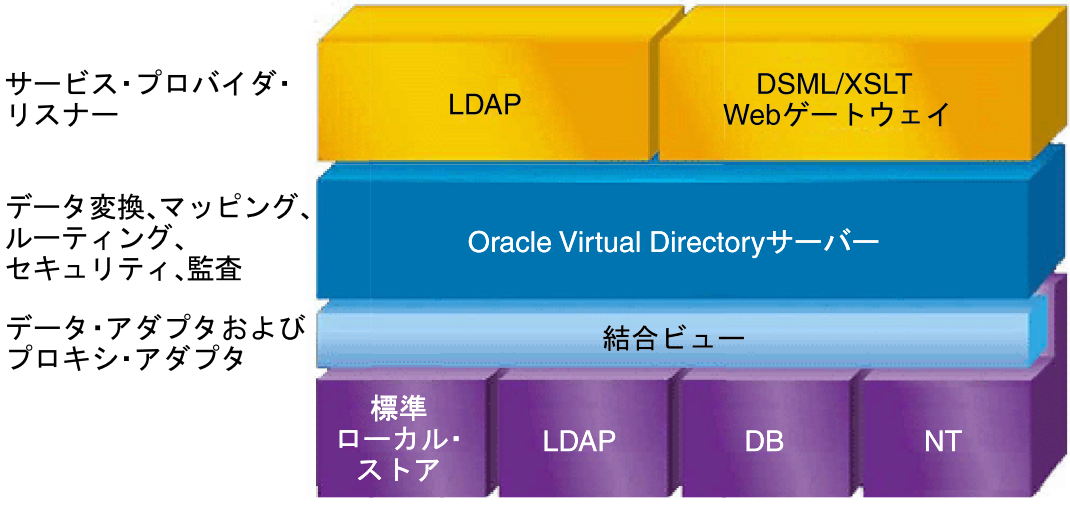
- ローカル・ストア・アダプタ

機能アダプタ：

- 結合ビュー・アダプタ

これらのアダプタの他に、Oracle Virtual Directory ではカスタム・アダプタの作成機能もサポートされています。これには、定義済の API によってほとんどすべてのデータ・ソースに接続できるプラグインを使用します。たとえば、カスタム・アダプタを使用すると、Web サービスで入手できる情報を抽出できます。

図 1-1 Oracle Virtual Directory のアーキテクチャ



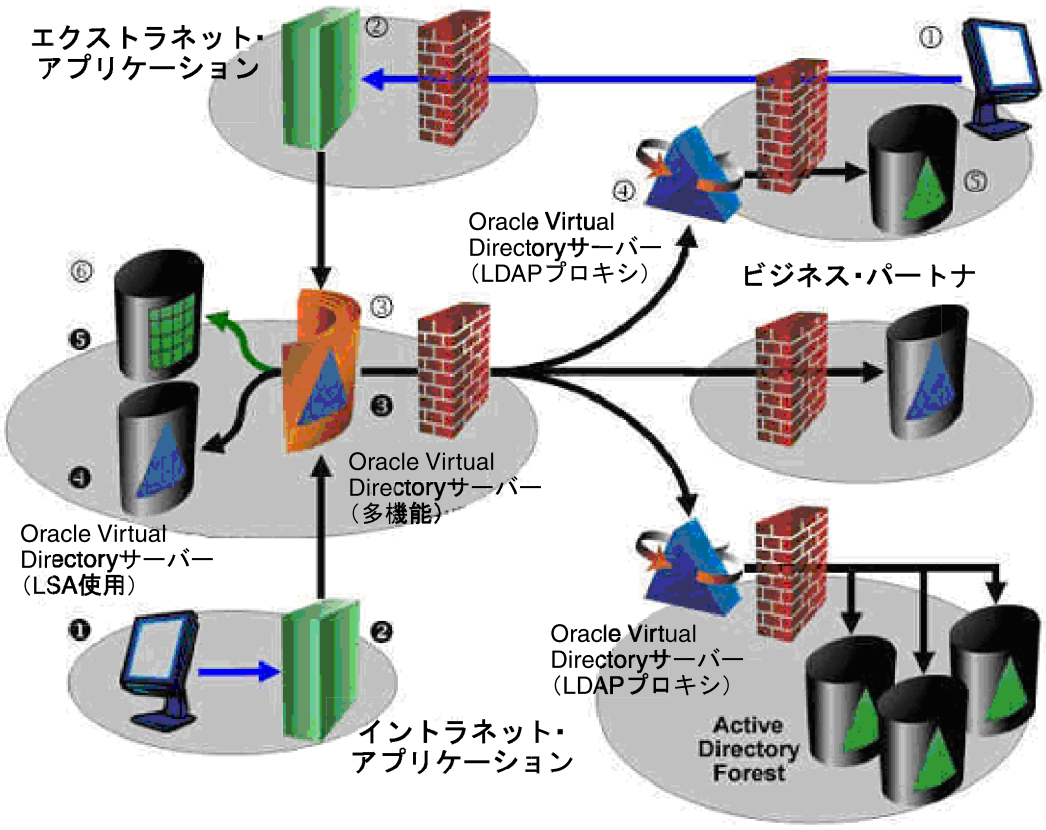
Oracle Virtual Directory をエンタープライズ・デプロイで使用する場合、アプリケーション・ディレクトリの要件に合うように既存のディレクトリ・データを変換することができます。

Oracle Virtual Directory が特に役立つのは、複数のディレクトリを既存環境に仮想的に統合して、1 つの仮想ディレクトリを形成するときです。

Oracle Virtual Directory は、アダプタ・アーキテクチャを介して複数のディレクトリ・ソースと通信する機能を使用して複数のディレクトリを統合することができます。また、スキーマとネームスペースの完全な変換サービスも提供されます。これにより、複数のプロキシ・ソースのデータが一貫した共通の形式でアプリケーションに提供されます。

イントラネットでの ID の例

図 1-2 企業ディレクトリ・ネットワーク環境での Oracle Virtual Directory



この図では、Oracle Virtual Directory がいくつかの方法で使用されています。左下では、内部エンド・ユーザー（❶）がイントラネット・ベースの Web アプリケーション（❷）にアクセスしています。

> **注意：** アプリケーションには、独自のインフラストラクチャの一部としてポリシー・サーバーが含まれる場合と含まれない場合があります。

アクセスの際に、アプリケーション（またはポリシー・サービス）がユーザーの ID とパスワードを要求します。アプリケーションまたはポリシー・サービスは、LDAP のバインド・リクエストを使用して資格証明を検証するために、LDAPv3 を使用して Oracle Virtual Directory（❸）にアクセスします。次に Oracle Virtual Directory は、このリクエストをローカル・ディレクトリ・サーバー・ストア（❹）にルーティングし、資格証明を検証します。検証が終わると、Oracle Virtual Directory は確認した結果をアプリケーション（❷）に返します。

その後のリクエストでは、アプリケーションは、アプリケーションのプロファイルと権限を取得できるように、Oracle Virtual Directory にユーザーのディレクトリ・エントリを要求します。このとき、Oracle Virtual Directory は透過的な結合を実行して、ローカル・ディレクトリ・サーバー（❹）と RDBMS（❺）の情報の属性をまとめます。収集が終わると、Oracle Virtual Directory は結果を 1 つの仮想エントリにマージして、そのエントリをイントラネット・アプリケーションに返します。

エクストラネットでの ID の例

この例では、外部の組織またはビジネス・パートナのユーザー（①）がエクストラネット・ベースの Web アプリケーション（②）にアクセスしています。このアプリケーションが、LDAP バインドを使用してユーザーの資格証明を確認するために、LDAPv3 を使用して Oracle Virtual Directory（③）にアクセスします。

この段階で、Oracle Virtual Directory は、資格証明が外部ディレクトリにマップされることを認識します。Oracle Virtual Directory は、ビジネス・パートナのところにある外部 Oracle Virtual Directory ディレクトリ（④）に SSL 暗号化リンクを使用して接続し、独自の資格証明を使用して事業単位間の問合せを検証します。ビジネス・パートナの Oracle Virtual Directory は Oracle Virtual Directory（③）を検証すると、リクエストを認識し、内部の LDAPv3 ディレクトリ（⑤）に渡します。Oracle Virtual Directory は、適切なビジネス間アクセス制御を適用し、フィルタ結果をディレクトリから Oracle Virtual Directory に返します。これで、Oracle Virtual Directory がビジネス・パートナ・ユーザーのパスワードを検証でき、成功または失敗をアプリケーション（②）に返すことができます。

最後に、イントラネット・アプリケーションの例と同じく、アプリケーションがユーザーのその他の属性を Oracle Virtual Directory に問い合せます。Oracle Virtual Directory は、ビジネス・パートナ・ディレクトリ（⑤）のクライアント提供情報と企業データベース（⑥）にローカルに格納されている情報を結び付ける結合を実行します。

シナリオの確認

ここでは、複雑なシナリオにそって機能を説明します。Oracle Virtual Directory は、情報のルーターおよびジョイナとして機能し、アプリケーションまたはセキュリティ・インフラストラクチャのニーズに合せて複数のセキュアなソースの情報を仲介します。Oracle Virtual Directory は 1 つのイントラネット内の情報をまとめるだけでなく、ビジネス・パートナの情報を活用することもできます。ビジネス・パートナがエクストラネット・アプリケーションを使用するときに、ホスト・ビジネスのディレクトリでのプロビジョニングや管理が必要なくなるため、この機能は特に重要です。ビジネス・パートナ・ユーザーは、自らのローカル・ディレクトリでリアル・タイムに認証されます。

また、Oracle Virtual Directory は LDAP プロキシ・サーバーとしても重要な役割を果します。オプションとしては、Oracle Virtual Directory をビジネス・パートナのディレクトリのファイアウォールとして使用することもできます。Oracle Virtual Directory は、内部ディレクトリ情報への外部からのアクセスを適切に承認および認可します。図の右下では、Oracle Virtual Directory 独自のルーティング機能により、複数の内部ディレクトリまたは Windows Active Directory Forest にルーティングして、クライアントから情報を保護する様子が示されています。ファイアウォールとしては、Oracle Virtual Directory は情報へのアクセスを管理および制限して、承認された外部ユーザーがアクセスできるようにします。仮想ディレクトリ・コンポーネントとしては、Oracle Virtual Directory は、データを単純化および再構成して公開し、ビジネス・パートナが使用できるようにします。

Oracle ディレクトリ・サービスの概要

オラクル社は、次のように幅広いディレクトリ・サービス・ソリューションを提供する唯一のベンダーです。

- Oracle Internet Directory によるスケーラブルなローカルストア・ベースのディレクトリ・サーバー
- Directory Integration Platform によるメタディレクトリ
- Oracle Virtual Directory によるディレクトリの仮想化

Oracle Internet Directory は、LDAP サーバーにデータを格納する必要があるが、既存のディレクトリ・サーバーがない場合に使用します。Directory Integration Platform は、データベースまたは他のディレクトリ情報を Oracle Internet Directory と同期化する必要がある場合に使用します。また、Directory Integration Platform を使用して、Oracle Internet Directory と特定の Oracle アプリケーション（Oracle eBusiness Suite など）の間でデータを同期化することもできます。ディレクトリの仮想化は、リアルタイムに直接データ・アクセスを介して、異機種ソースのデータを 1 つのディレクトリ・サービスに集約するために使用します。

Oracle ディレクトリ・サービス製品は、それぞれ個別に使用することも組み合せて使用することも可能です。たとえば、Oracle Virtual Directory を Oracle Internet Directory と一緒に使用すると、Oracle Internet Directory データに対する DSML インタフェースが提供されます。Oracle Internet Directory を使用するとスケーラブルな記憶域が提供されます。これにより、既存のディレクトリを利用できない場合に、Oracle Virtual Directory で情報を管理することができます。また、Directory Integration Platform と Oracle Internet Directory で Oracle Virtual Directory を使用すると、既存の仮想データストアにフォルト・トレランス・サポートを追加することができます。たとえば、なんらかの理由でプライマリ企業ディレクトリが使用不可になった場合に、Oracle Virtual Directory は Oracle Internet Directory ストアを使用できます。

機能

Oracle Virtual Directory の主な機能は次のとおりです。

- LDAPv2/v3
- DSMLv2/SOAP
- HTTP/XSLT ゲートウェイ
- 低コストの構成および管理
- TLSv1 および SSLv3 のサポートによる暗号化および厳密認証
- メモリーおよびハードウェアの最小限に抑えられた要件
- Java がサポートされるすべてのプラットフォームで使用可能
- LDAP 操作レベルで構成可能なフェイルオーバーおよびインテリジェント・ロード・バランシング
- IETF のアクセス制御実装インターネット草案に基づく粒度の細かいアクセス制御
- JNDI 準拠ディレクトリおよび JDBC 準拠データベースへのアクセスのサポート
- 複数ディレクトリの情報とスキーマの動的マッピング
- LDAP 問合せのインテリジェント・ルーティング
- DoS に対する保護
- オーバーラップ・ネームスペースの処理
- ローカル・ストア・アダプタ
- LDAP プロキシ・アダプタ
- RDBMS アダプタ
- NTLM アダプタ

- 結合アダプタ
  - メタディレクトリのような拡張可能な動的結合機能
  - ローカル・スキーマのサポート
  - 結合ディレクトリ（Active Directory または NT など）のクライアントの認証

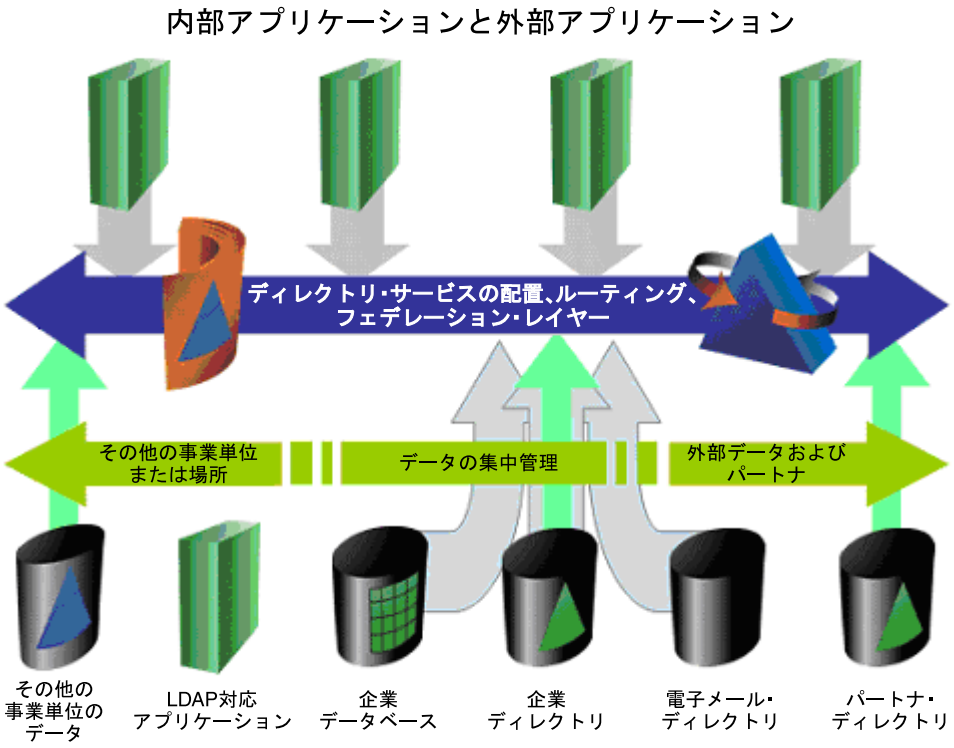
企業ディレクトリでは不十分な理由

現在のディレクトリ・サーバーは、単に特殊なデータベースとして設計されたものです。このようなディレクトリ・サーバーだけでは、関連するすべてのアプリケーションを 1 つの企業ディレクトリに接続するために必要なツールを企業に提供できません。わずかの例外を除き、1 つの企業ディレクトリしかない企業はありません。

アナリストによれば、ほとんどの企業は社内全体で使用するために複数（5 つ以上）のディレクトリを保持しています。また、複数のビジネス・パートナが使用するアプリケーションにデータを提供する目的がある場合、ディレクトリの数は、少なくともアプリケーションを使用するビジネス・パートナの分だけ増加します。オラクル社では、ほとんどの企業が社内および社外に複数層のディレクトリ・サービスを必要とすると考えています。Oracle Virtual Directory は、データを重複させず、大規模なレプリケート・インフラストラクチャのコストを発生することもないため、この要件を実現する最適な方法の 1 つです。

典型的なディレクトリおよびデータベース・テクノロジでは、独立した事業単位、部署およびパートナから企業が構成されているときに発生する問題を解決することができません。現在のディレクトリ・サーバー・テクノロジでは、企業は 1 つの管理データ・インフラストラクチャを構築することを強制されます。これには、含めるデータ、管理の責任者、また最も重要な資金提供者について、利害がからんだ長引く話合いが必要になります。ディレクトリのコストを誰が負担するか、またディレクトリを誰が管理するかという問題が、比較的単純なデータベース・テクノロジのデプロイを成功させるための重大な要因になります。

図 1-3 ディレクトリ・サービスの仮想化と配置



この図では、いくつものディレクトリ・ソース（下部）が様々な形式で様々な場所に設置されていますが、最も重要なのは所有者も多岐にわたるということです。このような従来の企業

ディレクトリには、リレーショナル・データベースや電子メール・システムといったその他のディレクトリが追加されます。

データの配置に伴う問題は、Lotus Domino や Microsoft Exchange のような LDAP 対応アプリケーションが加わるとさらに複雑になります。このようなアプリケーションに含まれるディレクトリ情報は、スキーマの要件が異なるため、既存の企業ディレクトリにそのまま統合することはできません。

開発者は、常に特定目的のデータベースの作成に優れた実績を上げています。これは、ビジネスを推進するアプリケーションのスポンサである個々の経営者によって意思決定が後押しされるためです。B2B Web サービスやビジネス間アプリケーションという新しいトレンドは、ディレクトリ・サービスおよびセキュリティ・インフラストラクチャ戦略を作成するときに外部パートナのデータ・ソースを考慮する必要があることを意味します。

必要となるのはディレクトリ・サービス統合レイヤーです。そこでは、分散セキュリティ（可用性、検証性）、ルーティング（様々なデータの取得方法）、統合（様々な形式の処理方法）、およびデータ・レベルのフェデレーション（信頼できるディレクトリのマージ）といった現実的な問題に対処する必要があります。

Oracle Virtual Directory はこのような課題に対するオラクル社の答えです。

Oracle Virtual Directory は次のようにこの課題に取り組みます。

- 利害関係や企業の境界を超えたディレクトリ・サービス・アクセスを可能にします（データ・フェデーレーション）。
- RDBMS など複数の形式のディレクトリ情報にアクセスできます（変換）。
- 複数の組織がデータを共有できるようにする一方で、完全な制御を保持し、最新の情報を正しく使用できるように保証します（データ所有権）。
- 新しいセキュリティ・ドメイン・コンテキストを提供してセキュリティを強化します（セキュリティ・ドメイン）。
- プロキシ・データのセキュリティと整合性を保証する機能を提供します（セキュアな公開）。
- アプリケーションを様々なディレクトリの設計や実装と統合できます（アプリケーションとディレクトリの統合）。
- 柔軟性の高いデプロイ・オプションを提供します。これにより、COTS（開発済市販製品）開発者やビジネス・アプリケーション開発者が、Oracle Virtual Directory をアプリケーションに組み込むことができます。また、企業の IT 部門が、共有ディレクトリ・サービス分散ネットワークとして Oracle Virtual Directory をデプロイできるようになります（柔軟性の高いデプロイ）。
- デプロイ、管理および実行が容易なソリューションを提供します（低コスト、高価値）。

データ・フェデレーション

Oracle Virtual Directory はディレクトリ・ゲートウェイとして機能します。これは、クライアント・リクエストを処理し、形式（LDAP、RDBMS など）にかかわらず 1 つ以上の既存ディレクトリに動的に再ルーティングします。このとき、Oracle Virtual Directory は仮想ディレクトリ階層（ツリー）をクライアントに示してから、指定された LDAP または RDBMS サーバーにツリーの階層ブランチを割り当てます。Oracle Virtual Directory が、ディレクトリ間のセキュリティ、プロトコルおよびデータ変換の問題を処理するため、LDAP クライアントは、すべての情報が信頼できる 1 つの LDAP ディレクトリ（Oracle Virtual Directory）から送られたと認識します。

データの所有権

データの所有権は、目立ちませんが仮想化の重要な利点の 1 つです。多くの場合、ディレクトリは組織が特定の目的を想定して設定します。ある組織が所有するデータに他の組織がアクセスすることを望む場合、データの所有者およびデータの管理者は誰かという問題が発生します。別の組織が情報を使用および共有する場合には利害関係が発生します。既存データの再利用の価値は広く認識されていますが、データの再利用には整備や管理に関する多くの問題が伴います。データを所有する多くの組織は、データのコピーが他の組織や外部に出ることを非常に心配します。誰が責任を取るのでしょうか。誰が正確さを保証するのでしょうか。誰がセキュリティと機密保持を保証するのでしょうか。情報がコピーされた場合、所有する組織は、他の組織がその情報を使用および管理する方法についてどのように確認すればよいのでしょうか。

プロキシ・テクノロジを介した仮想化では、データは所属する場所（所有者）に留まるため、このような利害にからむ問題の多くは解消されます。所有者はいつでもこのデータに対するアクセスを制限または禁止できます。さらに、所有者は自由にこの情報を改訂することができるため、パートナが常に適切な最新情報を利用することを確認できます。最も重要なのは、所有者が情報を保持することにより、情報の利用を所有者が継続して監視および制御できるということです。Oracle Virtual Directory では、情報をコピーしないことによりこのようなソリューションをサポートしています。Oracle Virtual Directory による情報のアクセスはリアルタイムで行われます。このため、情報が最新かつ正確で認可されていることがコンシューマとプロバイダに保証されます。

複数のデータ・アダプタ

Oracle Virtual Directory では、アダプタと呼ばれるディレクトリ・データ接続コンポーネントの数に制限はありません。各アダプタは、特定の親識別名（DN）で表される特定のネームスペースの管理を担当します。複数のアダプタの組合せやオーバーラップによって、カスタマイズ・ディレクトリ・ツリーを表すことができます。

Oracle Virtual Directory では次の種類のアダプタをサポートしています。

- **LDAP アダプタ :** LDAPv2/LDAPv3 ディレクトリ・サーバー（Microsoft Active Directory、Novell® eDirectory™、Sun™ONE Directory、IBM/Tivoli SecureWay® Directory、その他の Oracle Virtual Directory など）へのプロキシ・アクセスを提供します。LDAP プロキシは、ネームスペース変換、拡張接続プーリングおよび操作レベルのロード・バランシングを提供します。
- **データベース・アダプタ :** リレーショナル・データベース・データの LDAP 仮想化を提供します。ほぼすべてのデータ構造を LDAP オブジェクトの階層にマッピングできます。DB アダプタは、自動スキーマ・マッピングと属性値変換も提供します。
- **ローカル・ストア・アダプタ :** Oracle Virtual Directory がスタンドアロン・ディレクトリ・サーバーとして稼働するために使用できるローカル・ディレクトリ・ストアを提供します。この標準アダプタはシングルマスター・レプリケーションをサポートし、SLURPD レプリケーションをサポートする他のディレクトリ・サーバー（IBM/Tivoli SecureWay® または Netscape Directory など）と互換性があります。
- **Windows NTLM アダプタ :** Microsoft Windows NT ドメインの LDAP 仮想化を提供します（**注意 :** Win32 プラットフォームのみで使用可能）。
- **JoinView アダプタ :** 他の Oracle Virtual Directory アダプタにあるエントリ間のリアルタイム結合機能を提供します。JoinView アダプタは、顧客固有のジョイナを開発できる拡張可能 API を提供します。JoinView アダプタには、Simple、OneToMany および Shadow という 3 つのジョイナが含まれています。これらのジョイナにより、Oracle Virtual Directory ジョイナの幅広い機能と、実行できる様々な結合処理が示されます。これらのジョイナの詳細はこのガイドの後の章で説明します。

柔軟性の高いセキュリティ・ドメイン

新しいビジネス・アプリケーションを複数のビジネス組織にわたってデプロイする場合、複数のディレクトリ・セキュリティ・インフラストラクチャが存在するために ID とセキュリティが複雑になることがあります。Microsoft Active Directory の管理者が認識するように、複数の Windows インフラストラクチャ（フォレスト）があることは管理とパフォーマンスにとっては長所ですが、フォレスト間の信頼関係を自動的に結ぶ機能やフォレスト間のグローバル・カタログがないことは短所です。
http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/activedirectory/plan/addeladm.mspx にある Microsoft TechNet ペーパー『Design Considerations for Delegation of Administration in Active Directory』を参照してください。

Oracle Virtual Directory は、きめ細かなアクセス制御を備えた、新しい推移的なセキュリティ・コンテキストを作成することができます。アクセス制御に関するすべての IETF 規格をサポートし、実装のための IETF 草案モデルもサポートしています。また、Oracle Virtual Directory は、プロキシ対象のソース・ディレクトリのセキュリティ制限と適切に統合するように設計されています。これによって複数層または複数ドメインのセキュリティ概念がもたらされ、管理者が最大限のセキュリティ制御を得ます。

Oracle Virtual Directory は幅広い認証モデルをサポートしています。Oracle Virtual Directory は、SSL/TLS（StartTLS を含む）および証明書ベースの認証に加えて、プロキシ・サーバー（自らの認証）によるサーバー対サーバーの認証を使用できます。または、ユーザー・コンテキストをソース・ディレクトリに渡すこともできます。Oracle Virtual Directory とソース・ディレクトリにユーザー・コンテキストを提供することにより、両方のディレクトリはエンド・ユーザーにコンテキスト・セキュリティ制御を提供できます。

セキュアなデータ公開

Oracle Virtual Directory では次のような標準の機能を提供しています。

- **SSL/TLS:** Oracle Virtual Directory は、LDAP クライアントとのセキュアな通信セッションを実現する SSL/TLS 機能を提供します。これにより Oracle Virtual Directory は信頼できる転送メカニズムになり、セキュリティが向上します。
- **トランザクション・クレンジング :** Oracle Virtual Directory はプロトコル変換エンジンに基づいています。つまり、すべての問合せを分解し、再コンパイルして妥当性を調べてから、信頼できるプロキシ・ディレクトリ・ソースに送ります。これにより、不正な問合せや未認証の問合せから LDAP サーバーを保護します。不正なリクエストを除いた後で、Oracle Virtual Directory は次のような項目に制限を設定する機能により、悪意のある攻撃による重い負荷から限りあるリソースを保護することができます。
  - 1 接続当たりの最大操作数
  - 最大同時接続数
  - 特定のサブジェクトに対する特定期間の最大合計接続数
  - 特定のアドレスに対する特定期間の最大合計接続数

Oracle Virtual Directory は独自のアクセス制御を実装し、内部プロキシ・ディレクトリ・データに対してアクセスをフィルタ処理します。

高可用性のサポート

- **フォルト・トレランスとフェイルオーバー :** Oracle Virtual Directory はフォルト・トレランスを次の 2 つの方法で提供します。
  - フォルト・トレラント構成
  - フォルト・トレラント・プロキシ・ソースへのフローの管理

  構成ファイルをコピーするだけ、または構成ファイルも共有することにより、複数の Oracle Virtual Directory を迅速にデプロイできます。Oracle Virtual Directory をラウンドロビン DNS、リダイレクタまたはクラスタ・テクノロジと組み合せると、完璧なフォルト・トレラント・ソリューションが提供されます。

  プロキシ・ディレクトリ・ソースごとに、特定のソースに対して複数のホスト（レプリカ）にアクセスするように Oracle Virtual Directory を構成できます。Oracle Virtual Directory はホスト間でインテリジェント・フェイルオーバーを実行し、負荷を分散します。柔軟性の高い構成オプションにより、管理者は、特定のレプリカ・ノードに送る負荷のパーセンテージを制御できます。また、特定のホストを読取り専用レプリカにするか読取り / 書込みサーバー（マスター）にするかを指定できます。これにより、読取り専用レプリカへの書込みの試行による不必要な参照が避けられます。

- **ロード・バランシング :** Oracle Virtual Directory には強力なロード・バランシング機能が設計されています。これにより、プロキシ LDAP ディレクトリ・ソース間で負荷を分散して障害を管理することができます。

  Oracle Virtual Directory の仮想ディレクトリ・ツリー機能により、大容量のディレクトリ情報のセットを複数の個別ディレクトリ・サーバーに細分化することができます。Oracle Virtual Directory は、分かれたディレクトリ・ツリー・ブランチをまとめるだけで、分割したデータ・セットを 1 つの仮想ツリーに再結合することができます。アプリケーションまたはデータがこの機能をサポートしない場合、または分割したディレクトリのディレクトリ・ツリーをオーバーラップする必要がある場合、Oracle Virtual Directory はルーティングをサポートします。

  ルーティングでは、最適な検索ターゲットを決定するために、検索フィルタを検索ベースに追加することができます。このモードでは、Oracle Virtual Directory は問合せを適切な仮想ディレクトリ・ソースに自動的にルーティングし、数百万に及ぶディレクトリ・エントリの処理が可能になります。

アプリケーションとディレクトリの統合

ディレクトリが役立つのは、対応するアプリケーションが必要なデータに共通の形式またはスキーマでアクセスできる場合のみです。しかし、典型的なエンタープライズ環境には、様々なスキーマ、ネームスペースおよびデータ設計を備えた無数のディレクトリ・リポジトリが含まれています。

既存のディレクトリ情報にセキュアなブリッジを提供するだけでなく、Oracle Virtual Directory は、その場でデータを変換するメタディレクトリと同様の機能を提供します。この機能により、管理者は、異なる組織やディレクトリ・インフラストラクチャにおけるデータの違いを簡単に標準化することができます。

この結果として得られる仮想ディレクトリ・ビューには、アプリケーションが実行するために必要なすべてのディレクトリ情報が含まれます。大幅な変更や統合テクノロジをアプリケーションに組み込む必要はありません。

柔軟性の高いデプロイ

Oracle Virtual Directory の管理コンソールである Oracle Virtual Directory Manager は、オープン・ソースの Eclipse プラットフォームに基づく、豊富な機能を備えた拡張可能な管理環境です。この管理コンソールにより、1 つの Oracle Virtual Directory を 1 つの環境で使用する場合でも、複数のデータ・センターの数十台のサーバーからなる環境において複数のデプロイ・ステージで使用する場合でも、デプロイと管理が簡略化されます。

WSDL 仕様が公開されている Web サービス API を使用して管理を行うこともできます。管理者は、GUI を使用せずに、Oracle Virtual Directory をスクリプトまたはその他のプログラミングによって操作することができます。

カスタム・アプリケーション・プログラミング・インタフェース

Oracle Virtual Directory では 3 つの機能に拡張性があります。顧客およびコンサルタントは、特定のビジネスまたは技術統合のニーズに合うように Oracle Virtual Directory の機能を拡張することができます。

- **Oracle Virtual Directory プラグイン :** Oracle Virtual Directory は、Java サーブレット・フィルタをモデルとする柔軟性の高いプラグイン・フレームワークを提供します。プラグインを使用して、カスタム・ロジックをトランザクションの一部として提供したり、カスタム・データ・ソースに接続したりします。プラグインは、グローバルに挿入することも特定のアダプタのみに挿入することも可能です。プラグインの順序は変更できます。特定のタイプのトランザクションに対しては分離することもできます。Oracle Virtual Directory の管理ツールでは、新しいプラグインを作成するためのウィザードやすぐに使用できるサンプルが提供されます。
- **カスタム・ジョイナ :** Oracle Virtual Directory JoinView アダプタは、ジョイナと呼ばれる拡張可能なモデルに基づいています。カスタム・ジョイナを作成して、様々なジョイナの動作を提供できます。ジョイナは、マッピング、結合、プレ / ポスト / ハンドラのイベント処理などの機能を提供します。単純なエントリ・レベルの結合を提供するジョイナを作成することや、ジョイナを拡張して、複雑な結合ロジックまたはトランザクション処理とロールバック機能を提供することが可能です。
- **Web ゲートウェイ :** Oracle Virtual Directory にはカスタマイズ可能な DSML/XSLT ベースのゲートウェイが含まれます。このゲートウェイは、Apache Web サーバー・モデルに基づく基本的な Web サーバー・サポートを提供します。静的な HTML および XSLT レンダリング・コンテンツがサポートされます。ゲートウェイにはディレクトリ対応インタフェースが含まれ、問合せや変更操作が可能です。Web サーバー・セキュリティによって、このインタフェースに基づくカスタム委任管理アプリケーションを開発できるようになります。

低コスト、高価値のソリューション

従来のディレクトリ統合ソリューションは、複雑な LDAP プロビジョニングおよびレプリケーション・スキームが必要であり、同期化を実行することも必要です。このような新しいディレクトリも、維持および管理が必要な追加のディレクトリ・ソースになります。

軽量のリアルタイム・サービスである Oracle Virtual Directory は、同期化や複製は行わずに既存のディレクトリ・インフラストラクチャを再利用することで効率を向上します。Oracle Virtual Directory は、既存の企業ディレクトリの能力を拡張し、その価値を十分に活用します。

重要なビジネス機能と利点

- **管理コストの削減、セキュリティの向上、セキュリティ侵害リスクの減少 :** 複製の更新、同期化およびレプリケーションに関する問題がなくなります。データは常に最新で一貫性を保ちます。
- **エンタープライズ・アプリケーションの迅速な拡張 :** 企業ディレクトリ・アプリケーションおよびレガシー・データをデフォルトでサポートします。社内ユーザー、取引業者および顧客にとって、セキュアで正確なデータ・アクセスが可能になります。
- **情報へのユビキタス・アクセス :** ソフトウェアは LDAPv3（LDAP v2 を含む）に完全に準拠しています。ほとんどのアプリケーションに対応し、多くのディレクトリ製品、ツールおよびアプリケーションと互換性があります。
- **低コストの実装 :** カスタム・ソリューションまたは同期ベースのソリューションよりも導入または実装に費用がかかりません。特定のアプリケーション統合の問題を解決するための戦術として Oracle Virtual Directory をデプロイすることができます。または、ディレクトリ・インフラストラクチャ・アーキテクチャ全体に戦略的にデプロイすることもできます。

動作が保証されているオペレーティング・システム、ディレクトリおよびデータベース

Oracle Virtual Directory 10*g*（10.1.4.0.1）は次のコンポーネントに対して動作が保証されています。

カスタム・デプロイの一般的なガイドラインとしては、Oracle Virtual Directory は Java ベースであるため、Java が接続できるコンポーネントと相互運用可能です。

オペレーティング・システム

- Solaris 8 および 9
- Red Hat Linux 8.0 および 9.0 および ES 3.0
- SUSE Linux 9 および 10
- Windows NT 4.0 with SP6、Windows 2000 with SP3、Windows XP Professional および Windows 2003 Server
- HP-UX 11
- AIX 5.2

ディレクトリ

- Oracle Internet Directory
- Sun JS Directory Server
- CA eTrust Directory
- IBM Tivoli Directory Server
- Novell eDirectory
- Siemens DirX
- Microsoft AD
- Microsoft AD/AM

データベース

- Oracle 9.2.0.7、10.1.0.5、10.2.0.2 RAC およびスタンドアロン DB
- Microsoft SQL Server
- IBM DB2

インストールおよび要件

インストールと実行時の要件、および Oracle Virtual Directory と Oracle Virtual Directory Manager クライアントのインストール方法は、『Oracle Virtual Directory インストレーション・ガイド』を参照してください。

2

仮想ディレクトリの計画

この章では、Oracle Virtual Directory の効果が非常に高いシナリオについて説明し、フォルト・トレランスの計画についても説明します。

概要

この章では次のシナリオについて説明します。

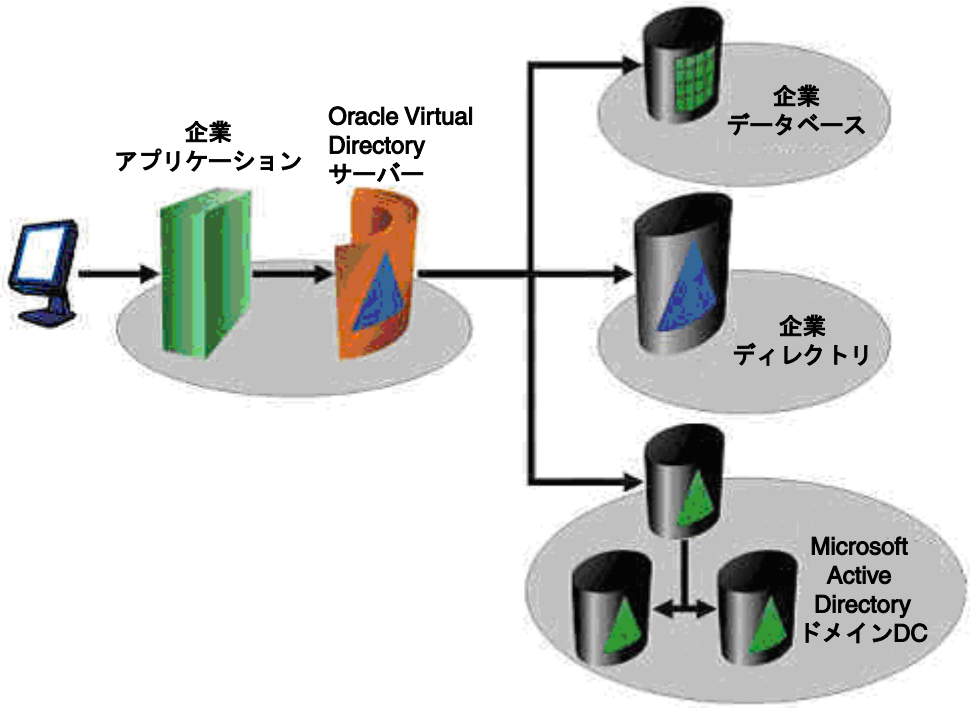
- 既存ディレクトリの統合ディレクトリ・サービスへの仮想化
- 仮想ディレクトリ情報へのアプリケーションのアクセス
- 拡張統合と構成の管理
- フォルト・トレラント・デプロイの計画
- ネームスペースの計画
- ルーティングの概念と構成
- JoinView を使用した動的結合

この章は、組織における仮想ディレクトリの導入による効果を容易に理解することに焦点を当てています。このような効果には、既存のディレクトリ・インフラストラクチャの柔軟性の向上と将来的な使用の保証、スケーラビリティの向上、新規アプリケーションおよび既存アプリケーションでの高い可用性などがあります。

シナリオ 1: ディレクトリ情報の仮想化

Oracle Virtual Directory を使用すると、企業は、1 つのアクセス・ポイントでディレクトリ情報の複数のソースを集約することができます。次の図では、企業の 3 つのソースのディレクトリ情報にアクセスするアプリケーションが示されます。このシナリオでは、各ソースには異なるユーザーのグループが含まれていると仮定します。

図 2-1 ディレクトリの仮想化



この図では、社内全体に対するアプリケーションが、すべての従業員が使用できるようにデプロイされています。Oracle Virtual Directory は、Windows Active Directory、企業ディレクトリ、およびリレーショナル・データベースに含まれる情報から ID 情報を集める仮想ディレクトリとしてデプロイされています。通常、企業の構造に応じて複数のソースが存在することがあります。たとえば、Active Directory には社内の従業員ユーザーのみが含まれ、企業ディレクト

リに社内の別の部署またはビジネス・パートナのユーザーが含まれます。さらに、社外契約者などその他のユーザーはリレーショナル・データベースに格納されます。

手順 1: 信頼の基礎

アプリケーションの設計者がアプリケーションの計画を開始するとき、アプリケーションに必要なすべての ID 情報がどこにあるのかを考慮する必要があります。ユーザーのソースが 1 つしか存在しないということは実際にはほとんどありません。多くの場合、組織には企業ディレクトリが指定されていますが、このディレクトリには全ユーザーの一部しか含まれていません。使用可能なユーザー資格証明を含むその他の社内ソースが存在する可能性があります。

すべての ID について適切な情報ソースが特定されたら、アプリケーション所有者は、この情報の共有と使用に関して理解を得る必要があります。アプリケーションの所有者と ID ソースの所有者が信頼関係を築く必要があります。情報を利用するアプリケーションのスポンサは、合意済の目的で特定の情報を使用すること同意する必要があります。また、情報を提供する各社は、データの品質と正確さを一定レベルで維持することに同意する必要があります。

手順 2: ソース・ディレクトリ情報とアプリケーション要件のマッピング

企業がディレクトリ情報を動的に共有できる前は、どのタイプの情報が必要か、また誰が情報を必要としているかを判別する必要がありました。たとえば、企業アプリケーションで必要な情報を判別します。さらに、供給側の各ディレクトリで、すべての正しい ID と属性情報を得られるかどうかを確認します。得られる場合は、各ディレクトリ情報サプライヤについて次の情報を収集する必要があります。

- 各サプライヤ・ディレクトリの接続情報（レプリカ・ホスト情報、アクセス資格証明、セキュリティ要件など）
- ディレクトリの構造、エントリの構成、および各サプライヤで使用される属性

ディレクトリ・ソースの計画		
ソース企業名 :	Company B, 111 West Side Ave, Vancouver, BC	
技術連絡先 :		
ユーザー・サポート連絡先 :		
セキュリティ連絡先 :		
ビジネス連絡先 :		
リモート・ベース :	ou=People, dc=CompanyB, dc=com	
サーバー・アドレスおよびポート :	192.168.0.2:389、192.168.0.3:389（マスター）、192.168.2.1:389	
プロトコルおよび接続文字列	LDAP	
デフォルト資格証明 :	cn=Company A App, ou=Applications, dc=CompanyB, dc=com	
SSL/TLS 要件 :	SSL 対応	
属性マップ		
アプリケーション	ディレクトリ・ソース	
属性	属性	データ・マッピング
Cn	Cn	
Uid	Uid	"@companyb.com" を追加
Sn	Surname	
Userpassword	Userpassword	パスワードにハッシュ・タグを付けるかどうか（たとえば {SHA}）

ディレクトリ・ソースの計画
その他

手順 3: ディレクトリのネットワーク設計

Oracle Virtual Directory の最も強力な機能は、個別のディレクトリ・ソースに接続して、それらを統合された 1 つのローカル・ディレクトリのように見せる機能です。

データの場所、形式、クライアント・アプリケーションのプロトコルといった複雑な状況は Oracle Virtual Directory では意識する必要がなくなります。これは、交換機とルーターに基づく TCP/IP インターネット・ネットワーク設計に似ています。交換機とルーターは、ネットワーク上の様々なアドレス間で接続やプロトコルを確立する方法の詳細を処理します。ルーターを使用すると全世界の情報がローカル・ネットワーク上にあるように見えるのと同じく、Oracle Virtual Directory を使用することで、多くのディレクトリが 1 つのローカル・リポジトリのように見えます。

前のシナリオ図の仮想化を検討します。クライアント Web アプリケーションを提供するグループは、Microsoft Active Directory、別の部署の企業ディレクトリ、およびリレーショナル・データベース内の外部ユーザー情報のデータを統合することを望んでいます。

Oracle Virtual Directory のジョブでは、すべてのディレクトリ・ソースの情報がルーティングおよび変換され、アプリケーションの要件に合う形式で提供されます。

Active Directory に対しては、グローバル・カタログ・ディレクトリまたはドメイン内の各ドメイン・コントローラを指すように Oracle Virtual Directory を構成できます。ドメイン・コントローラに直接接続する場合の大きな利点は、より多くの情報にアクセスできることです。どちらの場合も複数のグローバル・カタログまたはドメイン・コントローラを選択して、Oracle Virtual Directory と AD 間の各接続をフォルト・トレラントに設定することをお薦めします。

企業ディレクトリに対しても（Active Directory と同じく）、フォルト・トレランスのために Oracle Virtual Directory を複数のレプリカに接続するように構成してください。アプリケーションがディレクトリに書き込む必要がある場合は、最適なパフォーマンスを得るために指定マスター・ノードを定義する必要があります。

手順 4: フォルト・トレランスとロード・バランシングの設定

フォルト・トレランスの問題は仮想化のシナリオ図には示されていません。フォルト・トレラント設計では、複数の Oracle Virtual Directory インスタンスをデプロイする必要があります。このシナリオでは Oracle Virtual Directory がローカル・データを保持しないため、適切な SAN 構成で構成ファイルを簡単に複製または共有でき、コピーのデプロイと管理を容易に行うことができます。

Oracle Virtual Directory の組込み LDAP アダプタによって、複数のソース・ディレクトリ・レプリカおよびマスターへの接続の管理に優れたサポートが提供されます。Oracle Virtual Directory には、問合せの負荷を複数のディレクトリ・レプリカに分散させる機能があります。追加、変更、削除および名前変更の操作を指定ディレクトリのマスター・サーバーに送ることもできます。

障害の処理

フォルト・トレランスがない 1 つのソース・ディレクトリについて考えてみてください。LDAP クライアント・アプリケーションがすべてのディレクトリを対象とする問合せを発行した場合、LDAP RFC では、ディレクトリのすべての部分が正しく応答するか、すべての結果が無効になることが必要です。プロキシ・ディレクトリのいずれかが使用不可にならないかぎり、通常は問題なく作動します。冗長ディレクトリ・リンクのないソースで障害が発生すると、ユーザー・ベースの一部しか影響を受けない場合でも、グローバル問合せが全体で失敗するようになります。Oracle Virtual Directory を使用すると、個々のプロキシで障害が発生したときの応答や、障害のサービス全体への影響を制御することができます。

多くのシナリオでは、パートナ企業のユーザーがホスト企業のアプリケーションにアクセスできるようにプロキシ・ディレクトリが提供されます。パートナ・ディレクトリがオフラインまたはアクセス不可になると、その企業のユーザーもアプリケーションを使用できないことが多

く、障害はアプリケーションにとってクリティカルではないとみなされます（ただし、パートナにとってはクリティカルです）。この場合、停止中のサーバー接続を無視するように Oracle Virtual Directory を構成することで、その他のパートナが作業を続行することができます。

手順 5: ネームスペースの計画

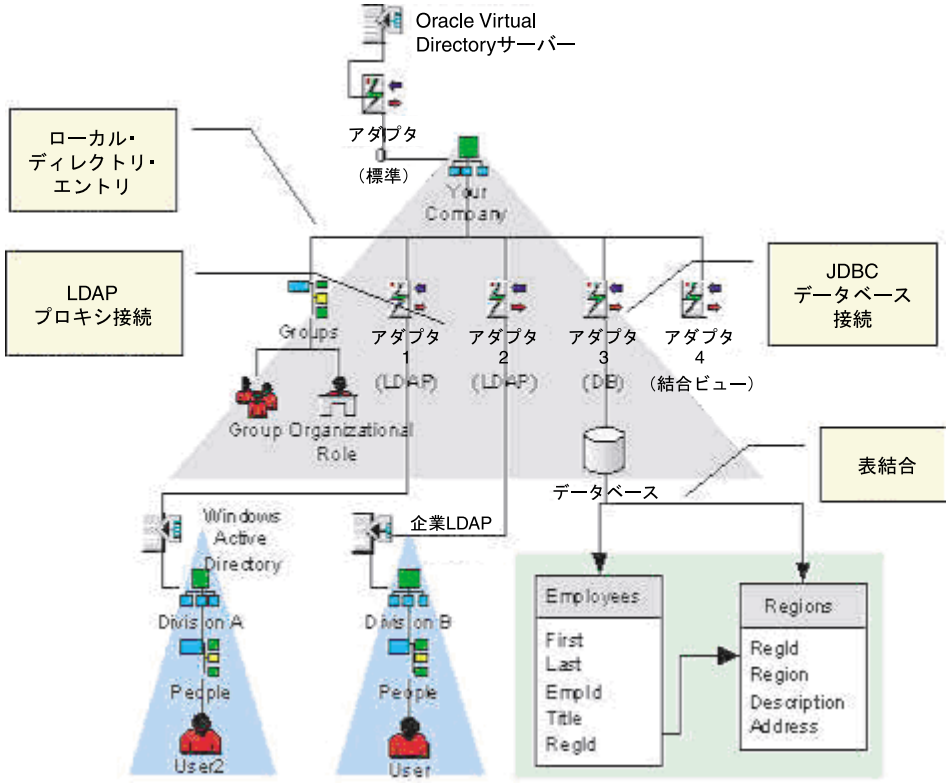
次の手順では、すべてのディレクトリを結合ディレクトリ・ツリーにリンクします。Oracle Virtual Directory を使用すると、すべてのソース・ディレクトリ・ツリーに接続でき、そのツリーを新しい仮想ツリーにマッピングできます。たとえば、ソース・ディレクトリのエントリの識別名（DN）が次の場合について考えます。

```
cn=Jim Smith,ou=People,o=Division B, c=UK
```

これは、次のようにマッピングできます。

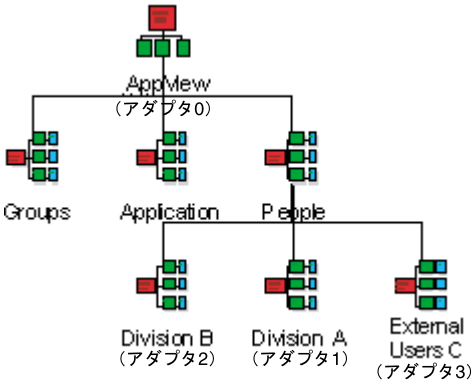
```
cn=Jim Smith,ou=People,ou=Division B, ou=People,o=AppView
```

この場合、Oracle Virtual Directory は、o=Division B, c=UK の下のすべてのエントリを ou=Division B, ou=People, o=AppView にマッピングします。Oracle Virtual Directory は、その場で変換を実行して、Division B のユーザーがアプリケーション固有ディレクトリに含まれるようにします。



この図は、アプリケーション固有のローカル・ディレクトリ・ブランチを示しています。このツリーのルートは o=AppView です。このブランチの下にローカル情報（アプリケーション・アクセス制御またはロール）を格納できます（たとえば、cn=User Group,ou=Groups, o=AppView）。

アプリケーションにアーキテクチャの制限があり、共通の people ブランチの下のユーザーしか検索できない場合があります。アプリケーションの要件に合せて、設計の目的を変更し、新しいディレクトリ設計を ou=People ブランチの下のすべてのディレクトリ・ソースにマッピングするようにできます。次の図ではこれも反映されています。



結論として、Oracle Virtual Directory は 4 つのアダプタを使用するように構成されます。

アダプタ 0: このアダプタはディレクトリ・ツリーのルートを形成します。このアダプタは o=AppView にマッピングします。このアダプタは、ツリーの仮想ルートを保持し、アクセス制御グループなどのローカル・エントリも保持します。

アダプタ 1 ～ 3: 各アダプタは、各ディレクトリ・ソースを新しいアプリケーション・ツリーの ou=People ブランチの下の位置にマッピングします。

手順 6: ルーティングの構成

アプリケーションにはディレクトリ・サービスに関するインテリジェント機能がほとんどないと仮定します。このアプリケーションは本来 1 つのビジネスのために設計されたため、複数のビジネス・ユーザー・グループが同じアプリケーションを使用することを想定していません。このアプリケーションは、多種多様なディレクトリ・ツリー構造には対応せずに、1 つの共通ディレクトリ階層ポイント（または 1 つの共通ベース）のみでディレクトリを検索します。この場合は ou=People,o=AppView です。

たとえば、ユーザーが jim.smith@divisionB.com というログオン ID を入力すると、アプリケーションは次の検索を発行します。

ベース : ou=People,o=AppView
有効範囲 : サブツリー
フィルタ : (uid=jim.smith@divisionb.com)

Oracle Virtual Directory は、この問合せを受け取ると、この問合せで使用できるすべてのアダプタを自動的に選択します。この場合、問合せはツリーのベースを対象としているため、すべてのアダプタが選択されます。これはパフォーマンスの問題を引き起こすため考慮する必要があります。他のすべての企業がディレクトリ構造の下部に存在する場合（たとえば ou=DivisionB, ou=People,o=AppView）、親である ou=People,o=AppView よりもそれらのブランチが下にあるため、デフォルトですべてのディレクトリ・ソースが検索されます。

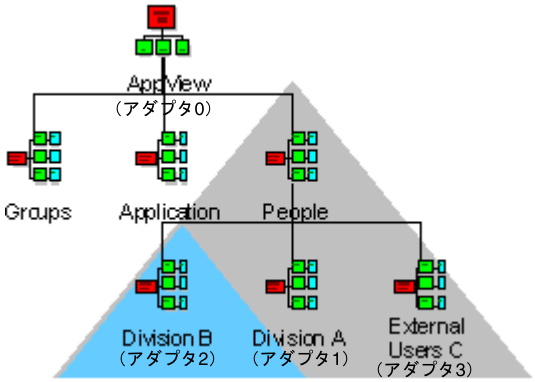
この問題を修正するために、Oracle Virtual Directory はルーティングの組込みと除外のフィルタを提供しています。これらのフィルタを使用して、特定のパートナ・ディレクトリへのトラフィックをフィルタ処理できます。このシナリオでは、管理者は次のルーティング組込みフィルタを設定できます。

Division A のアダプタ：（uid=*@divisiona.com）

Division B のアダプタ：（uid=*@divisionb.com）

Division C のアダプタ：（uid=*@divisionc.com）

LDAP クライアント検索のベースでは通常すべてのディレクトリが選択されますが、このフィルタによって、（uid=jim.smith@divisionb.com）の検索は Division B のディレクトリのみを対象とすることが指定されます。右の図は、通常選択される 3 つのアダプタを灰色（右側）で示し、フィルタ処理の後では Division B のデータのみが検索されることを青色（左側）で示します。



問合せのフィルタ処理の他に、Oracle Virtual Directory では各アダプタに優先度を割り当てることもできます。常に優先度の数値が小さいアダプタから順に問合せの対象になります。優先度の数値が同じアダプタは、構成ファイルでの定義順に検索されます。競合が発生した場合（2 つのエントリが同じ DN を持つ場合）、Oracle Virtual Directory は常に（優先度または構成の）数値が小さい方のアダプタの応答のみを受け入れます。つまり、ルーティングのフィルタで 1 つのアダプタを選択できなかった場合、競合が発生する可能性がありますが、優先度の選択によって解消されます。

シナリオ 2: アプリケーションとディレクトリの統合

背景

開発者がアプリケーションを作成するとき、次のような疑問が発生します。

- ユーザー
- ユーザーの権限
- アプリケーションに存在するロール
- ユーザーの認証方法

多くの場合、これらの疑問に関連する管理データ（資格証明、権限など）は、従来のアプリケーション・データとは大きく異なります。開発者は長年にわたって、このような情報をどこに保存すべきか、つまりアプリケーション・データと一緒にデータベースに保存するか、ディレクトリなどの共通セキュリティ・システム・インフラストラクチャに保存するかを議論してきました。

現在の企業は、特定のユーザーに対して新しいアカウントを管理する必要が生じるたびに、新たな負担が IT 管理者に発生することを認識しています。この問題に対処するために、ほとんどの企業では、多くの従業員の ID とパスワードをディレクトリ・サービスなどの共通インフラストラクチャに統合することを始めています。

LDAP のデータ・モデルは RDBMS よりも制約が多く、共通性を高めることを目的としていますが、データ実装はディレクトリ・サーバーごとに変化に富む傾向があります。これは、ディレクトリ製品の違いによるためもありますが、多くの場合は、実装ごとの顧客のデータ要件が多様であることから違いが生じています。

開発済市販ソフトウェアのアプリケーション開発者も、エンドビジネス開発者も、次のような同じ課題に直面します。サーバーが物理的に設定されている場所はどこか（トポロジ）、すべての情報が 1 つのディレクトリにあるか（仮想化と統合）、どのような属性が使用可能か、データの形式はどのようになっているか（スキーマ）といった課題です。

このような問題を解決するためによく使用される 3 つの方法を次に示します。

1. ディレクトリ・コンポーネントがディレクトリ・サービスに追加されるたびにアプリケーションを修正します（アプリケーションのカスタマイズ）。
2. メタディレクトリの同期化を使用し、多くのリソースのデータを 1 つの共通ディレクトリ・サーバーに抽出して、スーパー・ディレクトリを作成します（ディレクトリのカスタマイズ）。

3. 多様なディレクトリ環境とアプリケーション要件を調整できるディレクトリ統合サービスを使用します（サービスの拡張）。

1. アプリケーションの修正

この最初の方法では、開発者は、柔軟性の高いアプリケーションを作成するために様々なディレクトリ設計や実装環境を理解する必要があることに気付きます。新しい顧客ごとに新たな統合の課題がもたらされます。顧客は、既存のアプリケーションでは、ベンダーと交渉して独自の変更を要求できることはほとんどないことがわかります。一度に 1 つずつアプリケーションの実装を統合することにより、対処不可能なほどのコストが発生することがあります。

2. スーパー・ディレクトリの作成

2 番目の方法では、多くの組織は、すべての戦略システムの共通データを含むスーパー企業ディレクトリの構築に取り組みます。すべてのアプリケーションに対応できる設計を備えた 1 つのシステムをどのように構築するかが課題になります。メタディレクトリの構築では、複数のプロバイダのデータを 1 つの共通リポジトリにコピーする必要があるため、煩雑な利害関係が発生することがあります。整備と管理も大きな問題になります。これは 1 社内でも解決することは難しく、複数のビジネス・パートナ間ではさらに困難です。メタディレクトリ同期化プロジェクトのデプロイを成功させるには課題があるため、開発者はディレクトリ統合のソリューションの一部としてこの方法に頼ることはできません（COTS 製品の場合はより困難です）。

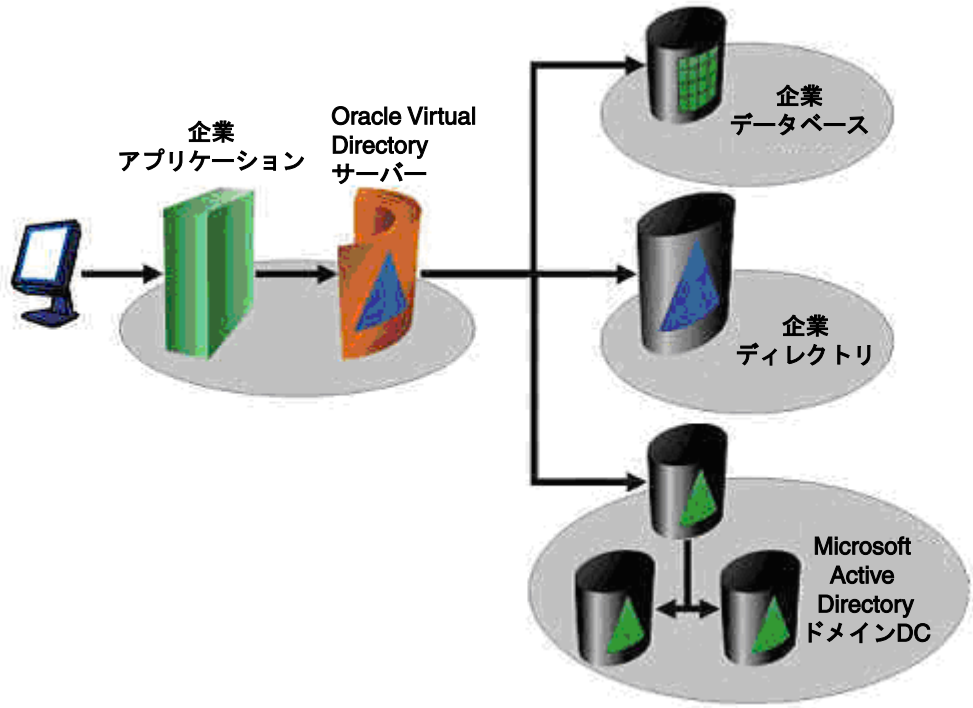
3. アプリケーション・アクセスの仮想化 : Oracle Virtual Directory

Oracle Virtual Directory は 3 つ目の方法に該当します。Oracle Virtual Directory はインテリジェント統合レイヤーとして機能するように設計されているため、これまでに示した問題をアプリケーション開発者が回避することができます。Oracle Virtual Directory は、共通の標準形式でアプリケーションに情報を提供できるように、情報が配置されている場所、情報を取得するタイミング、情報を変換する方法を認識することで役立ちます。アプリケーションに対しては、顧客の企業ディレクトリとビジネス・パートナのディレクトリが同じデータ・モデルの 1 つのディレクトリとして見えるようになります。

Oracle Virtual Directory によって、多種多様な顧客のディレクトリ環境に統合するためにアプリケーションを改良し続ける必要がなくなり、コストが削減されます。Oracle Virtual Directory は、透過的な統合コンポーネントとしてアプリケーションにデプロイできます。または、顧客が既存のアプリケーションを適合させる方法として Oracle Virtual Directory をデプロイすることもできます。

シナリオの概要

このシナリオでも仮想化シナリオの図を使用しますが、ここでは、かなり異なる要件を仮定します。このシナリオでは、右側の 3 つのディレクトリ・ソースすべてに同じユーザーのグループが含まれると仮定します。



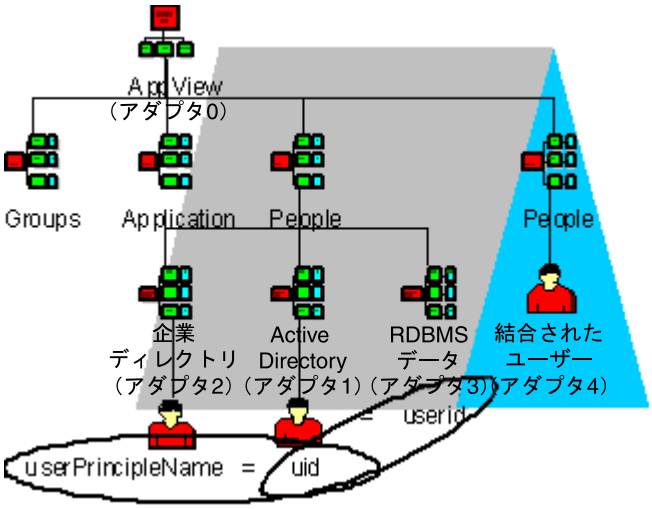
このシナリオでは、メインの企業ディレクトリに、すべてのユーザーの企業ディレクトリ情報の主なソースが含まれます。企業ディレクトリの各ユーザーについては、ユーザー認証のために Microsoft Active Directory の対応するアカウントと一致させる必要があります。また、企業データベースで特定のユーザーに関連するアプリケーション情報にアクセスするために、企業ディレクトリの各ユーザーをデータベースの表エントリに関連付ける必要があります。

仮想化シナリオの場合と同じように始めます（ただし、すべてのユーザーが同じ部署に属していることがわかっています）。アダプタ #0 は、ローカル・アプリケーション記憶域で使用され、ツリーのルート位置を保持します。

アダプタ 1 は Active Directory のプロキシとして定義されます。アダプタ 2 は、企業 LDAP ディレクトリのプロキシです。アダプタ 3 は DB アダプタで、RDBMS の該当するユーザー・レコードにマップされます。

最後のアダプタ 4 は JoinView アダプタに基づいて定義されます。JoinView によって、複数の結合アダプタを結合したビューが作成されます。JoinView は、プライマリ・アダプタと 1 つ以上の結合アダプタで構成されます。ここでは、アダプタ 2 の企業ディレクトリがプライマリ・アダプタとして使用されます。つまり、アダプタ 4 で表示されるすべてのエントリは、アダプタ 2 のエントリとまったく同一になります。他に何も定義しない場合、JoinView アダプタはアダプタ 2 のコピーにすぎません。

この段階で、次に結合関係の定義を開始します。JoinView アダプタは、ジョイナを使用して、プライマリ・アダプタのエントリを結合アダプタのエントリと結合します。ジョイナは特殊な Java クラスで、JoinView がアダプタ間での結合の実行方法と情報の共有方法を制御するために使用します。JoinView でサポートされるジョイナの数には制限はありません。オラクル社提供のジョイナ、または統合開発者、コンサルタント、ユーザー自身が開発したジョイナを使用できます。Oracle Virtual Directory には、Simple、OneToMany および Shadow というジョイナが含まれています（詳細は、後の章を参照してください）。



このシナリオでは、Active Directory に対して 1 つ、RDBMS に対して 1 つ、合せて 2 つの結合関係を定義します。Active Directory の結合では、アダプタ 2 に対する Simple 結合をアダプタ 4 に定義します（プライマリ・アダプタであるアダプタ 1 と実際に結合していることに注意してください）。

結合を完了するために、AD のエントリをプライマリ・アダプタに結合する一意の基準を指定する必要があります。

このとき、uid=userprincipalname を使用します。uid はアダプタ #1 にあり、userprincipalname はアダプタ 2（AD）にあります。

2 つ目の結合でも Simple 結合を使用してアダプタ 3 を結合します。このときは、uid=userid を使用して一意に一致するようにします。

最後に、プライマリ・アダプタ（アダプタ 2）ではなくアダプタ 1 を認証用に使用するため、bindadapter 設定は 1 とします。これにより JoinView アダプタは、プライマリ・アダプタではなく結合アダプタに対して認証をテストするようになります。

ユーザーを複数の RDBMS レコード（たとえば権限表）と一致させる場合は、approle=priv のような OneToMany 結合を指定できることに注意してください。この approle は、企業ディレクトリの属性です。この属性は、RDBMS の一連の権限と一致します。ここでこの結合を実行することで、単純なロールを一連の権限に変換します。

アダプタ 4 を構成すると、アダプタ #1、#2 および #3 はエンド・ユーザーに対して表示されなくなります。これには、これらの各アダプタの「Routing Visibility」を「Internal」に設定してください。最終的に、1 つのブランチ（ou=People,o=AppView）があるように見える 1 つのディレクトリが得られます。ディレクトリの ou=People の下をブラウズするときは、アダプタ #4（前の図の青色の部分）すなわち JoinView アダプタを問い合せることになります。このアダプタが問合せを受け取ると、問合せを自動的に変換して、表示されていない他の 3 つのアダプタに渡します。このように処理することで、アプリケーションに対しては、各ユーザーに 1 つのエントリしかないように見えます。

フォルト・トレラント・デプロイの計画

Oracle Virtual Directory は、フォルト・トレラント・システムの実装に際して非常に柔軟性の高い製品です。次の 3 種類のフェイルオーバーについて検討します。

- DNS およびネットワークのフェイルオーバー
- Oracle Virtual Directory のフェイルオーバー
- プロキシ・ソースのフェイルオーバー
- ローカル・ストア・アダプタ・データのレプリケーション

DNS およびネットワークのフェイルオーバー

Oracle Virtual Directory のフォルト・トレランスの実装計画によって異なりますが、クライアントを使用可能な Oracle Virtual Directory にルーティングするためにいくつかの方法を検討することができます。

最も単純な方法は、DNS ラウンド・ロビンを定義することです。この定義では、DNS 管理条件で特定の DNS 名が 2 つの IN A レコードを含みます。こうすることで、特定のアドレスへのリクエストが発生するたびに DNS サーバーが順番のアドレスを与えます（たとえば、**ldap.corp.com** は交互に 192.168.0.1 または 192.168.0.2 になります）。これが役に立つのは、2 つの使用可能なサーバーの間で負荷を分散する場合です。いずれかのサーバーが使用不可になるとそれほど有効ではありません。DNS は障害を認識しないため、障害が発生したサーバーのアドレスの番になるたびにクライアントをそのサーバーに送り続けるためです。

Cisco LocalDirector や F5 Big-IP などのハードウェア・ロード・バランサを使用することもできます。これらの製品では、各サーバーのパフォーマンスが監視されて、本来のロード・バランシングが提供されます。この分野には価格や機能の異なる多くの製品があります。

もう 1 つの方法として、クラスタ内の障害が発生したノードの IP アドレスを切り替えられるクラスタ構成（Veritas など）を使用することもできます。

Oracle Virtual Directory のフェイルオーバー

ローカル記憶域（ローカル・ストア・アダプタ）を使用していないかぎり、Oracle Virtual Directory のフェイルオーバーは比較的容易です。Oracle Virtual Directory は、起動時に読み取る以外は構成ファイルを使用せずに機能するためです。理論上は、同じ構成データを読み取る 2 つのサーバーは同じ機能を自動的に実行します。

ローカル・ストア・アダプタのフェイルオーバー

ローカル・ストア・アダプタを使用するときは、その他の問題すなわちレプリケーションについても考慮する必要があります。レプリケーションは、ローカル・データ・ストアの変更を反映するように 1 つのノードが別のノードを更新するプロセスです。ローカル・ストア・アダプタを使用している場合は、クラスタ・ノード間で（および、場合によっては他の非クラスタ・ノード間でも）レプリケーション承諾を設定する必要があります。レプリケーションを設定すると、1 つのノードがマスターになり、もう 1 つがスレーブになります。たとえば、ノード 1 がマスター、ノード 2 がスレーブになります。この構成では、両方のノードは同等に機能しますが、書込みを処理できるのはノード 1 のみです。処理が終了すると、ノード 1 が自動的にノード 2 を更新します。

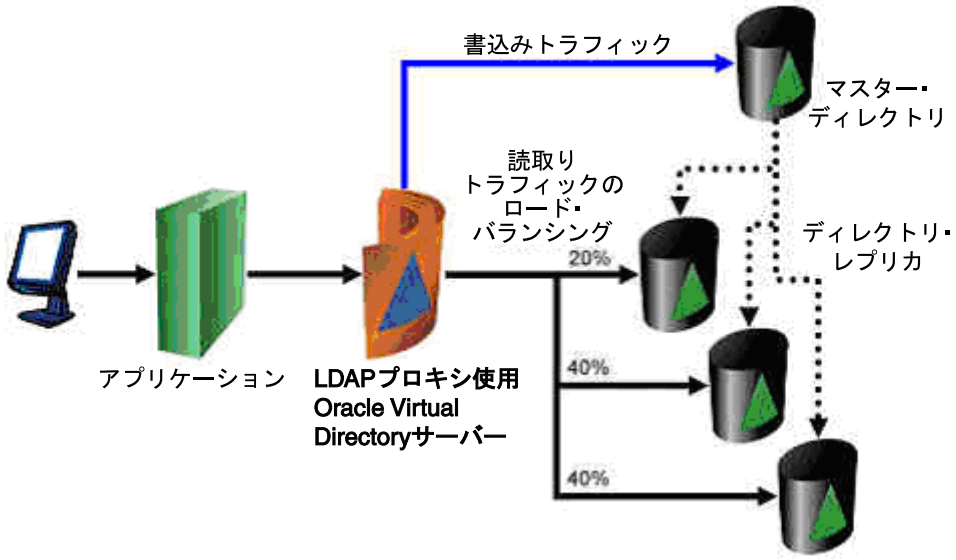
障害に備えて、適切なアクションを実行するようにクラスタ障害の処理スクリプトを構成する必要があります。ノード 2 で障害が発生すると、ノード 2 の復帰時にレプリケーションが再起動されるかぎり何も影響はありません。反対にノード 1 で障害が発生した場合は、ノード 2 をマスターに昇格する必要があります。こうすることにより、ノード 1 がなくてもノード 2 が書込み処理を続行できます。ノード 1 が稼働できる状態に戻る前に、レプリケーション承諾を無効にする必要があります（ローカル・ストア・アダプタの構成を参照してください）。

プロキシ・ソースのフェイルオーバー

Oracle Virtual Directory の組込み LDAP プロキシ・アダプタは、すべての LDAP 準拠データ・リポジトリに対して、高機能のフェイルオーバーおよびロード・バランシング管理を提供します。どのアダプタ（プロキシ・ソース）についても、複数のリモート・ホスト・レプリカを定義して、それぞれに次の特徴を指定できます。

- 読取り / 書込みまたは読取り専用（マスター・ノードまたはレプリカ・ノード）
- 負荷分散のパーセンテージまたは障害時のみの切替え

図 2-2 Oracle Virtual Directory LDAP アダプタとトランザクション・ロード・バランシング



Oracle Virtual Directory には構成可能な接続処理があり、次の項目を指定できます。

- **ハートビート間隔 :** Oracle Virtual Directory がプロキシのオンライン・ステータスを確認する間隔
- **タイムアウト時間 :** 接続に障害が発生したことを判別するまで Oracle Virtual Directory が待機する時間
- **重大性 :** プロキシの結果が問合せ全体に対して重大であることを Oracle Virtual Directory が判別する方法

ハートビート間隔は、Oracle Virtual Directory がサーバーの可用性を継続して確認するために使用します。プロキシがオフラインになると、Oracle Virtual Directory はアクティブ・サーバーのリストからそのプロキシを自動的に削除し、その他の定義済レプリカに負荷を分散させます。サーバーが再び使用可能になったことがハートビートで確認されると、サーバーは使用可能リストに戻されます。

タイムアウトは、TCP/IP 接続で障害が発生したことを判別するまで Oracle Virtual Directory が待機する時間（ミリ秒）です。接続で障害が発生すると、Oracle Virtual Directory はレプリカ・リストの次のサーバーを自動的に試行します。応答するプロキシ・サーバーがない場合、LDAP クライアントがエラー「DSA unavailable」を受け取ります。

問合せが複数のアダプタからの応答を必要とする場合、いずれかのソースが使用不可になっており（すべてのアダプタを問い合せできないため）、該当するアダプタが**「critical」**と指定されていると、Oracle Virtual Directory がエラーで応答します。場合によっては、一部のサーバーを問い合せできるときは Oracle Virtual Directory からの結果を希望することがあります。Oracle Virtual Directory で部分的な結果を返すようにするには、結果を得られなくてもよいアダプタを**「non-critical」**に設定します。

ネームスペースの計画

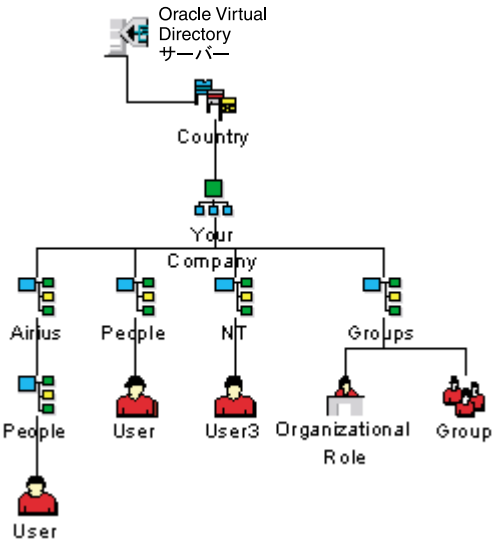
デプロイの計画は比較的単純です。構成を計画するには、新しいディレクトリに構築するディレクトリ構造を考慮する必要があります。この計画アクティビティの進め方の例を次に示します。

次のページの図は、ベースが o=YourCompany,c=US のディレクトリの、実現可能で望ましい仮想ディレクトリ構造を示します。

この仮想ディレクトリには次の 4 つのメイン・ブランチがあります。

- ou=Airius: パートナの LDAP ディレクトリを指します。
- ou=People: 社内の RDBMS を指します。
- ou=NT: Windows NT ドメインを指します。
- ou=Groups: ローカルの Oracle Virtual Directory ディレクトリ・ストアに格納されるグループとロールの情報です。

このとき、LDAP プロキシ、データベース・プロキシ、NTLM アダプタおよびローカル・ストア・アダプタという 4 つのアダプタを構成する必要があります。右の図に、望ましいディレクトリ・ツリーを実装するために構成される 4 つのアダプタを示します。

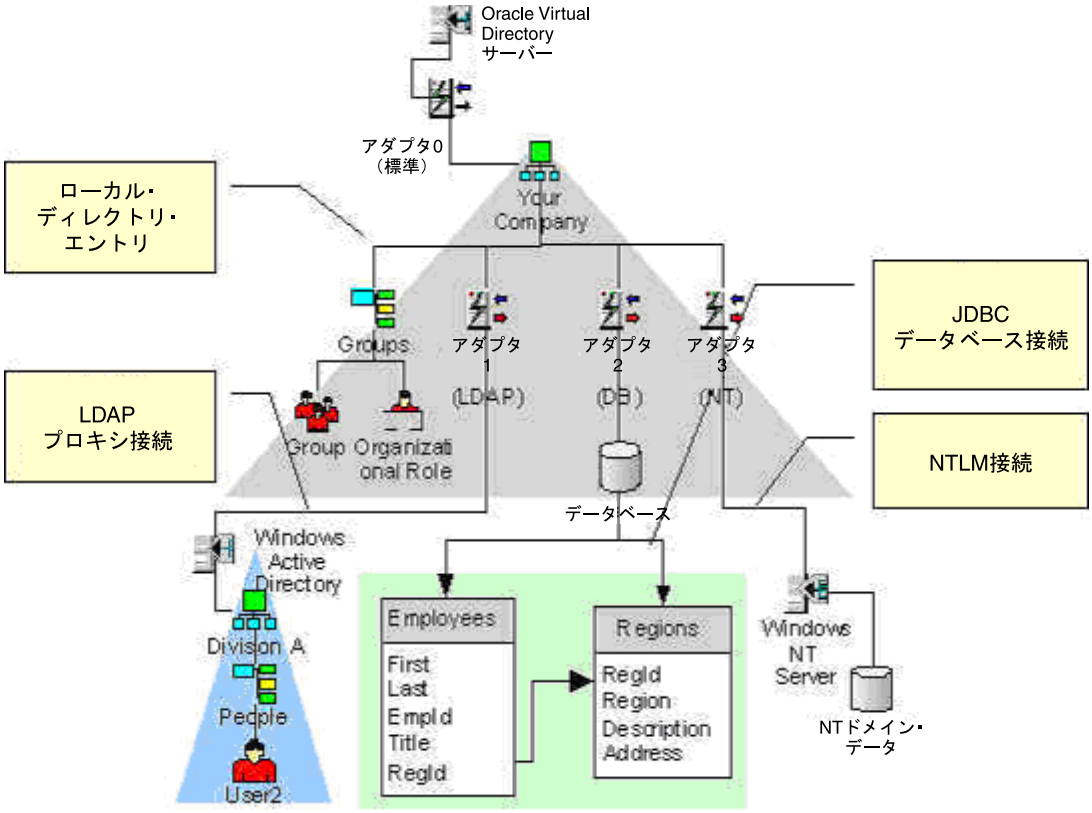


アダプタ #0 すなわちローカル・ストア・アダプタが最初のアダプタです。このアダプタは、ディレクトリのベースを形成し、プロキシの対象にならないエントリを保持します。この場合、Groups の下のディレクトリ・エントリはローカル・ディレクトリに格納されます。

アダプタ #1 は、リモート LDAP ディレクトリと、仮想ディレクトリ・ツリーにマッピングされるリモート・ベースを指定します。この場合、リモート・サーバーの o=Airius.com の下のすべてのエントリは、仮想ディレクトリでは ou=Airius,o=YourCompany,c=US として表されます。

アダプタ #2 はデータベース接続を定義します。このデータベース接続では、2 つの表を使用してディレクトリのエントリが形成されることが指定されます。この場合、2 つの表の結合問合せによるレコードが、仮想ディレクトリのネームスペース ou=People,o=YourCompany,c=US のユーザー・オブジェクトを形成するために使用されます。

アダプタ #3 は、Windows NT ドメインへの NTLM 接続を定義します。このアダプタで作成される階層では、ドメイン内のすべてのユーザー・アカウントが ou=People の下に表されます（たとえば、uid=jsmith,ou=people,ou=NTDomain,o=YourCompany,c=US）。

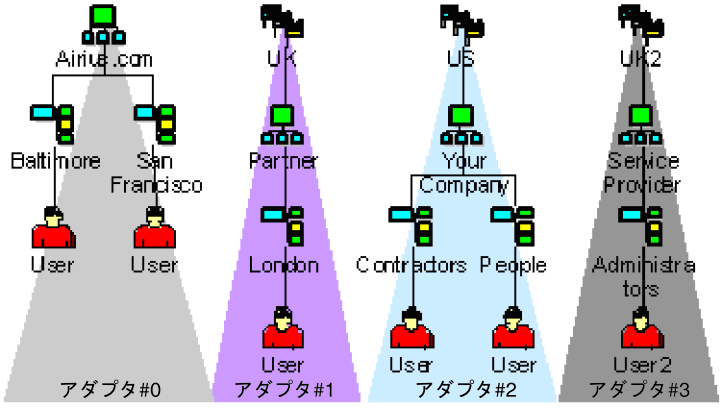


Oracle Virtual Directory では、ディレクトリ・ツリーの各部分のソースを様々な場所に求めることができます。仮想ディレクトリ情報ツリー構造を計画するときは、2 つのアダプタ・ルートが同じルート・ノードを占有しないようにしてください。ただし、アダプタを別のアダプタの子ノードとして表すことは可能です。

アダプタのネームスペース設計例

ここでは、さらに別のシナリオについて説明し、ディレクトリ・ツリー構造を設計するいくつかの方法について示します。

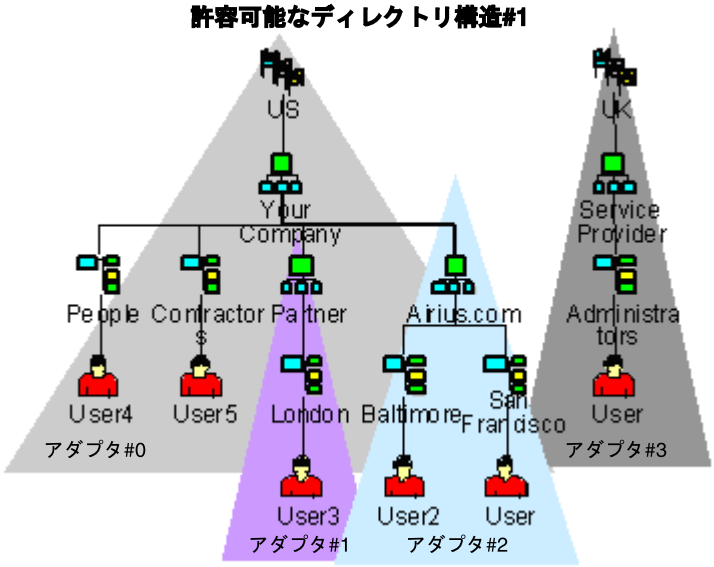
ソース・ディレクトリ構造



このソース・ディレクトリ構造の図には 4 つの個別のディレクトリがあります。目標は 4 つすべてを 1 つの新しいディレクトリ・ツリー設計にまとめることです。Oracle Virtual Directory で実装できる最も基本的なディレクトリ・ツリー設計は変換を行わない設定であることに注意してください。このとき、4 つの個別のディレクトリ・ツリーを含むソース・ディレクトリ構

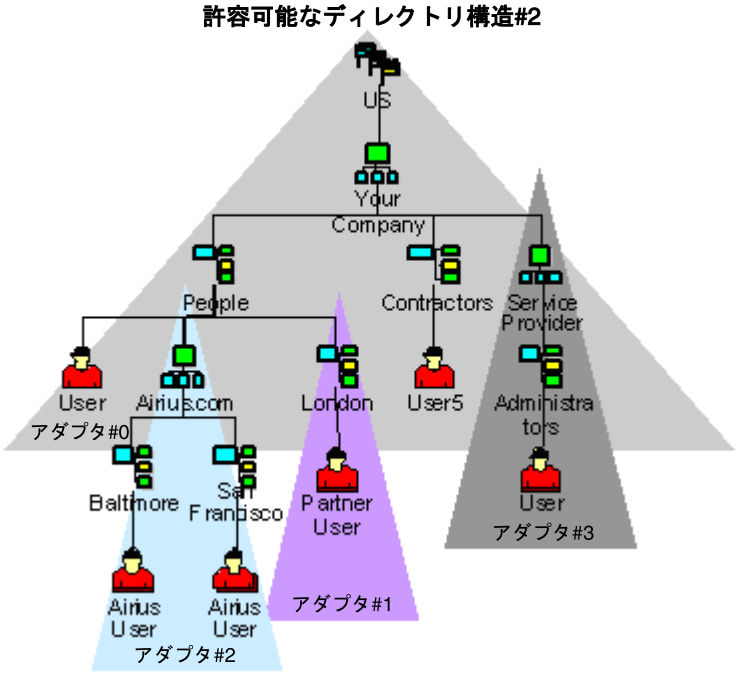
造が Oracle Virtual Directory 内で有効になります。このモードでは、Oracle Virtual Directory は純粋なディレクトリ・プロキシとして稼働し、変換機能は使用しません。

許容可能なディレクトリ構造#1



「許容可能なディレクトリ構造 #1」では、外部の 2 つの会社が o=YourCompany, c=US の子として追加されています。この場合、アダプタ #1 とアダプタ #2 はアダプタ #0 に対してサブツリー・エントリの位置を占めます。このとき、アダプタ #3 は別のネームスペースを占めています。このようにして、管理ネットワーク・サービスを提供するが、他の 3 つの組織のビジネス・アプリケーションには参加しないサード・パーティの会社を表すことができます。この場合、ISP ディレクトリ・エントリを別にしておくことに意味があります。

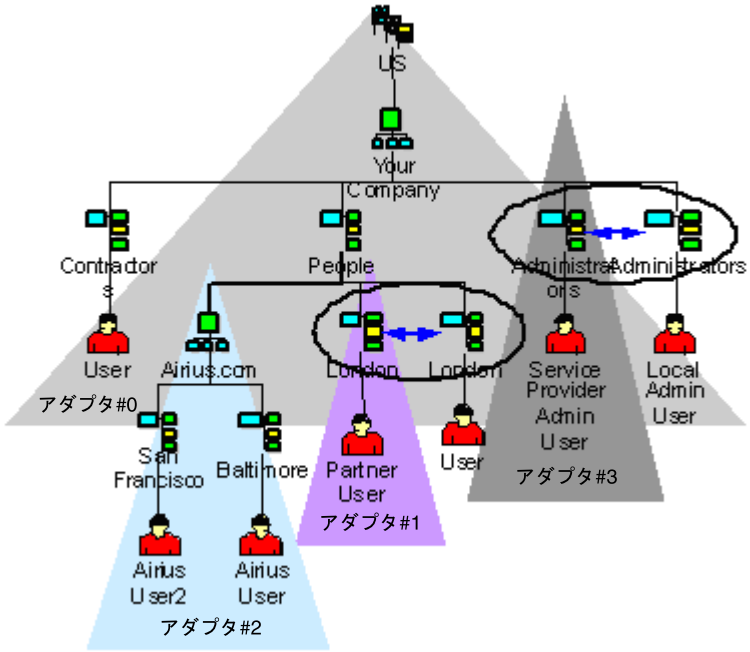
許容可能なディレクトリ構造#2



「許容可能なディレクトリ構造 #2」では、ディレクトリ構造が再設計され、すべてのパートナ・ユーザーが ou=People ブランチの下にあります。アプリケーションの制限のために要件がこのようになることがあります（アプリケーションが `ou=People, o=YourCompany, c=US` の 1 ベースのユーザーしか認証しない場合）。YourCompany の人のエントリは ou=People の直下にありますが、パートナのディレクトリは ou=People の下の組織単位に含まれることに注意してください。組織単位を配置することで、それらのパートナのユーザー固有のアクセス制御グ

ループやロールの作成が容易になり、ネームスペースの競合が回避されます。ツリーのどのノードも 1 つのアダプタが占有することに注意してください。

オーバーラップ・ディレクトリ構造



「オーバーラップ・ディレクトリ構造」の図では、アダプタ #0 に格納されるエントリとアダプタ #1 と #3 のルート・エントリとの間の競合のために 2 つの問題が生じます。検索操作時に、Oracle Virtual Directory はオーバーラップするネームスペースを検索し、すべてのアダプタから一致結果を返します。ただし、変更操作時には、Oracle Virtual Directory は、そのエントリからツリー内を下から順に進んで最初に一致したアダプタ内のエントリしか処理しません（この場合、アダプタ #0 はツリー内で上にあるため、アダプタ #1 と #3 のみが処理されます）。

従来の条件である「オーバーラップ・ディレクトリ構造」が受け入れられない場合、Oracle Virtual Directory は、オーバーラップ・ネームスペースの操作の処理方法を制御するルーティング・フィルタをサポートします。詳細は、次の「ルーティング構成」の項を参照してください。

ルーティングの構成

ルーティングは、さらに複雑なシナリオの設計を可能にする Oracle Virtual Directory の拡張機能です。これには、よりセキュアでパフォーマンスの高い構成を実装できるように Oracle Virtual Directory をチューニングする機能も含まれます。詳細は、このマニュアルの「ルーティング」の章を参照してください。

ルーティングとは

ルーティングは、特定の問合せに対して選択すべきアダプタをディレクトリ・サーバーが決定する処理です。最も基本的なレベルでは、Oracle Virtual Directory はアダプタ接尾辞比較プロセスを使用してアダプタを選択できます。このプロセスでは、特定の検索ベースまたはエントリ DN（追加、変更、削除、名前変更の場合）を調べて、各アダプタの接尾辞（ルート）と比較します。有効範囲によっては、Oracle Virtual Directory は、1 つ以上のアダプタが特定の問合せの影響を受けたかどうかを判別できます。

アダプタ接尾辞の比較は、アダプタ数が少なくアプリケーションの柔軟性が高い場合には非常によく機能します。ただし、通常はさらに柔軟性の高い決定が必要です。ここでルーティングが重要になります。

管理者はルーティングを使用すると、プロキシ・データ・ソースの情報を Oracle Virtual Directory にルーティング・インテリジェント機能という形で知らせることができます。ルーティングにより、Oracle Virtual Directory はディレクトリ操作に細かい条件を付けて、操作が必要な特定の場所に送ることができます。この方法では、既存のディレクトリに無関係な操作による過負荷をかけずに済み、自らのディレクトリに関係ない問合せがパートナに送られなくなります。

Oracle Virtual Directory ルーティング・エンジンは、従来の接尾辞比較に加えて LDAP クライアントの検索フィルタを分析し、処理のために適格なアダプタをさらに絞り込みます。

動的結合エンジンと結合ビュー

動的結合エンジンは、結合ビュー・アダプタと呼ばれる特別な種類のアダプタです。結合ビューは、他の 1 つ以上のアダプタのデータを結合して、リレーショナル・データベースの表のビューのように提供します。結合ビューでは、プライマリ・アダプタのエントリと別のアダプタの結合エントリをリンクする結合ルールを指定することで、LDAP エントリが形成されます。

Oracle Virtual Directory の結合の表現は、データベース結合と似ています。結合ビューは動的で、プロキシ・ソース間の同期は実行しません。目的は、マージしたデータを LDAP クライアントにリアルタイムに提供することです。

結合ビューの機能を次に示します。

- 複数のアダプタおよびオブジェクト・クラスの属性を 1 つの新しい仮想エントリにマージできます。
- 検索時に各アダプタが表示用の特定の属性を選択できます。
- 変更操作時に各アダプタが属性の変更を適用できます。

結合ビューは次の場合に非常に役に立ちます。

- **アプリケーションの統合 :** 現在の LDAP 対応アプリケーションの多くにはカスタム・スキーマ要件があります。多くの場合、企業ディレクトリ・マネージャに企業スキーマの拡張を納得させるのは困難です。ローカル・ディレクトリ・ストアと社内の企業ディレクトリの結合ビューを使用すると、アプリケーション固有のデータを企業ディレクトリに格納する必要なしに、アプリケーションが 1 つの拡張スキーマを認識することができます。さらに、アプリケーションは、企業ディレクトリとの直接統合でもたらされるビジネス上の利点をすべて得ることができます。
- **パスワードの統一 :** 多くの場合、パスワードの同期化は難しい問題です。特に Microsoft Active Directory を利用していると、別の Oracle Virtual Directory または SunOne Directory（以前の iPlanet Directory）など他の企業ディレクトリにパスワードをエクスポートすることが困難です。結合ビューを使用すると、LDAPv3 ディレクトリのエントリを Active Directory のエントリと結合することが可能です。結合ビューは、エントリが LDAPv3 ディレクトリに含まれているように継続して表しますが、認証（バインド）は Active Directory に対して実行します。LDAP クライアントは、LDAPv3 ディレクトリがバインドを完了したと認識しますが、実際のバインドは Active Directory によって実行されています。これにより、パスワードを Active Directory と他の LDAPv3 ディレクトリの間で同期化する必要がなくなります。
- **データの統合 :** 多くの場合、アプリケーションは、企業のメイン・ディレクトリに格納されているよりも多くの情報にアクセスする必要があります。情報はデータベースまたは他のディレクトリに格納されていることがあります。結合ビューを使用すると、個人に関する様々な情報を 1 つのエントリにまとめることができます。

結合ビューの構成

JoinView アダプタの構成は非常に単純です。他のアダプタと同じく、JoinView にも独自の接尾辞（ルート）があります。また、ルーティングなど標準の構成オプションすべての影響を受けます。このアダプタ固有の構成項目を次に示します。

- **Primary Adapter:** 結合ビュー内でデータのプライマリ・ドライバとして使用されるアダプタの名前。プライマリ・アダプタは、JoinView アダプタがディレクトリ階層を構築するときに使用されます。JoinView アダプタのエントリが存在するのは、プライマリ・アダプタに存在する場合のみです。他の任意のアダプタをプライマリ・アダプタにすることもできます。JoinView を定義してデバッグすると、プライマリ・アダプタ独自のルーティング構成を「invisible」に設定して、結合されないエントリを LDAP クライアントで非表示にできます。
- **Bind Adapter:** このパラメータを使用して、バインドの確認を実行するために使用されるアダプタを指定します。LDAP クライアントがプライマリ・アダプタに基づいて DN とバインドするとき、別のアダプタの結合エントリに対してパスワードが確認されることがあります。バインド・アダプタは、プライマリ・ビューまたは結合アダプタのいずれかである必要があります。
- **Join Rule:** 1 つ以上の結合ルールを使用して、他のアダプタとの関係を指定できます。結合ルールでは、結合するアダプタの名前、ジョイナの名前、および選択したジョイナに関連する構成情報を指定します。デフォルトでは、Oracle Virtual Directory には、Simple、OneToMany および Shadow という 3 つのジョイナが含まれています。詳細は次の項を参照してください。1 つの JoinView に複数の結合ルール（1 つの結合関係について 1 つずつ）を含むことが可能です。結合ルールごとに異なるジョイナを指定できます。

JoinView のジョイナ

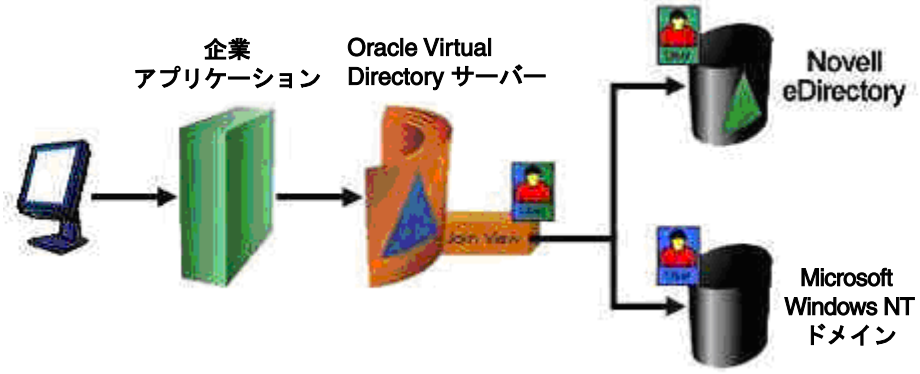
ジョイナは、結合の形成方法やすべてのディレクトリ操作に与える影響をクライアントのかわりに定義する一連のルーチンです。Oracle Virtual Directory は、顧客固有のロジックの作成に使用できる拡張可能なジョイナ・クラスを提供します。顧客は、このクラスで提供される API を使用することで、事前処理や DN マッピング・ロジックを各 LDAP 操作に加えることができます（たとえば、preAdd、preModify、preGet、preDelete、preRename）。

Oracle Virtual Directory には 3 つのジョイナが含まれており、ほとんどの目的に適した機能が提供されます。

- **Simple:** このジョイナは、単純な結合関係に基づいて 2 つのアダプタのデータを結合します。この関係は、remoteattr=primaryadapterattr という形です。条件をカンマ区切りリストで追加して基準を複雑にすることができます。このように追加すると、すべての条件を and で結合する複数条件の結合が作成されます。結合基準によって生成される結果が一意でない場合、Simple ジョイナは検索された最初の一致結果を使用することに注意してください。検索された実際の一致結果は、返されたエントリの操作属性 vdejoindn を確認して判別できます。

  Simple 結合は、結合エントリに対して取得と変更を実行しますが、結合ディレクトリのデータに対する追加、削除または名前変更は実行しません（ただし、操作はプライマリ・アダプタでは実行されます）。

図 2-3 認証のための Simple 結合の例



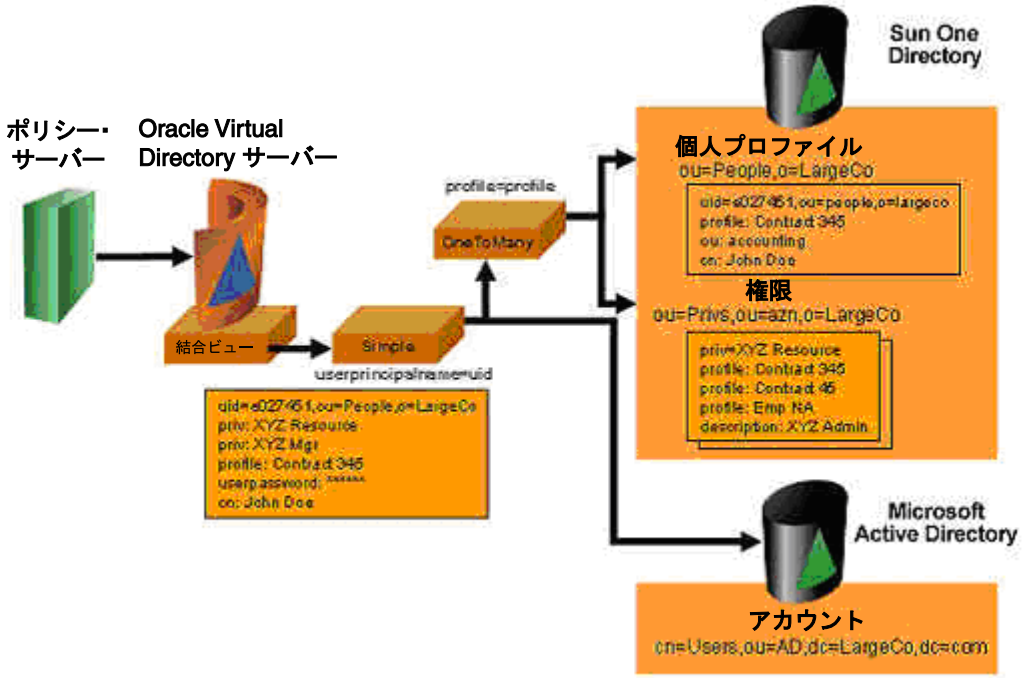
- **OneToMany:** このジョイナの機能は Simple と似ていますが、指定の結合条件で検出されたすべての一致結果を結合する点が異なります。この結合は、複数のロール・オブジェクトまたは ID を 1 つの仮想エントリに統合するときに役立ちます。

  Simple 結合の場合と同じく、OneToMany は結合エントリに対して取得と変更を実行しますが、結合ディレクトリのデータに対する追加、削除または名前変更は実行しません。

  次の図は、ポリシー・サーバーが個人についてポリシー決定をする必要があるシナリオを示します（この場合は Oblix NetPoint）。統合を目的とする場合、ポリシー・サーバーは、権限属性として公開されているユーザーの権限が含まれる 1 つのエントリを認識することが望ましいとされます。この場合、Oblix NetPoint は次の問合せで権限のアサーションをテストできます。

  ```
  ldapsearch -b "uid=e027451,ou=People,o=LargeCo" -s base "(priv=XYZ
  Mgr)"
  ```

図 2-4 OneToMany 結合の例

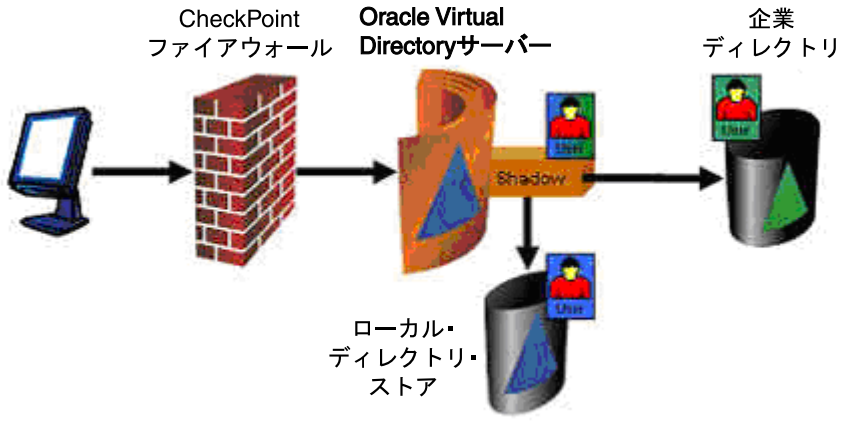


OneToMany ジョイナは、メインの ou=People エントリのプロファイル属性に基づいて、1 つ以上の権限を 1 ユーザーと一致させるために使用されます。OneToMany ジョイナは、エントリと同じプロファイル値のすべての権限を検索して、それらをエントリとマージします。

第 2 段階の結合では Simple ジョイナが使用され、SunOne Directory の組み合されたプロファイルが、ユーザーの Active Directory 資格証明と一緒に使用されます。

- **Shadow:** Shadow ジョイナは、属性データをプライマリ・アダプタとは別のアダプタに格納する必要があるアプリケーションをサポートするように設計されています。Shadow ジョイナは、プライマリ・アダプタのすべての DN をハッシュにエンコードすることで機能します。ハッシュにより、検索を実行する必要なしに、結合アダプタで結合エントリを見つけられるようになります。Shadow ジョイナは、対応するレコードを結合アダプタで見つけられないと、自動的に新しいレコードを作成し、指定の属性を結合アダプタに格納します。Shadow ジョイナはアプリケーションに対してできるかぎり透過的に稼働し、プライマリ・アダプタでの処理と同期してエントリの作成や名前の変更を処理します。

図 2-5 アプリケーション固有データをローカルに格納する Oracle Virtual Directory



この図では、CheckPoint ファイアウォールが Oracle Virtual Directory に接続するように構成されています。Oracle Virtual Directory は、CheckPoint ファイアウォール・スキーマを管理するためにローカル記憶域を使用します。これにより、CheckPoint ファイアウォールの企業ディレクトリへの統合が可能になります。企業ディレクトリ・スキーマをアプリケーション固有データで拡張する必要はありません。かわりに、アプリケーション固有データをローカルに格納することにより、CheckPoint ファイアウォール管理を担当するチームが管理できます。

カスタム・ジョイナ

これまでに説明したジョイナの使用方法に加えて、異なる関係またはより複雑な関係を実装するためにカスタム・ジョイナを開発することができます。特に、ジョイナで複数の段階からなる操作を実行する場合、カスタム・ジョイナにより、実際のシナリオに適した特別なリカバリ・メカニズムを提供できます。カスタム・ジョイナの詳細は、「Java プラグインの開発」の章を参照してください。

3

ディレクトリ・インフラストラクチャの計画

この章では、Oracle Virtual Directory を利用してディレクトリ・インフラストラクチャを計画する方法について説明します。

概要

この章では、従来の高パフォーマンス・ディレクトリ・インフラストラクチャをサポートするときに発生する基本的な問題や、仮想ディレクトリ・テクノロジを使用して大規模なディレクトリ・インフラストラクチャを補完し、フォルト・トレランス、パフォーマンスおよび全体のスケーラビリティを向上させる方法について説明します。

大規模なデプロイでは、次の点に注意してください。

- **レプリケーションの遅延 :** 通常、ディレクトリ・インフラストラクチャは分散計画に依存しています。この計画では、レプリケーションと呼ばれるプロセスでいくつかのサーバーが互いに同期化する必要があります。多くの場合、レプリケーションはマスターからレプリカ・スレーブに対して一方的に行われます。または、Microsoft Active Directory のようにマルチマスター（スレーブなし）のこともあります。
- 稼働している複数のレプリカ・ディレクトリへのレプリケーションを行う 1 つ（または 2 つ）のマスターについて考えてみます。多くの企業ではレプリケーションのレベルが複数あり、1 つのグローバル・マスター・サーバーがリージョン・ハブとレプリケートしています。リージョン・ハブが、クライアント・アプリケーションで使用される本番レプリカにレプリケートします。実際には 1 つのマスターがボトルネックになる可能性があります。または、レプリケーションそのものが止まる可能性もあります。これは、レプリカ・ノードでの索引付けによって各レプリカ・ノードで更新を処理する機能が遅くなるためです。
- **トランザクション・フェイルオーバー :** 大規模なデプロイでは、クライアント・アプリケーションが、ディレクトリ・サービスの障害や管理のために停止する必要がないことが重要です。一般的に、多くのクライアント・アプリケーションはディレクトリ・サーバーの突然の停止または障害に対応できません。クライアント・アプリケーションが適切に作成されていないと、例外が処理されなかったり、アプリケーション障害を引き起こしたりすることがあります。
- **接続の占有による現象 :** 多くのアプリケーションが、フォルト・トレランスや負荷の問題から影響を受けやすくなるのは、ディレクトリ・サービスとの接続を適度にリフレッシュできない場合です。接続が継続されることにより、重要なフォルト・トレランス・インフラストラクチャが機能しません。クライアント・アプリケーションが長時間にわたって LDAP 接続を継続すると、F5 や Cisco などのロード・バランシング製品がアプリケーションの負荷をディレクトリ・レプリカ間で分散できなくなります。接続占有アプリケーションは、レプリカを何時間も独占して、潜在的にそのレプリカに過負荷をかける傾向があります。このために、パフォーマンスが低下したように見えます。
- **多数のクライアントとの同時通信 :** 多くの場合、ディレクトリ・サーバーが処理を強制されている TCP/IP 接続リクエスト数は多すぎます。ディレクトリは LDAP 操作を実行するかわりに、数千のクライアントに対する TCP/IP 接続の確立に時間を費やしています。
- **処理できない大量データ :** シナリオによっては、ディレクトリに対して計画されるデータの純粋な容量が、効率よく処理できる 1 つのサーバーの機能を上回ることがあります。並列化したディレクトリ・サーバー・インフラストラクチャで配信できるように、ディレクトリ情報を複数に細分化する必要があります。デプロイの設計者にとっての課題は、細分化した情報を元どおりにまとめて、クライアント・アプリケーションが 1 つの大容量ディレクトリとして認識するようにする方法です。
- **独自の方法 :** 多くの場合、多目的インフラストラクチャではすべてのデータ統合の問題を解決することはできません。いずれかの時点で、特定のアプリケーションの固有の使用要件に応じてデータをカスタマイズする必要があります。この要件は、1 つのデータ・モデルを実装する従来のインフラストラクチャとは大きく異なります。

この章では、これらの問題それぞれについて説明し、仮想ディレクトリ・テクノロジを使用して大規模ディレクトリ・デプロイを改善および構築する方法を示します。オラクル社では次の仮想ディレクトリ・デプロイ・モードが提供されています。

- **LDAP プロキシ・アダプタ使用 Oracle Virtual Directory**（以前の **Directory Federator Express すなわち DFE**）。このモードでは、LDAP プロトコルのみが使用されます。主な目的は、基本的な情報のルーティング、ファイアウォール機能、および様々な LDAP ソース間での抽出を提供することです。

- **仮想ディレクトリ・サーバー**に LDAP 機能を追加します。リレーショナル・データベース情報（JDBC 経由）および NT ドメイン情報（NTLM）へのアクセスや、ローカル・ディレクトリ・ストア機能が提供されます。Oracle Virtual Directory でも、データ結合機能が提供され、複数のソースに基づくディレクトリ・オブジェクトを形成することができます。

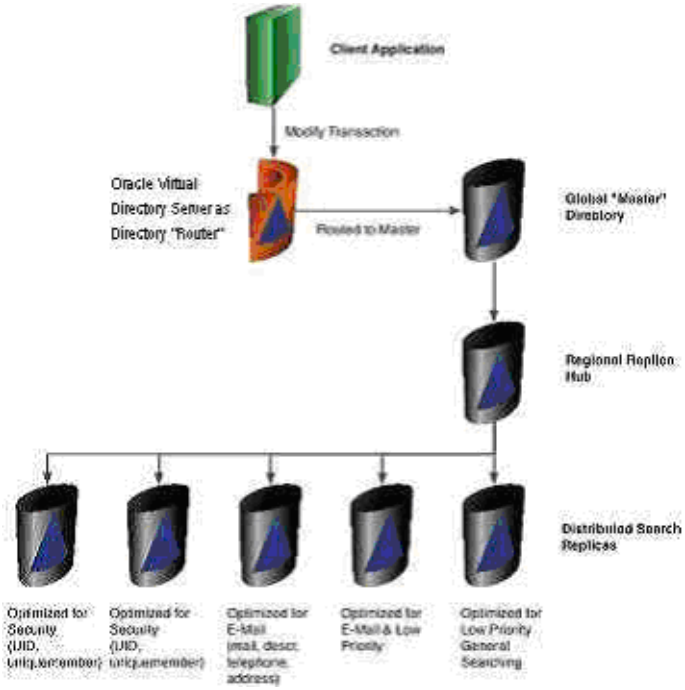
この章の目的では、ディレクトリ・サーバーとして、ほとんどすべての LDAP ディレクトリ・サーバー（IBM Directory Server、Sun ONE Directory Server、Novell eDirectory、他の Oracle Virtual Directory など）を使用できます。LDAP プロキシ・サーバーを参照するコンポーネントとしては、Oracle Virtual Directory 単独または LDAP プロキシ・アダプタ使用 Oracle Virtual Directory のどちらも同様に使用できます。結合またはデータベース・プロキシが必要なシナリオでは、Oracle Virtual Directory が必要です。

レプリケーションの遅延

多くの場合、ディレクトリ・サービスは、長期にわたって 1 つのマスターと複数のレプリケーション・ノードを使用してデプロイされます。その間、顧客がディレクトリの複数の階層を開発して、リージョン内でのレプリケーションを促進し、リージョン・ディレクトリ・ファームをサポートしていることがあります。

時間の経過につれてレプリケーションの速度が低下し、最終的にはディレクトリ・レプリカ・サーバーの情報が古いものになってしまいます。レプリケーションの速度低下の主な原因は、検索索引の管理にあります。多くの場合、索引をみなおして数を最小限に減らすことにより、既存のインフラストラクチャの使用期間を延ばすことが可能です。

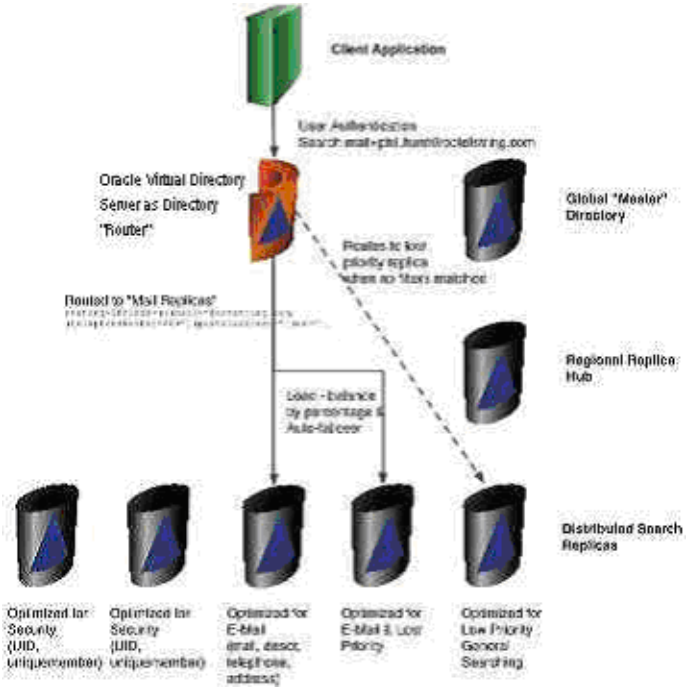
ただし、ある時点で、ディレクトリの索引付けの要求が、個々のサーバーが変更に追い付いてレプリケートできる能力を上回ります。代替策として、レプリカを目的別またはサービスクラス別のノードに分割することを検討してください。たとえば、1 組のレプリカを、ユーザー検索とポリシー・サーバー・リクエストの処理専用とします。もう 1 組をホワイト・ページまたは電子メール検索リクエストの処理専用にします。その他のサーバーは特定のアプリケーションのニーズに合せてチューニングします。



この図では、索引付けの戦略が調整され、サービスクラスのレプリカを作成してレプリケーションを拡張できるようにしています。各サービスクラスは、特定のアプリケーション・クライアント用に設計された一連のディレクトリ・レプリカを定義します。

この場合、Oracle Virtual Directory はマスター・サーバーの場所を自動的に認識し、変更されたトラフィックをマスター・サーバーに直接ルーティングします。不必要なディレクトリ参照操作が回避されます。次の課題は、サービスクラスに基づいた正しい検索に対してアプリケーションを正しいレプリカにルーティングする方法です。

容易な方法の 1 つは、ディレクトリ・レプリカをアプリケーションに直接割り当てることです。ただし、アプリケーションは、特定のディレクトリ・レプリカで構成できる内容を上回る多様な検索を使用することがあるため、この戦略は機能しない場合があります。かわりに、Oracle Virtual Directory 仮想ディレクトリのルーティングの組込みフィルタと除外フィルタを使用して、各検索リクエストを自動的にルーティングすることができます。これらのフィルタを使用して、各プロキシ・ノードが実行できる操作、および実行できない操作を管理者が決定できます。



この図は、典型的なリクエストを示しています。このリクエストでは、アプリケーションがユーザー識別名を確認しようとして UID の検索を行っています。Oracle Virtual Directory は検索フィルタを認識し、リクエストを適切なディレクトリ・レプリカにルーティングします。Oracle Virtual Directory は複数のノードから選択することができ、ノード間のロード・バランシングを提供することに注意してください。これにより、Oracle Virtual Directory は複数のノードに負荷を分散でき、1 つのノードに依存する必要がないためフォルト・トレランスを保証できます。

様々な検索操作（ホワイト・ページなど）またはサービスクラスの場合、Oracle Virtual Directory は代替のディレクトリ・レプリカにルーティングできます。

どのフィルタも一致しなかった場合は、デフォルトのサーバーを指定できます。このサーバーは、低パフォーマンス・サーバーとして設定され、多目的の索引付けが行われているためにレプリケーションが遅れることがあります。

これまでのすべての事例では、単一の Oracle Virtual Directory について説明していることに注意してください。実際は、フォルト・トレランスとサーバーの負荷要件に応じて同一構成の複数の Oracle Virtual Directory がデプロイされます。Oracle Virtual Directory はデータを保持しないため、アーキテクチャの完全な並列化が可能であり、無限に拡張することができます。たとえば、必要であればアプリケーション・クライアントごとに 1 組のサーバーをデプロイできます。各サーバーがデータを保持せず（つまりバックアップが不要）、単純にルーターとして機能するため、インフラストラクチャ全体には各サーバーによって最小限の管理コストが加えられるだけです。

トランザクション・フェイルオーバー

多くの企業は、フォルト・トレラント・インフラストラクチャのデプロイをすでに開始しており、使用可能なディレクトリ・サーバー・ノードに LDAP トラフィックをルーティングするために、F5 BigIP などのデバイスを使用しています。LDAP では非常に細かい単一ユニット・トランザクションが提供されるため、多くの場合、アプリケーションがトランザクション障害を処理できるとみなされます。LDAP 操作が失敗した場合は、アプリケーションが認識して再接続と再試行を行うと常に考えられていました。サービス設計者にとっては、このために大きな問題が発生していました。不備のあるアプリケーションが無効なトランザクションを複数のディレクトリ・サーバーで繰り返すことや、アプリケーションで障害が発生して再試行の必要を認識しないことがあります。

Oracle Virtual Directory は、アプリケーションの負荷を複数のディレクトリ・サーバーに分散するために設計されたインテリジェント接続プーリング・メカニズムを利用して、このような問題に対処します。接続プーリングは個々のトランザクションを複数のサーバーに分散するため、Oracle Virtual Directory は、トランザクションが特定のノードでタイムアウトになるか失敗したときに認識し、その操作を別のノードに送ることを許可します。Oracle Virtual Directory は、問題がデータ障害（クライアントに返される）かサービス障害かを判別します。サービス障害の場合、Oracle Virtual Directory は、トランザクションを使用可能な各サーバーで再試行し、すべてのサーバーについて試します。障害をクライアントに返すのはその後です。

保護を強化するには、ローカル以外のディレクトリ・ノードを Oracle Virtual Directory のサーバー・リストに追加してグローバル・フェイルオーバーを構成できます。このように構成すると、「Remote Hosts」構成でこれらの非ローカル・ノードには負荷パーセンテージとして 0 が割り当てられます。この構成では、Oracle Virtual Directory は、他のローカル・ノードが使用できない場合のみこれらのノードを使用するように強制されます。これにより、ローカルにトラフィックをルーティングする機能が顧客にもたらされます。必要な場合には他のサイトにトラフィックを送ることも可能です。

接続の占有による現象

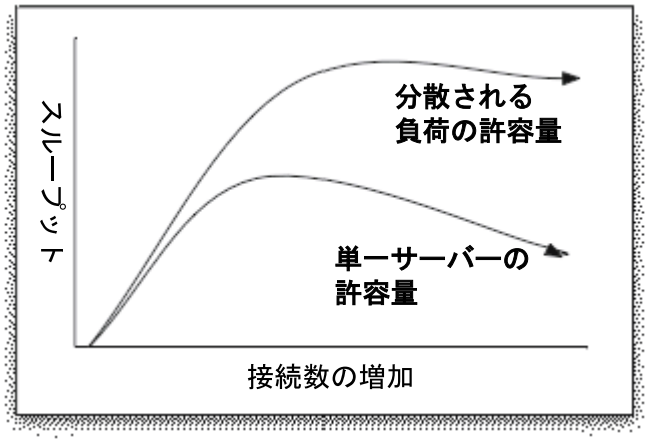
多くの大規模ディレクトリ環境では、いくつかのアプリケーションが接続を占有することがあります。アプリケーションのブートストラップ時に、アプリケーションはディレクトリ・サーバーに割り当てられ（たとえば、F5 BigIP によって）、そのサーバーとの永続接続を確立します。これが次の 2 つの問題の原因になります。

1. 事実上、フォルト・トレランスとロード・バランシングが省略されます。従来のほとんどの方法では、接続に基づいたロード・バランシングとフェイルオーバーが使用されるため、接続がないアプリケーションは過負荷のディレクトリ・サーバーから容易に移動することはできません。
2. 負荷が処理できないほど大きくなります。アプリケーションは 1 つのディレクトリのみを使用する傾向があるため、負荷要件が、割り当てられるノードの処理能力を超える場合があります。いつまでも接続が放棄されないため、管理システムが負荷を再調整する機会はありません。

仮想ディレクトリ・テクノロジは、トランザクションごとに負荷を分散する接続プーリング・メカニズムを提供することで役立ちます。Oracle Virtual Directory は、各プロキシ・ディレクトリと最小限の接続を維持し、負荷の必要に応じて接続を追加します。Oracle Virtual Directory は、単一クライアント接続の自動切替えを提供し、その操作を複数のディレクトリ・サーバーに分散します。これにより、1 つのディレクトリ・サーバー・ノードが過負荷になることがないため、ディレクトリ・サービスの安定感が増します。Oracle Virtual Directory は接続占有クライアントとの間に 1 つの接続セッションを維持する一方で、自らはインフラストラクチャ内の優良なコンポーネントとして負荷分散や定期的な接続リフレッシュを行います。

多数のクライアントとの同時通信

接続を行うアプリケーションが多すぎ、TCP/IP 接続と LDAP バインド・リクエストが多すぎてディレクトリ・サーバーが過負荷になると、接続占有の問題と正反対の状況が発生します。このとき、多数のディレクトリ・サーバー（LDAP および X.500）が、システム・リソースの不足または処理スレッドがなくなるために不安定になることがあります。多くのベンチマークによって、同時接続スレッド数が 7 ～ 10 の場合にほとんどのディレクトリ・サーバーがピーク・パフォーマンス・レベルに達することがわかります。つまり、サーバーが何倍ものコールに応答できる能力がある場合でも、サーバーがリクエストの処理よりも接続の確立に時間を割くようになるにつれて、ピーク・スループットは減少します。



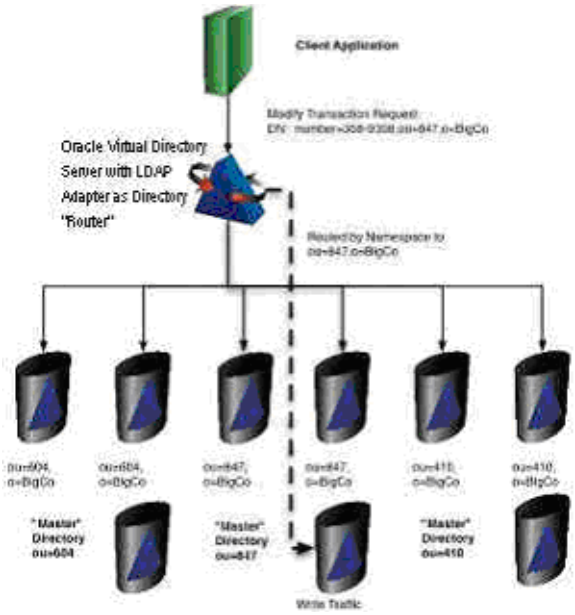
ここで、Oracle Virtual Directory 仮想ディレクトリ接続プーリング・メカニズムが役に立ちます。接続占有クライアントの場合と同様に、Oracle Virtual Directory は、プールを使用して多数のクライアントで接続を共有し、必要な場合にはユーザー・コンテキストを切り替えるために再バインドを使用します（Oracle Virtual Directory の「pass credentials」設定のモードによって異なります）。Oracle Virtual Directory は多数のクライアント接続とリクエストを処理し、プロキシ・サーバーとの減少した接続においてトランザクションを多重化します。既存の接続が再利用されるため、プロキシ・サーバーでのリソースの消費量は大幅に減少し、LDAP のトランザクション処理に集中することができます。

処理できない大量データ

場合によっては、ディレクトリ・レプリケーション・スキームの最適化（「レプリケーションの遅延」を参照）では、非常に大規模なディレクトリに必要な規模を確保するには十分ではありません。このような場合、1000 万あるいは 1 億にもおよぶエントリのディレクトリを作成する機能は、データを細分化して、細分化されたすべてのデータが 1 つのディレクトリ・ビューとして表示される仮想ビューを作成する機能（いわゆる分割統治の手法）に依存します。

Oracle Virtual Directory は、このような仮想化を達成するためにいくつかの手段を提供します。最も単純な方法はディレクトリ階層の活用です。データを階層構造に細分化できる場合、各グループが独自の 1 つ以上のネームスペースを持つ複数のディレクトリ・グループが作成されます。これにより、Oracle Virtual Directory が識別名を確認して、使用すべきディレクトリ・グループを判別し、トラフィックをルーティングできます。

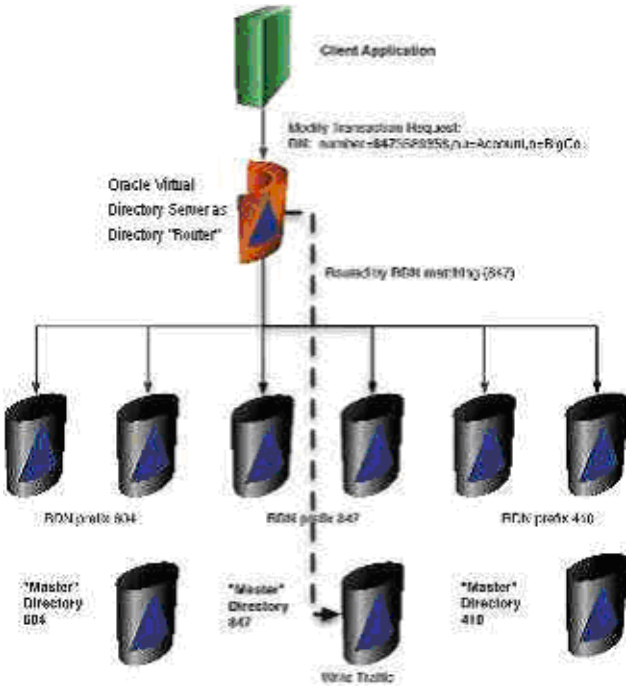
たとえば、ある架空の電話会社が市外局番別に顧客を分けていたとします。Oracle Virtual Directory は、次の図に示すように市外局番を含む組織単位を確認してトラフィックをルーティングできます。すべての変更とバインドの操作では、トラフィックは識別名のみに基づいてルーティングされます。検索操作では、前の「レプリケーションの遅延」シナリオと同じく、検索ベースが o=BigCo であることが必要でネームスペース固有にならない場合、ルーティングの組込みフィルタと除外フィルタが使用されて、検索フィルタに基づいてトラフィックが送られます。



平面的な階層（すべてのエントリが 1 つの親の下にある）が望ましい場合は、相対識別名または RDN（一番左の識別名または DN コンポーネント）の解析またはプリフェッチ操作の使用を検討する必要があります。RDN コンポーネントを解析できる場合は、RDN を解析してルーティング決定を行うプラグインの使用によってルーティングが確立されます。

たとえば、DN が `number=6046331751,ou=Account,o=BigCo` という形式の場合、ルーティング・フィルタは、最初の 3 桁（この場合は 604）に基づいて特定のディレクトリ・グループを選択できます。

DN が `uid=pjdhunt,ou=Accounts,o=BigCo` という形式の場合、ルーティングはハッシュ表を使用して、a ～ l、m ～ r および s ～ z で始まるアカウントがそれぞれ 3 つのサーバーに分かれていることを判別します。



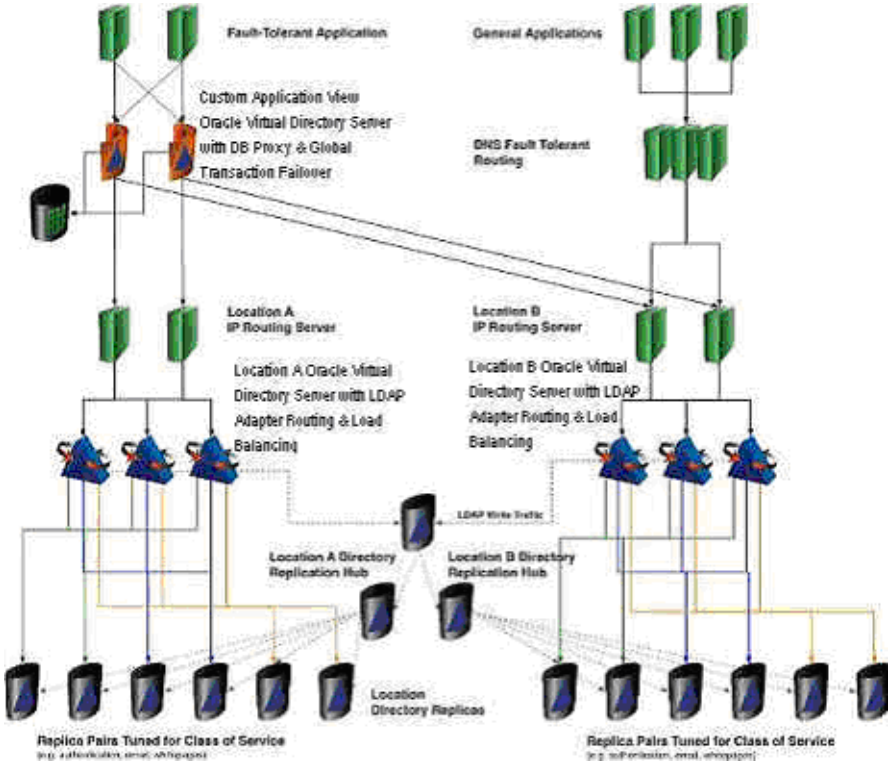
この場合は、標準ルーティング機能を補足するために使用できるルーティング・プラグイン・メカニズムを使用します。このプラグインの目的は、データ分けの基準を顧客が指定できるよ

うにすることです。このために、理想としては、選択される基準（DN の場合もフィルタとベースの場合にも）がすべてのトランザクションで使用可能であることが必要です。

最後の方法はプリフェッチの使用です。データの分割が（少なくとも Oracle Virtual Directory にとって）予測不可能な方法で行われた場合、Oracle Virtual Directory は顧客固有の基準に基づいてディレクトリを検索して、正しいリポジトリを探す必要があります。たとえば、LDAP の変更操作では、変更リポジトリを探す検索を最初に実行する必要があります。バインド、削除および名前変更の操作でも同様の要件があります。LDAP の追加操作では、どのリポジトリが追加リクエストを受け取るかを判別できるように、追加リクエストに十分な情報を含める必要があります。状況によっては、サーバーがインフラストラクチャでエントリを探すために使用する特殊なマスター・ディレクトリによって、パフォーマンスのオーバーヘッドが抑えられます。

結論 - 様々な機能の活用

Oracle Virtual Directory は、グローバル・ディレクトリ・サービス・デプロイの様々な局面においてフォルト・トレラント構成で使用できます。Oracle Virtual Directory は、ロード・バランサとしてサイトの IP ルーター（たとえば Websphere Edge Server、F5 BigIP）とサイトのレプリカ・サーバーの間に配置できます。Oracle Virtual Directory は、その位置でサーバー間のトランザクション・レベルのロード・バランシングとフォルト・トレランスを提供します。ロード・バランシングに加え、Oracle Virtual Directory は、リレーショナル・データベース・ソースの情報（JDBC 経由）など、インフラストラクチャ・レベルの複数のデータ・ビューを提供できます。



特殊なディレクトリ・ビューまたはグローバル・トランザクション・フェイルオーバー機能を必要とするアプリケーションでは、Oracle Virtual Directory をミドルウェア・コンポーネントとしてアプリケーション・サーバーに直接デプロイできます。この戦略により、サイトで障害が発生した場合に、アプリケーションが様々なロケーションを切り替えることができます。通常、この構成では、Oracle Virtual Directory がロード・バランシングを提供するのはローカル・レベルのみですが、ローカル・サービスが失敗した場合のみ他のロケーションのノードに切り替えます。

Oracle Virtual Directory は柔軟性が高いため、ディレクトリ設計者は、複雑で堅牢なディレクトリ・サービス・インフラストラクチャを開発することができます。統合ツールとしての

Oracle Virtual Directory は開発者を支援し、エンタープライズ・インフラストラクチャを使用する際に最も簡単な方法を活用できるようにします。情報のルーターとしての Oracle Virtual Directory は迅速にデプロイでき、管理が容易で、非常に低いコストで運用できます。

つまり、Oracle Virtual Directory では、大規模デプロイを効果的に管理するために必要な機能とパフォーマンスが提供されます。企業が社内全体にわたるデータ問題の解決を模索しているとき、Oracle Virtual Directory は最も困難な課題に対して複数のソリューションを提供します。

4

Oracle Virtual Directory Manager

この章では、Oracle Virtual Directory の管理システムである Oracle Virtual Directory Manager の概要を説明します。

概要

Oracle Virtual Directory 10.1.4 には、Eclipse™ 3.0 プラットフォームに基づいた、最新の高度な管理と操作のシステムである Oracle Virtual Directory Manager が含まれています。このシステムでは、1 つ以上のデータセンターの複数の Oracle サーバーを一箇所で管理および監視する容易な方法が提供されます。

Eclipse™ は、世界的に広く使用されているオープン・ソース・リッチ・クライアント・プラットフォームであり、最新の Rational ツール、IBM 開発者用ツールおよび次期バージョンの Lotus Notes の基礎になっています。システムを Eclipse に基づけることにより、Eclipse 環境に親しんだ数万人のユーザーが Oracle のプラットフォームを戸惑わずに使用することができます。また、無数の市販ソフトウェアや ISV 開発製品によって Oracle プラットフォームを拡張することが可能になります。

この管理システムには、Eclipse ベースのクライアント（Oracle Virtual Directory Manager）と、各 Oracle Virtual Directory で実行する管理 Web サービスの両方が含まれます。Oracle Virtual Directory Manager クライアントは、標準の Web サービス操作を使用してサーバーと通信します（SOAP over HTTP/HTTPS）。また、ほとんどすべての場合にハード・サーバーを再起動せずに実行構成を変更することができます。

Oracle Virtual Directory Manager システムでは、次の 3 種類の使用方法がサポートされています。

- **操作と管理 :** 管理システムを使用して、オペレータは、サーバーの状態を簡単に確認して監視でき、サーバーやサブコンポーネント（アダプタなど）の起動と停止といった操作タスクを実行できます。また、サーバーをリモートで管理でき、全体の変更制御プロセスの一部として構成を更新できます。
- **統合 :** 新しい管理 GUI を使用して、設計者および統合者は、新しい構成をサーバーを使用してテストおよび開発できます。この構成によって、Oracle ディレクトリおよび仮想ディレクトリ・サービスを使用できる新しいアプリケーションをサポートします。統合者は、データ・ソース間で情報をマップするときにカスタム変換を実行する、使用しやすいマッピング・スクリプトを開発できます。このようなスクリプトは、サーバーをリセットせずに、実行中のサーバーにインストールしてデプロイできます。
- **開発 :** 新しい管理 GUI には統合開発環境（IDE）が含まれます。これにより、Oracle サーバーの機能を拡張するために使用できる新しい Java ベース・プラグインの開発が可能になります。

Oracle Virtual Directory Manager クライアント

Oracle Virtual Directory Manager では、パースペクティブと呼ばれる複数のプレゼンテーション・モードで情報が整理されます。**ディレクトリ管理パースペクティブ**は、Oracle サーバーの管理と維持を行うために使用するメイン画面のセットであり、デフォルトで表示されます。

ディレクトリ管理パースペクティブ

ディレクトリ管理パースペクティブは大きく 3 つの領域に分かれます。左上の部分はサーバー・ナビゲータです。**サーバー・ナビゲータ**を使用すると、様々なサーバーのファイルをブラウズして、様々なアクション（編集、停止、起動）を実行することができます。該当するサーバー・コンポーネントを右クリックすることで、サーバーに対するほとんどのアクションを実行できます。

画面の右上は、エディタ領域です。画面のこの領域では、1 つ以上の編集セッションを開くことができます。開いている各エディタは、この領域の上部にあるタブで表されます。

画面の下の部分は、ブラウザ・ビューなど様々なユーティリティ・ビューのために用意されています。**ブラウザ・ビュー**では、LDAP サーバーやデータベースに接続して使用可能なデータをブラウズすることができ、Oracle Virtual Directory の構成や構成のテストに役立ちます。

もう 1 つのビュー、**ログ・ビュー**では、Oracle Virtual Directory にリモートで接続してアクティビティを表示できます。このビューにはこれから発生するアクティビティが表示されます。履歴ログ・データは表示できないことに注意してください。履歴ログ・データを取得するには、サーバーのログ・ディレクトリにあるサーバーのファイルを直接使用します。

ワークスペースとパースペクティブについて

Oracle Virtual Directory Manager/Eclipse 環境は、高機能のデスクトップ環境であり、ユーザーが情報のレイアウトをあらゆる方法で編成することができます。アプリケーションが実行しているとき、表示される画面の領域はワークスペースと呼ばれます。ワークスペースには 1 つ以上のビューとエディタ領域があり、合せてパースペクティブを形成します。パースペクティブは、デフォルトで編成されるビューとエディタをまとめたものです。ディレクトリ管理または Java 開発など特定の機能がサポートされます。他のパースペクティブにアクセスするには、「Window」→「Open Perspective」の下のドロップダウン・メニューを選択します。

Oracle Virtual Directory Manager では、個人のニーズに合せてパースペクティブを設定できます。ビューやエディタのオープンとクローズや、デスクトップのウィンドウの再編成をユーザーが行えるようになります。Oracle Virtual Directory Manager は、このようなレイアウトの変更を記録して、編集セッションの終了後も保存しておきます。デフォルト・レイアウトに戻る場合は、「Window」→「Reset Perspective」を選択して、デフォルト・パースペクティブのレイアウトとビューに戻します。

さらに作業領域が必要な場合は、次のように設定します。Oracle Virtual Directory Manager では、2 つ目のワークスペース・ウィンドウを作成できます。これは、複数のアクションを同時に実行できるマルチヘッド・ワークステーションで役立ちます。ウィンドウを作成するには、「Window」→「New Window」を選択します。

その他のパースペクティブ :

- **CVS Repository、Team Synchronizing**: これらのパースペクティブを使用するのは、CVS リポジトリでファイルを格納または取得する場合です。
- **Java、Java Browsing、Java Type Hierarchy、Debug**: これらのパースペクティブを使用するのは、Java クラスまたは Oracle Virtual Directory Java プラグインの開発を計画している場合です。
- **Resource**: Eclipse でファイル・オブジェクトのブラウズや処理を行うときに役立つ多目的のパースペクティブです。

ビューについて

Oracle Virtual Directory Manager/Eclipse では、ウィンドウが 2 つの基本的なタイプに分けられます。エディタとビューです。通常、エディタは 1 つ以上のファイルの内容を編集するときに使用し、ビューは情報のサマリーを表示するときや、様々な種類の情報をナビゲートするときに使用します。たとえば、ディレクトリ管理パースペクティブには、サーバー・ナビゲータというビューがあります。このビューを使用すると、Oracle Virtual Directory Manager で管理されているシステムのファイル構造をナビゲートすることができ、様々なサーバーやサーバーのサブコンポーネントをナビゲートしてそれらに対するアクションを実行できます。

次に一部のビューとその使用目的を示します。

ディレクトリ管理ビュー :

- **サーバー・ナビゲータ :** サーバー間をナビゲートするためのメイン・ビューです。様々な管理アクションが提供されます（アダプタの追加、マッピング、停止と起動など）。
- **ブラウザ :** このビューは、LDAP および RDBMS ソースをブラウズしたり、Oracle Virtual Directory 仮想ディレクトリ・データをブラウズしたりするために使用します。このビューによって、ディレクトリ・ソースをナビゲートしてデータを取得でき、仮想ディレクトリで表されるようにデータを確認できます。
- **サーバー・ログ :** このビューは、管理者が Oracle Virtual Directory の様々なコンソールやアクティビティのログを表示するために使用できます。このビューを起動すると新しいイベントしか表示されないことに注意してください。アクティビティの履歴は表示されません。

基本ビュー :

- コンソール : Eclipse コンソールで STDOUT に送信されるすべてのメッセージを見ることができます。
- ナビゲータ : 現在のワークスペース・ディレクトリに格納されている実際のファイル構造をブラウズできます（リソース・ナビゲータとも呼ばれる）。デフォルトの Oracle Virtual Directory エディタのかわりにテキスト・エディタを使用してサーバー構成ファイルを編集する場合に、このビューが特に役立ちます。
- タスク : 任意の数の予定タスクを保持できるビューです。アクティビティを管理するために使用できます。

Java ビュー :

- Java の開発をサポートするために様々なビューが提供されます。Java プロジェクトを開始するには、「New」→「Other」を選択し、「Java」→「Java Project」を選択します。

ディレクトリ・プロジェクトの開始

Oracle Virtual Directory Manager では、様々な編集ファイル、管理ファイル、開発ファイルをプロジェクトと呼ばれる任意のグループに分けることができます。Oracle Virtual Directory Manager のディレクトリ・プロジェクトは、管理対象の 1 つ以上の Oracle Virtual Directory と、ディレクトリ・ブラウザ・ビューを使用するディレクトリ・ブラウズ・セッションを保持できます。

ダイアログが表示されたら、該当するプロジェクト名を入力します。これがメイン・フォルダ名になり、この下にすべてのローカル・プロジェクト情報が格納されます。

通常、プロジェクトの内容は、Oracle Virtual Directory Manager ワークスペース・フォルダに格納されます。ファイルを他の場所に格納する場合は、「Use default」の選択を解除して新しいディレクトリを選択します。カスタム・フォルダはプロジェクト・レベルのみで作成してください。

新しいサーバーの定義

サーバーを管理するには、事前に Oracle Virtual Directory Manager にサーバーを追加する必要があります。これは new Oracle Virtual Directory ウィザードを使用して行います。このウィザードによって、既存の Oracle Virtual Directory の新しいサーバー定義が Oracle Virtual Directory Manager に定義されます。ウィザードが完了すると、定義した Oracle Virtual Directory 構成ファイルのコピーがローカル・エディタ・システムに送られます。

ヒント : Oracle Virtual Directory への接続を試行する前に、Oracle Virtual Directory が実行していることを確認します。Windows ではサービス・パネルを調べます。実行していない場合は、「スタート」メニューを使用してコンソール・モードでサーバーを起動します。UNIX プラットフォームでは、sh ./vde.sh 起動コマンドを使用してサーバーを起動します。

「New Oracle Virtual Directory」ダイアログが開くと、サーバー情報を格納するフォルダを指定するように求められます。通常、ここでプロジェクトまたはサブフォルダを指定します。プロジェクトをまだ作成していない場合は、前の項の「ディレクトリ・プロジェクトの開始」を参照してください。

サーバー名としては、サーバーをローカルで参照するための任意の名前を選択できます。これは、サーバーの構成に格納されるサーバーの実際の名前とは異なることに注意してください。

リモート・ホスト情報は、管理するサーバーの IP アドレスまたは DNS、および管理ポート（通常 8888）です。管理 DN とパスワードについては、ルート識別名か、管理メンバーシップ URL の一部として定義された識別名のいずれかを使用する必要があります。その他すべてのユーザーは拒否されます。

最後に、「secure」を選択することで、管理セッションの SSL 暗号化を有効にできます。SSL はサーバーではデフォルトで有効です。また「secure」をクリックすると、Oracle Virtual Directory Manager がサーバーの証明書を信頼しているかどうかが確認されます。信頼していない場合は、信頼する機会が与えられます。

注意： Secure モードがサーバーで現在構成されていない場合、サーバーで有効にすることはできません。この設定は、Oracle Virtual Directory Manager がサーバーと通信するときに SSL の使用を試行するかどうかのみに影響します。サーバーの構成を変更するには、「Manage Server Keys」を使用してサーバー証明書を準備し、Oracle Virtual Directory エディタを使用して、管理ゲートウェイと任意のクライアント・リスナー上で必要に応じて SSL を有効にします。

「Certificate Trust Validation」ダイアログを使用すると、リモート・サーバーによって提供される証明書を信頼するように Oracle Virtual Directory Manager を指定できます。このダイアログは、「new Oracle Virtual Directory」で「SSL」を選択すると自動的に表示されます。「Select Certificate」コンボ・ボックスに複数の証明書が表示されています。特定のサーバー証明書を信頼するように選択できます。または、証明書が親認証局（CA）で署名されている場合は、認証局を信頼するように選択できます。CA を信頼することで、Oracle Virtual Directory Manager は、同じ CA で生成された他のサーバーの証明書も自動的に信頼します（このダイアログは表示されません）。「Trust」をクリックすると、Oracle Virtual Directory Manager はその証明書を信頼できる証明書のローカル・ストアに追加します。信頼しない場合、Oracle Virtual Directory Manager は証明書を格納せず、サーバーとの SSL 通信は不可になります。

SSL 通信を選択した後で、Oracle Virtual Directory Manager がサーバーを前から信頼していることを判別すると、信頼がすでに確立されていることを確認するダイアログが表示されます。

接続情報を入力したら「Finish」ボタンをクリックします。Oracle Virtual Directory Manager がサーバーの認証を試行し、すべての構成情報を編集のためにローカル・クライアントにコピーします。ウィザードによって、すべての構成ファイル、マッピング・スクリプトおよびスキーマ・ファイルがローカルの Oracle Virtual Directory Manager クライアント・ファイル・システムにコピーされます。すべての構成情報が保存されたら、ウィザードによってエディタ・セッションが自動的に開始されます。

新しいアダプタの定義

アダプタを追加するには、サーバー・ナビゲータ・ビューでサーバーを右クリックして「New」→「< アダプタ・タイプ >」を選択します。< アダプタ・タイプ > は追加するアダプタのタイプです。

ウィザードが画面に表示されたら、アダプタを追加するサーバーを選択します。アダプタがテンプレートをサポートする場合は、バンドルされているテンプレートまたは前のセッションで保存されたテンプレートのいずれかを選択できます。テンプレートは、保存されている構成のコピーです。

テンプレートを使用すると、複雑な設定を容易に繰り返すことができるため構成にかかる時間を短縮できます。たとえば、Oracle Virtual Directory Manager には、Active Directory、IBM Directory、Novell e-Directory および Sun One Directory 用のテンプレートが同梱されています。これらのテンプレートには、各製造業者に対応する様々なデフォルト設定が含まれています。

アダプタ名としては、アダプタを参照するために使用する一意の名前を入力します（ウィザードによってデフォルトでも生成されます）。

ヒント： 多くのフィールドにはコンボ・ボックスやドロップダウン・ボックスがあることに注意してください。Oracle Virtual Directory Manager は、以前使用した設定を自動的に記録し、このようなドロップダウン・ボックスから簡単に選択できるようにしています。

すべての設定を完了したら、「Finish」をクリックします。ウィザードによってローカル・エディタ・セッションに構成が追加され、新しいアダプタのエディタ・ウィンドウが開かれ、プロセスが終了します。

データベース・アダプタに対する JDBC の構成

データベース・アダプタの設定には次の 3 つの手順が必要です。

1. 適切な JDBC ライブラリがロードされていることを確認します。
2. データベース接続を定義します。
3. 表データを LDAP オブジェクトにマッピングする方法を定義します。

JDBC ライブラリの準備

データベースに最初に接続するときは、サーバーに JDBC ライブラリを準備する必要があります。このためには、JDBC のデータベース製造業者が必要とする JAR、dll および他のライブラリを指定します。これらのファイルをインストールするには、右クリックして「Connect To Server」を選択し、サーバーに接続します（まだ接続していない場合）。接続したら、サーバーを右クリックして「Manage Libraries」を選択します。

ダイアログ・ボックスが開くと、「Deployed Libraries」と「Files To Be Deployed…」という 2 つのウィンドウが表示されます。左側のウィンドウには、サーバーにすでにデプロイされているライブラリが表示されます。右側（おそらく最初は空白）には、デプロイするために選択したライブラリが表示されます。

「Select New…」をクリックすると、ファイル・ブラウザが開き、デプロイする必要があるすべてのライブラリ・ファイルを選択することができます。ファイル・ブラウザでは一度に複数のファイルを選択できます。または、必要なファイルすべてを選択するまでプロセスを繰り返すこともできます。選択が終了したら、選択が完了したファイルのリストが右側のウィンドウに表示されます。準備ができたら、「Deploy…」ボタンをクリックしてすべてのライブラリをサーバーに一度にデプロイできます。

デプロイが終了したら、「Close」ボタンまたはライブラリ・ウィンドウの右上隅の「X」をクリックしてウィンドウを閉じます。

Oracle Virtual Directory への接続またはログイン

変更を保存するか他の操作コマンドを実行する前に、まずサーバーに接続する必要があります。サーバーのアイコンを右クリックし「Connect」を選択して、サーバーで認証を行います。Oracle Virtual Directory Manager で新しいサーバーを定義した直後の場合は、すでにサーバーに接続していることに注意してください。

サーバーにまだ接続していない場合は、管理 DN とパスワードの入力を求められます。入力した情報が正しい場合は確認を受け取ります。

今後確認を受け取る必要がない場合は、「Don't show this again」を選択にすると、Oracle Virtual Directory Manager によってこの希望が記憶されます。

接続すると、ナビゲータで次の新しいコマンドを使用できるようになります。

「Reload from Server」を使用すると、ローカルの構成データをサーバーの現在のデータでリフレッシュできます。

「Save to Server」を使用すると、Save ウィザードを使用して現在の編集内容をサーバーに保存できます。

「Start」、「Stop」、「Restart」を使用すると、必要に応じてサーバーを起動および停止できます。アダプタを直接右クリックすると、個々のアダプタを起動および停止することもできることに注意してください。

「Show Log」を使用すると、Oracle Virtual Directory Manager がサーバーから Oracle Virtual Directory のログ・メッセージの受信を開始します。

サーバー・ログ・ビュー

サーバー・ログ・ビューでは、Oracle Virtual Directory のコンソールとアクセスのログを対話的に監視することができます。ログ・ビューをアクティブ化するには、サーバーを右クリックして「Show Log」を選択します。

これらのメッセージは画面下部のサーバー・ログ・ビューに表示されます。

注意： サーバー・ログ・ビューには、サーバーで発生する新しいアクティビティが表示されます。このビューにはアクティビティの履歴は表示されません。

ログの停止と開始に緑色の「Go」ボタンと赤色の「Stop」ボタンを使用できます。このビューには各ログ・ファイル・タイプに対応する 3 つのタブが表示されます。

ヒント： このビューの使用には注意してください。ログを監視している管理者が多すぎる場合はサーバー・パフォーマンスが影響を受けることがあります。

分析のための CSV 形式での access.log データのエクスポート

Oracle Virtual Directory Manager には、「Export to CSV」機能があり、Oracle Virtual Directory の access.log ファイルに格納されている接続データに関するレポートを実行できます。「Export to CSV」機能では、Oracle Virtual Directory の access.log データがカンマ区切り値（CSV）形式で保存されます。これは、Microsoft Excel または Oracle Database などのレポート作成システムにインポートすることができます。この access.log データには次の情報が含まれます。

- アクセス時刻
- LDAP クライアントの IP アドレス
- 接続が認証されたユーザー
- 操作（たとえば、バインドまたは検索）
- LDAP 結果コード

次の手順を実行して、「Export to CSV」機能を使用し、Oracle Virtual Directory の access.log データを CSV 形式で保存します。

1. Oracle Virtual Directory Manager で、保存するデータを含むサーバー・インスタンスのエントリを開きます。
2. 「Engine」エントリを開きます。
3. 「Logs」を右クリックし、「Download Logs」オプションを選択します。
4. ダウンロードして CSV 形式で保存する access.log を選択します。「OK」をクリックします。access.log がダウンロードされた後で「Download Complete」プロンプトが表示されます。
5. 「Logs」を右クリックし、「Export to CSV」オプションを選択します。
6. ダウンロードした access.log を選択し、「OK」をクリックします。access.log が CSV 形式でエクスポートされた後で「Download Complete」プロンプトが表示されます。
7. ログ・データをフィルタ処理する日付と時刻の範囲を入力し、「OK」をクリックします。
8. CSV 形式にエクスポートした access.log データを表示するには、「Logs」エントリの下のアクセス・ログ名をクリックします。

Oracle Virtual Directory Manager は、CSV ファイルを <Oracle Virtual Directory Manager Home>¥workspace¥<project name>¥<Oracle Virtual Directory>¥log¥access.csv に保存します。このとき、<Oracle Virtual Directory Manager Home> は Oracle Virtual Directory Manager がイ

ンストールされているディレクトリ、<Oracle Virtual Directory> は Oracle Virtual Directory の名前、<project name> は Oracle Virtual Directory 管理プロジェクトの名前です。

変更の保存

Oracle Virtual Directory Manager システムでは、サーバー構成情報をオフラインで編集し、サーバーにデプロイする準備ができるまで変更内容を保存しておくことができます。準備ができたら、サーバーに接続して「Save to Server…」を選択します。これによって、Save ウィザード・システムが開始されます。

項目の選択

Save ウィザードが変更内容を自動的に確認して、更新する必要がある特定のコンポーネントを判別します。これにより、特定の変更のみをサーバーに加えることができ、コンポーネントの再起動を最小限に抑えられます。選択ウィンドウで項目を選択または選択解除することで、識別されたすべてのコンポーネントの保存または変更されたコンポーネントのサブセットのみの保存を選択できます。

バージョン競合の確認

Oracle Virtual Directory では、管理者がサーバーでの構成の変更を記録できるようにバージョン・システムを使用しています。サーバーでコンポーネントが更新されるたびに、バージョン番号が 1 ずつ増分されます。コンポーネントがサーバーに追加された最初の時点のバージョン番号は 0 です。保存トランザクションのたびに番号が 1 ずつ増分することに注意してください。保存トランザクションは、保存対象のコンポーネントの 1 つ以上の変更を含むことができます。

「Save to Server」プロセスにおいて、サーバーと Oracle Virtual Directory Manager クライアントのバージョン番号の不一致が Oracle Virtual Directory Manager で検出されると、バージョン競合が発生します。Oracle Virtual Directory Manager がバージョン競合を検出すると、問題をユーザーに通知するウィンドウが表示されます。競合が発生するのは、項目のバージョン番号がサーバーとクライアントで異なる場合です。つまり、管理者が古い情報を使用している可能性があることを意味します。

この図では、アダプタ「LDAP Adapter 1」の編集バージョンは 0 ですが、サーバーのバージョンは 1 です。つまり、「LDAP Adapter 1」のローカル・バージョンが最新ではありません。他の管理者がサーバーに変更を保存してバージョン番号が自動的に増分した場合に、このような状況が発生します。この段階で管理者は、バージョン競合を無視するかコンポーネントを選択せずに変更を中断するかを選択できます。コンポーネントを保存しない場合、「Save To Server」プロセスが完了した後に「Reload From Server」を選択してコンポーネントをリフレッシュできます。

コンポーネントのリロード

変更内容によって異なりますが、再起動してアクティブにする必要がある項目がウィザードによって通知されます。

Oracle Virtual Directory の構成システムでは、ユーザーが編集していたときに、他のユーザーがサーバーに変更を行ったことを検出するバージョン・システムが実装されています。Save ウィザードがバージョンの不一致を検出すると、別の画面が表示され、そこでバージョンの違いを確認し、不一致を無視して処理する変更を決定できます。

「Finish」ボタンをクリックすると、Oracle Virtual Directory Manager はサーバーで変更を処理し、リクエストされたリロードを実行します。最後に確認ダイアログが表示され、変更が確認されます。または、問題があれば報告されます。

サーバー構成の編集

エディタが開いたら、画面の左上のサーバー・ナビゲータまたはエディタ・ウィンドウの下のセレクタ・タブを使用して、サーバー構成の様々な部分にナビゲートできます。

たとえば、「Listeners」タブを選択すると LDAP リスナー構成を確認できます。エディタでフィールドの内容を変更して [Tab] キーでフィールドの外に移動すると、エディタは変更を検知して「editor」タブのドキュメント名の横にアスタリスク（*）を表示します。これにより、ディスク（またはサーバー）に保存する必要がある変更がメモリーにあることが示されます。

ヒント： 一般的に、フィールドに対する編集がエディタで検知されるのは、ユーザーが [Tab] キーを使用してフィールドの外に出たとき、または別の Oracle Virtual Directory Manager コンポーネントを選択したときです。それまでは認識されません。

おなじく、サーバー・ナビゲータでも変更されたサーバー・コンポーネントにアスタリスク（*）が表示されます。サーバー・ナビゲータに表示されるアスタリスクは、サーバーに保存されていない Oracle Virtual Directory Manager での変更を示します。これは、編集内容がディスクに保存されていないことを示すためにアスタリスクを表示する編集ウィンドウとは異なります。

編集の競合

複数の管理者による編集の競合を解決するために、Oracle Virtual Directory は内部バージョン管理システムを実装しています。サーバーのコンポーネントが変更されるたびに、バージョン番号が 1 ずつ増分します。番号が変更されるのは、管理ゲートウェイ経由でサーバーが変更を受け入れたときです。サーバーは、最後に構成ファイルを読み取った後にディスク上の構成ファイルが変更されたことを検出すると（チェックサムを使用）、構成ファイルのバージョン番号も増分します。ディスク変更の検出は、システムの起動時またはサーバー・リロード・イベントで実行されます。

別の管理者がサーバー構成を変更した場合、Oracle Virtual Directory Manager クライアントを使用する管理者に対して、サーバー・ナビゲータに「<」記号が表示されます。この不等号（<）はローカル構成情報が最新ではない（ローカル・バージョン番号がサーバーよりも小さい）ことを示します。

「>」は、ローカル構成のバージョン番号がサーバーのバージョン番号よりも大きいことを示します。この状況は、サーバー間でドキュメントのカット・アンド・ペーストを行う場合、または他のユーザーがバックアップの構成をサーバーにリストアした場合に発生します。

「<」または「>」の表示によって、Oracle Virtual Directory Manager のローカル・ファイルとサーバーのバージョン競合が示されます。これらの記号の目的は、クライアント構成がサーバーと同期していないことをユーザーに知らせることです。このような状況が発生した場合は、サーバーの右クリック・メニューで「Reload From Server」を選択してローカル構成ファイルを更新できます。保存プロセスでバージョン競合を無視することもできます。

マッピング・スクリプト

Oracle Virtual Directory は、マッピング・スクリプトという新しいタイプのプラグインをサポートしています。マッピング・スクリプトは Python（別名 Jython）に基づいており、操作を管理者が行えるように設計されています。このため、データがサーバーを経由するときにマッピングすることができます。たとえば、マッピングを使用すると、スキーマの正規化（Active Directory を InetOrgPerson のように見せるなど）、データ型の添付（ハッシュ・パスワードに対する {sha} など）、またはデータストアから取得した属性値に基づく仮想属性の作成を行うことができます。

通常、マッピングはコンパイル済 Java コードとしてサーバーにデプロイされ、Oracle Virtual Directory 環境において Mapper と呼ばれる特殊なプラグインで実行されます。マッピング・コードは通常は開発者が作成しますが、Mapping ウィザードで提供されるグラフィック表示のマッピングを使用すると、コーディング経験のないユーザーもマッピング・コードを再利用、編集および構成できるようになります。新しいカスタム・マッピングの作成に関心がある開発者は、詳細を Oracle サポート・サービスに問い合せてください。

マッピングは、グローバルに実行するか、1 つ以上のアダプタのコンテキスト内で実行するように構成できます。複数のマッピングとアダプタを組み合せて、総合的な変換サービスを実行する個々の機能のセットを作成できます。

マッピングの定義

マッピングを定義するには、サーバー上またはその下を右クリックして「New Mapping」を選択します。new mapping ウィザードが表示され、事前定義済のテンプレートからマッピングを選択するか、空白のマッピングを作成できます。ファイル名を入力してマッピング・ソースを保存します。

「finish」をクリックすると、Oracle Virtual Directory Manager によってグラフィック・マッピング・エディタが起動されます。このエディタで、マッピングの編集、編集内容の保存、マッピングのサーバーへのデプロイを行うことができます。マッピングをデプロイした後、アダプタ内のサーバーまたはグローバル・サーバー・レベルでマッピングを実行するように構成できます。これには、サーバー・ナビゲータのマッピング・ドキュメントを右クリックして表示される「Deploy To Server」アクションを使用します。

マッピングのデプロイ

マッピングをデプロイする準備ができたら、マッピング・ファイルを右クリックし「Deploy To Server…」を選択します。この方法では、ソースの変換、コンパイルおよびパッケージ化が行われてから、マッピング・プラグインとしてサーバーにデプロイされます。マッピングがデプロイされてから、マッピングの実行について構成できます。

マッピングを実行するための構成

マッピングを実行するには、マッピングをデプロイするサーバーまたはアダプタの「Plug-ins」タブに移動します。「New Mapping」ボタンをクリックして、Mapping ウィザードを起動します。

最初のページでサーバーからデプロイ済マッピングを選択して（「Select From Server」ボタン）、一意の名前を付けることで、マッピングを定義します。

ヒント： 同じマッピングまたはプラグインは、インスタンスごとに別の名前を使用すれば何度もデプロイできます。

マッピングを定義したら、マッピングを適用するディレクトリ・ツリーを選択できます（オプション）。たとえば、ネームスペース ou=People,dc=MyOrg,dc=com を追加することで、マッピングをディレクトリ・ツリーの people ブランチのみに適用することができます。ネームスペースのフィルタ処理が必要でない場合は、空白にしておきます。

準備ができたら、「Finish」をクリックします。プラグイン構成画面の構成が更新されます。プラグイン構成画面では、マッピングが Mapper プラグインによって実行されるように定義され

ていることが示されます。マッピングの名前は Mapper プラグインの入力パラメータとして指定されます。

ディレクトリ・ブラウザ・ビュー

ブラウザ・ビューでは、ユーザーは Oracle Virtual Directory、LDAP ディレクトリまたはデータベースの内容を表示できます。このビューを使用して、Oracle Virtual Directory の変換の結果、元のソース・データ、または Oracle Virtual Directory にまだ接続していないデータ・ストアの内容を確認することができます。

Oracle Virtual Directory（仮想化および未変換）に接続する管理ブラウザと、Oracle Virtual Directory Manager クライアントから特定の LDAP サーバーに接続する LDAP ブラウザという 2 種類のブラウザがあります。

管理ブラウザ

管理ブラウザは、Oracle Virtual Directory に対する管理ゲートウェイで DSMLv2 を使用して行われたデータ・ブラウズを表示します。管理ブラウザには 4 つのタイプがあります。クライアント・ビュー・ブラウザ、アダプタ・ソース・データ・ブラウザ、ディレクトリ・ブラウザ、データベース・ブラウザです。

すべての管理ブラウザには、ブラウザのツリー・ウィンドウで Oracle Virtual Directory オブジェクトを介してアクセスできます。Oracle Virtual Directory オブジェクトは、サーバーがサーバー・ナビゲータで定義されると自動的に作成されます。

これらのそれぞれの管理ブラウザは、特定の Oracle Virtual Directory に関連付けられます。サーバーのアイコンの下に取得されるデータは、変換されたデータ（クライアント・ビュー）でも未変換データ（アダプタ・データ・ブラウザ、ディレクトリ・ブラウザまたはデータベース・ブラウザ）でも、常に指定の Oracle Virtual Directory から取得されます。

クライアント・ビュー・ブラウザは、Oracle Virtual Directory Manager によって自動的に構成され、完全なディレクトリ・ツリーを示します。これは、すべてのアダプタで定義されたもので、Oracle Virtual Directory によってすべてのマッピングと変換が適用された後の状態です。このビューは処理後のビューとみなされます。

アダプタ・ソース・データ・ブラウザは、定義されたアダプタのリモート・データ・ストアに存在しているようにデータを示します。返されるデータは、リモート LDAP サーバーに存在する未変換のディレクトリ情報です。つまり、データベースの場合は、実際のデータベースに存在する表とフィールドが返されます。データ・モデリングに役立つサンプル表の行も含まれます。

Oracle Virtual Directory そのものや定義済アダプタのいずれかで定義されていない任意のディレクトリ情報をブラウズするには、ディレクトリ・ブラウザおよびデータベース・ブラウザを使用して任意のディレクトリに接続できます。これによって、Oracle Virtual Directory で認識するように、ネットワーク・コンテキストでのリポジトリの生データを表示することができます。たとえば、エクストラネット・ネットワーク・ゾーンにある Oracle Virtual Directory は、管理者が自分のワークステーションから企業イントラネットのサーバーを見て参照できるデータと比較すると、制限されたデータしか参照できません。

LDAP のブラウズ

もう 1 つのタイプは LDAP ブラウザです。このタイプのブラウザはディレクトリ・サーバー・ブラウズ・セッションを表します。このセッションは、リクエストされたサーバーに Oracle Virtual Directory Manager クライアントから LDAPv3 で直接接続することによって確立されます。

ユーザーは、LDAP ブラウザで右クリックして「New Directory Browser」を選択することで、Oracle Virtual Directory Manager クライアントと直接接続する新しい LDAP ブラウザを作成できます。選択すると、次のウィザードが表示されます。

ダイアログに接続情報を入力します。特定のブラウザのベース識別名を入力するか、空白のままにして、Oracle Virtual Directory Manager で LDAP ベース・エントリを問い合せて使用可能なネーミング・コンテキストを判別することができます。

警告：入力したパスワードのハッシュが Oracle Virtual Directory Manager/Eclipse 内部のプリファレンス・データベースに格納されます。

データの表示と検索

いずれかのブラウザを使用すると、ツリーのノードをクリックしてエントリを表示したり、正符号（+）をクリックして次のレベルの情報を表示したりできます。選択したエントリは右下に表示されます。

ブラウズするかわりに、ディレクトリで特定のエントリを検索することもできます。検索を実行するには、検索対象のディレクトリ・サーバーまたはベース・エントリを右クリックするか、検索機能を選択します。検索が実行されると、ビューが更新されて、検出されたエントリが表示されます。

このとき、検索の下のツリーには結果のエントリのみが表示されます。ツリーをリセットしてすべてのコンテンツを表示するには、ブラウザ・ビューの該当するエントリを右クリックして、「Clear Search」を選択します。

新しいユーザー名または識別名を選択して接続に関連するユーザーを変更する必要が生じた場合は、「Rebind」オプションを選択して処理します。「Rebind…」を選択すると接続のデフォルト資格証明にリセットできます。

LDIF のインポートとエクスポート

LDIF は、LDAP サーバー間でデータを交換するために設計されたテキスト交換形式です。LDIF ファイルは、サーバーとの間でバッチ・データをインポートおよびエクスポートするときに最もよく使用されます。LDIF はスキーマ構成の変更をインポートするときにも使用できます。

Oracle Virtual Directory Manager では、Oracle Virtual Directory のクライアント・ビューまたは LDAP ブラウザのディレクトリに対して、データを LDIF 形式でインポートまたはエクスポートできます。

LDIF 情報をインポートするときは、インポートするファイルを選択するように求められます。その後、Oracle Virtual Directory Manager が指定のサーバーへのデータのロードを試行します。

ヒント： Oracle Virtual Directory Manager の「Import LDIF」プロセスでは、開始の時点ですべての LDIF インポート・ファイルのバージョン番号が有効であることが必要です。ファイルにバージョン番号がない場合は、ファイルの最初に「Version: 1」を追加します。

LDIF 情報をエクスポートするとき、エクスポート・プロセスは、現在選択されているノードを使用して、デフォルトでエクスポートするディレクトリ・ツリーの部分を判別します。

エクスポートのダイアログでは検索のダイアログと同じく、エクスポートのための検索フィルタや有効範囲を指定できます。このダイアログは、デフォルトでは、現在のノードのサブツリーをすべての属性と一緒にエクスポートするように設定されています。

このエクスポートで生成される出力ファイルは、UTF-8 でエンコードされたテキスト・ファイルです。Oracle Virtual Directory Manager のテキスト・エディタまたは Oracle Virtual Directory Manager 外部の任意のテキスト・エディタを使用して編集できます。

ヒント： ディレクトリ全体のエクスポートは大容量になる可能性があり、Oracle Virtual Directory またはプロキシ・ソース・ディレクトリに設定された検索制限の影響を受けることがあります。大容量のディレクトリ情報のエクスポートが目的の場合は、Oracle Virtual Directory ルート・アカウント（cn=admin など）を使用していることと、Oracle Virtual Directory がリモート・ディレクトリに対して使用するプロキシ接続権限が、すべての結果を返すために十分であることを確認してください。

リフレッシュ

サーバーに対して変更が行われた場合はいつでもブラウザ表示をリフレッシュできます。リフレッシュするオブジェクトのノードを右クリックし、「Refresh Current Node」を選択します。

Oracle Virtual Directory を Microsoft Active Directory または IBM Tivoli Access Manager と統合するための「One-Step Configurations」機能の使用

Oracle Virtual Directory には、「One-Step Configurations」機能が含まれます。これによって、Oracle Virtual Directory と Microsoft Active Directory および IBM Tivoli Access Manager の統合などの一般的な統合タスクのプロセスが単純化されます。Oracle Virtual Directory Manager の「One-Step Configurations」機能で次の手順を使用して、Oracle Virtual Directory サーバーを Microsoft Active Directory または IBM Tivoli Access Manager に統合します。

Oracle Virtual Directory と Microsoft Active Directory の統合

Oracle Virtual Directory の一般的なデプロイの 1 つに、Microsoft Active Directory に対するプロキシとファイアウォールとしてのデプロイがあります。以前は、Oracle Virtual Directory のすべての設定を構成するには、様々な構成画面を使用する複数の手順が必要でした（特にファイアウォールの場合）。Oracle Virtual Directory 10*g*（10.1.4.0.1）の「One-Step Configurations」機能では、このような構成を 1 つのウィザードのみを使用して容易に設定できます。

次の手順を実行して、Oracle Virtual Directory をプロキシおよびファイアウォールとして Microsoft Active Directory に統合します。

1. Microsoft Active Directory と統合する Oracle Virtual Directory サーバー・インスタンスに接続します。
2. Microsoft Active Directory と統合する Oracle Virtual Directory サーバー・インスタンスの名前を右クリックし、「One-Step Configurations」を選択し、「Active Directory」を選択して、さらに「Configure Proxy/Firewall」を選択します。Microsoft Active Directory Proxy/Firewall ウィザードが表示されます。
3. プロキシを構成する Oracle Virtual Directory サーバー・インスタンスを選択し、設定を選択して、「Next」をクリックします。
4. Active Directory サーバーで使用するプロキシの資格証明と資格証明処理のオプションを入力し、「Next」をクリックします。
5. Active Directory サーバーのルート DNS ドメイン名、たとえば oracle.com を入力します。「Refresh/Lookup with DNS」をクリックするか、「Add DC」をクリックして Active Directory サーバーを手動で追加します。
6. 「Verify Directory Servers」をクリックして、設定が正しいことを確認します。
7. これまでの画面で Active Directory サーバーに対する SSL をサポートするオプションを選択した場合は、前の手順を繰り返して SSL インタフェースを構成し、「Next」をクリックします。
8. 使用するネームスペース・マッピングを入力して、「Next」をクリックします。
9. これまでの画面で「Denial of Service (DoS)」オプションを選択した場合は、DoS オプションを構成して、「Next」をクリックします。
10. 「Next」をクリックすると、Microsoft Active Directory Proxy/Firewall ウィザードによって、Oracle Virtual Directory Manager 構成システムに設定が構成されます。
11. 「Finish」をクリックして、設定を Oracle Virtual Directory サーバーに保存します。

Oracle Virtual Directory と IBM Tivoli Access Manager との統合

Oracle には独自のアクセス管理ソリューションとして Oracle Access Manager がありますが、Oracle Virtual Directory 10*g*（10.1.4.0.1）にも「One-Step Configurations」機能オプションがあり、IBM のアクセス管理ソリューションである Tivoli Access Manager（TAM）との統合が単純化されます。

次の手順を実行して、Oracle Virtual Directory を Tivoli Access Manager に統合します。

1. IBM Tivoli Access Manager と統合する Oracle Virtual Directory サーバー・インスタンスに接続します。
2. IBM Tivoli Access Manager と統合する Oracle Virtual Directory サーバー・インスタンスの名前を右クリックし、「One-Step Configurations」を選択し、「TAM (Tivoli Access Manager)」を選択して、さらに「Prepare VDE for TAM」を選択します。Prepare VDE for TAM Service ウィザードが表示されます。
3. TAM セキュリティと構成データを格納する Oracle Virtual Directory サーバー・アダプタをプルダウン・メニューから選択し、「Next」をクリックします。
4. プライマリ・ディレクトリ・タイプとして次のいずれかのオプションを選択し、「Next」をクリックします。
   - IBM Directory Server
   - Sun One Directory
   - Novell eDirectory
5. 「Preparing Primary Directory Server」画面の「Update Schema」オプションを有効または無効に設定して、「Next」をクリックします。「TAM Primary Directory Connection Information」画面が表示されます。
6. 各フィールドに次の情報を入力し、「Next」をクリックします。
   - LDAP マスター・サーバーの名前または IP アドレス
   - LDAP マスター・サーバーのポート番号
   - SSL の有効化または無効化
   - ブラウズ DN の名前
   - ブラウズ DN のパスワード
7. LDAP マスター・サーバーがクラスタにある場合は、次の追加 LDAP サーバー情報を「Additional LDAP Servers」画面に入力し、「Next」をクリックします。
   - LDAP サーバーが検証されているかどうか
   - LDAP サーバーのホスト名
   - LDAP サーバーのポート番号
   - 負荷のパーセンテージ
   - 読取り、書込みまたは実行の権限
   - 「Verifying Directory Servers」オプションの有効化または無効化
8. 「Next」をクリックすると、Prepare VDE for TAM Service ウィザードによって、Oracle Virtual Directory Manager 構成システムに設定が構成されます。
9. 「Finish」をクリックして、設定を Oracle Virtual Directory サーバーに保存します。

統合された Tivoli Access Manager 構成への新しいディレクトリの追加

「One-Step Configurations」機能の「Prepare VDE for TAM」オプションを使用して Oracle Virtual Directory を Tivoli Access Manager に統合した後で、次の手順を実行して新しい LDAP アダプタを構成に追加することができます。

1. IBM Tivoli Access Manager と統合した Oracle Virtual Directory の名前を右クリックし、「One-Step Configurations」を選択し、「TAM (Tivoli Access Manager)」を選択して、さらに「Add New User Directory to Proxy」を選択します。
2. 追加するディレクトリの情報を入力して、「OK」をクリックします。

5

Oracle Virtual Directory Manager のユーティリティ

この章では、Oracle Virtual Directory Manager に含まれる様々なサーバー管理ユーティリティについて説明します。すべてのユーティリティでは、管理対象のサーバーとのオンライン接続が必要です。次のユーティリティについて説明します。

- サーバー・ライブラリ・マネージャ : サーバーにインストールされた様々な JDBC、プラグインおよびジョイナ・クラスを管理するユーティリティ。
- サーバー・キー・マネージャ : サーバーの PKI キー・ライブラリを管理するユーティリティ。
- サーバー・ブラウザ : ソース・データと仮想データの両方をブラウズできるビュー。このブラウザでは、LDIF データのインポートとエクスポートなど基本的なデータ管理機能も提供されます。

サーバー・ライブラリの管理

Oracle Virtual Directory では 3 種類のライブラリ・ファイルをサポートしています。ライブラリ・マネージャを使用し、接続した Oracle Virtual Directory に対してこれらのライブラリ・クラスのデプロイや管理を行うことができます。

ライブラリ・マネージャの優れた機能は、サーバーのパスを調整せずにほとんどのライブラリをデプロイできることです。このため、ほとんどの場合、新しいライブラリをデプロイするときにサーバーの再起動は必要ありません。

ライブラリ・マネージャで管理するライブラリは大きく分けると次の 3 種類です。

- **Oracle Virtual Directory プラグイン:** Oracle Virtual Directory プラグインとして使用するためのすべての Java ライブラリ。このユーティリティを使用して、新しいプラグインを構築し、サーバーでのプラグインのデプロイ、取得および更新を行います。
- **Oracle Virtual Directory ジョイナ:** Oracle Virtual Directory ジョイナとして使用するためのすべての Java ライブラリ。このユーティリティを使用して、新しいジョイナを構築し、サーバーでのジョイナのデプロイ、取得および更新を行います。
- **ライブラリ :** Oracle Virtual Directory のプラグインやジョイナで必要なその他のライブラリ、または DB アダプタ用の JDBC ライブラリをインストールできます。

このようなライブラリの種類は、サーバー・ライブラリ・マネージャのタブに表示される機能と対応しています。

サーバー・ライブラリ・マネージャを起動するには、関連するサーバーを右クリックして「Manage Server Libraries…」を選択してください。

Library Management

このタブを使用して、Oracle Virtual Directory で必要なあらゆるタイプのライブラリ（Oracle Virtual Directory プラグインおよびジョイナ以外）のインストールを管理します。ほとんどの場合、このタブは JDBC をサポートするために必要なライブラリのインストールに使用されます。

画面の左側のデプロイ済ライブラリのリスト・ボックスに、サーバーにインストールされているジョイナ、プラグインおよび JDBC ライブラリを含むすべてのライブラリが表示されます。Oracle Virtual Directory プラグインまたはジョイナの詳細は、ウィンドウ上部の対応するタブをクリックしてください。

このツールを使用して通常のライブラリ（JDBC など）をインストールするには、「Select New…」ボタンを使用してインストールするライブラリを選択します。JDBC の場合は、サプライ・ベンダーで推奨されるすべてのライブラリ・ファイルを選択します。選択を行うにつれ、選択したファイルのリストが「Files To Be Deployed…」ウィンドウに表示されます。すべてのライブラリ・ファイルを選択したら、「Deploy」ボタンをクリックしてファイルをデプロイします。

デプロイするファイルのリストからファイルを削除するには、「Remove」ボタンを使用します。

ヒント :「Manage Sever Libraries」では、サーバーからライブラリを削除することはできません。サーバーからライブラリを削除するには、サーバーを停止して Oracle Virtual Directory インストールの plugins/lib ディレクトリからライブラリを削除し、サーバーを再起動します。

プラグインの管理

「Plug-in」タブでは、インストールされている特定の Oracle Virtual Directory プラグインの詳細情報を確認できます。「Plug-in」タブを使用すると、Java クラス・ファイル、Java プロジェクトまたは既存のプラグイン・ファイルから新しいプラグインをデプロイすることができます。

プラグイン・インベントリ

メイン画面に、Oracle Virtual Directory にインストールされているプラグインのインベントリが表示されます。特定のプラグインの詳細を確認するには、「Deployed Plug-ins」リスト・ボックスに表示されているプラグインをクリックします。こうすると、画面が更新され、右側にプラグイン・マニフェスト情報が表示されます。プラグイン・マニフェストによって、Oracle Virtual Directory は、適切なメソッドを含む Java クラスの名前やプラグインで受け入れられるパラメータを認識できます。説明やバージョン番号などの管理情報も提供されます。

ヒント： Oracle Virtual Directory プラグインの詳細は、マッピング、Java プラグインおよびプラグインの管理に関する各章を参照してください。

サーバーからのプラグインの取得

場合によっては、デプロイ済プラグインを別のサーバーにデプロイするために取り出す必要があります。これには、取得するプラグインを選択して、「Retrieve from Server…」ボタンをクリックします。デプロイ先のディレクトリを指定するように求められます。デフォルトでは、プラグインは Oracle Virtual Directory Manager ワークスペースにあるサーバーのローカルの plugins/lib ディレクトリに保存されます。

プラグインのデプロイ

「Package/Deploy/Update…」ボタンをクリックすると、Plug-in Deployment ウィザードが開始します。このウィザードでは、新しいプラグインの作成、既存のプラグイン・ファイル（ローカルに格納）の編集、またはサーバーに現在格納されているプラグインの編集を行うことができます。

- 一連の既存クラス・ファイルがある場合は、それらが Oracle Virtual Directory Manager 外部に格納されていても、Oracle Virtual Directory Manager Java プロジェクト内に格納されていても、「Create & Deploy」を使用します。
- 外部ソースからプラグインを取得した場合、または別のサーバーからプラグインを取得したばかりの場合は、「Edit & Deploy」を使用します。
- サーバーにすでにインストールされているプラグインを更新する場合は、「Revise & Re-deploy」を使用します。

Oracle Virtual Directory Manager Java プロジェクトを使用したプラグインの構築

既存の Oracle Virtual Directory Manager Java プロジェクトからファイルを選択することを指定すると、Plug-in Deployment ウィザードの「Choose Plugin Source Project」ページが表示されます。このページには、Oracle Virtual Directory Manager で使用可能な、Java コードが関連付けられたプロジェクトが示されます。フォルダを開き、プラグインのプラグイン・クラスが見つかるまでパッケージ名も開いて、プロジェクトを選択します。選択すると、「Files」リスト・ボックスに、プラグインに組み込む予定のファイルのリストが表示されます。このリストには、プロジェクトに含まれるすべてのファイルの他に、オプションで Java ソース・ファイルが含まれます（上のセレクタを参照）。

プラグインのファイルの編集

既存のプラグインの編集、または特定のファイル（たとえば、プロジェクト以外）を使用した新規プラグインの作成を選択した場合、Plug-in Deployment ウィザードの「Edit Plugin Description」ページが表示されます。

「Plug-in Manifest Editor」ページ

プラグインを作成するために選択した方法に関係なく、最後の手順ではプラグインのマニフェストを準備します。マニフェストによって、Oracle Virtual Directory が、プラグインの名前、処理できる内容、使用できるパラメータを認識します。

バージョン番号のフィールドは、サーバーが参照するリビジョン番号を反映するため編集できません。サーバー上で誰かがプラグインを更新すると、バージョン番号は自動的に増分されます。

「Finish」をクリックすると、ウィザードによってプラグインのアセンブリが完了され、プラグインがサーバーにデプロイされます。

ジョイナの管理

「Joiner」タブの機能は「Plug-in management」タブと同じです。「Joiner」タブでは、サーバーにインストールされている Oracle Virtual Directory ジョイナの詳細情報を確認できます。「Joiner」タブを使用すると、Java クラス・ファイル、Java プロジェクトまたは既存のジョイナ・パッケージ・ファイル（JAR）から新しいジョイナをデプロイすることができます。

ジョイナのインベントリ

「Joiner」タブに、Oracle Virtual Directory にインストールされているジョイナのインベントリが表示されます。特定のジョイナの詳細を確認するには、「Deployed Joiners」リスト・ボックスに表示されているジョイナをクリックします。こうすると、画面が更新されて、右側にジョイナのマニフェスト情報が表示されます。ジョイナ・マニフェストによって、Oracle Virtual Directory は、適切なメソッドを含む Java クラスの名前を認識できます。説明やバージョン番号などの管理情報も提供されます。

ジョイナの管理

ジョイナをパッケージ化してデプロイするには、前の項でプラグインについて説明したのと同じ手順を実行します。

サーバー・キーの管理

Oracle Virtual Directory では、SSL/TLS をサポートするために使用される鍵の格納に Java キー・ストアを使用します。通常、キー・ストアは Oracle Virtual Directory インストールの conf ディレクトリにあります。キー・ストアで鍵を管理するには、管理するサーバーを右クリックして「Manage Server Keys…」を選択します。「Key Manager」ダイアログ・ボックスが表示されます。

SSL の用語

Oracle Virtual Directory 環境で使用される SSL の用語を説明するために、次に簡単な用語集を示します。

- **キー・ストア** : 様々なサーバー証明書や信頼できる X.509 証明書を含むサーバー上のデータベース。
- **サーバー・キー** : サーバー・キーは、秘密鍵と公開鍵の両方を含む鍵です。Oracle Virtual Directory が SSL/TLS サービスをクライアントに提供するために使用します。Oracle Virtual Directory は、必要に応じてこのような鍵を 0 個以上利用できます。複数の鍵を使用すると、サーバー上の異なるリスナーに別の鍵を割り当てることができます。これは、複数のネットワーク・インタフェース・カードがあり、サーバーが応答する必要がある DNS 名が複数ある場合に特に役立ちます。各リスナーの構成には、特定のリスナーに対して使用される鍵の別名を指定するフィールドがあります。

  Oracle Virtual Directory が SSL/TLS を使用して他の LDAP ソースとプロキシ接続を設定するとき、クライアント証明書の認証がプロキシ・サーバーで必要な場合にはサーバー・キーの使用が必要になります。選択した鍵はサーバーの構成に指定されます。Oracle

Virtual Directory はすべてのプロキシ接続で 1 つの鍵しか使用できないことに注意してください。アダプタ固有の鍵の選択は現時点ではサポートされていません。

- **信頼できる証明書** : リモート・ソースとの接続を確立するとき、Oracle Virtual Directory Manager クライアントは、現在の Oracle Virtual Directory がリモート・サーバーの証明書を信頼しているかどうかを自動的にテストします。テストは、Oracle Virtual Directory キー・ストアにある信頼できる証明書をリモート・ホストの証明書と比較して実行されます。信頼が確立されるのは、有効な証明書が特定の信頼できるホスト証明書と一致する場合、またはリモート・ホストの証明書に署名した親認証局の 1 つに対して Oracle Virtual Directory キー・ストアに信頼できる鍵がある場合です。

  Oracle Virtual Directory がセキュアなリモート・ホストとの接続をテストした後でホストを信頼しないことを決定した場合は、ダイアログが表示され、管理者が証明書を信頼するかどうかをそこで決定できます。管理者が証明書を信頼することを選択すると、その証明書がキー・ストアに追加され、「Key Manager」で表示できるようになります。

- **別名 :** キー・ストアに格納される各証明書にはローカル名があります。この名前は、キー・ストア・データベースから特定の鍵を選択するために様々な Oracle Virtual Directory Manager 構成画面で使用されます。

- **証明書 :** 公開鍵証明書（形式が定められたデジタル情報）。サブジェクトやその他の管理情報を指定します。公開鍵には、証明書の情報を認証する認証局がデジタル署名を付けています。

- **認証局** : 認証局（CA）は鍵に署名をして保証するサービスです。具体的には、CA の公開鍵が信頼できる鍵として Oracle Virtual Directory キー・ストアにインストールされると、Oracle Virtual Directory はその CA で生成されたすべての鍵を暗黙的に信頼します（それらの鍵が期限切れでないかぎり）。

- **証明連鎖 :** サブジェクトの証明書とそのサブジェクトの証明書に署名した認証局の公開鍵を含む一連の証明書。さらに、ルートにいたるまでの親認証局の証明書も含まれます。Oracle Virtual Directory Manager が信頼を確立すると、証明書のサブジェクトまたは証明連鎖に格納される任意の認証局を信頼するかどうかをユーザーが決定できます。

- **証明書リクエスト :** これは、Oracle Virtual Directory からエクスポートされて署名のために CA に送られる特殊なバージョンのサーバー・キーです。Oracle Virtual Directory のサーバー・キーに CA が署名することにより、その CA を信頼する他のサーバーが暗黙的に Oracle Virtual Directory を信頼します。たとえば、Microsoft CA サービスが Oracle Virtual Directory の鍵に署名すると、Windows アプリケーションが Oracle Virtual Directory を信頼するようになります。

- **自己署名鍵 :** これは、サーバーが自ら署名したサーバー・キーです。保証する機関の署名は証明書に付きません。このような証明書を使用するとき、各クライアントは自己署名証明書を信頼するかどうかを明示的に決定する必要があります。これらの鍵は、外部の保証機関が証明書の正当性を確認するために署名していないため、それほどセキュアではないと考えられます。

サーバー・キーの生成

前に説明したように、1 つ以上のサーバー・キーを生成することができます。Oracle Virtual Directory では、自己署名鍵の生成または認証局が署名する鍵の生成がサポートされます。

自己署名鍵

自己署名鍵の生成は比較的単純なプロセスです。Key Manager ウィンドウの「Generate」をクリックしてプロセスを開始します。表示されるダイアログで、特定の証明書の情報を入力し、ダイアログの下部で「self-sign」オプションを選択する必要があります。

「Country」フィールドには 2 文字の略称を指定し、「State」フィールドには（略称ではなく）完全な名称を指定する必要があることに注意してください。「Common Name」フィールドには、通常、クライアントがサーバーに接続するために使用する DNS 名を指定します。これは、クライアントがサーバーへのアクセスを指定する方法と正確に一致する必要があります。ここでは IP アドレスも有効です。共通名がクライアントのホスト・アドレス指定と一致しない場合、クライアントがサーバーの証明書の受け入れを拒否することがあります。

すべてのフィールドに入力したら「Finish」をクリックし、サーバーで鍵を作成して自己署名します。すぐに「key manager」ダイアログに戻りますが、このダイアログには証明書が表示されているはずです。証明書を使用するには、適切なリスナーに別名を入力し、そのリスナーで SSL/TLS を有効にします。

署名付きサーバー・キー

署名付きサーバー・キーの設定は、自己署名鍵の生成よりも少し複雑です。プロセスは自己署名鍵の場合と似ていますが、その他の手順が必要になります。

自己署名のプロセスと同様に、「Generate Key」をクリックしてサーバー証明書を生成します。同様に必要なフィールドを指定しますが、「self-signed」は選択しないでください。「Finish」をクリックすると、鍵が「Key Manager」に表示されます。

「Request Certificate」を選択して、認証局で処理するために証明書をテキスト・ファイルとしてエクスポートします。証明書は PEM すなわち Base-64 エンコード形式でエクスポートされます。

認証局の指示に従って、エクスポートした証明書ファイルと一緒に証明書リクエストを提出し、署名を受けます。

署名された証明書が返されたら、「Import」ボタンを使用して証明書をインポートします。Base-64（PEM）エンコードの証明書の証明書ファイル名と、インポート時に使用する別名を指定するように求められます。

「Finish」をクリックしてから、いずれかのリスナーで SSL を有効化し、使用する証明書として証明書別名を指定して証明書をアクティブ化します。これで設定は完了です。

ヒント： 署名付き証明書をインポートするときの別名は、署名付き証明書リクエストの生成に使用した別名と一致する必要があります。

信頼できる証明書のインポート

「Import」ボタンを使用して、Base-64（PEM）エンコード証明書をインポートできます。信頼できる証明書（認証局の公開鍵など）をインポートするには、「Import」をクリックして、「certificate import」ダイアログを開きます。以前の署名リクエストから証明書をインポートする場合以外は、キー・ストアに存在しない新しい別名を入力します。「Finish」をクリックすると、鍵が信頼できる鍵としてサーバーにインストールされます。

エクスポート

「key manager」には、証明書を Base-64 PEM（テキスト）形式でエクスポートする機能があります。これは、クライアント・アプリケーションが Oracle Virtual Directory を信頼するように Oracle Virtual Directory の公開鍵をエクスポートする場合によく使用します。このアクションを使用するには、エクスポートする鍵を選択して、「Export」をクリックします。必要なファイル名を入力し、「Save」をクリックしてエクスポートを完了します。

Oracle Virtual Directory Manager ブラウザ

Oracle Virtual Directory Manager ブラウザは、仮想ディレクトリ・データとその他の外部データ（LDAP または RDBMS の格納データなど）の両方をブラウズできる多目的データ・ブラウザです。このブラウザを使用すると、仮想ディレクトリの構成前と構成後の状態を確認することができます。

Oracle Virtual Directory Manager ブラウザでは、2 種類のブラウズ・モードがサポートされています。LDAP ディレクトリは、Oracle Virtual Directory Manager とのローカル接続を使用してツリーの LDAP ブラウザ・オブジェクトを介してブラウズできます。または、プロキシとして作動する特定のサーバーが見るようにデータを表示することもできます。LDAP ブラウザを介してブラウズするとき、Oracle Virtual Directory Manager は LDAP サーバーに直接接続します。この場合、Oracle Virtual Directory を介したブラウズでは、Oracle Virtual Directory を使用してリモート LDAP または RDBMS ソースへの LDAP 問合せを実行します。Oracle Virtual Directory を介して問合せを行うことで、Oracle Virtual Directory がネットワークのコンテキストでどのようにデータを見ているかを確認できます。

Oracle Virtual Directory を介してブラウズするときは、項目の問合せを次のいずれかから行うことができます。

- **クライアント・ビュー :** エンドクライアントから見える仮想ディレクトリ・ビュー。

  注意 : クライアント・ビューにはアクセス制御の効果は表示されません。管理者のコンテキストを使用しているためです。実際のエンド・ユーザー・ビューを見るには、LDAP ブラウザを使用し、特定のユーザー資格証明を使用して Oracle Virtual Directory をブラウズします。

- **アダプタ・ビュー :** LDAP または DB アダプタのソース・データを表示できます。
- **ディレクトリ・ブラウザ・ビュー :** ゲートウェイとして Oracle Virtual Directory を使用してリモート LDAPv3 サーバーをブラウズできます。アダプタを設定する必要はありません。
- **データベース・ブラウザ・ビュー :** サーバーの JDBC ライブラリを使用してリモート・データベースをブラウズできます。アダプタを設定する必要はありません。

検索

ブラウザでは、ディレクトリおよびデータベース・データを調べるためのいくつかの機能が提供されます。データを調べる方法は 2 つあります。つまり、ツリーの様々な部分を開いてデータを調べるか、検索機能を使用してディレクトリの検索を実行します。

検索機能を使用するには、検索対象のノードを右クリックします。ブラウザによって、検索を処理するホストが自動的に決定され、検索のためのダイアログが表示されます。

このダイアログでも、LDAP の検索機能と同じく、検索ベース（ディレクトリ内の問合せの対象部分）、検索有効範囲（ベースのみ、ベースの 1 レベル下まで、またはベース以下のツリー全体）、LDAP フィルタ、返す属性を指定できます。ベース、フィルタおよび属性のフィールドはコンボ・ボックスになっていることに注意してください。ドロップダウン・ボタン（下矢印）をクリックすると、フィールドに以前の問合せのフィールド・エントリが表示され、以前のフィールド値を再利用できます。

結果は検索ポイントの下のツリーに表示されます。前回の検索の結果を消去つまりリセットする場合は、該当するノードを右クリックし、「Clear Search」をクリックしてツリーをリセットします。

インポートとエクスポート

「Import and Export LDIF」を使用すると、Oracle Virtual Directory または選択した LDAP サーバーに対して、LDIF のエクスポートとインポートを実行できます。インポート機能では、有効な LDIF ファイルが選択したサーバーにインポートされます。有効な LDIF ファイルには、ファイルの先頭に「version: 1」が必要です。

エクスポート機能では、ディレクトリのすべてまたは一部をテキスト・ファイルにエクスポートできます。エクスポート機能を選択すると、検索の場合と似たダイアログが表示され、エクスポートするエントリを選択できます。デフォルトでは、選択したディレクトリ・エントリの下のすべてのエントリがエクスポートされます。

リバインド

ブラウザの LDAP ブラウザの部分を使用している場合、サーバーの問合せに使用している資格証明の変更を選択できます。新規ユーザーとして最認証するには「Rebind」を選択します。

6

構成および設定

この章では、Oracle Virtual Directory の主要な構成コンポーネントについて説明します。この章では特に、全体的なサーバー構成、リスナー構成、Oracle Virtual Directory プラットフォームで使用できる多様なアダプタ、およびレプリケーション構成について説明します。

Oracle Virtual Directory の構成の概要

Oracle Virtual Directory の構成は複数のファイルに分かれており、それぞれのファイルは全体的なサーバー構成の異なる部分に対応しています。Oracle Virtual Directory Manager クライアントのサーバー・ナビゲータ・ビューでは、主要なコンポーネント・ファイルが独自の色で表示されます。

- **サーバー構成**（青緑）は、サーバー上の conf/server.os_xml です。
- **リスナー構成**（黄）は、サーバー上の conf/listeners.os_xml です。
- **アダプタ構成**（青）は、サーバー上の conf/adapters.os_xml です。アダプタ構成は、ローカル・ストア、LDAP プロキシ、DB プロキシ、NTLM プロキシ、結合ビューの 1 つ以上のアダプタをサポートします。
- **アクセス制御構成**（赤）は、サーバー上の conf/acls.os_xml です。
- **スキーマ構成**（緑）ファイルは、サーバーの conf ディレクトリにあり、ファイル名の形式は schema.core.xml または schema.user.xml です。
- **プラグイン構成**（青緑および青）は、サーバー構成（グローバル・プラグイン）およびアダプタ構成ファイル（アダプタ・プラグイン）の両方に格納されています。

サーバー構成

Oracle Virtual Directory のマスター構成は、Oracle Virtual Directory インストールの conf ディレクトリにある server.os_xml という XML ファイル内に含まれています。このファイルには、次のようなグローバル・サーバー構成のほとんどの情報が含まれています。

- 他の構成ファイルの名前（例 : adapters.os_xml）
- 一般情報およびライセンス情報
- グローバル・プラグイン構成
- ロギング構成
- システム全体の割当て
- DoS 設定
- ビュー定義
- 管理サービス構成

一般情報およびライセンス [Engine/Server/Info]

これらのパラメータのほとんどは、インストール処理時に自動的に値が指定されるか、デフォルト値が入力されます。

Name

このオプションは、構成ファイルで定義されるサーバー・インスタンスの説明的な名前に設定する必要があります。インターネット・ホスト名にする必要はありません。主に、レプリケーションの他のサーバーとの通信時に、固有のサーバー識別子として使用されます。この名前がサーバー・ナビゲータで使用されるわけではありませんが、同じ名前になる場合もあります。

Product Type

製品のタイプです。通常は値 VDE が含まれます。古い構成では、DSE または DFE が含まれる場合があります。（読取り専用）

Product Version

製品のインストール済のバージョンです。（読取り専用）

Product Kit
製品のビルド番号またはパッチ・レベルです。（読取り専用）

License Company
Oracle Virtual Directory の使用ライセンスを持つ組織です。ライセンスのアップグレードまたは更新時に変更される可能性があります。

License Product
使用ライセンスを持つコンポーネントの名前です（例 : VDE-Suite）。

License Expires
ライセンス・キーの有効期限です。有効期限がない場合は「never」になります。

License Key
Company/Product/Expires の値に対応する、コード化されたライセンス・キーです。

サーバー設定 [Engine/Server/Settings]

Schema Files
Oracle Virtual Directory でスキーマの定義に使用されるファイルのリストです。各ファイルは順番に適用され、競合が発生した場合には、後のファイルが優先されます。通常は、最後のファイルが schema.user.xml になります。（LDAP または Oracle Virtual Directory Manager GUI を介した）スキーマへのすべての変更は、必ずこのファイルに適用されます。これにより、schema.core.xml などの標準ファイルを変更せずにリリース間で継承しながら、schema.core.xml 内の出荷時のスキーマよりも schema.user.xml への変更を優先することで管理者が仮想的に変更を行えます。

デフォルト : conf/schema.core.xml、conf/schema.cosine.xml、conf/schema.inetorgperson.xml、conf/schema.nis.xml、conf/schema.dyngroup.xml、conf/schema.java.xml、conf/schema.user.xml

---

ヒント : 製造時に組み込まれたスキーマ（DSML 形式）をインストールする場合は、スキーマ・ファイル・リストの最後から 2 番目にこのファイルを挿入してください。これにより、製造時に配布されたファイルが変更されないように保護すると同時に、ローカルでのカスタマイズ（後で schema.user.xml に格納されます）が可能になります。

Schema Checking
選択すると（true）、LDAP エントリがチェックされ、このサーバーによって認識されるスキーマ定義に対する適合性が確認されます。スキーマのチェックは、このオプションを選択解除（false に設定）すると無効になりますが、外部のスキーマ・チェック機能を使用する場合以外は、無効にしないでください。
デフォルト : 有効

Access Control File
アクセス制御構成情報が格納されているファイルです。サーバーのインストール・ディレクトリに対して相対的に指定されます。通常、このファイルは Oracle Virtual Directory Manager の ACL エディタによって管理されます。ただし、LDAP コールを介して変更することも可能です。
デフォルト : conf/acls.os_xml

Access Control Checking
アクセス制御ファイルの定義に従って Oracle Virtual Directory でアクセス制御が実行されるかどうかを決定します。
デフォルト : 有効

Replication File
レプリケーション構成データを保持するファイルです（Oracle Virtual Directory のインストール・ディレクトリに対して相対的に指定されます）。
デフォルト : 空白

Replication Interval
マスター・サーバーによるレプリカへの新規変更の送信試行間隔を秒単位で表します。

Root DN（非表示）
この設定により、管理者は、Oracle Virtual Directory のルート DSE エントリ（base=""）の場所を仮想ディレクトリ・ツリー内の別の場所に変更できます。

ヒント : このような場所変更が最も行われるのは、Oracle Virtual Directory のルート・エントリを再配置するのに、別のサーバーのルート・エントリにプロキシを設定する必要がある場合です。Oracle Virtual Directory を別のディレクトリ・サーバーとして使用する場合にも、よく行われます。これは、通信先ディレクトリ製品をアプリケーションが推測する場合に便利です。Oracle Virtual Directory のルート・エントリが "" 以外の名前に変更されていれば、リモート・ベースが "" の LDAP アダプタを作成し、ローカル・ルートを "" に設定して、そのルート・エントリを置換することができます。これを行う際には、LDAP アダプタのルーティング構成で「Routing Levels」を 0 に設定することをお薦めします。この設定を行うと、ルートの問合せ時には、リモート・サーバーのルート・エントリのみの問合せが Oracle Virtual Directory によって試行されます。この設定を怠ると、Oracle Virtual Directory が受信したすべてのリクエストに対する問合せをリモート・サーバーが受信することになります。

サーバー・ロギング [Engine/Server/Logging]

Oracle Virtual Directory Manager エディタでは、標準の Oracle Virtual Directory ロギング値の変更がサポートされています。ただし、Oracle Virtual Directory ロギング・システムは Log4J に基づいており、追加の詳細設定（カスタム・フォーマットまたはデータベースなどの他のシステムへのロギングなど）を使用できます。詳細は、オラクル社カスタマ・サポート・センターに問い合せてください。

通常、Oracle Virtual Directory では、Oracle Virtual Directory General Log と Access Log の 2 つのログ・ファイルが提供されます。Oracle Virtual Directory General Log（コンソール・ログ）では、Oracle Virtual Directory の操作上のエラーおよびレポートが提供されます。このログには、接続の失敗およびその他の管理イベントに関する情報が保存されます。Access Log は、Oracle Virtual Directory システムで発生したトランザクションをレポートするトランザクション・ログです。

デフォルトでは、ログ・ファイルのローテーションが毎日午前 0 時に行われます。現在のログ・ファイルは vde.log という形で表され、以前のログ・ファイルには vde.log.2004-09.21 のようにデータが付加されます。この動作を変更するには、Log4J 構成ファイル（次項目を参照）を使用してカスタム構成を設定します。

Use Log4J Configuration File
選択すると、Oracle Virtual Directory によってプロパティ・ファイルからカスタム Log4J 構成が読み取られます。

Log4J Configuration File

Oracle Virtual Directory のロギング・アクティビティのカスタム構成が含まれる構成ファイルです。詳細は、オラクル社カスタマ・サポート・センターに問い合せてください。

Log Level

Oracle Virtual Directory のコンソール・ログで要求されるレポート作成のレベルです。Oracle Virtual Directory では、次のレベルがサポートされます。

- Off: すべての出力が抑止されます。
- Fatal: 致命的なエラーのみがレポートされます。
- Warn: 致命的なエラーおよび警告がレポートされます。
- Info: 基本的なステータス・メッセージ、エラーおよび警告がレポートされます（デフォルト）。
- Debug: Oracle Virtual Directory によるすべてのアクティビティの詳細なレポートがログに出力されます。通常、この設定は、新しいアプリケーションまたは構成をテストする際の、統合や品質保証の試みをサポートするために使用されます。
  注意 : この設定では大規模なログ・ファイルが生成されるため、パフォーマンスへの影響が生じます。
- Dump: Oracle Virtual Directory の操作に関する最大限の診断および構成情報が提供されます。
  注意 : この設定では大規模なログ・ファイルが生成されるため、パフォーマンスへの影響が生じます。

Log To Console

このフラグを使用すると、システム・コンソール（stdout）へのロギングが可能になります。このフラグをオフにして、指定されたコンソール・ログ・ファイル（有効な場合）のみにロギングされるようにすると、最大限のパフォーマンスが得られます。
デフォルト : 有効

Log File

記録されたイベントが格納されるファイルです（サーバーのインストール・ディレクトリに対して相対的に指定されます）。このファイルは、ログのローテーション・スケジュールに従って、1 日 1 回ローテーションが行われます。
デフォルト : log/vde.log

Access Log To File

Oracle Virtual Directory でアクセス・ログ・ファイルにトランザクション・アクティビティを出力するかどうかを示します。
デフォルト : 有効

Access Log File

記録されたトランザクション・イベントが記録されるファイルです（サーバーのインストール・ディレクトリに対して相対的に指定されます）。
デフォルト : log/access.log

サーバー割当て [Engine/Server/Quotas]

Oracle Virtual Directory では、匿名ユーザーまたは認証済ユーザーに対してサーバーが戻すことが可能なエントリ数など、項目を規制する機能が提供されます。また、インバウンド・トランザクションの通信量を制限する機能も提供されます。この機能を使用すると、プロキシ設定されたソースを DoS 攻撃から保護したり、LDAP の通信量を制限して制限付きディレクトリ・インフラストラクチャ・リソースへのアクセスを制御できます。

Anonymous Search Limit

匿名クライアントに対して戻されるエントリの最大数です。
デフォルト : 1000

Authenticated Search Limit

認証済（バインド済）ユーザーに対して、各問合せによって戻されるエントリの最大数です（Oracle Virtual Directory のルート・アカウントは制限の対象外です）。
デフォルト : 10,000

Enforce Quotas

Oracle Virtual Directory によってアクティビティの割当てが強制的に実行されるかどうかを決定する、全体的なフラグです。これは、すべての「Activity Limit」設定に適用されます。
デフォルト : 無効

Rate Period

割当てが評価される期間をミリ秒単位で表します。特定の期間において、設定済の割当てを超過したすべてのサブジェクトまたは IP アドレスは削除されます。評価期間が終了すると、使用状況統計はリセットされます。
デフォルト : 30,000（30 秒）

Maximum Connections

ソースに関係なく Oracle Virtual Directory によって許可される接続の最大数です。
デフォルト : 0

> **警告 :** この値を設定した場合、割当てを超過するとすべてのクライアントがブロックされるため、DoS の状況を自主発生させる可能性があります。プロキシ設定されたデータ・ソースのオーバーロードを防ぐ目的でのみ使用してください。値 0 は無制限を意味します。

Maximum Operations / Connection

接続ごとに実行可能な操作の最大数です。値 0 は無制限を意味します。
デフォルト : 0

Maximum Connections / Subject

個々のバインド済 DN（または匿名）が同時に持つことができる最大接続数です。値 0 は無制限を意味します。
デフォルト : 0

Maximum Connections / IP Address

個々の IP アドレスが同時に確立することができる最大接続数です。値 0 は無制限を意味します。
デフォルト : 0

Exempt Subjects
強制の対象外となるサブジェクト（識別名）をカンマ区切りで示したリストです。ルート・アカウント（cn=admin）は常に除外されます。

Exempt IP Addresses
割当て強制の対象外となる IP アドレスをカンマ区切りで示したリストです。たとえば、一般的に、レプリケーション・パートナまたは他のポーリング・システムが対象外になる場合があります。

サーバー・セキュリティ [Engine/Server/Security]

Root User DN
スーパーユーザー・アカウント・ディレクトリの識別名です。（Oracle Virtual Directory Manager クライアントから）このディレクトリまたは管理ゲートウェイにアクセスした場合、ACL チェックは省略されます。
デフォルト : cn=admin

Root Password
ルート・ユーザー・アカウントのパスワードです。このパスワードを設定するには、Oracle Virtual Directory Manager の GUI でクリアテキスト値を入力するか、server.os_xml ファイルを直接編集します。サーバーが再起動されると、SSHA アルゴリズムを使用して値が自動的にハッシュされます。

Admin Member URL
管理サービス・ゲートウェイの使用権限を持つユーザーのグループを定義する LDAP URL です。これらのユーザーは、管理ゲートウェイを介して（たとえば Oracle Virtual Directory Manager から）サーバーにアクセスする際に、ルートに近い権限を持ちます。

SSL Key Database
サーバー・キーを保持する Java キーストアの場所です（サーバーのルート・ディレクトリに対して相対的に指定されます）。
デフォルト : conf/keys.jks

SSL Key Db Password
キーストアにアクセスするためのパスワードです。
デフォルト : changeit

Adapters Key Alias
SSL によるクライアント認証がアダプタによって要求された場合に Oracle Virtual Directory で使用するキーの名前（別名）です。

サーバー・ビュー [Engine/Server/Views]

ビューは、ビューのメンバーの割当てに基づいて、異なるアプリケーションで Oracle Virtual Directory の異なる情報を表示するために使用できます。ビューの定義には任意の名前が使用され、その後、そのビューのメンバーであるユーザー識別名（DN）または IP アドレスの静的リストが定義されます。

特定のビューで表示されるデータは、ルーティング・システムによって決定されます。各アダプタのルーティング構成では、いずれかのアダプタが属す可能性があるビューに関するオプションの設定があります。アダプタに対してビューが選択されていない場合、そのアダプタはデフォルト・ビューに属します。ビューに割り当てられていないすべてのクライアントは、デフォルト・ビューの一部であるすべてのアダプタを表示できます。アダプタが特定のビューに割り当てられている場合は、そのビューのメンバーのみが、割り当てられたアダプタからのデータを表示できます。

注意 : ビューは、アダプタに割り当てられるまでアクティブになりません。

詳細は、このマニュアルの「ルーティング」の章を参照してください。

サーバー管理ゲートウェイ構成 [Engine/Server/Admin Gateway]

管理ゲートウェイは、Oracle Virtual Directory Manager の管理システムが Oracle Virtual Directory との通信に使用するサーバー・インタフェースです。このリスナーによって、Oracle Virtual Directory の完全な管理要件を構成する SOAP、DSMLv2 および HTTP ベースのサービスが提供されます。

Host NIC

ホスト・システムに複数のネットワーク・インタフェース・カード（NIC）が使用されている場合、Oracle Virtual Directory を割り当てて、1 つのネットワーク・カードのみに管理サービスを提供できます。NIC の IP アドレスをこのフィールドの値として使用してください。すべてのネットワーク・インタフェースをリスニングする場合は、このフィールドを空白にしてください。

Host Port

Oracle Virtual Directory によって管理サービスが提供されるポートです。このポートは、Oracle Virtual Directory Manager の管理クライアントによって、Oracle Virtual Directory との通信に使用されます。
デフォルト : 8888

SSL/TLS

セキュアな通信モードを有効にするために、SSL/TLS を選択します。警告 : このインタフェースは、Oracle Virtual Directory のルート・レベルの管理を実行するために使用されるため、SSL/TLS の使用を強くお薦めします。
デフォルト : 有効（サーバー上）

Server Key Alias

SSL 通信を提供するために Oracle Virtual Directory で使用されるサーバーの秘密鍵の別名を選択します。それぞれのリスナーで異なるキーを使用できます。選択されたサーバー・キーには、クライアントがサーバーのアドレス指定に使用する DNS ホスト名と同じ共通名を使用する必要があります。

Admin Member URL

管理サービス・ゲートウェイの使用権限を持つユーザーのグループを定義する LDAP URL です。これらのユーザーは、管理ゲートウェイを介して（たとえば Oracle Virtual Directory Manager から）サーバーにアクセスする際に、ルートに近い権限を持ちます。

リスナー構成 [Listener/"LDAP"]

Oracle Virtual Directory では、LDAP と HTTP の 2 種類の接続を介して、クライアントへのサービスを提供します。LDAP は、LDAPv2/v3 ベースのサービスを提供する場合に使用されます。一方、HTTP は、DSMLv2 などの 1 つ以上のサービス、または、XSLT が有効化されているゲートウェイによって提供される基本的なホワイト・ページ機能が提供されます。

Oracle Virtual Directory では、リスナー数の制限はなく、リスナーを使用しない（管理ゲートウェイへのアクセスのみを制限する）ことも可能です。通常、セキュア LDAP および非セキュア LDAP の実行時には、少なくとも 2 つのリスナーが構成されます。1 つはポート 389 でリスニングする標準 LDAP 用で、もう 1 つはポート 636 で実行されるセキュア LDAP（SSL）用です。

すべてのリスナーの構成は、通常、Oracle Virtual Directory Manager の管理クライアントを介して管理されるか、サーバー上の listeners.os_xml 構成ファイルを編集して管理されます。

LDAP リスナー

LDAP リスナーは、Oracle Virtual Directory で最も使用頻度が高いリスナーで、Oracle Virtual Directory の基本インストールの際にデフォルトで構成されます。

Listener Name

デプロイメント用に最も有意味な、リスナーの一意の名前です。この名前は、Oracle Virtual Directory Manager の構成タブおよびナビゲータで使用されます。

Host Address

このオプションは、複数のネットワーク・カードを使用するシステム上で、バインド先ネットワーク・インタフェース・カードのアドレスを指定します。空白、つまり未指定のままにすると、リスナーはすべてのネットワーク・アダプタで使用可能になります。特定のネットワーク・インタフェースにバインドするには、そのインタフェースの IP アドレスまたは DNS ホスト名を指定します。

Host Port

このオプションは、リスナーによってサービスが提供されるポートを指定します。LDAP のデフォルト・ポートは 389 です。HTTP ベースのサービスのデフォルト・ポートは 8080 です。各ポートでは、常に、サーバーごとに 1 つのリスナーのみがアクティブになります。

ヒント： 既存のサーバー（たとえば Active Directory ドメイン・コントローラ）と同じサーバー上に Oracle Virtual Directory をインストールする場合は、3890 などの既存のサービスと競合しない新規ポートを選択してください。

Secure

選択すると、Oracle Virtual Directory により、指定されたポートで SSL/TLS サービスが有効になります。

Key Name

Oracle Virtual Directory が SSL サービスの提供のために使用する、サーバー証明書の別名を指定します。
注意： Oracle Virtual Directory では、LDAP リスナーごとに異なるキーを使用できます。

ヒント： サーバー・キーを管理するには、右クリックしてアクション・メニューを表示し、「Server Navigator」アクションの「Manage Server Keys」機能を使用します。

Threads

LDAP リスナーでは、このオプションを使用して、受信したリクエストを処理するために同時に実行される、アクティブなワーカー・スレッドの数を指定できます。不足している場合は、スレッドの数がリスナーによって自動的に増加されます。このパラメータは、Oracle Virtual Directory に対して、リソースを事前に割り当てるために必要な同時クライアントの数を示します。

HTTP リスナー [Listener/"HTTP"]

HTTP リスナーは、ディレクトリ・データへの単純な Web ブラウザ（html）アクセスまたは DSMLv2 形式の Web サービス・アクセス、もしくはその両方を提供する多機能リスナーです。

基本構成

Listener Name

Oracle Virtual Directory Manager の GUI および構成ファイルで HTTP リスナーを一意に識別するための固有の名前です。
デフォルト : HTTP

Host Address

このオプションは、複数のネットワーク・カードを使用するシステム上で、バインド先ネットワーク・インタフェース・カードのアドレスを指定します。空白、つまり未指定のままにすると、リスナーはすべてのネットワーク・アダプタで使用可能になります。特定のネットワーク・インタフェースにバインドするには、そのインタフェースの IP アドレスまたは DNS ホスト名を指定します。

Host Port

このオプションは、リスナーによってサービスが提供されるポートを指定します。HTTP ベースのサービスのデフォルト・ポートは 8080 です。各ポートでは、常に、サーバーごとに 1 つのリスナーのみがアクティブになります。

Secure

選択すると、Oracle Virtual Directory により、指定されたポートで SSL/TLS サービスが有効になります。

Key Name

Oracle Virtual Directory が SSL サービスの提供のために使用する、サーバー証明書の別名を指定します。
注意 : Oracle Virtual Directory では、HTTP リスナーごとに異なるキーを使用できます。

> **ヒント :** サーバー・キーを管理するには、右クリックしてアクション・メニューを表示し、「Server Navigator」アクションの「Manage Server Keys」機能を使用します。

Threads

HTTP リスナーでは、このオプションを使用して、受信したリクエストを処理するために同時に実行される、アクティブなワーカー・スレッドの数を指定できます。不足している場合は、スレッドの数がリスナーによって自動的に増加されます。このパラメータは、Oracle Virtual Directory に対して、リソースを事前に割り当てるために必要な同時クライアントの数を示します。

DSMLv2 サービス

Realm Name
DSMLv2 ゲートウェイで認証に使用されるレルムです。これは、ユーザーへの HTTP ブラウザ・チャレンジに表示される名前です。

Web ゲートウェイ・サービス・セクション
Web ゲートウェイ・サービスの詳細は、このドキュメントの付録の「Web ゲートウェイ」を参照してください。

HTDocs Path
XSLT ファイルおよび HTML ファイルが格納されるパスです。Oracle Virtual Directory のルート・インストールに対して相対的に指定されます。

Certificate Attributes
テンプレートに対して、バイナリ PKI 証明書情報を含む属性を示します。
デフォルト : usercertificate

Photo/Image Attrs
テンプレートに対して、グラフィック・イメージを含む属性を示します。テンプレートによって HTML が自動的に変更され、イメージが指定のサイズで表示されます。

Image Display Size
テンプレートによって表示される写真 / イメージの属性のサイズです。

Allow Anonymous
匿名ユーザーが Web ゲートウェイにアクセスできるかどうかを決定します。

Realm Name
HTTP ブラウザのログイン・チャレンジでユーザーに表示される認証レルム名です。

Search Root
サブツリーでのユーザー識別名（UID）の検索を Web ゲートウェイが開始するディレクトリ・ツリーのルート識別名（ネームスペース）です。この検索はユーザー認証チャレンジの後に提供されます。

Search Attributes
UID の検索時に、Web ゲートウェイは、提供された値を uid、mail、cn などのリスト内の属性のいずれかと照合することを試みます。

User Objectclasses
Web ゲートウェイが認証対象ユーザーの検索に使用するオブジェクト・クラスです。

Cache Life
終了ディレクトリ・ソースとともに格納されているユーザー資格証明を Oracle Virtual Directory が再び問い合せるまでの最大時間を分単位で示します。

アダプタ構成

Oracle Virtual Directory は、次の 5 つのアダプタ・タイプを使用できるように設定できます。

- ローカルファイルに格納されているデータ（ローカル・ストア・アダプタ - 以前の標準アダプタ）
- 別の LDAP ディレクトリからのデータ（LDAP プロキシ・アダプタ）
- リレーショナル・データベースからのデータ（DB プロキシ・アダプタ）
- NT ドメインからのデータ（NTLM アダプタ）
- 複数のソースからのデータ（結合ビュー・アダプタ）

アダプタは通常、Oracle Virtual Directory Manager の管理システムを使用して管理されます。また、サーバー上の adapters.os_xml 構成ファイルを編集して管理することもできます。

一般的にアダプタは、Oracle Virtual Directory Manager のサーバー・ナビゲータにあるサーバー・アイコンを右クリックし、追加するアダプタ・タイプを選択してサーバーに追加します。同様の方法で、アダプタの名前変更や削除も必要に応じて行えます。アダプタを追加または編集した後は、サーバー上で右クリックし、「Save To Server…」または「Save All To Server」を選択すると、構成を保存できます。

それぞれのアダプタには、「Configuration」、「Routing」および「Plug-ins」の 3 つの構成タブがあります。次項では、各アダプタ・タイプおよびそれぞれの固有の構成プロパティについて説明します。ルーティングは、個々のアダプタで特定の操作が有効であるかどうかを判別したり、属性のフィルタ処理などの一般的な機能を実行するために Oracle Virtual Directory で使用される構成データです。ルーティングの詳細は、後の章で説明します。

プラグイン構成は、特定のアダプタのコンテキスト内で実行される Oracle Virtual Directory のプラグインおよびマッピングを定義します。詳細は、後述のプラグイン管理に関する章を参照してください。

ローカル・ストア・アダプタ

Oracle Virtual Directory では、ローカルの単純なフラットファイル・データベースにデータを格納するローカル記憶域アダプタ（LSA）が提供されます。LSA を使用した場合、Oracle Virtual Directory は従来のディレクトリ・サーバーと同様に機能するため、本番品質のディレクトリ・サーバーとしての使用には適していません。

LSA は、次の 2 つの目的のために設計されています。

- 開発者が、別の LDAP サーバーをインストールせずに独自のワークステーション上で開発ディレクトリ実行できるようにする。特に、（従来の仮想設定で）Oracle Virtual Directory を本番で使用する場合。
- 複数のアダプタがインストールされており、その全アダプタがブランチとして作成されている場合に、ディレクトリ・ツリー最上位の単一のエントリを保持する。

たとえば、LDAP の 1 つがネームスペース `ou=staff,dc=mycompany,dc=com`、もう 1 つが `ou=customers,dc=mycompany,dc=com` として構成されている場合、最上位のエントリとして `dc=mycompany,dc=com` を指定できます。Oracle Virtual Directory は最上位のエントリが未指定でも動作し、ほとんどの LDAP クライアントでも支障はありません。ただし、一部の LDAP クライアント・アプリケーション、特に GUI デスクトップの LDAP 管理クライアントでは、サーバーの最上位にエントリが指定されていないと、誤ったエラー・メッセージが戻される場合があります。そのため、Oracle Virtual Directory の LSA を使用して、他のアダプタより上位の単一エントリを格納できます。

高い可用性、スケーラビリティおよびバックアップをサポートできる従来型のディレクトリ・サーバーが必要な場合は、Oracle Internet Directory の使用を検討してください。この従来型の LDAP サーバーではストレージ・システムとして Oracle Relational Database が使用されます。

Oracle Virtual Directory では、複数の LSA アダプタを定義できますが、一般的には、Oracle Virtual Directory の各インストールに対して少なくとも 1 つの LSA アダプタが構成されます。その主な理由は、異なるデータベース・ファイルにディレクトリ・ツリーの異なる部分を格納するためです。通常は、バックアップおよびリカバリを考慮して行われます。たとえば、ディ

レクトリ・ツリーの中で相対的に静的なブランチは、頻繁にバックアップする必要はない可能性があります。

基本設定

Name
アダプタの一意の名前です。このアダプタを参照する他の構成フィールド、およびサーバー・ナビゲータや編集タブで使用されます。

Root
このフィールドは、このアダプタが情報を提供するルートの識別名を定義します。定義されたDN およびその下にある子エントリは、アダプタのネームスペースと呼ばれます。このフィールドに入力された値は、仮想ディレクトリのクライアントに表示される値です。この値は、識別名（たとえば o=Oracle, c=US または dc=oracle,dc=com）として指定される必要があります。

Active
アダプタはアクティブまたは非アクティブとして構成されます。非アクティブとして構成されたアダプタは、サーバーの再起動時またはアダプタの起動の試行時には起動しません。この設定の背後にある主要な目的は、古い構成を構成から削除せずに使用可能にしておく、つまり待機させておくことにあります。
デフォルト : 有効

Read Only
選択すると、LSA アダプタでは変更トランザクションが受け入れられなくなり、検索のみで使用可能になります。
デフォルト : false（読取り / 書込みモード）

索引設定

Presence
このディレクティブには、エントリに含まれていることを迅速に識別する必要がある属性タイプをカンマ区切りで示したリストが含まれます。これは、(attrname=*) スタイルの検索フィルタが動作するために必要です。

Exact
完全索引は、完全一致索引（例 : sn=smith）の検索をサポートするためのものです。順序付け索引を使用する場合は、この索引は不要になります。

Ordering
実際には、このディレクティブは順序付け検索（例 : sn<=Smith）を有効にするだけでなく、完全一致検索のサポートが含まれており、最初の部分列の検索（例 : sn=Smi*）を有効にします。このフィールドでは、カンマ区切りの属性リストが保持されます。LDAP フィルタでは、<= および >= の順序付け関係のみが許可されます。< および > は、LDAPv3 ではサポートされていません。

Substring
このオプションは、順序付け索引に加えて最後の部分列の検索（例 : sn=*ith）が必要な場合のみ、必要になります。最初の部分列の検索は、多くの場合順序付け索引を使用して処理されます。このオプションは、カンマ区切りの属性名リストを使用して構成されます。

Search Un-indexed
このオプションは、特に索引付けされていない属性の、パフォーマンスの低い検索を有効にします。

ヒント:「Search Un-indexed」が無効の場合は、索引付けされていない属性を検索しても結果が戻されません（false と評価されます）。

ローカル・データベース設定

File

LSA データ・ファイルのパスおよび一意のファイル名です（Oracle Virtual Directory のインストールに対して相対的に指定されます）。

警告: 1 つ以上の LSA アダプタが使用されている場合、各アダプタに対して一意の値を指定しないと、データが破損します。

Password Hash Mode

アダプタに使用するパスワード・ハッシュ・アルゴリズムを定義します。有効な値は、CRYPT、SSHA、SHA および PLAIN です。これらの中で最もセキュアなアルゴリズムは SSHA ですが、互換性を考慮してその他のアルゴリズムも提供されています。PLAIN を指定すると、値はハッシュされずに内部の LSA データ・ストアに残ります。
デフォルト : SSHA

Auto RDN

エントリの追加時には、LDAP RFC によって、RDN（相対識別名）すなわち一番左の DN 項目が追加対象のエントリの属性リストに含まれていることが要求されます。一部のディレクトリ製品ベンダーではこれが無視されており、属性リストに RDN 値が含まれていない場合でも許可されます。これにより、この動作に依存するアプリケーションで互換性の問題が生じます。「Auto RDN」を有効にすると、Oracle Virtual Directory によって、不足している属性が自動的に作成されます。
デフォルト : 無効

Auto Compact

データベースのバックアップが正常に実行された後、Oracle Virtual Directory では、必要に応じてデータベース・ファイルを圧縮できます。LSA データが頻繁に変更される場合は、データベース・サイズの拡大を抑えるのに役立ちます。
デフォルト : 無効

警告: Windows プラットフォームでは、この機能を無効にすることを強くお薦めします。Windows の一部のシナリオでは、ファイルの名前変更機能が保証されません。そのため、データが破損または損失する可能性があります。

Transaction Log Size

新しいエントリが追加または変更された場合は、まずトランザクション・ログに書き込まれます。これにより、トランザクションを確実にディスクに書き込むと同時に、迅速なアプリケーション・レスポンスが可能になります。

このオプションは、トランザクション・ログの切捨てサイズをバイト単位で指定します。未処理のトランザクションによって、ログのサイズがこのオプションの値を超えたとしても、データ・ストアに格納されず索引付けされていないエントリは、トランザクション・ログから削除されることはありません。

頻繁に切り捨てられる小規模のトランザクション・ログを使用した場合、大量のエントリを追加すると、かなりのオーバーヘッドが追加されます。実際には、初期のバルク・ロードではトランザクション・ログを最大限に大きくしておき、本番に入る前にサイズを小さくする方が有利です。

Cache Size

このオプションは、LSA アダプタによってメモリーにキャッシュされるエントリの数を指定します。常に、最後にアクセスまたは書き込まれたエントリが含まれます。必要なメモリーの量は、エントリのサイズによって決定されます。

> **ヒント：** 大規模なエントリ（たとえばグループやバイナリ・オブジェクト）を格納すると、Oracle Virtual Directory で、通常より多くのメモリーが消費される可能性があります。Oracle Virtual Directory で使用できる全体的なメモリー量を増やすことを検討してください。

バックアップ設定

File

自動バックアップの格納先パスおよびファイル名です（Oracle Virtual Directory のインストールに対して相対的に指定されます）。新しいバックアップ・ファイルが作成されるたびに、ファイルの番号が変わります（backup-file は最新のファイル、backup-file0 は 1 つ前のファイル、backup-file1 は 2 つ前のファイル、以下同様）。ファイル名には、ファイル形式 .zip が自動的に付加されます。別の LSA アダプタによる上書きを防ぐため、バックアップ・ファイル名は LSA アダプタ固有のものにする必要があります。

Backup Time

自動バックアップの開始時間を時間（0 ～ 23）および分（0 ～ 59）で指定します。

Max Backup Files

このパラメータは、バックアップ・ファイルのローテーションで保持されるバックアップの最大数を指定します。

LDAP アダプタ構成

Oracle Virtual Directory の LDAP アダプタでは、外部の LDAPv3 ディレクトリにプロキシを設定し、外部コンテンツを Oracle Virtual Directory の一部として表示する機能が提供されます。これを可能にしているのは、ディレクトリ構造化およびスキーマ変換の自動処理です。

> **ヒント：** 複数のアダプタを構成することで、Oracle Virtual Directory では、分離された複数のディレクトリを単一の仮想ディレクトリ・ツリーにフェデレートできます。

LDAP アダプタは、追加のフォルト・トレランスおよびパフォーマンス・ソリューションを提供します。これには、特定の検索条件や可用性によるロード・バランシング要件、および操作タイプ（たとえば検索と変更）に基づいて、LDAP 問合せの通信を複数の LDAP ディレクトリのレプリカに向ける機能が含まれます。

LDAP アダプタの機能により、パフォーマンスの高いシナリオでは、かぎられた接続セット全体で多数のクライアントからの負荷を分散させることで、ソース・ディレクトリで発生する接続負荷が平滑化されます。これが可能なのは、プロキシによって Oracle Virtual Directory とソース・ディレクトリ（およびそのレプリカ）間の接続プールが管理されるためです。これによって、接続処理の負荷が軽減され、スレッドの最大負荷を超過しないように Oracle Virtual Directory でのソース・ディレクトリへの通信が制限されるため、ソース・ディレクトリの動作が速くなります。

LDAP アダプタ固有の特徴は、接続レベルではなく、LDAP 操作レベルでのバランシングおよびフォルト・トレランスを提供することです。たとえば、何千ものアクティブ・クライアントがある場合、Oracle Virtual Directory は負荷を 10 または 20 の安定した接続まで減らして何千もの問合せを処理します。また、特にビジーなクライアントの場合は、定義されたロード・バランシングの目標に従って、各ワーカー・スレッド間で負荷を分散します。

ディレクトリ・ネームスペースおよび属性マッピング

Oracle Virtual Directory の LDAP プロキシの構成方法は 2 つあります。純粋なプロキシとして構成する方法と仮想ディレクトリのプロキシとして構成する方法です。

純粋なプロキシとして構成すると、Oracle Virtual Directory で変換処理がまったく実行されないようにすることができます。これは単純に、Oracle Virtual Directory を可用性とファイアウォールの問題を解決するためのディレクトリ・ルーターとして機能させることになります。構成上では、パラメータ「Adapter Root」および「Adapter Remote Base」にまったく同じ値を指定します。

「Adapter Root」および「Adapter Remote Base」に異なる値を設定してネームスペース・マッピングを暗示する単純なネームスペース・マッピングを提供するには、多くの場合、単純に 1 つのディレクトリ・ツリーのネームスペースを別の名前に変更する必要があります。

例 :

プロキシ設定されたディレクトリの base: `ou=People,o=Airius.com` に `People` エントリがあるとします。ローカル・ディレクトリでは、Airius エントリへのコールがローカルの People フォルダに表示されるように設計されています。ここで、構成を次のように設定したとします。

ルート : ou=Airius,ou=People,dc=YourCompany,dc=com
リモート・ベース : ou=People,o=Airius.com

結果として、LDAP アダプタを介して問合せを実行すると、ou=People, o=Airius.com の下にあるすべてのエントリの DN が、指定された DN ルート ou=Airius,ou=People, dc=YourCompany,dc=com の下にあるように表示されます。

デフォルトでは、すべての属性は、プロキシ設定されたディレクトリから Oracle Virtual Directory を介して Oracle Virtual Directory クライアントにそのまま渡されます。LDAP アダプタでは、DN を含む属性の基本的な DN 変換を実行できます。これは、構成パラメータ DN 属性リストによって指定されます。また、Oracle Virtual Directory には、オンザフライで属性を変換する機能が含まれています。この機能では、マッピングおよびプラグイン（後続の各章を参照）を使用して、リモートからのデータが直接 Oracle Virtual Directory に渡されます。

読取り / 書込み / 名前変更 / 比較のサポート

LDAP アダプタでは、読取り、追加、変更、削除および名前変更の機能が完全にサポートされます。

LDAP の名前変更操作のサポートには、アダプタ間の名前変更、つまり移動の機能も含まれます。アダプタ間で行われる LDAP 名前変更（すなわち move）の LDAPv3 moddn 機能では、実際にはカット・アンド・ペーストが行われます。このトランザクションを安全に行うために、Oracle Virtual Directory によって移動先ディレクトリへの追加が正常に行われたことが確認されるまで、元のエントリは削除されません。なんらかの理由で移動先ディレクトリでの追加処理が失敗した場合、操作は中断され、移動先ディレクトリからエラーが戻されます。

アダプタ間またはプロトコル間で共通のサポートを提供するために、LDAP Compare 操作は、自動的に LDAP Get または LDAP Bind 操作に変換されます。

アクセス制御および LDAP アダプタ

Oracle Virtual Directory は、必要なアクセス制御機能を指定する IETF RFC 2820「Access Control Requirements for LDAP」をすべてサポートします。また、Oracle Virtual Directory では、IETF の「LDAP Access Control Model for LDAPv3」（2001 年 3 月 2 日草稿）に基づくアクセス制御が強制されます。この草案では、RFC 2820 をベンダーに中立的に実装する方法が指定されています。

Oracle Virtual Directory のアクセス制御は、2 つのレベルで適用されます。

- リモート LDAP サーバーのアクセス制御
- Oracle Virtual Directory のアクセス制御

リモート LDAP サーバーのアクセス制御

Oracle Virtual Directory は LDAP クライアントとして機能するため、リモート・アダプタ・ディレクトリ（たとえば LDAP）でのアクセス制御に従う必要があります。この動作方法は、ベンダーのアクセス制御の実装によって異なります。

Oracle Virtual Directory は、プロキシ設定されたディレクトリ・サーバーに接続すると、独自の資格証明またはバインドされているエンド・クライアントの資格証明を使用できます。これにより、全体的なアクセス制御の動作方法がかなりの影響を受けます。Oracle Virtual Directory によってソース・ディレクトリにユーザー資格証明が渡された場合は、エンドツーエンドのユーザー・コンテキスト・アクセス制御が有効になります。これにより、リモート・サーバーで、バインドされたエンド・クライアントの資格証明が認証されます。Oracle Virtual Directory によって独自の資格証明が渡される場合、Oracle Virtual Directory は、プロキシ設定されたディレクトリのサーバー資格証明に対して強制されるアクセス制御を受けると同時に、独自のユーザー固有アクセス制御を提供する必要があります。

ユーザーのエントリが LDAP プロキシで表される場合にそのユーザーのパスワードを使用してバインド・リクエスト（認証）を処理する際には、Oracle Virtual Directory は、2 つのうちどちらかの方法で資格証明を検証できます。

「Pass Credentials」が「Always」または「Bind-only」に設定されている場合は、Oracle Virtual Directory により、指定されたユーザー識別名およびパスワードが、プロキシ設定された LDAP ディレクトリに渡されます。プロキシ設定されたディレクトリは、ユーザーが入力したパスワードの有効性の判別に対応します。リモート・パスワード検証の成功または失敗は、透過的にユーザーに戻されます。

「Pass Credentials」が「Never」に設定されている場合は、Oracle Virtual Directory はアダプタで指定されたアカウントを使用してリモート・ディレクトリに接続し、暗号化されたパスワード値を取得します。

注意： アダプタ構成で指定されたアカウントは、プロキシ設定されたディレクトリによってこの値へのアクセスが付与されている必要があります。付与されていない場合は、バインドが失敗として処理されます。暗号化された値が Oracle Virtual Directory に戻されると、Oracle Virtual Directory では、ユーザーのパスワードが暗号化され、プロキシ設定されたサーバーから戻された暗号化済の値と比較されます。これらの値が一致した場合は、Oracle Virtual Directory によって正常なバインドが戻されます。

注意： ユーザーが証明書のバインドを実行する場合は、「Pass Credentials」の「Never」モードが暗黙に指定されます。この場合は、Oracle Virtual Directory により、リモートの LDAP プロキシからクライアントの公開鍵を取得してクライアント資格証明が確認されます。

Oracle Virtual Directory のアクセス制御

Oracle Virtual Directory には、独自の標準に加えて、すべてのアダプタに対して強制される草案の標準準拠アクセス制御があります。このセキュリティは、すべてのアダプタで一様に適用され、一貫したセキュリティ実装を提供します。資格証明の引渡しに関する問題、およびグループに関するアクセス制御識別子の詳細は、セキュリティおよびアクセス制御に関する章を参照してください。

基本設定

Name
アダプタの一意の名前です。このアダプタを参照する他の構成フィールド、およびサーバー・ナビゲータや編集タブで使用されます。

Root
このフィールドは、このアダプタが情報を提供するルートの識別名を定義します。定義されたDNおよびその下にある子エントリは、アダプタのネームスペースと呼ばれます。このフィールドに入力された値は、仮想ディレクトリのクライアントに表示される値です。この値は、識別名（たとえばo=Oracle, c=USまたはdc=oracle,dc=com）として指定される必要があります。

Active
アダプタはアクティブまたは非アクティブとして構成されます。非アクティブとして構成されたアダプタは、サーバーの再起動時またはアダプタの起動の試行時には起動しません。この設定の背後にある主要な目的は、古い構成を構成から削除せずに使用可能にしておく、つまり待機させておくことにあります。
デフォルト : 有効

LDAP 設定

Remote Base
ローカルOracle Virtual Directoryのルート接尾辞に対応する、リモート・サーバーのディレクトリ・ツリー構造内の場所です。これは、Oracle Virtual Directoryによって現在のアダプタの検索および操作がすべて実行される、リモート・ディレクトリ内の場所です。

LDAPアダプタにより、リモート・ベースからアダプタ・ルート・ベースへの全エントリの自動マッピングが適用されます。

DN Attributes
ネームスペース変換が要求される識別名（DN）として処理される属性のリストです（例 : `member, uniquemember, manager`）。たとえば、プロキシ設定されたディレクトリからグループ・エントリを読み取る場合、Oracle Virtual Directoryによって自動的にそのグループ・エントリ自体のDNが変換されると同時に、`uniquemember`属性または`member`属性が「DN Attributes」リストに含まれる場合は、これらの属性も変換されます。

> **ヒント :** クライアント・アプリケーションで必要になることが判明している属性のみを変換してください。使用可能なDN属性をすべて入力する必要はありません。これらの属性をすべて入力した場合は、多少ながらプロキシで余分なCPU時間が消費されます。

Escape Slashes
ディレクトリに / が含まれている場合、Oracle Virtual Directoryでは、必要に応じてバックスラッシュ（\）を使用してスラッシュをエスケープできます。一部のディレクトリ・サーバー製品では、エスケープされていないスラッシュが受け入れられますが、それ以外の製品では拒否されます。必要に応じて、このオプションを選択してスラッシュのエスケープを有効にしてください。

Follow Referrals
有効にすると、LDAPアダプタは、クライアントのかわりにソース・ディレクトリからの参照を追跡します。無効にすると、参照はブロックされ、クライアントに戻されなくなります。

Proxy Paging Size

プロキシ設定されたディレクトリ内で、プロキシがページングされた結果を使用できるようにします。このオプションが最も使用されるのは、ディレクトリによって問合せ内の結果数が制限されている場合です。値が 0 の場合は制御がオフになり、結果数は制限されません。正の値は、ページごとに戻される結果数を示します。この制御は、Oracle Virtual Directory のクライアントのかわりに透過的に使用されます。
デフォルト : 0

接続設定

Remote Hosts

プロキシ設定された LDAP サーバーのホストとポートおよびロード・バランシングに関する情報（オプション）のリストが含まれます。リストに含まれる各サーバーは、同じコンテンツ（レプリカ）を提供するとみなされます。Oracle Virtual Directory Manager でこのパラメータの構成を編集するには、「…」ボタンをクリックして「Remote Hosts Editor」を開きます。

- **Host:** プロキシ設定対象のサーバーの IP アドレスまたは DNS 名。
- **Port:** LDAP サービスが提供されるポート番号。
- **Load Percent:** 0 ～ 100 のいずれかのパーセント数値。特定のホストに送られる負荷の、無作為に抽出される目的のパーセント値です。入力されたホストのロード値の合計が 100 にならない場合は、Oracle Virtual Directory によってホストのロード・パーセント数値が自動的に再調整されます。この処理では、入力されたロード・パーセント値が、全ホストに対して指定されたすべてのロード・パーセント数値の合計で除算されます。

  例 : 入力されたホストが 2 つのみで、それぞれのロード・パーセント値が 33 パーセントの場合、33 は 66（2 つのサーバーの合計）の半分であるため、実際にはそれぞれのホストに 50% のロードが割り当てられます。
- **Read Only:** 読取り専用ホストにより、LDAP アダプタに対して、ホストが検索操作のみに使用されることが示されます。LDAP アダプタは、すべての変更トラフィックをリスト内の読取り / 書込みホストに自動的に送ります。

ヒント :「DNS for Auto Discovery」（一般的にはサーバーレス・バインドと呼ばれる）を使用する場合は、これらのフィールドは無視されます。

ヒント : プロキシ用に単一のホストのみを指定する場合は、注意してください。フェイルオーバー・ホストが指定されていないと、LDAP アダプタは、自動的に別のホストにフェイルオーバーすることができません。Oracle Virtual Directory がロードバランシング・システムを経由して論理 LDAP サービスに接続している場合は、単一のホストが有効になります。単一ホストを使用するシナリオでは、**「extended trying」**の設定を利用して、LDAP アダプタがシングルサーバー・サービスの障害からリカバリする方法を指定してください。

Failover Mode

「Sequential」に設定すると、障害が発生しないかぎりは、**「Remote Hosts」**で指定された最初のホストが使用されます（ロード・バランシング設定は無視されます）。障害が発生した場合は、次のホストの使用が試行されます。

「Distributed」に設定した場合は、確立された各新規接続が、「Remote Hosts」フィールドで定義されたリストを介してロード・バランスされます。

分散フェイルオーバーが最も使用されるのは、通常同じデータ・センターに含まれている LDAP サーバーのセットにプロキシを設定する場合で、ネットワーク・パフォーマンスの点では平等に使用できます。

順次フェイルオーバーは、地理的位置間でのフェイルオーバー時に最も使用されます。順次フェイルオーバーでは、LDAP アダプタにより、失敗するまで指定されたホストの使用が試行されます。現時点では、別のデータ・センターつまり地理的に離れた別の場所で使用可能な同等のホストにフェイルオーバーされます。

Extended Trying

「Extended Trying」がオンの場合は、最後のリモート・サーバーが失敗したと判断されても、このサーバーはリストから削除されません。つまり、サーバーは、それぞれの新規リクエストによって、ハートビート間隔まで待機するのではなく、継続的に接続を試行するよう強制されます。分散ディレクトリを使用する一部の環境では、**「Routing Critical」**の設定とともにこのオプションをオフにして、即座に部分的な結果が戻されるようにした方が効果的な場合があります。

Heartbeat Interval

LDAP アダプタでは、定義されている各サーバーの可用性が定期的に検証されます。この検証サイクルでは、現在無効化されているホストが復活する可能性があるほか、TCP/IP 接続のテストに失敗した現在アクティブなホストが無効としてマーク付けされます。このパラメータは、検証通過の間隔を秒数で指定します。低すぎる値を設定すると、リモート・ディレクトリへの不要な接続が発生します。高すぎる値を設定すると、障害発生時のリカバリ検出時間が長くなります。
デフォルト : 60 秒

Operation Timeout

LDAP リクエストがリモート・ホストによって承認されるのをサーバーが待機する時間をミリ秒単位で表します。この操作が失敗した場合、LDAP アダプタにより、**「Remote Host」**リスト内の次のサーバーが試行されます。構成可能な最小値は 100 ミリ秒です。設定した値が低すぎると、ビジーなサーバーで誤った傷害が発生する可能性があります。
デフォルト : 5000（5 秒）

Max Pool Connections

「Max Pool Connections」は、単一のサーバーで確立できる同時接続数を制御するための、チューニング・パラメータです。
デフォルト : 10

Max Pool Wait

LDAP アダプタによって新しい接続が生成される前に、既存の接続を使用するために LDAP 操作が待機する最大時間をミリ秒単位で表します。
デフォルト : 1000（1 秒）

Max Pool Tries

「Max Pool Connections」パラメータを上書きして新しい接続を生成する前に、操作が LDAP 接続のために待機する最大回数です。最大回数は、「Max Pool Wait」秒数を試行回数で乗算した結果です。「Max Pool Wait」が 1 秒で「Max Pool Tries」が 10 回の場合、標準プールで LDAP 接続が使用不可になってから 10 秒後に、増大した負荷の処理のためにプールが拡張されます。「Max Pool Connections」を超過してプールが拡張されるのを回避するには、試行回数を高い数値に設定してください。
デフォルト : 10

Use Kerberos

この設定を選択すると、LDAP アダプタによって、Kerberos プロトコルを使用して LDAP バインド操作が実行されます。このオプションを使用するには、krb5.conf ファイルをコピーまたは作成し、Oracle Virtual Directory の構成フォルダ内に置く必要があります。
デフォルト : 無効

krb5.conf には多数のプロパティが含まれます。

- **default_realm:** マッピングによって提供されていない場合に使用されるデフォルト・ドメインです。たとえば、あるユーザーが `uid=jsmith,ou=people,dc=myorg,dc=com` としてバインドされると、`jsmith@myorg.com` として処理されます。マップされたネームスペースにドメイン・コンポーネント（DC）ベースのルートが含まれていない場合は、かわりにこの値が置き換えられます。
- **[domain_realm]:** ドメインとレルム定義の間のマッピングを定義します（例 : .oracle.com = ORACLE.COM）。
- **[realms]:** 1 つ以上のレルムを定義します（例 : ORACLE.COM = { … }）。
- **kdc:** 特定のレルム定義の Kerberos サービスを実行するサーバーの DNS 名です。

Use DNS For Auto Discovery

「Remote Hosts」フィールドで特定の LDAP ホストを構成するかわりにこのオプションを選択すると、Oracle Virtual Directory で DNS が使用され、定義済リモート・ベースの適切な LDAP サーバーが検出されます（別名はサーバーレス・バインド・モード）。LDAP アダプタは、3 つの操作モードをサポートします。

- **None:** 標準の「Remote Hosts」フィールドの構成を使用します。
- **Standard:** Microsoft 以外のサーバーの標準 DNS 参照を使用します。すべてのサーバーは読取り - 書込みとしてマーク付けされるため、LDAP の書込みサポートを許可するために**「Follow Referrals」**を設定することをお薦めします。
- **Microsoft:** DNS サーバーは Microsoft の動的 DNS であり、ロード・バランシング構成もサポートします。Microsoft の動的 DNS サーバーに対してプロキシ設定する場合は、この設定が最もよく使用されます。これは、Oracle Virtual Directory の機能により、読取り / 書込みサーバーと読取り専用サーバーが自動検出されるためです。

注意 : この設定の使用時には、リモート・ベースにドメイン・コンポーネントのスタイル名が指定されている必要があります（例 : `dc=myorg,dc=com`）。これにより、Oracle Virtual Directory で（`myorg.com` を検索して）DNS サービス内の LDAP ホストが検出されます。

資格証明処理

Proxy DN

プロキシ設定されたディレクトリへのアクセス時に LDAP アダプタのバインドに使用されるデフォルト DN です。「Pass-through Mode」の設定に応じて、この DN がすべての操作で使用されるか、特別な場合のみ使用されます（「Pass-through Mode」を参照）。識別名の書式は、リモート・ディレクトリの書式に合せる必要があります。空白にすると匿名として処理されます。

Proxy Password

前述の「Proxy DN」値とともに使用される認証パスワードです。パスワードを設定するには、単純にクリアテキスト値を入力します。サーバーにロードされると、追加のセキュリティを提供するために、可逆的なマスクを使用して値が自動的にハッシュされます（例：`{OMASK}jN63CfzDP8XrnmauvsWs1g==`）。

Pass-through Mode

すべての操作で、Oracle Virtual Directory に示されたユーザー資格証明をプロキシ設定された LDAP に渡す場合は、「Always」に設定します。バインド用にのみユーザー資格証明をプロキシ設定された LDAP サーバーに渡し、その他のすべての操作でデフォルトのサーバー資格証明を使用する場合は、「Bind Only」に設定します。プロキシ DN 資格証明をすべての操作で使用する場合は、「Never」に設定します。

ヒント： 状況によっては、「Pass-through Mode」が「Always」に設定されていても、LDAP アダプタによってプロキシ DN が使用される場合があります。これは、ユーザー資格証明が（たとえば別のアダプタ・ネームスペースから）マップできない場合や、ユーザー資格証明がルート・アカウントの場合に発生します。

ヒント： Microsoft Active Directory フォレスト内の異なるドメイン・コントローラに複数のアダプタを定義する場合は、**「Routing Bind-Include」**設定を使用して、LDAP アダプタが他のアダプタ（同じ AD フォレストを指す複数のアダプタ）からの資格証明にプロキシを設定するように指定できます。

DB アダプタ構成

DB アダプタは、完全な機能を持つ LDAP-JDBC 間のゲートウェイで、すべての LDAP 操作（追加、バインド、削除、取得、変更、名前変更）を等価の準備済 SQL 文のコードに変換できます。DB アダプタでは、LDAP 検索を実行する目的で、JDBC クラス・ライブラリを使用してデータベースへの接続を作成します。論理的には、（JDBC-ODBC ライブラリを介して）JDBC または ODBC をサポートするすべてのデータベースは接続可能です。必要なデータベース・ライブラリは、通常、データベース分散で提供されるか、データベース・ベンダーのサポート・サイトから入手できます。

DB アダプタは、柔軟な表 / 列マッピングをサポートしているため、ほとんどの LDAP オブジェクトをオンザフライで構築できます。これを行うには、表からの LDAP オブジェクト・クラスの導出方法を定義するマッピング・ウィザードを使用します。DB アダプタのサポートには、次のものが含まれます。

- Oracle Virtual Directory 内または RDBMS サーバーのビューを介して実行される複数表結合。
- 柔軟な、表および列 -LDAP 間属性名マッピング。
- （主キー経由で）行グループ化の使用による複数値属性マッピング。
- 単一のアダプタ接続内で複数のオブジェクト・クラスおよび階層を構成する機能。

DB アダプタは、ソース・データベースの設計を一切修正せずに機能するように設計されています。DB アダプタには、次のプロトコル関連の制限があります。

- LDAP は原子性を持つプロトコル（各操作が単純なトランザクション・ユニットで、ロールバック機能が不要）であるため、DB アダプタでは、単一の表に適用される変更のみが許可されます。
- データベース・アダプタは、複数値属性の書込みをサポートします。アダプタは、自動的にデータを非正規化して単一の表に値を書き込みます。つまり、エントリに含まれるすべての値のすべて順列が表に挿入されます。

たとえば、次のデータがあるとします。

```
cn: User
description: value one
description: value two
```

このデータは、次のように表に挿入されます。

CN	Description
User	value one
User	value two

- ストアド・プロシージャはデプロイ固有であるため、DB アダプタでは、標準の SQL アクセスのみがサポートされます。データベースのプロシージャを利用するためには、アダプタのプラグインを開発してこの統合を処理することを検討してください。
- DB アダプタの使用時に複数の LDAP 属性を単一のデータベース列にマップしようとすると、Oracle Virtual Directory によって 1 つの属性のみが戻されます。現時点では、DB アダプタの使用時に複数の属性を同じデータベース・フィールドにマップすることはできません。ただし、次善策として、マッピング・スクリプトを DB アダプタに適用できます。

アクセス制御および DB アダプタ

DB アダプタは、Oracle Virtual Directory の固有のアクセス制御に加えて、リモート・データベースに存在するアクセス制御にも関係します。LDAP アダプタとは異なり、認証されたユーザーの資格証明を Oracle Virtual Directory からリモート・データベースに渡すことはできません。つまり、Oracle Virtual Directory は、JDBC クライアントとして、「JDBC Connection」情報で指定されるアダプタの**「Database Username」**によって定義される、アダプタの RDBMS サーバーによって認証された機能に対する制約を受けます。すべての問合せは、**「Database Username」**フィールドで指定されたアカウントを使用して実行されます。

Oracle Virtual Directory では、データベースから戻された値をユーザーが指定した値と比較して、パスワード検証が実行されます。ハッシュまたは暗号化された形式でパスワードが格納される場合は、その値の接頭辞として適切なハッシュ修飾子（{crypt}、{sha}、{ssha}）を使用する必要があります。この接頭辞により、Oracle Virtual Directory に対して、パスワードの検証時に使用するパスワード・ハッシュの比較アルゴリズムのタイプが指定されます。この接頭辞は LDAP の表記規則であるため、テキスト・ファイルには存在しない可能性があります。アダプタ・マッピングを使用して接頭辞を割り当ててください（マッピングに関する章を参照してください）。

Oracle Virtual Directory では、独自の標準ベースの LDAP セキュリティおよびアクセス制御モデルが、すべてのアダプタおよびアダプタ・タイプに同時に適用されます。

ヒント： アクセス制御の適用に関する詳細は、ACL に関する章を参照してください。

JDBC Java クラス・ライブラリ

特定の JDBC ドライバを使用するには、そのドライバ・ファイルを Oracle Virtual Directory インストールの `plugins/lib` ディレクトリにインストールしておく必要があります。ドライバ・ファイルは、各製造業者からダウンロードして直接インストールするか、Oracle Virtual Directory Manager の「Manage Server Libraries」機能を使用してインストールできます。

ドライバの検出の詳細は、http://www.oracle.com/octetstring/integration.html から表示できる Oracle JDBC ドライバのページを参照してください。

JDBC ライブラリがサーバーにロードされたら、事前定義済のデータベース・タイプのいずれかを選択するか、製造業者によって定義された JDBC URL を指定して、新しい接続を定義できます。

DB アダプタ・ウィザードでは、次のデータベースの事前定義済 JDBC URL がサポートされます。

- Hypersonic
- IBM DB2
- Microsoft SQL*Server
- MySQL
- OpenBase
- Oracle
- PostgreSQL
- Sybase
- Sun ODBC-JDBC Bridge

データベース・マッピングの考慮事項

データベース・アダプタの構成時には、リレーショナル・データ構造を階層ディレクトリにマップする際に考慮する必要がある重要な側面がいくつかあります。

エントリ名の形成

エントリ名の一部として使用するすべてのデータベース・フィールド（アダプタのルートである識別名の一部分を除く）は、データベース・アダプタ経由で Oracle Virtual Directory にマッピングおよび戻される表の行に含まれている必要があります。

たとえば、作成する階層のユーザー・オブジェクトに共通名（cn）と組織単位（ou）の両方を識別名（cn=Joe User,ou=Marketing）で含める場合は、cn と ou の両方が作成中のエントリの一部であることが必要です。

純粋な LDAP では、ou は親エントリの一部であるため、本来は必要ありません。データベースは階層構造ではないため、これによってマルチレベルの識別名を生成できるようになり、大量の新規メタデータを作成および管理して階層を定義する必要もありません。

複数表への書込み

複数の表への書込みは、単一のデータベース・アダプタを介して直接行うことはできません。これは、複数の表の列を表示しているときには表示の更新を直接行えないという、データベースで発生する制限と同じです。

Oracle Virtual Directory では、Oracle Virtual Directory の結合ビュー・アダプタを使用することで、この制限を回避できます。複数のデータベース・アダプタを作成し（一般的に表ごとに 1 つ）、アダプタ間の関係を定義すると、複数の表を介して構成されたエントリへの Oracle Virtual Directory による書込みが可能になります。

複数値属性

データベースでは、一般的に表の単一の行にある単一フィールドに複数の値を使用できません。配列タイプがサポートされる例外もありますが、これらのデータ・タイプは相対的に制限される傾向にあります。一部では、カンマやパイプ（|）などのデリミタを使用してフィールド内のデータ（アカウントのフラグなど）を区切ることで、複数の値を行に挿入するということが行われています。

従来型のデータベース設計では、複数の値を持つフィールドは、正規化して追加の表にすることが求められます。データ・ウェアハウスの一部であるデータベースでは、すべてのフィールドのすべての順列を非正規化された表に配置するという、異なる方法が採られる場合があります。

Oracle Virtual Directory では、通常どちらかのモデルがサポートされます。一般的に Oracle Virtual Directory では、指定された RDN 属性を使用して複数の行がグループ化されます。Oracle Virtual Directory のスキーマにより複数の値がサポートされている場所では、Oracle Virtual Directory によってこれらの行がグループ化され、結合されたエントリが形成されます。

たとえば、グループのメンバーシップの定義に使用する表があるとします。最初の列にはグループ名が記載されています。2 番目の列には、メンバーが定義されています。

GroupName	Member
My First Group	cn=Paul Jacobs,cn=Users,dc=Oracle,dc=com
My First Group	cn=Alice Wing,cn=Users,dc=Oracle,dc=com
My First Group	cn=Jim Smith,cn=Users,dc=Oracle,dc=com
Administrators	cn=Paul Jacobs,cn=Users,dc=Oracle,dc=com
Administrators	cn=Jim Smith,cn=Users,dc=Oracle,dc=com

SQL では、各グループのメンバーを表示する SQL 問合せは次のようになります。

```
select * from grouptable group by groupName;
```

データベース・プロキシを介してプロキシ設定された場合、Oracle Virtual Directory は、上のデータを次のように戻します。

```
dn: cn=My First Group,ou=Groups,dc=Yourcompany,dc=com
objectclass: groupofuniquenames
objectclass: top
cn: My First Group
uniquemember: cn=Paul Jacobs,cn=Users,dc=Oracle,dc=com
uniquemember: cn=Alice Wing,cn=Users,dc=Oracle,dc=com
uniquemember: cn=Jim Smith,cn=Users,dc=Oracle,dc=com

dn: cn=Administrators,ou=Groups,dc=Yourcompany,dc=com
objectclass: groupofuniquenames
objectclass: top
cn: Administrators
uniquemember: cn=Paul Jacobs,cn=Users,dc=Oracle,dc=com
uniquemember: cn=Jim Smith,cn=Users,dc=Oracle,dc=com
```

上の結果では、グループ名が一意の単一値に折りたたまれ、グループのメンバーが複数値属性 uniquemember に結合されています。

検索および複数行オブジェクト

正規化および非正規化された表に対しての検索がサポートされます。必要に応じてデータベース・レベルの結合を構成する以外は、何も行う必要はありません。

検索に関して注意すべき点は、現在のデータベース・アダプタは最も一般的なユースケースに合うように、複数値属性が検索されると、検索に一致した行のみをエントリの一部として返すように設計されていることです。正規の LDAP 条件では、通常は属性のすべての値が戻されます。

例として次の検索についてみてみます。

```
ldapsearch -b "cn=administrators,ou=groups,dc=yourcompany,dc=com" -s base
"(uniquemember=Paul Jacobs,cn=Users,dc=Oracle,dc=com"
```

通常の LDAP サーバーでは、結果は次のようになるのが一般的です。

```
dn: cn=Administrators,ou=Groups,dc=Yourcompany,dc=com
objectclass: groupofuniquenames
objectclass: top
cn: Administrators
uniquemember: cn=Paul Jacobs,cn=Users,dc=Oracle,dc=com
uniquemember: cn=Jim Smith,cn=Users,dc=Oracle,dc=com
```

実際に DB プロキシによって戻される結果は次のとおりです。

```
dn: cn=Administrators,ou=Groups,dc=Yourcompany,dc=com
objectclass: groupofuniquenames
objectclass: top
cn: Administrators
uniquemember: cn=Paul Jacobs,cn=Users,dc=Oracle,dc=com
```

2 つ目の結果では、他のメンバーが欠けています。この動作は明らかに標準とは異なりますが、意図的なものであり、大規模なデータベース・エンティティでメンバーシップをグループとしてテストするパフォーマンスの高い検索を支援するために設計されています。これらのシナリオでは、顧客のほとんどの目的は、特定の人物がメンバーであるかどうかを調べることのみでした。このアダプタでは、グループ全体が送信されるかわりに共通集合のグループが戻されます。

標準の LDAP 動作が必要な場合は、アダプタにマッピングを追加してフィルタ条件を (objectclass=*) に置き換えることができます。このマッピングによって、グループ情報が戻された後に結果がテストされ、結果がフィルタと一致するかどうかが確認されます。

表への書込み

複数表オブジェクト（正規化された表）への書込みは、結合ビュー経由で実行される必要があります。結合ビューではデータベースの設計に基づいて属性がそれぞれの表に分割されます。この処理が必要となるのは、データベース設計に一般的なガイドラインがある一方で、それぞれの顧客のデータベースが異なり、結合された表同士の関係が大きく変わる可能性があるためです。

既存の重要な表を使用する顧客は、これらの表の更新制御の一部としてストアド・プロシージャも使用します。これらの表は API コールと類似しており、表が構築および呼び出される方法はそれぞれのデータベースに専用のものです。API コール自体は顧客独自のものです。Oracle では、プラグイン・システムを使用することによってストアド・プロシージャをサポートしています。

Oracle Virtual Directory では非正規化された表に直接書き込むことができます。これらの表にはそれぞれのフィールド値が格納され、エントリの RDN として使用されるフィールドに関連するエントリで使用されます。追加、変更、削除などのすべての操作がサポートされます。

変更操作では、LDAP における変更→置換の動作は、既存の属性値を削除してから新しい属性値を追加する方法で行われることに留意してください。この動作は、SQL の挿入、更新、および削除の複合セットではなく、1 つの SQL 削除および 1 つの SQL 挿入に変換されます。非正規化された表を持つデータベースには、挿入または削除、もしくはその両方によって、データ

ベース内で別のアクションがトリガーされる可能性があるという難点があります。このような場合は、変更→置換操作を処理するためのプラグインを作成する必要があります。

ほとんどの顧客は、変更に直接 SQL アクセスを使用しない、読取りのみに複数の値を使用する、または変更→追加と変更→削除のみを使用するため、このような状況に遭遇することはありません。たとえば、Oracle Virtual Directory 経由でデータベース内にグループを格納することで、大規模なグループの問題を顧客が解決するとします。グループのほとんどのメンバーシップ変更は、置換ではなく追加と削除です。

カスケード削除

前述の「表への書込み」で示した問題で、データベース・プロキシの使用時に最も注意すべきことは、カスケード削除を使用するデータベースへの直接書込みを行うデータベースの処理が、Oracle Virtual Directory によっていつ行われるかということです。カスケード削除では、1 つの表で削除を行うと、データベースによりストアド・プロシージャがアクティブ化され、それによって他の表でも削除が発生します。カスケード削除を使用した場合、前項で説明した変更→置換によって、データベース・プロキシ・アダプタと直接接している表の外側での削除がトリガーされます。これが発生するのは、既存の値を削除する SQL DELETE および新規の置換値を追加する SQL INSERT を実行する単一のトランザクションが、データベースによって送信されるためです。

データベースのトリガーが単一の値を持つ表に基づいており、複数の値を持つフィールドが他の表に正規化されている場合は、これは問題になりません。これは、変更対象のエントリに関連付けられている単一の行が表に含まれている場合の置換時に、Oracle Virtual Directory によって DELETE-INSERT の組合せではなく UPDATE が実行されるためです。

DB アダプタ・ウィザードの使用方法

新しいデータベース・アダプタ・ウィザードを使用すると、接続の定義から完全なオブジェクト・クラスのマッピングおよび属性の定義を 4 つのステップで行えます。

接続を指定したら、「Validate Connection」ボタンをクリックして次に進みます。Oracle Virtual Directory Manager は、Oracle Virtual Directory を使用してデータベースへの接続を試みます。接続が成功すると「Next>」ボタンが有効になります。「Next>」ボタンをクリックして、データベース・マッピングのステップに進みます。

「Choose Tables」ページに、データベース・スキーマのリストおよびそれらのスキーマで使用可能な表が表示されます。使用する表を選択するには、「Available Tables」リストで表の名前を選択し、「>」ボタンをクリックします。これにより、選択した表が「Used Tables」に移動します。すべての表を選択する場合は、「>>」ボタンを使用します。反対に、「Used Table」リストから表を削除する場合は、「<」または「<<」を使用します。

必要な表をすべて選択したら、「Next>」ボタンをクリックします。

「Joins」ページでは、表と表の関係または結合を定義できます。結合を定義しない場合は、単純に「Next>」ボタンをクリックして次のステップに進みます。

結合を定義するには、左側で表を選択し、対応するフィールドを選択します。次に、右側で異なる表を選択し、対応するキー・フィールドを選択します。準備が完了したら「Add」をクリックして、下にある定義済結合の表に追加します。結合を削除するには、単純に表の中で結合を選択し、「Remove」をクリックします。

「Map Attributes」ページでは、作成するオブジェクト・クラスのタイプおよび対応する属性を定義できます。起動するには、「Object Classes」リストの下にある追加ボタンをクリックして、目的のオブジェクト・クラスを入力または選択します。

「RDN」フィールドには、RDN の作成に使用する属性名を単純に入力します。

ヒント： 既存のオブジェクト・クラスを選択して「Add」ボタンをクリックすると、ネストされたオブジェクト・クラスを作成できます。これが正しく機能するには、ネストされたクラスの RDN が子オブジェクト・クラスの属性であることが必要です。場所情報が使用可能で組織単位（ou）情報の駆動にも使用できる、人々に関する表のレコードに対して親組織単位を作成することは、この一例です。

オブジェクト・クラスが追加された場合は、左側の「Object Classes」リストでそのオブジェクト・クラスを選択し、「Attributes」表の下にある「Add」をクリックします。

このステップでは、選択したオブジェクト・クラスに属性を追加し、その属性と先に選択した表との関係を定義します。任意の属性名（Oracle Virtual Directory によって自動的にそのスキーマに追加されます）を入力するか、コンボ・ボックスのセレクタを使用して既存の属性から選択できます。LDAP 属性名を入力したら、表、フィールドおよびデータベース・タイプ（オプション）を選択します。データベース・タイプは、接続したデータベースでメタデータの取得がサポートされていない場合や、Oracle Virtual Directory で異なる書式のフィールドを処理する場合に使用します。

アダプタが起動すると、Oracle Virtual Directory がデータベースに接続され、属性を現在定義されている LDAP スキーマと一致させるために、定義済のすべての LDAP 属性および対応する表と列の情報が取得されます。マップされた LDAP 属性がすでに定義されている場合は、データベース・ソースの書式からターゲット LDAP スキーマの書式へのマッピングの作成が試行されます。LDAP 属性が未定義の場合は、DB プロキシによって、一時的に属性がデータベース書式に最も綿密にマップするサーバー・スキーマに追加されます（この定義は永続 Oracle Virtual Directory スキーマ構成には追加されません）。

表 6-1 LDAP から DB への型マッピング

LDAP 型	LDAP 構文	SQL 型
Integer	1.3.6.1.4.1.1466.115.121.1.27	TINYINT
Integer	1.3.6.1.4.1.1466.115.121.1.27	SMALLINT
Integer	1.3.6.1.4.1.1466.115.121.1.27	INTEGER
Integer	1.3.6.1.4.1.1466.115.121.1.27	BIG INT
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	REAL
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	FLOAT
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	DECIMAL
Integer	1.3.6.1.4.1.1466.115.121.1.27	DOUBLE
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	NUMERIC
Boolean	1.3.6.1.4.1.1466.115.121.1.7	BIT
IA5 String （大文字と小文字が区別される文字列）	1.3.6.1.4.1.1466.115.121.1.26	CHAR
IA5 String （大文字と小文字が区別される文字列）	1.3.6.1.4.1.1466.115.121.1.26	VARCHAR
Binary	1.3.6.1.4.1.1466.115.121.1.5	BINARY
Binary	1.3.6.1.4.1.1466.115.121.1.5	VARBINARY
Binary	1.3.6.1.4.1.1466.115.121.1.5	LONGVARBINARY
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	DATE
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	TIME
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	TIME STAMP

基本構成パラメータ

Name
アダプタの一意の名前です。このアダプタを参照する他の構成フィールド、およびサーバー・ナビゲータや編集タブで使用されます。

Root
このフィールドは、このアダプタが情報を提供するルートの識別名を定義します。定義されたDN およびその下にある子エントリは、アダプタのネームスペースと呼ばれます。このフィールドに入力された値は、仮想ディレクトリのクライアントに表示される値です。この値は、カンマ区切りの識別名として指定する必要があります（たとえば ou=DB,o=Oracle, c=US または ou=DB,dc=Oracle,dc=com）。

Active
アダプタはアクティブまたは非アクティブとして構成されます。非アクティブとして構成されたアダプタは、サーバーの再起動時またはアダプタの起動の試行時には起動しません。この設定の背後にある主要な目的は、古い構成を構成から削除せずに使用可能にしておく、つまり待機させておくことにあります。
デフォルト : 有効

DB 設定

Modify Objectclass
データベースのオブジェクト・クラスは論理オブジェクトであり、マッピングにおいて表の列に直接マップされないため、オブジェクト・クラス属性を変更するとエラーが発生します。このオプションを選択すると、DB アダプタによってオブジェクト・クラス属性のすべての参照が削除されるため、クライアント・アプリケーションにエラーが送信されません。
デフォルト : Ignore Modifies

Include Objectclass Super Classes
この設定を使用すると、DB アダプタによって、オブジェクト・クラスの親クラスおよびオブジェクト・クラス属性の主要なオブジェクト・クラスのリストが表示されます。Active Directory サーバー・スキーマをエミュレートする場合は、この設定を選択解除してください。ほとんどの状況では、objectclass=xxx の問合せを親オブジェクト・クラスの値に対して実行できるように、この設定を有効にすると便利です。
デフォルト : Include

Maximum Connections
DB アダプタがデータベースに対して確立できる最大接続数を定義します。

接続設定

URL Type
データベース URL が Oracle Virtual Directory Manager で事前定義されたものか、ユーザーが指定した URL かを示します。

Database Type
事前定義済のデータベース（「URL Type」）を使用する場合は、データベース URL が計算されるように、このフィールドでデータベース型を選択します。

Database Driver
JDBC ドライバ・クラスの名前です。「URL Type」が「Custom」モードに設定されていない場合は、このフィールドは無効です。

Database URL
データベースへのアクセスに Oracle Virtual Directory で使用される URL です。「URL Type」が「Custom」に設定されている場合には、このフィールドが編集可能になり、値を指定する必要があります。そうでない場合には、このフィールドは無効になり、Oracle Virtual Directory Manager によって自動的に計算されます。

Host
DNS ホスト名またはデータベース・サーバーの IP アドレスです。

Port
データベースのポート番号です。事前定義済データベースを使用する場合は、このフィールドの値が自動的に入力されます。ただし、編集可能です。

Database Name
接続先データベースの名前です。

Database User
データベースへの接続に DB アダプタによって使用されるユーザー名です。このアカウントは、すべてのデータベース操作に使用されます。

ヒント： Microsoft SQL*Server に接続する場合は、混合認証モード用にデータベースが構成されていることを確認してください。Windows ドメインのログインは、DB アダプタではサポートされていません。

Password
データベースへのログイン用にデータベース・ユーザーに関連付けられている、プロキシのパスワードです。パスワードを設定するには、プレーン・テキスト値を入力します。起動時には、セキュリティ向上のため、サーバーによってこの値が可逆的にマスクされた値に自動的に置き換えられます。

Database Attribute Mapping
「Database Attribute Mapping」セクションは、前述の DB Attribute Mapping ウィザードによって管理されます。このウィザードをアクティブにするには、このボタンをクリックします（サーバーに接続している必要があります）。左側のウィンドウ内でオブジェクト・クラスをクリックすると、現在のマッピングを表示できます。

DB アダプタの自動および手動のスキーマ構成
DB アダプタには、自動スキーマ管理機能が含まれています。この機能では、RDBMS のメタデータ情報の問合せや、等価の LDAP スキーマの作成が、初期化中に自動で行われます。

デフォルトでは、DB アダプタの初期化時にそれぞれの定義済属性が確認されて、アダプタ構成が処理されます。それぞれの定義済属性に対して、Oracle Virtual Directory では、データベース・メタデータへの問合せを実行して、リモート SQL 型が判別されます。この型は、6-28 ページの表 6-1「LDAP から DB への型マッピング」で示したように、等価の LDAP 構文の割当てに使用されます。

NT アダプタ構成

NT アダプタは、Windows NT ドメインのユーザーのプロキシを有効にします。このアダプタの主な使用目的は、LDAP アプリケーションで NTLM 経由でユーザーを認証できるようにすることです。

> **警告：** この機能は、プロキシ設定対象のドメインに含まれる（またはプロキシ設定されたドメインと信頼を共有する）Windows サーバーでのみサポートされます。

NT アダプタは、ユーザー・アカウントを定義済ルートの下にある簡易 LDAP オブジェクトにマップすることで機能します。CORPORATE というドメインの jsmith というユーザーを認証するには、CORPORATE ドメインにプロキシを設定するように NT アダプタを定義します。ユーザー・アカウントは、次のように識別名にマップされます。

```
uid=jsmith,ou=People,{adapter root}
```

上の `{adapter root}` は、アダプタの定義済ルートを指します。

ユーザーを認証するために、次のように dn を使用してバインドが実行されます。

```
uid=jsmith,ou=people,ou=ntdomain,o=yourcompany.com
```

uid=jsmith はアダプタ `ou=ntdomain,o=yourcompany.com` にマップされるため、Oracle Virtual Directory ではこれが、NT ユーザーの CORPORATE¥jsmith を認証することと認識されます。

NT アダプタの特別構成要件

オペレーティング・システムのサポート

プロトコル要件により、NT アダプタは Win32 システムでのみサポートされます。

ライブラリ

NT ドメインと通信するための、ファイル **NTSystem.dll**、**osnt.dll**、**mfc71.dll** および **msvcr71.dll**（Oracle Virtual Directory インストールの `server/lib` ディレクトリ内）。選択すると、インストール時に Oracle Virtual Directory のインストーラによって必要な変更がすべて行われます（サーバーを再起動する必要があります）。

サービス・アカウント

Oracle Virtual Directory サービスの実行に使用されるアカウント、および Oracle Virtual Directory で NT ドメインの問合せに使用される NT User は、同じにする必要があります。SAM データベースを読み取るのに必要な管理権限を持つアカウントであることが必要です。

サービス・アカウントを設定するには、サービス・コントロール・パネルを開き、Oracle Virtual Directory サービス（一般的には VDE_8888）を選択します。右クリックして「Properties」を選択します。ダイアログ内で「Log On」タブをクリックし、「Log On as:」を「Local System Account」から目的の特定のアカウントに切り替えます。

ドメイン間プロキシ

Oracle Virtual Directory が別のドメインのアカウントにアクセスする場合は、ローカル Oracle Virtual Directory ドメインとターゲット・ドメインとの間に適切な信頼関係が必要です。また、Oracle Virtual Directory がリモート・ドメインへのアクセスに使用するアカウントは、ローカル・ドメインとリモート・ドメインの両方で、同じ名前およびパスワードであることが必要です。

基本設定

Name
アダプタの一意の名前です。このアダプタを参照する他の構成フィールド、およびサーバー・ナビゲータや編集タブで使用されます。

Root
このフィールドは、このアダプタが情報を提供するルートの識別名を定義します。定義されたDN およびその下にある子エントリは、アダプタのネームスペースと呼ばれます。このフィールドに入力された値は、仮想ディレクトリのクライアントに表示される値です。この値は、カンマ区切りの識別名として指定する必要があります（たとえば ou=Domain,o=Oracle, c=US または ou=Domain,dc=Oracle,dc=com）。

Active
アダプタはアクティブまたは非アクティブとして構成されます。非アクティブとして構成されたアダプタは、サーバーの再起動時またはアダプタの起動の試行時には起動しません。この設定の背後にある主要な目的は、古い構成を構成から削除せずに使用可能にしておく、つまり待機させておくことにあります。
デフォルト : 有効

NT 設定

NT Domain
ユーザーの認証を行う NT ドメイン名です。これは、NT ドメイン名、ドメインとサーバー（ドメイン / サーバー）、またはホスト・アドレス（例 : 127.0.0.1）のいずれかで指定できます。（デフォルト : 127.0.0.1）。プロキシを実行する Oracle Virtual Directory がドメイン内の Windows サーバーであるか、ホスト・ドメインがプロキシ設定対象ドメインと信頼関係にあることが必要になります。この点に注意してください。

User Objectclass
ユーザー・オブジェクトに対して NT アダプタによって作成されるオブジェクト・クラスです。
デフォルト : user

NT User
検索の実行に使用するドメイン・アカウントです。有効な値は、nothing（プロセス・コンテキスト）、uid（例 : Administrator）、domain/uid です。
デフォルト : Administrator

NT Password
NT User アカウントに関連付けられているパスワードです。

JoinView アダプタ構成

JoinView アダプタは、他のアダプタからの複数の LDAP エントリを結合して新しい結合エントリを形成する機能を提供する、特別なアダプタです。エントリの結合方法は、ジョイナという特別なプラグインによって制御されます。ジョイナは、エントリの結合を行うかどうか、またその時期とマージ方法に加え、必要に応じてエラーのリカバリ方法について定義する 1 つのロジックです。JoinView アダプタには、Simple、OneToMany および Shadow という 3 つの基本的なジョイナが付属しています。「Java プラグインの開発」の章で説明しているジョイナ API を使用して、独自のジョイナを記述することもできます。

JoinView アダプタは、次の原理で機能します。

- 各結合ビュー・アダプタには、プライマリ・アダプタを 1 つだけ設定する必要があります。JoinView アダプタは、ディレクトリ・ツリーの作成および検索にこのアダプタを使用します。JoinView は、プライマリ・アダプタで検出された各エントリを使用し、結合規則に従って別のアダプタ内のエントリを結合することで機能します。
- 各結合ビュー・アダプタにはゼロ以上の結合規則があり、その規則によって、プライマリ・アダプタのエントリと結合アダプタのエントリとの結合関係が指定されます。結合規則は、ジョイナ（Simple、OneToMany または Shadow）および単純結合構成情報（たとえば userprincipalname=uid などの検索条件）で構成されます。

> **ヒント：** ジョイナを設定していない JoinView を使用して、ディレクトリ・ツリーの新規部分に既存のアダプタをコピーできます。この利点は、重複アダプタを作成せずに、元のアダプタの接続プールを共有できることです。ジョイナが定義されていない場合は、結合が未定義であることを示す初期設定の警告が表示されます。

- デフォルトでは、プライマリ・アダプタを使用して認証の資格証明が処理されます。結合アダプタもバインディングに使用されます。
- 結合規則は、定義順に処理され、累積的な方法で実行されます。それぞれの結合が処理されると、作成された結合エントリが次の結合関係に使用されます。
- JoinView はネスト可能です。そのため、プライマリ・アダプタまたは結合アダプタも、別の結合ビューにすることができます。
- JoinView は繰返し使用できます。そのため、同じ JoinView に対して結合を実行できます。この再帰構造は、結合エントリが検出されなくなると終了します。無限ループに注意してください。
- 多くの場合、JoinView アダプタは、各アダプタについて検出および編集可能な属性を判別する際に、ルーティング規則（「**routing retrieveable**」および「**routing storeable**」**属性**）に依存します。
- 「**routing visibility**」を使用すると、JoinView アダプタの形成に使用されるプライマリ・アダプタと結合アダプタを非表示にすることができます。
- 戻されたそれぞれの結合アダプタ・エントリに対して、Oracle Virtual Directory によって属性値 **vdejoindn** が追加されます。この値は、統合エントリの形成に使用される結合アダプタ内のエントリを示します。

> **ヒント：** JoinView アダプタは、ロジックを制御するために、「**Routing Retrieve**」および「**Routing Store Attributes**」の設定に大きく依存します。これらのパラメータの設定方法は、「ルーティング」の章を参照してください。

基本設定

Name
アダプタの一意の名前です。このアダプタを参照する他の構成フィールド、およびサーバー・ナビゲータや編集タブで使用されます。

Root
このフィールドは、このアダプタが情報を提供するルートの識別名を定義します。定義されたDN およびその下にある子エントリは、アダプタのネームスペースと呼ばれます。このフィールドに入力された値は、仮想ディレクトリのクライアントに表示される値です。この値は、カンマ区切りの識別名として指定する必要があります（例 : ou=People,o=Oracle, c=US または ou=People,dc=Oracle,dc=com）。

注意 : JoinView アダプタのルートが、プライマリ・アダプタまたはすべての結合アダプタのルートとは異なることを確認してください。これを怠ると、予期しない結果の重複が発生します。

Active
アダプタはアクティブまたは非アクティブとして構成されます。非アクティブとして構成されたアダプタは、サーバーの再起動時またはアダプタの起動の試行時には起動しません。この設定の背後にある主要な目的は、古い構成を構成から削除せずに使用可能にしておく、つまり待機させておくことにあります。
デフォルト : 有効

設定

Primary Adapter
プライマリ・アダプタとして使用するアダプタの名前です。プライマリ・アダプタは、JoinView アダプタ内のディレクトリ構造およびエントリの存在（つまり、サブツリー検索の索引）を定義します。プライマリ・アダプタのルートの下で検出されたエントリは、結合ビューに自動的に含められます。プライマリ・アダプタ内で検出された各エントリに対して、定義済の各結合ビュー・アダプタとの結合が試行されます。（必須、デフォルトなし）

Bind Adapter
バインド処理に使用される 1 つ以上のアダプタ名のリストです。デフォルトでは、プライマリ・アダプタが使用されます。この設定を上書きして、1 つ以上の他のアダプタを含めることができます。JoinView は、ターゲット・アダプタに対して結合を実行することで結合を完了し、バインドを処理しようと試みます。バインドが正常に行われると、処理は停止し、成功がクライアントに戻されます。バインドが失敗した場合は、JoinView によって「Bind Adapter」リストにある各アダプタが試されます。バインドの失敗が戻されるのは、すべてのバインドアダプタが失敗した場合のみです。ユーザー ID が複数のディレクトリに存在する場合に、複数のディレクトリに対するパスワード検証をクライアントが行えるようにするのに役立ちます。

結合規則
結合規則は、プライマリ・アダプタと 1 つ以上のアダプタの間の結合関係で構成されます。結合は、Oracle Virtual Directory に付属のいずれかの結合クラスまたはいずれかの結合規則です。結合関係を編集する場合は、結合規則に示されている結合条件を入力します。たとえば、Simple および OneToMany では、(joinattr=primaryattr) という結合条件が要求されます。joinattr はターゲットの結合アダプタの属性で、プライマリ属性はプライマリ・アダプタの属性です。

Simple ジョイナ

Simple ジョイナは、単純な属性マッチング関係に基づいて、プライマリ・アダプタのエントリと結合アダプタのエントリの間の単純結合を実行します。複数の結合候補を検索する処理で、複数の候補が検出された場合は、最初の候補のみが使用されます。

Simple ジョイナは、プライマリ・アダプタと結合アダプタの両方に対する単純検索および LDAP 変更の操作をサポートします。ただし、LDAP 入力操作の Add、Delete および Rename は無視されます（プライマリ・アダプタのみに対して処理されます）。

OneToMany ジョイナ

OneToMany ジョイナは、単純な属性マッチング関係に基づいて、プライマリ・アダプタのエントリと結合アダプタのエントリの間の単純結合を実行します。OneToMany ジョイナは、Simple ジョイナと非常に似ていますが、複数の候補が検出された場合に、すべての結合候補が結合に使用されます。

OneToMany ジョイナは、プライマリ・アダプタと結合アダプタの両方に対する単純検索および LDAP 変更の操作をサポートします。ただし、LDAP 入力操作の Add、Delete および Rename は無視されます（プライマリ・アダプタのみに対して処理されます）。

Shadow ジョイナ

Shadow ジョイナは、Oracle Virtual Directory または別のディレクトリでローカルに属性を格納しながら、アプリケーションと企業ディレクトリの統合を可能にします。アプリケーションは、すべての属性を格納するディレクトリと通信しているとみなしますが、Oracle Virtual Directory は、アプリケーション固有のデータを密かに代替シャドウ・ディレクトリに格納します。

Shadow ジョイナは、別のディレクトリをプライマリ・アダプタに指定することで、これを行います。プライマリ・アダプタの各エントリについて、Shadow ジョイナにより、特定の属性を保持できるローカルのシャドウ・エントリが管理されます。ローカル・エントリの DN は、主要エントリの識別名（DN）の **MD5 hash** を使用し、それをローカルのルート（例 : o=localstore）と組み合せて、自動的に構成されます。

LDAP 変更イベントが発生するたびに、Shadow ジョイナは「Routing Store-able Attributes」パラメータを参照して、シャドウ結合アダプタに格納する属性があるかどうかを確認します。対象となる属性が存在する場合、Shadow ジョイナは、主要エントリの **MD5 hash** を使用してローカル・エントリの特定を試みます。検出されると、適切な LDAP 変更操作がローカルで実行されます。ローカル・エントリが検出されない場合、Shadow ジョイナは、主キーを使用してエントリを特定するための 2 次検索（主要 DN が変更されている場合）を試行します。検出されない場合は、新しいエントリが自動的に作成されます。

ローカル・エントリには 1 つ以上の属性のみが含まれ、有効なオブジェクト・クラスが形成されないため、Shadow ジョイナは、オブジェクト・クラス **extensibleobject** および **vdeshadowobject** を使用して、ローカル・エントリを格納します。extensibleobject オブジェクト・クラスをサポートするすべての LDAP ディレクトリが使用可能ですが、通常は、標準アダプタを使用してローカル・エントリが格納されます。**extensibleobject** オブジェクト・クラスは、単純に、なんらかの属性を含む可能性があるオブジェクトを定義するもので、すべての LDAPv3 サーバー製品でサポートされていないことに注意してください（IETF RFC 2251 ではオプション項目です）。

Shadow ジョイナは、すべての LDAP 操作をサポートし、必要に応じてローカル・エントリを自動的に追加します。

Shadow ジョイナの起動方法

Shadow アダプタの設定は、少々複雑です。次に手順を示します。

企業ディレクトリを指すプライマリ・アダプタ（LDAP、DB など）を設定します。

ローカル・ストア・ディレクトリを設定します。これは、LSA アダプタ、または LDAP アダプタを介して対外的にプロキシ設定された LDAP ディレクトリになります。プロキシ設定されたディレクトリは、**extensibleobject** オブジェクト・クラスをサポートし、**vdeshadowobject** オ

ブジェクト・クラスが定義されていることが必要です（Oracle Virtual Directory のスキーマ・エディタで vdeshadowobject を確認できます）。

ローカル・ストア・ディレクトリに対して「Routing Store-able Attributes」を一時的に空白にします。

ローカル・ストア・ディレクトリで、ローカル・ストアのベース・エントリを追加します（例 : o=localstore）。

ローカルで書き込まれる属性のみがリストに表示されるように、ローカル・ストア・アダプタの「Routing Storeable Attributes」を変更します。結合規則で使用する一意のキー属性を含め、属性 vdeprimaryref を含めます。LDAP クライアントに対して表示されないようにする場合は、ローカル・ストアの「Routing Visibility」を「Internal」に設定します。

「Routing Retrievable」および「Routing Storeable」を変更し、プライマリ・アダプタに書き込まれる属性を制御します。Oracle Virtual Directory でプライマリ・アダプタへの変更の書込みが一切行われないようにするには、「Routing Storeable Attributes」を**「_never」**に設定します。

手順 1 のプライマリ・アダプタを指定して結合ビュー・アダプタを追加します。結合規則を Shadow ジョイナとして手順 2 のアダプタを指すように指定します。条件として、名前変更イベントでのレコード検出に使用できる一意キー（例 : uid）を指定します。この条件は、プライマリ・アダプタの一意キーの属性名であることが必要です。他のジョイナとは異なり、この条件は等価条件ではありません。

アダプタを再起動して操作をテストします。

スキーマ構成

サーバー設定の項では、「Schema Files」パラメータについて、Oracle Virtual Directory で使用できるスキーマ・ファイルを指定すると説明しました。この項では、スキーマ・エディタ、およびスキーマの確認と変更の方法について説明します。

属性定義

属性エディタでは、Oracle Virtual Directory 内で使用する属性を指定できます。

Name

定義対象の属性の名前です。

Superior

親属性の名前です。空白になる場合もあります。

Description

任意で指定する属性の説明です。

Syntax

属性値の書式です。
注意 : Oracle Virtual Directory では、親構文の値のみが使用されます。サポートされている値は、「Directory String」（大 / 小文字の区別なし）、「Binary」、「Distinguished Name」、「IA5String」（大 / 小文字の区別あり）および「Integer」です。

Bounds

属性の長さです。0 または空白は無制限を表します。（Oracle Virtual Directory では強制されません）

OID（Object Identifier）
ICANNS によって指定される一意のオブジェクト識別子です。登録されていない場合は、任意の一意の値を使用できます。なんらかの属性名を登録することをお薦めします。

Usage
属性の使用方法を示します。「User」、「Application」、「Operational」のいずれかになります。

Single Valued
属性が一度に単一の値のみを保持する場合は、このオプションを選択します。

Equality
等価性の一致規則 OID です。（Oracle Virtual Directory では使用されません）

Ordering
順序付け検索の一致規則 OID です。（Oracle Virtual Directory では使用されません）

Substring
部分列検索の一致規則 OID です。（Oracle Virtual Directory では使用されません）

オブジェクト・クラス定義

Name
オブジェクト・クラスの名前です。

Superior
親オブジェクト・クラスです。親クラスでない場合、クラスは上位から派生している必要があります。

Description
オブジェクト・クラスの説明です。（強制には使用されません）

OID（Object Identifier）
オブジェクト・クラスの一意のオブジェクト識別子です。一意の文字列を使用できます（一意の OID を使用してすべてのカスタム・オブジェクト・クラスおよび属性を登録することをお薦めします）。

Type
オブジェクトのタイプ（値は「Structural」、「Auxiliary」または「Abstract」）です。「Abstract」は、オブジェクトによって直接使用されるのではなく、別のクラスから継承されるオブジェクト・クラスを表します。「Structural」は、エントリを形成できるオブジェクト・クラスです。「Auxiliary」オブジェクト・クラスは、（構造化オブジェクト・クラスに基づいて）既存のオブジェクトに追加の属性を追加するために使用します。（Oracle Virtual Directory では強制されません）

Obsolete
オブジェクト・クラスに不要のマークを付ける管理設定です。（Oracle Virtual Directory では強制されません）

Required Attributes
オブジェクト・クラスに含める必要がある属性のリストです。

Optional Attributes
任意で指定できる属性のリストです。

7

ルーティング

この章では、Oracle Virtual Directory のルーティングについて説明します。このルーティングは、LDAP 操作に対して選択するアダプタを決定する処理です。

概要

ルーティングは、LDAP 操作に対して選択するアダプタを Oracle Virtual Directory が決定するために行われる処理です。一般的に、ルーティングには、選択されるアダプタ数を制限し、リクエストされたクライアント・データを含み、現在の LDAP 操作と関連するアダプタのみに絞り込む働きがあります。ルーティングはタイプに関係なくすべてのアダプタに適用される機能です。

従来型のディレクトリ・サーバーには、1 つ以上の定義されたデータベースがあり、それぞれのデータベースがディレクトリ・ツリーのネームスペースの一部に対応しています。通常のディレクトリ・サーバーでは、データベースの選択は、ネームスペースの比較のみに基づいて決定されます。仮想ディレクトリでは、複数のアダプタで同じネームスペースを共有させることができるため、選択はより複雑で、より詳細に制御できます。ルーティングを使用すると、LDAP フィルタ、属性および DN パターンなど、より多くのトランザクション情報を参照してアダプタの選択を制御できます。

Oracle Virtual Directory のルーティングには、パフォーマンスの最適化およびセキュリティ・フィルタ処理の追加レベルの提供という、2 つの追加機能が含まれます。特定のトランザクションに使用するアダプタの数を減らすことで、複数のアダプタでの操作を処理する仮想ディレクトリ・サーバーでのオーバーヘッドの発生が大幅に縮小されます。その結果、接尾辞の比較のみに基づいてアダプタを選択する無条件ルーティングと比較した場合に、LDAP 操作に対するサーバーの全体的なパフォーマンスが向上します。その他の状況では、ルーティングのフィルタ処理は、基本的な DN ネームスペースではなく条件に基づくアダプタの選択および選択解除に使用されます。このフィルタ処理には、DN パターン一致、属性のフィルタ処理、問合せフィルタのフィルタ処理およびビュー選択などがあります。

ルーティングの概要

次の各項では、多様なルーティングの概念について詳細に説明します。

アダプタの可視性

アダプタを定義して、**「Yes」**、**「No」**、**「Internal」**の 3 つの可視性モードのいずれかで使用できます。可視アダプタ（デフォルト）は、`namingcontexts` 属性の一部としてサーバーのルート・エントリにルートが公開されるアダプタです。可視性を「No」に設定すると、アダプタは `namingcontexts` 属性に含まれませんが、外部の LDAP クライアントでは使用できます。複数のアダプタが同時に機能して単一のディレクトリ・ツリー・ブランチを形成している場合に便利です。論理ツリー全体は暗黙的に利用できる上、子アダプタの公開はアプリケーションにとって冗長で複雑なため、`namingcontexts` に含まれる親およびすべての子アダプタを公開せずに、ルート・アダプタのみを公開することもできます。

「Internal」のアダプタは、Oracle Virtual Directory 内部で実行されるプラグインおよびジョイナでのみ使用できるアダプタです。外部の LDAP クライアントでは使用できません。結合ビュー・アダプタで使用するために構成されたアダプタは、これに当てはまります。外部の仮想ディレクトリで情報を 2 度公開するかわりに、主要アダプタおよびジョイナ・アダプタを「Internal」としてマーク付けすると、これらのアダプタで定義された情報に JoinView アダプタのみがアクセスできるようになります。

サーバーの重大性

「Critical」では、**「True」**、**「False」**、**「Partial」**の 3 つの設定がサポートされています。**「Critical」**プロパティを**「True」**（デフォルト）に設定すると、アダプタが（たとえば操作上のエラーによって）結果を戻すのに失敗した場合に、その他のアダプタでデータが検出されたかどうかに関係なく、Oracle Virtual Directory によって、クライアントに「DSA Unavailable」というエラーが返されます。

一方、重大性を**「False」**に設定すると、現在のアダプタで操作の実行に失敗しても、Oracle Virtual Directory では、全体的な結果には重大ではないと認識されます。重大でないアダプタで操作上のエラーが発生した場合は、単純に、LDAP 検索結果全体からその結果が省略されます。

また、状況によっては、アプリケーションを介して、結果の一部が取得されたことをユーザーに通知できます。これを行うには、アダプタの重大性を**「Partial」**に設定します。エラーが発生すると、Oracle Virtual Directory によって「Admin Limit Exceeded」というエラーが戻されます。これは、実際の内容とは異なるエラーですが、この設定の目的は、クライアント・アプリケーションのロジックによって、表示されている結果が全体の一部にすぎないことが示されることです。

LDAP 操作でのアダプタ処理順序の決定

他のアダプタより先に特定のアダプタが処理されるように、Oracle Virtual Directory を強制することが必要になる場合があります。よくある例は、複数のアダプタでネームスペースが重複している場合です。このような状況は、既存のディレクトリをオンライン状態にしたまま新しいディレクトリを使用可能にする場合に発生します。この場合は、より新しい（またはより重要な）アダプタの**「Routing Priority」**プロパティを、より高い優先度（つまり、デフォルトの 50 より小さく、なおかつネームスペースが重複するアダプタより小さい数値）に設定します。

優先度の数値が同じアダプタは、構成ファイルでの定義順に検索されます。競合（同じ DN をサポートする 2 つのアダプタ）が発生した場合、Oracle Virtual Directory での検索操作では、優先度の数値が最も小さく、構成で先に定義されたアダプタが使用されます。変更操作では、エントリからツリーを上位にさかのぼって先に一致したアダプタ内のエントリのみが処理されます。

ヒント： 正確を期すために、複数のアダプタが選択される状況での判断基準として、「Routing Include」、「Routing Exclude」および「DN Pattern」の設定を使用することをお薦めします。

ルーティング・レベル

「Routing Levels」プロパティを LDAP 検索と併用すると、検索ベースとなる、アダプタ・ルートの下のレベル数を指定できます。たとえば、値 0 では、検索ベースがアダプタのルートと同じになり、値 1 では、検索ベースがアダプタのルートまたは 1 つ下のレベルになります。

このプロパティは、複数のアダプタが複雑にネストされた状況を混在させる場合に便利です。たとえば、ルート・アダプタは、仮想化ツリーのすべての問合せに対して選択される可能性がありますが、その他のアダプタが関連データを含むツリーの一部を指すように設定されている可能性があるため、望ましくありません。ルート・アダプタを、実際にルート・エントリを調査する問合せ以外のすべての問合せから除外するには（つまりサーバーのパフォーマンスを向上させるには）、レベルを 0 に設定します。

バインド操作に使用する資格証明の決定

管理者は、「**Routing Include Binds From**」プロパティおよび「**Routing Exclude Binds From**」プロパティを使用して、互いの資格証明を共有するアダプタを指定できます。アダプタは、これに基づいて、ユーザー資格証明とアダプタのプロキシ・アカウントのどちらを操作で渡すかを決定できます。この例として、Microsoft Active Directory フォレスト内の 2 つの異なるドメイン・コントローラをプロキシ設定する、異なる LDAP アダプタについて考えてみます。Oracle Virtual Directory では、1 つのドメインからのユーザー資格証明は、別のドメインの一部として認識されません。それでも、両方のドメインが同じフォレストに属すため、2 番目のドメインでは実際に別のドメインからの資格証明が受け入れられることは明白です。管理者は、「Include Binds」パラメータおよび「Exclude Binds」パラメータを使用して、このような状況の処理方法を Oracle Virtual Directory に対して指定できます。

ユーザー資格証明を渡せるかどうかを決定する際に、Oracle Virtual Directory では 1）提供された資格証明が現在のアダプタのルートの下にあるか、2）ユーザー資格証明が「**Include Binds From**」プロパティにリストされているアダプタの下にマップされているか、3）ユーザー資格証明が「**Exclude Binds From**」プロパティにリストされている除外アダプタの下にマップされているか、という 3 つの条件が考慮されます。2 つ目の条件が満たされている場合、資格証明を渡すことができるように、アダプタは他のアダプタに対し、資格証明をマップすることを要求します。

アダプタのルートが `ou=admin,o=depts, dc=octet,dc=com` である場合の例を次に示します。ユーザー資格証明は、次のいずれかに当てはまります。

1. `ou=admin,o=depts,dc=octet,dc=com` のネームスペース内にマップされます。
2. `ou=admin,o=depts,dc=octet, dc=com` のネームスペース内にマップされません（たとえば、資格証明の DN が `ou=sales,o=depts, dc=octet,dc=com` で終わるなど）。

ケース A

ユーザー資格証明が `ou=admin,o=depts,dc=octet, dc=com` で終わる場合：

「**Exclude Binds From**」プロパティが空でない場合は、ユーザー資格証明をチェックして、いずれかの除外アダプタの子であるかどうかを確認する必要があります。この条件に当てはまる場合は、（クライアント資格証明を渡すかわりに）プロキシ資格証明を使用する必要があります。ユーザー資格証明が除外アダプタに属さない場合は、ユーザー資格証明が現在のアダプタを通過します。

この状況が最も発生するのは、2 つの LDAP プロキシが定義されており、2 番目のプロキシが最初のプロキシ（親プロキシ）の子である場合です。子アダプタの一部である資格証明は、誤って親アダプタの一部でもあるとみなされる可能性があります。「Exclude Binds」プロパティを使用すると、子アダプタの資格証明が不正に親アダプタに渡されるという問題を修正するのに役立ちます。この設定を使用することで、特定の子 DN が親アダプタの資格証明セットにマップされないことを Oracle Virtual Directory に認識させることができます。

ケース B

ユーザー資格証明が `ou=admin,o=depts, dc=octet,dc=com` とは異なるルートで終わる場合：

「**Include Binds From**」が空ではないが、共有として定義されているアダプタがある場合は、ユーザー資格証明をチェックして、共有アダプタのいずれかにマップされているかどうか確認する必要があります。この条件に当てはまる場合は、共有アダプタによって資格証明がマップされ、元のアダプタに戻されます。その後、元のアダプタで、共有アダプタによってマップされた資格証明を渡すことが可能になります。

資格証明が現在のアダプタまたはいずれかの共有アダプタにマップされない場合は、提供された資格証明を渡すかわりに、プロキシ資格証明を使用する必要があります。

Oracle Virtual Directory が複数の Microsoft Active Directory ドメインにプロキシを設定するのは、この一例です。ユーザー資格証明のルートが異なる場合でも、すべてのプロキシの行き先は同じフォレストであるため、1 つのドメイン・コントローラによって別のドメイン・コントローラからの DN が認証されます。この場合は、どちらのアダプタからの資格証明でも、両方のアダプタで同様に共有できます。たとえば、Domain A アダプタによって Domain A がプロ

キシ設定され、Domain B アダプタによって Domain B がプロキシ設定されます。Domain A および Domain B は同じフォレストに存在します。

したがって、次のように設定できます。

Domain A アダプタ：
Include Binds From: Domain A, Domain B

Domain B アダプタ：
Include Binds From: Domain A, Domain B

これは、Domain A アダプタおよび Domain B アダプタがそれぞれの資格証明を相互に渡すことができることを意味します。

属性の取得および格納の制限

ターゲット・ディレクトリにあるアダプタで取得または格納できる属性を制御する必要がある場合は、**「Routing Retrieve」**プロパティおよび**「Routing Store」**プロパティを使用します。このような制限を行った場合、指定された属性のみがプロキシ設定されたサーバーからリクエスト可能になり（検索の場合）、特定の属性（およびその値）のみがプロキシ設定されたサーバーに送信可能になるため（格納の場合）、パフォーマンスや（場合によっては）セキュリティの向上に貢献します。

この設定は、セキュリティおよびフィルタ処理の目的で、任意のアダプタに対して使用できます。JoinView アダプタの使用時に属性を制御する目的にも使用できます。JoinView では、2 つ以上のアダプタからのエントリが結合されるため、どの属性がどの関連アダプタを受信し、どの属性をどの関連アダプタに送信するのかを制御する必要があります。これを構成するには、構成済の結合ビュー内の各アダプタで「Routing Retrieve」パラメータおよび「Routing Store」パラメータを設定します。

クライアント検索でのエントリの包含および除外

属性に適用される**「Routing Retrieve」**や**「Routing Store」**とは異なり、**「Routing Include」**および**「Routing Exclude」**は、クライアントによって指定された LDAP 検索フィルタ、つまりフィルタのフィルタに適用されます。

クライアント検索フィルタによって「Routing Include」プロパティの論理的要件が満たされると、そのアダプタが選択され、検索に使用するアダプタ・セットに含められます。同様に、「Routing Exclude」の場合は、論理的要件が満たされると、そのアダプタが、クライアント検索に使用するアダプタ・セットから選択解除されます。「Routing Include」と「Routing Exclude」の両方を同時に使用することで、より複雑な条件セットを作成して、クライアント検索操作で使用されるアダプタを管理できます。

たとえば、ファイアウォールとしてデプロイされた Oracle Virtual Directory LDAP プロキシを介する、特定の種類の検索を許可するとします。特定の検索のみを許可するには、次のようなフィルタを使用します。

```
(|(mail=*@myorg.com)(uid=*@myorg.com)(sn=*)(givenname=*)(cn=*))
```

上のフィルタでは、mail、uid、sn、givenname または cn 以外の条件の検索がブロックされます。許可されるのは、これらの条件の 1 つ以上を伴う検索のみです。たとえば、(cn=Jim Smith) は受け入れられますが、(uid=smith@octetstring.com) は myorg.com で終わらないため受け入れられません。

また、包含（include）と除外（exclude）のフィルタには、クライアント検索のスコープ（scope）要件が含まれる場合があります。使用できるスコープ・タグは、sub、one および base です。include プロパティのデフォルト・スコープは sub、exclude プロパティのデフォルト・スコープは one です。

ほとんどのアダプタ構成では、単純な検索条件が使用されますが、より複雑な例を参考にすると、ロジックの適用方法を理解するのに役立ちます。そこで、次に示す包含 / 除外フィルタの例をみてください。

クライアント検索コマンド：

```
$ ldapsearch -b dc=octet,dc=com -s sub "(|(sn=user2)(cn=user2b))"
```

ルーティング・フィルタ：
(&(|(uid=*)(cn=*))(sn=*))

このコマンドでは、クライアント検索フィルタに 1 つの sn 属性、および uid または cn のどちから一方の条件が含まれる場合に、一致となります。上の例では、他の条件を無視して、フィルタが**「Routing Include」**に割り当てられている場合はアダプタが選択され、**「Routing Exclude」**に割り当てられている場合は選択解除されます。これは、クライアント・フィルタに sn および cn の条件が含まれており、フィルタのロジックが満たされるためです。

DN パターン一致

DN パターン一致は、アダプタ間で同じアダプタ・ルートを共有する場合や、どのエントリがどのアダプタに属すかを判断する方法が必要な場合に、最も使用されます。DN パターン一致を使用すると、プロキシ設定された 2 つのソース間で発生するネーミングの相違を利用できます。たとえば、大規模な開発では、アルファベットに基づいてエントリを分けることもできます。パターン一致では、アルファベットの範囲を選択し、範囲の一致に基づいて Oracle Virtual Directory でアダプタを選択できます。たとえば、名前を 3 つの範囲に分割する場合は、ユーザーの ID の先頭文字を基準に、a から j を 1 つのディレクトリ、k から r を別のディレクトリ、s から z を最後のディレクトリ部分のように指定できます。

もう 1 つの役立つシナリオとしてあげられるのは、Active Directory ユーザーと Open LDAP やその他のディレクトリなどの外部ディレクトリに含まれるユーザーとのフェデレーションです。Open LDAP のユーザーに uid 属性に基づく RDN（相対識別名）が設定されており、Active Directory に cn 属性に基づくユーザー・エントリがある場合は、RDN タイプに基づいてアダプタを選択する正規表現を設定できます。

Active Directory アダプタの DN 一致は次のようになります。

```
(.*)cn=[a-z0-9]*$
```

Open LDAP の場合は次のようになります。

```
(.*)uid=[a-z0-9]*$
```

この方法を使用することで、Oracle Virtual Directory では、既存のソースの相違を利用して、重複するアダプタを効果的に管理できます。

ルーティングの管理

ルーティングは Oracle Virtual Directory Manager で特定のアダプタに関連するプロパティを設定して、複数の方法で制御できます。これらのプロパティは、ルーティング操作の多様な側面に対応しており、いずれも相互排他的ではありません。

選択内容

Priority

優先度に基づいて、このアダプタが他のアダプタに対して相対的に処理されます。最も高い優先度は 1 で、100 が最も低くなります。優先度は、他のルーティング・パラメータがすべて処理されてから、最後のセレクタとして使用されます。その他の点で同等の 2 つの候補がある場合、優先度が高い方のアダプタが先に処理されます。

Filters to Include

選択したアダプタに対して許可されている、LDAP 検索フィルタのフィルタ（つまり、フィルタのフィルタ）です。たとえば、myorg.com の電子メールおよび同じドメインからのユーザー ID に関する検索のみを許可する場合は、次のようなフィルタを使用できます。

(|(mail=*@myorg.com)(samaccountname=*@myorg.com))#one

これは、`mail` または `samaccountname` を伴い `@myorg.com` で終わる検索条件のみが、このアダプタによって処理されることを意味します。

このフィールドの書式は、標準の LDAP 検索フィルタの後にスコープ条件（#base、#one または #sub）が続きます。このスコープは、フィルタを適用するスコープ・レベルを示します。たとえば、上の包含フィルタは 1 レベルまたはサブツリーの検索のみに適用されます。ベース検索はフィルタ処理されません。包含フィルタのデフォルト・スコープは #sub で、サブツリー全体に関する問合せのみが除外されます。すべてのスコープにフィルタを適用するには、スコープを #base に設定します。これにより、フィルタはベース、1 レベル、およびサブツリーの検索に適用されます。

Filters to Exclude

LDAP 検索フィルタの除外フィルタ（つまり、フィルタの除外フィルタ）です。管理者はこれを使用して、アダプタと無関係の検索やパフォーマンスの低い検索、または不必要な検索を除外できます。包含フィルタと同様に、このフィルタは、標準の LDAP フィルタの後にスコープ条件を付加したものです。除外フィルタのデフォルト・スコープは #one で、特定の検索がブロックされます。すべてのスコープにフィルタを適用するには、スコープを #base に設定します。サブツリー検索のみにフィルタを適用するには、スコープを #sub に設定します。

DN Matching

アダプタ内の DN の書式を示す正規表現です。この正規表現は、アダプタ・ルートの下の DN 部分に適用されます。たとえば、アダプタのルートが ou=People,o=MyBigOrg.com で、次のレベルにある RDN の先頭文字が A から J までのエントリのみを許可する場合は、次のような式を指定します。

m/^(.*)uid=[a-j][a-z0-9]*$/

この式は、DN に uid= の条件が含まれ、その後に A から J の文字があり、その後に英数字（いくつでも可）が続く必要があることを示しています。$ 記号は文字列の終わりを示します。この場合、文字列の終わりのカンマを除外することは許可されないため、uid= がこのアダプタ内の DN の最後の構成要素であることが必要です。UID 値は A から J で始まる必要があるため、この条件と一致する UID のみが受け入れられます。最後に、正規表現の ^(.*) 部分は、なんらかの文字（種類と数は特定しない）が文字列の最初（^ の部分）と特定の値 uid= の間に含まれることを示しています。

ヒント： DN は大文字と小文字が区別されないため、正規表現の一致は、大文字と小文字を区別せずに実行されます。

ヒント： 一致表現の m/ および末尾の / の部分はオプションです。

Levels

使用する複数のアダプタの一部がその他のアダプタの子である場合は、親アダプタを構成して子アダプタのネームスペース内で発生する問合せが親アダプタに送信されないようにする必要が出てきます。これは、LDAP 操作の DN が、通常のネームスペース選択を介して子アダプタと親アダプタの両方に関係する場合に発生します。親アダプタの深さを設定すると、Oracle Virtual Directory で子トランザクションから親アダプタを排除できます。

たとえば、ローカル・ストア・アダプタが o=Oracle.com と定義されているとします。これは、ou=Partner1, o=Oracle.com および ou=Partner2, o=Oracle.com などの LDAP プロキシ・アダプタの共通の親に使用できます。この場合、o=Oracle.com は単純に子アダプタのプレースホルダになります。アダプタのエントリは 1 つのみであるため、検索ベースが o=Oracle.com である操作に対してのみの問合せを行う必要があります。検索ベースが ou=Partner1, o=Oracle.com の場合には、このアダプタを検索する必要はありません。この場合、「Routing Levels」の値は 0 が適切です。

属性フロー

属性フローは、特定のアダプタへの属性のフロー、およびそのアダプタからの属性のフローを制御します。属性フローは、セキュリティ上の理由から、認証されていないクライアントからの情報のリクエストや、そのようなクライアントに情報が戻されるのを防ぐために使用します。また、アダプタが結合ビューの一部である場合に属性フローをフィルタ処理する際にも使用できます。

> **注意：** アクセス制御とは異なり、属性フロー規則では**隠れた強制**が行われます。つまり、単純にリクエストがフィルタ処理されるのみで、クライアントにはエラーは戻されません。セキュリティ性の高い設定を使用すると、クライアント側では、特定の属性の表示が許可されているかどうかさえ知ることができません。

結合ビューの場合は、複数のアダプタが同じ仮想結合エントリをもたらす可能性があるため、どの属性がどのアダプタにフローするかを制御するのに属性フローが役立ちます。

Retrievable Attributes

アダプタから取得できる属性の明示的なリストです。リストが空の場合は、すべての属性が取得可能です。指定されている場合は、リストに含まれている属性のみがプロキシ設定されたディレクトリからリクエストされます。

Storable Attributes

現在のアダプタに書き込める属性のリストです。リストが空の場合は、すべての属性が格納可能です（下の**「Unstorable Attributes」**が定義されている場合を除きます）。指定されている場合は、リストに含まれている特定の値のみが格納可能です。

> **注意：** アダプタを読取り専用にするには、格納可能な属性のリストに値「_never」を入力します。属性名に「_」を使用することはできないため、この条件が満たされることはありません。結果的に、アダプタは読取り専用になります。

UnStorable Attributes

変更可能な属性よりも変更不可の属性を示す方が簡単な場合には、このリストを使用します。通常、格納可能属性のリストと格納不可属性のリストは、両方ではなく、どちらか一方を指定します。

一般設定

Visibility

外部クライアントによるアダプタへの問合せが可能かどうか、また、ルート・エントリの下にあるサーバーの namingcontexts 属性で公開されるかどうかを制御します。有効な値は次のとおりです。

- Yes: 外部から使用でき、namingcontexts に表示されます。
- No: 外部から使用できますが表示されません。
- Internal: 結合ビュー・アダプタまたはプラグインでのみ使用できます。

Views

ビューは、識別名または IP アドレス、もしくはその両方の定義済リストです。アダプタは、1 つ以上の定義済ビューに表示されます。定義すると、ビューのメンバーであるユーザーは、同じビューに割り当てられているアダプタからの情報のみを参照できます。このパラメータがアダプタの割当て先ビューを定義するのに対し、**「View Membership」**は、全体的なサーバー設定（Engine/Server/Views）で定義されます。

Bind Support

アダプタが LDAP バインド操作を処理できるかどうかを示します。アダプタがバインド機能をサポートしない場合、Oracle Virtual Directory は、指定されている DN に対応するエントリからの属性 userPassword の取得を試行し、ローカル・パスワードの比較操作を実行します。これは、LDAP プロキシ・アダプタで「Pass Credentials」を「Never」に設定することと同じです。バインド操作をサポートするかどうかが不明なカスタム・アダプタを定義する際に、この設定を使用します。

Criticality

ホストのエラーによってアダプタでの検索操作が失敗した場合の Oracle Virtual Directory の反応を、次の 3 つのいずれかに指定できます。

- **true:** アダプタが結果を戻すのに失敗した場合（たとえば、すべてのリモート・ホストが失敗した場合）、Oracle Virtual Directory では検索操作全体が失敗します。
- **partial:** アダプタが失敗した場合、Oracle Virtual Directory では結果の一部が戻されますが、結果セット制限超過のエラーが示されます。これは正確なエラーではありませんが、アプリケーションがこれを使用することで、すべての結果が戻されたわけではなく、一部の結果が表示されることをクライアントに示すことができます。
- **false:** アダプタが失敗した場合、Oracle Virtual Directory によってその他のアダプタからの結果の一部が戻されますが、エラーは示されません。

Include Binds From

同じディレクトリ・サービスが複数のアダプタでプロキシ設定されている場合、適切なアダプタを選択して、別のアダプタからの資格証明を共有できることを示すことができます。たとえば、同じ Active Directory Forest の複数のドメイン・コントローラがプロキシ設定されている場合、「Include Binds From」値を選択して、プロキシ設定された資格証明をアダプタが共有できることを示すことができます。

Exclude Binds From

アダプタが別のアダプタ（親）の子アダプタである場合は、DN ルートが一致するために、子アダプタからの資格証明が親アダプタで有効であると Oracle Virtual Directory によってみなされる問題が発生します。この問題の発生を阻止するために、重複するアダプタからのマッピングを除外できます。

8

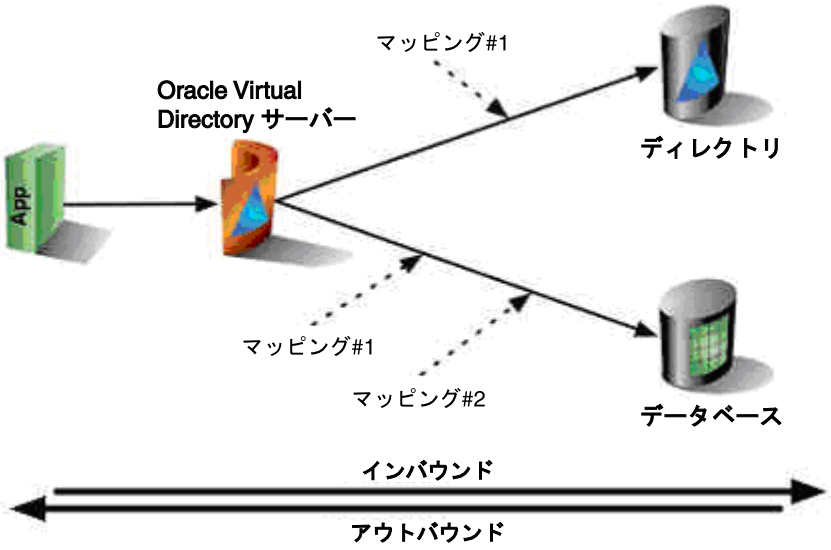
マッピング・システム

この章では、Python に基づく Oracle Virtual Directory のマッピング・システムについて説明します。

概要

Oracle Virtual Directory には、Python スクリプト言語に基づく、双方向のマッピング・システムが含まれています。**マッパー**は、特別な Python スクリプト（ファイル形式は .mpy）で、Oracle Virtual Directory 内でのインバウンドおよびアウトバウンドのトランザクション・データ・フローを処理します。マッピング・スクリプトは、リクエストがデータソースに到達するためにシステムに入る際にリクエストを調整し、クライアントに戻る途中でレスポンスを変換できます。コンパイルされてサーバーに配布されたマッピング・スクリプトをマッパーといいます。

マッパーは、プラグイン・チェーンと呼ばれる Oracle Virtual Directory のプラグイン・システム内で稼働するように構成されます。プラグインと同様、マッパーは、必要に応じてグローバルに実行することも、アダプタ・レベルで実行することもできます。下の図は、1 つのマッピングを複数のアダプタで実行し、別のマッピングを特定のアダプタのみで実行する場合の一般的なシナリオを示しています。



それぞれのマッピングには、インバウンドとアウトバウンドのフローがあります。つまり、マッピングでは、リクエストを受信するときに一方向の変換処理を行い、リクエスト元のアプリケーションに結果が戻されるときには、その変換を逆方向にすることができます。サーバーがこの意図を予測することは通常不可能なため、このようにプログラム上で方向を反転させることは重要です。

Python について

Oracle のマッパーでは、LDAP データ操作への対応を目的とする Oracle 提供の追加機能を使用して、Python 言語を最大限に利用できます。

Python に関する詳細は、次の URL の情報を参照してください。

- http://www.python.org - Python の全般情報
- http://docs.python.org/tut/tut.html - Python の概要およびチュートリアル

マッピングの管理

マッピングの作成、編集、デプロイおよび構成の方法は、「Oracle Virtual Directory Manager」の章の「マッピング・スクリプト」を参照してください。

シナリオ例

ここでは、マッパーについての知識を深めるのに役立つシナリオを 2 つ採り上げます。最初のシナリオでは、マッピングを使用して **givenname** と **sn** から **cn**（共通名）が構成されます。このマッパーは、**cn** 値が存在しないデータベース・アダプタで使用できます。2 つ目のシナリオでは、Microsoft Active Directory が、**inetorg** オブジェクト・クラスを使用して構造化されたディレクトリのようになります。

例 : CN 属性の構成

この例では、指定された名前と姓から共通名を作成する方法を説明します。このような状況は、データベース・アダプタを使用して、データベースに格納されているユーザー・データへの LDAP インタフェースを提供する際に発生します。LDAP ディレクトリには通常 **cn** が格納されますが、データベースでは名前と姓のみが格納される傾向があります。検索の実行時には、共通名へのフィルタ処理が非常に複雑になります。たとえば、フィルタ (cn=Marc Boorshtein) は、(&(givenName=Marc)(sn=Boorshtein)) と読み取る必要があります。これをより簡単に行うために、マッピングでは次のようなフィルタ操作ツールが提供されます。

```
From string import split

def parceCN(val): ❷
      return split(val,' ',2)

def inbound():
      #map the "cn" filters
      if operation == 'get': ❸
          if haveAttribute('cn'): ❹
               addAttribute('givenName')
               addAttribute('sn')

      cnFilters = findFilters('cn') ❺
      for filter in cnFilters:
           target,op,val = filter.contents
           givenNameVal, snVal = parceCN(val)
           givenNameFilter = createFilter('givenName',op,givenNameVal)
           snFilter = createFilter('sn',op,snVal)
      filter.contents =
createAndFilter([givenNameFilter,snFilter]) ❻

def outbound():
      #outbound stuff
      addAttributeValue('cn',getAttributeValue('givenName') + ' ' +
getAttributeValue('sn'))❶
```

このマッパーの要求事項はかなり単純です。ここでの目的は、アダプタからデータが取得されたら、**givenname** と **sn** を組み合せて **cn** を作成することです。ただし、インバウンド側では、**cn** を **givenname** と **sn** に分割する必要があります。属性リクエスト・リストに cn が含まれている場合は、リストが変更され、**givenname** および **sn** が含められます。最後に、インバウンド操作が検索操作の場合は、検索フィルタを確認して **cn** を適切に変換する必要があります。

アウトバウンド処理

まず、単純な方のマッピング処理、**outbound()** 関数について説明します。アウトバウンド関数は、アダプタからクライアントへフローするすべてのトランザクションを処理します。ここでは、他の 2 つの値から cn を作成することが目的です。そこで、**addAttributeValue** 関数（❶）を使用して、givenname、スペース 1 個、および sn 値を組み合せて新しい cn 値を作成します。**getAttributeValue** 関数を使用して既存の値がどのように取得されているかに注目してください。この関数では、クライアントに戻される現在のエントリから、特定の属性が取得されます。

インバウンド処理

次に、より複雑な **inbound()** 関数について説明します。ここでは、この関数を使用してすべての CN を変換し、givenname 属性と sn 属性に分割することを目的としています。検索に関しては、cn の検索フィルタを givenname と sn に対応する結合フィルタに変換する必要があります。新しい関数 **parceCN()** を作成してこのタスクを実行します。

最初の行では、Python の string モジュールから split 関数がインポートされます。**parseCN()** という Python 関数が定義されます（❷）。この関数では、共通名（cn）が取得され、スペースの検出に基づいて名前と姓に分割されます。この処理は、実際にはより複雑になりますが（たとえばミドル・ネームを考慮するなど）、この例の目的として、ここでは単純な事例を採り上げています。

次に、目的の **inbound()** 関数を定義します。inbound 関数はすべての LDAP 操作に対応していますが、ここで必要としているのは検索操作です。したがって、インバウンドの後の最初の行は、値 operation をテストする if ブロックになります（❸）。この変数操作には、add、bind、delete、get、modify、rename のいずれかが含まれます。

操作が get の場合、マッパーは、検索リクエストの属性リクエスト・リスト内に cn があるかどうかを判別して続行します（❹）。cn を作成するには、givenname と sn を組み合せる以外に方法がないため、**addAttribute** 関数を使用して、検索リストに givenname と sn を追加する必要があります。

cn のフィルタ・リクエストを処理するために、マッパーは、ターゲットが cn 属性であるすべてのフィルタ要素をフェッチします（❺）。マッパーは、フィルタごとに要素を解析し、parseCN をコールして対応する givenname 値と sn 値を計算し、新しい givenName フィルタおよび sn フィルタを作成します。最後に、cn を伴うフィルタ条件を、givenName および sn を含む結合フィルタに置き換えます（❻）。

例 : Active Directory スキーマのマッピング

一般的に、アプリケーションでは、**inetorgperson** および **groupofuniquenames** のスキーマ・オブジェクトを使用して LDAP ディレクトリを使用することが求められます。ただし、多くの組織では、**user** オブジェクトと **group** オブジェクトのみをサポートする Microsoft Active Directory が使用されています。この相互操作性の隔たりを埋めるために、マッピングで **inetorgperson** または **groupofuniquenames** を使用すると、Active Directory のスキーマを **inetorg** スタイルのスキーマのようにすることができます。

これらの 2 システム間の変換は、実際にはかなり複雑で、満たす必要がある要件が多数あります。具体的には次のとおりです。

- **属性名の双方向マッピング。**たとえば、uniquemember = member、uid = samaccountname などです。
- **オブジェクト・クラス名の変換。**基本のオブジェクト・クラス名を変更するだけでなく、Microsoft AD では補助オブジェクト・クラスが使用されないことも考慮する必要があります。たとえば、オブジェクト・クラス値の interorgperson、organizationalperson または person は、まとめて単に user にする必要があります。
- **特別な属性値の追加。**Microsoft では、groupType や userAccountControl などの追加のオブジェクト・タイプ・コードを使用することが要求されます。操作に応じて、特別なタグをリクエストに追加する必要があります。
- **RDN 変換。**通常、Microsoft では、cn はユーザー・アカウントの相対識別名として使用されます。多くのアプリケーションでは、uid を使用することが求められます。

```
def inbound():
      #first rename the attributes

rename({'uniqueMember':'member','uid':'samaccountname','userpassword':'unicodepwd','ntg
rouptype':'grouptype'})❶

      #map nessasary object class values ❷
      if haveAttributeValue('objectclass','groupifuniquenames'):
          removeAttributeValue('objectclass','groupofuniquenames')
          addAttributeValue('objectclass','group')

      if haveAttributeValue('objectclass','organizationalPerson'):
          removeAttributeValue('objectclass','organizationalPerson')
          addAttributeValue('objectclass','user')

      if haveAttributeValue('objectclass','inetOrgPerson'):
          removeAttributeValue('objectclass','inetOrgPerson')
          addAttributeValue('objectclass','user')

      #when adding an entry, certain values need to be added
      if operation == 'add': ❸
          if haveAttributeValue('objectClass','group'):
              addAttributeValue('groupType','-2147483646')
              if not haveAttribute('samaccountname'):
              copy('cn','samaccountname')

          if haveAttributeValue('objectClass','user'):
              addAttributeValue('userAccountControl','66048')

      #collapse aux classes❹
      removeAttributeValue('objectClass','person')
      removeAttributeValue('objectClass','organizationalPerson')

      #set the rdn
      setRDN('samaccountname','cn')❺

def outbound():
      #first rename the attributes


rename({'member':'uniqueMember','samaccountname':'uid','unicodepwd':'userpassword','gro
uptype':'ntgrouptype'})

      #map nessasary object class values
      if haveAttributeValue('objectclass','group'):
          removeAttributeValue('objectclass','group')
          addAttributeValue('objectclass','groupofuniquenames')

      if haveAttributeValue('objectclass','user'):
          removeAttributeValue('objectclass','user')
          addAttributeValue('objectclass','organizationalPerson')
```

上のマッピングでは、**inbound()** および **outbound()** の 2 つの関数が定義されています。インバウンドの最初の行では、すべての **inetorg** 属性が Active Directory 属性に名前変更されます（❶）。すべての操作に対して rename 関数がコールされます。たとえば、操作が検索の場合、リクエストされたすべての属性およびフィルタ内のすべての属性が名前変更されます。操作が追加または変更の場合は、影響するすべての属性が名前変更されます。

次のブロックでは、inetOrg のオブジェクト・クラスが InetAD のオブジェクト・クラスに置き換えられます（❷）。条件文を使用して、実行するアクションを指定できることに注目してください。

3 番目のブロックでは、操作が追加（add）かどうかがチェックされ（❸）、追加であれば、Active Directory で必要になる特定の属性情報が追加されます。

次に、すべての補助オブジェクト・クラスが削除されます（❹）。これは、Active Directory では、追加操作の際に補助オブジェクト・クラスを直接指定できないためです。

最後に RDN が uid から cn に変更されます（❺）。このコードでは、samaccountname が cn に変換されます。これは、uid がすでに samaccountname に変換されていたためです（❶）。また、rdn を uid から cn に変更するだけでなく、（変更や検索などで）cn が指定されていない場合は、cn の検出にも対応します。inbound 関数が終了すると、変更リクエストや検索リクエストが Active Directory で受け入れられるようになります。

Active Directory からリクエストが戻されると、outbound() 関数が実行されます。この関数は、inbound 関数を逆にします。つまり、最初にすべての適用可能な属性を名前変更し、オブジェクト・クラス名をマッピングして、最後にすべての結果の rdn を変更します。小規模のスクリプトを使用することで、inetOrg アプリケーションで Active Directory をシームレスに使用できます。

マッピングでは、リクエストとレスポンスを操作するための単純で柔軟な手段が提供されます。マッピングを介して、アプリケーションで要求される形式にデータを変換または成型できます。

マッピング機能

マッパーは、Python に基づいており、Python 言語で使用できるすべての機能またはサブルーチンを使用できます。Python 以外にも、次のライブラリ関数が提供されます。

注意： マッピングを指定するメソッドの場合、値のリストを {'uniqueMember':'member','uid':'samaccountname',[…]} という書式で指定できることを意味します。これは、本質的に、1 つ以上のマップされた値の配列です。異なる名前のペア関係に特定のメソッドを複数回使用する場合は（例として上のスクリプト例の rename を参照）、この構文がサポートされているメソッドでこれを使用します。この構文は、省略表現として優れているだけでなく、パフォーマンスの向上にも役立ちます。

メソッド

appendAttribute(source,destination)

操作： `add`、`modify`、`get`、`entry`

appendAttribute 関数は、source 属性の値を destination 属性に追加します。ソース属性は同じ場所に留まります。この関数は、検索フィルタに影響します。

```
appendAttribute('sn','givenName')
```

add/entry:
```
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
実行後：
dn: cn=User
objectClass: person
cn: User
givenName: User
givenName: name
sn: name
```

modify:
```
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
```

givenName: First
実行後 :
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
givenName: First
givenName: Last

get: (&(givenName=first)(sn=last))
実行後 :
(&(|(sn=last)(givenName=last))(givenName=first))

copyAttribute(source,destination)

操作 : `add`、`modify`、`get`、`entry`

copyAttribute 関数は、source 属性の属性値を destination 属性にコピーします。destination 属性がすでに存在する場合は、属性が上書きされます。

```
copyAttribute('sn','givenName')
```

add/entry: dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
実行後 :
dn: cn=User
objectClass: person
cn: User
givenName: User
givenName: name
sn: name

modify: dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
givenName: First
実行後 :
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
givenName: First
givenName: Last

get: (&(givenName=first)(sn=last))
実行後 :
(|(sn=last)(givenName=last))

renameAttribute(source,destination)

操作 : `add、modify、get、entry`

source 属性を destination 属性に名前変更します。destination 属性がすでに存在する場合は、属性が上書きされます。source 属性が存在せず、destination 属性が存在する場合は、destination 属性が削除されます。

```
renameAttribute('sn','givenName')
```

add/entry:
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
実行後 :
dn: cn=User
objectClass: person
cn: User
givenName: name

modify:
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
givenName: First
実行後 :
dn: cn=User
changetype: modify
add: givenName
givenName: Last

get:
(&(givenName=first)(sn=last))
実行後 :
(givenName=last)

removeAttribute(attribute)

操作 : `add、modify、get、entry`

指定された属性が削除され、その値がリストに戻されます。属性がエントリの一部である場合は、値が戻されます。値が changelist の一部である場合は、EntryChange オブジェクトが戻されます。

```
removeAttribute('sn')
```

add/entry:
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
実行後 :
dn: cn=User
objectClass: person
cn: User
givenName: User

modify:
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName

givenName: First
実行後 :
dn: cn=User
changetype: modify
add: givenName
givenName: First
givenName: Last

get: (&(givenName=first)(sn=last))
実行後 :
(givenName=last)

revalueAttribute(attribute,currentValue,newValue)

操作 : add、modify、entry、get

```
revalueAttribute('sn','name','newname')
```

add/entry: dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
実行後 :
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: newname

modify: dn: cn=User
changetype: modify
add: sn
sn: name
-
add: givenName
givenName: First
実行後 :
dn: cn=User
changetype: modify
add: sn
sn: newname
-
add: givenName
givenName: First
givenName: Last

get: (&(givenName=first)(sn=name))
実行後 :
(&(givenName=last)(sn=newname))

mapSyntax(value,newSyntax)

mapSyntax(attribute,newSyntax)

操作 : add、modify、entry

構文値を新しい構文にマップするか、属性を新しい構文にマップします。最初の引数が構文オブジェクトである場合は、newSyntax で指定された構文のインスタンスが関数によって戻されます。最初の引数が属性の名前である場合は、その属性のすべてのインスタンスが新しい構文にマップされます。有効な構文は、DirectoryString、IA5String、BinarySyntax、BinarySyntax です。

splitValue(newNames,currentName,parseFunction,index,remove)

splitValue(newNames,currentName,parseFunction,index)

splitValue(newNames,currentName,parseFunction,remove)

splitValue(newNames,currentName,parseFunction)

splitValue(newNames,currentName,,index,,remove)

splitValue(newNames,currentName,index)

splitValue(newNames,currentName,remove)

splitValue(newNames,currentName)

操作 : add、modify、entry

```
splitValue(['givenName','sn','cn',1)
```

add/entry:
dn: uid=User
objectClass: person
cn: First Last
uid: User
実行後 :
dn: uid=User
objectClass: person
givenName: First
sn: Last
uid: User

modify:
dn: uid=User
changeType: modify
replace: cn
cn: First1 Last1
実行後 :
dn: uid=User
changeType: modify
replace: givenName
givenName: First1
-
replace: sn
sn: :Last1

get:
(cn=First Last)
実行後 :
(&(givenName=First)(sn=last))

addAttributeValue(name,value)

操作 : `add、modify、entry`

名前属性に値を追加するか、この属性が存在しない場合は属性を作成します。modify 操作で属性が存在しない場合は、変更項目 Add が作成されます。

```
addAttributeValue('myattrib','myval')

addAttributeValue('noattrib','hasvalue')
```

add/entry:
dn: uid=User
objectClass: person
cn: First Last
uid: User
myattrib: noval
実行後 :
dn: uid=User
objectClass: person
givenName: First
sn: Last
uid: User
myattrib: noval
myattrib: myval
noattrib: hasValue

modify:
dn: uid=User
changeType: modify
delete: myattrib
myattrib: someval
実行後 :
dn: uid=User
changeType: modify
delete: myattrib
myattrib: someval
myattrib: myval
-
changetype: add
add: noattrib
noattrib: hasValue

haveAttribute(attributeName)

haveAttribute(attributeName,fetchFromServer)

操作 : `add、modify、entry、get`

指定された属性が存在する場合は、1 または 0（true または false）を戻します。fetchFromServer が 1 の場合は、サーバーからエントリがフェッチされます。

haveAttributeValue(attributeName,attributeValue)

haveAttribute(attributeName, ,attributeValue,fetchFromServer)

操作 : `add、modify、entry、get`

指定された属性および関連付けられている値が存在する場合は、1 または 0（true または false）を戻します。fetchFromServer が 1 の場合は、比較のためにサーバーからエントリがフェッチされます。

removeAttributeValue(attributeName,attributeValue)

操作 : `add、modify、get、entry`

属性値を削除して、値が削除された場合は true を戻します。

```
removeAttributeValue('myattribute','myvalue')
```

add/entry: dn: cn=user
objectClass: person
cn: user
myattribute: myvalue
myattribute: myvalue2
実行後 :
dn: cn=user
objectClass: person
cn: user
myattribute: myvalue2

modify: dn: cn=User
changetype: modify
replace: myattribute
myattribute: myvalue
-
add: sn
sn: last
実行後 :
dn: cn=User
changetype: modify
add: sn
sn: last

get: (&(sn=last)(myattribute=myvalue))
実行後 :
(sn=last)

setRDN(oldRDNAttribute,newRDNAttribute)

操作 : `add、modify、delete、bind、rename、entry`

現在の名前の RDN（**get** のベース）を古い RDN 属性から新しい RDN 属性に変更します。

```
setRDN('cn','uid')
```

add/entry: dn: cn=user
objectClass: inetOrgPerson
uid: userid
cn: user
実行後 :
dn: uid=userid
cn: user
objectClass: inetOrgPerson

modify: dn: cn=user
changetype: modify
add: sn
sn: last
実行後 :
dn: uid=userid
changetype: modify
add: sn
sn: last

bind/get/delete: dn: cn=user
実行後 :
dn: uid=userid

addReturnAttribute(attributeName)

操作 : get

検索時に戻り属性のリストに属性を追加します。

findFilters(attributeName)

操作 : get

指定された属性を伴うすべてのフィルタのリストを戻します。

createfilter(target,operation,value)

操作 : get

target をテストされる属性、operation を使用できるコンパレータの 1 つ、および value をフィルタ処理対象の値として、新しいフィルタ・オブジェクトを作成します。

createAndFilter(filters)

操作 : get

フィルタのリストから and フィルタを作成します。

createOrFilter(filters)

操作 : get

フィルタのリストから or フィルタを作成します。

getAttributeValue(attributeName)

操作 : add、entry

指定された属性の最初の値を戻します。

getAttributeValues(entry,attributeName)

操作 : すべて

提供されたエントリから指定された属性の値を取得します。

createEntryChange(type,attribute,value)

操作 : modify

新しい EntryChange オブジェクトを作成して戻します。

addEntryChange(entryChange)

操作 : modify

エントリ変更のリストにエントリの変更を追加します。

getByName(dn)

操作 : すべて

指定されたエントリを戻します。

convertBase(attributeName,oldBase,newBase)

操作 : `add、modify、entry、get`

指定された属性の値について、oldBase を newBase に置き換えます。

データ・オブジェクト

データ・オブジェクトは、Oracle Virtual Directory によって Python 環境での使用が可能になった変数です。これらの変数は、Oracle Virtual Directory データ構造のハンドルの取得、および様々なオブジェクトとステータス項目の解釈に使用します。

operation

現在の操作です。
有効な値 : add、modify、delete、rename、get、entry、bind

vsi

VSI のハンドルを取得します。

attributes

戻すことが要求されている属性です。
操作 : get

base

現在の検索ベースです。
操作 : get

target, op, val = filter.contents and filter.contents = *newfilter*

フィルタを戻し、タプルの書式（target, operation, value）で設定します。

boolean filter.isAnd

フィルタが AND フィルタの場合に、TRUE を戻します。

boolean filter.isOr

フィルタが OR フィルタの場合に、TRUE を戻します。

boolean filter.isNot

フィルタが NOT フィルタの場合に、TRUE を戻します。

changeEntries

modify 操作に関する一連の変更です。
操作 : modify

creds

ユーザーの現在の資格証明（DN）です。
操作 : すべて

entry

追加するエントリまたは検索から戻されるエントリです。
操作 : get、add

filter

現在の検索フィルタです。
操作 : get

name

追加、バインド、変更または削除されるエントリです。
操作 : add、bind、delete、modify

request

値の取得 : val = request([String name])

値の格納 : request(['myname'])='myvalue'

指定された現在のリクエスト情報のオブジェクト属性を戻し、設定します。このオブジェクトは、特定のトランザクションの間存在する異なるマッパーまたはプラグイン間で、任意の情報を渡すためのメソッドとして使用されます。たとえば、インバウンド操作中に特定の情報を格納して、後でアウトバウンド・リクエストでの処理に使用できます。

results

エラーが発生した場合に、結果コードを戻して設定します。
操作 : Add、Delete、Modify

scope

現在の検索範囲を 0、1 または 2 で示します（0 はベース、1 は 1 レベル、2 はサブツリー）。
操作 : get

typesOnly

サーバーが型のみを戻して値を戻さないかどうかを示します。
操作 : get

9

Java プラグインの開発

この章では、Oracle Virtual Directory の LDAP 操作を処理するための、カスタム・プラグインの作成方法を説明します。

概要

Oracle Virtual Directory には、Oracle Virtual Directory の LDAP 操作を処理および操作するカスタム・プラグインの作成機能があります。プラグインは、すべてのリクエストを把握し処理するグローバル・レベル、または特定のアダプタに対するリクエストのみを把握し処理するアダプタ・レベルに配置できます。また、特定の操作やネームスペースに対して実行されるようにプラグインを構成することもできます。

各プラグインには、次の実装ポイントがあります。

実装	説明
構成	プラグインの構成データ。構成のカスタム部分は、初期化パラメータの名前 / 値ペアで構成されています。
起動 / 停止	init(HashMap params,String name) および destroy() メソッドは、プラグインの初期化および初期化解除時に呼び出されます。
可用性	available(Chain chain, DirectoryString base) メソッドは、プラグインを起動するかどうかを決定するために、プラグインの実行前に呼び出されます。
操作	呼び出される様々な操作メソッド。

カスタム・プラグインの作成方法を知るために、Bad Password Count プラグインを作成することで、前述の実装ポイントを 1 つずつ説明します。このプラグインは、バインド操作が失敗したか成功したかを検出します。操作が成功した場合は件数が消去され、バインドが失敗した場合は件数が増加します。また、このプラグインを使用すると、不正なパスワード件数をディレクトリの外から変更することができなくなります。このプラグインの完全なソースは、New Oracle Virtual Directory Plug-in ウィザード・テンプレートとして Oracle Virtual Directory Manager に含まれています。

チェーン・システム

Oracle Virtual Directory プラグインは、Java Servlet 2.3 フィルタ・モデルに基づく実装に準拠しています。このモデルでは、操作が続行される場合、前操作、後操作および決定の処理に単一のメソッドが使用されます。複数のプラグインを結合して、プラグインの**チェーン**が形成されます。次に、Bad Password Count プラグインが、ディレクトリに追加されるエントリに不正なパスワード件数属性を追加するかどうかを決定する例を示します。

たとえば、add メソッドが呼び出された場合は、リクエストを操作することができます。これにより、渡された属性やその値を操作すること（ActiveDirectory を標準の LDAP ディレクトリとしてマスクしている場合は、オブジェクト・クラス値 inetOrgPerson の user への変更など）や、データの記憶域を（カスタム・アダプタの）ディレクトリ以外のシステムまたはデータベース・システムに処理することが可能になります。別のプラグインを経由して仮想ディレクトリでリクエストをさらに処理できるようにするには、chain.nextAdd メソッドを呼び出します。多くのプラグイン・メソッドには、対応する chain.next<XXX> メソッドがあります。プラグインによるリクエストの処理をそれ以上許可しない場合は、chain.next<XXX> コールを除外します。

もう 1 つの重要な点は、ターゲット・アダプタのタイプは問題ではないということです。LDAP プロキシでも、DB プロキシでも、標準のアダプタでもかまいません。プラグインにとっては、いずれも単なるアダプタです。

プラグイン実装タイプ

プラグインの作成前に、com.octetstring.vde.chain.Plugin インタフェースを実装するか、com.octetstring.vde.chain.BasePlugin クラスを拡張するかを選択する必要があります。BasePlugin クラスを使用すると、プラグイン開発者は、プラグインによって処理される操作用のメソッドのみを実装できます。例では、このクラスを拡張して実装を簡略化しています。

プラグインの構成、初期化および破棄

最初の実装ポイントは構成です。プラグインは、Oracle Virtual Directory 構成システムによって提供される、一連の単純な名前 / 値ペアを使用して構成されます。これらのペアは、プラグインの init メソッドに対する param 引数を介してプラグイン開発者に提供されます。例として示すプラグインには、次の構成オプションがあります。

構成オプション	説明
countAttribute	不正なパスワード件数を保存するすべてのユーザー・エントリに追加される属性。
addOnCreate	ユーザーの作成時にプラグインがこの属性を追加する場合に true になるブール値
objectClassForAdd	属性が追加されるユーザーを表すオブジェクト・クラス。
ignoreOnModify	countAttribute の変更リクエストが無視される場合に true になるブール値。

これらの構成オプションは、2 番目の実装ポイントであるライフ・サイクル・メソッドで取得されます。init メソッドはサーバー起動時にプラグインを初期化する際に呼び出され、destroy メソッドはプラグインを停止する際に呼び出されます。次にサンプルの init メソッドを示します。

```
/**
* Passes initialization information to the Plug-in
 *
* @param initParams
*            Hashmap of key/value pairs specified in initial config
* @param name
*            The name specified in the config for this Plug-in
 */
public void init(PluginInit initParams, String name) throws ChainException {
      //the countAttribute parameter is required
      if (!initParams.containsKey(BadPasswordCount.CONFIG_COUNT_ATTRIBUTE)) {
           throw new ChainException(name + ": The "
               + BadPasswordCount.CONFIG_COUNT_ATTRIBUTE
               + " attribute is required");
      }

      this.countAttribute = new DirectoryString(initParams
           .get(BadPasswordCount.CONFIG_COUNT_ATTRIBUTE));
      this.attribType = SchemaChecker.getInstance().getAttributeType(
           this.countAttribute);
      //determine if add on create
      this.addOnCreate = initParams
           .containsKey(BadPasswordCount.CONFIG_ADD_ON_CREATE)
           && initParams.get(BadPasswordCount.CONFIG_ADD_ON_CREATE)
           .equalsIgnoreCase("true");

      if (this.addOnCreate) {
           if (this.addOnCreate
                && !initParams
               .containsKey(BadPasswordCount.CONFIG_OBJECTCLASS_FOR_ADD)) {
           throw new ChainException(name
               + ": When adding count attribute, the parameter "
```

```
                + BadPasswordCount.CONFIG_OBJECTCLASS_FOR_ADD
                + " is required");
            }

            String[] objectClasses = initParams
                .getVals(BadPasswordCount.CONFIG_OBJECTCLASS_FOR_ADD);

            this.objectClasses = new HashSet();

            for (int i = 0, m = objectClasses.length; i < m; i++) {
                this.objectClasses.add(new
DirectoryString(objectClasses[i]));
            }

        }else{
            this.addOnCreate = false;
        }

        logger.info("Adding on create : " + this.addOnCreate);

        //determine if the modify operation should be ignored
        this.ignoreModify = initParams
            .containsKey(BadPasswordCount.CONFIG_IGNORE_MODIFY)
            && initParams.get(BadPasswordCount.CONFIG_IGNORE_MODIFY)
            .equalsIgnoreCase("true");
}
```

前述のメソッドでは、プラグインを設定するための初期化パラメータが確認されています。構成情報が不十分な場合は、プラグインにより例外がスローされます（サーバーによる操作目的での使用が可能になるようプラグインを構成できない原因になります）。

接続の解放またはサービスの停止が必要な場合以外は、destroy メソッドを実装する必要はありません。

プラグインの可用性

3 番目の実装ポイントは `available` メソッドです。このメソッドは、特定の LDAP 操作に対する各プラグインの呼出しが可能になる前に呼び出されます。true が戻されると、プラグインが実行されます。次に示す例では、`available` メソッドにより `Request` オブジェクトの `ignoreOnModify` オプションの有無が確認されています（前述の初期化の項を参照してください）。オプションが定義されている場合、プラグインはスキップされます。同様に、`addonCreate` オプションが false に設定されている場合も、プラグインはスキップされます。

```
/**
* Determines if a plugin is available for the current chain
 *
* @param chain
* @param base
* @return True or False if available for a particular chain & base
 */
public boolean available(Chain chain, DirectoryString base) {

      if (chain.getOperationType() == Chain.ADD_OP && !this.addOnCreate) {
          return false;
      }else if(chain.getOperationType() == Chain.MOD_OP
                && this.ignoreOnModify) {
          return false;
      }else{
          return true;
      }
}
```

available メソッドが true を戻すと、リクエストの操作部分が実行されます。

プラグインの操作の実装

4 番目および最後の実装ポイントは、操作の実装です。例として、次のバインド操作のコード実装を示します。

```
/**
* Moves through the "bind" operation's chain
 *
* @param chain
*            the current chain
* @param dn
*            The DN for the user
* @param password
*            The user's password
* @param result
*            The result of the bind
 */
public void bind(Chain chain, Credentials creds, DirectoryString dn,
          BinarySyntax password, Bool result) throws DirectoryException,
          ChainException {

     // Pre-event processing

// calls the next plug-in in the chain (or comment out if a handler)
     try {
          chain.nextBind(creds, dn, password, result);
     }catch (DirectoryException e){
          throw e;
     }

     // Post-event processing
     if (result.booleanValue()) {
          // success, reset count
          setPasswordCount(chain, creds, dn, 0);
     }else{
          Vector searchAttributes = new Vector();
          searchAttributes.add(this.countAttribute);

          ChainVector results = new ChainVector();
          chain.getVSI().get(chain.getRequest(), creds, dn,
              new Int8((byte) 0), ParseFilter.parse("(objectClass=*)"),
              new Bool(false), searchAttributes, results);

          if (results.size() > 0) {
              EntrySet es = (EntrySet) results.get(0);
              Entry entry = es.getNext();
              Vector values = entry.get(this.countAttribute);
              Syntax value = (Syntax) values.get(0);
              IntegerSyntax is = new IntegerSyntax(value.getValue());
              setPasswordCount(chain, creds, dn,
              ((int) is.getLongValue()) + 1);
          }else{
              setPasswordCount(chain, creds, dn, 1);
          }
     }
}

/**
* @param chain
* @param creds
* @param dn
* @param result
* @throws DirectoryException
```

```
* @throws ChainException
 */
private void setPasswordCount(Chain chain, Credentials creds,
          DirectoryString dn, int count) throws DirectoryException,
          ChainException {

      Vector values = new Vector();
      values.add(new IntegerSyntax(count));
      EntryChange modify = new EntryChange(EntryChange.MOD_REPLACE,
              this.countAttribute, values);
      Vector changes = new Vector();
      changes.add(modify);
      chain.getVSI().modify(chain.getRequest(), creds, dn, changes);

}
```

このメソッドでは、パスワード失敗件数が（パスワード・ポリシーという形で）ディレクトリ内に維持されている例が示されています。メソッドでは、操作の前処理も実行されておらず、バインド操作の引継ぎも試行されていないことに注意してください。プラグインの bind メソッドが chain.nextBind メソッドをすぐに呼び出し、独自のロジックで続行する前にバインドが完了するまで待機します。バインドが完了（コントロールが chain.nextBind から戻されるなど）すると、バインドが成功したかどうかが確認されます。バインドが成功した場合には、失敗件数属性がゼロに設定されます。バインドが失敗した場合には、現行の失敗件数が取得され、増加した値が設定されます。

bind メソッドは、仮想サービス・インタフェース（VSI）を使用してバインディング・ユーザーのレコードを変更します。このインタフェースは Oracle Virtual Directory 全体で使用でき、プラグインがグローバルにデプロイされているかアダプタのコンテキスト内にデプロイされているかにかかわらず、ディレクトリ情報への一貫したアクセス方法を提供します。これは VSI が、現行のプラグインの後にチェーン内の次のプラグインを開始することにより、常に Oracle Virtual Directory をコールすることで実現されます。たとえば、プラグインの前にマッパーが存在し、後にキャッシュが存在する場合、VSI へのコールはキャッシュのみを経由します。

プラグインはバインド失敗件数の維持を論理的に担っているため、プラグインの modify メソッドを実装する必要があります。このため、LDAP クライアントが件数の変更を試行してもブロックされます。次の例では、プラグインの modify メソッドは、modify 変更リストに count 属性が含まれている場合に例外をスローするために実装されています。

```
/**
* Moves through the "modify" operation's chain
 *
* @param chain
*           The current chain
* @param creds
*           The currnet user's credentials
* @param name
*           The name of the object being modified
* @param changeEntries
*           The group of EntryChange Objects
 */
public void modify(Chain chain, Credentials creds, DirectoryString name,
          Vector changeEntries) throws DirectoryException, ChainException {

      Iterator it = changeEntries.iterator();
      while (it.hasNext()) {
          EntryChange ec = (EntryChange) it.next();
          if (ec.getAttr().equals(this.countAttribute)) {
             throw new
              DirectoryException(LDAPResult.CONSTRAINT_VIOLATION
              .intValue(), "Can not modify password count attribute");
          }
      }

      chain.nextModify(creds, name, changeEntries);
```

```
}
```

ステータス・コードおよびメッセージとともに DirectoryException がスローされています。この例外が別のプラグインによって捕捉されない場合、例外はメッセージとコードの両方でクライアントに戻されます。ignoreOnModify が構成されているかどうかの判断は available メソッドに委任してあるため、この例では確認しません。設定されている場合、前述のプラグインの modify メソッドは呼び出されていません。

これまでの例で説明されていない概念は、検索プロセスです。検索には、実装されるメソッドが 1 つではなく 3 つあるため、その他の操作とは異なります。1 つ目のメソッドは get で、その他の操作メソッドと同様の役割を果します。このメソッドを使用すると、プラグイン開発者は検索リクエストを前処理し、戻り値を後処理できます。戻される各 Entry を処理できる一方、メモリー使用率の点から見ると非常に非効率的です。get メソッドの結果を処理することは、クライアントに戻す前に、すべての結果をメモリーに取得することを意味します。このため、効率的な方法で属性を変更または追加でき、クライアントに戻される各 Entry に対して実行される postSearchEntry メソッドが用意されています。3 つ目は、検索操作が完了したことをマークする postSearchComplete メソッドです。

postSearchEntry 処理を効率的に使用するために、Oracle Virtual Directory では、EntrySet と呼ばれる特別なクラスを使用して結果セットの処理が行われます。get メソッドにより結果が戻されると、1 つ以上の EntrySet を含む Vector が戻され処理が実行されます。

EntrySet の作成

Oracle Virtual Directory では、ディレクトリの各オブジェクトは com.Oracle.vde.Entry オブジェクトで表されます。各エントリには、オブジェクトの名前と属性値を持つ属性が含まれています。すべてのエントリ・オブジェクトは、com.Oracle.vde.EntrySet インタフェースの実装を使用して Oracle Virtual Directory で処理されます。エントリ・セットでは、特定のデータ・ソースから戻されたすべてのエントリが保存または処理されます。通常の Oracle Virtual Directory 処理では、検索リクエスト中に呼び出された各アダプタにより、Oracle Virtual Directory から戻される結果のリストに独自の EntrySet 実装が追加されます。また、プラグインにより、結果の Vector 配列に追加の EntrySet オブジェクトが挿入される場合もあります。検索リクエストを実行するためにすべてのアダプタへの問合せが完了すると、各 EntrySet はクライアントに送信されるエントリに関して調査されます。

すべてのアダプタで EntrySet 実装が作成されますが、プラグインでも EntrySet インタフェースのインスタンスが作成され、それを使用して検索リクエスト中にクライアントにエントリが戻されます。

拡張可能なエントリ・セット（簡易結果）

プラグインによる EntrySet の作成方法は 2 つあります。最も簡単な方法は、`com.Oracle.vde.backend.extensible.ExtensibleEntrySet` クラスを使用して、Entry オブジェクトの `java.util.Vector` に基づく `EntrySet` を作成する方法です。

1. 新規の java.util.Vector 配列を作成します。
2. Vector にすべての Entry オブジェクトを追加します。
3. コンストラクタに前述の Vector を渡す ExtensibleEntrySet の新規インスタンスを作成します。

次に、銘柄記号に基づいて株価を取得するために Web サービスを使用する例を示します。このプラグインは、カスタム・アダプタの概念を実装するように設計されています。カスタム・アダプタは、それ自体には機能がない単純なアダプタです。アダプタ・レベルのプラグインを構成して、かわりに機能を実装できるプレースホルダです。Stock Service の例では、プラグインはカスタム・アダプタに対して構成されます。プラグインは、すべてのイベントの処理を担当します。つまり、Stock Service プラグインでは chain.getNext() メソッドは呼び出されません。

プラグインの get メソッドにより、Vector に Entry オブジェクト（株価エントリ）のリストが追加され、その Vector に基づいて ExtensibleEntrySet が作成されます。

```
public void get(Chain chain, Credentials creds, DirectoryString base,
          Int8 scope, Filter filter, Bool typesonly, Vector attributes,
          Vector result) throws DirectoryException, ChainException {

     // Since this method is a handler, chain.getNext is not called.

     if (scope.intValue() == SearchScope.BASEOBJECT &&
          base.equals(this.suffix)) {
          // handle the logical root of the adapter
          Entry root = this.getSimpleEntry(this.suffix);
          Vector entries = new Vector();
          entries.add(root);
          result.add(new ExtensibleEntrySet(entries));
          return;
     }

     //This adapter only supports searches based on an equality match
//or an or'ing of equality matches
     if (filter.getSelector() != Filter.EQUALITYMATCH_SELECTED &&
          filter.getSelector() != Filter.OR_SELECTED) {
          throw new DirectoryException("Only equality match or an or'ing "+
     "of equality matches are allowed");
     }

     Vector entries = new Vector();

     //If the filter is an OR filter, we can iterate over every quote
     if (filter.getSelector() == Filter.OR_SELECTED) {
          Iterator it = filter.getOr().iterator();
          while (it.hasNext()) {
               Entry entry = getStockEntry((Filter) it.next());
               if (entry != null) {
               entries.add(entry);
               }
          }
     }else{
          //single quote
          Entry entry = getStockEntry(filter);
          if (entry != null) {
               entries.add(entry);
          }
     }

     //We use the ExtensibleEntrySet as a simple holder for entry sets.
     result.add( new ExtensibleEntrySet(entries));
}
```

ExtensibleEntrySet を使用して EntrySet を作成するのは最も簡単な方法ですが、処理がクライアントに戻される前にすべての結果をコンパイルする必要があるため、最も効率のよい方法ではありません。この場合、フィルタの各項目に対するサービスがコールされます。このリクエストをより効率的に処理できるのは、Stock Service からリクエストされる時に新規エントリを取得して EntrySets を作成する方法です。

LDAP プロキシ・アダプタはこのように動作します。LDAP プロキシ・アダプタの EntrySet に次の Entry が要求されると、システムにより、リモート・サーバーから一度に 1 つずつ次のエントリが取得されます。これは、LDAP プロトコルが意図する動作です。クライアントはすべてのエントリが処理される前にエントリの取得を開始できるため、この方法ははるかに効率的です。また、クライアントによるエントリ取得の停止や、問合せの中断も可能です。

プラグインがリクエストどおりにエントリを戻す EntrySet を作成するためには、com.Oracle.vde.EntrySet の実装を作成する必要があります。各 EntrySet で、次のメソッドを実装する必要があります。

boolean hasMore()

EntrySet に他のエントリがある場合に true を戻します。このメソッドにより悪い影響が出ないように注意する必要があります。

Entry getNext()

EntrySet の次のエントリを戻します。他にエントリがない場合には NULL を戻します。

void cancelEntrySet()

このメソッドは、EntrySet が完全に実行されない場合に呼び出されます。これにより、カスタムの EntrySet 実装が保持しているシステム・リソースが解放されます。

次に、証券 Web サービスからアダプタを作成する前述のプラグイン実装を示します。ただし、この例で get メソッドにより作成されるのは、カスタムの EntrySet に渡される記号のリストのみです。

```
public void get(Chain chain, Credentials creds, DirectoryString base,
          Int8 scope, Filter filter, Bool typesonly, Vector attributes,
          Vector result) throws DirectoryException, ChainException {
     if (scope.intValue() == SearchScope.BASEOBJECT && base.equals(this.suffix)) {
          Entry root = this.getSimpleEntry(this.suffix);
          Vector entries = new Vector();
          entries.add(root);
          result.add(new ExtensibleEntrySet(entries));
          return;
     }

     //This adapter only supports searches based on an equality match
//or an or'ing of equality matches
     if (filter.getSelector() != Filter.EQUALITYMATCH_SELECTED && filter.getSelector()
!= Filter.OR_SELECTED) {
          throw new DirectoryException("Only equality match or an or'ing"+
     " of equality matches are allowed");
     }

     String rdn="uid";
     ArrayList symbols = new ArrayList();
     //If the filter is an OR filter, we can iterate over every quote
     if (filter.getSelector() == Filter.OR_SELECTED) {
          Iterator it = filter.getOr().iterator();
          while (it.hasNext()) {
//             Extract the symbol from the filter
               String symbol = new String(filter.getEqualityMatch().
               getAssertionValue().toByteArray());

               //The attribute being checked in the equality search
               //doesn't really matter, but we need an RDN for each entry
               rdn = new String(filter.getEqualityMatch().
               getAttributeDesc().toByteArray());
               symbols.add(symbol);
          }
     }else{
          //single quote
          //Extract the symbol from the filter
          String symbol = new String(filter.getEqualityMatch().
               getAssertionValue().toByteArray());

          //The attribute being checked in the equality search doesn't
          //really matter, but we need an RDN for each entry
          rdn = new String(filter.getEqualityMatch().
               getAttributeDesc().toByteArray());
          symbols.add(symbol);
     }
```

```
        //We use the ExtensibleEntrySet as a simple holder for entry sets.
        result.add( new StockEntrySet(symbols.iterator(),rdn,this.base));


    }
```

この例では、記号のリストは、検索フィルタの or を反復することによって作成されています。コンパイルされたリストは、次に示すカスタムの EntrySet 実装に渡されます。

```
public class StockEntrySet implements EntrySet {

    Iterator quotes;
    String rdn;
    String base;

    public StockEntrySet(Iterator quotes, String rdn,String base) {
        this.rdn = rdn;
        this.quotes = quotes;
        this.base = base;
    }

    public Entry getNext() throws DirectoryException {
        Entry entry = this.getStockEntry((String) quotes.next());
        if (entry == null) {
            if (this.hasMore()) {
            return this.getNext();
            }else{
            return null;
            }
        }else{
            return entry;
        }
    }

    public boolean hasMore() {
        return quotes.hasNext();
    }

    /**
     * Returns an entry for a stock quote
     * @param filter
     * @return An entry for the stock quote, or null for none.
     * @throws DirectoryException
     */
    public Entry getStockEntry(String symbol) throws DirectoryException {
        //Create a new entry with the symbol as the RDN
        Entry entry = new Entry(new DirectoryString(rdn + "=" + symbol + "," +
this.base));

            //This uses an Apache Axis generated client stub
            NetXmethodsServicesStockquoteStockQuoteService service = new
            NetXmethodsServicesStockquoteStockQuoteServiceLocator();
            try {
                NetXmethodsServicesStockquoteStockQuotePortType
                quoteService = service.
                getNetXmethodsServicesStockquoteStockQuotePort();
                double value = quoteService.getQuote(symbol);
                if (value == -1) {
                return null;
                }

                //Create the attribute for the entry
```

```
                Vector vals = new Vector();
                vals.add(new DirectoryString(symbol));
                entry.put(new DirectoryString(rdn),vals);

                vals = new Vector();
                vals.add(new DirectoryString("top"));
                vals.add(new DirectoryString("stockForOrganization"));
                entry.put(new DirectoryString("objectClass"),vals);

                vals = new Vector();
                vals.add(new DirectoryString(Double.toString(value)));
                entry.put(new DirectoryString("quote"),vals);

                return entry;

            } catch (ServiceException e) {
                throw new DirectoryException("Could not load web service : " +
e.getMessage());
            } catch (RemoteException e) {
                throw new DirectoryException("Could not load web service : " +
e.getMessage());
            }

        }

        public void cancelEntrySet() {
            // nothing to do
        }
}
```

この EntrySet 実装では、現在どの記号が処理されているかを追跡する java.util.Iterator が使用されています。StockEntrySet クラスでは、クライアントのかわりに Oracle Virtual Directory によって Entry がリクエストされるまで、エントリ結果作成のために Web サービスが呼び出されることはありません。

このシナリオでは、注意する必要のあるイベントが 1 つあります。プラグインでは、1 つ以上の株式の検索がサポートされているため、すべての検索で有効な結果が戻されるとはかぎりません。株式のリストを完全に検索し終わる前に、getNext で Oracle Virtual Directory に NULL という結果が戻されると、Oracle Virtual Directory では結果が空であるとみなされることに注意する必要があります。この状況に対処するために、getNext には、getStockEntry の呼出しの後に特別なコード・ブロックが追加されています。getStockEntry で NULL が戻され、リクエストされた一連の株式の反復が終了していない場合には、次の候補を処理するために getNext により getNext 自体が呼び出されます。この再帰は、有効な結果が 1 つ以上戻されるか、すべての問合せが完了するまで続きます。

フィルタ処理

LDAP フィルタ処理は注意が必要なタスクになる場合もあります。Oracle Virtual Directory プラグインのコンテキストでは、前処理または後処理の際にフィルタの解析に役立ちます。どちらの方法にもそれぞれのメリットとデメリットがあり、必ずしも相互に排他的であるとはかぎりません。

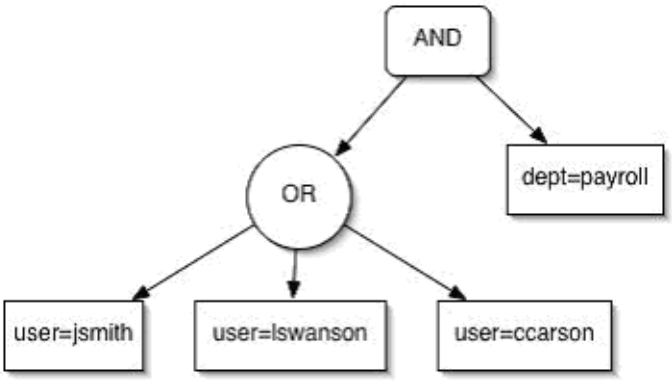
後処理のフィルタ処理では、com.Oracle.vde.util.FilterUtils.evalFilter(Entry e,Filter f) メソッドを使用して、結果として戻されるエントリが必要なフィルタに一致するかどうかが確認されます。これはフィルタを処理する最も簡単な方法で、Entry オブジェクトの集合としてメモリーに残る事前定義済の少量のデータ・セットを処理する際に便利です。一般的に、フィルタを別の形式（SQL の WHERE 句や外部 API の特別なオブジェクト・モデルなど）に変換する必要がある場合には、このメソッドは適していません。

前処理フィルタはフィルタを解析するように設計されており、解析したフィルタを変更された検索に適用するか、検索ターゲットが解釈できる別の形式に変換します。これを理解する最善の方法は、LDAP フィルタを SQL の WHERE 句に変換してみることです。これを実行するに

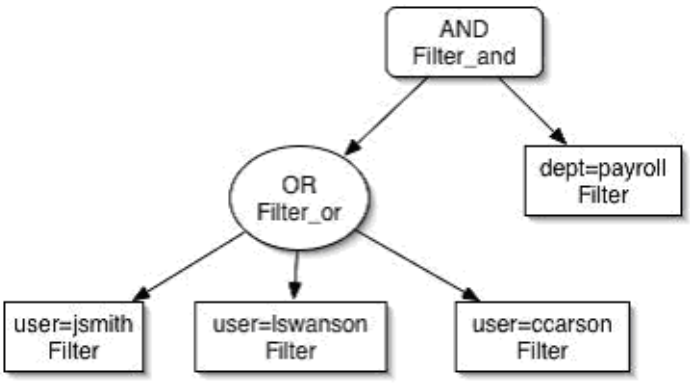
は、フィルタ・オブジェクトを調査する必要があります。例として次の LDAP フィルタを変換します。

(&(|(user=jsmith)(user=lswanson)(user=ccarson))(dept=payroll))

このフィルタを SQL の WHERE 句に変換します。このフィルタには、ユーザーが jsmith、lswanson または ccarson で部門が payroll のすべてのレコードと指定されています。次にこのフィルタを図で示します。

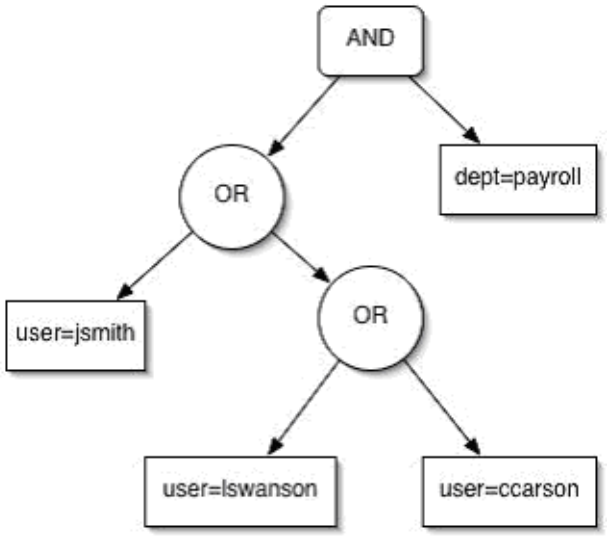


これを SQL の WHERE 句に変換するには、LDAP フィルタを変換するためにツリーを調査する**再帰**機能を使用します。Oracle Virtual Directory では、前述のフィルタは Filter オブジェクトの階層として表されます。これらのオブジェクトには、調査可能なツリーを作成するために別の Filter オブジェクトの集合が含まれます。前述のフィルタには、次に示すオブジェクト・モデルが含まれています（フィルタ要素を表すために使用されているクラス名は、操作または演算子の下に表示されています）。



このツリーを調査するために、フィルタの getSelector() メソッドに問い合せてフィルタのタイプを判断する再帰メソッドが使用されます。フィルタのタイプが判明したら、getFilterType メソッド経由で値を抽出する必要があります。たとえば、フィルタが `user=jsmith` などの等価フィルタの場合、フィルタ・オブジェクトの値は currentFilter.getEqualityMatch() から取得します。この場合の戻り値は AttributeValueAssertion で、属性名および値が Oracle として保存されます。取得したら、Oracle 値は非常に簡単に文字列オブジェクトに変換できます（後続のサンプルを参照）。Filter_and および Filter_or オブジェクトは、操作中の子フィルタを反復するための java.util.Iterator クラスを戻します。

LDAP フィルタでは、関連ごとに項目が 2 つに制限されるわけではありません。次に示すように、OR 部分には 3 つのオペランドがあります。SQL で許可されているオペランドは操作ごとに 2 つのみであるため、前述のツリーはバイナリ・ツリーに変換する必要があります。



OR 操作は、2 つの別々の OR 操作に分割されています。フィルタの最後の WHERE 句は、((user=jsmith) OR ((user=lswanson) OR (user=ccarson))) AND (dept=payroll) です。LDAP の前置記法は、操作ごとに 2 つのオペランドのみを伴う SQL の LIKE 中置記法に変換されています。次に、変換のソース・コードを示します。

```
import com.Oracle.vde.util.*;
import com.Oracle.ldapv3.*;
import java.util.*;

public class ConvertFilter {

public static void main(String[] args) throws Exception {
String ldapFilter = "(&(|(user=jsmith)(user=lswanson)" +
          "(user=ccarson))(dept=payroll))";

System.out.println("Ldap Filter : " + ldapFilter);
System.out.println("SQL WHERE : " +
          filterToSQL(ParseFilter.parse(ldapFilter)));
    }
    /**
*Converts an ldap filter to an SQL WERE clause
*@param currentFilter The filter being converted
     */
public static String filterToSQL(Filter currentFilter) {
String[] filterVal;
String infix="";
switch (currentFilter.getSelector()) {
      case Filter.EQUALITYMATCH_SELECTED :  // (attrib=val)
          filterVal = getString(currentFilter.getEqualityMatch());
return filterVal[0] + "=" + filterVal[1];

case Filter.PRESENT_SELECTED : // (attrib=*)
return new String(currentFilter.getPresent().toByteArray()) +
              "=*";

case Filter.GREATEROREQUAL_SELECTED : // (attrib>=val)
filterVal = getString(currentFilter.getGreaterOrEqual());
return filterVal[0] + ">=" + filterVal[1];

case Filter.LESSOREQUAL_SELECTED :  // (attrib<=val)
filterVal = getString(currentFilter.getLessOrEqual());
return filterVal[0] + "<=" + filterVal[1];
```

```
case Filter.SUBSTRINGS_SELECTED : // (attrib=val*ue)
filterVal = getString(currentFilter.getLessOrEqual());
return filterVal[0] + " LIKE " + filterVal[1];

case Filter.AND_SELECTED : // &((attrib=val)(attrib2=val2))
Filter_and andFilter = currentFilter.getAnd();

infix = "";
for (Iterator andEnum = andFilter.iterator();
              andEnum.hasNext();) {
Filter aFilter = (Filter) andEnum.next();
infix += "(" + filterToSQL(aFilter) + ") AND ";
                }

infix = infix.substring(0,infix.lastIndexOf("AND")) + " ";
return infix;

case Filter.OR_SELECTED : // &((attrib=val)(attrib2=val2))
Filter_or orFilter = currentFilter.getOr();
infix = "";
for (Iterator orEnum = orFilter.iterator();orEnum.hasNext();)
             {
     Filter aFilter = (Filter) orEnum.next();
infix += " ( " + filterToSQL(aFilter) + " ) OR ";
                }
infix = infix.substring(0,infix.lastIndexOf("OR")) + " ";
return infix;

case Filter.NOT_SELECTED : // !(&((attrib=val)(attrib2=val2)))
return " NOT (" + filterToSQL(currentFilter.getNot()) +
                  ") ";

case Filter.APPROXMATCH_SELECTED : // (attrib~=val)
filterVal = getString(currentFilter.getApproxMatch());
return filterVal[0] + " LIKE " + filterVal[1];

case Filter.EXTENSIBLEMATCH_SELECTED : //not standard
return ""; //not supported
        }

//will never reach
return "";
    }

    /**
*Converts an AttributeValueAssertion to a two element array with the
*first being the attribute name and the second being the value
      */
public static String[] getString(AttributeValueAssertion ava) {
String matchAttr = new String(ava.getAttributeDesc().toByteArray());
String matchVal = new
String(ava.getAssertionValue().toByteArray(),"UTF8");

return new String[] {matchAttr,matchVal};
    }
}
```

クラスの概要

プログラミング・ドキュメント（Javadocs）は、ソフトウェア配布やオラクル社の様々な Web サイト、または Oracle Virtual Directory Manager のオンライン・ドキュメント内にあります。次の項では、使用可能なクラスの高度な概要を説明します。

仮想サービス・インタフェース

仮想サービス・インタフェース（VSI）には、LDAP と同じように Oracle Virtual Directory をコールするメソッドが用意されています。VSI は、コンテキストがグローバル・プラグインであるかアダプタ・レベルのプラグインであるかにかかわらず、同様に機能します。チェーン内に 3 つのプラグインがあり、その中の 1 つ目のプラグインが VSI へのコールを実行する場合、2 つ目および 3 つ目のプラグインにも出現するように、コールのコンテキストは Oracle Virtual Directory です。コールが 2 つ目のプラグインから実行された場合、コールは 3 つ目のプラグインのみを経由します。最後に、コールが 3 つ目のプラグインから実行された場合、そのコールはどのプラグインも経由しません。これは、プラグインがグローバルの場合にもローカルの場合にも当てはまります。プラグインがグローバルの場合、コールは Oracle Virtual Directory のルーティング・システムを経由してアダプタ・レベルのチェーンまで、グローバル・チェーンの外でも続行されます（ルーターによって 1 つ以上のアダプタが選択されているかどうかによります）。これにより、Oracle Virtual Directory へのコールの一貫性が保証されるだけでなく、プラグインが自身をコールする無限ループからも保護されます。VSI は、各プラグイン・メソッドに渡されるチェーン・オブジェクト経由で取得されます。

グローバル・サービス・インタフェース

グローバル・サービス・インタフェース（GSI）には、Oracle Virtual Directory に対して、エンド・クライアントから実行されているような LDAP に似たコールを行うメソッドが用意されています。各コールはアクセス制御システム（有効な場合）を介して処理され、ルーターによって操作に適切なアダプタを選択することが可能になります。GSI は、LDAP リスナーや Web サービス・リスナーが通信するのと同じインタフェースです。

グローバル・サービス・インタフェースは、chain.getVSI().getGSI() を呼び出すことで VSI 経由で取得できます。このハンドルを使用すると、`add`、`bind`、`delete`、`get`、`getByDN`、`modify` および `rename` メソッドをすべて呼び出せます。

アダプタ・サービス・インタフェース

アダプタ・サービス・インタフェース（ASI）には、ルーター・レベルにおいて、または特定のアダプタに対して直接、LDAP と同じように Oracle Virtual Directory をコールするメソッドが用意されています。Oracle Virtual Directory の JoinView アダプタおよびそのジョイナでは、検索および結合の対象であるアダプタとの通信に ASI が使用されます。このインタフェースは、プラグイン・クラスの検索情報を提供するために構成した内部アダプタから情報を取得する際に便利です。

アダプタ・サービス・インタフェースは、chain.getVSI().getASI() を呼び出すことで VSI 経由で取得されます。このハンドルを使用すると、`add`、`bind`、`delete`、`get`、`getByDN`、`modify` および `rename` メソッドをすべて呼び出せます。各メソッドには、アダプタ名のパラメータを指定するものと指定しないものの 2 種類があります。アダプタ名が指定されているメソッドを使用して特定のアダプタを選択するか、別のメソッドを使用して、ルーターがルーティング・ロジックおよびルーティング構成に基づいて適切なアダプタを選択するようにします。

警告： ASI および GSI のどちらにおいても、現在のプラグインの前にあるコンテキストをコールすると、プラグインが無限ループに捕捉される場合があります。これにより、プラグイン・コードが繰り返しコールされ、予期しない結果を招く原因になる可能性があります。意図的に発生させる場合を除き、プラグインがスタックを遡ってコールし、ループが発生する可能性のあるシナリオには注意してください。一般的に、特定のアダプタをコールする必要がある場合を除き、VSI を使用するのが最も安全です。

Oracle Virtual Directory には、ループ検出メカニズムは用意されていません。スタックのオーバーフローまたはメモリー不足が原因による Oracle Virtual Directory とカスタム・プラグインとのクラッシュが検出された場合、ループが原因である可能性が高いです。

VSI、GSI および ASI のいずれも共通のインタフェースを共有しており、追加機能が用意されているインタフェースもあります。詳細は、API ドキュメントを参照してください。

add()
LDAP の追加操作を実行します。このメソッドには 2 つのバージョンがあり、Oracle Virtual Directory ルーターによるターゲット・アダプタの選択、または特定のアダプタの選択が可能です。

bind()
LDAP のバインド操作を実行します。Oracle Virtual Directory ルーターによるアダプタの選択、または特定のアダプタの適用が可能です。

delete()
LDAP の削除操作を実行します。Oracle Virtual Directory ルーターによるアダプタの選択、または特定のアダプタの適用が可能です。

get()
LDAP の取得操作を実行します。Oracle Virtual Directory ルーターによる適切なアダプタの選択が可能です。get メソッドは、EntrySet 値の java.util.Vector を戻します。EntrySet は、問合せが行われた各アダプタに含まれます。

getByDN()
特定の DN を使用して LDAP ベースの検索を実行する便利なメソッドです。コール元は、選択して特定のアダプタを指定するか、Oracle Virtual Directory ルーターによる選択を行います。

modify()
LDAP の変更操作を実行します。コール元は、特定のアダプタを指定するか、ルーターによる自動選択を選択します。

rename()
LDAP の名前変更操作を実行します。コール元は、特定の送信元および宛先アダプタを指定するか、ルーターによる自動選択を選択します。

ジョイナ

Oracle Virtual Directory の JoinView アダプタではジョイナを使用して、特定のアダプタからのエントリを結合し、プライマリ・アダプタのエントリとマージする処理を実行します。ジョイナは、新しいジョイナの実装に必要な基本的な操作やメソッドを定義する抽象クラスです。ジョイナは、結合済のエントリに対して操作を実行する必要がある場合に JoinView アダプタによりコールされます。また、LDAP 操作前のデータ操作を可能にする事前アクション操作を定義します。コールされる操作に応じて、ジョイナによるターゲット結合アダプタ内のターゲット・エントリの選択を可能にする mapOperationTargetByEntry メソッドも定義します。

ジョイナは、プライマリ・アダプタおよびターゲット・アダプタとともにインスタンス化されます。JoinView アダプタは常にプライマリ・アダプタのコンテキスト内で機能し、マッピングや操作をターゲット結合アダプタで実行する必要がある場合にジョイナ・メソッドを呼び出します。

JoinView アダプタの取得操作により、プライマリ・アダプタの結果のみに基づいて JoinEntrySet が作成されます。続いて Oracle Virtual Directory クライアントが Oracle Virtual Directory の結果をポーリングすると、JoinEntrySet クラスによりジョイナ JoinByEntry メソッドが呼び出され、結合アダプタへのコールとエントリ結果のマージが行われます。2 つ以上の結合関係が構成されている場合、定義されているすべての関係に基づいてエントリが完全に結合されるまで、エントリ・セットの処理はすべての結合をループします。

ジョイナ・コンストラクタ・メソッドは、JoinView アダプタによってジョイナがインスタンス化される際に呼び出されます。

注意： この状況は、最初の LDAP 操作が（遅延構成という形で）JoinView アダプタによって処理されるまで発生しません。コンストラクタは、関連するターゲット・アダプタ名とともに、構成ファイルのジョイナの構成パラメータに渡されます。

---

createJoinFilter メソッドは、一般的にはローカル・メソッドで、AdapterServiceInterface への後続のコール用に検索フルターを作成するために joinByEntry メソッドによって呼び出されます。

ユーティリティ・クラス

PluginUtils

このクラスは、renameAttribute、copyAttribute などを含むマッピング機能の基本的なツールボックスです。通常、これらのクラスは、マッピング・スクリプトで使用されますが、Java プラグインでも使用できます。

FilterTools

このクラスには、LDAP 検索フィルタを作成および操作するためのメソッドがあります。

ParseFilter

ParseFilter クラスには、**文字列**と**フィルタ**を相互に変換する機能があります。

DNUtility

このクラスには、個々の DN 名コンポーネントの操作を可能にする識別名の展開や作成など、識別名の操作ルーチンが用意されています。

LDAPResult

LDAPResult は、**Int8** 値を戻す任意のメソッドから戻される結果の比較に使用できるユーティリティ・クラスです。これらの定数は、結果コードの LDAP エラー・コードへの変換に便利です。

VDELogger

Logger クラスには、Oracle Virtual Directory ロギング・ファシリティ（Log4J）へのインタフェースがあります。このクラスを使用して、コンソール / 監査メッセージを Oracle Virtual Directory のメッセージと統合します。

PasswordEncryptor

このクラスには、文字列値を Crypt、SHA、SSHA などの様々なハッシュ形式に暗号化する多様なメッソドがあります。

データ・クラス

Attribute

Attribute は、Entry クラスとともに使用される基本的なオブジェクトです。Attribute では（属性名などの）タイプが定義され、その値が含まれます。また、クローニングおよび等価テスト用のメソッドが用意されています。

Credentials

セッションの資格証明を保持する基本的なオブジェクト。このオブジェクトには、セッションの IP アドレス、binddn およびパスワード（必要な場合）が含まれます。通常、AdapterServiceInterface に関連する多くの操作では、関連するのは binddn のみです（getUser() および setUser(DirectoryString) を参照してください）。

Entry

このオブジェクトは、LDAP エントリおよび LDAPModify リクエストなどの部分エントリの保持に使用されます。多くの場合、FilterTool ユーティリティをこれらのオブジェクトと組み合せてフィルタをテストします。

EntryChange

このオブジェクトには、LDAP 変更項目が含まれます。変更リクエストを処理する際には、通常、EntryChange オブジェクトの Vector が AdapterServiceInterface に渡されます。各 EntryChange には、単一エントリへの単一の変更が含まれます。

EntrySet

EntrySet には、アダプタからの一連の問合せ結果が含まれます。メソッドが最初に EntrySet を受け取る際には、エントリの結果セットがメモリーに存在しない場合があることに注意してください。getNext メソッドを呼び出すことで、EntrySet から一意の Entry オブジェクトが戻されます。getNext が呼び出されるたびに、関連するアダプタまたはプラグイン・クラス・コードが呼び出され、存在する場合には次の Entry が取得されます。他のエントリの可用性をテストするには、hasMore() メソッドを使用します。

ヒント： 結果セット全体を処理する場合以外は、getNext() を直接呼び出さないようにしてください。LDAP クライアントから呼出しを実行することをお薦めします。Oracle Virtual Directory プラグイン・クラスの場合には、特別なメソッド postSearchEntry() が用意されており、クライアントに戻される際に各エントリを変更できます。アダプタから到着するたびにエントリを処理するのではなく、Oracle Virtual Directory で結果セット全体を一度にロードするなど、不必要に getNext() を呼び出すと、メモリーの過剰な使用およびパフォーマンス低下の原因になる可能性があります。

Filter
Filter オブジェクトは、標準の LDAP フィルタを表します。このオブジェクトには、LDAP フィルタの設定、テストおよび比較に便利なメソッドがあります。Filter オブジェクトには、その他のフィルタ・オブジェクトの階層（Filter_and、Filter_or など）が含まれる場合もあります。

LDAPURL
このクラスには、標準の LDAPURL を解析または作成するためのメソッドがあります。

データ・タイプ

BinarySyntax
パスワードまたはユーザー証明書などの任意のバイナリ値。

DirectoryString
大 / 小文字が区別されない文字列。

IA5String
大 / 小文字が区別される文字列。

IntegerSyntax
整数値。

DistinguishedName
識別名の値（比較は DN 等価ルールに準拠します）。

例外

DirectoryBindException
バインドに失敗した場合にスローされる例外。

DirectoryException
任意のディレクトリ・エラー時にスローされる一般的な例外。詳細は getMessage() を参照してください。LDAP エラー・コードを確認する場合は getLDAPErrorCode() を参照してください。この例外は、アダプタまたは別のプラグインによって生成されます。

DirectorySchemaViolation
リモート・スキーマまたはローカル・スキーマに違反するエントリの追加や変更が試行されると、スキーマ違反が発生します。

InvalidDNException
InvalidDNException は、無効な DN がパラメータとして渡されるとオブジェクトおよびユーティリティによってスローされます。

10

プラグインおよびマッピングの構成

この章では、Oracle Virtual Directory のカスタム・プラグインおよびマッピングの構成方法を説明します。

概要

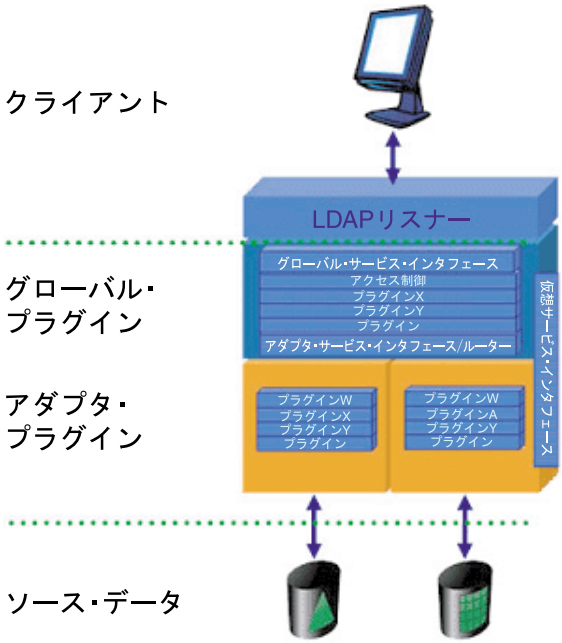
Oracle Virtual Directory には、Oracle Virtual Directory へのリクエストを処理および操作するカスタム・プラグインの作成機能があります。Oracle Virtual Directory は、Java プラグインおよび Python Mapper プラグインの 2 種類をサポートしています。操作上、Oracle Virtual Directory ではこれら 2 種類のプラグインは同等に扱われます。この章では、プラグインの実行および構成方法を説明します。特に記載がないかぎり、プラグインはマッピング・プラグインまたは Java プラグインを意味します。

プラグインは、すべてのリクエストを処理するグローバル操作レベル、または特定のアダプタに対するリクエストを処理するアダプタ・レベルに構成できます。また、特定の操作やネームスペースに対して実行されるようにプラグインを構成することもできます。プラグインの作成方法の詳細は、開発者ガイドを参照してください。

プラグインに関しては、Oracle Virtual Directory には 3 層のアーキテクチャがあります。

プラグインが結合されて**チェーン**が形成されます。リクエストが受信されると、まず**グローバル・サービス・インタフェース**に到達し、グローバル・チェーン、**アダプタ・サービス・インタフェース**および最後にアダプタ・チェーンに到達します。

グローバル・プラグインは、インバウンド・パスのアダプタ・プラグイン、およびその後に戻りパスのアダプタ・プラグインが続き、すべてのリクエストに対してコールされます。



インバウンド・リクエストの場合、すべてのグローバル・プラグインが実行されると、リクエストはアダプタ・サービス・インタフェース（ASI）に移動します。ASI は、ルーティングおよびアダプタの処理が決定される Oracle Virtual Directory の一部です。ルーティング・システムによりリクエストにどのアダプタが含まれるかが決定されると、選択されたアダプタ・プラグインが実行されます。アダプタによりレスポンスが生成され、さらに処理およびフィルタ処理を実行するためにチェーンに戻されます。

前述したように、チェーンには複数のプラグインが結合されています。Oracle Virtual Directory では、各 LDAP 操作（add、bind、delete、get、modify、rename）に異なるチェーンを構成できます。各 LDAP 操作のグローバル・レベルおよびアダプタ・レベルの両方に、異なるチェーンを構成できます。たとえば、構成のデバッグまたは監査を支援するために DumpTransactions プラグインを構成しているとします。add、modify および delete のために DumpTransactions プラグインをコールする必要があるが、search および bind リクエストの記録は不要だとします。この設定では、DumpTransactions プラグインは、グローバル・プラグインとして Add、Modify、Delete および Rename チェーンに構成されているとします。

Oracle Virtual Directory Manager では、Oracle Virtual Directory エディタを使用すると、サンプルおよび拡張の 2 つのモードのいずれかでプラグインを構成できます。サンプル構成モードでは、すべてのプラグイン・チェーンが同一で、すべてのプラグインがすべての操作に対してコールされます。拡張モードでは、どのプラグインがどのような順序で各 LDAP 操作に対してコールされるかを制御できます。

グローバル・プラグインの例

たとえば、Dynamic Groups プラグインだけでなく、変更リクエストを監査するために DumpTransactions プラグインも使用するグローバル構成があるとします。

これらはグローバルに構成されているため、前述の構成は、Oracle Virtual Directory Manager の Oracle Virtual Directory エディタにある「Engine」の「Plug-ins」タブに表示されています。DumpTransactions はすべてのチェーンで実行する必要がありますが、Dynamic Groups は get チェーンでのみ実行します。これを実行するには、管理モードを「Advanced」に設定する必要があります。DumpTransactions プラグインを構成するには、「New Plug-in」ボタンを使用して plug-in configuration ウィザードをアクティブ化します。使用可能なプラグインのリストから「Dump Transaction」を選択したら、「Add」ボタンをクリックしてプラグインの構成にパラメータ loglevel を追加し、「Info」に設定します。これにより、ダンプ・トランザクションは、ログ・レベルが「Info」の場合に出力するよう指示されます。ウィザードの次のページでは、操作チェーンはすべて選択されたままにします。

2 番目のプラグインでは、新しい plug-in ウィザードを再度実行し、サーバーから Dynamic Groups プラグインを選択します。このプラグインにはパラメータは不要です。「Chain Selection」ページで、「get」以外のすべての操作を選択解除します。

この構成はサーバー・レベルであるため、これらのグローバル・チェーンは選択されているアダプタのタイプにかかわらず、すべてのリクエストに対して実行されます。

アダプタ・プラグインの例

アダプタ・レベルのプラグインは、特定のアダプタを変換する必要はあるが、他のアダプタの変換は不要な場合に便利です。この例では、Oracle Virtual Directory は、Microsoft Active Directory および IBM Directory Server のかわりをするよう構成されています。クライアント・アプリケーションからの要件は、InetOrgPerson スキーマを使用してユーザー ID をマップすることです。これを実現するには、Active Directory アダプタに Microsoft Active Directory の user オブジェクトを、RFC 標準の InetOrgPerson オブジェクトに変換するマッピングが存在している必要があります。

この問題を解決するには、LDAP プロキシ・アダプタが Microsoft Active Directory のかわりをするために使用した「plug-ins」タブに移動します。

注意： Oracle Virtual Directory/Oracle Virtual Directory Manager には、プラグイン（InetAD）および同じ機能を実行するマッピング・スクリプトの両方が同梱されています。マッピング・スクリプトを使用するとカスタマイズが簡単になり、プラグインにはオラクル社から出荷されたものと同じ機能が提供されています。この例では、マッピング・スクリプトを使用します。

実行していない場合には、サーバーを右クリックして「New」→「Mapping」を選択し、Active Directory から InetOrg への変換を行うスクリプトを作成します。「AD to InetOrg」テンプレートを選択して、マッピング・ファイルを作成します。マッピング・ファイルを作成したら、mpy ファイルを右クリックして「Deploy To Server」を選択します。デプロイしたら、スクリプトをプラグインとして構成できます。

「Active Directory Adapter」の「plug-ins」タブで New Mapping ウィザードを起動します。

画面には、com.Oracle.vde.chain.plugins.mapper.Mapper が実際に稼働しているプラグインであることが表示されています。パラメータとして、マッピング・スクリプトのファイル名が使用されます。この画面の LDAP 操作モードは「Simple」に設定されており、アダプタ「LDAP Adapter 1」に送信されるすべてのタイプの LDAP 操作に対して AD マッピングが実行されることを意味します。

このシナリオの構成を完成させるために、管理者は IBM Directory Server のかわりをする 2 番目のアダプタを定義します。IBM Directory Server ではデフォルトで InetOrgPerson スキーマが使用されているため、この 2 番目のアダプタにはプラグインは構成されません。

システム構成全体において、IBM Directory Server に送信される操作はそのままですが、Active Directory に対する操作は AD から InetOrg へのマッピング機能を介して送信されます。このようにして、単一の Oracle Virtual Directory 内の様々なソースに、異なるマッピングおよび機能を適用できます。

一意の ID プラグイン

一部の Oracle Virtual Directory 環境のユーザーは、Oracle Virtual Directory アダプタを使用して接続する複数のサービスに対して重複したアカウントを持っています。結合を作成して複数のアダプタを統一し、単一の仮想ユーザー・エントリを作成すると、この問題は解決します。ただし、結合を作成しなくてもよい場合があります。たとえば、社員用のディレクトリと顧客用のディレクトリがあり、仕事上の正当な目的で両方のディレクトリにアカウントを持っている社員がいるとします。この場合、結合ビューは作成できません。

Oracle Virtual Directory の一意の ID プラグインは、それぞれの管理ソースで様々なアダプタをランク付けすることでこの問題を解決します。たとえば、重複しているユーザー ID のどれが社員のアカウントであるかの判断を試行している場合、社員ディレクトリの方が顧客ディレクトリよりも問題に関連があるため、顧客ディレクトリ・アダプタよりも社員ディレクトリ・アダプタを優先します。

アダプタの優先度をランク付けする場合、ランク付けの数値が低いほど優先度が高くなります。たとえば、2 つのアダプタを検索していて、1 つのアダプタの優先度のランクが 5 で、もう 1 つのアダプタのランクが 10 の場合、ランクが 10 のアダプタよりもランクが 5 のアダプタが先に検出されます。

Oracle Virtual Directory の一意の ID プラグインを有効化するには、次の手順を実行します。

構成済の各 Oracle Virtual Directory アダプタの「Routing」タブにアクセスし、ルーティングの優先度を割り当てます。

「Global Plugins」に移動し、新しいプラグインの追加を選択します。「UniqueEntryPlugin」を選択します。

一意キーとして使用されている属性の名前を入力して、uniqueattribute パラメータを追加します。

「Apply to all namespaces」を選択します。

ネームスペースのフィルタ処理

前の例で示したように、Oracle Virtual Directory により、プラグインをグローバルに実行するか単一アダプタのコンテキスト内で実行するかを非常に柔軟に判断できます。いつプラグインが使用されるかをさらに制限する必要が生じる場合もあります。たとえば、アダプタが Microsoft Active Directory ドメインのかわりをするように設定されているとします。アダプタは、`DC=VAN,DC=Oracle,DC=com` を指定しています。ツリー内のそのポイントの下には、`CN=Users` フォルダおよび `CN=Groups` フォルダがあります。マッピングまたはプラグインをツリーの一部にのみ適用し他の部分には適用しないようにするために、定義中に任意のプラグインまたはマッピングにネームスペース・フィルタを追加できます。または、値を入力して、選択したプラグインまたはマッピングのメイン画面の「Add Namespace」ボタンをクリックして追加することもできます。

プラグイン構成処理

New Mapping

このボタンを使用すると、（以前デプロイした）実行用のマッピングを構成できます。マッピングの選択および適用先のネームスペースの定義が可能です。

New Plug-In

このボタンを使用すると、デプロイ済の Java プラグインを実行用に構成できます。ウィザードを使用すると、どの操作をサポートし、どのネームスペースに対してプラグインを使用するかを選択するだけでなく、プラグインの選択やプラグイン・パラメータの構成を実行できます。

Edit

「Edit」ボタンを使用すると、現在選択されているマッピング・プラグインまたは Java プラグインを編集できます。

Delete

「delete」ボタンを使用すると、現在選択されているプラグイン、マッピングまたはネームスペース・パラメータが削除されます。

Ordering

エディタには、「Up」および「Down」ボタンがあります。これらのボタンを使用すると、相互に関連するプラグインの実行順序を変更できます。

Add Namespace

「add namespace」ボタンを使用すると、現在選択されている Mapper プラグインまたは Java プラグインに新規のネームスペースを追加できます。「Add Namespace」ボタンの右にあるフィールドに有効な識別名が入力されるまで、ボタンが有効化されないことに注意してください。

11

アクセス制御

この章では、セキュリティやアクセス制御、およびそれらが Oracle Virtual Directory 製品にどのように実装されているかを説明します。

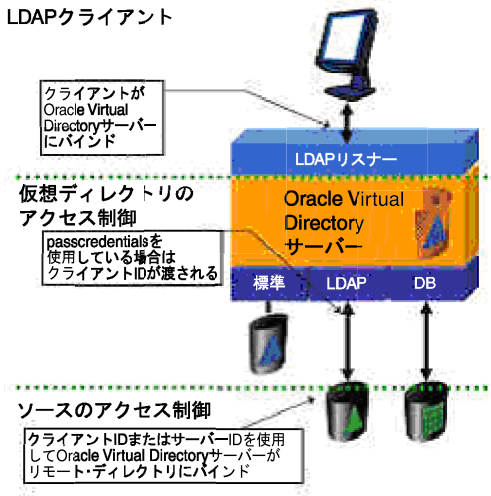
概要

Oracle Virtual Directory は、接続されているすべてのデータ・ストアに一様に適用可能な粒度の細かいアクセス制御をサポートしています。これは、実現する必要のあるアクセス制御機能が指定されている、Internet Engineering Task Force の RFC2820「Access Control Requirements for LDAP」に準拠しています。アクセス制御のルールは、IETF の「LDAP Access Control Model for LDAPv3」（2001 年 3 月 2 日草稿）というタイトルのインターネット・ドラフトに基づいて作成されています。

複数層の認証およびアクセス制御

Oracle Virtual Directory は、複数層のアクセス制御および認証をサポートしています。アダプタを介してアクセスされるリモート・ディレクトリの場合、Oracle Virtual Directory ではそれらのシステムに固有のセキュリティをサポートします。使用されるアダプタやその機能に応じて、passcredentials オプションを設定し、認証およびアクセス制御を実行するためにエンド・ユーザーのバインディング資格証明をリモート・ディレクトリに渡すかどうかを判断できます。

図 11-1 Oracle Virtual Directory の複数層のアクセス制御



パススルー認証

アダプタのパススルー・モードが有効化されていて、ユーザーが Oracle Virtual Directory に対して認証される場合、Oracle Virtual Directory では受け取ったユーザー ID およびパスワード資格証明を使用して、ユーザーのかわりにリモート・ディレクトリにログインします（パスワード認証の場合にのみ可能です）。リモート・ディレクトリへの認証（バインド）に失敗すると、ユーザーの試行したバインドも失敗します。このモードでは、リモート・ディレクトリがユーザーの資格証明を確認する役割を果しています。

never に設定されている場合（または選択したアダプタでサポートされていない場合）は、Oracle Virtual Directory でクライアントの認証を実行する必要があります。これを実行するためには、外部ディレクトリのパスワードをクリアテキストで保存するか、CRYPT、SHA または SSHA の一方向暗号化ハッシュを使用する必要があります。Oracle Virtual Directory で使用されている暗号化ハッシュを判断するには、暗号化されたテキストに {crypt} という形式の接頭辞が適用されている必要があります。代替元のソースでこの形式が使用されていない場合は、マッピング・ルールを設定してこれを定義する必要があります。マッピング・ルールにより、Oracle Virtual Directory に特定の暗号化書式の処理方法を伝達する接頭辞が追加されます（詳細および例は、第 8 章「マッピング・システム」を参照してください）。接頭辞が存在しない場合は、通常のテキスト比較が行われます。

passcredentials の never モードでは、ユーザーによって指定されたパスワードのハッシュ・アルゴリズム（アダプタから戻された値に指定されている）を実行し、アダプタから戻された値と結果を比較することで、Oracle Virtual Directory によって認証が完了します。

注意： クライアントがクリアテキスト・パスワードを使用せずに（証明書などを使用して）バインドすると、サーバーはユーザーの資格証明を代替元のディレクトリに渡すことができません。Oracle Virtual Directory では、構成済のアダプタ・アカウントを使用して、バインド検証およびリクエストされた LDAP サービスを実行します。これは、passcredentials が never に設定されている場合の処理と同じです。

CRAM-MD5 および SASL バインディング

CRAM-MD5 は、Oracle Virtual Directory への認証に使用可能な SASL バインド・メカニズムです。クライアントでサポートされている場合は、SSL を使用せずにネットワーク上でパスワードを保護するために使用できます。ただし、CRAM-MD5 SASL メカニズムでは、パスワード以外の情報の交換に使用するパスワードをサーバーが平文で保持している必要があります。これにより、サーバーは、LDAP クライアントによって指定されたパスワードが有効かどうかを判断できます。このモードを使用する場合、すべてのローカル（標準）ソースおよび代替元ソースでパスワードをクリアテキストで保存する必要があります。

プロキシ・アカウント認証

Oracle Virtual Directory では、使用可能なパスワードのないユーザーを認証する場合、バインド DN がアダプタのネームスペースの外にあるユーザーのかわりを行う場合、または passcredentials が never に設定されている場合にプロキシ（またはデフォルト）アカウントが使用されます。また、接続された LDAP アダプタのディレクトリに対して Oracle Virtual Directory のルート管理者アカウントおよび匿名の両方を認証する際にも、アダプタのプロキシ・ユーザー ID およびパスワードが使用されます。デフォルト・アカウントは、証明書を使用して認証するユーザーにも使用されます。そのため、passcredentials モードが有効化されている場合には、リモート・ディレクトリで**デフォルト・アカウントを権限のないアカウント**（匿名など）**に設定しておく**ことが重要です。これは、現行のアダプタにマップできないアカウントを処理するために、プロキシ・アカウントが必要な状況が数多くあるためです。

クライアント証明書の認証

Oracle Virtual Directory では、X.509 デジタル証明書を使用してクライアントが仮想ディレクトリに対して認証する機能がサポートされています。X.509 デジタル証明書を使用してクライアントが Oracle Virtual Directory に対して認証できるようにするには、次のようにします。

1. SSL が有効な状態で LDAP リスナーを作成します。
2. UNIX システムの場合は vde.sh を、Windows システムの場合は OVIDServer.lax を編集します。vde.sh では、Oracle Virtual Directory を起動するエントリを探し、`Dvde.ldap.requireClientAuth=true` を追加します。OVIDServer.lax では、ファイルの最後に `vde.ldap.requireClientAuth=true` を追加します。

LDAP クライアントでは、X.509 デジタル証明書を使用して仮想ディレクトリに対して認証するには、SSL および SASL がサポートされている必要があります。次に、SSL 認証が機能する 2 つのモードを示します。

- 接続は保護するが実際のディレクトリに対しては認証しない方法としてクライアント証明書を使用
- 証明書を使用して Oracle Virtual Directory をバインドするために SASL を使用

クライアント証明書の認証のガイドライン

- Oracle Virtual Directory へのバインドに証明書を使用している場合、証明書はバックエンドのデータ・ストアに対してではなく、Oracle Virtual Directory に対する認証にのみ使用されます。クライアント証明書の認証に必要な秘密鍵へのアクセス権があるのは LDAP クライアントのみであるため、バックエンドのデータ・ストアに対する認証は公開鍵インフラストラクチャ（PKI）によって阻止されます。そのため、LDAP アダプタを使用している場合には、バックエンドのデータ・ストアに対するすべての Oracle Virtual Directory 操作はプロキシ DN アカウントとして実行されます。
- 証明書には独自の識別名（DN）が含まれます。この識別名は、実際にバインディングしているユーザーの DN と一致しない場合もあります。その場合、アクセス制御リストが適切に機能するように、証明書の DN を Oracle Virtual Directory のユーザーの DN にマップする必要があります。このマッピングは、プラグインを使用して実行できます。
- Oracle Virtual Directory では、keys.jks ファイルに保存されている、ルート CA によって発行された証明書はすべて受け入れられます。

ソース・ディレクトリのアクセス制御

リモート・ディレクトリに対するクライアントとして、Oracle Virtual Directory は、リモート・ディレクトリで実行されている認証ルールに従う必要があります。適用されるルールは、リモート・ディレクトリに渡されたユーザー・コンテキスト（Oracle Virtual Directory が代替元のディレクトリへの接続にどのアカウントを使用しているかなど）によって異なります。passcredentials が 1 に設定されている場合、リモート・ディレクトリでは、Oracle Virtual Directory がリモート・ディレクトリに渡すユーザー・コンテキストに応じてルールが実行されます。Oracle Virtual Directory により、適切に変換されたデータの結果およびエラーがユーザーに戻されます。たとえば、「access denied」というエラーが戻された場合、Oracle Virtual Directory によりそのメッセージがクライアントに戻されます。アクセス制御によりデータがフィルタ処理された場合には、Oracle Virtual Directory によりフィルタ処理されたデータが取得され、構成済の任意の変換が適用されてユーザーに渡されます。

passcredentials が 0 に設定されている場合、リモート・ディレクトリではすべてのリクエストのプロキシ・ユーザーのみが認識され、どのユーザーが Oracle Virtual Directory にバインドされているかにかかわらず同じ認証が適用されます。

どちらのモードでも、Oracle Virtual Directory では独自のアクセス制御および認証が実行されます（次の「Oracle Virtual Directory のアクセス制御」を参照してください）。

Oracle Virtual Directory のアクセス制御

Oracle Virtual Directory では、仮想ディレクトリのネームスペース全体でアクセス制御がサポートされています。Oracle Virtual Directory では、アクセス制御情報を構成ファイル（**acls.os_xml**）に保存することでこれを実現しています。この情報は、**entryACI** および **subtreeACI** 属性への変更リクエストを捕捉することで自動的に保守されます。同様に、これらの属性は、問合せが実行された場合には適切なエントリの一部として提供されます。これにより、Oracle Virtual Directory では、仮想ディレクトリのネームスペース全体でアクセス制御が実行されています。

注意： 同じアクセス制御のドラフト標準を使用している代替元の LDAP ディレクトリにエントリが属する場合、Oracle Virtual Directory 経由でそのソース・ディレクトリ内のアクセス制御を変更することはできません。これは、Oracle Virtual Directory により変更が捕捉され、その内容がエントリの独自の ACI 値に適用されるためです。この場合、それらのエントリに対する変更は、ソース・ディレクトリに直接行う必要があります。他のベンダーのディレクトリ・サーバーではアクセス制御情報の保存に異なる属性が使用されているため、通常これは問題ではありません。

アクセス制御およびグループ

アクセス制御のサブジェクトとしてグループを使用している場合、グループの位置およびアダプタ変換によるグループへの影響を考慮することが重要です。定義されている各 LDAP プロキシ・アダプタでは、DN 属性リストの値が定義されていることを確認してください。これにより、エントリ DN に加えて、仮想ディレクトリ・ツリーにマップする必要のある属性のリストが定義されます。DN を含む属性値に依存するアクセス制御の場合は、値が正しくマップされている必要があります。

複数のアダプタ（ローカル・ストア・アダプタおよび LDAP プロキシ・アダプタなど）からのメンバーが含まれるグループが必要な場合には、グループを LSA アダプタのネームスペース（仮想ディレクトリのローカル・ストアなど）に配置する必要があります。グループが外部の LDAP ディレクトリ内に配置されていて、そのディレクトリには存在しないメンバーが含まれている場合には、変換は正常に実行されません。これは、外部ディレクトリのネームスペースに存在しないエントリには、コンテキストがないためです。

Oracle Virtual Directory のアクセス制御の構成

前の項で説明されているように、Oracle Virtual Directory では、仮想ディレクトリ全体でアクセス制御がサポートされています。Oracle Virtual Directory は、IETF の「LDAP Access Control Model for LDAPv3」（2001 年 3 月 2 日草稿）をサポートしています。この項では、それらのルールが Oracle Virtual Directory にどのように実装されているかを説明します。標準で意図されているように、これらのルールはディレクトリ内のエントリに直接適用できます。または、XML ファイル **acls.os_xml** を編集するか、Oracle Virtual Directory Manager のサーバー・エディタを使用して構成および管理できます。

ACL とも呼ばれるアクセス制御リストまたは ACI（アクセス制御命令）によって、どのクライアントおよびディレクトリ・ツリーのどの部分へのアクセスを許可するかが指定され、ディレクトリ・エントリや属性へのアクセス方法が制御されます。そのため、ACL の主な目的は、データへのアクセス権が必要なユーザーにはその権限が付与され、アクセス権を付与する必要のないユーザーの権限は拒否される、安全な環境の実現における役割を担うことです。

Oracle Virtual Directory では、仮想ディレクトリのネームスペース全体でアクセス制御がサポートされています。そのため、作成される一連の ACL では、保護が必要な DIT の様々な部分が包括的に定義されている必要があります。最近公開された IETF のドラフト[1] によると、各 ACL に対して 6 つの対象またはコンポーネントが定義されています。

- **target/location**: DIT の ACL が適用される場所
- **scope**: ACL が適用される場所より下の深さ
- **rights**: ACL によりアクセス権が付与または拒否されるかどうか
- **permissions**: 付与または拒否される特定のタイプの権限
- **attributes**: ACL を属性またはエントリ全体に適用するかどうか
- **subject**: ACL が適用されるクライアント

定義されている ACL はアダプタを経由して接続されているディレクトではなく、仮想ディレクトリのネームスペース内にあります。つまり、単一の ACL を、複数のアダプタ全体のデータに対するアクセス権を制御するように定義できます。これら 6 つのコンポーネントは、Oracle Virtual Directory のインスタンスに属する一連の ACL の各ルールに対して、Oracle Virtual Directory Manager コンソールに構成されています。構成時、Oracle Virtual Directory のインストーラにより、インストール・プロセス中に入力されたディレクトリ接尾辞に基づいて、デフォルトの ACL セットが作成されます。

[1] IETF の「LDAP Access Control Model for LDAPv3 <draft-ietf-ldapext-acl-model-08.txt>」（2001 年 6 月 29 日のドラフトで失効済）

エディタの使用

エディタを使用するには、Oracle Virtual Directory エディタのメインの「Engine」タブの「ACLs」サブタブを選択します。エディタを使用すると、新規 ACL の指定や、表示されているボタンを使用した既存の ACL の編集ができます。

新規 ACL の作成

仮想ディレクトリの対象のターゲット・エントリの識別名を入力します。省略記号ボタンを使用してディレクトリを探し、サーバーに接続している場合にはブラウザを使用して選択します。「DN」フィールドに DN を入力したら、「New…」をクリックして New ACL ウィザードを起動します。

既存の ACL の編集

既存の ACL を編集するには、ツリーで対象の ACL を選択して「Edit」をクリックします。ACL Edit ウィザードが起動され、選択したエントリを編集できます。

ACL の削除

削除する ACL を選択して、「Delete」ボタンをクリックします。ACL が削除されます。

評価順序の変更

ACL を選択し、必要に応じて「Up」または「Down」ボタンをクリックすると、ACL が定義されている（および解釈される）順序を変更できます。

アクセス制御ルール

target/location コンポーネント

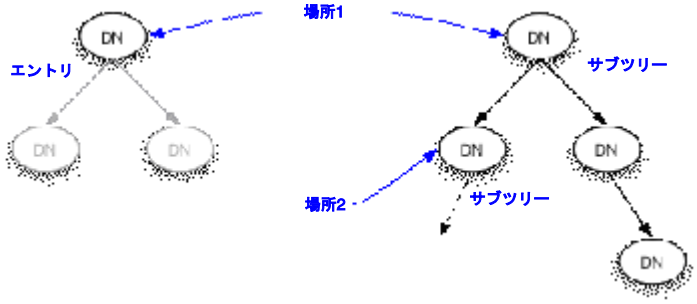
各 ACL は、ディレクトリの指定された場所に適用されます。通常、場所は識別名ですが、特別な場所 [root] を使用してツリーのルートが表される場合もあります。

サーバーのアクセス対象または変更対象のエントリが ACL と同じまたはそれより下の場所に存在しない場合、指定された ACL はそれ以上評価されません。存在する場合には、scope ルール（次を参照）が評価されます。

scope コンポーネント

アクセス制御には、2 つのタイプの有効範囲が定義されています。それは**エントリ**と**サブツリー**で、次の図に示すような役割を果します。

図 11-2 エントリとサブツリーの有効範囲



上の図では、場所と有効範囲が相互に関連している様子が示されており、場所は有効範囲が評価される DIT での位置を示しています。図の左側のエントリ部分では、アクセス対象または変更対象のディレクトリ・エントリ（DN）が、場所 1 で示されているのと同じ DN である場合

にのみ ACL が適用されます。図のサブツリー部分では、場所 1 から始まり下に伸びているすべての DN が、有効範囲がサブツリーである ACL の影響を受けます。指定された ACL の場合、サブツリー有効範囲の唯一のエンドポイントは、1 つ目の ACL よりも下の場所（場所 2 を参照）で宣言された他の ACL によって、1 つ目の ACL で定義されているルールが変更される場合に発生します。

エントリ有効範囲は、多くの場合、様々なサブジェクトおよび拒否の設定（後で説明）とともに使用されます。あるエントリに、同じレベルまたはそれより下のエントリよりも機密性の高い情報が含まれていて、プライベートにしておく必要がある場合に特に便利です。

タイプのみが異なる 2 つの有効範囲のルールが存在する場合、エントリ有効範囲がサブツリー有効範囲より優先されます。

rights コンポーネント

アクセスは許可または拒否できますが、アクセス権はエントリ全体（DN）か、エントリの一部またはすべての属性に適用されます。定義されたルールがディレクトリ・エントリに適用されるエントリ・レベル ACL を選択するか、すべての属性または特定の属性を選択することで属性 ACL を選択できます。

attributes コンポーネント

attributes コンポーネントは、権限をエントリ全体に適用するか一部またはすべての属性に適用するかが決定されるため、permissions コンポーネントと強力にリンクされています。そのため、エントリに影響を与えるエントリを選択するとエントリ・レベルの権限のある ACL が作成され、すべてまたは特定の属性を選択すると属性の権限を設定する ACL が作成されます。

permissions コンポーネント

属性またはエントリに適用する権限は、実行可能な LDAP 操作に似ています。それぞれの LDAP アクセス権限は独立しており、権限は相互に関連しません。実際、ACL の構成時には、2 つの ACL が同じ場所に関連することがよくあります。これは、個々の属性の権限（属性権限）と、エントリ自体の権限（エントリ権限）を分離する必要から生じます。属性権限には、[all]（すべての属性）または属性権限を適用する属性のリストの attribute **コンポーネント**が必要です。エントリ権限には、[entry] の attribute **コンポーネント**が必要です。

属性権限

- **read**: 属性値に対する読取りまたは検索操作権限の付与または拒否
- **write**: 変更または追加される属性値に対する変更操作権限の付与または拒否
- **obliterate**: 削除される属性値に対する変更操作権限の付与または拒否
- **search**: 属性値に対する検索操作権限の付与または拒否
- **compare**: 属性値に対する比較操作権限の付与または拒否
- **make**: このエントリよりも下の新規エントリに新しい属性を作成するための権限の付与または拒否

属性権限の注意事項

make 属性権限は、エントリの作成時にエントリのすべての属性に必要です。特に、add エントリ権限と関連付けられます。

その他の要件で禁止されていないかぎり、ディレクトリのデータを完全に取得できるよう大部分の場所に search、read および compare 属性権限を指定するのが一般的です。同様に、データの変更が必要になる可能性がある場所には、write および obliterate 権限の両方を付与するのが一般的です。

write および **obliterate** には追加操作との関連はなく、make には変更操作との関連はありません。新規エントリが存在しないため、その作成に必要な add および make 権限を新規エントリの親に付与する必要があります。この点は、変更対象のエントリに付与する必要のある write

および obliterate とは異なります。make 権限は write および obliterate 権限とは異なるため、エントリへの新しい子の追加に必要な権限や、同じエントリの既存の子の変更に必要な権限との競合はありません。

変更操作で属性値を置き換えるには、属性に対してエントリの write および obliterate 権限の両方が有効化されている必要があります。

エントリ権限

次に示す権限がエントリ全体の処理に適用されます。

- **add**: 指定された場所より下にエントリを追加する権限の付与または拒否。クライアント・アプリケーションがエントリを追加するには、少なくとも各 objectclass の必須属性を追加するための make 属性権限も付与されている必要があります。
- **delete**: エントリ内の既存の属性権限にかかわらず、エントリを DIT から削除する権限の付与または拒否。
- **renameDN**: エントリの名前を変更または移動する権限の付与または拒否。
- **browseDN**: エントリを参照する権限の付与または拒否。付与されている場合、エントリの名前が明示的に指定されていないディレクトリ操作を使用してエントリにアクセスできます。
- **returnDN**: エントリの DN を LDAP 操作の結果で公開する権限の付与または拒否。

エントリ権限の注意事項

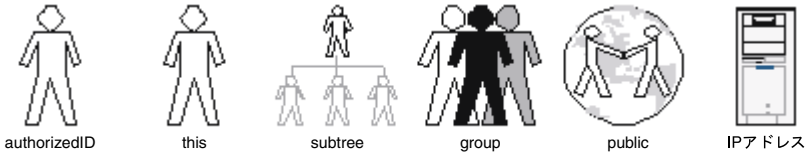
新規 RDN で名前が変更されるエントリには、renameDN を付与する必要があります。ただし、下位エントリがある場合には、その識別名も間接的に変更されることを考慮する必要があります。エントリ名の変更では、RDN 自体の属性など、含まれる属性の前提条件権限は不要です。

さらに、名前の変更を行うには 2 つの条件があることに注意してください。(1) 上位 DN の名前は変更されません（つまり、文字どおり RDN の名前が変更されます)。また、(2) 名前の変更により DN および DIT の下位項目が移動され、エントリの上位 DN が新しくなります（つまり移動されます)。

どのタイプの名前の変更の場合でも、移動される DN には renameDN 権限が必要で、(その下に新しいエントリが配置される) ターゲット DN には add および renameDN の両方の権限が必要です。

その他の要件で禁止されていないかぎり、ディレクトリのデータを完全に取得できるよう大部分の場所に browseDN および returnDN エントリ権限を指定するのが一般的です。同様に、データの変更が必要になる可能性がある場所には、write および obliterate 権限の両方を付与するのが一般的です。

subject コンポーネント



ACL の subject では、ACL の適用対象が決定されます。Oracle Virtual Directory では、予期される様々なサブジェクト・タイプが利用されます。このサブジェクト・タイプでは、ユーザー指定またはプロキシ・アカウント（binddn など）からのクライアント資格証明が使用され、次の条件で適用されます。

- **public:** 認証されているかどうかにかかわらず、ACL はディレクトリに接続されているすべてのクライアントに適用されます。
- **this:** 資格証明がアクセス対象のエントリの資格証明と一致するクライアントに適用されます。
- **authorizedID:** クライアント DN の資格証明がこの値と比較された場合に ACL が適用されます。ACL は一致した場合に適用されます。
- **subtree:** ACL は、クライアント資格証明の期限が切れた場合、またはサブジェクトで指定された DN より下に適用されます。
- **group**: subject コンポーネントによって指定された DN が検出され、クライアント資格証明が **groupOfUniqueNames**、**groupOfNames**、**groupOfUniqueURLs** の 3 つのオブジェクト・クラスのいずれかによって決定されたグループのメンバーである場合に ACL が適用されます。最初の 2 つのタイプのグループによって、ユーザーの静的なリストが指定されます。**groupOfUniqueNames** では、uniquemembers 属性のいずれかがクライアント資格証明に一致している必要があります。**groupOfNames** では、member 属性のいずれかがクライアント資格証明に一致している必要があります。3 つ目のグループ・タイプ **groupOfUniqueURLs** では、動的グループが処理されます。また、**memberurl** 属性の値によって指定された検索を実行して取得された 1 つ以上の DN と、クライアント資格証明が一致している必要があります。

subject コンポーネントの注意事項

public

通常 grant とともに public を使用する ACL では、考えられる最も広い範囲のクライアント・ベースに対して DIT へのアクセス権が付与されます。通常、有効範囲が entry の場所 [root] に対して実行されます。また、特に有効範囲が entry の別の ACL および有効範囲が subtree の ACL が適用されているツリー・ベースに対しても実行されます。ツリーの特定部分、および userpassword などの特定の属性が public によって表示または変更されるのを制限するために、その他の ACL も追加されます。

例 :

	ACL 1	ACL 2	ACL 3	ACL 4
location:	**[root]**	**[root]**	**o=oracle.com**	**o=oracle.com**
scope:	**entry**	**entry**	**subtree**	**subtree**
rights:	**grant**	**grant**	**grant**	**grant**
permissions:	**search, read**	**browse, return**	**search, read**	**browse, return**
attributes:	**[all]**	**[entry]**	**[all]**	**[entry]**
subject:	**public**	**public**	**public**	**public**

this

this サブジェクトは、DIT 内に配置されている指定された任意のクライアントに、userpassword、postaladdress、telephonenumber など個人情報の読取りおよび変更の両方を行う権限を付与するために使用されます。通常、一般の public にそのようなアクセスを拒否するために、別の ACL も使用されます。

例 :

	ACL 1	ACL 2	ACL 3
location:	**o=oracle.com**	**o=oracle.com**	**o=oracle.com**
scope:	**subtree**	**subtree**	**subtree**
rights:	**deny**	**grant**	**grant**
permissions:	**read, compare**	**read**	**write, obliterate**
attributes:	**userpassword**	**userpassword**	**userpassword, postaladdress, telephonenumber**
subject:	**public**	**this**	**this**

authorizedID

authorizedID サブジェクトは、プロキシ binddn などの特定のクライアントに、DIT の一部に対するアクセス権を付与または拒否するために使用されます。たとえば、ユーザーが匿名でバインドすると、サーバーにプロキシ資格証明が送信されます。これを考慮すると、プロキシ binddn に DIT の機密エントリや属性の表示または変更を許可するのは賢明ではありません。

例（binddn が cn=service の場合）

	ACL 1	ACL 2
location:	**ou=security, o=octet**	**ou=security, o=octet**
scope:	**subtree**	**subtree**
rights:	**deny**	**deny**
permissions:	**read, search, make, write, obliterate**	**add, delete, renameDN, browseDN, returnDN**
attributes:	**[all]**	**[entry]**
subject:	*authorizedID*=**cn=service**	*authorizedID*=**cn=service**

subtree

subtree サブジェクトは、クライアント資格証明が DIT のサブツリー内のどこからでも検出される場合に便利です。subtree には、関連性の強いブランチの特権アカウントのみが含まれている場合があります。

例（DN を持つ admin）

cn=jbowen、ou=eng_admins、ou=admins、ou=security、o=octet を認証する必要があります。その他の同じような権限を持つ admins とともにこの admin を認証し、サブツリーを使用する ACL 実装の例は次のようになります。

	ACL 1	ACL 2
location:	**o=octet**	**o=octet**
scope:	**subtree**	**subtree**
rights:	**grant**	**grant**
permissions:	**read, search, make, write, obliterate**	**add, delete, renameDN, browseDN, returnDN**
attributes:	**[all]**	**[entry]**
subject:	*subtree*=**ou=admins, ou=security, o=octet**	*subtree*=**ou=admins, ou=security, o=octet**

group

group サブジェクトは、認証を実行するエントリを DIT 内のどこで検索すればよいかを指定する際の柔軟性が最も高いため、特定の権限および許可を様々なカテゴリのユーザーに付与する場合に最も多く選択されます。

たとえば、資格証明を持つ admin の場合、cn=jbowen、ou=admins、ou=security、o=octet を認証する必要があります。オブジェクト・クラスが **groupOfURLs** で属性が **memberURL** の group サブジェクトを使用してこの admin を認証する、ACL とエントリの組合せの例は次のようになります。

	ACL 1	ACL 2
location:	**o=octet**	**o=octet**
scope:	**subtree**	**subtree**
rights:	**grant**	**grant**
permissions:	**read, search, make, write, obliterate**	**add, delete, renameDN, browseDN, returnDN**
attributes:	**[all]**	**[entry]**
subject:	*group*=**ou=sales, ou=depts, o=octet**	*group*=**ou=sales, ou=depts, o=octet**

グループを指定するオブジェクト・クラスに対して調査される subject コンポーネントの DN は、次のようになります。

dn: **ou=sales, ou=depts, o=octet**
objectclass: organizationalUnit
objectclass: groupofurls
ou=sales
memberurl: ldap:///ou=admins,ou=security,o=octet?sub?(objectclass=inetOrgPerson)

上の DN では、memberurl 属性によって指定されている LDAP 検索には少なくとも、クライアント資格証明を認証する次の DN が含まれます。

```
dn: cn=jbowen, ou=admins, ou=security, o=octet
objectclass: inetOrgPerson
cn: jbowen
userpassword: xyz123
```

一般的な付与 / 拒否の評価ルール

クライアントに特定情報の一部へのアクセス権を付与するか拒否するかは、アクセス制御ルールおよび保護されているエントリに関連する多くの要因に基づいて決定されます。次に、決定プロセスでガイドラインとなる規則を示します。

- **特性** : より明確なルールが曖昧なルールより優先されます（ACI の特定のクライアント DN はグループ参照よりも優先されます）。
- **拒否** : 付与よりも優先されます。ACI の値が競合している場合は、拒否が付与よりも優先されます。
- **付与** : アクセス制御情報がない場合はデフォルトです。
- subject および attributes の特性が同一の場合、エントリ有効範囲はサブツリー有効範囲よりも優先されます。

subject のタイプのみが異なる ACL の評価において、Oracle Virtual Directory で採用されている優先順位は、（1）**authorizeID**、（2）**this**、（3）**groups**、（4）**subtrees**、（5）**public** です。

A

Oracle Virtual Directory 2.0 および 10.1.4 のプロパティ表

次の表には、Oracle Virtual Directory 2.0 のプロパティ設定を新しい Oracle Virtual Directory 10.1.4 の XML 設定と比較するための情報を表示します。定義および使用上の注意事項も説明されています。

アダプタ構成

すべてのアダプタの一般的なプロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
num	（対応する要素なし）	Oracle Virtual Directory 2.0 では、Oracle Virtual Directory の現行のインストールに構成されているアダプタの数が指定されます。	Oracle Virtual Directory 10.1.4 では、ユーザーがアダプタの数を直接設定することはできません。
type	standard、ldap、ntlm、dataBase、join、ghost	構成されるアダプタのタイプを定義します。ntlm および ghost は新しいタイプです。	デフォルト : なし
root	root	アダプタのルートを定義します。	例 : dc=oracle、dc=com デフォルト : なし

一般的なアダプタ - ルーティング・プロパティ[1]

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
routing-retrieve	retrieve	リモート・サーバーからの検索でどの LDAP 属性を取得するかを指定します。	許可されている下位要素は include のみです。 デフォルト : すべての属性を取得します。
routing-store	store	リモート・サーバーにどの LDAP 属性を保存するかを指定します。	許可されている下位要素は include および exclude のみです。 デフォルト : すべての属性が保存されます。
routing-include	include	RFC2254 形式の検索フィルタ式。アダプタに渡される問合せのタイプを指定します。有効範囲は式の後に、# および subtree、onelevel または base を入力して指定します。（プロパティのデフォルトのフィルタはありません。有効範囲のない式が指定されている場合、有効範囲のデフォルトは sub です。）	RFC2254 形式の検索フィルタ式。 例 : (&(uid=abc)(ou=oracle))#sub include は retrieve および store の下位要素です。 デフォルト : フィルタ処理は実行されません。
routing-exclude	exclude	アダプタに渡さない問合せのタイプを指定します。有効範囲は式の後に、# および subtree、onelevel または base を入力して指定します。（プロパティのデフォルトのフィルタはありません。有効範囲のない式が指定されている場合、有効範囲のデフォルトは one です。）	RFC2254 形式の検索フィルタ式。例 : (&(uid=abc)(ou=oracle))#one exclude は retrieve の下位要素です。 デフォルト : フィルタ処理は実行されません。

[1] 2.0 のルーティング・プロパティは、backend.X.config.routing-<property>= で始まる adapters.prop の行によって認識されます。これらのプロパティは、2.0 および 10.1.4 の両方で複数のタイプのアダプタを構成する際に使用されます。

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
routing-active	active	アダプタがアクティブ（サービス中）であるかどうかを指定します。アクティブでない場合、名前空間より下の DN に対する LDAP 操作は実行されません。active は、routing の下位要素ではなくなりました。	値 : true の場合アダプタはアクティブで、false の場合はアクティブではありません。 デフォルト : なし
routing-visible	visible	LDAP クライアントにアダプタを表示するかどうかを指定します。表示しない場合、クライアントはクライアントに対するサーバーのネーミング・コンテキストからアダプタを削除できます。ただし、join 操作では表示されたままになります。	値 : Yes の場合、アダプタのルートは namingContexts にリストされ、ユーザー操作でアダプタを使用できます。No の場合、アダプタのルートは namingContexts にリストされませんが、ユーザー操作で使用できます。Internal の場合、アダプタのルートは namingContexts にリストされず、ユーザー操作でも使用できません。 デフォルト : なし
routing-critical	critical	LDAP 操作の結果がクリティカルであるかどうかを指定します。クリティカルでない場合には、（ネットワーク・エラーなどで）結果を戻す際に失敗しても、Oracle Virtual Directory で不完全な結果エラーが発行されることはありません。結果がクリティカルなアダプタで結果を戻す際に失敗すると、Oracle Virtual Directory で不完全な結果エラーが発行され、その他すべてのアダプタからの結果が抑止されます。	値 : true の場合アダプタはクリティカルで、false の場合はクリティカルではありません。partial の場合結果は戻ってくる途中ですが、検索結果コードは 11（管理制限超過）です。 デフォルトは true です。
routing-priority	priority	アダプタが検索される順序を指定します（数値が低いほど優先度は高くなります）。また、レベルは、複数のアダプタから問合せの値が戻される場合に、どのアダプタの結果を使用するかを指定する際にも使用されます。2 つ以上のアダプタで優先度の数値が同一で、複数のアダプタで同じ DN を問い合せている場合、最も早く定義された（数値が最も小さい）アダプタがクライアントに結果を戻します。	値 : 1 ～ 50 デフォルト : すべてのアダプタで指定なし

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
routing-levels	levels	指定された値により、アダプタ・ルートの何レベル下が検索ベースになるかが決定されます。たとえば、値が 0 の場合、検索ベースはアダプタ・ルートと同じである必要があります。値が 1 の場合、検索ベースはアダプタ・ルートまたはその 1 レベル下のいずれかになります。	デフォルト : -1（レベルには制限はありません。）
routing-bindadapters	bindadapters	カンマで区切られたアダプタ番号のリスト。先頭に ! （NOT）が付いている場合もあります。管理者は、同じ LDAP ディレクトリのかわりをしている複数のアダプタなど、共有資格証明の受渡しが可能なアダプタを特定できます。あるアダプタが別のアダプタの子である場合、管理者は除外も強制できるため、子アダプタの資格証明は親アダプタには渡されません。	デフォルト : アダプタは指定されていません。
routing-plugin	plugin	検索フィルタの制御に使用するプラグインを指定します。使用されている場合、include および exclude の設定を上書きします。	デフォルト : プラグインは指定されていません。
routing-dnpattern	dnpattern	アダプタ・ルートより下の DN 接尾辞の正規表現パターンを確認します。（式の右側に当たるアダプタ・ルートは、式の一部とみなされません。）（検索フィルタで実行される）include/exclude とは異なり、dnpattern はすべての操作で実行されます。検索中は検索ベースで実行され、その他すべての操作では DN で実行されます。	例 : m/(.*)uid=user[0-9]a$/ または (.*)uid=user[0-9]a$ デフォルト : パターンは指定されていません。

標準アダプタ - 一般プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
dbname	dbname	ローカル・データ・ファイルのパスおよびファイル名の接頭辞を定義します。標準アダプタが複数使用されている場合、各インスタンスのファイル名は一意である必要があります。	デフォルト : data/standardDB**N**。**N** は {0,1…} の中のいずれかです。
tlogsize	transactionLogSize	トランザクション・ログの切捨てサイズをバイト単位で指定します。未処理のトランザクションによって、ログのサイズがこのオプションの値を超えたとしても、データ・ストアに格納されず索引付けされていないエントリは、トランザクション・ログから削除されることはありません。	デフォルト : なし
cachesize	cacheSize	メモリーにキャッシュされるエントリ数を指定します。常に、最後にアクセスまたは書き込まれたエントリが含まれます。必要なメモリーの量は、エントリのサイズによって決定されます。	デフォルト : なし
passwordencrypt	passwordEncryption	自動パスワード暗号化を指定されたアルゴリズムに設定します。	範囲 : {ssha, sha, crypt, plain} デフォルト : なし
autordn	autoRdn	エントリの属性リストに RDN 属性が含まれないエントリの追加を許可します。	範囲 : {no, yes} デフォルト : なし
compact	compactDatabase	データベースの毎晩の圧縮を有効化します。	範囲 : {no, yes} デフォルト : なし
readonly	readOnly	true に設定すると、データベースは読取り専用モードに設定され、すべての変更操作が失敗します。	範囲 : {false, true} デフォルト : なし
backup-file	file	自動バックアップが配置されるファイルの名前。最新のバックアップは、名前に番号が付いていません。次に新しいバックアップは、最後に 0 が付きます。新しいバックアップ・ファイルが作成されるたびに、ファイルの番号は付けなおされます（たとえば、standardDBbck は standardDBbck0 になります）。ファイル拡張子 .zip は、自動的にファイル名に追加されます。	範囲 : {null,0,..max-2} デフォルト : backup/standardDBbck**N**.zip。**N** は {null,0,1..max-2} の中のいずれかです。
backup-hour	hour	1 時間ごとに自動的にバックアップが実行されます。	範囲 : {0..23} デフォルト : 23

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
backup-minute	minute	1 分ごとに自動的にバックアップが実行されます。	範囲 : {0..59} デフォルト : 59
backup-max	max	繰り返し使用されるバックアップ・ファイルの最大数。	範囲 : {1..7} デフォルト : 7

標準アダプタ - 索引プロパティ[1] [2]

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
substringIndex	substring	属性の値（(sn=*th) など）に一致する最後のサブストリングを検出するための検索フィルタで LDAP 属性が使用されている場合に、どの属性に索引を作成するかを指定します。先頭のサブストリングの検索は、order 索引を使用して処理されます。	要素として、様々な LDAP 属性のリストが含まれます。 デフォルト : sn と索引付けされた属性
search-unindexed	searchUnIndexed	true に設定すると、すべての検索で任意の属性を検索できるようになります。 **注意 :** このオプションを有効化すると、検索の全体的なパフォーマンスが低下します。	範囲 : {true, false} デフォルト : false
presenceIndex	presence	エントリ内の属性の存在（(sn=*) など）を検出するための検索フィルタで LDAP 属性が使用されている場合に、どの属性に索引を作成するかを指定します。	要素として、様々な LDAP 属性のリストが含まれます。 デフォルト : sn、cn、description、seealso、objectclass と索引付けされた属性
exactIndex	exact	属性値（(sn=Smith) など）の完全一致を検出するための検索フィルタで LDAP 属性が使用されている場合に、どの属性に索引を作成するかを指定します。	要素として、様々な LDAP 属性のリストが含まれます。 デフォルト : 索引付けされている属性はありません。
orderingIndex	ordering	属性値（(sn>=Smith) など）の順序付けされた一致を検出するための検索フィルタで LDAP 属性が使用されている場合に、どの属性に索引を作成するかを指定します。 この要素は、完全一致検索のパフォーマンスを向上し、先頭のサブストリングの検索（sn=Smi* など）も有効化します。orderingIndex としてリストされている属性タイプを、exact のリストに含めることはできません。	要素として、様々な LDAP 属性のリストが含まれます。 デフォルト : cn、sn、description、objectclass、ou、uid、mail と索引付けされた属性

[1] 2.0 の索引プロパティは backend.X.config."…Index"= で始まる adapters.prop の行によって認識されます。

[2] 10.1.4 では、searchUnIndexed=true でないかぎり、索引付けされていない属性を持つ最小単位を含む検索は実行されません。

LDAP アダプタ - 一般プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
remote base	remote Base	ルートが対応するリモート DIT の場所。	例 : dc=oracle、dc=com デフォルト : なし
remote hosts	host	各 host インスタンスは、別のサーバーが宣言したのと同一のデータが保持されているサーバーを示す host:port ペアを表します。複数のホストが宣言されている場合は、単一の論理バックエンドのフェイルオーバーまたはロード・バランシング（あるいは両方）を実行する構成であることを意味します。 各 host インスタンスには、次のデータを定義します。percentage: このサーバーに対して LDAP 操作が実行される割合。 port: ホスト名 /IP に対応するポート。 readonly: このサーバーが読取り専用であるかどうか。 percentage および readonly は、Oracle Virtual Directory 2.0 の LDAP プロキシ構成では使用されません。	複数の host インスタンスが許可されています。 デフォルト host: なし（例 : 127.0.0.1） percentage: 100 port: 389 readonly: false
secure	secure	設定するとリモート・ホストへのセキュアな SSL/TLS 接続を有効化します。	範囲 : {true, false} デフォルト : false
failover only	failover	true に設定すると、前述の 1 番目のホストが最初に試行され、接続が確立されるまで 2 番目、3 番目と続きます。 false（デフォルト）に設定すると、前述のリストされているホスト間で、新規の接続がラウンド・ロビン方式で繰り返し使用されます。	範囲 : {true, false} デフォルト : false

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
pass credentials	passcredentials	Oracle Virtual Directory により、バックエンド（リモート・ホスト）の LDAP サーバーに資格証明が渡された場合の処理を制御します。 binddn および bindpw（プロキシ・アカウントなど）にリストされている資格証明のみを使用する場合は、never（Oracle Virtual Directory 2.0 では 0）を選択します。 クライアントが指定した資格証明を常に使用する場合は、always（Oracle Virtual Directory 2.0 では 1）を選択します。 後続のすべての LDAP 操作のプロキシ・アカウントではなく、バインドに指定されたユーザー資格証明を使用する場合は、bindOnly（Oracle Virtual Directory 2.0 では 2）を選択します。	範囲 : {never, always, bindOnly} デフォルト : なし。ただし、通常は always に設定します。
binddn	binddn	初期バインド後に、passcredentials=never および passcredentials=bindOnly に送信される DN。	例 : cn=admin デフォルト : なし
bindpw	bindpw	初期バインド後に、passcredentials=never および passcredentials=bindOnly に送信されるパスワード。	デフォルト : なし
heartbeat interval	heartbeat Interval	hosts に指定されている各サーバーの可用性の、サーバーによる検証を許可します。パラメータには、検証が終了するまでの秒数を指定します。	デフォルト : なし。ただし、通常は 60（秒）に設定します。
timeout	timeout	プロキシ接続が失敗しているとみなすまでにサーバーが待機する時間をミリ秒単位で指定します。複数のリモート・ホストが指定されている場合は、小さな値（5000 ミリ秒）が適切です。ただし、代替ホストが指定されていない場合は大きな値（15000 ミリ秒など）が適切です。	デフォルト : なし

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
dnattrlist	dnAttributeList	発生元がリモート・サーバーで、変換が必要な一連の LDAP 属性。アクセス制御のサブジェクトとしてグループを使用している場合、DIT におけるグループの位置およびアダプタ変換によるグループへの影響を考慮することが重要です。 この構成により、外部 LDAP ディレクトリからのエントリの読取り中に、グループの識別名（DN）だけでなくこれらの属性も変換するよう Oracle Virtual Directory に伝達されます。この場合グループ・メンバーは、外部 LDAP ディレクトリの名前空間でも、仮想ディレクトリの名前空間内でも有効です。 複数のアダプタ（標準および JNDI）からのメンバーが含まれるグループが必要な場合には、グループを標準ディレクトリの名前空間（ローカル・ディレクトリなど）に配置する必要があります。 すべての属性が、この要素の下の <attribute> タグにリストされている必要があります。	例 : member、manager、uniquemember デフォルト : なし
pagesize	pageSize	ページ・コントロールをサポートしている LDAP サーバー（Active Directory など）の場合、LDAP アダプタは結果を制限されたページ・サイズにできます。	デフォルト : なし
rdnmap	（対応する要素なし）	Oracle Virtual Directory 2.0 では、RDN 属性マッピングに許可されていました。	
referrals	referrals	Oracle Virtual Directory で、リモート・サーバーから戻される参照に従うか無視するかを指定します。ignore に設定すると、参照はクライアントに戻されません。follow に設定すると、Oracle Virtual Directory は、クライアントのかわりに自動的に参照に従うために連鎖の実行を試行します。	デフォルト : ignore
maxpool size	maxPool Size	各リモート・サーバーに対して保持されている接続プールの最大サイズ。使用される現行の接続の平均値に設定される必要のあるハード・リミットです。	デフォルト : 10

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
quickfail	quickFail	true に設定すると、停止するとすぐに使用可能な最後のサーバーが削除され、ネットワーク・タイムアウトが原因で停止に時間がかかりそうな場合にクライアントを迅速に停止できます。false に設定すると、最後のサーバーはリストに残り、再試行が続行されます。	デフォルト : false
escape Slashes	escape Slashes	true または false に設定して、特定のディレクトリ間における変換時に、識別名の引用およびエスケープを制御します。	デフォルト : ディレクトリにより異なります（テンプレートに含まれます）。
（対応するプロパティなし）	plugins	pluginChains の下位要素。アダプタに関連付けられているプラグインを指定します。	

データベース・アダプタ - 一般プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
dbdriver	driver	使用される JDBC ドライバを表します。	例 : org.hsqldb.jdbcDriver デフォルト : なし
dburl	url	データベース URL を指定します。	例 : jdbc:oracle:thin:@prodds.vccs.edu:1521:prodds デフォルト : なし
dbuser	user	データベースに接続するためのユーザー名	デフォルト : なし
dbpass	password	データベースに接続するためのユーザー・パスワード	デフォルト : なし
dbmapfile	dbmapping	マッピング定義は、データベース・アダプタ構成内に含まれています。	例 : 製品マニュアルの「DB アダプタ構成」を参照してください。 デフォルト : なし
（対応するプロパティなし）	ignoreObjectClass OnModify	変更操作で objectClass 属性を無視します。	デフォルト : false
（対応するプロパティなし）	maxConnections	データベースの同時接続の最大数	デフォルト : なし

JoinView アダプタ - 一般プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
primary	primary	プライマリ・アダプタ（結合操作で最初に検索されるアダプタ）の識別子を指定します。	例 : adapter-0 デフォルト : なし
bind adapter	bindTo	バインド・アダプタ（プロキシ・アカウントが LDAP 操作でのバインドに使用されるアダプタ）の識別子を指定します。この要素の下に <adapter> タグのリストとして構成されます。	例 : adapter-1 デフォルト : なし
joinrule	joins joinrule	1 つ以上の joinrule セットのコンテナで、それぞれ次の項目があります。 jointo: 結合するアダプタ 例 : adapter-1 type: 結合のタイプ 例 : com.oracle.vde.join.OneToManyJoiner joinon: 結合する属性 例 : sn=sn	デフォルト : なし

NTLM アダプタ - 一般プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
domain	domain	マップ対象の NT ドメイン	デフォルト : なし
ntuser	ntuser	NT に接続するためのユーザー名	デフォルト : なし
ntpass	ntpass	NT に接続するためのパスワード	デフォルト : なし
（対応するプロパティなし）	userclass	各ユーザーの ObjectClass	デフォルト : なし

リスナー - 一般プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
type	listeners	リスナーの Java クラスを指定します。	範囲 : {ldap, ldapnio, web} デフォルト : なし
host	host	バインド先のアドレスを指定します。指定しないと、リスナーによりローカル・ホストにバインドされます。複数のアドレスを指定すると、リスナーはホームが複数あるホストと構成されます。	通常 : Blank(localhost)。任意の IP アドレスまたはドメイン名。 デフォルト : Blank(localhost)
port	port	リスナーがサービスを提供するポート。ポートでアクティブ化できるリスナーは、常に 1 つのみです。	範囲 : 任意のポート番号 通常 : LDAP: 389、web: 8080 または web: 8081 デフォルト : なし
secure	secure	すべての接続の暗号化に、サーバーで SSL を使用するかどうかを指定します。	範囲 : {true, false} デフォルト : なし
threads	threads	LDAP リスナーの場合、着信リクエストの処理のために同時に稼働しているアクティブなワーカー・スレッドの数を指定します。	範囲 : [1..] デフォルト : なし

リスナー - Web ゲートウェイ・サービス・プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
matchattrs	matchAttributes	ユーザー名が含まれている属性	デフォルト : なし
searchroot	searchRoot	ユーザーを検索するための Oracle Virtual Directory 内のルート	デフォルト : なし
(対応する属性なし)	certificateAttributes	資格証明として表示される属性	デフォルト : なし
(対応する属性なし)	userCacheLife	ユーザーの資格証明がキャッシュされるミリ秒単位の時間	デフォルト : なし
(対応する属性なし)	allowAnon	匿名でのアクセスを許可	デフォルト : true
(対応する属性なし)	htdocsRoot	すべての XSL 変換が保存されている Oracle Virtual Directory インストールの相対パス	デフォルト : htdocs
(対応する属性なし)	photoConfig	attributes : 画像を含む属性 height: 画像を表示する高さ width: 画像を表示する幅	デフォルト : なし
(対応する属性なし)	matchObjectClass	ユーザーを表す objectClasses のリスト	デフォルト : なし

リスナー - Web リスナー・サービスの基本実装

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
realm	securityRealm	securityRealm は HTAccessHandler と連携し、Web ゲートウェイをディレクトリ・サービス（Oracle Virtual Directory）と統合します。securityRealm はディレクトリに対するユーザー認証を実行し、識別名マッピングへの UID の提供など、他の操作のユーザー・コンテキストを追跡します。	デフォルト : なし

Oracle Virtual Directory - 一般プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
software. type	type	Oracle 製品の種類を示します。（カスタマイズ不可）	範囲 : {Oracle Virtual Directory, DFE, DSE} 通常 : Oracle Virtual Directory デフォルト : なし
software. version	version	リリースのリリース番号（カスタマイズ不可）	通常 : 10.1.4 デフォルト : なし
software. kit	kit	リリースのビルド番号（カスタマイズ不可）	通常 : NNNN（4812C など） デフォルト : なし
server. name	name	サーバー名（ユーザー選択）	通常 : Oracle Virtual Directory デフォルト : なし
company (license.prop file)	company	顧客の企業名	デフォルト : なし
product (license.prop file)	product	インストールされている Oracle 製品の名前	デフォルト : なし
expires (license.prop file)	expires	製品の有効期限の日付	デフォルト : なし
key (license.prop file)	key	製品のライセンス・キー	デフォルト : なし
（対応する属性なし）	rootSuffix	ディレクトリのルート接尾辞	デフォルト : なし
（対応する属性なし）	legacy	SystemConfig オブジェクトを介してアクセスされるプロパティ	デフォルト : なし
（対応する属性なし）	exitOnFatal	致命的エラーの発生時に Oracle Virtual Directory を終了するかどうかを指定します。	デフォルト : true

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
（対応する属性なし）	plugins	この Oracle Virtual Directory 構成と関連付けられているプラグインのリスト	デフォルト : なし
（対応する属性なし）	searchLimit / anonymous	匿名クライアントに戻されるエントリに対する制限	デフォルト : なし
（対応する属性なし）	searchLimit / authenticated	バインド・クライアントに戻されるエントリに対する制限	デフォルト : なし
（対応する属性なし）	persistentSearch	永続的な検索制御を許可するかどうかを指定します。	デフォルト : なし
（対応する属性なし）	adminService / host	管理サービスが実行される NIC	デフォルト : なし
（対応する属性なし）	adminService / port	管理サービスが実行されるポート	デフォルト : なし
（対応する属性なし）	adminService / secure	管理サービスが実行されるセキュア・ポート	デフォルト : なし
（対応する属性なし）	adminService / groupUrl	ルート・ユーザーの動的グループ	デフォルト : なし
vde.listeners.file	listeners	Oracle Virtual Directory がインストールされているディレクトリからの相対パスとして表されているリスナー・ファイルの場所	通常 : conf/listeners.xml デフォルト : ""
vde.schema.std	schema / location	スキーマ・ファイルの場所	通常 : conf/schema.core.xml デフォルト : ""
vde.schemacheck	schema / check	スキーマを確認するかどうかを指定します。	範囲 : {true, false}（必須） 通常 : true デフォルト : なし
vde.aclcheck	acls / check	ACL のスキーマを確認するかどうかを指定します。	範囲 : {true, false}（必須） 通常 : true デフォルト : なし
vde.aclfile	acls	ACL ファイルの場所	通常 : conf/acls.xml デフォルト : ""
vde.rootuser	rootuser / name	仮想ディレクトリの管理者の DN	デフォルト : なし
vde.rootpw	rootuser / password	仮想ディレクトリの管理者のパスワード	デフォルト : なし
vde.debug	logging / details-level	Oracle Virtual Directory の操作、レスポンス、警告およびデバッグ情報を反映する、生成されるデバッグ情報のレベル	範囲 : {Off, Fatal, Error, Warn, Info, Debug, Dump} 通常 : warn デフォルト : なし
vde.tls.keystore	tls / keystore	キーストア・ファイルの場所。ターゲット・アダプタとの TLS/SSL の安全な通信とともに使用されます。	通常 : conf/vde.keys デフォルト : なし

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
vde.tls.pass	tls / password	Oracle Virtual Directory でキーストアとともに使用される必要のあるパスワード	デフォルト : なし

Oracle Virtual Directory - ロギング・プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
vde.logfile	logging / file	レベル details-level における出力が表示されるファイルの場所	通常 : log/vde.log デフォルト : ""
vde.logconsole	logging / logConsole	クライアント・コンソールのログ・ファイル情報を表示するかどうかを指定します。	範囲 : {true, false}（必須） 通常 : true デフォルト : なし
vde.access logfile	logging / accessFile	アクセス・ファイルの場所。ファイルには、バインド、検索などのすべてのクライアント・アクセス操作が記録されます。	通常 : log/access.log デフォルト : ""
（対応する属性なし）	logging / enabled	accessFile にアクセス情報を記録するかどうかを指定します。	範囲 : {true, false}（必須） 通常 : true デフォルト : なし
（対応する属性なし）	log4jConfig	Log4j 構成ファイルの場所	デフォルト : ""

Oracle Virtual Directory - レプリケーション・プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
Vde.changelog	replicas / changelog / enabled	レプリケートされたサーバーで使用され、変更ログを作成するかどうかを指定します。	10.1.4 では非推奨
Vde.changelog.suffix	replicas / changelog / suffix	レプリケートされたサーバーで使用され、問合せ中の変更ログへのアクセスに接尾辞が使用されます。	10.1.4 では非推奨
vde.changelog.file	replicas / changelog / file	レプリケートされたサーバーで使用され、変更ログ・ファイルの場所を指定します。	10.1.4 では非推奨
vde.replicas	replicas / changelog / replicationFile	レプリケートされたサーバーで使用され、レプリケーション承諾構成を指定します。	10.1.4 では非推奨
replica.x.base	replicas / replica / base	特定のレプリケーション承諾の一部としてレプリケートされるベース。dc=test,dc=com など。	デフォルト : なし
replica.x.masterid	replicas / replica / masterid	メイン・サーバーの構成ファイルに表示される際のマスター・サーバーの論理名。	デフォルト : なし

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
replica.x.masterurl	replicas / replica / masterurl	コンシューマ・サーバーにより、マスター・サーバーへの接続に使用される LDAP URL	デフォルト : なし
replica.x. consumerid	replicas / replica / consumerid	この承諾に関連付けられているコンシューマの論理名。サーバー構成のコンシューマ名に一致する必要があります。	デフォルト : なし
replica.x. hostname	replicas / replica / server	コンシューマ・サーバーのホスト名または IP アドレス	デフォルト : なし
replica.x. port	replicas / replica / port	Oracle Virtual Directory のコンシューマ・サーバーへの接続に使用されるポート番号	デフォルト : なし
replica.x. secure	replicas / replica / serverSecure	コンシューマ・サーバーへの接続に SSL を使用するかどうかを指定します。	デフォルト : false
replica.x. binddn	replicas / replica / binddn	コンシューマに対するマスター自体の認証に使用されるユーザー名。コンシューマへの書込みを許可される唯一の DN になります。	デフォルト : なし
replica.x. bindpw	replicas / replica / bindpw	コンシューマ・サーバーへの接続に使用されるパスワード	デフォルト : なし
replica.x. name	replicas / replica / name	特定のレプリカに対するレプリケーション承諾の名前	デフォルト : なし

Oracle Virtual Directory - 割当て制限プロパティ

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
vde.quota. check	quota / active	割当て制限マネージャがオンかオフかを示します。	範囲 : {true, false}（必須） 通常 : false デフォルト : なし
vde.quota. exemptips	quota / exemptip	割当て制限チェックから除外される IP アドレス	通常 : 127.0.0.1 デフォルト : なし
vde.quota. exemptsubjects	quota / exemptsubjects	割当て制限チェックから除外されるサブジェクト（DN）	例 : cn=Directory Manager デフォルト : なし
Vde.quota.max. opspercon	quota / opspercon	1 接続当たりの最大操作数	デフォルト : -1
Vde.quota. max. connections	quota / connections	最大接続数	デフォルト : -1
Vde.quota. max. conpersubject	quota / consPerSubject	1 サブジェクト（DN）当たりの最大接続数	デフォルト : -1

2.0 のプロパティ	10.1.4 の要素	定義	10.1.4 の使用上の注意事項
Vde.quota.max.conperip	quota / consPerIp	1IP アドレス当たりの最大操作数	デフォルト : -1
Vde.quota.period	quota / ratePeriod	除外されない各 IP/ サブジェクトを評価する期間。ミリ秒単位で評価されます。	デフォルト : -1

B

バンドル・プラグイン

この付録では、Oracle Virtual Directory に含まれるプラグインを説明します。

概要

Oracle Virtual Directory には、複数のプラグインが含まれます。これらのプラグインにより、特別な関数やマッピングを実行する Oracle Virtual Directory の機能が拡張されます。

> **注意：** Oracle Virtual Directory には、ここには記載されていないその他のプラグインも含まれます。詳細は、オラクル社カスタマ・サポート・センターに連絡してください。オラクル社によって配布されたプラグインではない場合、詳細は、プラグインの作成者 / 販売元に問い合せてください。

汎用目的のプラグイン

この項では、次に示す汎用目的のプラグインを説明します。

- Caching
- Dynamic Groups
- Transaction Logger/Dumper
- Objectclass Mapper
- Dynamic Entry Tree
- Flat Tree

「Microsoft Active Directory および Microsoft ADAM」の項では、INetOrg 形式のスキーマと Microsoft スキーマとの間でのスキーマ変換に関連するプラグインを説明しています。

Caching プラグイン

Caching プラグインは、Oracle Virtual Directory にメモリー内キャッシュを提供します。LDAP クライアントが再利用できるよう、ソースからの問合せ結果をキャッシュできます。このプラグインを使用すると、問合せが何度も繰り返されるアプリケーションのパフォーマンスを大幅に向上できます。

キャッシュ操作および構成を見なおすには、VE ロギング・レベルを 7 または 9 に設定し詳細を参照します。

Caching は通常のプラグインであるため、Oracle Virtual Directory 内の任意の場所でキャッシュを構成できます。グローバルに実行することも、単一アダプタのコンテキスト内で実行することも可能です。標準のプラグイン構成で使用可能なネームスペース・フィルタ処理を使用して、特定のネームスペースに制限することもできます。

キャッシュ・ヒット・ロジック

キャッシュは、問合せ結果を保存して後から使用できるようにします。同じユーザーにより問合せが繰り返され、同じ属性または属性のサブセットがリクエストされた場合、Oracle Virtual Directory でソースから情報を取得するかわりに、キャッシュが結果を戻します。

ユーザー間でキャッシュ・ヒットを共有するようプラグインを構成することもできます。

> **重要：** passcredentials がバックエンド・ソースに渡されず、セキュリティの実行が Oracle Virtual Directory のみによって管理されている場合以外は、ユーザー間でキャッシュ・エントリを共有しないでください。より強力な権限を持つユーザーのキャッシュ結果にアクセスする可能性があり、許可されていない情報を表示することも可能であるため、ユーザー間でキャッシュ・ヒットを共有する際には慎重に計画を立ててください。

メモリー管理

キャッシュはプラグインにより定期的に参照され、有効期限の切れた結果や、前の変更トランザクションにより無効化されたエントリがないか確認されます。キャッシュ割当て制限を超えた場合には、プラグインにより、使用頻度が最低（LRU）の問合せをパージすることによるメモリーの削除が試行されます。

構成パラメータ

size

一度にキャッシュされるエントリの最大数。（デフォルト : 10000）

maxresultsize

特定の問合せにキャッシュされるエントリの最大数。（デフォルト : 10000）

trimsize

最大キャッシュ・サイズを超えた場合に、キャッシュ・マネージャが調整する必要のある量。**注意 :** 必要な場合には、最初に有効期限の切れた問合せ、続いて使用頻度が最低の問合せをパージすることで調整が行われます。（デフォルト : 10000）

maximumage

キャッシュに保存される問合せ / エントリの秒単位の最大有効期間。（デフォルト : 600）

maintenanceinterval

キャッシュ・マネージャが期限切れの問合せの有無を確認する秒単位の間隔。（デフォルト : 60）

zeroresults

エントリを含まない問合せ結果をキャッシュするかどうかを示すフラグ（1 または 0）。（デフォルト : 0）

bysubject

サブジェクト間でキャッシュ結果を共有するかどうかを示すフラグ（1 または 0）。1 は、サブジェクト間で結果が共有されないことを意味します。（デフォルト : 1）

Dynamic Groups プラグイン

Dynamic Groups プラグインを使用すると、Oracle Virtual Directory で（動的グループと呼ばれる）`groupofuniquenames` および `groupofurls` の両方の LDAP オブジェクト・クラスを処理し、仮想の静的なグループ（または `groupofuniquenames` 等価）に変換できるようになります。プラグインにより、戻された LDAP オブジェクトが監視され、`memberurl` 属性が存在してオブジェクト・クラスが `groupofuniquenames` および `groupofurls` の**両方**のオブジェクトが検出されます。

検出された場合には、プラグインにより `memberurl` 値が自動的に処理され、`uniquemember` 属性に結果が追加されます。動的にオブジェクトが処理されることにより、管理者は、通常なら groupofurls オブジェクト・クラスがサポートされないアプリケーションとの互換性を維持しながら、静的メンバーと動的メンバーの両方が含まれるグループを定義できます。

```
C:¥>ldapsearch -D cn=admin -w manager -b ou=groups,ou=airius,o=yourcompany.com -s sub
"(memberurl=*)"

cn=test,ou=groups,ou=airius,o=yourcompany.com
cn=test
memberurl=ldap:///ou=accounting,o=yourcompany.com??sub?(&(objectclass=person)(objectcla
ss=organizationalperson))
objectclass=groupofuniquenames
```

```
objectclass=groupofurls
objectclass=top
uniquemember=cn=Paul Jacobs,ou=People,ou=Airius,o=yourcompany.com
uniquemember=cn=Wendy Verbaas,ou=People,ou=Airius,o=YourCompany.com

cn=TestCheck,ou=groups,ou=airius,o=yourcompany.com
memberurl=ldap:///ou=alt bind,o=yourcompany.com??sub?(&(userprincipalname=*))
objectclass=groupofuniquenames
objectclass=groupofurls
cn=TestCheck
```

上の例では、2 つのグループが戻されます。1 つ目のグループには、2 つの静的メンバーと、メンバーとなる特定のディレクトリ・サブツリーを定義する memberurl が含まれています。上の問合せはプラグインが無効化されたまま実行されています。

次の問合せで、プラグインが有効化されている際の結果を示します。

```
C:¥>ldapsearch -D cn=admin -w manager -b ou=groups,ou=airius,o=yourcompany.com -s sub
"(cn=test)"
cn=test,ou=groups,ou=airius,o=yourcompany.com
memberurl=ldap:///ou=accounting,o=yourcompany.com??sub?(&(objectclass=person)(objectcla
ss=organizationalperson))
objectclass=groupofuniquenames
objectclass=groupofurls
objectclass=top
cn=test
uniquemember=cn=Paul Jacobs,ou=People,ou=Airius,o=yourcompany.com
uniquemember=cn=Wendy Verbaas,ou=People,ou=Airius,o=YourCompany.com
uniquemember=cn=Vipi Velasquez,ou=accounting,o=yourcompany.com
uniquemember=cn=Preston Pena-Fernandez,ou=accounting,o=yourcompany.com
uniquemember=cn=Andreas O'Hara,ou=accounting,o=yourcompany.com
uniquemember=cn=Chitra Guenette,ou=accounting,o=yourcompany.com
...
uniquemember=cn=Jim Ward,ou=accounting,o=yourcompany.com
```

メンバーシップ・アサーションのテスト

ポリシー・サーバーなどの製品では、特定のユーザーがグループのメンバーであるかどうかの確認が必要な場合がよくあります。グループが大規模な場合や、動的グループである場合には非常にコストのかかる作業です。1 万、時には 100 万ものメンバーを計算して戻すのは、非常に負荷がかかります。

かわりに、Dynamic Groups プラグインを使用すると、cn および uniquemember フィルタ項目の有無を確認することで、メンバーシップ・テストの問合せを検出できます。存在する場合には、問合せは少し異なる方法で処理されます。プラグインにより、クライアントがメンバーシップ・アサーションのテストのみを要求しているかどうかが判断されます。プラグインにより結果が変更され、メンバーとしてテストされる単一のユーザーのみが戻されます。

```
C:¥>ldapsearch -D cn=admin -w manager -b ou=groups,ou=airius,o=yourcompany.com -s sub
"(&(cn=TestCheck)(uniquemember=cn=Jim Ward,ou=accounting,o=yourcompany.com))"
cn=TestCheck,ou=groups,ou=airius,o=yourcompany.com
memberurl=ldap:///ou=accounting,o=yourcompany.com??sub?(&(userprincipalname=*))
objectclass=groupofuniquenames
objectclass=groupofurls
cn=TestCheck
uniquemember=cn=Jim Ward,ou=Accounting,o=YourCompany.com
```

構成パラメータ

このプラグインには構成パラメータはありません。プラグイン・チェーンに追加するのみで有効化されます。

DumpTransactions

このプラグインは、各 LDAP 操作の全トランザクションのレコードを生成し、Oracle Virtual Directory コンソールのログに出力します。このプラグインは、任意のログ・レベルで実行するために構成されます。このプラグインは、ロジックが Oracle Virtual Directory システムを通過する際に、マッピングおよび統合作業を診断するのに特に便利です。DumpTransactions は、Virtual Directory のプロトコル・アナライザのようなものです。

構成パラメータ

loglevel

プラグインによりトランザクションが記録されるログ・レベル。（デフォルト : Debug）

ObjectClass Mapper

ObjectClass Mapper は、ある objectClass（`inetOrgPerson` など）を別の objectClass（`user` など）として認識させる一般的なマッピング・プラグインです。ディレクトリで特定の objectClass または属性がサポートされていない場合に、アプリケーションでその objectClass および属性が予期されている際に便利です。

ObjectClass Mapper プラグインは、（構成パラメータに基づいて）複数の操作を実行します。

- 属性マッピング
- オブジェクト・クラス・マッピング
- オブジェクト・クラスへの属性条件の追加
- 属性の削除
- 補助クラスのフィルタ処理
- アクティブ化および非アクティブ化の処理

構成パラメータ

注意 : 属性のマッピング関係が生じた場合、接頭辞 client- はクライアント側を、source- はデータ・ソース側を表します。たとえば、Active Directory サーバーの user を InetOrgPerson を表すようにマッピングした場合、AD はソース側で InetOrgPerson はクライアント側になります。

mapAttribute

client-Attribute=source-attribute という書式でマップされる属性。次に例を示します。
プロパティ :`mapAttribute`
値 :`uniqueMember=member`
この構成の複数インスタンスにより複数のマッピングが可能になります。

mapObjectClass

client-ObjectClass=source-ObejctClass という書式でマップされる objectClass。次に例を示します。

```
プロパティ :  mapObjectclass
値 :          inetOrgPerson=user
```

このオプションの複数インスタンスは複数のマッピングに使用されます。

addAttribute[-objectclassvalue]

追加プロセス中にユーザーに属性を追加します。特定のオブジェクト・クラスにのみ属性を追加するために、構成名にオプションのオブジェクト・クラス値が追加されます。
`userAccountControl` 属性を追加するには、次のようにします。

```
プロパティ :  addAttribute-user
値 :          userAccountControl=546
```

注意： 属性名を % で囲んで入力することにより、式の値側で属性値を参照できます。

filterAttribute[-objectclassvalue]

追加操作中に戻されるすべてのエントリから削除される属性のカンマ区切りのリスト。特定のオブジェクト・クラスの属性を除外するために、条件付のオブジェクト・クラス値がパラメータ名に追加されます。次に例を示します。

プロパティ： `filterAttribute`
値： `objectsid,memberof,samaccountname`

filterAuxiliaryClass

追加操作中に削除する必要のあるオブジェクト・クラスのカンマ区切りのリスト。たとえば、Microsoft Active Directory for Windows 2000© では、エントリの追加中に補助オブジェクト・クラスをリストすることはできませんが、Microsoft Active Directory および ADAM for Windows Server 2003© では補助クラスをリストできます。次に例を示します。

プロパティ： `filterAuxiliaryClass`
値： `person,myorgPerson`

activationAttribute

アプリケーションに、基礎となるディレクトリのユーザー・アクティベーション・システムに関する情報がない場合に使用されます。この構成により、ユーザー・アクティベーション・フラグが含まれている着信属性が、Oracle Virtual Directory に通知されます。その後、このフラグは、ディレクトリ固有の属性およびフラグにマップされます。次に例を示します。

プロパティ： `activationAttribute`
値： `myuseraccountcontrol`

activationValue

アクティブであるとマークされる必要のあるユーザーを示す **activationAttribute** の値。

deactivationValue

非アクティブであるとマークされる必要のあるユーザーを示す **activationAttribute** の値。

directoryType

ユーザーのアクティブ化を実行する際に使用するディレクトリのタイプ。使用できる値は SunOne、eDirectory、ADAM および ActiveDirectory です。次に例を示します。

プロパティ： `directoryType`
値： `ActiveDirectory`

Dynamic Entry Tree

Dynamic Entry Tree は、エントリのリーフ・ノードで検出される属性を使用して、仮想ディレクトリ・ツリー階層を生成する際に使用できる汎用目的のプラグインです。たとえば、アダプタに o=Airius.com のルートがあり、ユーザーが uid=scarter,ou=people,o=airius.com として表されている場合、ユーザーのエントリにあるデータを使用して、このプラグインにより新しい階層が挿入されます。たとえば、
`uid=scarter,ou=accounting,ou=people,o=airius.com` のようになります。

構成パラメータ

patterns

このプラグインは、初期化パラメータ patterns を使用して一致パターンを指定することにより構成されます。

構文 :

attr[=entryattr[(SUPPRESS | value)]],...

これは、各 DN コンポーネントで、等号の右側の属性に代入された値を属性名に含められることを意味しています。代入の必要がない場合には、属性のみがリストされます。基本的に、元の DN に一致するコンポーネントが参照されます。値が代入されている際には、entryattr に値がない場合の処理をさらに条件付けることもできます。SUPPRESS を指定してエントリを完全に抑制するか、entryattr の名前の後の大カッコ内にデフォルト値を指定できます。

| を使用して区切ることで、複数のパターンを定義できます。たとえば、次のようになります。

uid,ou=department(contract),ou | cn,ou=code(SUPPRESS)

このルールには 2 つのパターンがあります。1 つ目のパターンは、dn コンポーネントが uid,ou に一致するアダプタ・ルート（o=airius.com）の下のオブジェクトに一致します。そのため、uid=scarter,ou=people,o=airius.com がマッピングに選択されます。戻される際には、department 属性に値があるかどうかが確認されます。存在しない場合には、静的なテキスト contract が代入されます。検索時に、ベースが uid=scarter,ou=accounting,ou=people,o=airius.com に設定されている場合には、新しいベースが uid=scarter,ou=people,o=airius.com になるように検索が変更され、フィルタは追加の anded 項目 ou=accounting で変更されます。

2 つ目のパターンは、cn rdn コンポーネントを持つ o=airius.com の子オブジェクトに一致することが意図されています。一致した場合、エントリに属性 code があると、その値には cn=mygroup,ou=code12,o=airius.com が代入されます。属性 code が存在しない場合には、SUPPRESS キーワードがあるためエントリ結果は抑制されます。

Flat Tree

Dynamic Entry Tree と同様に、Flat Tree プラグインも仮想ディレクトリ・ツリーの動的マッピングを実行します。その名前が示すように、Flat Tree ではディレクトリ・ソースがフラット化されているため、すべてのエントリがアダプタ・ルートの下に直接表示されます。

このプラグインは、2 種類のデプロイ・モードで動作します。既存のアダプタの一部としてデプロイし、既存のネームスペースをフラット化できます。または、カスタム・アダプタ・デプロイの一部としてデプロイすることもできます。adapter パラメータを使用すると、アダプタにより、カスタム・アダプタのネームスペースの一部として表されている指定されたアダプタからデータがフェッチされます。この方法で構成すると、アダプタ・ルート・オブジェクトが定義されないことに注意してください。重複した親ノードを作成せずに、親アダプタの上に複数のアダプタをオーバーレイする場合に便利です。

構成パラメータ

criteria

criteria には、このプラグインを介して検索できるエントリを制限する LDAP フィルタを定義します。たとえば、criteria を (objectclass=user) と設定した場合、このプラグインを介して戻されるのは user オブジェクトのみです。

adapter

定義されていない場合、プラグインは、データは親アダプタを介して取得されるとみなします。定義されている場合、アダプタは、Oracle Virtual Directory 構成内の別のアダプタ名である必要があります。定義すると、プラグインにより別のアダプタからデータが取得され、そのエントリが親アダプタのルートにマップされます。このモードで実行すると、子エントリのみが戻され、ルート・オブジェクトは戻されません。

Microsoft Active Directory および Microsoft ADAM

Active Directory には、多くのアプリケーションで処理できない機能が複数あります。Oracle Virtual Directory には、アプリケーションに影響を与えず、アプリケーション自体の再コーディングや再構成もせずに、アプリケーションがそれらの機能を最大限に活用できるようにする複数のプラグインが同梱されています。これらのプラグインは、主に次のような問題に対応します。

多様な属性

Active Directory および ADAM の属性には 1000 を超える値があり、戻された値の範囲を含む名前が付けられて、一度に 1000 の属性が戻されます（Windows 2003 では 1500）。クライアントには、次の書式で範囲が戻されます。

```
member;1-1000: somevalue
```

次の 1000 のエントリを取得するためには、クライアント・アプリケーションが問合せを繰り返し、属性メンバー 1001-2000 をリクエストすることを把握している必要があります。このため、アプリケーションでは、その他のディレクトリ製品に比べて Microsoft Active Directory に特別な方法で対応する必要があります。

パスワードの更新

Microsoft Active Directory および ADAM には、LDAP 経由でのユーザー・パスワードの更新に特別なルールがあります。

- パスワードの更新はセキュアな SSL 接続経由でのみ行う。
- ユーザーが自分のパスワードを更新する場合は、同じ変更操作で新しいパスワードが変更追加に含まれている状態で、元のパスワードが変更削除に含まれている必要がある。
- 以前のパスワードがわからなくても、ユーザー・パスワードをリセットできるのは管理者のみ。
- Active Directory では userPassword 属性は使用せず、unicodePwd 属性（Unicode 形式）を使用する。

オブジェクト・クラス・マッピング

多くの LDAP ディレクトリでは、ユーザーやグループに `inetOrgPerson` および `groupOfUniqueNames` オブジェクト・クラスを使用します。Microsoft Active Directory では、Microsoft の Active Directory NOS 要件に固有の属性を持つ `user` および `group` オブジェクト・クラスが使用されます。

次に示す各プラグインには、アプリケーションでこれらの Microsoft Active Directory の機能を処理できるよう、コンポーネントが用意されています。最初の 2 つのプラグインは **Active Directory Ranged Attributes** および **Active Directory Password** で、特定の機能を個々に使用できます。一方、**InetAD** プラグインでは、**ObjectClass Mapper** プラグインと **Active Directory Ranged Attributes** および **Active Directory Password** の両方が結合され、1 つのプラグインで前述のすべての機能をより単純な構成で処理できます。

Active Directory Ranged Attributes

このプラグインにより、Microsoft Active Directory または ADAM によって範囲が指定された属性が検出され、自動的にすべての値が取得されます。値の範囲が属性名に追加されるため、LDAP アダプタの dn 属性構成オプションは適用されません。

> **ヒント：** このプラグインにより、アダプタのどの属性を dn 属性としてマークするかが決定され、適切なベース・マッピングが実行されます。このため、このプラグインは、**LDAP アダプタのアダプタ・プラグインとしてのみ構成されます**。

構成パラメータ

このプラグインには構成パラメータはありません。プラグイン・チェーンに追加するのみで有効化されます。

Active Directory Password

このプラグインは、使用しているアプリケーションが前述のパスワード更新ルールを使用するように作成されておらず、すべての操作において SSL 経由で Microsoft Active Directory または ADAM に接続するのが不都合な場合に、管理者を支援します。このプラグインを SSL に対応していないアダプタに構成し、プラグインで SSL 対応のアダプタを指定することにより、Active Directory のパスワード更新が inetOrgPerson ディレクトリのパスワード更新として機能します。

> **ヒント：** このプラグインは、LDAP プロキシ・アダプタに構成する必要があります（通常は Microsoft Active Directory）。

構成パラメータ

adapter

リクエストに `userPassword` 属性が含まれる場合に、このプラグインがリクエストを再ルーティングするアダプタの名前。指定されたアダプタでは、仮想ルートが現行のアダプタと同一で、ルーティングの可視性が Internal とマークされている必要があります。値が設定されていない場合には、現行のアダプタが使用されます。

mapPassword

true または false。パスワードを `unicodePwd` 属性に変換する必要がある場合は true で、不要な場合（ADAM）は false。（**デフォルト：true**）

InetAD

InetAD プラグインを使用すると、Oracle Virtual Directory で Microsoft Active Directory または ADAM ディレクトリ・サーバーに inetOrgPerson スキーマがあるように使用できます。構成に基づいて、属性やオブジェクト・クラスの名前を変更し、Active Directory のユーザーが必要な属性をすべて使用できるように属性を追加できます。このすべてが組み込まれたプラグインには、**Active Directory Password** および **Active Directory Ranged Attributes** プラグインの機能が含まれています。このプラグインを使用すると、構成パラメータなしで、Microsoft Active Directory の `user` または `group` を `inetOrgPerson` または `groupOfUniqueNames` オブジェクト・クラスとして使用できます。

注意： 属性のマッピング関係が生じた場合、接頭辞 client- はクライアント側を、source- はデータ・ソース側を表します。たとえば、Active Directory サーバーの user を InetOrgPerson を表すようにマッピングした場合、AD はソース側で InetOrgPerson はクライアント側になります。

構成パラメータ

mapAttribute

client-Attribute=AD-attribute という書式でマップされる属性。次に例を示します。
プロパティ： `mapAttribute`
値： `uniqueMember=member`
この構成プロパティの複数のインスタンスを複数のマッピングができるように定義できます。（デフォルト値：`uniqueMember=member`、`uid=samaccountname`、`ntgrouptype=grouptype`）

mapObjectClass

client-ObjectClass=AD-ObejctClass という書式でマップされる objectClass。次に例を示します。
プロパティ： `mapObjectClass`
値： `inetOrgPerson=user`
このオプションの複数インスタンスは複数のマッピングに使用されます。（デフォルト値：`groupOfUniqueNames=group`、`inetOrgPerson=user`）

addAttribute[-objectclassvalue]

追加プロセス中にユーザーに属性を追加します。特定のオブジェクト・クラスにのみ属性を追加するために、構成名にオプションのオブジェクト・クラス値が追加されます。
userAccountControl を追加するには、次のようにします。
プロパティ： `addAttribute-user`
値： `userAccountControl=546`
名前を % で囲んで入力することにより、別の属性値が式として代入されます。デフォルト構成は次のとおりです。
"`addAttribute-user: useraccountcontrol=544`"、"`addAttribute-group:samaccountname=%cn%`"、"`addAttribute-group: grouptype=-2147483646`"

filterAttribute[-objectclassvalue]

追加操作中に戻されるすべてのエントリから削除される属性のカンマ区切りのリスト。オブジェクト・クラスごとに属性を除外するために、オブジェクト・クラス値がパラメータ名に追加されます。

filterAuxiliaryClass

追加操作中に削除する必要のあるオブジェクト・クラスのカンマ区切りのリスト。Microsoft Active Directory for Windows 2000© では、エントリの追加中に補助オブジェクト・クラスをリストすることはできません。ただし、Active Directory および ADAM for Windows Server 2003© では、補助クラスをリストできます。（デフォルト：`person`、`organizationalPerson`）

sslAdapter

リクエストに `userPassword` が含まれる場合に、このプラグインがリクエストを再ルーティングするアダプタの名前。設定されていない場合には、現行のアダプタが使用されます。このオプションで指定されたアダプタには、このプラグインが構成されているアダプタと同じローカル・ベースがあり、ルーティングの可視性が Internal に設定されている必要があります。

mapPassword

true または false。パスワードを unicodePwd にマップする必要がある場合は true で、不要な場合（ADAM）は false。（デフォルト : true）

activationAttribute

アプリケーションに、基礎となるディレクトリのユーザー・アクティベーション・システムに関する情報がない場合に使用されます。この構成により、ユーザー・アクティベーション・フラグが含まれている着信属性が、Oracle Virtual Directory に通知されます。その後、このフラグは、ディレクトリ固有の属性およびフラグにマップされます。

activationValue

アクティブであるとマークされる必要のあるユーザーを示す **activationAttribute** の値。

deactivationValue

非アクティブであるとマークされる必要のあるユーザーを示す **activationAttribute** の値。

directoryType

ユーザーのアクティブ化を実行する際に使用するディレクトリ。使用できるディレクトリ・タイプ値は SunOne、eDirectory、ADAM および ActiveDirectory です。

C

Web ゲートウェイ

この付録では、DSML および XSLT でレンダリングされたディレクトリ・レポートを提供する Oracle Virtual Directory の HTML ベースを説明します。

概要

Oracle Virtual Directory は、DSML および XSLT でレンダリングされたディレクトリ・レポートを提供する HTTP ベースのゲートウェイ・サーブレットという方式で XML をサポートしています。この Web ゲートウェイには、静的な Web コンテンツを処理する機能もあり、複雑なディレクトリ・ホワイト・ページや委任管理機能の構成もできます。

注意： この付録で参照する DSML の機能は、URL ベースの操作に基づいて DSML 形式で XML の結果を戻す Oracle の URL ベースの問合せゲートウェイ（Oracle Virtual Directory 2.0 で導入）に基づいています。**DSMLv2 サービス**は、HTTP リスナーを介して別途提供されます。DSMLv2 のサポートに関する詳細は、構成に関する章を参照してください。

WebGateway リスナー

WebGateway リスナーは、DSML および XML が統合された強力な Web サーバーです。次のような機能があります。

- Oracle Virtual Directory ディレクトリ・サービスとの動的な統合
- 認証と認可が可能な LDAP 統合を使用した Apache のようなセキュリティ
- 出力を XHTML、DSML、WML または任意の XML 派生形式にレンダリングする動的な XSLT 変換
- 典型的な HTML ファイルなど静的なコンテンツのサポート
- 追加、変更および削除を含むすべての LDAP 操作をサポートするフォームベースの入力の拡張サポート
- jpegphoto または usercertificates などのバイナリ属性取得の拡張サポート
- Oracle Virtual Directory の DoS 保護メカニズムの統合サポート

XSLT リスナーに組み込まれている機能により、カスタムの委任管理機能やユーザー・セルフサービス・オプションだけでなく、セキュアなホワイト・ページ・サービスおよびディレクトリ・レポートの開発が可能です。

デモンストレーション・ディレクトリ・ブラウザ

デモンストレーション・ディレクトリ・ブラウザは Oracle Virtual Directory パッケージに含まれており、XSLT リスナーの XML サービスの機能とそれを使用するメリットを明らかにします。

このデモンストレーション・ディレクトリ・ブラウザは、複雑な製品で使用するツールとしては設計されていません。開発者が確認や特定の目的での拡張を行うサンプル・コードとして提供されています。このブラウザのソース・コードは、Oracle Virtual Directory のインストール・ディレクトリの htdocs ディレクトリにあります。

注意： オラクル社は、この製品のコーディング・レベルのサポートは行っておりません。開発支援を目的としたコンサルティング・サービス・プログラムでの支援を行っております。

WebGateway リスナーのアーキテクチャ

WebGateway は Jetty サーバーに基づいており、一般的なオープン・ソースの Apache AXIS および Tomcat サーバーで使用されます。このサーバーは、特別な目的の製品に埋め込むように設計された拡張可能なサーバーです。WebGateway は、HTTP GET および POST の両方のリクエストを受け入れる HTTP リスナーで構成されており、一連のハンドラを介してリクエストを渡します。

- **DoS ハンドラ**: すべてのリクエストが、(**vde.prop** に定義されている) DoS マネージャの割当て制限全体に対してサブジェクト名および IP アドレスで追跡されます。割当て制限を超えるイベントが発生した場合には、サーバーは原因であるクライアントに反応しなくなります。これにより、過剰にアクティブなクライアントまたは DoS 攻撃から Web サービスが保護されます。
- **LDAPUserRealm**: LDAPUserRealm は HTAccessHandler と連携し、WebGateway をディレクトリ・サービス (Oracle Virtual Directory) と統合します。LDAPUserRealm はディレクトリに対するユーザー認証を実行し、識別名マッピングへの UID の提供など、他の操作のユーザー・コンテキストを追跡します。
- **HTAccess ハンドラ**: このハンドラは、Apache のような Web サービス・アクセス制御を提供します。HTAccessHandler はリクエストされた各リソースを確認し、リソース・ディレクトリまたは親ディレクトリで **.htaccess** ファイルを検索します。何も検出されない場合は、**conf/htaccess.prop** に指定されたデフォルトのアクセス制御が使用されます。現行ユーザーのコンテキストが現行のリソースの参照を認可されていない場合には、HTAccessHandler により HTTP クライアントにアクセスが拒否されたというレスポンスが戻されます。ブラウザがユーザー ID およびパスワードを使用して応答すると、LDAPUserRealm サービスにより、認証目的での HTAccessHandler による参照が可能な LDAP ユーザー・プリンシパルが自動的に作成されます。
- **サーブレット・ハンドラ**: サーブレット・ハンドラは、**.xsl** で終わる URI リクエストまたは **/dsml** で始まる URI リクエストを、それぞれ WebGateway サーブレットまたは DSML サーブレットに送信します。WebGateway サーブレットでは XSLT 書式の結果を提供し、ディレクトリ問合せおよびエントリ変更操作の両方をサポートしています。DSML サーブレットには、単純なディレクトリ問合せおよび DSML 書式のレスポンス機能があります。
- **リソース・ハンドラ**: その他すべてのリクエストは、ドキュメント・リソース・リクエストとして扱われます。リソースがサーバーの **htdocs** ドキュメント・ルートに配置されている場合、ドキュメントは HTTP クライアントに戻されます。

DSML および XSLT サーブレットの LDAP 問合せパラメータ

XSLT および DSML サーブレットの問合せは同じように機能します。XSLT が DSML サーブレットと異なる点は、指定された XSL テンプレートを使用して結果をレンダリングする追加機能があることです。XSLT および DSML の問合せは次のような書式です。

```
http://localhost:8080/dir/file.xsl?base=o=myorg.com&scope=base&filter=objectclass=*
```

または

```
http://localhost:8080/dsml/dir/?base=o=myorg.com&scope=base&filter=objectclass=*
```

XSLT サーブレットは、**.xsl** で終わるファイル名 (**file.xsl** など) によってアクティブ化されます。DSML サーブレットは、**dsml** で始まる URI (**/dsml/dir** など) によってトリガーされます。DSML は、通常 htdocs ベース・ドキュメントとは関連付けられないため、ディレクトリ・パス・ベースです (一方、XSLT は常にスタイルシートを参照します)。

DSML サーブレットの `/dir/` は、セキュリティ・コンテキストを確立するために使用されていることに注意してください (ファイル名は無視されます)。一方、XSLT サーブレットの `/dir/file.xsl` は、セキュリティ・コンテキストおよび `htdocs/dir/` ディレクトリにある **file.xsl** のレンダリング・スタイルシートの両方を示します。

次のパラメータがサポートされています。

base=[dn]
検索が開始されるベースの識別名です。空の場合は NULL とみなされます。

filter=[searchfilter]
LDAP 標準の検索フィルタ。空の場合は (objectclass=*) とみなされます。

scope=[base|onelevel|sub]
検索の有効範囲。base は、base に指定されたエントリのみが検索されることを指定します（エントリを直接指定する場合に便利です）。**onelevel** は、検索で **base dn** のすぐ下の子エントリのみを確認することを指定します。**subtree** は、**basedn** のすべての子が検索されることを指定します。デフォルトでは、有効範囲は base に設定されています。

binddn=[userDN]
検索の実行基準となるユーザー識別名。空の場合は、デフォルトで匿名とみなされます。

password=[password]
binddn アカウントと関連付けられたパスワード。通常このパラメータは、binddn とともに指定されます。

xsl=[xsl-file]（非推奨）
このパラメータは、XSLT リスナーにのみ適用されます。DSML リスナーは無視します。指定されている場合、Oracle Virtual Directory のルート・インストール・ディレクトリに関連するファイルが参照されます（HTML ファイルは **htdocs** ディレクトリに関連することに注意してください）。空の場合、デフォルト値は **conf/html.xsl** です。

[any]=[value]
XSLT リスナーは、ユーザーが指定するその他のパラメータを受け入れます。コンテキスト情報（**parentname="Home"** など）を指定するために、XSL スタイルシートにパラメータを渡す場合に便利です。

WebGateway リスナー

リソース・ハンドラ

静的コンテンツ（html、gif など）は、標準の Web（http）サーバー上のものとして WebGateway から提供されている場合があります。静的コンテンツに対するアクセス制御は、ドキュメント・ディレクトリまたは親ディレクトリにある **.htaccess** セキュリティ構成ファイルの存在に従って実行されています（後続の「セキュリティ・コンテキスト」を参照）。Oracle Virtual Directory インストールの **htdocs** ディレクトリにある静的ドキュメントは、メイン・サーバーのルートに直接マップされます。たとえば、**htdocs/index.html** は次のようにマップされます。

http://localhost:8080/index.html

任意のディレクトリのデフォルトの html ファイルは **index.html** です。

DSML サーブレット

適切な問合せとともに接頭辞 /dsml を使用することで、標準の DSML コンテンツに関してディレクトリを問い合せることができます。たとえば、次のようになります。

http://localhost:8080/**dsml**/*directory*?base=o=myorg.com&scope=base&filter=objectclass=*

リスナーが URL を受信すると、問合せは HTTP セッションのコンテキストで処理されます。ユーザー・セキュリティ・プリンシパルが存在しない場合、問合せは匿名の問合せとして処理されます。

問合せが処理されると、結果は DSML の標準形式で HTTP クライアントに戻されます。

注意 : セキュリティ・コンテキストは、**directory** をサーバー・ディレクトリ・コンテキストに置き換えることで確立されます。DSML のセキュリティ・コンテキストは、**.htaccess ファイル**（後で説明）を **htdocs** の適切なサブディレクトリに作成することで確立されます。

例 :

制限された問合せドメインを設定するには、**.htaccess ファイル**をディレクトリ htdocs/dsml/administrators に配置します。これは、http://localhost:8080/dsml/administrators への問合せは、アクセス・ファイル htdocs/dsml/administrators/.htaccess に指定されているアクセス要件に一致する必要があることを意味します。

WebGateway サーブレット（XSLT）

WebGateway サーブレットはサーバーの主力部分です。サーブレットには、コンテンツに関してディレクトリを検索し、指定された XSLT 変換ファイルに従って変換する機能があります。

XSLT サーブレットでは、ディレクトリ問合せおよび特別な処理をサポートする特別なコマンドの両方がサポートされています。

http://localhost:8080/search/results.xsl?params=...

ここで、.xsl ファイル接尾辞は XSLT サーブレットを表し、search/results.xsl は結果のレンダリングに使用される XSL 変換ファイルを指定しています。/search/results.xsl は、htdocs/search/results.xsl に自動的にマップされます。

WebGateway コマンドの詳細

WebGateway **コマンド**には、バイナリ属性の取得やフォーム処理などの追加の機能があります。XSLT コマンドは、バイナリ属性の取得、フォームベース検索およびエントリ変更の 3 つのタイプに分類されます。

バイナリ属性の取得

getcert および **getphoto** コマンドは、バイナリ属性を（DSML でエンコードされるのとは対照的に）ネイティブのバイナリ形式で取得する際に使用されます。getcert への呼出しにより、MIME タイプが **application/x-x509-email-cert** の http コンテンツが戻されます。同様に getphoto への呼出しにより、MIME タイプが **image/jpeg** のコンテンツが戻されます。

cmd=getcert

パラメータ **dn** および **attr** によって指定されている証明書を取得します。

例 :

http://localhost:8080/xslt/search/?**cmd=getcert**&dn=uid=jsmith,ou=people,o=airius.com&attr=usercertificate

cmd=getphoto

パラメータ **dn** および **attr** によって指定されているように jpegphoto を取得します。

dn=[distinguished name]

エントリの識別名を指定します。通常、コマンド・パラメータとともに使用されます。

attr=[attribute name]

getcert または getphoto コマンドに関して属性名を指定します。

フォームベース検索

LDAP のような XSLT/DSML 形式の検索を実行することは可能ですが、**cmd=search** オプションを使用するとわかりやすい検索フォームをより柔軟にサポートできます。このコマンドではフォーム値が使用され、コンテンツが標準のディレクトリ問合せに変換されます。

cmd=search

フォームベース検索をサポートします。関連するパラメータ、base、scope、kind、searchvalue および filterattrs を参照してください。XSLT は、フォームのパラメータから検索フィルタが自動的に作成される検索フォームのプログラミングをより簡単にするよう設計されています。その点を除き、このコマンドには XSLT 問合せと類似の機能があります。例 :

```
http://localhost:8080/xslt/search/results.xsl?cmd=search&base=o=airius.com&scope=sub&kind=all&filterattrs=cn,sn,givenname&searchvalue=smith
```

base=[distinguished name]

フォームベース検索コマンドの検索ベースを指定します。

scope=[base|one|sub]

検索の有効範囲を指定します。有効な値は base、one または sub です。

kind=[all|attrname]

特定の属性と、**filterattrs** パラメータに指定されているすべての属性のどちらに基づいて検索フィルタを作成するかを指定します。

filterattrs=[attr1,attr2,…]

検索する属性のリスト。

searchvalue=[user searchstring value]

ユーザーが **searchstring** 値として入力した値。

検索フォームの処理は、GET または POST として作成されます。次に、検索フォームによって生成された GET リクエストの例を示します。

```
http://localhost:8080/xslt/search/results.xsl?cmd=search&base=o=airius.com&scope=sub&kind=all&filterattrs=cn,sn,givenname&searchvalue=smith
```

このコマンドはサーバーによって解釈され、次の内容と同等のものとして処理されます。

```
http://localhost:8080/xslt/search/results.xsl?base=o=airius.com&scope=sub&filter=(|(cn=*smith*)(sn=*smith*)(givename=*smith*))
```

問合せの処理後、XSL ファイル `/xslt/search/results.xsl` を使用して結果が変換されます。

フォームベース・エントリの操作

変更コマンドを使用すると、ディレクトリ・エントリの変更に使用可能な html フォームを作成できます。このコマンドには、実際に LDAP ディレクトリ・エントリを追加、削除または変更する機能があります。

cmd=modify

クライアントがディレクトリ・エントリの変更をリクエストしたことを示します。変更のタイプはフィールド **changetype** に指定されます。

changetype=[add|modify|delete]

changetype には、ディレクトリ・エントリで実行する操作（追加、変更または削除）を指定します。操作で変更できるディレクトリ・エントリは 1 つのみです。

dn=[distinguished name]

追加、編集または削除されるエントリの識別名。

modify_attrs=[attr1,attr2,attr3]

フォームから処理される属性のリスト。このリストは、フォーム上のその他すべての属性パラメータ（attr1_type、attr1_action、attr1_new、attr1_confirm など）の検索に使用されます。

xslerror=[/dir/error.xsl]

エラー処理時にエラーを送信する xsl ファイルを指定します。エラー時に、指定されているすべてのフォーム・パラメータ、およびパラメータ result と message に XSL フォームが渡されます。これらは、エンド・ユーザーに表示する適切なエラー・メッセージを決定するために、**error.xsl** シートで解析されます。

ヒント： 次の項では、**attrx** は任意の有効な LDAP 属性です。変更操作フォームとともに使用されます。

---

attrx_type=[single|multi|photo|cert]

attrx で表される属性のタイプを定義します。single の場合は、**attrx** の最初の値のみが受け入れられます。cert または photo の場合は、バイナリ値はマルチパート・フォームの一部として配置されます（一度に追加できる値は 1 つのみです）。

attrx_required=[true|false]

true に設定されている場合、**attrx** に値が指定されていることがプロセッサにより確認されます。

attrx_delete=[value]

attrx から削除する値。
注意 : 1 つ以上の値を指定できます。空の値は無視されます。

attrx_modify=[value]

属性 **attrx** の変更に使用される値。**attrx_action** の値に応じて、この値が **attrx** に追加されるか、**attrx** のすべての値を置き換えます。

attrx_action=[add|replace]

attrx_modify の値を **attrx** に追加するか、既存の値をすべて置き換えるかを指定します。
注意 : changetype=add の場合、**attrx_action** は強制的に add になります。

rdn_attr=[attr]

add 操作中、**dn** は親エントリとみなされ、**rdn_attr** により **rdn** の値に使用する属性がプロセッサに伝達されます。

ヒント : 多くの場合、一部のフォーム・パラメータは非表示フィールドとして提供されます（attrs、changetype、objectclass など）。新規エントリの追加時には、必ず objectclass 属性（非表示フィールドとして提供）を指定してください。

新規エントリの追加時（changetype=add）には、attrx_modify=[value] 値のみが処理されます。エントリの削除時には、すべての **attrx** パラメータが無視されます。dn フォーム値を複数指定すると、複数の LDAP エントリを削除できます。

HTTP POST

HTTP POST は、HTTP GET に置き換えて使用できます。DoS 攻撃またはバッファ・オーバーフロー攻撃を防ぐために、**listeners.prop** ファイルには最大入力ファイル・サイズを設定する必要があることに注意してください。バイナリ属性（**jpegphotos** および **certificates** など）をアップロードする場合には、RFC 1867 によりフォームはマルチパート・フォームである必要があります。

HTTP GET

HTTP GET 操作を使用する場合には、すべてのパラメータが URL として渡されることに注意してください。メリットの 1 つは、URL をブックマークできることです。デメリットの 1 つは、URL に binddn などの機密情報が使用されることです。このため、匿名以外の問合せには POST を使用します。

GET を使用する場合には、アンパサンド（&）および空白を適切にエスケープして URL 要件に準拠する必要があります。これには javaScript を追加して、発行前のフィールド・パラメータを変更する必要があります。一般的に、アンパサンド（&）は & に置き換える必要があります。

セキュリティ・コンテキスト

HTAccessHandler を使用すると、サーバー内にセキュリティ・コンテキストを定義できます。セキュリティ・コンテキストでは、サーバー上の特定のリソース・ディレクトリへのアクセスを許可する IP アドレス、サブジェクトまたはグループを定義します。セキュリティ・コンテキストは、**htdocs** ディレクトリまたは任意のサブディレクトリに **.htaccess** セキュリティ構成ファイルを作成することで確立できます。各 **.htaccess** ファイルは、そのディレクトリまたはその下の子ディレクトリに適用されます。デフォルトのセキュリティ・コンテキストは、**conf/htaccess.prop** ファイルを編集して定義できます。直下のディレクトリまたは最も近い親の **.htaccess** ファイルは、任意のリソースに適用されるファイルです。

.htaccess ファイルの要件

- 平文ファイル
- 有効なディレクティブではないコンテンツには接頭辞としてコメント（#）文字が必要
- **.htaccess** ファイルのディレクティブは現行または子ディレクトリにのみ適用
- サブディレクトリに **.htaccess** ファイルが含まれる場合は親の **.htaccess** ファイルを上書き

.htaccess ファイルのディレクティブ

AuthUserFile [filename]

ユーザーがテキスト・ファイルから認証されることを示します。このファイルには、CRYPT ハッシュで暗号化されたユーザーおよびパスワードのリストが含まれます（後続の **htpasswd** の例を参照してください）。

AuthUserLDAP [base]

ユーザーが Oracle Virtual Directory ディレクトリ・サーバーから認証されることを示します。**base** は、認証されたユーザーの検索ベースの識別名を指定します。

AuthGroupFile [filename]

ユーザーが filename によって指定されているグループ・ファイルのメンバーである必要があることを示します（後続の **htgroup** ファイルの例を参照してください）。

AuthName [name]

認証レルムの名前を設定します。ユーザーへのチャレンジに表示されるテキストです（ディレクトリ・ブラウザなど）。

<Limit>

[restrictions]

</Limit>

リソース制限ブロックを定義します。制限はトークン **<Limit>** と **</Limit>** の間に配置されます。

リソース制限

リソース制限は、上に示したように **<Limit>** 文の間に配置されます。有効な制限は次のとおりです。

satisfy [all|any]

すべてまたはいずれかの要件を満たす必要があるかどうかを指定します。

require [user|valid-user| group|ldapgroup] [entities]

ユーザーが次に示すいずれかのメンバーである必要があることを示します。

- user: AuthUserFile または AuthUserLDAP のメンバー
- valid-user: 任意の検証済のユーザー
- group: AuthUserGroup のメンバー
- ldapgroup: ldap グループのメンバー

entities は、丸カッコ（）で囲まれている ldapgroup か、ユーザー ID 名またはグループの明示的なリストを指定します。

order [allow,deny|deny,allow|mutual-failure]

ルールを許可 / 拒否するためのルールの優先順位を指定します。

- allow,deny は、deny リストではなく allow リストに存在する必要があることを意味します。
- deny,allow または mutual-failure は、deny リストに存在しない場合、**または** allow リストに存在する場合に許可することを意味します。

allow from [ip address list]

リクエストを許可するための、空白で区切られた IP アドレスのリストを指定します。

deny from [ip address list]

リクエストを拒否する IP アドレスの空白で区切られたリストを指定します。次に、セキュリティ・コンテキスト・ファイルの例を示します。

.htaccess ファイルの例

```
# HTAccess example.
#

AuthName "Airius Directory Browser"
AuthType Basic
#AuthUserFile /etc/htpasswd
AuthUserLDAP ou=People,o=Airius.com
AuthGroupFile /etc/htgroup

<Limit>
satisfy any
require ldapgroup (cn=a group,ou=groups,o=Airius.com)
require ldapgroup (cn=alternate group,ou=groups,o=Airius.com)
order deny,allow
deny from 192.168.0.3 192.168.0.4
</Limit>
```

htpasswd ファイルの例

```
# <PRE>
# HTPassword example.
#
# Passwords must be in CRYPT format
#

admin: adpexzg3FUZAk
tom: tofn8Rh1L.xhQ
dick: diF1tp4K2rvw.
harry: haVyKPUXAHTjA
```

htgroup ファイルの例

```
# HT Group example.
# Place uid values after group name
#

Users: tom dick harry jturner

TestGroup: tom harry

AdminGroup: admin
```

XSL スタイルシート・テンプレートの使用方法

動的フォームを作成するための XSLT サーブレット問合せの使用

XSLT 問合せをスタイルシートと組み合せて使用し、クライアント用の編集フォームを作成します。

例 :

```
http://localhost:8080/manage/edit.xsl?base=ou=Atlanta,o=myorg.com&scope=base&filter=objectclass=*&mode=edit
```

上の URL により、サーバーがディレクトリ・エントリ ou**=Atlanta,o=myorg.com** を取得し、xsl フォーム **/manage/edit.xsl** に渡します。xsl フォーム内で、結果のレンダリング方法を決定するパラメータ mode を検索します。この場合 **edit** は、問合せによって戻されたオブジェクトを編集できるようにフォームをレンダリングする必要があることを意味しています。別のモード **addchild** は、組織への子エントリの追加に使用できるフォームを準備するために使用されます。

mode は、開発者が作成できるパラメータです。XSL の場合、mode は、様々なレンダリング・サブルーチンに対して異なるロジックを存在させるために、テンプレートで頻繁に使用される特別なパラメータです。デフォルトの出荷デモンストレーション・アプリケーションでは、**htdocs/xslt/secure/admin.xsl** の 1 つのスタイルシート内に複数モードの編集（表示、追加、変更、削除）が用意されています。

XSL の document() および import/include コマンドのサポート

Oracle Virtual Directory の XSLT サーブレットのプロセッサでは、**document()** コマンドがサポートされています。完全なドキュメント **URL** または**ファイル仕様**を指定します。ディレクトリまたはドキュメントのみを指定すると、**htdocs** ドキュメント・ルートへの関連とみなされます。

たとえば、ファイル **vdexsl.xml** を使用して、ldap 属性名をよりわかりやすい名前に変換するとします。

```
<xsl:variable name="lname"
select="translate(@name,'ABCDEFGHIJKLMNOPQRSTUVWXYZ','abcdefghijklmnopqrstuvwxyz')"/>
<xsl:variable name="prettyname"
select="document('xslt/vdexsl.xml')/vdexsl:vdexsl/vdexsl:fields/vdexsl:attr[@name=$lnam
e]/vdexsl:displayname"/>
```

document() コマンドを介して取得されたファイルは、パフォーマンス上の理由から自動的にキャッシュされます。

同様に、**xsl:import** および **xsl:include** コマンドもサポートされています。これらはグローバル・サブルーチンなどを XSL テンプレートにインポートするために使用できます。document コマンド同様、参照先は URL またはファイルです。デフォルトで、ファイル参照は **htdocs** ドキュメント・ルートへの関連です。たとえば、次のようになります。

```
<xsl:import href="http://192.168.0.2:8050/lib/globals.xslt"/>
```

または

```
<xsl:import href="lib/globals.xslt"/>
```

注意： Oracle Virtual Directory からスタイルシートを取得するために http を使用している場合、.xsl ファイル接尾辞を使用すると、XSLT サーブレットが介入し、xsl ファイルが問合せの一部であるかのようにレンダリングされる原因となります。別の Oracle Virtual Directory から未処理の XSL ファイルを取得する場合には、接尾辞名が .xsl 以外であることが必要です。上の例では、かわりにディレクトリ .xslt が使用されています。

XSL スタイルシートへのパラメータの受渡し

XSLT サーブレットにより、XSLT サーブレットに渡されたすべてのパラメータが XSL スタイルシートで使用可能になります。これにより、DSML 検索結果には存在しないコンテキスト情報を XSL スタイルシートで受信できるようになります。パラメータは、前述の有効な検索パラメータの一部またはこの目的で定義されたパラメータです。次に示すコード・フラグメントでは、パラメータ **parentname** およびベースが取得され、XSL コード内で変数として使用可能になっています。

```
…
<xml:output method="html" indent="yes"/>
<xsl:param name="parentname" select="'Root'"/>
<xsl:param name="base" select="'base'"/>
<xsl:template match="/">
<html>
…
```

xsl:param 文の select 部分は、存在しない場合にはデフォルト値を表すことに注意してください。

XSL のサンプル

XSL プログラミングの詳細は、XSLT のガイドを参照してください。次に示す例は、ディレクトリ・ナビゲーション・バーを作成するように設計されています。**コンテキスト**を確立するためにパラメータが 2 つ使用されています。パラメータ **parentname** および **base** は親エントリ名を記憶するために使用されているため、ツリーを上下に移動できます。親の親 DN は、現行の base 問合せパラメータから RDN を削除することで推測できます。

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

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"xmlns="http://www.w3.org/TR/REC-html40"
xmlns:dsml="http://www.dsml.org/DSML">
<xml:output method="html" indent="yes"/>
<xsl:param name="parentname" select="'parentname'"/>
<xsl:param name="base" select="'base'"/>

<xsl:template match="/">
<html>
<head>
<title>Oracle Browser Navigation Bar</title>
</head>
<body bgcolor="#99CCCC">
<xsl:choose>
<xsl:when test="$base = 'base'">
<xsl:variable name="rdn" select="'root'"/>
<xsl:variable name="rvalue" select="'Root'"/>
<xsl:variable name="parentdn" select="'base=&amp;scope=base'"/>
<FONT FACE="Verdana,Arial" size="-1">Root</FONT>
  </xsl:when>
<xsl:when test="$base = ''">
<xsl:variable name="rdn" select="'root'"/>
<xsl:variable name="rvalue" select="'Root'"/>
<xsl:variable name="parentdn" select="'base=&amp;scope=base'"/>
<FONT FACE="Verdana,Arial" size="-1">Root</FONT>
  </xsl:when>
<xsl:otherwise>
    <xsl:variable name="rdn">
     <xsl:choose>
  <xsl:when test="contains($base,',') = 'true'">
<xsl:value-of select="substring-before($base,',')"/>
  </xsl:when>
  <xsl:otherwise>
<xsl:value-of select="$base"/>
  </xsl:otherwise>
 </xsl:choose>
</xsl:variable>
<xsl:variable name="rvalue" select="substring-after($rdn,'=')"/>
    <xsl:variable name="parentdn">
     <xsl:choose>
  <xsl:when test="contains($base,',') = 'true'">base=<xsl:value-of
select="substring-after($base,',')"/>&amp;scope=onelevel</xsl:when>
  <xsl:otherwise>base=&amp;scope=base</xsl:otherwise>
 </xsl:choose>
</xsl:variable>
<a
href="/xslt?{$parentdn}&amp;filter=(objectclass=*)&amp;parentname={$parentdn}&amp;xsl=h
tdocs/Browser-navbar.xsl" target="leftFrame">
<FONT FACE="Verdana,Arial" size="-1"><xsl:value-of select="$rvalue"/></FONT></a>
  </xsl:otherwise>
</xsl:choose>
<table border="0" cellpadding="0" cellspacing="0">
<xsl:apply-templates select="dsml:dsml/dsml:directory-entries/dsml:entry"/>
```

```
</table>

</body>
</html>
</xsl:template>
<xsl:template match="dsml:entry">
<xsl:variable name="edn" select="@dn"/>
<xsl:variable name="evalue">
<xsl:choose>
<xsl:when test="contains(@dn,',') = 'true'">
<xsl:value-of select="substring-after(substring-before(@dn,','),'=')"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="substring-after(@dn,'=')"/>
</xsl:otherwise>
</xsl:choose>
</xsl:variable>
<xsl:if test="@dn = ''">
 <xsl:for-each select="dsml:attr[@name='namingContexts']/dsml:value">
   <tr><td><li><a
href="/xslt?base={.}&amp;scope=onelevel&amp;filter=(objectclass=*)&amp;parentname={.}&a
mp;xsl=htdocs/Browser-navbar.xsl" target="leftFrame"><FONT FACE="Verdana,Arial"
size="-1"><xsl:value-of select="."/></FONT></a>
   <FONT FACE="Verdana,Arial" size="-1">[<a
href="/xslt?base={.}&amp;scope=base&amp;filter=(objectclass=*)&amp;xsl=htdocs/results.x
sl" target="mainFrame">-></a>]</FONT></li>
   </td></tr>
</xsl:for-each>
</xsl:if>
<xsl:if test="@dn > ''">
<tr><td><li><a
href="/xslt?base={@dn}&amp;scope=onelevel&amp;filter=(objectclass=*)&amp;parentname={$e
value}&amp;xsl=htdocs/Browser-navbar.xsl" target="leftFrame"><FONT FACE="Verdana,Arial"
size="-1"><xsl:value-of select="$evalue"/></FONT></a>
<FONT FACE="Verdana,Arial" size="-1">[<a
href="/xslt?base={@dn}&amp;scope=base&amp;filter=(objectclass=*)&amp;xsl=htdocs/results
.xsl" target="mainFrame">-></a>]</FONT></li>
</td></tr>
</xsl:if>
```