Oracle® Fusion Middleware

Administrator's Guide for Oracle Virtual Directory

11*g* Release 1 (11.1.1)

E10046-03

December 2009

ORACLE®

Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory, 11*g* Release 1 (11.1.1)

E10046-03

Primary Author: Don Biasotti

Contributor: Mark Wilcox, Phil Hunt, Saurabh Shrivastava, David Loo, David Lin

Contents

3 Understanding Oracle Virtual Directory Routing

4 Understanding Oracle Virtual Directory Plug-Ins

5 Understanding Oracle Virtual Directory Mapping

9 Configuring and Managing the Oracle Virtual Directory Server

10 Managing Oracle Virtual Directory Server Processes

11 Creating and Managing Oracle Virtual Directory Listeners

12 Creating and Configuring Oracle Virtual Directory Adapters

Part III Advanced Administration

18 Customizing Oracle Virtual Directory

19 Configuring Oracle Virtual Directory for Integrated Directory Solutions

20 Oracle Communications Universal User Profile

Part IV Appendixes

A Comparing Oracle Virtual Directory 11g Release 1 (11.1.1) and 10g Releases (10.1.4.x)

List of Figures

Preface

The *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory* provides information on how to utilize and administer Oracle Virtual Directory in an enterprise directory infrastructure.

Audience

The *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory* is intended for Oracle Virtual Directory administrators and for network architects planning directory infrastructure. You should be familiar with either the UNIX operating system or the Microsoft Windows operating system in order to understand the command-line syntax and examples in this document. You also must be familiar with the Lightweight Directory Access Protocol (LDAP).

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request

process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Related Documents

For more information, see the following documents in the Oracle Identity Management 11*g* Release 1 (11.1.1) documentation set:

- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Tutorial for Oracle Identity Management*
- *Oracle Fusion Middleware Java API Reference for Oracle Virtual Directory*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- *Oracle Fusion Middleware User Reference for Oracle Identity Management*
- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Security Guide*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*

Note: For more information on integrating Oracle Access Manager and Oracle Virtual Directory, refer to the *Oracle Access Manager Installation Guide*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in This Guide?

This preface introduces the new and changed administrative features of Oracle Virtual Directory that are described in this guide, and provides pointers to additional information.

New Features for Release 11*g* Release 1 (11.1.1)

The following is a list of the new features included in Oracle Virtual Directory 11*g* Release 1 (11.1.1):

- Integration with Fusion Middleware Control

  Refer to "Fusion Middleware Control" on page 1-9 for more information.

- New, unified Oracle Directory Services Manager graphical user interface (GUI) for both Oracle Virtual Directory and Oracle Internet Directory

  Refer to "Oracle Directory Services Manager" on page 1-9 for more information.

- Integration with Oracle Application Server's audit infrastructure

  Refer to "Managing Oracle Virtual Directory Auditing" on page 17-3 for more information.

- New Quick Config Wizard to simplify Oracle Virtual Directory's setup for Oracle Access Manager

  Refer to "Configuring Oracle Virtual Directory for Oracle Access Manager" on page 19-1 for more information.

- Several new plug-ins, including:
  - Hide Entries By Filter Plug-In
  - Change User RDN Plug-In
  - UPN Bind Plug-In
  - Fork Join Plug-In
  - Dynamic Tree Plug-In
  - Virtual Member of Plug-In
  - OAM Policy Control Plug-In
  - Virtual Attribute Plug-in
  - User Management Plug-In
  - Changelog Plug-Ins

Refer to Chapter 4, "Understanding Oracle Virtual Directory Plug-Ins" for more information about each of these new plug-ins.

Part I

Understanding Oracle Virtual Directory Services

This part presents introductory and conceptual information about Oracle Virtual Directory. It contains the following chapters:

1 Understanding Oracle Virtual Directory

This chapter introduces you to Oracle Virtual Directory, its services and architecture, and includes the following topics

- What is Oracle Virtual Directory?
- Why the Enterprise Directory Is Not Enough
- Oracle Virtual Directory In Enterprise Directory Network Environments

1.1 What is Oracle Virtual Directory?

This topic provides an introduction to Oracle Virtual Directory and contains the following sections:

- Overview
- Features
- Functionality
- Architecture and Topology
- Oracle Virtual Directory in Oracle Fusion Middleware
- Oracle's Directory Services Portfolio

1.1.1 Overview

Welcome to Oracle Virtual Directory, an LDAP version 3 enabled service that provides *virtualized* abstraction of one or more enterprise data sources into a single directory view. Oracle Virtual Directory provides the ability to integrate LDAP-aware applications into diverse directory environments while minimizing or eliminating the need to change either the infrastructure or the applications. Oracle Virtual Directory supports a diverse set of clients, such as Web Applications and portals, and it can connect to directories, databases and Web Services as shown in Figure 1–1.

Figure 1–1 Oracle Virtual Directory Clients and Connectable Data Stores

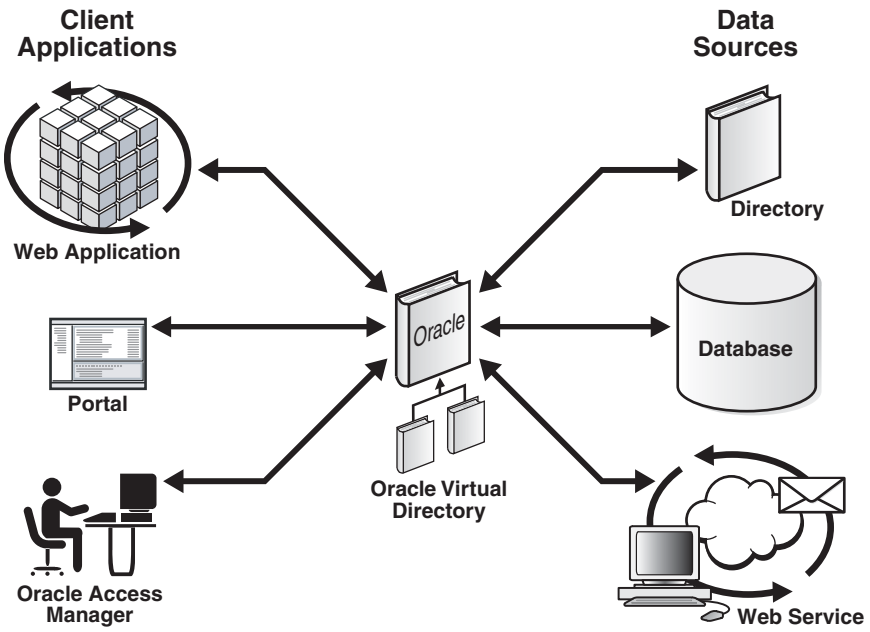
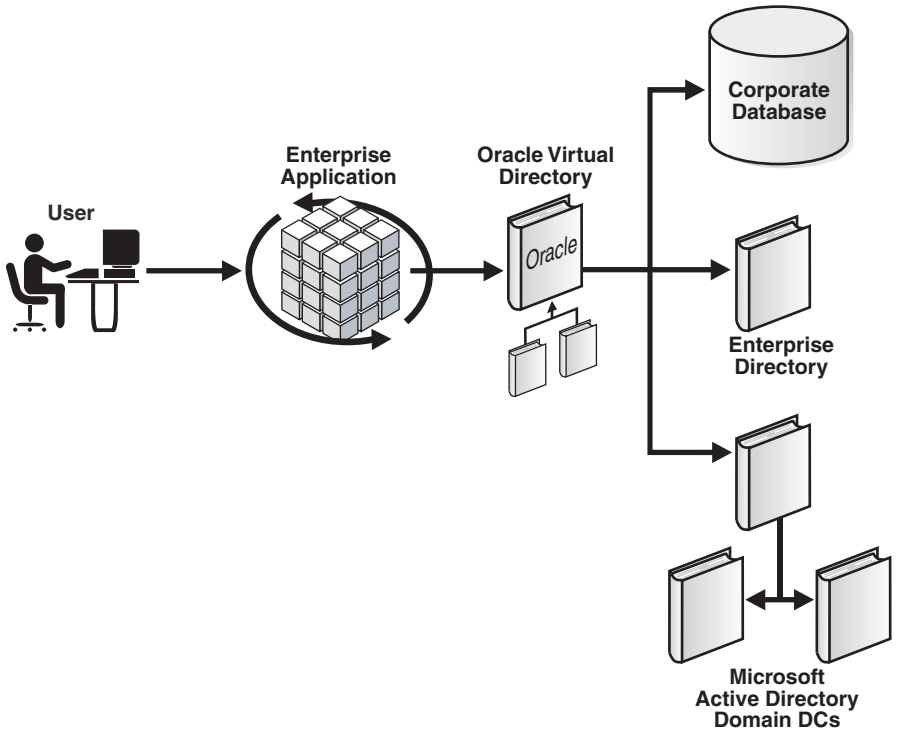


Figure 1–2 shows an example of an enterprise application used by all employees in a company. The application accesses directory information from three different sources and each contains a separate population of users, which is typical for many organizations due to corporate structure. For example, the Active Directory repositories contain only internal employee users, the single enterprise directory contains users from a different corporate division or business partner, and another set of users, such as external contractors, is contained in a relational database. As shown in the figure, Oracle Virtual Directory can be deployed to bring together the identity information from all three sources.

Figure 1–2 Directory Virtualization for Different User Populations



Oracle Virtual Directory hides the complexity of data location, format, and protocol from client applications, similar to a TCP/IP Internet network design based on switches and routers. Switches and routers handle the details of how to establish connections and protocols between different addresses on the network. Oracle Virtual Directory makes many directories appear to be one local repository in much the same ways that routers make the entire world appear like it's on your local network.

1.1.2 Features

The following is a list of some of Oracle Virtual Directory's key features:

Product Features

- LDAPv2/v3 support
- DSMLv2/SOAP support
- HTTP/XSLT Gateway support
- Low-cost configuration and maintenance
- Globalization features such as multi-byte character support and localized language translations
- Encryption and Strong Authentication with TLSv1 and SSLv3 support
- Can be deployed to function as a directory Proxy and Firewall
- Extremely small memory and hardware requirements
- Available on any platform where Java is supported
- Configurable Fail-Over and Intelligent Load-Balancing at the LDAP operation level

- Granular Access Controls based on IETF's Access Control Implementation Internet Draft
- Support for access to JNDI compliant directories and JDBC compliant databases
- Dynamic mapping of information and schema in multiple directories
- Intelligent Routing of LDAP Queries
- Denial of Service protection
- Overlapped namespace handling
- Multiple types of adapters for various deployments
- Extensible meta directory-like dynamic join features
- Local schema support
- Authentication of clients from joined directory, for example, from Active Directory
- Granular plug-in systems to support custom extensions
- Ability to compartmentalize information using dynamic views
- Native support for web services at both integration and data access layers

Business Features and Benefits

- Reduce implementation and administrative costs
- Maximize and extend your existing infrastructure investments
- Place all of your identity information under centralized management
- Improve security and compliance
- Unify multiple directories without synchronization
- Provide LDAP interface to non-directory data
- Combine data from multiple data-stores to create virtual entries
- Provide application specific views of directory information
- Expose Web Services as LDAP

1.1.3 Functionality

Oracle Virtual Directory answers the challenge of addressing today's enterprise directory needs by delivering the following:

Data Federation and Translation

Oracle Virtual Directory enables directory services access that crosses political and corporate boundaries by acting as a directory gateway that processes client requests and dynamically re-routes them to one or more existing directories—regardless of format, be it LDAP, RDBMS, or others. Oracle Virtual Directory presents a virtual directory hierarchy, or tree, to its clients and then assigns hierarchy branches of that tree to designated LDAP or RDBMS servers. Oracle Virtual Directory handles the issues of inter-directory security, protocol, and data translation so that LDAP clients assume that all information comes from a single trusted LDAP directory, the Oracle Virtual Directory.

Data Ownership

One of the least obvious—but most important—benefits of virtualization is data ownership. Organizations often create directories with specific purposes and

objectives in mind. When another organization wants to access data owned by the first organization, questions arise as to who ultimately owns the data and who controls it. Politics can occur when different parties wish to use and share information. Everyone acknowledges the value in re-using existing data, but re-using data brings up many care and control issues. Many organizations become very concerned when copies of the data they feel they own goes to other organizations or outside parties. Questions such as the following are sure to arise:

- Who is responsible for the data?
- Who will ensure its accuracy?
- Who will ensure its security and confidentiality?
- If the information is copied, how does the owning organization assure itself that the information is being used and controlled by the other party?

Virtualization through proxy technology solves many of these political problems by keeping data where it belongs—with the data's owner. At any time, the owner can restrict or eliminate access to this data. Additionally, the owner is free to revise this information at will and can be assured that partners are always working with the latest relevant information. Most importantly, by keeping information with the owner, the use of that information can be continuously monitored and controlled by the owner.

Oracle Virtual Directory provides this type of data ownership by not copying information. Information accessed by Oracle Virtual Directory occurs in real time, assuring the consumer and provider that the information is current, accurate, and authorized.

Flexible Security Domains

Oracle Virtual Directory enhances security by providing new security domain contexts. When deploying new business applications across multiple business organizations, identity and security can be complicated by the existence of multiple directory security infrastructures. As Microsoft Active Directory administrators know, having multiple windows infrastructures (sometimes called forests) is great for administration and performance, but has a downside in that there is no automatic trust between forests and no inter-forest global catalogue.

Oracle Virtual Directory creates a new transitive security context with fine-grained access controls built to support all IETF standards for access control, while supporting the IETF models for implementation. Oracle Virtual Directory is also designed to properly integrate with security restrictions from the source directories it proxies, resulting in a multi-layer or multi-domain security concept that gives administrators the ultimate security control.

Oracle Virtual Directory supports a wide array of authentication models. In addition to SSL/TLS (including StartTLS) and certificate-based authentication, Oracle Virtual Directory is able to use server-to-server authentication with proxied servers (authenticating itself), or alternatively is able to pass user context through to source directories. By providing user-context at the Oracle Virtual Directory and source directory, both directories can provide end-user contextual security control.

Secure Data Publication

Oracle Virtual Directory offers several data security features, for example:

- **SSL/TLS support**: Oracle Virtual Directory offers SSL/TLS capabilities that provide for secure communication sessions with LDAP clients. This allows you greater security by allowing Oracle Virtual Directory to be the trusted transport mechanism.

- **Transaction Cleansing**: Oracle Virtual Directory is based on a protocol conversion engine, which means that it deconstructs every query, recompiling and assessing validity before transmission to trusted proxied directory sources. This protects source LDAP servers from malformed or unauthorized queries. After cleaning the garbage requests, Oracle Virtual Directory is able to protect limited resources from exposure to huge loads from malicious attacks by providing the ability to set limits on items such as:
  - Maximum operations per connection
  - Maximum concurrent connections
  - Maximum total connections in a specified period for a particular subject
  - Maximum total connections in a specified period for a particular address
- **Access Control**: Oracle Virtual Directory implements its own access controls and provides filtered access to internal proxied directory data.

Application to Directory Integration

A directory is only useful if the applications it serves can gain access to the data it needs in a form that has consistent formats or schema. But the typical enterprise environment contains a myriad of directory repositories with different schema, namespace, and data designs.

In addition to providing a secure bridge to existing directory information, Oracle Virtual Directory provides functionality like a meta-directory to translate and transform data in real time, enabling administrators to easily normalize differences in data found between different organizations and directory infrastructures. The resulting virtualized directory view contains all the directory information an application needs to run, without needing drastic changes or integration technology to be built into the application.

Flexible Deployment Options

Oracle Virtual Directory provides flexible deployment options that allow it to be embedded with commercial-off-the-shelf applications by developers and business application developers. Additionally, Oracle Virtual Directory can be deployed by a corporate IT department as a shared directory service distribution network.

High Availability Support

Oracle Virtual Directory offers multiple high availability capabilities, including:

- **Fault Tolerance and Fail-Over**: Oracle Virtual Directories provide fault tolerance in two forms:
  - they can be configured in fault tolerant configurations
  - they can manage flow to fault tolerant proxied sources

  Multiple Oracle Virtual Directories can be quickly deployed simply by copying, or even sharing configuration files. When combined with round-robin DNS, redirector, or cluster technology, Oracle Virtual Directory provides a complete fault-tolerant solution.

  For each proxied directory source, Oracle Virtual Directory can be configured to access multiple hosts (replicas) for any particular source. It intelligently fails over between hosts and spreads the load between them. Flexible configuration options allow administrators to control percentages of a load to be directed towards specific replica nodes and to indicate whether a particular host is a read-only replica or a read-write server (master). This avoids unnecessary referrals resulting from attempts to write to a read-only replica.

- **Load-Balancing**: Oracle Virtual Directory was designed with powerful load balancing features that allow it to spread load and manage failures between its proxied LDAP directory sources.

  Oracle Virtual Directory's virtual directory tree capability allows large sets of directory information to be broken up into multiple distinct directory servers. Oracle Virtual Directory is able to recombine the separated data sets back into one virtual tree by combining the separate directory tree branches.

  If you have multiple LDAP servers for a particular source, the Oracle Virtual Directory LDAP Adapter can load-balance and fail-over for these servers on its own. This load-balancing and fail-over happens transparently to the client and does not require any additional hardware or changes to the client connecting to Oracle Virtual Directory.

  The Database adapter supports load-balancing and fail-over if the underlying JDBC driver provides this functionality. Additionally, Oracle Virtual Directory is certified for use with Oracle Real Application Clusters.

  Oracle Virtual Directory Routing also provides load-balancing capabilities. Routing allows search filters to be included in addition to the search base to determine optimized search targets. In this load-balancing approach, Oracle Virtual Directory automatically routes queries to the appropriate virtualized directory sources enabling the ability to work with many millions of directory entries.

  Note: Oracle Virtual Directory's value is as a virtualization and proxy service, not as a directory store. If you need a highly available directory storage system, Oracle recommends using Oracle Internet Directory.

Custom Application Programming Interfaces

Oracle Virtual Directory provides the following three main areas of extensibility, allowing customers and consultants to enhance the functionality of Oracle Virtual Directory to meet specific business or technical integration needs:

- O**racle Virtual Directory Plug-ins**: Oracle Virtual Directory provides a flexible plug-in framework modeled on Java Servlet Filters. You can use plug-ins to provide custom logic as part of a transaction or simply to connect to a custom data source. You can insert plug-ins globally or only for specific adapters. You can change the ordering of plug-ins and they can be isolated to particular types of transactions. Oracle Virtual Directory's management tools provide wizards for creating new plug-ins along with examples that can be used to get started quickly.
- **Custom Joiners**: The Oracle Virtual Directory Join View Adapter is based on an extensible model known as Joiners. You can develop Custom Joiners to provide different joiner behaviors. Joiners provide functions such as mapping, joining, and pre- and post-handler event handling. You can write Custom Joiners to provide simple entry level joins, or extended Joiners to provide complex join logic, transaction handling, and rollback capability.
- **Web Gateway**: Oracle Virtual Directory includes a customizable DSML/XSLT based gateway that provides basic web server support based on the Apache web server model that supports static HTML and XSLT rendered content. The gateway includes a directory-enabled interface allowing for queries as well as modification operations. Web server security enables custom delegated administration applications to be developed based on this interface.

Low-Cost, High-Value Solutions

Traditional directory integration solutions require complex LDAP provisioning and replication schemes and even synchronization to operate. These new directories then become yet another directory source that has to be maintained and managed.

As a light, real-time service, Oracle Virtual Directory improves efficiency by reusing existing directory infrastructure, rather than synchronizing and duplicating it. Oracle Virtual Directory extends the reach of existing enterprise directories and capitalizes on their value.

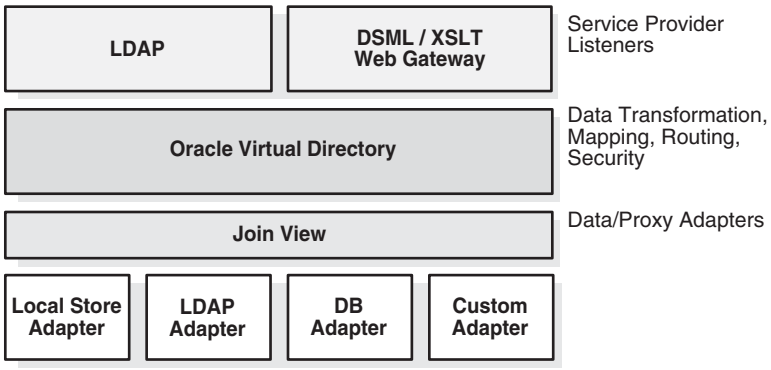
1.1.4 Architecture and Topology

The following sections describe the Oracle Virtual Directory architecture and topology.

Oracle Virtual Directory Architecture

The Oracle Virtual Directory server is written in Java and internally it is organized into multiple layers, as shown in Figure 1–3. These layers are logical layers—Oracle Virtual Directory appears as a single complete service to the administrator and to clients.

Figure 1–3 Oracle Virtual Directory Architecture



The first layer is Oracle Virtual Directory's listener layer where socket-level protocol is spoken. Oracle Virtual Directory provides two types of listeners: LDAP and HTTP. Both listeners support SSL/TLS on top of their basic protocols. The LDAP layer also provides the ability to support LDAP-SASL to support digital certificate authentication.

The listener hands off requests to a worker thread which handles further processing to determine which action to take, such as a search or update. Operations appear the same internally to Oracle Virtual Directory whether it is an LDAP or DSML request. After the operation is determined, the first level of security checks are performed, including making sure the request is not in violation of any Denial of Service policies or inbound Oracle Virtual Directory-level access controls.

If the request satisfies the in-bound security requirements, the next step is to invoke any global level mappings and plug-ins. Mapping and plug-ins have the ability modify the operation such as changing the name or value of attributes. After invoking configured global-plug-ins, Oracle Virtual Directory determines which adapters can handle the request by processing the information provided in the operation.

The DN of the operation, that is, the search base in the search or the DN of the entry in an all other LDAP operations like a bind or add, is the primary information used. Oracle Virtual Directory examines the DN and determines which adapters could potentially support an operation for that DN. This is possible because each adapter

configuration indicates what LDAP namespace it is responsible for. If multiple adapters can support the incoming DN namespace, for example, a search whose base is the root of the directory namespace, such as dc=oracle,dc=com, then Oracle Virtual Directory performs the operation on each of the selected adapters eligible for handling that request. The order of precedence is configurable based on priority, attributes, or supported LDAP search filters.

After Oracle Virtual Directory chooses an adapter, the next step is to invoke any inbound adapter level plug-ins, which are like global plug-ins except operate only on the specific adapter. After any plug-ins are invoked, then the adapter translates the Oracle Virtual Directory request into an operation that maps to its specific adapter level protocol. In the case of the LDAP adapter, there is often very little translation, perhaps only to translate the incoming DN to a value that maps to its actual namespace. For example the incoming search might be for ou=staff,dc=oracle,dc=com and this is mapped to ou=hr,o=oraclecorp. However, in the case of other adapters, like for JDBC using the Database Adapter, the requests are translated into SQL calls, or for custom adapters, the requests are changed into methods that match their proprietary protocols, such as Web Service calls.

After the operation is performed, the result proceeds in reverse order back to the client. In non-search operations, there is normally no further processing. In the case of a search operation where data is returned, plug-ins (optionally) and access controls are processed on the data. The Oracle Virtual Directory access controls are designed to work with any existing access controls you may have in place with your data and act more as additional—not replacement—access controls.

At the conclusion of the operation, the listener level ensures the data is returned to the client in the proper format, such as LDAP or DSML entries.

Oracle Directory Services Manager

As of 11*g* Release 1 (11.1.1), Oracle Virtual Directory and Oracle Internet Directory have a unified graphical user interface (GUI) called Oracle Directory Services Manager. Oracle Directory Services Manager simplifies the administration and configuration of Oracle Virtual Directory and Oracle Internet Directory by allowing you to use web-based forms and templates. Refer to "Getting Started With Oracle Directory Services Manager" on page 8-3 for more information.

Fusion Middleware Control

As of 11*g* Release 1 (11.1.1), you configure many Oracle Virtual Directory features from Oracle Enterprise Manager Fusion Middleware Control. This console enables you to configure and manage all Oracle products from one user interface.

Using the Oracle Enterprise Manager Fusion Middleware Control, you can monitor the Oracle Virtual Directory Server and related components and activities. The Oracle Enterprise Manager Fusion Middleware Control collects host names and ports that you specify during installation or configure at a later time. A Resource Discovery Service (RDS) identifies Server instances and associated components and sends information about these components to the Oracle Enterprise Manager Fusion Middleware Control. The Oracle Enterprise Manager Fusion Middleware Control depends on RDS to detect when nodes in the network are down, or if additional nodes are installed and configured from the Oracle Universal Installer.

Using the monitoring functions, you can gain insight into system activity and performance, for example, total logins, successful and unsuccessful logins, average login time, request latencies, LDAP connections, and so on.

You can monitor the following items:

- **Metrics**: To monitor system health
- **General**: A high-level rollup of load, performance, security, login, CPU utilization, and other data
- **Performance**: Key metrics for the directory server and its host
- **Reports**: Data on operation success and failure
- **Topology**: Information on the Oracle HTTP Server instances, directory server instances, single sign-on servers, associated databases, and so on

1.1.5 Oracle Virtual Directory in Oracle Fusion Middleware

Oracle Fusion Middleware is a collection of standards-based software products that spans a range of tools and services: From Java EE and developer tools, to integration services, business intelligence, and collaboration. Oracle Fusion Middleware offers complete support for development, deployment, and management.

Oracle Virtual Directory is a component of Oracle Fusion Middleware as a standalone Java 2 Standard Edition (J2SE) process. Oracle Virtual Directory utilizes several aspects of the Oracle Fusion Middleware framework, including integrating with the following:

- Common Audit Framework
- Common Logging Framework
- Credential Store Framework
- Oracle Enterprise Manager Fusion Middleware Control

The following is a list of Oracle Fusion Middleware concepts and terms related to Oracle Virtual Directory:

WebLogic Server Domain

A WebLogic Server administration domain is a logically related group of Java components. A WebLogic Server domain includes a special WebLogic Server instance called the Administration Server, which is the central point from which you configure and manage all resources in the domain.

An Oracle WebLogic Server domain is a peer of an Oracle instance. Both contain specific configurations outside of their Oracle homes.

WebLogic Server Home

A WebLogic Server home contains installed files necessary to host a WebLogic Server. The WebLogic Server home directory is a peer of Oracle home directories and resides within the directory structure of the Middleware home.

Oracle Instance

A Oracle instance contains one or more system components, such as Oracle Virtual Directory. The system components in an Oracle instance must reside on the same machine. An Oracle instance directory contains updatable files, such as configuration files, log files, and temporary files.

Oracle Home

An Oracle home contains installed files necessary to host a specific product. For example, the Oracle Virtual Directory home contains a directories that contain Oracle Virtual Directory binary and library files. An Oracle home resides within the directory structure of the Middleware home. Each Oracle home can be associated with multiple Oracle instances or Oracle WebLogic Server domains.

Middleware Home

A Middleware home consists of the Oracle WebLogic Server home, and, optionally, one or more Oracle homes. A Middleware home can reside on a local file system or on a remote shared disk that is accessible through NFS.

> **See:** *Oracle Fusion Middleware Administrator's Guide* for complete information about Oracle Fusion Middleware.

1.1.6 Oracle's Directory Services Portfolio

Oracle is the only vendor that provides a complete range of directory service solutions, including:

- Scalable local-store based directory server with Oracle Internet Directory
- Meta-directory with Directory Integration Platform
- Directory virtualization with Oracle Virtual Directory

Use Oracle Internet Directory when you need to store data in an LDAP server but do not have an existing directory server. Use Directory Integration Platform when you need to synchronize databases or other directory information to Oracle Internet Directory. You can also use Directory Integration Platform to synchronize data between Oracle Internet Directory and certain Oracle applications, like Oracle eBusiness Suite. Use Oracle Virtual Directory to aggregate data from heterogeneous sources into a single directory service in real-time through direct data access.

You can use the Oracle Directory Services products independently of each other or with each other. For example, you can use Oracle Virtual Directory with Oracle Internet Directory to provide a DSML interface to Oracle Internet Directory data. You can use Oracle Internet Directory to provide scalable storage for information that you want to manage via Oracle Virtual Directory and that does not have an existing directory to leverage. Also, Directory Integration Platform with Oracle Internet Directory can use Oracle Virtual Directory to provide additional fault-tolerance support for existing virtualized data-stores. For example, if for some reason your primary enterprise directory becomes unavailable, Oracle Virtual Directory can use the Oracle Internet Directory store.

1.2 Why the Enterprise Directory Is Not Enough

This topic describes many of the obstacles traditional directory servers and enterprise directories face today when deployed for identity management configurations and also explains how Oracle Virtual Directory can solve them.

Overview: Traditional Directory Server Shortcomings

Today's directory servers are designed as specialized databases and by themselves, they do not provide enterprises with the tools needed to connect all possible applications into a single enterprise directory. With very few exceptions, no company has a single enterprise directory.

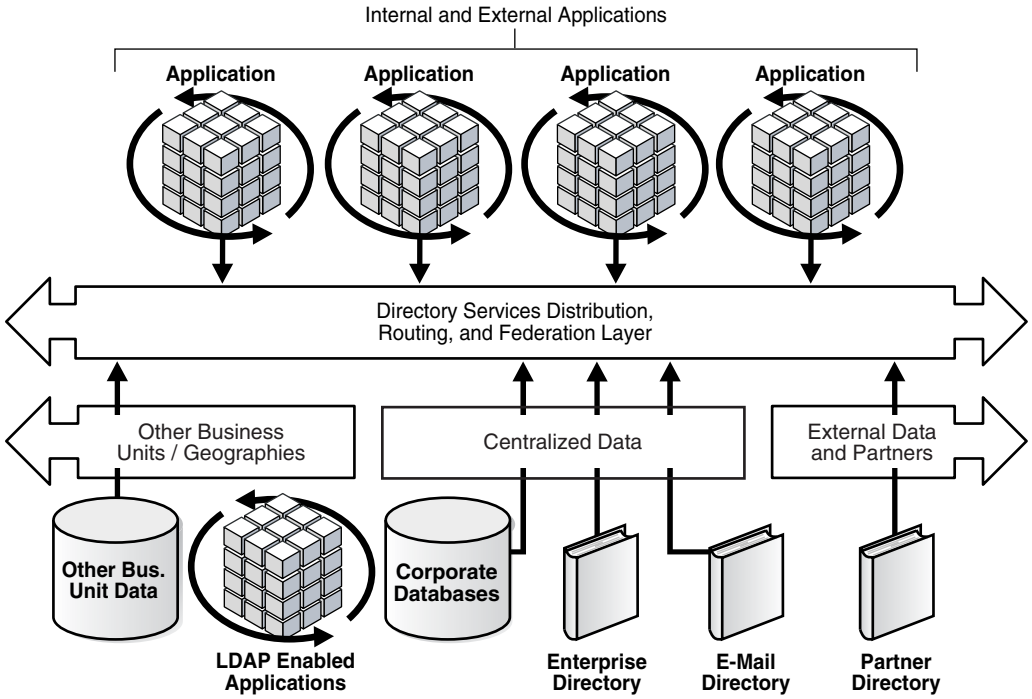
According to analysts, the majority of companies have several (five or more) directories used company-wide. If their intent is to provide data to an application that is used by multiple business partners, then the number of directories increases by at least the number of business partners using the application. Oracle believes that most enterprises will need multiple tiers of directory services both internally and externally. Oracle Virtual Directory is one of the best ways to provide this requirement without duplicating data and without incurring large replicated infrastructure costs.

Typical directory and database technology fails to resolve issues that arise when corporations are made up of independent business units, divisions and partners. Today's directory server technology forces companies to build a single managed data infrastructure that requires huge political discussions on the following topics:

- What data should the directory infrastructure contain?
- Who will manage it?
- Who will fund it?

Issues such as who should pay for directories and who should manage them become critical factors that affect the success of deploying what should be relatively simple database technology. As shown in Figure 1–4, there can be a number of directory sources in different formats and geographies, but also, owned by different parties. Additionally, other directories such as relational databases and email systems can and are added to these traditional enterprise directories.

Figure 1–4 Distributed Directory Services



The issues surrounding distribution of data are further complicated by the addition of LDAP-enabled applications such as Lotus Domino and Microsoft Exchange that have directory information but do not readily integrate into existing enterprise directories due to differing requirements in schema.

Developers have traditionally succeeded at creating databases for specific purposes, because decision-making is driven by individual business managers sponsoring business-driven applications. Now, the new trends of business-to-business web services and inter-business applications means that the data sources within external partners must be considered in the creation of a directory services and security infrastructure strategy.

A directory service integration layer is needed to handle practical issues such as:

- **Distributed Security**: availability and verification

- **Routing**: how to get to different data
- **Integration**: how to handle differing formats
- **Data-level Federation**: merging trusted directories

Oracle Virtual Directory is Oracle's answer to this challenge.

The following sections describe in detail common obstacles traditional directory servers face and how Oracle Virtual Directory can be used to resolve them.

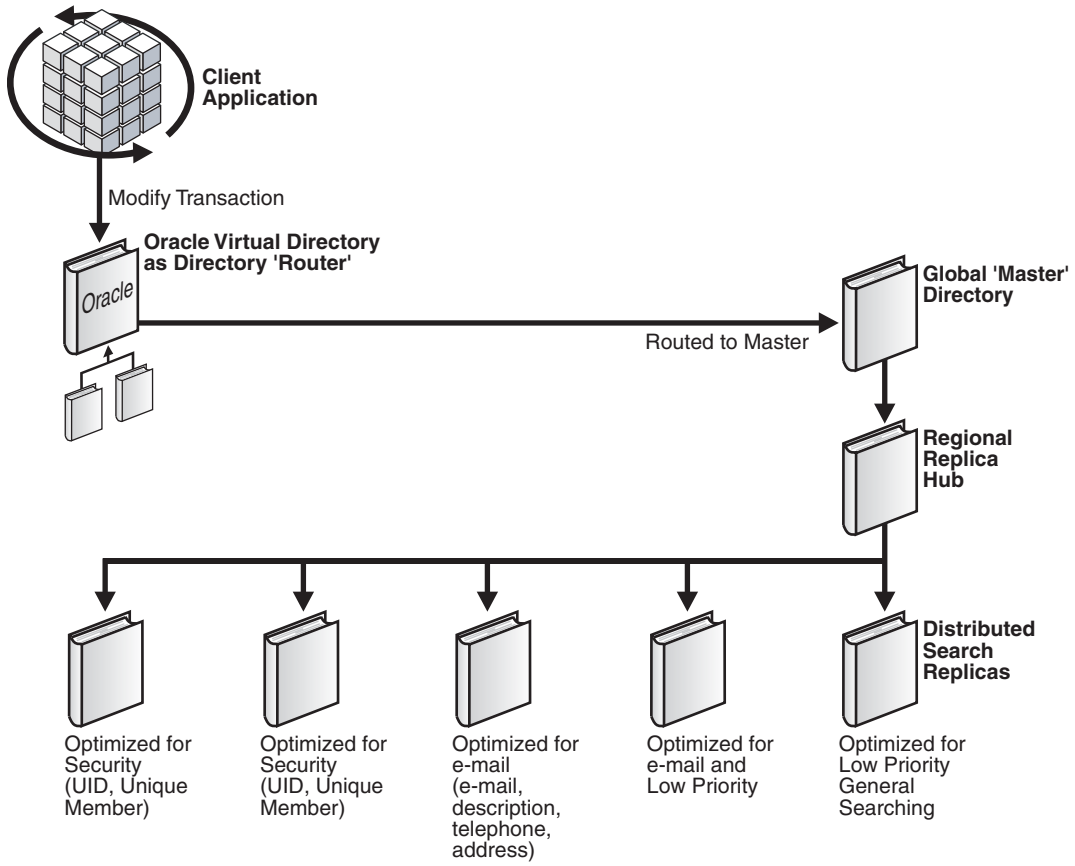
Clogged Replication

In many cases, directory services are deployed over time with a single primary or "master" node and multiple replication nodes. Over time, you may developed multiple hierarchies of directories to facilitate regionalized replication, which in turn support regional directory farms.

As time passed though, replication may slow down to the point where the directory replica servers are outdated. The main cause for slow replication is the maintenance of searching indexes. Often, reviewing the indexes and minimizing indexes can extend the life of an existing infrastructure.

However, at some point, directory indexing demands will outweigh any individual server's ability to keep up with and replicate changes for it. An alternative is to consider breaking the replicas into special purpose or class-of-service nodes. For example, one pair of replicas might be dedicated to handling user search and policy server requests. Another pair might be dedicated to performing white page or email searching requests. Other servers might be tuned to the needs of specific applications.

Figure 1–5 Example of Class of Service Replica Indexing



In Figure 1–5, the indexing strategy has been adjusted to create Class-of-Service replicas enabling replication to scale. Each class-of-service defines a set of directory replicas designed for specific application clients.

In this case, Oracle Virtual Directory automatically knows where the master server is and routes modified traffic directly to it—avoiding a needless directory referral operation. The next challenge is how to route applications to the correct replicas for the correct searches based on class-of-service.

One easy way is to assign directory replicas directly to applications. However, this strategy might not work since applications may use a greater variety of searches than can be configured on any particular directory replica. Instead, the Oracle Virtual Directory virtual directory can be used to automatically route each search request through the use of its routing include and exclude filters. These filters allow the administrator to decide which operations each proxied node may and may not perform.

Figure 1–6 Example of Oracle Virtual Directory Routing an Application Search on UID

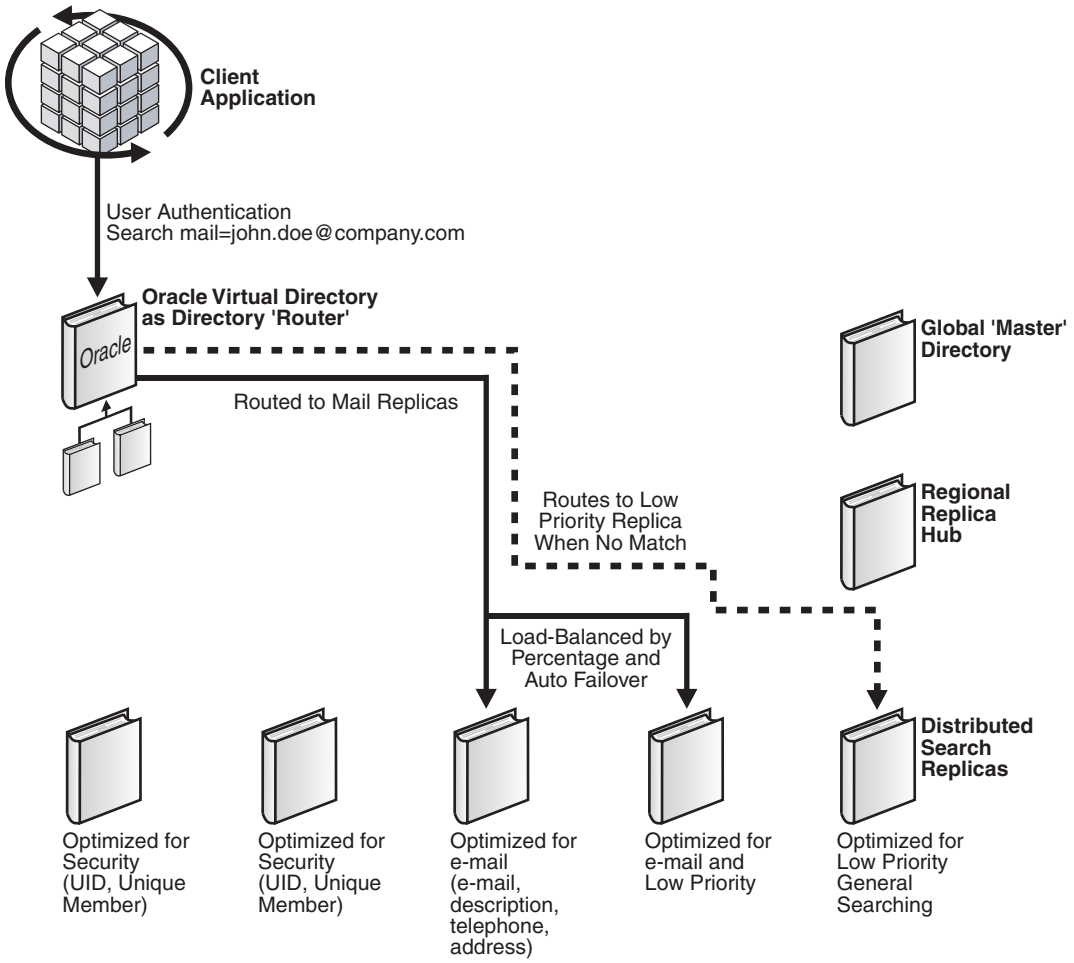


Figure 1–6 shows a typical request where that application is looking to locate a user-distinguished name by searching on UID. Oracle Virtual Directory recognizes the search filter and routes the request to the appropriate directory replicas. Notice that Oracle Virtual Directory can select from multiple nodes and provide load balancing between the nodes, allowing it to spread load across multiple nodes and ensure fault tolerance by not having to rely on any single node.

For different kinds of searching operations, for example, white pages, or classes-of-service, Oracle Virtual Directory can route to alternate directory replicas.

If none of the filters are matched, a default server can be designated. This server might be set up as a low-performance server and lag in replication since it may have more general purpose indexing.

In all of these cases, you see only a single Oracle Virtual Directory. In reality, multiple, identically configured Oracle Virtual Directories can be deployed according to the fault tolerance and loading requirements of the servers. Because Oracle Virtual Directory holds no data, the architecture is 100 percent parallel, allowing for unlimited growth. For example, it is possible to deploy a server pair per application client if needed. Since each server holds no data, and therefore requires no backups, and simply acts as a router, each server adds minimal management costs to the overall infrastructure.

Transaction Failover

Many enterprises have deployed fault-tolerant infrastructures, using devices such as F5 BigIP to route LDAP traffic to available directory server nodes. Because LDAP provides atomic single unit transactions, it has often been assumed that applications would be able to deal with transaction failures. If an LDAP operation fails, it has always been assumed that the application will know to reconnect and try again. For service architects, this has presented many obstacles as some applications replay invalid transactions on multiple directory servers or the application fails and does not realize it just needs to try again.

Oracle Virtual Directory addresses this problem through an intelligent connection pooling mechanism designed to spread application load across multiple directory servers. Since the connection pooling spreads individual transactions across multiple servers, Oracle Virtual Directory realizes when a transaction times out or fails on a particular node and allows the operation to be transferred to another node. Oracle Virtual Directory determines whether this is a data failure, in which case it is returned to the client, or whether it is a service failure. In the event of a service failure, Oracle Virtual Directory attempts to repeat the transaction on each available server until all servers have been exhausted. Only then is a failure returned to the client.

For advanced protection, global failover can be configured by adding non-local directory nodes to Oracle Virtual Directory's server list. When configured this way, a load percentage of 0 is assigned to these non-local nodes in the LDAP Adapter's LDAP Servers configuration. This forces Oracle Virtual Directory to use these nodes only when no other local node is available, providing the ability to route traffic locally while still being able to send it to other sites in times of need.

Connection Domination

In many large-scale directory environments, there may be several applications that dominate and do not share connections. At application bootstrap time, an application is assigned a directory server, for example, by F5 BigIP, and it establishes a permanent connection with that server, causing the following problems:

- Fault-tolerance and load-balancing is effectively bypassed. Since most traditional approaches use connection-based load balancing and failover, a connection-hungry application cannot be easily moved from an overloaded directory server.
- An overwhelming load is created. Since the application tends to use only one directory, its load requirements may exceed the capabilities of the node to which it is assigned. By never relinquishing a connection, there is no opportunity for the management systems to re-adjust the load.

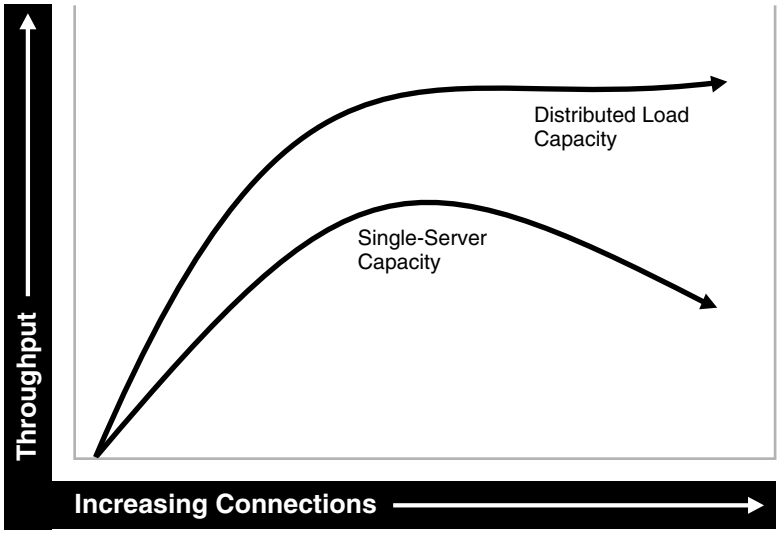
Virtual directory technology helps by providing a connection pooling mechanism that distributes load on a transaction-by-transaction basis. Oracle Virtual Directory maintains a minimum number of connections with each of its proxied directories and adds connections as load requires. Oracle Virtual Directory provides automatic switching of the single client connection and spreads its operations over multiple directory servers, making the directory service more stable because no single directory server node is overloaded. While Oracle Virtual Directory maintains a single connected session between it and the connection dominate client, Oracle Virtual Directory itself is a "good citizen" with the infrastructure and provides load distribution and periodic connection refreshes.

Application Connection Overload

The opposite scenario to the connection domination problem occurs when too many applications make connections and overload the directory server with too many

TCP/IP connections and LDAP bind requests. When this happens many directory servers, including LDAP and X.500 versions, can become unstable as they run out of system resources or simply run out of processing threads. Many benchmarks show that most directory servers have a peak performance level at around seven to ten simultaneous connection threads. This means that while the server may be capable of answering many more calls, peak throughput is diminished as the server starts to spend its time establishing connections rather than servicing requests, as shown in Figure 1–7.

Figure 1–7 Diminishing Throughput Due To Establishing Connections



Oracle Virtual Directory's connection pooling mechanism resolves this issue. As with the connection dominating client, Oracle Virtual Directory uses its pool to share connection between many clients, and uses rebinding to switch between user contexts where necessary (depending on the mode of the Oracle Virtual Directory pass credentials setting). Oracle Virtual Directory takes many client connections and their requests and multiplexes transactions over a reduced number of connections to the proxied servers. Since existing connections are reused, the amount of consumed resources on the proxied server is greatly reduced, allowing it to focus on LDAP transaction processing.

Data Overload

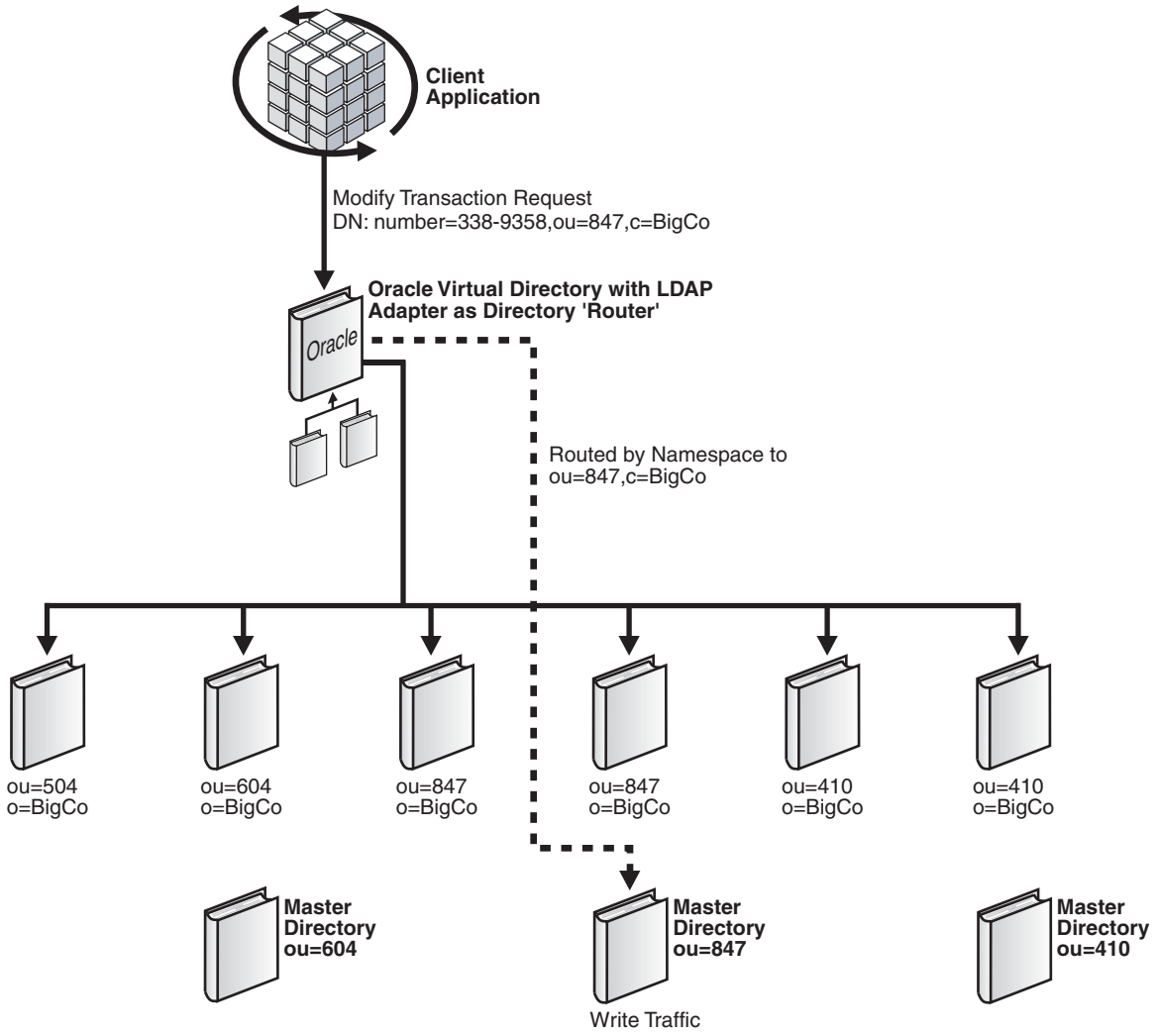
In some cases, optimizing directory replication schemes is not sufficient to create the scale needed for extremely large directories, as described in the Clogged Replication section. In these cases, the ability to create a directory in the tens or hundreds of millions of entries depends on the ability to divide data into smaller pieces and create a virtual view where all the separate pieces appear in a single directory view—a divide and conquer approach.

Oracle Virtual Directory provides several means to accomplish this virtualization. The simplest way is to exploit directory hierarchy. If the data can be broken down into a hierarchical structure, then multiple directory groupings can be created where each grouping owns one or more namespaces. This allows Oracle Virtual Directory to route traffic by simply looking at the distinguished name and deciding which directory grouping should be used.

For example, if a hypothetical telephone company had customers grouped by area code, then Oracle Virtual Directory could route traffic by looking at the organization

unit containing the area code as shown in Figure 1–8. For all modify and bind operations, traffic is routed solely on distinguished name. For searching operations, as described in the Clogged Replication section, routing include and exclude filters would be used to direct traffic based on search filters in the event the search base needs to be o=BigCo and can't be namespace-specific.

Figure 1–8 Example of Routing Traffic Based on Organizational Units

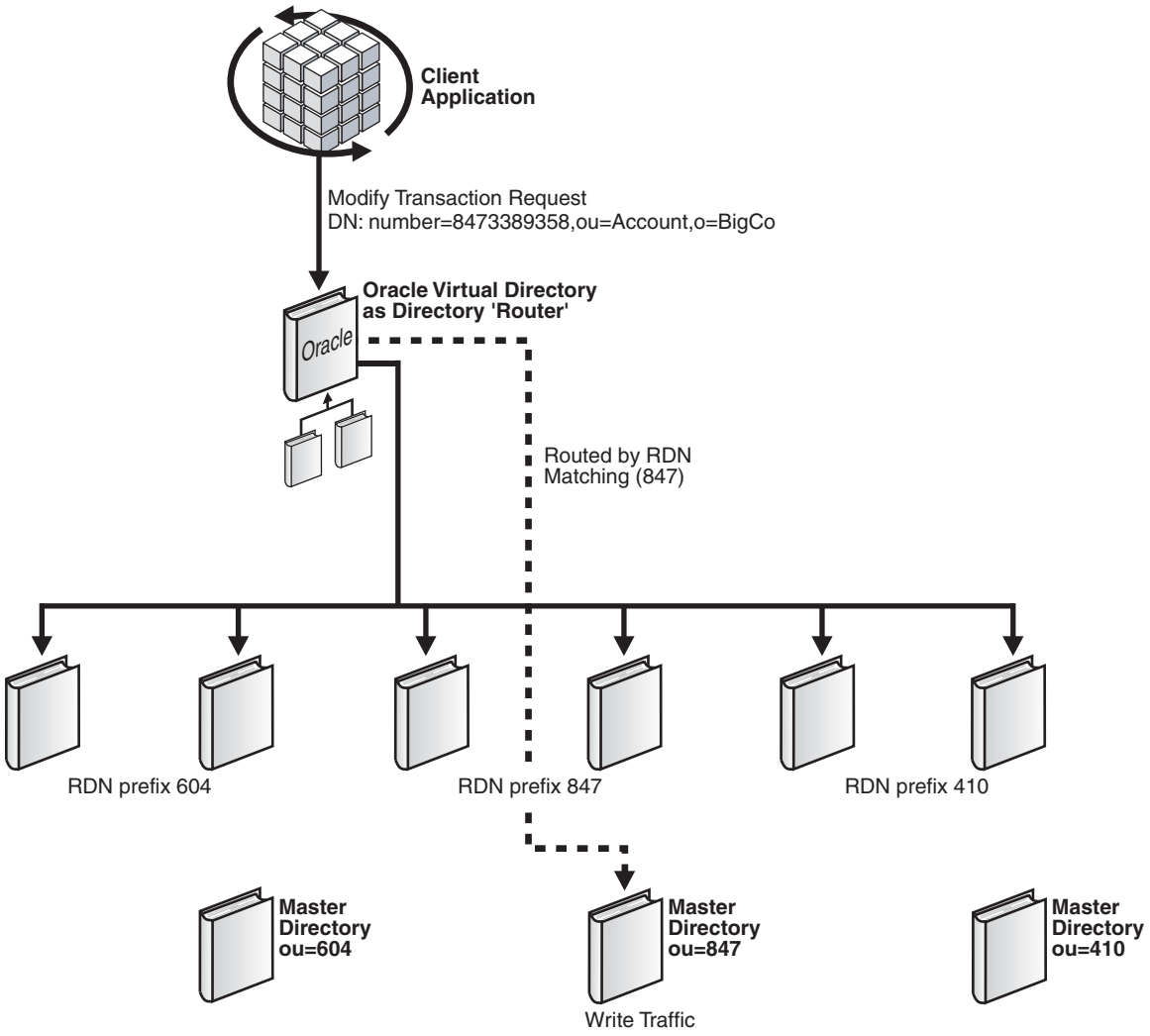


If a flat hierarchy is needed, for example, where all entries appear to be under the same parent, you can choose to either parse the relative distinguished name (RDN), which is the left-most distinguished name or DN component, or to use prefetch operations. If the RDN component can be parsed, then routing could be established through the use of a plug-in that parses the RDN to make routing decisions, as shown in Figure 1–9.

For example, if a DN were of the form: number=6046331751,ou=Account,o=BigCo, then the routing filter could select based on the first 3 digits (in this case 604) to select a particular directory grouping.

If the DN were of the form uid=jdoe,ou=Accounts,o=BigCo, the routing could use a hash table to decide that accounts beginning "a-l" are in one server, "m-r" in another, and "s-z" in a final grouping.

Figure 1–9 Example of Parsing in a Flat Hierarchy



In this case, a routing plug-in mechanism can be used to supplement standard routing features. The intent of the plug-in is to allow you to describe the criteria under which the data will be separated. In order to do this, the criteria selected must ideally be available within every transaction, either in the DN or the filters and base.

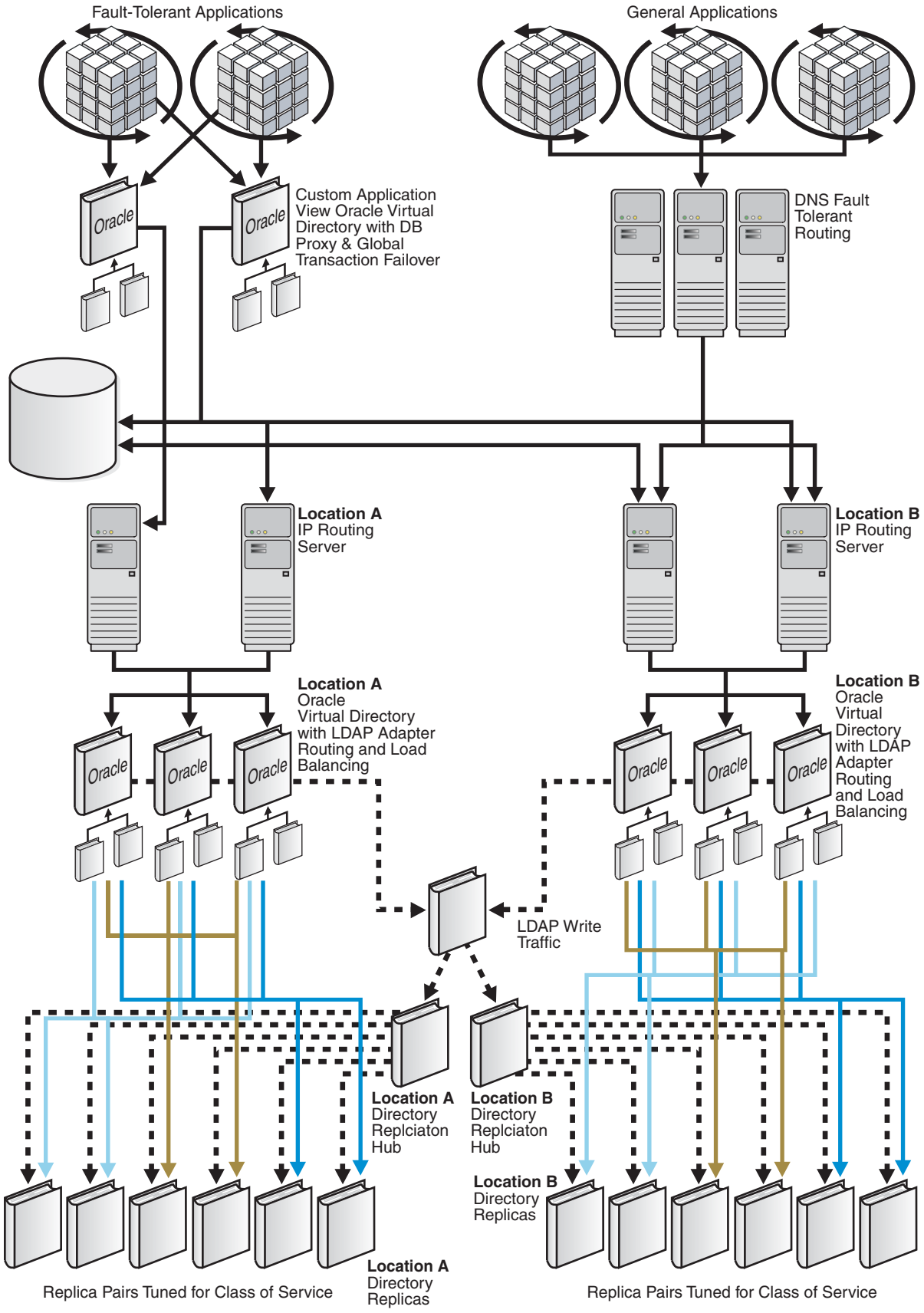
The other alternative is to use prefetch. If the data has been divided such that there is no predictable way, at least to Oracle Virtual Directory, to determine where it occurs, Oracle Virtual Directory must then search directories based on customer-specific criteria to locate the correct repository. For example, on an LDAP modify operation a search must occur first to locate the modify repository. There will be a similar requirement for bind, delete and rename operations. On an LDAP add operation, there must be sufficient information in the add request to determine which repository will receive the add request. In some situations, performance overhead could be moderated through the use of a special master directory that the server uses simply to locate entries in the infrastructure.

Conclusion

Oracle Virtual Directory can be used in fault-tolerant configurations at various points in a global directory services deployment as shown in Figure 1–10. As a load balancer,

Oracle Virtual Directory can be placed between site IP Routers, for example, WebSphere Edge Server and F5 BigIP, and site replica servers. Oracle Virtual Directory provides transaction level load balancing and fault tolerance between servers in the location. In addition to load balancing, Oracle Virtual Directory can offer multiple infrastructure level views of the data, including information from relational database sources via JDBC.

Figure 1–10 Example of Oracle Virtual Directory Load Balancing



For those applications requiring a special directory view or the ability to have global transaction failover, Oracle Virtual Directory can be deployed as a middleware component directly on application servers. This strategy makes the application capable of switching between different locations if a site failure occurs. Normally, in this configuration, Oracle Virtual Directory would provide load balancing only at the local level while switching to other location nodes only when local services have failed.

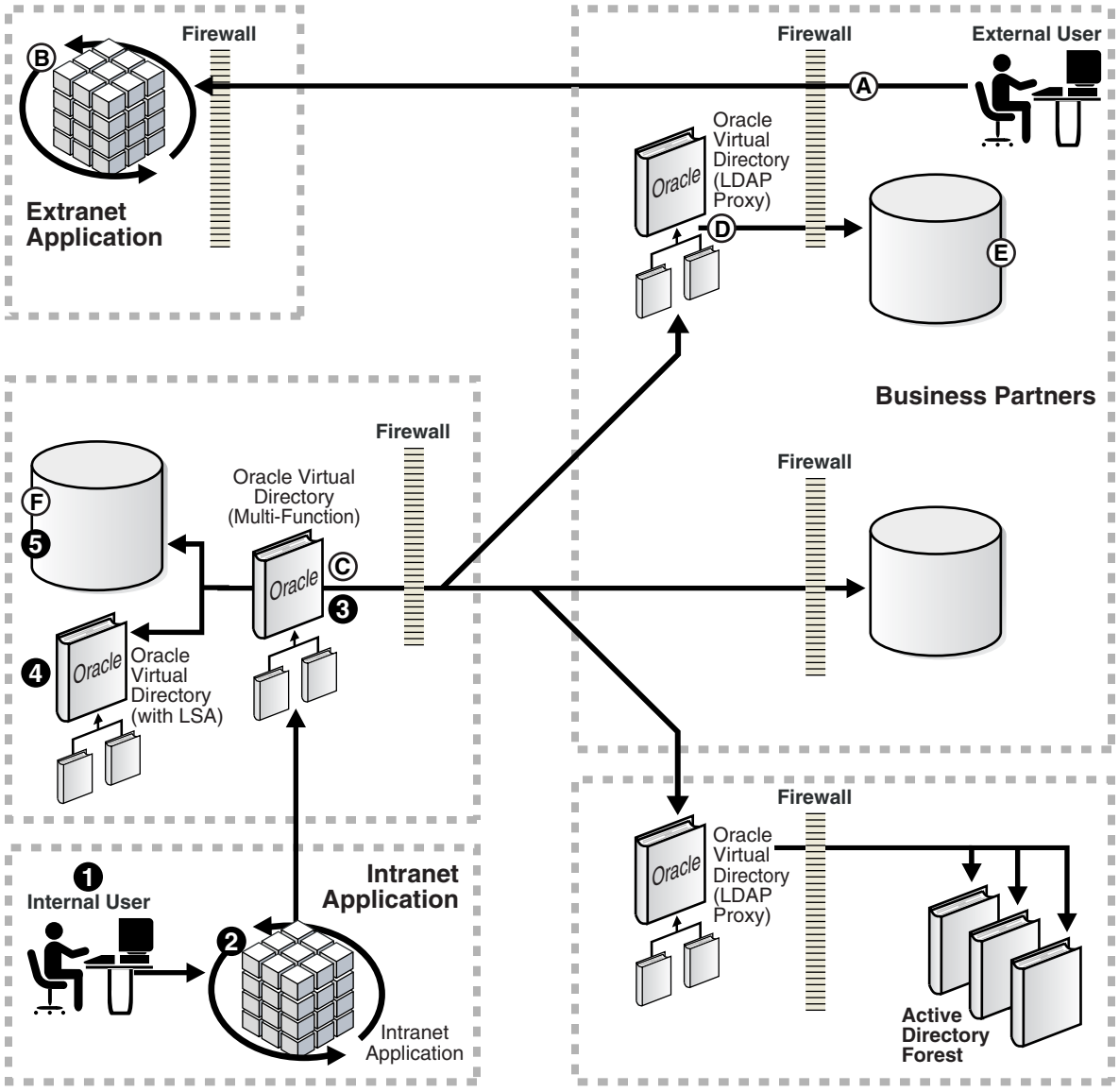
Oracle Virtual Directory's flexibility enables directory architects to develop complex, robust directory service infrastructures. As an integration tool, Oracle Virtual Directory assists developers in using enterprise infrastructure as leverage in the easiest possible way. As an information router, Oracle Virtual Directory is quick to deploy, easy to manage and has an extremely low cost of operation.

Oracle Virtual Directory provides the functionality and performance required to manage large-scale deployments more effectively. As organizations look to solve enterprise-level data issues, Oracle Virtual Directory offers multiple solutions to some of their most challenging concerns.

1.3 Oracle Virtual Directory In Enterprise Directory Network Environments

You can deploy Oracle Virtual Directory in several different environments to resolve obstacles faced by traditional directory solutions. Figure 1–11 shows example deployments for Oracle Virtual Directory in two different environments, Intranet and Extranet.

Figure 1–11 Oracle Virtual Directory In Enterprise Directory Network Environments



Intranet Identity Example

The following steps explain the sequence of Oracle Virtual Directory's role in the intranet example displayed in Figure 1–11:

1. At the lower left-hand corner of the figure, an internal end-user accesses an intranet based web application. The application may or may not include a policy server as part of its own infrastructure.
2. The application or policy service requests the user's identification and password when the end-user accesses the application.
3. The application or policy service accesses Oracle Virtual Directory using LDAPv3 to validate the credentials using an LDAP bind request.
4. Oracle Virtual Directory in turn routes this request to the local directory server store and validates the credentials. On validation, Oracle Virtual Directory returns the verified results to the application.

5. In a further request, the application requests the user's directory entry from Oracle Virtual Directory so that their application profile and rights can be retrieved. Oracle Virtual Directory performs a transparent join, combining attributes from both the local directory server and information from a RDBMS. Once collected, Oracle Virtual Directory merges the result into a single virtual entry and returns it to the intranet application.

Extranet Identity Example

The following steps explain the sequence of Oracle Virtual Directory's role in the extranet example displayed in Figure 1–11:

a. In the upper right-hand corner, an external organization or business partner end-user accesses an extranet-based web application.

b. The application contacts Oracle Virtual Directory using LDAPv3 to verify the user's credentials using an LDAP bind.

c. Oracle Virtual Directory recognizes the credential maps to an external directory. Oracle Virtual Directory connects to the external Oracle Virtual Directory as the business partner using an SSL encrypted link and uses its own credentials to validate the inter-business unit query.

d. Once the business partner's Oracle Virtual Directory has validated the Oracle Virtual Directory, it recognizes the request and passes it on to the internal LDAPv3 directory.

e. Oracle Virtual Directory applies the appropriate inter-business access control and returns the filtered results from the directory back to Oracle Virtual Directory, which is then able to validate the password of the business partner user and return success or failure to the application.

f. Finally, as in the intranet application example, the application might then query Oracle Virtual Directory for additional attributes about the user. Oracle Virtual Directory performs a join linking client-supplied information from the business partner directory with locally stored information in the corporate database.

Example Summary

The examples in Figure 1–11 demonstrate capabilities across a complex scenario. You see Oracle Virtual Directory acting as an information router and joiner, brokering information from multiple secure sources to meet the needs of an application or security infrastructure. Not only can Oracle Virtual Directory bring together information from within a single intranet, it can also leverage information from business partners. This is particularly important because it allows business partners to use the extranet application without having to be provisioned or managed in the host business's directory. Business partner users are authenticated by their own local directory in real time.

Oracle Virtual Directory can also play an important role as a LDAP Proxy server. Oracle Virtual Directory may optionally be used by business partners to act as a directory firewall. Oracle Virtual Directory properly authenticates and authorizes external access to internal directory information. In the bottom right of the diagram we also see how Oracle Virtual Directory's own routing capabilities allow it to route to multiple internal directories or Windows Active Directory forests keeping this information away from the client. As a firewall, Oracle Virtual Directory controls and limits access to information as seen by authorized external parties. As a virtual-directory component, Oracle Virtual Directory simplifies and restructures data for publication of data to be used by business partners.

1.3.1 Virtual Namespace Mapping

Oracle Virtual Directory allows you to connect to any source directory tree and map it to a new virtual tree. For example, an entry in a source directory has the following distinguished name (DN):

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

This source directory entry can be mapped to:

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

In this example, Oracle Virtual Directory maps all entries below o=Division B, c=UK to ou=Division B, ou=People, o=AppView. Oracle Virtual Directory is performing an on-the-fly translation making Division B users appear to be part of the application-specific directory.

Figure 1–12 Example Application-specific Local Directory Branch

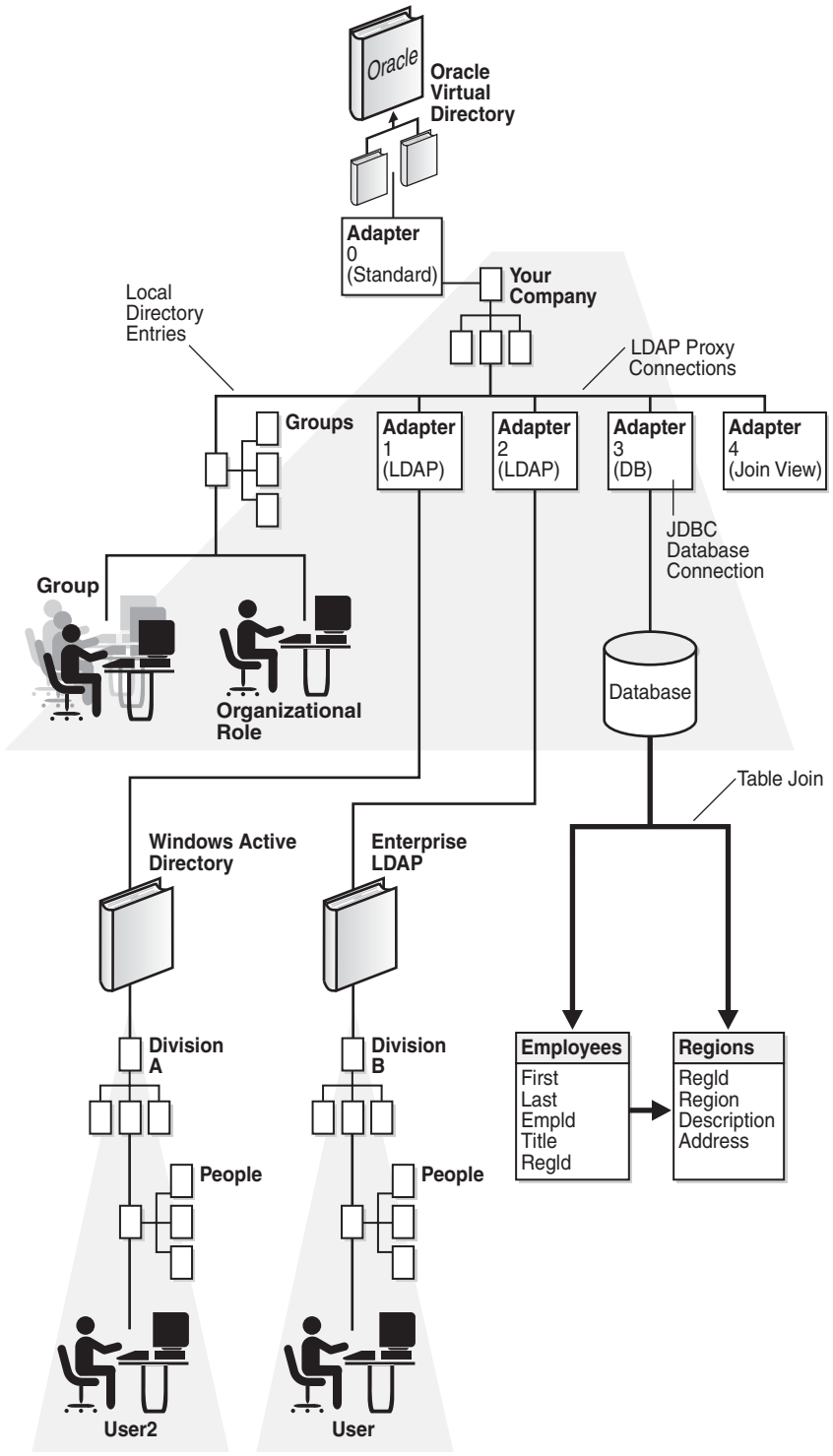
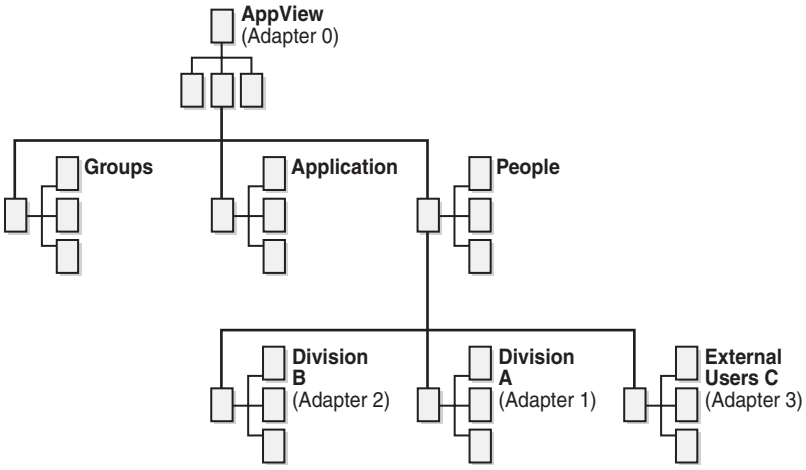


Figure 1–12 shows a local directory branch specific to the application. The root of the tree is o=AppView. Under this branch, local information such as application access control and roles can be stored, for example, cn=User Group,ou=Groups,o=AppView.

The application may have an architectural limitation that it will only search for users under a common people branch. To meet the application requirement, the design

objective can be changed to have the new directory design map all directory sources underneath the ou=People branch. Figure 1–13 shows how this can also be represented:

Figure 1–13 Example Directory Mapping



In Figure 1–13, Oracle Virtual Directory is configured with four adapters:

- **Adapter 0** forms the root of the directory tree and maps to o=AppView. This adapter holds the virtual root of the tree and local entries such as access control groups.
- **Adapters 1-3** map each directory source to positions beneath the ou=People branch of the new application tree.

2

Understanding Oracle Virtual Directory Adapters

This chapter describes Oracle Virtual Directory adapters and contains the following topics:

2.1 What is an Adapter?

To present the single virtual directory view of data in multiple and various data repositories, Oracle Virtual Directory must connect to those repositories so it can virtualize the data and route data to and from the repositories. Oracle Virtual Directory uses adapters to connect to its underlying data repositories. Each adapter manages a namespace in the directory identified by a specific parent distinguished name (DN). There is no limit to how many adapters you can configure. You can also combine and overlap adapters to present a customized directory tree.

The topics in this chapter describe each Oracle Virtual Directory adapter in detail. The following is a list of the adapters Oracle Virtual Directory provides:

- LDAP Adapter
- Database Adapter
- Local Store Adapter
- Join View Adapter
- Custom Adapters

2.2 Understanding the LDAP Adapter

The LDAP Adapter connects Oracle Virtual Directory to LDAP version 2 and 3 directories and Microsoft Active Directories. The LDAP Adapter enables Oracle Virtual Directory to present LDAP version 2 and 3 and Microsoft Active Directory data as a subtree of the virtual directory by providing automatic real-time directory structure and schema translation. You can configure the LDAP Adapter to root the virtual subtree at the top of the directory so that it spans the entire virtual directory information tree (DIT). One LDAP Adapter is required for each distinct LDAP source you want to connect to. For example, if you have an Active Directory domain controller that has four replica servers, you would deploy only one LDAP Adapter and configure it to list each of the four host names and ports of the replicas.

Oracle Virtual Directory can load-balance requests across the replicas or fail-over sequentially. The Oracle Virtual Directory administrator can adjust load-balancing ratios.

The LDAP Adapter provides additional fault tolerance and performance solutions, such as the ability to direct LDAP query traffic to multiple LDAP directory replicas according to particular search criteria or load-balancing requirements according to availability and operation type, such as, query operations compared to modify operations. For example, in a situation with thousands of active clients, Oracle Virtual Directory reduces load to 10 or 20 steady connections processing thousands of queries—or, for a particularly busy client, Oracle Virtual Directory distributes the load across each of its worker threads according to defined load balancing objectives.

In high performance scenarios, the LDAP Adapter has the ability to smooth the connection load seen by source directories by spreading the load from multiple clients across a limited set of connections. The LDAP Adapter does this by maintaining a pool of connections between the Oracle Virtual Directory and the source directory and its replicas, allowing the source directory to work faster by having a reduced connection handling load and by having Oracle Virtual Directory limit traffic to it so that its maximum thread load is not exceeded.

Note: Oracle Virtual Directory allows you to secure the LDAP Adapter interface using SSL.

The remaining sections in this topic provide additional information on the LDAP Adapter, including:

- LDAP Adapter Deployments
- LDAP Adapter Read, Write, Rename, and Compare Support
- Access Control and the LDAP Adapter

2.2.1 LDAP Adapter Deployments

The LDAP Adapter can be deployed as:

- a pure proxy
- a virtual directory proxy

LDAP Adapter as a Pure Proxy

As a pure proxy, the LDAP Adapter can be configured to perform no translation, which makes Oracle Virtual Directory act as a directory router for solving availability and firewall issues. To configure the LDAP Adapter as a pure proxy, you set the

Adapter Root and Adapter Remote Base configuration parameters with exactly the same values.

LDAP Adapter as a Virtual Directory Proxy

By contrast, there can often be a need to change one directory tree namespace to another name to provide a simple namespace mapping. By setting different values for the Adapter Root and Adapter Remote Base configuration parameters, a namespace mapping is implied. For example, imagine that the proxied directory has people entries at the base ou=People,o=Airius.com. The local directory design calls for the Airius entries to appear within the local people folder.

Consider the following settings for the adapter configuration parameters:

- **Adapter Root**: ou=Airius,ou=People,dc=YourCompany,dc=com
- **Remote Base**: ou=People,o=Airius.com

When queried through the LDAP Adapter with these settings, all entries below ou=People,o=Airius.com appear to have DNs under the designated DN root of ou=Airius,ou=People,dc=YourCompany,dc=com.

By default, all attributes are passed through as-is from the proxied directory through the Oracle Virtual Directory to the Oracle Virtual Directory client. The LDAP Adapter can perform basic DN translation of attributes containing DNs specified in the **DN Attributes** configuration parameter list. Only attributes listed in the **DN Attributes** parameter configuration list **and** attributes that have DN values that contain the **Remote Base** DN configuration parameter value for the LDAP adapter as suffix will be translated. For example, consider the following:

The adapter root DN is `dc=emer,dc=orion,dc=com` and the remote base DN is `ou=emer,o=orion.com`.

A group object in the `dc=emer,dc=orion,dc=com` namespace has members in both the `dc=emer,dc=orion,dc=com` and `dc=amer,dc=orion,dc=com` namespace (for example, because of an Active Directory trust relationship) would be translated as:

```
DN: cn=Group 1, cn=groups,ou=emer,o=orion.com
objectclass: top
objectclass: group
member: cn=Jane Doe,cn=users,ou=emer,o=orion.com
member: cn=Ted Davies,cn=users,dc=amer,dc=orion,dc=com
```

> **Note:** The `dc=amer,dc=orion,dc=com` value did not change because the LDAP adapter cannot translate values that are outside of the namespace.
>
> This is an uncommon scenario because usually the DN attribute values either exist within the directory namespace or values are properly translated before being added into the directory.

If you do encounter this scenario, use a custom Mapping script to translate the values that are outside of the adapter's root namespace. Oracle Virtual Directory also has the capability to translate attributes on-the-fly as data from the remote directory passes through the Oracle Virtual Directory through the use of Mappings and Plug-ins.

2.2.2 LDAP Adapter Read, Write, Rename, and Compare Support

The LDAP Adapter supports full read, add, modify, delete, and rename functionality. Support for the LDAP rename operation includes the ability for renaming or moves

between adapters. An LDAPv3 moddn function of LDAP rename, that is, a move, done between adapters causes a cut and paste to occur. To make this transaction safe, the originating entry is not removed until Oracle Virtual Directory has confirmed a successful add into the destination directory. If the add fails in the destination directory the operation is aborted the error from the destination directory is returned.

LDAP compare operations are automatically converted to LDAP get or LDAP bind operations to provide cross-adapter and cross-protocol support.

2.2.3 Access Control and the LDAP Adapter

Oracle Virtual Directory access control is applied at two levels:

- Oracle Virtual Directory Access Control
- Remote LDAP Server Access Control

Oracle Virtual Directory Access Control

Oracle Virtual Directory supports IETF RFC 2820 and also enforces access control according to the IETF *LDAP Access Control Model for LDAPv3* (March 2, 2001 draft). This draft specifies how RFC 2820 should be implemented in a vendor neutral way. Oracle Virtual Directory enforces these standard- and draft-compliant access controls uniformly across all adapters providing a consistent security implementation. Refer to "Understanding Oracle Virtual Directory Access Control" on page 6-4 for more information.

Remote LDAP Server Access Control

Because Oracle Virtual Directory acts as an LDAP client, it must conform to access controls in remote adapter directories, such as LDAP directories. How this works will depends on the vendor's implementation of access control.

When Oracle Virtual Directory connects to a proxied directory server, it can use its own credentials or the bound end-client credentials.

- If Oracle Virtual Directory passes through user credentials to the source directory, then end-to-end user contextual access control is enabled. The implication is that the remote server will authenticate the bound end-client credentials.
- If Oracle Virtual Directory passes its own credentials, then Oracle Virtual Directory must provide its own user-specific access control while being subject to access control enforced against its server credential with the proxied directory.

Depending on the value you set for the LDAP Adapter's **Pass Credentials** configuration parameter, Oracle Virtual Directory can verify credentials in one of the following two methods when processing bind requests using passwords for users whose entry is represented by an LDAP Adapter:

- If the **Pass Credentials** configuration parameter is set to Always or Bind-only, Oracle Virtual Directory passes the supplied user DN and password to the proxied LDAP directory. The proxied directory is responsible for determining the validity of a user-supplied password. The success or failure of the remote password verification is then returned transparently to the user.
- If the **Pass Credentials** configuration parameter is set to Never, Oracle Virtual Directory will use the adapter specified account to connect to the remote directory to retrieve the encrypted password value.

Note: The account specified in the adapter configuration must be granted access to the encrypted password value by the proxied directory or the bind will be treated as a failure. When the encrypted value is returned to the Oracle Virtual Directory, Oracle Virtual Directory will encrypt the user's password and compare it with the encrypted value returned from the proxied server. If a match is determined, Oracle Virtual Directory will return a successful bind.

If users are performing certificate binding, the **Pass Credentials** setting of Never is implied where Oracle Virtual Directory confirms client credentials by pulling the client's public key from the remote LDAP proxy.

2.3 Understanding the Database Adapter

The Database Adapter is a fully featured LDAP-to-Java Database Connectivity (JDBC) gateway supporting translation of all LDAP operations (add, bind, delete, get, modify, rename) into equivalent SQL prepared statement code. The Database Adapter can connect to most databases that support JDBC or Open Database Connectivity (ODBC) through the JDBC-ODBC library. The Database Adapter uses JDBC class libraries to form connections to databases for performing LDAP searches. The required database libraries are generally provided with the database distribution or are available through the database vendor.

Database Adapter Features

The following is a list of some of the Database Adapter features:

- No modifications required for source database
- Flexible table and column mapping, allowing almost any LDAP object to be constructed on-the-fly by using Mappings that define how LDAP objectclasses are derived from tables.
- Connection pool for streamlining requests.
- Multi-table Joins performed within Oracle Virtual Directory or through views in the RDBMS server.
- Multi-value attribute mapping through row grouping using a primary key.
- The ability to construct several object classes and hierarchies within a single adapter connection.

Database Adapter Considerations

When using the Database Adapter, keep the following in mind:

- LDAP operations do not support transactions, so the Database Adapter modifies one database table at a time.
- LDAP supports multivalue attributes, however, database tables support one value for each column. To create an entry with multivalue attributes in the database table, the Database Adapter denormalizes the table and creates one row for each variation of the attributes in the entry.
- The Database Adapter connects to databases using only Oracle Virtual Directory's credentials—it cannot pass through client user credentials as the LDAP Adapter can.

- The Database Adapter supports only normal SQL operations and does not support deployment specific stored procedures. Consider using a plug-in with the Database Adapter to support database stored procedure integration.
- All components that form an entry's DN must be contained in the database table—except for the portion of the DN that is the adapter root namespace.
- Oracle Virtual Directory returns only one attribute when multiple attributes are mapped to one database column. You can address this issue by creating a custom mapping script.
- If you use the Database Adapter to manage client binds, for example, to validate passwords stored in a field in a database table, you must store the password in a `CHAR/VARCHAR` field in the database and you must deploy the Map DB Password mapping to the Database Adapter.
- Currently, the Database Adapter only supports exact matches on telephone number values if used in a search filter. If you are using the Database Adapter to support lookups on telephone number, you can use a plug-in to normalize the input value before sending the query, which allows LDAP applications that do not adhere to the LDAP standard to continue to function.
- The Database Adapter does not support multi-value RDNs.

The remaining sections in this topic provide additional information on the Database Adapter, including:

- Access Control and the Database Adapter
- JDBC Java Class Libraries
- Understanding Database Adapter Mapping

2.3.1 Access Control and the Database Adapter

The Database Adapter respects the access controls that exist on the remote database in addition to Oracle Virtual Directory's own native access controls. Unlike the LDAP Adapter, Oracle Virtual Directory cannot pass the authenticated user's credentials to the remote database. As a JDBC client, Oracle Virtual Directory is restricted in capabilities to those authorized by the RDBMS server for the adapter defined by the adapter's Database Username specified in the JDBC Connection information. All queries are performed using the account specified in the Database Username field.

Oracle Virtual Directory performs password verification by comparing the value returned from the database with the value supplied by the user. If the password is stored in hashed/encrypted form, the value must be prefixed with the appropriate hash qualifier ({crypt}, {sha}, {ssha}). The prefix informs Oracle Virtual Directory about what type of password hash comparison algorithm to use to validate the password. Since this is an LDAP convention, the prefix may not exist in the text file. Use an adapter mapping to assign a prefix.

Oracle Virtual Directory applies its own standards-based LDAP security and access control model simultaneously across all adapters and adapter types.

2.3.2 JDBC Java Class Libraries

Before you can use a particular JDBC driver, you must load the driver file Oracle Virtual Directory as described in "Loading Libraries into the Oracle Virtual Directory Server" on page 9-12. The driver files can be downloaded from their respective manufacturers. For more information on locating drivers, refer to the Oracle Technology Network at: `http://www.oracle.com/technology/index.html`.

After the JDBC libraries are loaded into Oracle Virtual Directory you can define a new Database Adapter connection by either selecting one of the predefined database types, or by specifying the JDBC URL as defined by the manufacturer.

The Database Adapter supports predefined JDBC URLs for the following databases:

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

2.3.3 Understanding Database Adapter Mapping

The following is a list of aspects related to mapping relational data structures into a hierarchical directory when using the Database Adapter, including:

- Entry Name Formation
- Indexes for Mapped Tables
- Multiple Table Writes
- Multiple Valued Attributes
- Searches and Multiple Row Objects
- Writes to Tables
- Cascading Deletes
- Substring Searches

Entry Name Formation

All database fields that will be used as part of an entry's name—with the exception of the portion of the DN that is the adapter's root—must be contained in the table rows that are mapped and returned to Oracle Virtual Directory via the Database Adapter. For example, if you want to create a hierarchy in which user objects contain both a common name (cn) and an organizational unit (ou) in their DN, for example, cn=Joe User,ou=Marketing, both the cn and ou must be part of the entry.

In pure LDAP, the ou attribute would not normally be required as it is part of the parent entry. Since databases are not hierarchical, this enables multi-level DNs to be generated without requiring considerable new metadata to be created and managed to define hierarchy.

Indexes for Mapped Tables

To improve performance when using a Database Adapter, Oracle recommends creating the appropriate indexes on the database tables that are being mapped. For example, a table named `employee` contains columns named `salary` and `employeename` and the cn name is mapped to `employeename`. To improve performance, create a function index for `employeename` as follows:

```
create index upper_employee on employee (upper(employeename));
```

Multiple Table Writes

Multiple table writes are not possible directly through a single Database Adapter. This is the same limitation exhibited by databases where views cannot be updated directly when they present columns from multiple tables.

Oracle Virtual Directory can work around this limitation through the use of the Oracle Virtual Directory Join View Adapter. By creating multiple database adapters (perhaps one for each table) and defining the relationship between them, it is possible to have Oracle Virtual Directory manage writes to entries that are constructed through multiple tables.

Multiple Valued Attributes

Databases typically do not allow for multiple values for a single field within a single table row. There are exceptions where an array type is supported, but these data types tend to be relatively limited. Some users put multiple values into a row by separating data, such as account flags, within a field using delimiters such as commas or pipes (|).

Traditional database design dictates that fields that have multiple values should be normalized into an additional table. Databases that are part of a data warehouse may take a different approach in which every permutation of every field is placed into a denormalized table.

Oracle Virtual Directory can generally support either model. In general, Oracle Virtual Directory will take the designated RDN attributes and use it to group multiple rows together. Where Oracle Virtual Directory schema supports multiple values, Oracle Virtual Directory will group these rows together to form a combined entry.

Consider a table used to define group memberships, as shown in the following example. In the first column there are group names. A member is defined in the second column as follows:

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`

In SQL, to list the members of each group, a SQL query would look like the following:

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

When proxied through the Database Adapter, Oracle Virtual Directory returns the data as follows:

```
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
```

In these results, Oracle Virtual Directory has collapsed the group name into the single unique value and combined the group members into the multi-valued `uniquemember` attribute.

Searches and Multiple Row Objects

Searches are supported to normalized or denormalized tables without doing anything other than configuring a database-level join as necessary. As expected in proper LDAP terms, all values of the attribute will be returned. For example, consider the following search:

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

The Database Adapter returns:

```
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
```

Typical group entry lookups are usually done to establish user authorization. They involve getting the list of groups that a given user is a member of. The full membership information of the group entries in the result set is typically not required. For the purposes of good performance, it is recommended that the LDAP client request the required attributes explicitly and list attributes other than uniquemember whenever the uniquemember attribute is not required. For example, instead of the preceding search, use the following search:

```
ldapsearch -b "ou=groups,dc=yourcompany,dc=com" -s sub "uniquemember=Paul
Jacobs,cn=Users,dc=Oracle,dc=com" dn cn
```

The Database Adapter returns:

```
cn=Administrators,ou=Groups,dc=Yourcompany,dc=com
cn=Administrators
```

Writes to Tables

Writes to multi-table objects (normalized tables) must be performed via a Join View Adapter that splits out attributes to each table based on the design of the database. This is required because while there are general guidelines for database design, every customer's database is different and the resulting relationship between joined tables can vary widely.

Most customers that use existing and important tables also use stored procedures as part of controlling updates to those tables. These are similar to API calls and are proprietary to each database in the way they are constructed and called. The calls themselves are proprietary to the customer. Oracle Virtual Directory supports stored procedures through the use of its plug-in system.

Oracle Virtual Directory supports direct writes to denormalized tables that have each value for the field that will be used in the entry associated with the field used as the RDN for the entry. All operations including add, modify, and delete are supported.

Within the modify operation, the way the LDAP modify-replace works is to remove existing attribute values and add new ones. This translates into a SQL delete and a SQL insert rather than a complex set of SQL inserts, updates, and deletes. The potential difficulty is with databases that have normalized tables in which the insert and delete will trigger other actions within the database. In such a case, a plug-in would need to be constructed that would handle the modify-replace operation.

Most customers do not run into this issue as they either are not using direct SQL access for changes, are using multiple values only for read, or are only using modify-add and modify-remove directly. For example, customers solving issues with big groups are storing groups in databases via Oracle Virtual Directory. Most group membership changes are adds and removes rather than replaces.

Cascading Deletes

Of the issues mentioned in the "Writes to Tables" description the biggest issue to watch for when using the Database Adapter is when Oracle Virtual Directory will be handling database writes directly to a database that is using cascading deletes. A cascading delete is where a delete in one table causes the database to activate a stored procedure that causes deletes in other tables. With cascading deletes, it is possible that a modify-replace would trigger deletes outside of the table being directly touched by the Database Adapter. This happens because the database is sent a single transaction that performs a SQL DELETE that removes existing values and a SQL INSERT that adds new replacement values.

This is not an issue if the database trigger is based on a single-valued table in which fields with multiple values have been normalized to other tables. Oracle Virtual Directory will do an UPDATE rather than a DELETE-INSERT combination on replaces in which the table has a single row associated with the entry being changed.

Note: The Database Adapter sets unmapped database column values to NULL during an LDAP update. This limitation is planned to be removed in a future version of Oracle Virtual Directory.

To work-around this limitation, either map all columns (and use routing rules to hide them in search results) or map the table to a view with an update trigger.

Substring Searches

If applications use substring searches, the Database Adapter-mapped attribute must be mapped as VARCHAR database syntax. This is the only way for SQL to support substring searching.

2.4 Understanding the Local Store Adapter

The Local Store Adapter allows you to store small amounts of data that does not need to be made highly available in a local flat file. The following is a list of the most common uses for the Local Store Adapter:

- In a situation where you need to store a small amount of data that must be visible to client applications but does not change very often and also does not need to be made highly-available. For example, the base entries used by Oracle Virtual

Directory to emulate Oracle Internet Directory as part of the Oracle Virtual Directory-Enterprise User Security integration.

- Developers can use the Local Store Adapter when they are building an application that will eventually use the enterprise directory, but they currently do not have access to an enterprise directory service. In this situation, developers use the Local Store Adapter to store test entries as part of the development process, and later migrate to an enterprise directory storage system, such as Oracle Internet Directory, when the application is ready for production use.

  Note: Oracle Virtual Directory's value is as a virtualization and proxy service, not as a directory store. If you need a highly available directory storage system, Oracle recommends using Oracle Internet Directory—not Oracle Virtual Directory's Local Store Adapter.

- To hold a single top of the directory tree entry when you have multiple adapters and they are all branches of the directory tree. For example:

  Say you have the following two LDAP adapters configured:

  – ou=staff,dc=mycompany,dc=com

  – ou=customers,dc=mycompany,dc=com

  You can use the Local Store Adapter to store a top entry of dc=mycompany,dc=com above the other adapters. Oracle Virtual Directory does not require this top entry and it will be transparent to most LDAP clients, however some LDAP client applications, in particular Graphical User Interface desktop LDAP management clients, can return erroneous error messages if they do not find an entry at the top of the server.

You can deploy multiple Local Store Adapters for each Oracle Virtual Directory server, however most servers deploy only one. The most common use for deploying multiple Local Store Adapters is to store different parts of the directory tree in different files for backup and recovery considerations. For example, one branch of the directory tree might be relatively static and therefore does not require frequent backups.

2.4.1 Migrating Local Store Adapter Data

You can use the Oracle Internet Directory `oidcmprec` tool to migrate LDAP data out of a Local Store Adapter and into another repository.

Note: Only the `compare` and `reconcile` operations of the `oidcmprec` tool are supported for Local Store Adapter data migration.

This section provides an *overview* of migrating data out of a Local Store Adapter using `oidcmprec`.

See Also: Refer to the following documents for complete information on the Oracle Internet Directory `oidcmprec` tool:

- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- *Oracle Fusion Middleware User Reference for Oracle Identity Management*

This section contains the examples for migrating data out of a Local Store Adapter using `oidcmprec` in the following scenarios:

- Initial One-Way Data Migration Between Two Local Store Adapters
- Synchronization Between Two Local Store Adapters
- Local Store Adapter to Oracle Internet Directory Data Migration

Notes:

- After executing the `oidcmprec` command, you will be prompted for the replication dn password for both the source and destination directories.
- Oracle Virtual Directory does not allow you to modify the orclguid. Oracle recommends excluding orclguid when comparing two directories using `oidcmprec`.

Initial One-Way Data Migration Between Two Local Store Adapters

For example, consider the following use case. You have two Oracle Virtual Directory servers running on two different hosts, myhost1 and myhost2. Each host has a Local Store Adapter (LSA) configured: LSA1 on myhost1 and LSA2 on myhost2. You want to migrate the data under c=US in LSA1 to LSA2 for the first time. To perform this migration, run the following command:

```
oidcmprec operation=reconcile source=myhost1:port
destination=myhost2.port base="'c=US'" scope=subtree exclattr=orclguid
```

Note: Oracle recommends using the `reconcile` operation of `oidcmprec` for one-way data migration.

Synchronization Between Two Local Store Adapters

For example, consider the following use case. You performed an initial data migration from one Local Store Adapter (for example, *LSA1*) on one Oracle Virtual Directory (for example, *myhost1*) to another Local Store Adapter (for example, *LSA2*) on a different Oracle Virtual Directory (for example, *myhost2*) as described in Initial One-Way Data Migration Between Two Local Store Adapters. Later, you want to synchronize the data under c=US between the two Local Store Adapters. You do not need to migrate all data under c=US from the source. Instead, migrate only recently modified data using the `oidcmprec` tool's timestamp based filters, which are based on RFC 2254. All data in the source that was modified after the value specified by the timestamp filter will be migrated to the destination.

Notes:

- When synchronizing data between two Local Store Adapters, only one-way data migration is supported. That is, if you initially synchronize data in *LSA1* (source) to *LSA2* (destination), you should not synchronize data from *LSA2* to *LSA1* in the future.
- The target (destination) Oracle Virtual Directory must be running to migrate the data.

To perform this type of migration, run the following command:

```
oidcmprec operation=reconcile source=myhost1:port
destination=myhost2.port base="'c=US'" scope=subtree
filter='("modifytimestamp>=point_in_time")' exclattr=orclguid
```

Local Store Adapter to Oracle Internet Directory Data Migration

You can also use the `oidcmprec` to migrate data from a Local Store Adapter to Oracle Internet Directory. The following command migrates data under c=US in the Local Store Adapter on Oracle Virtual Directory running on *myhost1* to the Oracle Internet Directory running on *myhost2*:

```
oidcmprec operation=reconcile source=myhost1:OVD_port
destination=myhost2.OID_port base="'c=US'" scope=subtree exclattr=orclguid
```

2.5 Understanding the Join View Adapter

The Join View Adapter combines multiple different data sources into one unified LDAP view similar to a relational database's table join. The Join View Adapter does not connect to the underlying data repository like the other Oracle Virtual Directory adapters—it builds on top of one or more existing adapters to assemble its data. Think of the Join View Adapter as joining two or more data repositories by defining join relationships, sometimes called "Joiners," between adapters.

A Join View is dynamic and performs no synchronization between proxied sources. The Join View objective is to present merged data to an LDAP client in real time. A Join Adapter solves the problem of split-profiles, meaning cases where data for a single entry is split between two or more sources. For example, a Join Adapter is used to link a person's LDAP entry with their job title from the Human Resources database. A Join Adapter is *not* used to solve the problem of allowing applications to search multiple and different sources, such as one LDAP for staff and one LDAP for customers. This particular problem can be solved by making the sources appear as virtual branches under a common root.

The following is a list of some of the Join View Adapter's capabilities:

- Attributes from multiple adapters and object classes can be merged into a new virtual entry.
- Each adapter can have specific attributes selected for display during a search.
- Each adapter can subscribe to changes for any attribute during a modify operation.
- Support for nested Join View Adapters where a primary or joined adapter can be another Join View Adapter.

- Support for recursive Join View Adapters that perform a join with itself and the recursive structure ends when no join entry is found.
- For each joined adapter entry returned, the Join View Adapter adds an attribute value, vdejoindn, that indicates which entries in the joined adapters were used to form the consolidated entry.

The following are important aspects about the Join View Adapter, including:

- Join View Adapter's Primary Adapter
- Join Rules
- Join View Adapter Routing
- Searching Join View Adapters
- Duplicate Attributes in Join View Adapters

Join View Adapter's Primary Adapter

Each Join View Adapter requires one—and only one—primary adapter to be identified. The primary adapter is used for creating and searching the directory tree entries. The Join View Adapter operates by taking each entry found in the primary adapter and joining it with entries in other adapters according to the defined join relationship. An entry must exist in the Join View Adapter's primary adapter for it to also exist in the Join View Adapter. By default, the primary adapter is used for processing authentication credentials. Any joined adapter may also be used for binding.

Join Rules

Each Join View Adapter may have zero or more join rules that specify a join relationship between an entry in a primary adapter and an entry in a joined adapter. A join rule consists of a join relationship, the adapter it is joining, and some simple join configuration information, such as userprincipalname=uid search criteria. Join rules are processed in the order defined and run in cumulative fashion. As each join is processed, the resulting joined entry is used for the next join relationship.

Refer to Join Relationships for more information.

Join View Adapter Routing

The Join View Adapter relies on several routing attributes, specifically, Routing Retrievable and Routing Storable, to determine which attributes are retrievable and modifiable for each adapter. You can use the Visibility routing attribute to hide primary and joined adapters that are used to form the Join View Adapter.

Searching Join View Adapters

By default, Join View Adapters only search the primary adapter. However, you can use the ForkJoin plug-in to enable searching across attributes in "Joined" sources.

See: "ForkJoin Plug-In" on page 4-5 for more information.

You have two adapters: A and B. Adapter A is the Primary Adapter and contains uid, sn, and cn. Adapter B contains uid, mail, and employeeNumber. You create a Join View Adapter C with adapter A as the Primary Adapter and adapter B as the secondary adapter, and search on uid. Oracle Virtual Directory searches adapter A and finds the entry, then searches adapter B to find any matching join entries. The search returns an entry that contains uid, sn, cn, mail, and employeeNumber.

If client requests all attributes to be returned, that is, there is no list of attributes at the end of search operation, Oracle Virtual Directory requests all attributes from the joined

adapters. However, if the client specifies a list of attributes to be returned, those attributes are extended internally with the attributes from the join condition and Oracle Virtual Directory requests only that list of attributes from the joined adapters. For performance reasons, only the attributes specified by the client—not the attributes in the join condition—are returned. Limiting the number of attributes to be returned from an LDAP server, regardless of Oracle Virtual Directory or using a Join View adapter, helps improve performance.

Duplicate Attributes in Join View Adapters

The Join View Adapter will show only one single value if there are multiple attributes with the same name from multiple data sources (such as two uids from two different LDAP Adapters) and you configure the Join View Adapter to retrieve attribute values only from a specific adapter. To show only the attribute from a specific adapter, ensure the attribute is not listed in the Retrievable Attributes field in the Routing settings for the adapter you do not want to show the attribute for. Optionally, you can hide specific attributes using a Mapping or Plug-in.

2.5.1 Typical Join View Adapter Deployments

The Join View Adapter presents dynamic, real-time joined views of data—no synchronization is required, allowing it to be especially useful to address the following typical identity management tasks:

- **Password consolidation with Microsoft Active Directory**

  If you are using Microsoft Active Directory as your central user identity repository, you can avoid the cumbersome password consolidation process by using the Join View Adapter. The Join View Adapter eliminates the need to export password information from Active Directory to other application directories by joining the two. Using the Join View Adapter, clients will have a view of the application directory data, but bind using the Active Directory data.

- **Data Integration**

  Using the Join View Adapter allows Oracle Virtual Directory to create a unified view of a user by joining multiple data sources, including databases and directories.

- **Application Integration**

  Some applications, especially custom applications, have unique data requirements and it is often difficult to alter the enterprise's directory schema to accommodate those unique requirements. Using the Join View Adapter, you can use data in the enterprise directory and also manage the unique custom application data in a different directory or the Local Store Adapter.

2.5.2 Join Relationships

Each Join View Adapter requires a separate primary adapter. You combine the Join View Adapter and the primary adapter by creating join relationships, sometimes called "Joiners." The join relationship is a set of routines that define how joins are formed and how they impact all directory operations on behalf of the client. Oracle Virtual Directory provides an extensible Joiner class that you can use to provide custom logic. The Joiner class also includes an API that allows you to provide pre-processing and DN mapping logic for each LDAP operation (for example, preAdd, preModify, preGet, preDelete, preRename).

You can also use a Join View Adapter without a join relationship to copy an existing adapter into a new part of the directory tree. This has the advantage of sharing the original adapter's connection pool without creating duplicate adapters.

The following is a description of the supported join relationships types, including:

- Simple Joiner
- OneToMany Joiner
- Shadow Joiner
- Custom Join

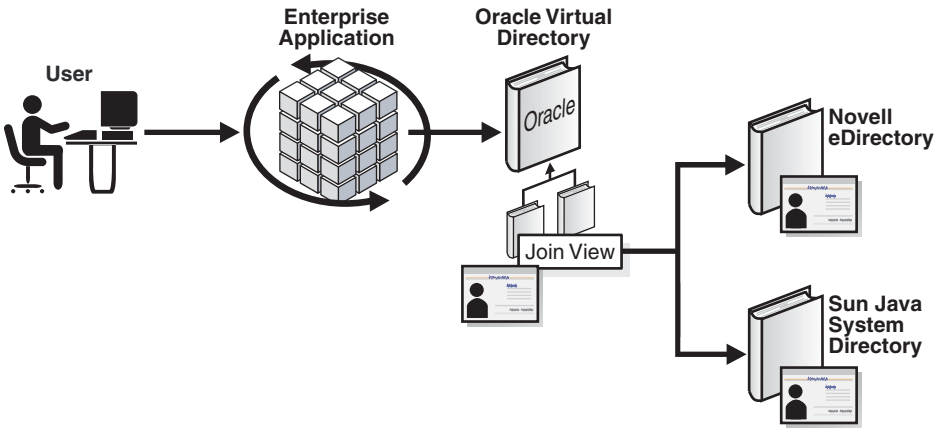
2.5.2.1 Simple Joiner

The Simple join relationship defines a one-to-one relationship between the entries in the primary adapter and the Join View Adapter by comparing their attributes using a simple join relationship in the form `remoteattribute=primaryadapterattribute` where `remoteattribute` is an attribute in the target joined adapter and `primaryadapterattrinute` is an attribute from the primary adapter. A more complex criteria is possible by using a comma separated list of conditions which creates a multi-condition join where all conditions are *anded* together. For example, `uid=uid,sn=sn,mail=mail`.

If more than one matching attributes exists, Oracle Virtual Directory uses the first matching attribute. Oracle Virtual Directory applies Add, Delete, and Rename LDAP operations only to the primary adapter and ignores these operations in the Join View Adapter.

Figure 2–1 shows a a high-level example of a Simple Join used for authentication:

Figure 2–1 Example Simple Join for Authentication



2.5.2.2 Conditional Simple Joiner

As described in the "Simple Joiner" section, the Simple Joiner joins entries from two adapters based on a shared attribute value. The Conditional Simple join relationship extends the Simple Joiner functionality so that the join only occurs because of an additional condition. The Conditional Simple join relationship uses a `;` character in the join rule to extend a Simple Join relation and add an additional condition, such as `"employeenumber>0"` for which the join may only occur on.

For example, a Simple Joiner condition could be:
`employeenumber=employeenumber`

You can extend this Simple Joiner rule and add an additional condition using the ConditionalSimpleJoiner and the ; character, for example:

```
employeenumber=employeenumber;(&(employeenumber=101)(sn=Smith))
```

2.5.2.3 OneToMany Joiner

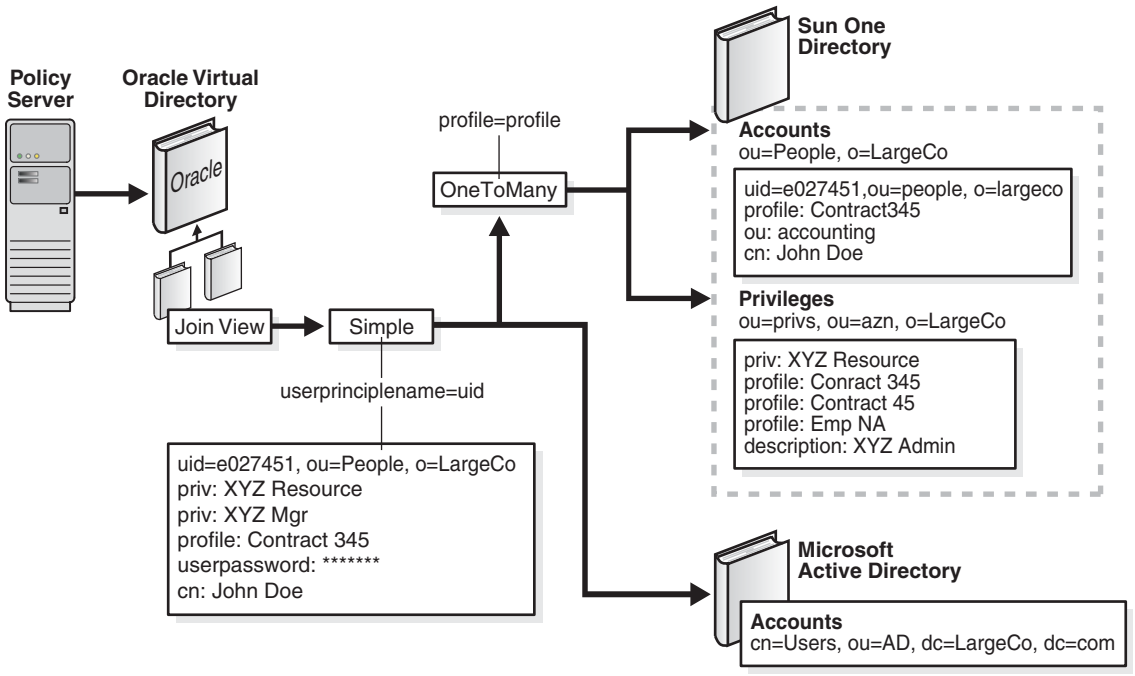
The OneToMany join relationship defines a one-to-many relationship between multiple primary adapters and the Join View Adapter. Similar to the Simple join relationship, the OneToMany join relationship locates attributes in the Join View Adapter by comparing attributes, however, all matching entries in the Join View Adapter are included in the relationship between adapters. The OneToMany Join is useful if you want to consolidate multiple role objects or identities into one virtual entry. Also similar to the Simple join relationship, Oracle Virtual Directory applies Add, Delete, and Rename LDAP operations only to the primary adapter and ignores these operations in the Join View Adapter when using the OneToMany join relationship.

Figure 2–2 shows an example where a policy server must make policy decisions about an individual. For integration purposes, the policy server prefers to see a single entry with the rights of the user exposed as a privilege attribute, which allows the policy server to test rights assertions with queries such as:

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

The OneToMany Joiner is used to match one or more privileges to a user on the basis of a profile attribute in their main `ou=People` entry. The OneToMany Joiner looks for all privileges with the same profile value as in the entry and merges them with the entry. A second stage join uses the Simple Joiner so that the SunOne Directory combined profile is used in conjunction with the user's Active Directory credentials.

Figure 2–2 Example OneToMany Join



2.5.2.4 Shadow Joiner

Use the Shadow Joiner when you need to store entries in a source such as an LDAP or Database Adapter that requires a schema extension—but the schema extension is not possible either for business or technical reasons. The Shadow Joiner allows you to the extended attributes in another store, such as a Local Store Adapter or in another LDAP directory that supports the extensibleobject objectclass like Oracle Internet Directory or Sun Java System Directory Server (Microsoft Active Directory does not currently support the extensibleobject objectclass).

> **Note:** Oracle recommends using the Local Store Adapter as a store for a Shadow Join attributes only for testing and demonstration purposes. For production environments, use Oracle Internet Directory as the Shadow Join data store.

The Shadow join relationship maintains the same structure of the entry in the primary adapter, but stores additional attributes by creating shadow entries using a separate adapter. Using the Shadow Join relationship, applications can use the enterprise directory and also store application-specific attributes in the Join View Adapter's data store. The application believes it is communicating to a directory that stores all attributes, but Oracle Virtual Directory is actually quietly storing application-specific data in an alternate *shadow* directory. The shadow attributes are only visible after attributes with values are added to the joined entry DN.

The Shadow Joiner requires that the directory that stores the data must support the ExtensibleEntry objectclass and the vdeshadowobject objectclass must be defined. The extensibleobject objectclass defines an object that may contain any attribute and is not supported by all LDAPv3 server products (it is an optional item in the IETF RFC 2251). The Shadow Joiner uses the extensibleobject and the vdeshadowobject objectclasses to store the local entries because local entries may only contain one or more attributes and may not form valid objectclasses. Typically, the Local Store Adapter is used to store local entries, however you can use any LDAP directory that supports the extensibleobject objectclass and where the vdeshadowobject objectclass has been defined.

> **Note:** Oracle Internet Directory supports the ExtensibleEntry objectclass.

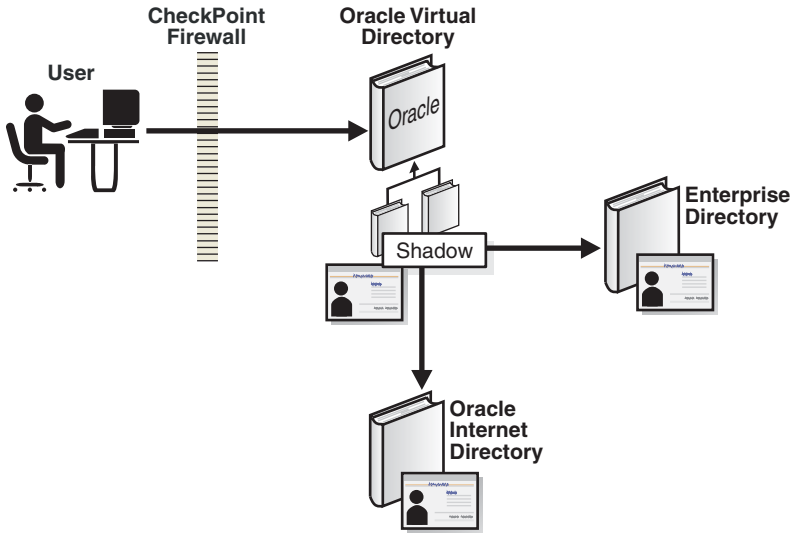
The Shadow Joiner works by encoding all primary adapter DN's into a hash that can be used to locate the joined entry in the joined adapter without needing to perform a search. When the Shadow Joiner fails to locate a corresponding record in the Join View adapter, it automatically creates a new one, storing designated attributes in the joined adapter. As much as possible, the Shadow Joiner operates transparently to the application, taking care of creating and renaming entries in sync with that of the primary adapter.

The Shadow Joiner supports all LDAP operations. When an LDAP modify operation occurs, the Shadow Joiner examines the parameters identified by the adapter's **Storable Attributes** routing parameters to see if any attributes are to be stored in the Shadow Join adapter. If any such attributes exist, the Shadow Joiner attempts to locate the local entry by taking the MD5 hash of the primary entry and locating the local entry. After it is located the appropriate LDAP modify operation is performed locally. If a local entry is not found, the Shadow Joiner attempts a secondary search, in case the

primary DN changed, to locate the entry using a primary key. If no local entry is found, a new entry is automatically created.

Figure 2–3 shows a CheckPoint firewall configured to connect to an Oracle Virtual Directory. The Oracle Virtual Directory uses Oracle Internet Directory to maintain the CheckPoint firewall schema, allowing integration of the CheckPoint firewall into the corporate enterprise directory without requiring that the corporate enterprise directory schema be extended with application-specific data. Instead, by storing it in Oracle Internet Directory, the application-specific data can be managed by the team responsible for the CheckPoint firewall management.

Figure 2–3 Example Shadow Join Use for Storing Application-Specific Data Locally



2.5.2.5 Custom Join

The Join View Adapter also supports Java-based plug-in join relationships allowing you to implement different and complex relationships. For example, where joins are performing multi-step operations, a custom join can provide specific recovery mechanisms suitable for the scenario at hand. Refer to Chapter 4, "Understanding Oracle Virtual Directory Plug-Ins" for more information on Java-based plug-ins.

2.6 Understanding the Custom Adapter

Oracle Virtual Directory supports the ability to create custom adapters using plug-ins that can connect to almost any data source with a defined API. Refer to "Creating and Configuring Custom Adapters" on page 18-2 for information on deploying custom adapters.

2.7 Understanding How Adapters Create the Virtual Directory

This topic provides the following examples to demonstrate how adapters are used to create the virtual directory:

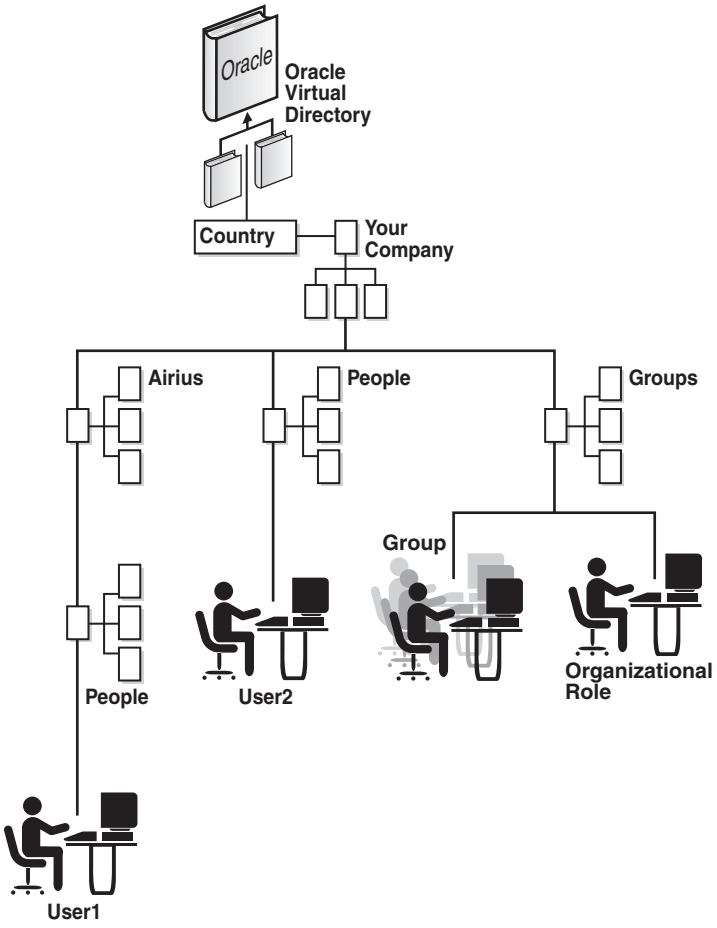
- Example of a Basic Virtual Directory
- Example of a Virtual Directory Using the Join View Adapter

2.7.1 Example of a Basic Virtual Directory

Figure 2–4 shows a virtual directory structure for a directory that has a base of `o=YourCompany,c=US` and the following main branches:

- `ou=Airius` which points to a partner LDAP directory
- `ou=People` which points to an internal RDBMS
- `ou=Groups` with group and role information to be stored in the local Oracle Virtual Directory directory store

Figure 2–4 Example Virtual Directory Structure



The virtual directory in Figure 2–4 requires the following adapters:

- an LDAP Adapter
- a Database Adapter
- a Local Store Adapter

The following list describes how the adapters are configured in Figure 2–5 to create the virtual directory:

- **Adapter 0: Local Store Adapter**

  This adapter forms the base of the directory and holds entries that are not proxied. In this case, the directory entries under Groups are stored in the local directory.

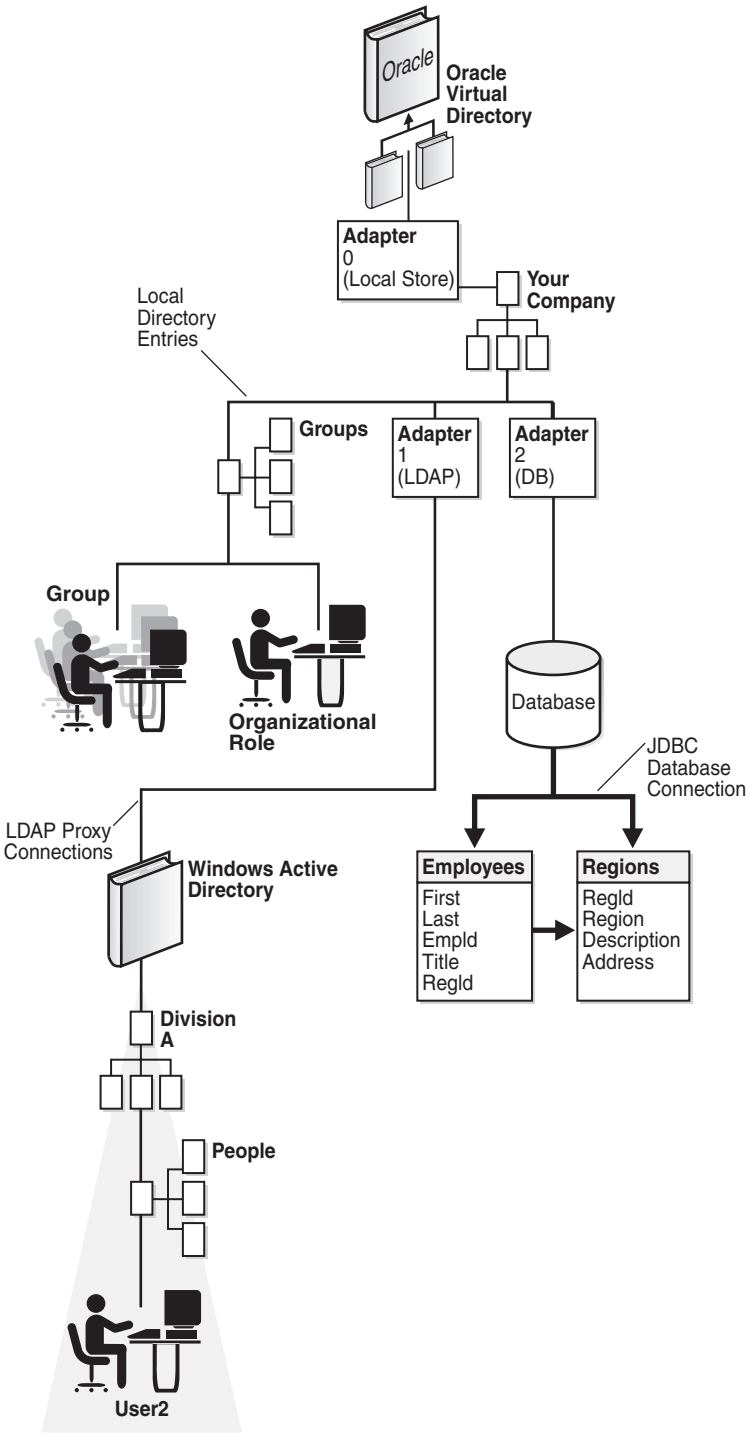
- **Adapter 1: LDAP Adapter**

  This adapter specifies a remote LDAP directory and a remote base which is mapped into the virtual directory tree. In this case, all entries under o=Airius.com in the remote server are made to appear as if they were `ou=Airius,o=YourCompany,c=US` in the virtual directory.

- **Adapter 2: Database Adapter**

  This Database adapter is a database connection specifying that two tables are used to form entries in the directory. In this case, records from the joined queries of two tables will be used to form user objects in the namespace `ou=People,o=YourCompany,c=US` in the virtual directory.

Figure 2–5 Adapters Configured for Example Virtual Directory



As shown in Figure 2–5, Oracle Virtual Directory and its adapters allow different portions of the directory tree to be sourced from different repositories. In planning your virtual directory information tree structure, be sure you do not have two adapter roots that occupy the same root node. However, you may have an adapter appear to be a child node of another adapter.

2.7.2 Example of a Virtual Directory Using the Join View Adapter

Figure 1–2 in Chapter 1, "Understanding Oracle Virtual Directory," shows an example of an enterprise application used by all employees in a company. The application accesses directory information from three different sources and each contains a separate population of users. In Figure 2–6 the topology is the same as in Figure 1–2, however, all three directory sources on the right side of Figure 2–6 *contain the same user population*.

Figure 2–6 Directory Virtualization with the Same User Population

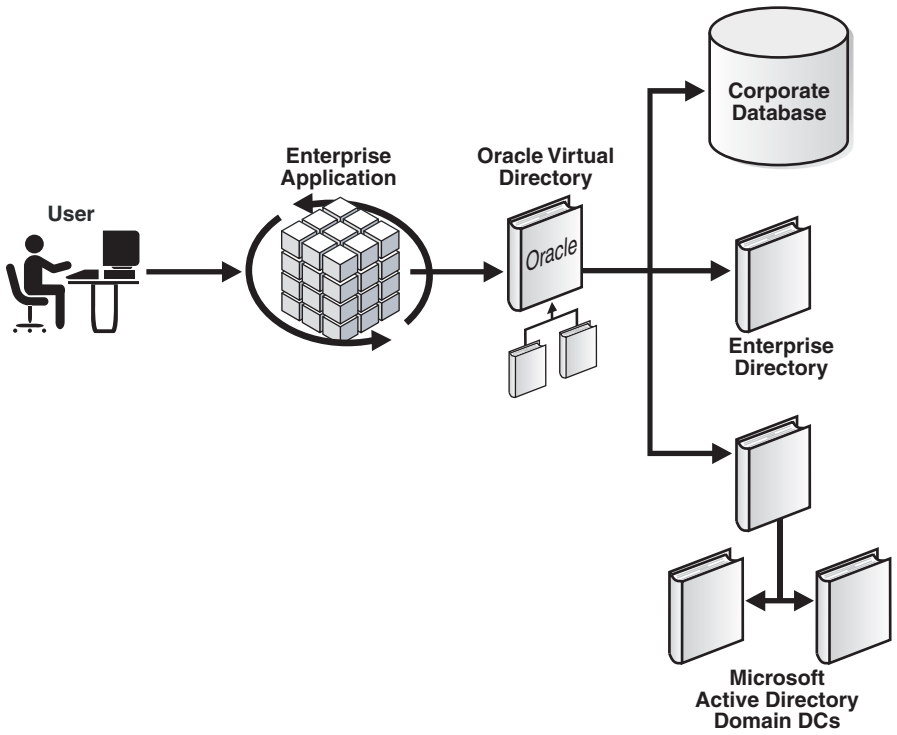


Figure 2–6 shows a main enterprise directory which contains the main source of enterprise directory information for all users. For example, imagine for each user in the enterprise directory you want to match them to a corresponding account in Microsoft Active Directory for user-authentication purposes. Also, to gain access to personnel related application information in the corporate database, you must associate each user in the enterprise directory with a table entry in the enterprise database.

To address the requirements in Figure 2–6, you would configure Oracle Virtual Directory with the following four adapters:

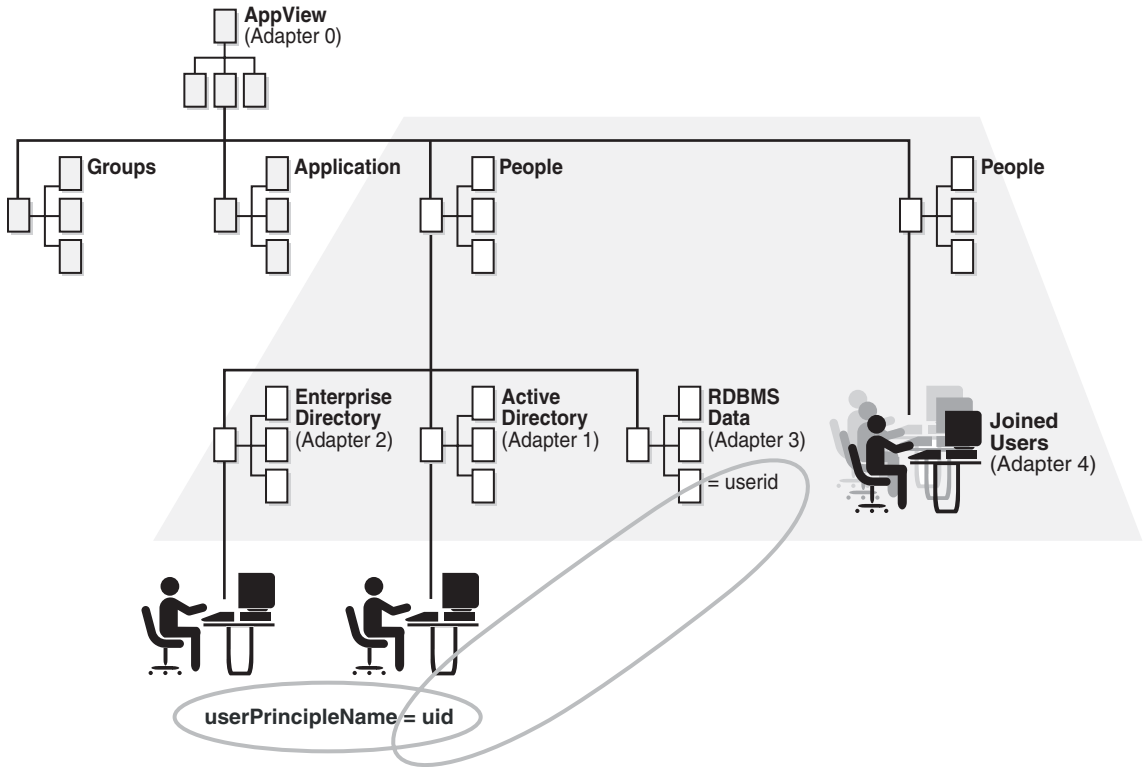
- **Adapter 0** is for local application storage and to hold the root position in the tree.
- **Adapter 1** is defined to proxy to Active Directory.
- **Adapter 2** proxies to the Enterprise LDAP directory.
- **Adapter 3** is a Database Adapter that maps in the appropriate user records within the corporate RDBMS.
- **Adapter 4** is a Join View Adapter. In this case, Adapter 2 for the Enterprise Directory, is used as the primary adapter and therefore all entries displayed by

Adapter 4 will exactly mirror the entries in Adapter 2. With nothing else defined, the Join View Adapter is a carbon copy of Adapter 2.

After configuring the adapters you must define join relationships. In this situation you define two join relationships: one to Active Directory and one to the corporate RDBMS. For the Active Directory join, you define a Simple Join in Adapter 4 to Adapter 2—note that you are really joining with the primary adapter, Adapter 1.

To complete the join, specify unique criteria that joins entries in Active Directory to the primary adapter. As shown in Figure 2–7, use `uid=userprincipalname` where uid is in Adapter 1 and `userprincipalname` is in Adapter 2 (Active Directory). For the second join, you join with Adapter 3 using a Simple Join and use `uid=userid` to achieve a unique match.

Figure 2–7 Specifying Unique Criteria Between Joins



Lastly, because we want to use Adapter 1 for authentication rather than the primary adapter (Adapter 2), you set the bindadapter setting to 1, causing the Join View Adapter to test authentication against the joined adapter rather than the primary adapter.

Note: If you wanted users to match multiple RDBMS records (for example, a privilege table), you could specify a OneToMany Join such as `approle=priv`. In this case `approle` would be an attribute in the enterprise directory. The `approle` attribute matches up with a series of privileges in the RDBMS. By performing the join in this example, you would translate a simple role into a series of privileges.

After Adapter 4 is configured, you can hide Adapters 1, 2, and 3 from end users by setting Routing Visibility on each of these adapters to Internal. This results in a directory that appears to have a single ou=People,o=AppView branch. As you browse the directory below ou=People, you will be querying Adapter 4, the Join View Adapter. When the Join View Adapter receives queries, it automatically transforms and passes them on to the other three hidden adapters. Ultimately the application perceives that there is only a single entry for each person.

2.8 Understanding Adapter Namespaces

Figure 2–8 shows a source directory tree structure where there are four separate directories. The goal is to combine all four separate directories into one new directory tree design. The most basic directory tree design you can implement with Oracle Virtual Directory is with no translation so the source directory structure with four separate directory trees is valid within Oracle Virtual Directory and it is operating as a pure directory proxy using no translation features.

Figure 2–8 Example Source Directory Tree Structure with Four Separate Directories

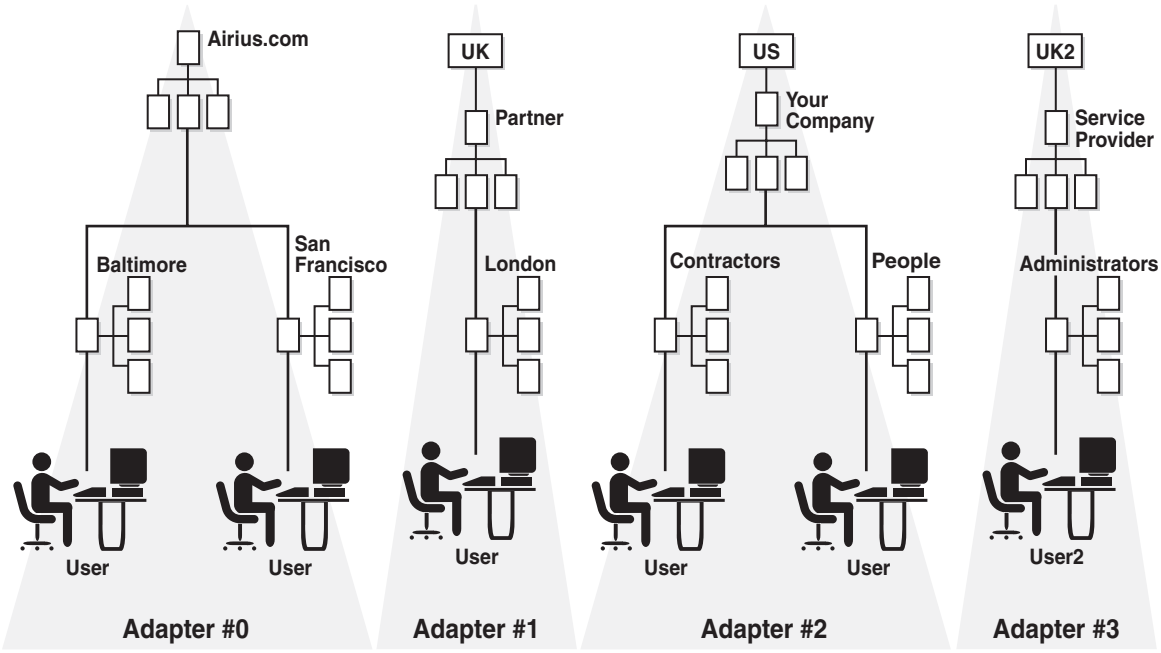


Figure 2–9 shows that two external companies were added as descendants to o=YourCompany, c=US. In this example, Adapter 1 and Adapter 2 occupy subtree entry positions relative to Adapter 0. At the same time, Adapter 3 is left occupying a separate namespace, which could represent a third-party company that provides administrative network services and may not participate in the business application of the other three organizations. In this is the case, it may be best to keep the ISP directory entries separated.

Figure 2–9 Example Source Directory Tree Structure with Two Additions

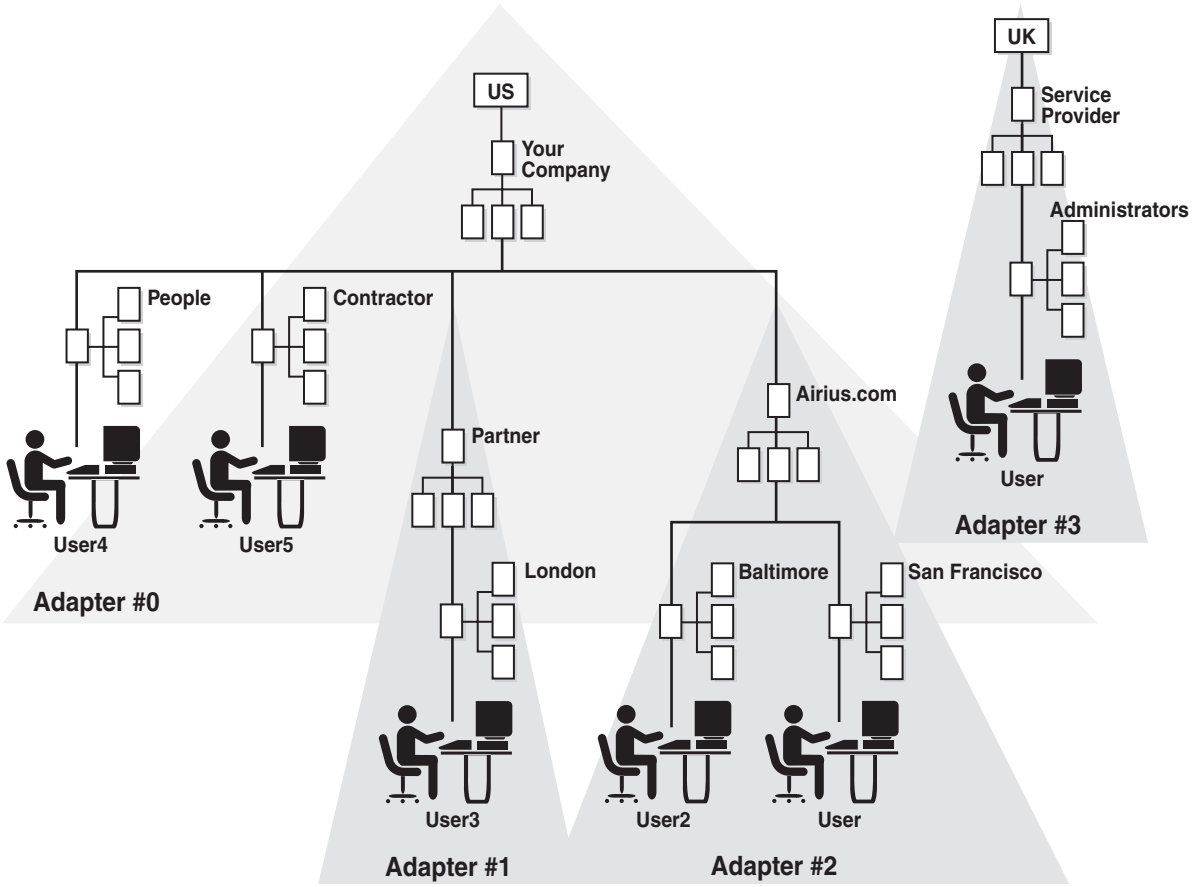


Figure 2–10 shows the directory structure after it was redesigned so that all partner users are under the ou=People branch. This requirement may be due to a limitation in an application (for example, it will only authenticate users from a single base of ou=People, o=YourCompany, c=US). Notice that YourCompany's people entries are directly under ou=People while the partner directories are contained within organization units under ou=People. Keeping the organization unit allows for easier creation of access control groups and roles specific to users of those partners and avoids namespace conflicts. Notice also that only one adapter may occupy any particular node in the tree.

Figure 2–10 Redesigned Source Directory Tree Example

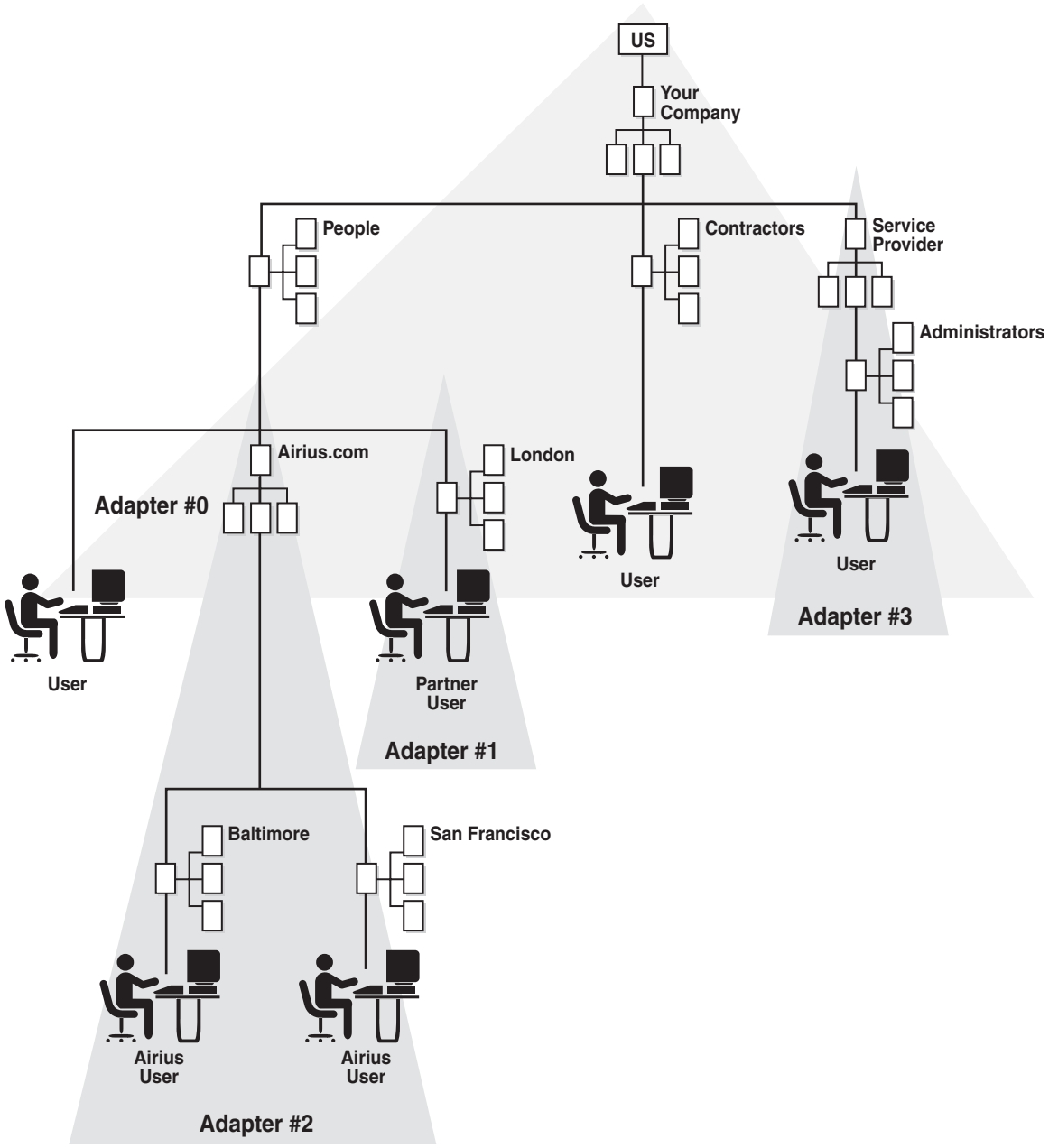
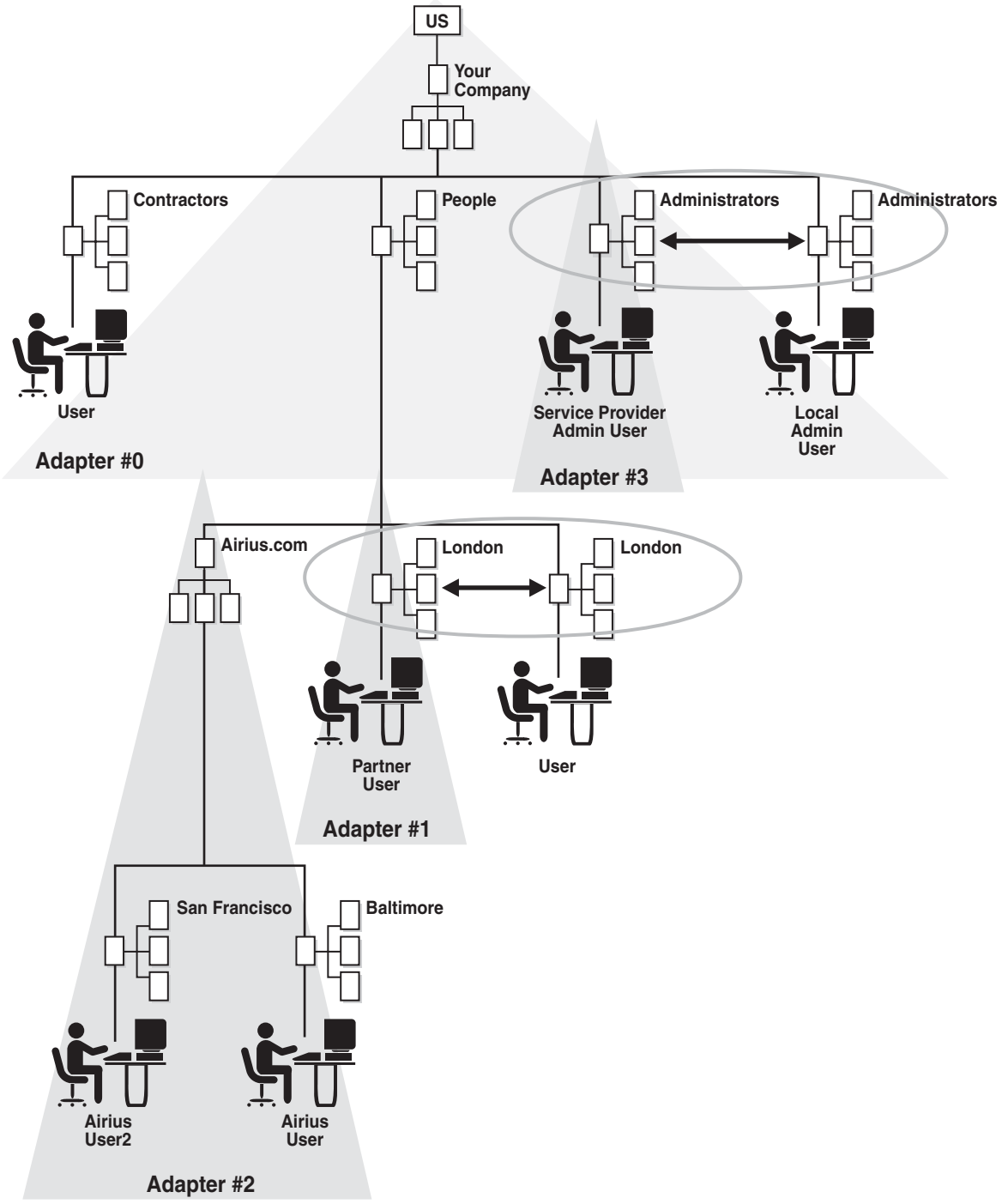


Figure 2–11 shows an example where two problems were created due to conflicts between entries stored in Adapter 0 and the root entry of Adapters 1 and 3. During search operations, Oracle Virtual Directory will search the overlapping namespace and return matches from all adapters. However, during modify operations, Oracle Virtual Directory will only process entries in the adapter that is matched first walking up the tree from the entry (in this example, Adapters 1 and 3 because Adapter 0 is further up the tree).

Figure 2–11 Example of an Overlapping Directory Structure



Traditionally, the overlapped directory structure would be unacceptable, however, Oracle Virtual Directory supports routing filters that control how it handles operations to overlapping namespaces.

2.9 Understanding Adapter Templates

Oracle Virtual Directory includes adapter templates to help simplify the process for configuring adapters. When you create an adapter, the New Adapter screen contains

an Adapter Template list that displays the available templates for each adapter. After selecting an adapter template, Oracle Directory Services Manager automatically populates default values for some of the adapter settings. You should alter these default settings according to your environment.

The following sections list and describe the adapter templates:

- Default Template
- LDAP Adapter Templates
- Local Store Adapter Templates
- Database Adapter Templates

2.9.1 Default Template

The Default template is a general template available for all adapter types and is not specific to any one vendor's directory. You should use the Default template if no other template satisfies your needs.

2.9.2 LDAP Adapter Templates

The following sections describe the LDAP Adapter templates:

- Active_Directory
- CA_eTrust
- EUS_ActiveDirectory
- EUS_OID
- EUS_Sun
- General_LDAP_Directory
- IBM_Directory
- Novell_eDirectory
- OAM/AD Adapter with Mapper
- OAM/AD Adapter with SSL, Mapper
- OAM/AD Adapter with Script
- OAM/ADAM Adapter with Mapper
- OAM/ADAM Adapter with SSL, Mapper
- OAM/ADAM Adapter with Script
- OAM/SunOne Adapter with Mapper
- OAM/SunOne Adapter with Script
- Oracle_Internet_Directory
- Siemens_DirX
- SunOne_Directory

Note: Some adapter templates use Python mapping scripts. The template will configure the mapping script with the adapter's information, but the template does not deploy the mapping script. If you use an adapter template that uses a mapping script, you must explicitly deploy the mapping script to the Oracle Virtual Directory server after configuring the adapter.

2.9.2.1 Active_Directory

Use the Active_Directory template when connecting to a Microsoft Active Directory or Active Directory Application Mode target and you do not want to map Active Directory objects to InetOrgPerson objects.

Note: If you are connecting to a Microsoft Active Directory or Active Directory Application Mode target and you do want to map Active Directory objects to InetOrgPerson objects, use the OAM/AD Adapter with Mapper template even if you are not integrating with Oracle Access Manager.

2.9.2.2 CA_eTrust

Use the CA_eTrust template when connecting to a Computer Associates (CA) eTrust directory.

2.9.2.3 EUS_ActiveDirectory

Use the EUS_ActiveDirectory template for an Oracle Virtual Directory-Enterprise User Security integration that uses Active Directory. The EUS_ActiveDirectory simplifies the integration by deploying the following plug-ins:

- **Objectclass Mapper**: Maps certain Oracle attributes and object classes so they can be managed in Active Directory.
- **Active Directory Password**: Allows storage of database password into Active Directory when database registers with the directory.
- **EUSActiveDirectory**: Converts certain Active Directory attributes, such as GUID, into a format that Enterprise User Security can use.

2.9.2.4 EUS_OID

Use the EUS_OID template for an Oracle Virtual Directory-Enterprise User Security integration that uses Oracle Internet Directory. The EUS_OID simplifies the integration by deploying the EUSOID plug-in, which converts certain attributes to a consistent format for use with Enterprise User Security.

2.9.2.5 EUS_Sun

Use the EUS_Sun template for an Oracle Virtual Directory-Enterprise User Security integration that uses Sun Java System Directory Server. The EUS_Sun simplifies the integration by deploying the following plug-ins:

- **Objectclass Mapper**: Maps certain Oracle attributes and object classes so they can be managed in Sun Java System Directory Server.
- **EUSSun**: Converts certain Sun Java System Directory Server attributes, such as GUID, into a format that Enterprise User Security can use.

2.9.2.6 General_LDAP_Directory

The General_LDAP_Directory template is identical to the Default Template.

2.9.2.7 IBM_Directory

Use the IBM_Directory template when connecting to an IBM Directory Server.

2.9.2.8 Novell_eDirectory

Use the Novell_eDirectory template when connecting to a Novell eDirectory.

2.9.2.9 OAM/AD Adapter with Mapper

Oracle suggests using this template for an Oracle Virtual Directory-Oracle Access Manager integration that uses Microsoft Active Directory, though other applications can benefit from using this template. The OAM/AD Adapter with Mapper template simplifies the LDAP Adapter's interaction with Active Directory by deploying the following plug-ins:

- **Active Directory Ranged Attributes**: converts ranged attributes, which are attributes with several multi-valued values that Active Directory splits into multiple requests, often confusing clients, into a single request.
- **ObjectClass Mapper**: maps Active Directory User/Group objects into InetOrgPerson/GroupOfUniqueName objects.
- **ActiveDirectory Password**: converts the standard userPassword attribute into Microsoft's unicodePWD attribute. Additionally, if you want to have the adapter connect to Active Directory over a non-SSL port, this plug-in can route password updates to a different adapter instance that connects to the Active Directory server over SSL (because Active Directory requires password changes over LDAP to use SSL). If the adapter is set to use SSL, remove the host name from the plug-in configuration. If the adapter is set not to use SSL, set the plug-in host name to the name of the Active Directory adapter connected to Active Directory over SSL.
- **Dump Before**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log before passing data to plug-ins.
- **Dump After**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log after passing data to plug-ins.

> **Note:** The OAM/AD Adapter with Mapper template is similar to the OAM/AD Adapter with Script template but it does not require you to deploy an additional mapping script like the OAM/AD Adapter with Script template does.

2.9.2.10 OAM/AD Adapter with SSL, Mapper

Configures an LDAP Adapter to connect to an Active Directory target over SSL for password change operations only in an Oracle Virtual Directory-Oracle Access Manager integration. By default, the adapter's Visibility routing setting is set to internal, hiding the adapter to clients and making it accessible only through plug-ins like the Active Directory Password plug-in.

2.9.2.11 OAM/AD Adapter with Script

Similar to the OAM/AD Adapter with Mapper template except that it uses the OblixADMapping Python mapping script to do attribute renaming instead of the

ObjectClass mapper. The OAM/AD Adapter with Script template simplifies the LDAP Adapter's interaction with Active Directory by deploying the following plug-ins:

- **Active Directory Ranged Attributes**: converts ranged attributes, which are attributes with several multi-valued values that Active Directory splits into multiple requests, often confusing clients, into a single request.
- **ActiveDirectory Password**: converts the standard userPassword attribute into Microsoft's unicodePWD attribute. Additionally, if you want to have the adapter connect to Active Directory over a non-SSL port, this plug-in can route password updates to a different adapter instance that connects to the Active Directory server over SSL (because Active Directory requires password changes over LDAP to use SSL). If the adapter is set to use SSL, remove the host name from the plug-in configuration. If the adapter is set not to use SSL, set the plug-in host name to the name of the Active Directory adapter connected to Active Directory over SSL.
- **Dump Before**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log before passing data to plug-ins.
- **Dump After**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log after passing data to plug-ins.

The OAM/AD Adapter with Script template also configures the OblixADMapping script using the adapter's information. The OblixADMapping script is similar to ObjectClass mapper which maps Active Directory User/Group objects into InetOrgPerson/GroupOfUniqueName objects.

Note: You must explicitly deploy the OblixADMapping mapper script to the Oracle Virtual Directory server after configuring the adapter with the OAM/AD Adapter with Script template.

If you can use either the OAM/AD Adapter with Script template or the OAM/AD Adapter with Mapper template to obtain equal results, you may want to use the OAM/AD Adapter with Mapper template because the OAM/AD Adapter with Script template requires you to explicitly deploy the OblixADMapping mapper script to the Oracle Virtual Directory server after configuring the adapter and the OAM/AD Adapter with Mapper template does not.

2.9.2.12 OAM/ADAM Adapter with Mapper

Oracle suggests using this template for an Oracle Virtual Directory-Oracle Access Manager integration that uses Microsoft Active Directory Application Mode, though other applications can benefit from using this template. The OAM/ADAM Adapter with Mapper template simplifies the LDAP Adapter's interaction with Active Directory Application Mode by deploying the following plug-ins:

- **Active Directory Ranged Attributes**: converts ranged attributes, which are attributes with several multi-valued values that Active Directory splits into multiple requests, often confusing clients, into a single request.
- **ObjectClass Mapper**: maps Active Directory User/Group objects into InetOrgPerson/GroupOfUniqueName objects.
- **ActiveDirectory Password**: converts the standard userPassword attribute into Microsoft's unicodePWD attribute. Additionally, if you want to have the adapter connect to Active Directory over a non-SSL port, this plug-in can route password updates to a different adapter instance that connects to the Active Directory server over SSL (because Active Directory requires password changes over LDAP to use

SSL). If the adapter is set to use SSL, remove the host name from the plug-in configuration. If the adapter is set not to use SSL, set the plug-in host name to the name of the Active Directory adapter connected to Active Directory over SSL.

- **Dump Before**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log before passing data to plug-ins.
- **Dump After**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log after passing data to plug-ins.

Note: The OAM/ADAM Adapter with Mapper template is similar to the OAM/ADAM Adapter with Script template, but it does not require you to deploy an additional mapping script like the OAM/ADAM Adapter with Script template does.

2.9.2.13 OAM/ADAM Adapter with SSL, Mapper

Configures an LDAP Adapter to connect to an Active Directory Application Mode target over SSL for password change operations only in an Oracle Virtual Directory-Oracle Access Manager integration. By default, the adapter's Visibility routing setting is set to internal, hiding the adapter to clients and making it accessible only through plug-ins like the Active Directory Password plug-in.

2.9.2.14 OAM/ADAM Adapter with Script

Similar to the OAM/ADAM Adapter with Mapper template except that it uses the OblixADMapping Python mapping script to do attribute renaming instead of the ObjectClass mapper. The OAM/ADAM Adapter with Script template simplifies the LDAP Adapter's interaction with Active Directory Application Mode by deploying the following plug-ins:

- **Active Directory Ranged Attributes**: converts ranged attributes, which are attributes with several multi-valued values that Active Directory splits into multiple requests, often confusing clients, into a single request.
- **ActiveDirectory Password**: converts the standard userPassword attribute into Microsoft's unicodePWD attribute. Additionally, if you want to have the adapter connect to Active Directory over a non-SSL port, this plug-in can route password updates to a different adapter instance that connects to the Active Directory server over SSL (because Active Directory requires password changes over LDAP to use SSL). If the adapter is set to use SSL, remove the host name from the plug-in configuration. If the adapter is set not to use SSL, set the plug-in host name to the name of the Active Directory Application Mode adapter connected to Active Directory over SSL.
- **Dump Before**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log before passing data to plug-ins.
- **Dump After**: a version of the Dump Transaction plug-in that dumps the values of the operation to vde.log after passing data to plug-ins.

The OAM/ADAM Adapter with Script template also configures the OblixADMapping script using the adapter's information. The OblixADMapping script is similar to ObjectClass mapper which maps Active Directory User/Group objects into InetOrgPerson/GroupOfUniqueName objects.

Note: You must explicitly deploy the OblixADMapping mapper script to the Oracle Virtual Directory server after configuring the adapter with the OAM/ADAM Adapter with Script template.

If you can use either the OAM/ADAM Adapter with Script template or the OAM/ADAM Adapter with Mapper template to obtain equal results, you may want to use the OAM/ADAM Adapter with Mapper template because the OAM/ADAM Adapter with Script template requires you to explicitly deploy the OblixADMapping mapper script to the Oracle Virtual Directory server after configuring the adapter and the OAM/ADAM Adapter with Mapper template does not.

2.9.2.15 OAM/SunOne Adapter with Mapper

Configures an LDAP Adapter to connect to a SunOne directory target in an Oracle Virtual Directory-Oracle Access Manager integration and converts SunOne attributes for use with Oracle Access Manager. The OAM/SunOne Adapter with Mapper template simplifies the LDAP Adapter's interaction with SunOne by deploying the following plug-ins:

- **ObjectClass Mapper**: Filters out the nsaccountlock attribute and marks the directory type as SunOne.
- **Dump SunOne**: Dumps output of plug-in activity to vde.log.

2.9.2.16 OAM/SunOne Adapter with Script

Similar to the OAM/SunOne Adapter with Mapper template except that it uses the OblixSunOneMapping Python mapping script to do attribute renaming instead of the ObjectClass mapper. Oracle suggests using the OAM/SunOne Adapter with Mapper template instead of the OAM/SunOne Adapter with Script template for first and fresh installations.

The OAM/SunOne Adapter with Script template simplifies the LDAP Adapter's interaction with SunOne by deploying the Dump Transactions plug-in, which dumps output of plug-in and mapping activity to vde.log.

The OAM/SunOne Adapter with Script template also configures the OblixSunOneMapping script using the adapter's information. The OblixSunOneMapping is similar to ObjectClass mapper which filters out the nsaccountlock attribute and marks the directory type as SunOne.

Note: You must explicitly deploy the OblixSunOneMapping mapper script to the Oracle Virtual Directory server after configuring the adapter with the OAM/SunOne Adapter with Script template.

If you can use either the OAM/SunOne Adapter with Script template or the OAM/SunOne Adapter with Mapper template to obtain equal results, you may want to use the OAM/SunOne Adapter with Mapper template because the OAM/SunOne Adapter with Script template requires you to explicitly deploy the OblixSunOneMapping mapper script to the Oracle Virtual Directory server after configuring the adapter and the OAM/SunOne Adapter with Mapper template does not.

2.9.2.17 Oracle_Internet_Directory

Use the Oracle_Internet_Directory template when connecting to an Oracle Internet Directory (OID).

2.9.2.18 Siemens_DirX

Use the Siemens_DirX template when connecting to a Siemens DirX directory.

2.9.2.19 SunOne_Directory

Use the SunOne_Directory template when connecting to any directory in the Netscape family of directories, including Netscape, Sun Microsystems, and Fedora.

2.9.3 Local Store Adapter Templates

The following sections describe the Local Store Adapter templates:

- Local_Storage_Adapter

2.9.3.1 Local_Storage_Adapter

The Local_Storage_Adapter template is identical to the Default Template.

2.9.4 Database Adapter Templates

The following section describes the Database Adapter templates:

- OAM/DB Adapter with Script

2.9.4.1 OAM/DB Adapter with Script

Configures a Database Adapter to connect to a database target in an Oracle Virtual Directory-Oracle Access Manager integration and uses a Python mapping script handle business logic. The OAM/DB Adapter with Script template simplifies the Database Adapter's interaction with Oracle Access Manager by deploying the following plug-ins:

- **DumpDB1**: a version of the Dump Transaction plug-in that dumps the output of operations to vde.log before passing data to plug-ins and mappings.
- **DumpDB2**: a version of the Dump Transaction plug-in that dumps the output of operations to vde.log after passing data to plug-ins and mappings.

The OAM/DB Adapter with Script template also configures the Oblix_OAMMapping script using the adapter's information. The Oblix_OAMMapping script provides business logic for the Oracle Access Manager integration, such as removing Oracle Access Manager specific objectclasses that must be removed before entries can be added.

Note: You must explicitly deploy the Oblix_OAMMapping mapper script to the Oracle Virtual Directory server after configuring the adapter with the OAM/DB Adapter with Script template.

3

Understanding Oracle Virtual Directory Routing

This chapter describes Oracle Virtual Directory routing and includes the following topics:

- What is Routing?
- Understanding Routing Settings

3.1 What is Routing?

In a traditional directory server, multiple databases are defined and each are responsible for part of the directory tree namespace and selection is determined strictly on namespace comparison. In a virtual directory, since it is possible to have multiple adapters sharing the same namespace, selection is more complex—yet more controllable.

Routing is the process by which Oracle Virtual Directory decides which adapter should be selected for an LDAP operation. Routing is applied to all adapters regardless of type and serves several purposes, including:

- limiting the number of adapters selected to just the ones which contain the requested client data and are relevant to the current LDAP operation.
- enabling you to design for complex environments.
- enabling you to tune Oracle Virtual Directory to implement a more secure, higher-performing configuration by reducing the number of adapters for a particular transaction.

Routing controls adapter selection by examining not just the basic DN namespace, but also other aspects of transaction information including DN pattern matching, LDAP filters, attributes filters, and query filters. At its most basic level, Oracle Virtual Directory can select adapters through a process of adapter suffix comparison. The adapter suffix comparison involves looking at any particular search base or entry DN, such as in the case of add, modify, delete, and rename, and then comparing it with the suffix (root) of each adapter. Depending on the scope, Oracle Virtual Directory can determine if one or more adapters was impacted by any particular query.

Adapter suffix comparison works well with a small number of adapters, however, more flexible decisions are usually required—where routing is explicitly important. Routing lets administrators teach Oracle Virtual Directory about proxied data sources in the form of routing intelligence. Routing allows Oracle Virtual Directory to further qualify directory operations and send them to the specific places where they are needed, which helps keep existing directories from being overloaded with irrelevant

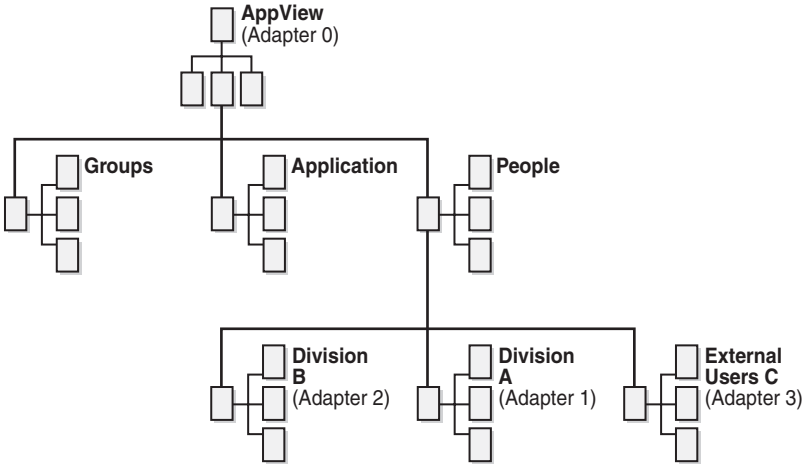
operations and keeps partners from seeing queries that are not related to their own directory. The Oracle Virtual Directory routing process analyzes LDAP client search filters in addition to traditional adapter suffix comparison and further refines eligible adapters for processing.

Routing Example

Consider the example virtual directory structure shown in Figure 3–1 that has the following four adapters configured:

- **Adapter 0** forms the root of the directory tree and maps to o=AppView. This adapter holds the virtual root of the tree and local entries such as access control groups.
- **Adapters 1-3** map each directory source to positions beneath the ou=People branch of the new application tree.

Figure 3–1 Example Virtual Directory Structure



For example, say an application that uses the directory in Figure 3–1 has little intelligence with regard to a directory service and it was originally designed for a single business and does not understand that multiple business user groups may be using the same application. Instead of expecting a varied and diverse directory tree structure, the application only searches the directory from one common directory hierarchy point (or one common base). For this example, say the application only searches the directory from ou=People,o=AppView. When a user enters a login credential such as jim.smith@divisionB.com, the application issues the following search:

- **base**: ou=People,o=AppView
- **scope**: subtree
- **filter**: (uid=jim.smith@divisionb.com)

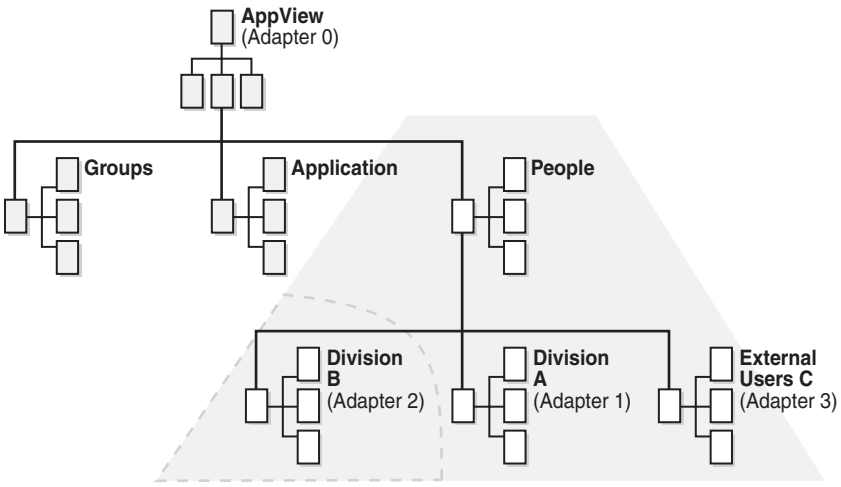
After receiving this query, Oracle Virtual Directory automatically selects all adapters eligible for this query. Since the query is at the base of the tree, all adapters are selected, leading to a performance problem to examine. If all the other companies exist lower in the directory structure (for example, ou=DivisionB, ou=People,o=AppView), then by default, all directory sources are searched because their branches are below the parent ou=People,o=AppView.

To resolve this issue, Oracle Virtual Directory provides routing inclusion and exclusion filters. You can use these filters to filter traffic for any particular partner directory. In this example, the administrator can set up the following Routing Include filters:

- **Division A Adapter**: (uid=*@divisiona.com)
- **Division B Adapter**: (uid=*@divisionb.com)
- **Division C Adapter**: (uid=*@divisionc.com)

Even though the base of the LDAP client search would normally have selected all directories, the filters specify that the search for (uid=jim.smith@divisionb.com) should go only to the Division B directory. Figure 3–2 shows the three shaded adapters that would normally be selected, while the dotted area shows that after filter processing, only Division B's data is searched.

Figure 3–2 Example of Adapter Search With Filters



In addition to filtering queries, Oracle Virtual Directory also lets you assign priorities to each adapter. The adapter with the lower priority number is always queried first. Adapters with the same priority number are searched in order of definition in the configuration file. When conflicts occur, for example, two entries with the same DN, Oracle Virtual Directory will always accept only the response from the lower numbered adapter in priority or configuration. When routing filters fail to select a single adapter, potential conflicts are resolved by priority selection.

3.2 Understanding Routing Settings

After you create an adapter, you can configure the routing for that adapter using the adapter's Routing tab in Oracle Directory Services Manager. This topic describes the adapter routing settings available on the Routing tab and includes the following sections:

- Priority
- Filters to Include and Filters to Exclude
- DN Matching
- Levels
- Attribute Flow Settings

- Visibility
- Bind Support
- Criticality
- Views
- Include Binds From and Exclude Binds From

> **Note:** Click the **Apply** button on an adapter's Routing tab to apply changes you made to the adapter's Routing settings. Click the **Revert** button to revert (go back) to the Routing settings that were configured before you made changes. You cannot revert the settings after clicking **Apply**.

3.2.1 Priority

Sometimes it may be necessary to constrain Oracle Virtual Directory to process certain adapters before others, for example, when two or more adapters have overlapping namespaces. This situation can occur when bringing new directories into service while the existing directories must remain online.

The Priority setting determines the priority with which the adapter is to be treated relative to other adapters. 1 is the highest priority, 100 is the lowest priority, and 50 is the default setting.

In the example situation described above when bringing new directories into service while the existing directories must remain online, the Priority setting of the newer, more significant adapter should be set to a higher priority—that is, a number lower than the default 50 and also lower in respect to the existing adapter whose namespace overlaps with it.

Priority is used as the last chance selector when all other routing parameters have been processed. Given two otherwise equal candidates, the adapter with the higher priority, meaning lower number, is processed first. Adapters with the same priority number are searched in order of definition in the configuration file. When conflicts occur for a search operation, for example, two adapters who support the same DN, Oracle Virtual Directory will use the adapter with lowest priority number in the configuration first. During modify operations, Oracle Virtual Directory will only process entries within the adapter that are matched first moving up the tree from the entry.

> **Note:** For maximum precision, Oracle recommends using the Filters to Include, Filters to Exclude, and DN Matching settings to arbitrate in configurations where multiple adapters may be selected.

3.2.2 Filters to Include and Filters to Exclude

The Filters to Include and Filters to Exclude settings are essentially filters of a filter and apply to the LDAP search filters specified by a client. If a client search filter fulfills the logical requirements defined in the Filters to Include setting, that adapter is selected for inclusion in the set of adapters used in the search. Similarly, for the Filters to Exclude setting, if the logical requirements are met, that adapter is deselected from the set of adapters used in the client search.

The format for the Filters to Include and Filters to Exclude fields is a standard LDAP search filter followed by a scope term— either #base, #one, or #sub. The scope indicates at what scope level the filter should be applied. For example, filters using the

#one scope apply to one level or sub tree searches and base searches would not be filtered.

The default scope for an include filter is #sub to filter out only queries involving an entire sub tree. To apply the filter applied for all scopes, set the scope to #base, which means the filter is applied to base, one-level, and sub-tree searches.

The default scope for an exclude filter is #one to allow blocking of specific searches. To apply the filter for all scopes, set the scope to #base. To apply the filter for just sub-tree searches, set the scope to #sub.

Both the Filters to Include and Filters to Exclude setting can be used together to form a more complex set of conditions governing the adapters used in a client search operation. For example, imagine you want to allow specific types of searches through an LDAP Adapter deployed as a firewall. To allow only certain searches, you could use a Filters to Include setting such as:

```
(|(mail=*@myorg.com)(uid=*@myorg.com)(sn=*)(givenname=*)(cn=*))
```

This filter would block any search with terms other than mail, uid, sn, givenname, or cn and allow only searches involving one or more of these terms. For example (cn=Jim Smith) is acceptable, while (uid=smith@oracle.com) is not acceptable since it does not end in myorg.com

Although most adapter configurations use simple search terms, a more complex example may better illustrate how the logic is applied. Consider the following filter example:

Client Search Command

```
$ ldapsearch -b dc=oracle,dc=com -s sub "(|(sn=user2)(cn=user2b))"
```

Routing Filter:

```
(&(|(uid=*)(cn=*))(sn=*))
```

The routing filter indicates that if the client search filter contains an sn attribute and either a uid or cn term, than a match is made. In this example, without regard to other conditions, the adapter would be selected if the given routing filter were assigned to the Filters to Include setting and would be deselected if assigned to Filters to Exclude setting because the client filter includes an sn term and a cn term which fulfills the logic of the filter.

3.2.3 DN Matching

DN Matching is most often used when you want to have adapters sharing the same adapter root and you need a way to arbitrate which entries belong to which adapter. DN Matching allows you to exploit the differences in naming that might occur between two proxied sources. For example, in a large scale deployment you may wish to divide the entries based on the alphabet. DN Matching allows you to select alphabetic ranges and then allows Oracle Virtual Directory to select adapters based on range match. Thus, if you divided names into three ranges, users with IDs beginning A through J could be one directory, K through R might be another directory, and S through Z might be the final directory portion.

Another useful scenario for DN Matching is federating Microsoft Active Directory users with users in an external directory such as Open LDAP or some other directory. If the users in Open LDAP have relative distinguished names (RDNs) that are based on the uid attribute and Active Directory has user entries based on the cn attribute,

then you can establish a regular expression that selects adapters based on the RDN type.

For Active Directory Adapter, the DN match might be:

```
(.*)cn=[a-z0-9]*$
```

For Open LDAP, the DN match might be:

```
(.*)uid=[a-z0-9]*$
```

By using DN Matching, Oracle Virtual Directory can effectively manage overlapped adapters by exploiting the differences in the existing sources.

In the DN Matching field you can enter a regular expression indicating how DN's within the adapter must be formed. The regular expression applies to the portion of the DN below the adapter's root. For example, if the adapter's root is ou=People,o=MyBigOrg.com and you only want to allow entries in the next level whose RDN begins with the letters A through J, you can specify an expression such as:

```
m/^(.*)uid=[a-j][a-z0-9]*$/
```

This expression indicates that the DN must contain a uid= term, followed by the letters A thru J, followed by any number of alpha numeric characters. The $ sign indicates the end of the string. In this case, because a comma is excluded at the end of the string, the uid= must be the last component of the DN within this adapter. Because the UID value must begin with A through J, then only UIDs matching that criteria are accepted. Finally, the ^(.*) part of the regular expression indicates that any number of characters of any type can occur between the start of the string (indicated by ^) and the specific value uid=.

Notes:

- Because DN's are case-insensitive, regular expression matching is performed in a case-insensitive manner.
- The m/ and trailing / part of the match expression is optional.

3.2.4 Levels

When using multiple adapters where some adapters are children of other adapters, it may be desirable to configure the parent adapter so that queries occurring within the namespace of a child adapter are not also sent to the parent adapter. This happens when the DN of an LDAP operation pertains to both a child adapter and a parent adapter through normal namespace selection. By setting the depth, or level of the parent adapter, Oracle Virtual Directory can eliminate the parent adapter from participating in child transactions.

Used in conjunction with LDAP searches, the routing Levels setting determines how many levels below the adapter root the search base may be. For example, a value of 0 requires the search base to be the same as the adapter root, a value of 1 allows the search base to be at the adapter root or one level down, and so on. An empty (blank) **Levels** setting, which is the default setting, allows searches at all levels.

The Levels setting is useful as a performance parameter when mixing highly nested multiple adapter scenarios. Although the root adapter has the potential for being selected for all queries of a virtualized tree, this may not be desirable since other adapters may be set to point to parts of the tree containing the relevant data. To keep

the root adapter out of all queries except those actually examining the root entry, thus increasing server performance, the Levels setting should be set to 0.

For example, if a Local Store Adapter was defined to be o=Oracle.com, it might be used to be a common parent for a series of LDAP Adapters who will be ou=Partner1, o=Oracle.com and ou=Partner2, o=Oracle.com, and so on. In this case, o=Oracle.com is a place holder for the child adapters. Since the adapter has only one entry, it only needs to be queried for operations where the search base is specifically o=Oracle.com. The adapter does not need to be searched when the search base is ou=Partner1, o=Oracle.com. In this case, a routing Levels value of 0 is appropriate.

3.2.5 Attribute Flow Settings

The Attribute Flow routing settings control how attributes flow into and out of an adapter. The Attribute Flow routing settings provide security by preventing information from being requested or returned to an unauthorized client. Also, for Join View adapters, the Attribute Flow routing settings control which attributes flow to which adapters since multiple adapters can contribute to the same virtual joined entry.

> **Note:** Unlike access controls, attribute flow rules provide quiet enforcement—they simply filter the request without returning an error to the client. In a high security setting this quiet enforcement prevents the client from knowing whether or not they are even allowed to see a particular attribute.

The following is a list of the Attribute Flow routing settings. Each setting is described in detail in the remaining subsections in this section:

- Retrievable Attributes
- Unretrievable Attributes
- Storeable Attributes
- Unstoreable Attributes

3.2.5.1 Retrievable Attributes

The Retrievable Attributes setting controls which attributes may be retrieved by the adapter on the target directory. The Retrievable Attributes setting contributes to server performance and in some cases, security, since only the attributes named can be requested from a proxied server for add, modify, delete operations.

Additionally, you can use the Retrievable Attributes setting to control attribute flow when using the Join View Adapter. Since a Join View Adapter joins entries from two or more adapters, you need to control which attributes should come from the participating adapters. To control which attributes can come from the participating adapters in the Join View, configure the Retrievable Attributes settings on each adapter in the Join View.

In the Retrievable Attributes field, identify an explicit list of attributes that may be retrieved from an adapter. An empty list implies all attributes are retrievable. A specific list in the Retrievable Attributes field indicates that only the listed attributes may be requested from the proxied directory.

Note: DN and objectclass are always returned from ldapsearch regardless of an adapter's Retrievable Attributes routing settings. If needed, you can use a plug-in, such as the ObjectClass Mapper, to modify a DN or objectclass.

3.2.5.2 Unretrievable Attributes

The Unretrievable Attributes setting controls which attributes may not be retrieved by the adapter on the target directory. An empty list implies all attributes are retrievable.

3.2.5.3 Storeable Attributes

The Storeable Attributes setting controls which attributes may be stored by the adapter on the target directory. The Storeable Attributes setting contributes to server performance and in some cases, security, since only specific attributes and their values may be sent to the proxied server for add, modify, delete operations.

Additionally, you can use the Storeable Attributes setting to control attribute flow when using the Join View Adapter. Since a Join View Adapter joins entries from two or more adapters, you need to control which attributes should go to the participating adapters. To control which attributes can go to the participating adapters in the Join View, configure the Storeable Attributes settings on each adapter in the Join View.

In the Storeable Attributes field, enter a list of attributes that may be written to the adapter. An empty list implies all attributes are storable—unless Unstoreable Attributes are defined. If Unstorable Attributes are specified, only the specific values listed in the Storeable Attributes field are storable.

To make an adapter read only, enter `_never` in the list of Storable Attributes. The _ character is illegal in an attribute name and the condition can never be true, causing the adapter to be read only.

3.2.5.4 Unstoreable Attributes

Use this list if it is easier to express which attributes cannot be modified, rather than those that can be modified (as indicated using the Storeable Attributes field). Normally either a Storable Attributes list or an Unstorable Attribute list is specified, but not both.

3.2.6 Visibility

An adapter's Visibility routing setting controls whether an adapter can be queried by an external client and whether it is published in the server namingcontexts attribute under the root entry. The following is a list and description of each Visibility setting:

Note: The Visibility options are listed in the Oracle Directory Services Manager interface in English only, however the description for each Visibility option is supported in localized language translations.

Yes

The default setting, a visible adapter is an adapter whose root is published to the servers root entry as part of the namingcontexts attribute.

No

When visibility is set to No, the adapter is not listed in the namingcontexts attribute, but is still available to external LDAP clients. This is useful when you have multiple

adapters working together to form a single directory tree branch. Rather than publish the parent and all of the child adapters in namingcontexts, you can publish just the root adapter since the whole logical tree is implied and publishing the child adapters would be redundant or confusing to applications.

Internal

An Internal adapter is an adapter that is only available to plug-ins and Join View adapters running inside of Oracle Virtual Directory. Internal adapters are not available for use by external LDAP clients. An example of this is an adapter configured for use by a Join View adapter. Rather than publish information twice in the external virtual directory, the primary and joiner adapters can be marked as internal so that only the Join View Adapter may access the information defined in these adapters.

3.2.7 Bind Support

The Bind Support option indicates whether the adapter can process LDAP bind operations. If the adapter does not support a bind function, Oracle Virtual Directory will attempt to obtain the userPassword attribute from the entry corresponding to the DN specified and will perform a local password compare operation. This is equivalent to having the Pass-through Mode setting set to Never in an LDAP Adapter. Enable the Bind Support setting when defining Custom Adapters that may or may not support a bind operation.

3.2.8 Criticality

When a search operation with an adapter fails due to a host error, Oracle Virtual Directory reacts according to the Criticality setting. The following is a list and brief description of each of the Criticality settings:

Note: The Criticality options are listed in the Oracle Directory Services Manager interface in English only, however the description of the Criticality field is supported in localized language translations.

True

The default setting, this setting indicates that if the adapter fails to return a result, for example, due to an operations error or when all remote hosts have failed, Oracle Virtual Directory will cause the overall search operation to fail and return a DSA Unavailable error to the client regardless of whether data was found in any other adapter or not.

False

This setting instructs Oracle Virtual Directory that the failure to perform an operation in the adapter is not critical to the overall result. If a non-critical adapter incurs an operations error, than the result is simply omitted from the overall LDAP search results and Oracle Virtual Directory returns partial results from any other adapters and does not indicate any error.

Partial

Setting the adapter criticality to Partial means the application can notify its own users that partial results were obtained. When an error occurs, Oracle Virtual Directory will return an Admin Limit Exceeded error. While this is not the expected error, the intention of this setting is to cause client application logic to indicate that not all results are shown.

3.2.9 Views

Views allow applications to see different information in Oracle Virtual Directory. Views are defined by the distinguished names (DN) and IP addresses configured for the View. If an Adapter is enabled for a View, then only the DNs or IP Addresses configured in the View may see data from that Adapter. An Adapter can be enabled for one or more Views. A user that is a member of a View can only see information from Adapters that are enabled to the same View.

To enable an Adapter for a View, in the Views section on the adapter's Routing tab, select the **Enable** option for the appropriate View. If an Adapter is not enabled for a View, it is part of the default View. Any client not assigned to a View may see any Adapter that is part of the default View.

3.2.9.1 Creating and Configuring Views

Perform the following steps to create and configure a View:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Server Views** entry in the tree. The list of existing Views appear.

   To create a new View:

   a. Click the **Add New View** button. The Add New View dialog box appears.

   b. Enter a name for the View in the View Name field and click the **OK** button to create the new View. The new View appears in the list of existing Views.

      Perform the following steps to configure a View.

   To configure a View:

   a. Click the name of the View that you want to configure in the list of existing Views. A screen appears where you can configure the DNs and IP Addresses for the View.

      To add a DN or IP address to the View, click the create button in the appropriate field, enter a value, and click the **Apply** button.

      To delete a DN or IP address from the View, select the value you want to delete and click the **Delete** button.

3.2.10 Include Binds From and Exclude Binds From

The Include Binds From and Exclude Binds From settings allow the administrator to indicate adapters which can share each other's credentials. The Include Binds From and Exclude Binds From settings also help the adapter determine whether the user credentials or the adapter's proxy account should be passed through on an operation. For example, consider different LDAP Adapters proxying two different domain controllers within a Microsoft Active Directory forest. To Oracle Virtual Directory, a user credential from one domain does not appear to be part of another domain. Also, because both domains are from the same forest, we know that the second domain can in fact accept a credential from another domain. The Include Binds From and Exclude Binds From settings allow the administrator to instruct Oracle Virtual Directory on how to handle these situations.

When deciding whether a user credential can be passed through, Oracle Virtual Directory considers the following two conditions:

- whether the supplied credentials are under the current adapter root
- whether the user credentials map under an adapter listed in the Include Binds From field, and also, whether the user credential maps under an excluded adapter listed in the Exclude Binds From field.

Consider the following example with adapter root ou=admin,o=depts,dc=oracle,dc=com. A user credential may either:

1. **Case A**: Map within the namespace of `ou=admin,o=depts,dc=oracle,dc=com`
2. **Case B**: Not map within the namespace of `ou=admin,o=depts,dc=oracle,dc=com` (for example, the credential has DN ends with `ou=sales,o=depts,dc=oracle,dc=com`).

Case A

User credential ends with `ou=admin,o=depts,dc=oracle,dc=com`:

If the Exclude Binds From field is not empty, then the user's credential must be checked to see if they are a child of one of the excluded adapters. If it is, then the Proxy credential must be used (instead of passing through the client's credential). If the user's credential does not belong to an excluded adapter, then the user's credential may be passed through the current adapter.

This scenario most often occurs when two LDAP Adapters are defined where the second adapter is a child of the first or parent adapter. A credential that is part of the child adapter could also erroneously be considered to be part of the parent adapter. Using the Exclude Binds From setting helps correct the problem where the credential from the child adapter would be incorrectly passed through to the parent adapter. Using the Exclude Binds From setting allows Oracle Virtual Directory to understand that certain child DNs do not map to the parent adapter's credential set.

Case B

User credential ends with root different from ou=admin,o=depts, dc=oracle,dc=com:

If the Include Binds From field is not empty, but has adapters defined as shared, the user credential must be checked to see if it maps to one of the shared adapters. If it does, the credential is mapped by the shared adapter and returned to the original adapter. The original adapter is then able to pass through the credential mapped by the shared adapter.

If the credential does not map to the current adapter, or any of the shared adapters, then the proxy credential must be used rather than passing through the provided credential.

An example of this is an Oracle Virtual Directory that proxies multiple Microsoft Active Directory domains. User credentials may have different roots, but since all proxies go to the same forest, it is possible that one domain controller can authenticate a DN from another domain controller. In this situation, credentials from either adapter can be shared in common across both adapters. For example, Domain A adapter proxies Domain A, Domain B adapter proxies Domain B. Domain A and B are in the same forest. Therefore, on both the Domain A and Domain B adapter, you can set the Include Binds From setting to Domain A, Domain B and both adapters are able to pass through each-other's credentials.

4

Understanding Oracle Virtual Directory Plug-Ins

This chapter describes Java plug-ins for Oracle Virtual Directory and includes the following topics:

- What is a Plug-In?
- Understanding the General Purpose Plug-Ins
- Understanding the Enterprise User Security and Oracle Net Services Plug-Ins
- Understanding the Microsoft Active Directory Plug-Ins
- Understanding the Oracle Identity Manager Plug-Ins
- Understanding the Oracle Access Manager Plug-Ins

4.1 What is a Plug-In?

Plug-ins allow you to extend Oracle Virtual Directory with compartmentalized functionality to meet specific business and technical requirements. You can use plug-ins to provide custom logic as part of a transaction or to connect to a custom data source. Oracle Virtual Directory supports two types of plug-ins: Java plug-ins and Python Mapping plug-ins.

> **Note:** This chapter is specific to Java plug-ins for Oracle Virtual Directory, though from an operational perspective, Oracle Virtual Directory treats the two types of plug-ins equally. References to plug-ins in this chapter refer to Java plug-ins. See Chapter 5, "Understanding Oracle Virtual Directory Mapping" for information on mapping plug-ins.

Oracle Virtual Directory includes several plug-ins for various purposes, including plug-ins for analysis, auditing, caching, and dynamic group presentation. You can also deploy plug-ins using different approaches and with full control over the positioning of the plug-in inside the virtual directory, including:

- "Stringing-together" plug-ins to create combined and reusable functionality.
- Deploying plug-ins at the "global" server level so the plug-in affects all requests and responses.
- Deploying plug-ins at the adapter level so the plug-ins manipulate requests and responses for a particular adapter.

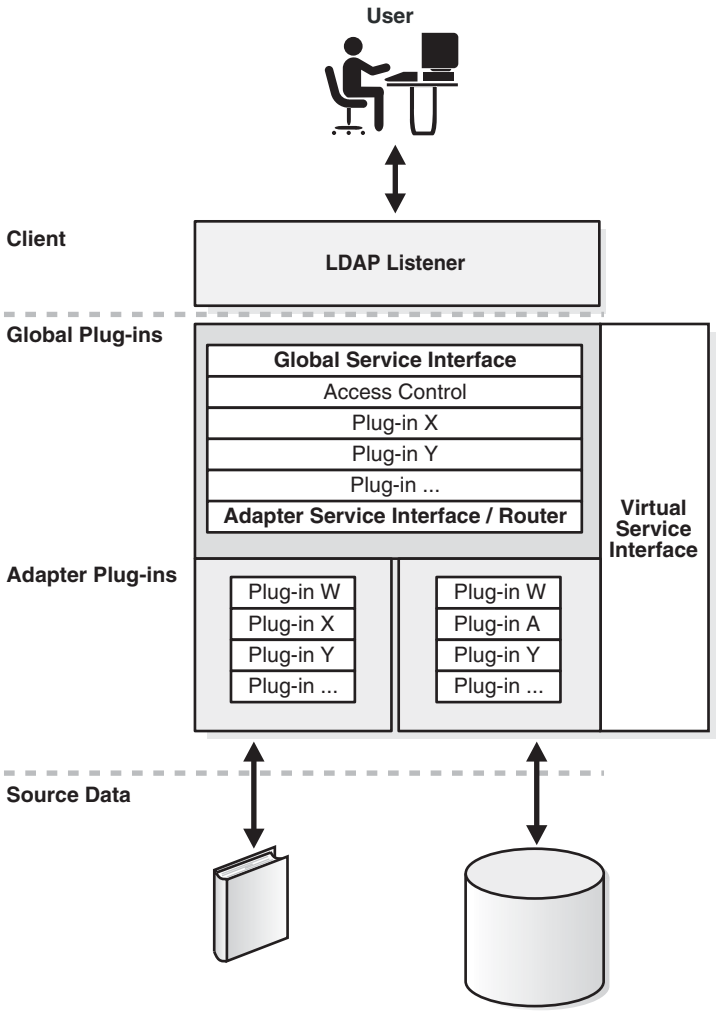
For example, you can deploy the Cache plug-in to run only on a particular Microsoft Active Directory server and only after a mapping has been run to perform translation of the Active Directory user to the standard inetOrgPerson objectclass.

Note:

- To configure plug-ins at the "global" server level, select **Advanced** from the Oracle Directory Services Manager task selection bar and refer to "Managing Global Server Plug-ins" on page 13-4 for the complete steps in the process.
- To configure plug-ins at the adapter level, select **Adapter** from the Oracle Directory Services Manager task selection bar and refer to "Managing Adapter Plug-ins" on page 13-1 for the complete steps in the process.

Figure 4–1 shows the three layer architecture of the Oracle Virtual Directory plug-in framework:

Figure 4–1 Oracle Virtual Directory Plug-In Framework



Plug-ins are arranged to form chains. When Oracle Virtual Directory receives a request, the request passes through Oracle Virtual Directory in the following sequence (as shown in Figure 4–1):

1. The Global Service Interface (GSI), then to
2. The chain of global plug-ins, then to
3. The Adapter Service Interface (ASI), then to
4. The chain of adapter plug-ins.

Global plug-ins are called for all requests. For an inbound request, after all global plug-ins have been executed, the request goes to the ASI, which is the component of Oracle Virtual Directory where routing and adapter handling decisions are made. After the routing system has decided which adapters will be included in a request, the chosen adapter plug-ins are executed. Responses are generated by adapters and then sent back up the chain for further possessing and filtering.

4.1.1 Namespace Filtering

Oracle Virtual Directory provides flexibility in determining whether a plug-in should be executed globally or within the context of a single adapter. In some situations, you may need to further restrict when a plug-in gets used. For example, an adapter is set-up to proxy a Microsoft Active Directory domain and points to DC=VAN,DC=Oracle,DC=com. Under that point in the directory tree, there is a CN=Users container and a CN=Groups container. You can add a namespace filter to any plug-in or mapping to apply them to only one part of the tree.

4.2 Understanding the General Purpose Plug-Ins

This topic describes the general purpose plug-ins included in Oracle Virtual Directory. The general purpose plug-ins extend Oracle Virtual Directory capabilities to perform special functions or mappings. This topic contains the following sections:

- HideEntriesByFilter Plug-In
- ChangeUserRDN Plug-in
- UPNBind Plug-In
- ForkJoin Plug-In
- VirtualMemberof Plug-In
- VirtualAttribute Plug-In
- Dump Transactions Plug-In
- DynamicTree Plug-In
- DynamicEntryTree Plug-In
- FlatTree Plug-In
- DynamicGroups Plug-In
- Cache Plug-In
- ObjectClass Mapper Plug-In
- Sub-Tree Plug-In
- Performance Monitor Plug-In

- UniqueEntry Plug-In
- Adapter Plug-In Version

4.2.1 HideEntriesByFilter Plug-In

The HideEntriesByFilter plug-in allows you to control in fine detail which entries are returned by searches. Using its `hideFilter` configuration parameter, the HideEntriesByFilter plug-in allows you to explicitly control which entries are returned. For example, you are using Oracle Virtual Directory as an address book directory and you want to display only the entries for customer service representatives, which have ou values of CSR. You could use the HideEntriesbyFilter plug-in with hideFilter set to ou=CSR so that when the adapter is searched, only the customer service representatives entries are returned.

4.2.1.1 Configuration Parameters

The HideEntriesByFilter plug-in has the following configuration parameter:

hideFilter
The LDAP filter that controls which entries are restricted for searches. For example, if `hideFilter` is set to **(ou=Sales)**, then searches return only entries that contain ou=Sales. If `hideFilter` is set to **(!(ou=Sales))**, then searches return only entries that do not contain ou=Sales.

4.2.2 ChangeUserRDN Plug-in

The ChangeUserRDN plug-in allows you to rename or replace RDN values from the source directory to Oracle Virtual Directory. This plug-in is useful during LDAP directory migrations.

4.2.2.1 Configuration Parameters

The ChangeUserRDN plug-in has the following configuration parameters:

objectType
Identifies the objectclass type that RDN renaming is performed on. The default setting is person.

replaceValue
True or False: Indicates whether or not the original RDN value (identified by the `fromRDN` parameter) should be replaced by the new RDN value (identified by the `toRDN` parameter). The default setting is true.

fromRDN
Identifies the original RDN attribute name from the source directory that will be replaced or renamed in Oracle Virtual Directory.

toRDN
Identifies the new RDN attribute name that will be used in Oracle Virtual Directory and which replaces the attribute name identified by the fromRDN configuration parameter.

dnAttributes
List of attributes with DNs to perform RDN renaming on. The default list of attributes is member, uniquemember, manager, owner, managedby.

4.2.3 UPNBind Plug-In

The UPNBind plug-in allows you to bind using any configured username attribute, such as cn, SamAccountName, sn, uid, and so on, or username@suffix attributes, such as UserPrincipalName, Mail, and so on. This plug-in is useful for Active Directory-centric applications and can simplify application development by removing the requirement to look-up DNs.

Note: The UPNBind plug-in is supported only for deployment as a global plug-in—do not deploy the UPNBind plug-in on adapters.

4.2.3.1 Configuration Parameters

The UPNBind plug-in has the following configuration parameters:

Note: At a minimum, either the NameAttributes or NameAndSuffixAttributes configuration parameter must be set.

NameAttributes

Identifies a list of naming attributes to use for authenticating users. For example: cn, sn, uid, SAMAccountName.

NameAndSuffixAttributes

A list of attributes in the form of name@suffix, such as UPN or mail, to use for authenticating users.

BindOption

Determines how the UPNBind plug-in should process resulting entries when multiple users have same values for the NameAttributes and NameAndSuffixAttributes parameters. The following is a list and description of the supported values for the BindOption option. The default setting is QuickFail.

- QuickFail: When multiple users have same values for configured attributes, the UPNBind plug-in displays a Found more than one user entry error and the LDAP bind fails.
- FirstUser: The UPNBind plug-in binds as the first user and returns the result of that bind.
- FirstSuccess: The UPNBind plug-in binds as the first user. If that bind fails, the UPNBind plug-in proceeds through the list of all users until it finds a successful bind.

AdapterName

Identifies the adapter to use for authentication. If the user entry is not present in the specified adapter, the LDAP bind fails with an Invalid Credentials message. This is an optional parameter. If you do not specify this parameter, the plug-in considers all the available adapters for authenticating the user. This parameter is useful for improving the plug-in's performance when you know which adapter is used for authentication.

4.2.4 ForkJoin Plug-In

Supported only for Join View Adapters, the ForkJoin plug-in allows you to search against the primary adapter and/or secondary adapters in a Join View. During LDAP search, when a search filter contains one or more attributes that are available only in secondary adapter, without this plug-in, Oracle Virtual Directory cannot return Joined

entries that satisfies the filter, as the entire search filter is sent only to the primary adapter. Using the ForkJoin Plug-in, Oracle Virtual Directory can search on attributes only in the primary adapter, only in the secondary adapter, and in both the primary and secondary adapters.

For example, user data resides in multiple identity sources, with samaccountname, sn, givenname, employeenumber in Active Directory. However, the title attribute is stored in a Human Resources database. If Active Directory is configured as the primary adapter; Human Resources database as the secondary adapter; and if the ForkJoin plug-in is deployed, when an LDAP enabled application queries the user data based on samaccountname and/or title, Oracle Virtual Directory returns the entry that satisfies the filter and includes both Active Directory and the Human Resources database data.

4.2.4.1 Configuration Parameters

The ForkJoin plug-in has the following configuration parameters:

> **Note:** At a minimum, you must set either the SecondaryOnlyAttributes or PrimaryAndSecondaryAttributes configuration parameter to deploy the ForkJoin plug-in.

SecondaryOnlyAttributes
A list of attributes that are present only in the secondary adapter and which the application can use in the search filter. The attributes identified by the SecondaryOnlyAttributes configuration parameter *cannot* also be identified by the PrimaryAndSecondaryAttributes configuration parameter.

PrimaryAndSecondaryAttributes
A list of attributes that are present in both the primary and secondary adapters and which the application can use in the search filter. The attributes identified by the PrimaryAndSecondaryAttributes configuration parameter *cannot* also be identified by the SecondaryOnlyAttributes configuration parameter.

JoinPolicy
Supports the following settings:

- Standard Join: Returns all entries that satisfy the search filter in the primary adapter after joining the corresponding entries in secondary adapters.
- Left Outer Join: Returns all the entries in the primary adapter after joining with corresponding entries in secondary adapter (Standard Join), and returns entries from the secondary adapter that satisfy the join condition and which have a corresponding match in primary adapter. This is the equivalent to Left Outer Join in database terminology.
- Full Outer Join: Returns all the entries in the primary adapter after joining with corresponding entries in secondary adapter (Standard Join); returns entries from the secondary adapter that satisfy the join condition and which have a corresponding match in primary adapter (Left Outer Join); and returns entries from the secondary adapter that satisfy the search filter but do not have a matching entry in primary adapter. This is the equivalent to Left Outer Join + Right Outer Join in database terminology.

Note: For conditional joins when Full Outer Join is configured, the entries in secondary adapter are mapped to entries in primary adapter without considering the filter in join condition.

4.2.4.2 Example ForkJoin Plug-In Deployment

This section provides an example ForkJoin plug-in deployment. The following data resides in the primary adapter and secondary adapter. The primary adapter namespace is dc=primary and the secondary adapter namespace is dc=secondary.

Data in Primary Adapter	Data in Secondary Adapter
dn: cn=Rock, dc=primary objectclass: inetorgperson cn: Rock sn: Anne givenname: Anne rock telephonenumber: 54300	dn: cn=Rock, dc=secondary objectclass: inetorgperson cn: Rock sn: Anne title: Manager
dn: cn=Sandy, dc=primary objectclass: inetorgperson cn: Sandy sn: Ketty manager: cn=Rock, dc=primary telephonenumber: 54301	dn: cn=Sandy, dc=secondary objectclass: inetorgperson cn: Sandy sn: Ketty title: SMTS
dn: cn=Rivry,dc=primary objectclass: inetorgperson cn: Rivry sn: Rod title: Trainee manager: cn=Rock, dc=secondary telephonenumber: 54303 description: Trainee for dept 543 departmentNumber: 543	dn: cn=Rivry,dc=secondary objectclass: inetorgperson cn: Rivry sn: Rod title: Trainee
dn: cn=Woods,dc=primary objectclass: inetorgperson cn: Woods sn: Tent description: User with no title	dn: cn=Mounty,dc=secondary objectclass: inetorgperson cn: Mounty sn: Ret title: MTS - dept_sec

The Join Condition is `cn=cn` and the SecondaryOnlyAttributes parameter is set to title. An ldapsearch with a subtree scope and filter of `(|(title=*e*)(sn=*e*))` is performed against the Join View Adapter.

The following are the search results for each type of JoinPolicy:

Standard Join:

```
cn=Rock,dc=Join
sn=Anne
cn=Rock
title=Manager

cn=Sandy,dc=Join
```

```
sn=Ketty
cn=Sandy
title=SMTS

cn=Woods,dc=Join
sn=Tent
cn=Woods
```

Left Outer Join:

```
cn=Rock,dc=Join
sn=Anne
cn=Rock
title=Manager

cn=Sandy,dc=Join
sn=Ketty
cn=Sandy
title=SMTS

cn=Woods,dc=Join
sn=Tent
cn=Woods

cn=Rivry,dc=Join
sn=Rod
cn=Rivry
title=Trainee
```

Full Outer Join:

```
cn=Rock,dc=Join
sn=Anne
cn=Rock
title=Manager

cn=Sandy,dc=Join
sn=Ketty
cn=Sandy
title=SMTS

cn=Woods,dc=Join
sn=Tent
cn=Woods

cn=Rivry,dc=Join
sn=Rod
cn=Rivry
title=Trainee

cn=Mounty,dc=Join
sn=Ret
cn=Mounty
title=MTS - dept_sec
```

4.2.5 VirtualMemberof Plug-In

The VirtualMemberof plug-in adds the memberof attribute to person entries. The values of this memberof attribute are the DNs of any groups the person entry belongs

to. The VirtualMemberof plug-in is useful when applications want to see group membership, but do not want to perform a secondary search for the groups.

4.2.5.1 Configuration Parameters

The VirtualMemberof plug-in has the following configuration parameters:

searchBase
The DN of the base to search for groups containing person entries.

explicitrequestonly
True or False: If set to True, the memberof attribute will be added to the entry *only* if it is explicitly requested as a returned attribute. The default value is True.

adapterName
List of adapters containing group entries. This parameter helps minimize the scope of the search for groups.

4.2.6 VirtualAttribute Plug-In

The VirtualAttribute plug-in allows you to reuse attribute data in multiple attributes without deploying a mapping. For example, say a new application requires a person's full name to be used in their display, but instead of looking for cn, the application wants to use the displayname attribute—which does not exist in the adapter data. Using the VirtualAttribute plug-in, you can map displayname to the value of cn and have the cn attribute remain available for other applications that are looking for it. If the displayname attribute already exists in the adapter data, you can specify whether to add the values of cn to the existing values of displayname or to replace the values of displayname with the values of cn.

In addition to adding and replacing attribute values, the VirtualAttribute plug-in can also perform the functionality of ConditionalPublish mapping. That is, using the RemoveAttributes configuration parameter, you can identify attributes to virtually remove from entries before they are returned to the client.

Note: Constant values can be specified for the attributes that are added or replaced by the VirtualAttribute plug-in.

4.2.6.1 Configuration Parameters

The VirtualAttribute plug-in has the following configuration parameters:

Note: At a minimum, you must set either the AddAttribute or ReplaceAttribute configuration parameter.

ContainerDN
An optional parameter, ContainerDN identifies the containers the VirtualAttribute plug-in applies to. The VirtualAttribute plug-in is applicable for all entries under the containerDN which match the filter specified by the matchFilter parameter. This parameter can be repeated to specify multiple containerDNs. This parameter also restricts the VirtualAttribute plug-in to a certain branch within data exposed by the adapter. The default setting is "", or DSE—that is, if you do not set the ContainerDN parameter, the VirtualAttribute plug-in searches the entire directory tree.

MatchFilter
An optional parameter, MatchFilter identifies the entries the VirtualAttribute plug-in applies to. The VirtualAttribute plug-in is applicable for all entries under containerDN that match the filter specified by the MatchFilter parameter. If you do not set the MatchFilter parameter, the VirtualAttribute plug-in is applicable for all the entries under the DNs identified by ContainerDN parameter.

AddAttribute
A list of attributes to add to the entry and a value to assign to these virtual attributes. If an attribute identified by the AddAttribute parameter already exists in the entry, the VirtualAttribute plug-in appends the specified values to the existing set of values. The value can be either a constant or another attribute value.

You can set the AddAttribute parameter multiple times for the VirtualAttribute plug-in. The value you set for the AddAttribute parameter must be mutually exclusive, that is, different, from the ReplaceAttribute.

For example:

- To add the values of the cn attribute to uid and displayName, set the value of the AddAttribute parameter to: `uid=displayName=%cn%`
- To add the constant value Acme to the companyName attribute, set the value for AddAttribute parameter to: `companyName=Acme`

ReplaceAttribute
A list of attributes to add to the entry and a value to assign to these virtual attributes. If an attribute identified by the ReplaceAttribute parameter already exists in the entry, the VirtualAttribute plug-in replaces them using the value you supply. The value can be either a constant or another attribute value.

You can set the ReplaceAttribute parameter multiple times for the VirtualAttribute plug-in. The value you set for the ReplaceAttribute parameter must be mutually exclusive, that is, different, from the AddAttribute.

For example:

- To replace the values of uid and displayName with the values of the cn attribute, the ReplaceAttribute parameter can be configured as: `uid=displayName=%cn%`
- To replace the value of the companyName attribute with the constant value Acme, the ReplaceAttribute parameter can be configured as: `companyName=Acme`

RemoveAttributes
Comma-separated list of attributes to virtually remove from entries that satisfy the MatchFilter under the specified ContainerDN.

4.2.6.2 Example VirtualAttribute Plug-In Deployment

This section provides an example VirtualAttribute plug-in deployment. Assume the following VirtualAttribute plug-in configuration parameters are set:

Configuration Parameter	Value
AddAttribute	uid=displayName=%cn%
AddAttribute	certificate=Verisign
ReplaceAttribute	title=%designation%
RemoveAttributes	designation,uid

Assume the original entry to be processed is:

```
dn: cn=john, dc=com
cn: john
cn: jsmith
uid: 1234
certificate: selfsigned
designation: SMTS
title: Senior Software Engineer
sn: smith
objectclass: person
objectclass: top
```

The following shows how the VirtualAttribute plug-in transforms the original entry:

After processing the AddAttributes configuration parameter, the entry is:

```
dn: cn=john, dc=com
cn: john
cn: jsmith
uid: 1234
uid: john
uid: jsmith
displayName: john
displayName: jsmith
certificate: verisign
certificate: selfsigned
designation: SMTS
title: Senior Software Engineer
sn: smith
objectclass: person
objectclass: top
```

After processing the ReplaceAttributes configuration parameter, the entry is:

```
dn: cn=john, dc=com
cn: john
cn: jsmith
uid: 1234
uid: john
uid: jsmith
displayName: john
displayName: jsmith
certificate: verisign
certificate: selfsigned
designation: SMTS
title: SMTS
sn: smith
objectclass: person
objectclass: top
```

After processing the RemoveAttributes configuration parameter, the entry is:

```
dn: cn=john, dc=com
cn: john
cn: jsmith
displayName: john
displayName: jsmith
certificate: verisign
certificate: selfsigned
title: SMTS
sn: smith
```

```
objectclass: person
objectclass: top
```

4.2.7 Dump Transactions Plug-In

The Dump Transactions plug-in generates a record of all transactions for each LDAP operation and logs the record to the Oracle Virtual Directory console log. You can configure the Dump Transactions plug-in to run on any log level. The Dump Transactions plug-in is particularly useful for diagnosing mapping and integration efforts while logic flows through the Oracle Virtual Directory system. You can use the Dump Transaction plug-in to analyze issues on a specific adapter without setting the entire server log level to a more verbose level. Think of the Dump Transactions plug-in as a protocol analyzer for Oracle Virtual Directory.

4.2.7.1 Configuration Parameters

The Dump Transactions plug-in has the following configuration parameter:

loglevel
The log level the plug-in will log transactions at. Supported values are: SEVERE, WARNING, INFO, FINE, FINER, and FINEST. There is no default value.

4.2.8 DynamicTree Plug-In

The DynamicTree plug-in allows you to construct DNs in Oracle Virtual Directory using the attribute values of source directory entries that have parent entries which are real, virtual, or pointers to different real entries.

Note: The DynamicTree plug-in is supported only for deployment on adapters—do not deploy the DynamicTree plug-in as a global plug-in.

This plug-in is useful for generating organization charts and reports in LDAP hierarchy format based on structural data contained in the source directory entry. The Dynamic Tree plug-in provides more flexibility than the DynamicEntry Tree plug-in because it allows you to browse the directory tree.

Consider the following figures and examples.

Note: In the following graphics, attribute values in the source directory appear in **bold** type.

Figure 4–2 shows an example directory structure residing in the source directory:

Figure 4–2 Example Directory Structure in Source Directory

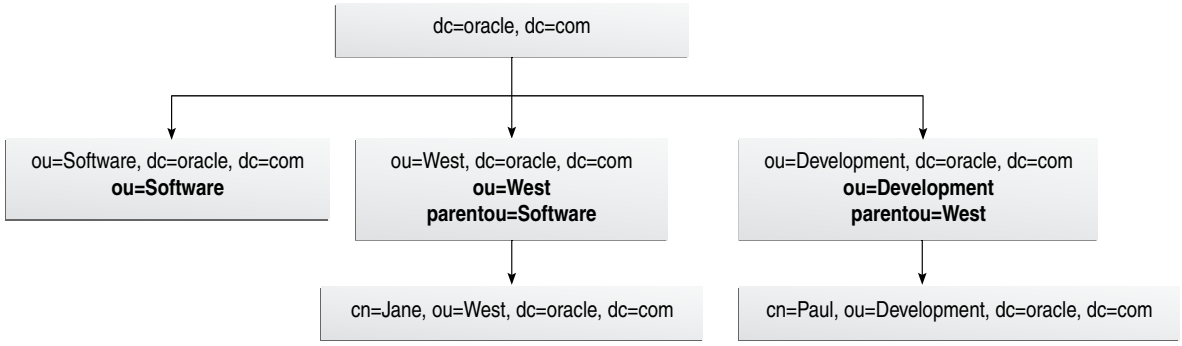


Figure 4–3 shows an example of how the DynamicTree plug-in, with the parentEntryType parameter set to 1 and the attributeName parameter set to parentou, transforms the source directory's flat hierarchy shown in Figure 4–2 into a layered hierarchy in Oracle Virtual Directory.

Figure 4–3 Directory Structure in OVD Using DynamicTree Plug-In

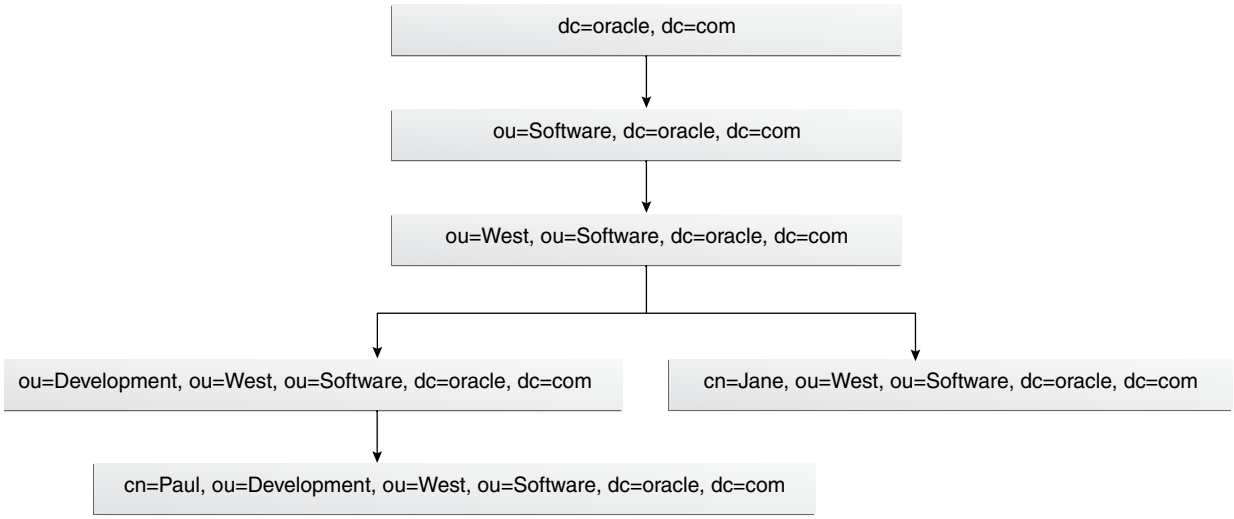


Figure 4–4 shows another example directory structure residing in the source directory:

Figure 4–4 Example Directory Structure in Source Directory

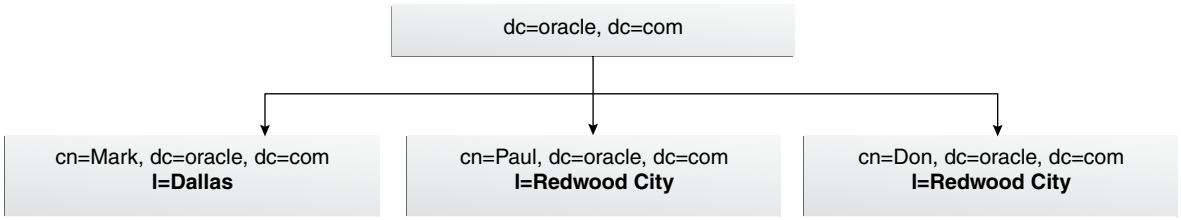
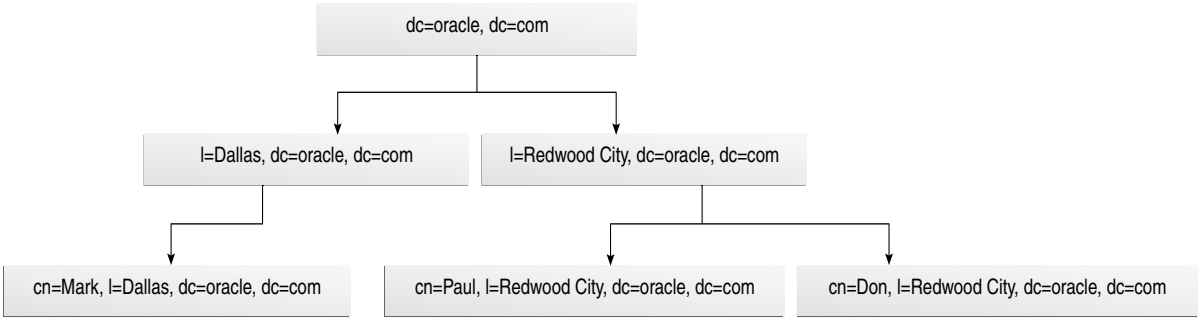


Figure 4–5 shows an example of how the DynamicTree plug-in, with the parentEntryType parameter set to 0 and the attributeName parameter set to l, creates a layered hierarchy in Oracle Virtual Directory based on the l attribute value in the source directory shown in Figure 4–4.

Figure 4–5 Directory Structure in OVD Using DynamicTree Plug-In



4.2.8.1 Configuration Parameters

The DynamicTree plug-in has the following configuration parameter:

attributeName
The attribute used to identify the entry's parent entry.

parentEntryType
Identifies the type of parent entry. Supported values are 0 and 1. Select 0 for virtual parent entries, which are entries that Oracle Virtual Directory creates based on attribute values in the source directory. Select 1 for parent entries created by multiple ou containers.

4.2.9 DynamicEntryTree Plug-In

The DynamicEntryTree is a general purpose plug-in that can be used to generate virtual directory tree hierarchy by using attributes found in entry leaf nodes. For example, if an adapter has a root of o=Airius.com and users are present as uid=scarter,ou=people,o=airius.com, the DynamicEntryTree plug-in can insert new hierarchy using data present in the entry of the user, such as: uid=scarter,ou=accounting,ou=people,o=airius.com.

4.2.9.1 Configuration Parameters

The DynamicEntryTree plug-in has the following configuration parameter:

patterns
The DynamicEntryTree plug-in is configured by specifying matching patterns using the patterns configuration parameter. The syntax for the patterns parameter value is

```
attr[=entryattr[(SUPPRESS|value)]],...
```

For each DN component, an attribute name can have a value substituted with the attribute on the right side of the equals sign (=). If no substitution is required, then just the attribute is listed, which essentially refers to a matching component with the original DN. When a value is substituted, you can also further qualify what happens when entryattr has no value. You may either specify SUPPRESS to suppress the entry

completely, or you may specify a default value within brackets after the entryattr name.

Multiple patterns can be defined by separating them with a |. For example:

```
uid,ou=department(contract),ou|cn,ou=code(SUPPRESS)
```

This rule has two patterns. One pattern matches objects under the adapter root (o=airius.com) that have dn components matching uid,ou. Therefore, uid=scarter,ou=people,o=airius.com would be selected for mapping. Upon return, the department attribute would be checked for values. If none are present, the static text `contract` is substituted. On a search, if base is set to uid=scarter,ou=accounting,ou=people,o=airius.com, the search will be modified so that the new base is uid=scarter,ou=people,o=airius.com and the filter will be modified with an additional *anded* term of ou=accounting.

The second pattern is intended to match child objects of o=airius.com that have a cn rdn component. When matched, if the entry has an attribute code, its value is substituted by creating cn=mygroup,ou=code12,o=airius.com. If the attribute `code` is not present, the entry result will be suppressed due to the SUPPRESS keyword.

4.2.10 FlatTree Plug-In

The FlatTree plug-in, as with the DynamicEntryTree, performs dynamic mapping of the virtual directory tree. The FlatTree plug-in compresses a directory source so that all entries appear directly under the root of the adapter.

FlatTree plug-in operates in two deployed modes:

- as part of any existing adapter to flatten the existing namespace
- as part of a Custom Adapter deployment

As part of a Custom Adapter deployment, you can use the FlatTree plug-in's adapter parameter to retrieve data from the designated adapter so that the data appears as part of the namespace of the Custom Adapter. When deployed this way, the adapter root object is *not* defined. This type of deployment can be useful if you want to overlay multiple adapters on top of a parent adapter without creating duplicate parent nodes.

4.2.10.1 Configuration Parameters

The following is a list and description of the FlatTree plug-in configuration parameters:

criteria
Criteria defines an LDAP filter that restricts the entries that can be searched for through the FlatTree plug-in. For example, if criteria was set to objectclass=user, then only user objects would be returned through the FlatTree plug-in.

adapter
If the adapter parameter is not defined, the FlatTree plug-in assumes data will be retrieved through its parent adapter. When defined, the adapter parameter must be the name of another adapter in the Oracle Virtual Directory configuration and the FlatTree plug-in retrieves data from this adapter and maps the entries to its parent adapter's root. If the adapter parameter is defined the root object is not returned—only the child entries are returned.

4.2.11 DynamicGroups Plug-In

The DynamicGroups plug-in enables Oracle Virtual Directory to process LDAP objectclasses that are both `groupofuniquenames` and `groupofurls` (referred to as a "dynamic group") and convert it into a virtual static group, or groupofuniquenames equivalent. The plug-in works by monitoring returned LDAP objects and detects objects where the memberurl attribute is present and the objectclass is both groupofuniquenames and groupofurls.

When detected, the plug-in automatically processes any memberurl values and adds the results to the uniquemember attribute. This dynamic object processing allows administrators to define groups that hold both static members and dynamic members while maintaining compatibility with applications that may not normally support the groupofurls objectclass.

Example 4–1 shows an example query when the Dynamic Groups plug-in is not enabled. Two groups are returned and the first group holds two static members and has a memberurl defining a particular directory subtree to also be members.

Example 4–1 Example Query When Dynamic Groups Plug-in Not Enabled

```
C:\>ldapsearch -D bindDN -q -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)(obj
ectclass=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
```

Example 4–2 shows the results of the same query as in Example 4–1, however, the Dynamic Groups plug-in is enabled:

Example 4–2 Example Query When Dynamic Groups Plug-in is Enabled

```
C:\>ldapsearch -D bindDN -q -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)(obj
ectclass=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
```

4.2.11.1 Testing Group Membership

Products such as policy servers often need to check whether a particular person is a member of a group, which can be a significant task if a group is very large or if the group is a dynamic group. However, the Dynamic Groups plug-in detects a membership test query by detecting the presence of both cn and uniquemember filter terms. When present, the Dynamic Groups plug-in processes the query differently by recognizing that the client wants to test a membership assertion. The plug-in modifies the results and returns only the single user being tested as the member, as shown in Example 4–3:

Example 4–3 Example Membership Test with the Dynamic Groups Plug-in

```
C:\>ldapsearch -D bindDN -q -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
```

4.2.11.2 Configuration Parameters

The Dynamic Groups plug-in has no configuration parameters. To enable the Dynamic Groups plug-in, add it to a plug-in chain.

4.2.12 Cache Plug-In

The Cache plug-in provides a true cache for Oracle Virtual Directory—it is an in-memory copy of a LDAP search result set and is designed to improve the performance of repeated queries. For example, imagine a 3rd party portal server as part of a daily synchronization process that queries for all of the groups in the enterprise LDAP server (that is, Oracle Virtual Directory) three consecutive times. The cache plug-in can keep a copy of the result of the first search for re-use by LDAP clients. While it did result in a slightly faster result to the client application—more importantly it reduced the load of the enterprise source systems.

Oracle Virtual Directory is designed to function without the Cache plug-in and in many cases it is not necessary because many applications either cache results on their own, or because most enterprise source systems already cache data. Also, because Oracle Virtual Directory is often used in secure environments, keeping a cache exposes security elements, such as if an employee is terminated and their user result is still in the cache.

> **Note:** If you use the Cache plug-in, the cache timeout should be kept small to avoid cache update problems in cases such as these.

The Cache plug-in can greatly improve performance for applications where queries are highly repetitive and can be configured to run any where within Oracle Virtual

Directory. The Cache plug-in can be executed globally or within the context of a single adapter, and can also be restricted to specific namespaces using namespace filtering.

If a query is repeated by the same user and the same attributes or a subset of attributes are requested, the cache will return its results instead Oracle Virtual Directory retrieving the information from the source. For example, consider the following two theoretical queries where the client changes the search attributes in Query 2:

Query 1

- **Bind as**: `"cn=Example User,cn=users,dc=mycompany,dc=com"`
- **Base of**: `"dc=mycompany,dc=com"`
- **Query of**: `"uid=jdoe"`
- **Attributes**: `"cn,mail"`

Query 2

- **Bind as**: `"cn=Example User,cn=users,dc=mycompany,dc=com"`
- **Base of**: `"dc=mycompany,dc=com"`
- **Query of**: `"uid=jdoe"`
- **Attributes**: `"telephonenumber,manager"`

By using the Cache plug-in, Oracle Virtual Directory does not treat Query 1 and Query 2 as two separate queries and does not search the back-end source twice. Oracle Virtual Directory requests all the attributes for matching entries in Query 1, but only returns the attributes requested by the client and keeps an in-memory copy of the search results in its cache.

Note: Oracle Virtual Directory retrieves only the attributes which the bound user has access to based on the back-end source repository Access Control Lists (ACLs), Oracle Virtual Directory ACLs, and adapter routing rules.

The Cache plug-in can also be configured to allow cache hits to be shared between users. Sharing cache entries between users should not be used unless `passcredentials` is not being passed to back-end sources and Oracle Virtual Directory is solely responsible for security enforcement.

Note: Careful planning should take place when sharing cache hits between users as it may then be possible for one user to see something they should not, since they may have access to a cache result from a more privileged user.

The Cache plug-in periodically reviews the cache and checks for any expired results or entries that have been invalidated by a previous modify transaction. If the cache quota is exceeded, the Cache plug-in attempts to trim memory by purging the queries that were least recently used.

4.2.12.1 Configuration Parameters

The following is a list and description of the Cache plug-in configuration parameters:

storeallattrs
Controls whether or not the Cache plug-in silently requests all attributes. Supported values are 0 (disable) or 1 (enable). By default, the storeallattrs parameter is disabled (0).

bysubject
Indicates whether or not cache results are shared between subjects. Supported values are 0 or 1. A value of 0 indicates results *will be* shared between subjects and a value of 1 indicates that results *will not be* shared between subjects. The default value is 1.

maintenanceinterval
The amount of time (in seconds) between when the cache manager checks for expired queries. The default value is 60.

size
The maximum number of entries that may be cached at any one time. The default value is 10000.

maximumage
The maximum age (or time), in seconds, that any query/entry can be stored in the cache. The default value is 600.

maxresultsize
The maximum number of entries that may be cached for any particular query. The default setting is 1000.

trimsize
When the maximum cache size is exceeded, this parameter indicates the amount by which the cache manager must reduce the balance. The default value is 10000.

Note: When necessary, trimming is done by purging expired queries first followed by queries in order of least recent use.

---

zeroresults
Indicates whether or not to cache query results containing zero entries. Supported values are 0 and 1.

4.2.13 ObjectClass Mapper Plug-In

The ObjectClass Mapper plug-in is a general mapping plug-in that can make one objectClass, such as user, appear like another objectClass, such as inetOrgPerson. This ability is useful when an application is expecting a particular objectClass and attributes but the directory does not support that objectClass or attributes.

The ObjectClass Mapper plug-in can perform several different types of manipulation based on configuration parameters you select, including:

- attribute mapping
- objectclass mapping
- adding attributes conditional on objectclass
- removing attributes
- filtering auxiliary classes
- handling activation and deactivation

Where attribute mapping relationships occur, the prefix `client-` indicates client side and `source-` indicates data source side. For example, mapping Active Directory server `user` to represent it as `InetOrgPerson` would imply that Active Directory is the source side and `InetOrgPerson` is the client side.

4.2.13.1 Configuration Parameters

The following is a list and description of the ObjectClass Mapper plug-in configuration parameters:

directoryType
The directory type to use when performing user activation. Supported values are SunOne, eDirectory, ADAM, and ActiveDirectory. For example:

Parameter Name: directoryType

Parameter Value: ActiveDirectory

activationAttribute
Use the activationAttribute parameter when an application has no knowledge of the underlying directory's user activation system. The activationAttribute parameter informs Oracle Virtual Directory which incoming attribute contains the user activation flag, which is then mapped to a directory specific attribute and flag. For example:

Parameter Name: activationAttribute

Parameter Value: myuseraccountcontrol

deactivationValue
Comma separated list of attribute values specified in `activationAttribute` that indicate this user should be marked as inactive.

activationValue
Comma separated list of attribute values specified in `activationAttribute` that indicate this user should be marked as active.

mapObjectClass
An objectClass to be mapped in the form of *`client-ObjectClass=source-ObejctClass`*. For example:

Parameter Name: mapObjectClass

Parameter Value: inetOrgPerson=user

You can use the mapObjectClass parameter multiple times for multiple mappings.

addAttribute[*-objectclassvalue*]
Adds attributes for a user during the add process. An optional objectclass value may be added to the configuration name to add the attribute only for certain objectclasses. For example, to add a userAccountControl attribute to only the user objectclass, use:

Parameter Name: addAttribute-user

Parameter Value: userAccountControl=546

Note: An attribute value may be referenced on the value side of the expression by supplying the attribute name surrounded by the percentage character (`%`). For example:

Parameter Name: addAttribute-group

Parameter Value: samaccountname=%cn%

filterAttribute[*-objectclassvalue*]
Comma-separated list of attributes that will be removed during the add operation and from all returned entries. A conditional objectclass value may be added to the name of the parameter to filter out attributes for a specific objectclass. For example:

Parameter Name: filterAttribute

Parameter Value: objectsid,memberof,samaccountname

mapAttribute
An attribute to be mapped in the form of `client-Attribute=source-attribute`. For example:

Parameter Name: mapAttribute

Parameter Value: uniqueMember=member

You can use the mapAttribute parameter multiple times for multiple mappings.

filterAuxiliaryClass
Comma separated list of objectclasses that must be removed on an add operation. An example is Microsoft Active Directory for Windows 2000 does not allow auxiliary object classes to be listed while adding an entry, while Microsoft Active Directory and ADAM for Windows Server 2003 does allow for auxiliary classes to be listed. For example:

Parameter Name: filterAuxiliaryClass

Parameter Value: person,myorgPerson

filterObjectClassOnModify
Comma-separated list of attributes that will be removed during the modify operation for a specific objectclass. For example:

Parameter Name: filterObjectClassOnModify

Parameter Value: objectsid,memberof,samaccountname

4.2.14 Sub-Tree Plug-In

The Sub-Tree plug-in was originally developed to support early versions of IBM's Tivoli Access Manager product, which had a requirement where it stored policy information about a person's entry *under* the person's entry, thereby changing the person entry from its typical leaf model to be a branch. Some directories did not support such a model or did not want to populate their enterprise directory this way.

The Sub-Tree plug-in allows you to store these sub-tree entries in a different adapter while presenting the expected directory tree view to the application—in this case Tivoli Access Manager. The requirement to add entries under a normal leaf entry is uncommon. The more common case is to make a data-store appear as a branch under an existing data-store and any adapter can perform this by properly setting its root namespace value.

4.2.14.1 Configuration Parameters

The following is a list and description of the Sub-Tree plug-in configuration parameters:

storeadapter
The adapter to store the user subtree objects in.

storeroot
The location in the store adapter where you want to store the user subtree objects.

subtreematch
Identifies the subtree distinguished name (DN) component that the Sub-Tree plug-in should intercept and redirect to the store adapter. The default value is secAuthority=Default.

matchdn
Numbered parameters that specify distinguished names (DN) under which user objects are found and the user object RDN. For example:
`0=ou=People,o=Airius.com\:uid`

4.2.15 Performance Monitor Plug-In

The Performance Monitor plug-in allows you to monitor the performance of a specific adapter. To use the Performance Monitor plug-in, attach it to an adapter and then perform operations against that adapter. To view the adapter performance, you must perform a specific type of base level LDAP search on the adapter's root namespace with a filter of `vdeSearchtime=*`.

The search will return results similar to Example 4–4, where all time measurements are in milliseconds:

Example 4–4 Example of Data Returned with the Performance Monitor Plug-In

```
dn: dc=demo,dc=com
vdeNumSearches: 4
vdeNumEntries: 5
vdeMinSearchTime: 0
vdeMaxSearchTime: 16
vdeTotalSearchTime: 16
vdeAverageSearchTime: 4
vdeMinEntryTime: 0
vdeMaxEntryTime: 0
vdeTotalEntryTime: 0
vdeAverageEntryTime: 0
vdeMinSearchCompleteTime: 0
vdeMaxSearchCompleteTime: 203
vdeTotalSearchCompletionTime: 219
vdeAverageSearchCompletionTime: 54
```

Note: The Performance Monitor data is reset after the Oracle Virtual Directory server restarts.

4.2.15.1 Configuration Parameters

The Performance Monitor plug-in has no configuration parameters. To enable the Performance Monitor plug-in, add it to a plug-in chain.

4.2.16 UniqueEntry Plug-In

In some Oracle Virtual Directory environments users have duplicate accounts for more than one service they connect to using Oracle Virtual Directory adapters. Typically, building a Join View Adapter to unify multiple adapters and create a single virtual user entry would resolve this problem. However, there are circumstances where building a Join View Adapter is not an option. For example, there might be a directory for staff personnel and a directory for customers, but certain staff members have accounts in both directories for legitimate business purposes and you cannot create a Join View Adapter.

The UniqueEntry plug-in solves this problem by enabling you to rank various adapters by their authoritative source. For example, if you are attempting to determine which of the duplicate user identities is a staff account, then the staff directory is more relevant than the customer directory and you would place a higher priority on the staff directory adapter than the customer directory adapter.

When ranking adapter priorities, the lower the value of the numerical ranking is, the higher priority. For example, if you are searching two adapters and one adapter has a priority ranking of five and the other adapter has a priority ranking of ten, the adapter with the priority ranking of five is searched first, before the adapter with the priority ranking of ten.

Note: Do not apply the UniqueEntry plug-in at the adapter level—it should always be applied only as a Global plug-in.

4.2.16.1 Configuration Parameters

The UniqueEntry plug-in has the following configuration parameter:

uniqueattribute
The attribute to use as the unique key.

4.2.17 Adapter Plug-In Version

Do not deploy the Adapter Plug-in Version plug-in—it is for information only and has no server functionality. It provides the adapter version information that appears on the Oracle Directory Services Manager home page.

4.3 Understanding the Enterprise User Security and Oracle Net Services Plug-Ins

Oracle Virtual Directory includes plug-ins to simplify the integration with Enterprise User Security (EUS) and Oracle Net Services. This topic describes the plug-ins related to these integrations and contains the following sections:

- EUSActiveDirectory Plug-In
- EUSiPlanet Plug-In
- EUSOID Plug-In
- EUSeDirectory Plug-In
- EUSMemberDNMapping Plug-In
- EUSLockout Plug-In
- ONames Plug-In

- SubschemaSubentry Plug-In

See Also:

- Integrating with Oracle's Enterprise User Security on page 19-3
- Integrating with Oracle's Net Services on page 19-28

4.3.1 EUSActiveDirectory Plug-In

Use the EUSActiveDirectory plug-in only when integrating Oracle Virtual Directory with Oracle's Enterprise User Security database product and your user identities are stored in Microsoft Active Directory. The EUSActiveDirectory plug-in translates Active Directory attributes to a format that can be used by the Enterprise User Security database.

4.3.1.1 Configuration Parameters

The EUSActiveDirectory plug-in has the following configuration parameter:

ADLockoutDuration
The amount of time (in minutes) before the user account for Enterprise User Security authentication expires.

4.3.2 EUSiPlanet Plug-In

Use the EUSiPlanet plug-in only when integrating Oracle Virtual Directory with Oracle's Enterprise User Security database product and your user identities are stored in Sun Java System Directory Server. The EUSiPlanet plug-in translates Sun Java System Directory Server attributes to a format that can be used by the Enterprise User Security database.

4.3.2.1 Configuration Parameters

The EUSiPlanet plug-in has no configuration parameters. To enable the EUSiPlanet plug-in, add it to a plug-in chain.

4.3.3 EUSOID Plug-In

Use the EUSOID plug-in only when integrating Oracle Virtual Directory with Oracle's Enterprise User Security database product and your user identities are stored in Oracle Internet Directory. The EUSOID plug-in translates Oracle Internet Directory attributes to a format that can be used by the Enterprise User Security database.

4.3.3.1 Configuration Parameters

The EUSOID plug-in has no configuration parameters. To enable the EUSOID plug-in, add it to a plug-in chain.

4.3.4 EUSeDirectory Plug-In

Use the EUSeDirectory plug-in only when integrating Oracle Virtual Directory with Oracle's Enterprise User Security database product and your user identities are stored in Novell eDirectory. The EUSeDirectory plug-in translates Novell eDirectory attributes to a format that can be used by the Enterprise User Security database.

4.3.4.1 Configuration Parameters

The EUSeDirectory plug-in has no configuration parameters. To enable the EUSeDirectory plug-in, add it to a plug-in chain.

4.3.5 EUSMemberDNMapping Plug-In

Use the EUSMemberDNMapping plug-in only when integrating Oracle Virtual Directory with Oracle's Enterprise User Security database product. The EUSMemberDNMapping plug-in translates the distinguished name (DN) namespace for the Enterprise User Security database administrators group stored in an external repository to the same payload that Oracle Virtual Directory sends to the database.

4.3.5.1 Configuration Parameters

The following is a list and description of the EUSMemberDNMapping plug-in configuration parameters:

remoteDomainDN
The base DN in the remote external repository (Active Directory, Oracle Internet Directory, Sun Java System Directory Server) where the Enterprise User Security database administrators group is located.

localDomainDN
The base DN of the name of the group that Oracle Virtual Directory exposes.

Note: Typically, the values for the remoteDomainDN and localDomainDn configuration parameters are identical.

4.3.6 EUSLockout Plug-In

Use the EUSLockout plug-in only when integrating Oracle Virtual Directory with Oracle's Enterprise User Security database product. LDAP servers have the ability to lock a user account after several bind attempts fail. The EUSLockout plug-in allows the Oracle Virtual Directory-Enterprise User Security integration to utilize this lockout feature and enforce the back-end LDAP server's password lockout policy as follows:

- An incorrect login to the Oracle Database records a login failure to the back-end LDAP server
- A correct login to the Oracle Database resets the login failure count in the back-end LDAP server

---

Note: This functionality is not available for integrations that use Active Directory.

- A locked user account cannot be used to log in to the Oracle Database.

See Also: Enabling User Account Lockout on page 19-25

When you configure the EUSLockout plug-in, you must:

- Create a **directoryType** parameter with a value according to your back-end LDAP server, such as **ActiveDirectory** for Active Directory, **iPlanet** for Sun Java System Directory Server, or **eDirectory** for Novell eDirectory.

- Create a namespace using the name of your Oracle Virtual Directory-Enterprise User Security integration user container.

4.3.6.1 Configuration Parameters

The following is a list and description of the EUSLockout plug-in configuration parameters:

directoryType
The type of back-end directory server in the Oracle Virtual Directory-Enterprise User Security integration where the user identities are stored. Supported values are **ActiveDirectory** for Active Directory, **iPlanet** for Sun Java System Directory Server, or **eDirectory** for Novell eDirectory.

4.3.7 ONames Plug-In

The Onames plug-in is used only when integrating Oracle Virtual Directory with Oracle Net Services. The plug-in removes Oracle Internet Directory-specific entries to facilitate the Oracle Virtual Directory-Oracle Net Services integration.

4.3.7.1 Configuration Parameters

The ONames plug-in has no configuration parameters. To enable the ONames plug-in, add it to a plug-in chain.

4.3.8 SubschemaSubentry Plug-In

When Oracle database queries Oracle Virtual Directory in Enterprise User Security and Oracle Net Services integrations it expects LDAP schema to be in same name as Oracle Internet Directory. However, Oracle Internet Directory and Oracle Virtual Directory store the LDAP schema differently. The SubschemaSubentry plug-in transparently redirects the Oracle database queries to allow Oracle Virtual Directory-Enterprise User Security or Oracle Virtual Directory-Oracle Net Services integrations to function.

4.3.8.1 Configuration Parameters

The SubschemaSubentry plug-in has no configuration parameters. To enable the SubschemaSubentry plug-in, add it to a plug-in chain.

4.4 Understanding the Microsoft Active Directory Plug-Ins

Microsoft Active Directory has several features that many applications do not know how to handle. Oracle Virtual Directory includes multiple plug-ins to allow applications to use these unique features without affecting, recoding, or reconfiguring the application.

This topic describes the Microsoft Active Directory and Active Directory Application Mode (ADAM) plug-ins included in Oracle Virtual Directory and contains the following sections:

- ActiveDirectory Password Plug-In
- Active Directory Ranged Attributes Plug-In
- InetAD Plug-In

4.4.1 ActiveDirectory Password Plug-In

Active Directory and ADAM have special rules about how the password of a user may be updated by using LDAP, including:

- Passwords may only be updated via secure SSL connection
- If a user is updating their own password, the original password must be included in a modify delete with the new password being a modify add in the same modify operation.
- Only an administrator may reset the password of a user without knowing the previous password.
- Active Directory does not use the userPassword attribute—it uses the unciodePwd attribute which is in Unicode format.

The ActiveDirectory Password plug-in helps administrators with the Active Directory's password update rules when an application is not designed to use them and it is not advantageous to connect to Active Directory or ADAM via SSL for all operations. By configuring the ActiveDirectory Password plug-in on a non-SSL enabled adapter and pointing the plug-in to an SSL-enabled adapter, this plug-in allows a password update on Active Directory to work as a password update on an inetOrgPerson directory would.

Important: The ActiveDirectory plug-in must be configured only on an LDAP Adapter, typically against Microsoft Active Directory.

4.4.1.1 Configuration Parameters

The ActiveDirectory Password plug-in has the following configuration parameter:

adapter
The name of the adapter the ActiveDirectory Password plug-in will reroute requests to if they contain a userPassword attribute. The adapter identified must have its virtual root be the same as the current adapter and its Routing Visibility setting must be set as Internal. If the adapter parameter is not defined, the current adapter is used.

mapPassword
Indicates whether the password must be converted to the unicodePwd attribute (true), or not (false). Supported values are true or false. The default value is true.

4.4.2 Active Directory Ranged Attributes Plug-In

Attributes in Active Directory and ADAM with more then 1000 values are returned 1000 at a time with a name that includes the range of values that were returned (or 1500 for Windows 2003). The range is returned to the client in the following format:

```
member;1-1000: somevalue
```

To get the next thousand entries, the client application must know to repeat the query and request the attribute member;1001-2000. This requires applications to handle Active Directory in a unique method when compared to other directory products.

The Active Directory Ranged Attributes plug-in detects attributes that Active Directory or ADAM has "ranged" and automatically retrieves all values. If the Active Directory Ranged Attributes plug-in is not enabled, the LDAP adapter's dn-attribute configuration option is not applied because the range of values are appended to the attribute name.

Important: The Active Directory Ranged Attributes plug-in determines what attributes on an adapter are marked as dn-attributes and performs the appropriate base mapping. You can only use the Active Directory Ranged Attributes plug-in as an adapter plug-in on an LDAP adapter.

4.4.2.1 Configuration Parameters

The Active Directory Ranged Attributes plug-in has no configuration parameters. To enable the Active Directory Ranged Attributes plug-in, add it to a plug-in chain.

4.4.3 InetAD Plug-In

The InetAD plug-in combines the functionality of the ObjectClass Mapper Plug-In, Active Directory Ranged Attributes Plug-In, and ActiveDirectory Password Plug-In to allow one single plug-in to be configured to handle multiple unique Active Directory features.

The InetAD plug-in utilizes the ObjectClass Mapping plug-in because most LDAP directories use the inetOrgPerson and groupOfUniqueNames object classes for users and groups, while Active Directory uses the user and group objectClasses with attributes specific to Active Directory's NOS requirements.

The InetAD plug-in allows Oracle Virtual Directory to make an Active Directory or ADAM directory server appear to have inetOrgPerson schema. Based on the parameter configuration, the InetAD plug-in can rename attributes and object classes and add attributes for a user in Active Directory to have all needed attributes. If no configuration parameters are used, the InetAd plug-in makes an Active Directory user or group appear to be an inetOrgPerson or groupOfUniqueNames object class.

Note: Where attribute mapping relationships occur, the `client-` prefix indicates client side and the `source-` prefix indicates data source side. For example, mapping Active Directory server user to represent it as InetOrgPerson would imply Active Directory is the source side and InetOrgPerson is the client side.

4.4.3.1 Configuration Parameters

The following is a list and description of the InetAD plug-in configuration parameters:

directoryType
The directory type to use when performing user activation. Supported values are SunOne, eDirectory, ADAM, and ActiveDirectory. For example:

Parameter Name: directoryType

Parameter Value: ActiveDirectory

activationAttribute
Use the activationAttribute parameter when an application has no knowledge of the underlying directory's user activation system. The activationAttribute parameter informs Oracle Virtual Directory which incoming attribute contains the user activation flag, which is then mapped to a directory specific attribute and flag. For example:

Parameter Name: activationAttribute

Parameter Value: myuseraccountcontrol

deactivationValue
Comma separated list of attribute values specified in `activationAttribute` that indicate this user should be marked as inactive.

activationValue
Comma separated list of attribute values specified in `activationAttribute` that indicate this user should be marked as active.

mapObjectClass
An objectClass to be mapped in the form of *`client-ObjectClass = AD-ObejctClass`*. For example:

Parameter Name: mapObjectClass

Parameter Value: inetOrgPerson=user

You can use the mapObjectClass parameter multiple times for multiple mappings. The default values are groupOfUniqueNames=group, inetOrgPerson=user.

addAttribute[*-objectclassvalue*]
Adds attributes for a user during the add process. An optional objectclass value may be added to the configuration name to add the attribute only for certain objectclasses. For example, to add a userAccountControl attribute to only the user objectclass, use:

Parameter Name: addAttribute-user

Parameter Value: userAccountControl=546

Note: An additional attribute value may be substituted as an expression by supplying its name surrounded by the percentage character (%). The default configuration is:

```
addAttribute-user: useraccountcontrol=544,
addAttribute-group:samaccountname=%cn%,
addAttribute-group: grouptype=-2147483646
```

---

filterAttribute[*-objectclassvalue*]
Comma-separated list of attributes that will be removed during the add operation and from all returned entries. A conditional objectclass value may be added to the name of the parameter to filter out attributes for a specific objectclass. For example:

Parameter Name: filterAttribute

Parameter Value: objectsid,memberof,samaccountname

mapAttribute
An attribute to be mapped in the form of `client-Attribute=AD-attribute`. For example:

Parameter Name: mapAttribute

Parameter Value: uniqueMember=member

You can use the mapAttribute parameter multiple times for multiple mappings. The default values are "uniqueMember=member", "uid=samaccountname", "ntgrouptype=grouptype".

filterAuxiliaryClass
Comma separated list of objectclasses that must be removed on an add operation. Active Directory for Windows 2000 does not allow auxiliary object classes to be listed while adding an entry, while Microsoft Active Directory and ADAM for Windows Server 2003 does allow for auxiliary classes to be listed. The default value is person, organizationalPerson.

filterObjectClassOnModify
Comma-separated list of attributes that will be removed during the modify operation for a specific objectclass. For example:

Parameter Name: filterObjectClassOnModify

Parameter Value: objectsid,memberof,samaccountname

mapPassword
Indicates whether the password must be converted to the unicodePwd attribute (true), or false if not (ADAM). Supported values are true or false. The default value is true.

sslAdapter
The name of the adapter this plug-in will reroute requests to if they contain `userPassword`. The current adapter is used if this parameter is not set. The adapter named in this option *must* have:

- The same local base as the adapter this plug-in is configured on
- Its Routing Visibility set to **Internal**

4.5 Understanding the Oracle Identity Manager Plug-Ins

This topic describes the plug-ins designed for use when Oracle Virtual Directory is a connector target for Oracle Identity Manager integrations. This topic contains the following sections:

- UserManagement Plug-In
- Changelog Plug-Ins

Note: Some of the UserManagement and changelog plug-in configuration parameters must have identical values.

See: The Oracle Identity Manager documentation library for specific information about integrating Oracle Virtual Directory and Oracle Identity Manager.

You can access the Oracle Identity Manager documentation library on the Oracle Technology Network web site at:

`http://www.oracle.com/technology/index.html`

4.5.1 UserManagement Plug-In

The UserManagement plug-in provides data mapping for Oracle Identity Manager attributes, in particular, attributes needed by Oracle Fusion Applications, to LDAP directory servers.

4.5.1.1 Configuration Parameters

The UserManagement plug-in has the following configuration parameters:

oimLanguages
Comma separated list of language codes to be used in attribute language subtypes. This parameter is functional only when the directoryType parameter is set to ActiveDirectory.

oamEnabled
True or False: Indicates whether or not Oracle Access Manager is deployed with Oracle Identity Manager. By default, Oracle Access Manager is not deployed, therefore the default setting for this parameter is false.

> **Note:** The oamEnabled parameter for the UserManagement plug-in and the changelog plug-in must have identical values.

directoryType
Identifies the type of source LDAP directory server. Supported values are OID, ActiveDirectory, and SunOne. The default value is OID.

> **Note:** The directoryType parameter for the UserManagement plug-in and the changelog plug-in must have identical values.

ssladapter
When the directoryType configuration parameter is set to ActiveDirectory, the ssladapter parameter identifies the name of the adapter the UserManagement plug-in will route requests to when userPassword is contained in requests. The default value is the current adapter.

mapAttribute
Defines the attribute translation in the form of *OVD-attribute=OIM-attribute*, for example: orclGUID=objectGuid. You can set the mapAttribute configuration parameter multiple times to define translations for multiple attributes.

mapPassword
True or False. When the directoryType configuration parameter is set to ActiveDirectory, the mapPassword parameter controls whether or not to convert the user password to the unicodePwd attribute. The default value is false.

mapRDNAttribute
Defines the RDN attribute translation in the form of *OVD-RDNattribute=OIM-RDNattribute*, for example: uid=cn.

pwdMaxFailure
Identifies the maximum number of failed logins the source LDAP directory server requires to lock an account (as defined by the password policy effective on the user entries being exposed through the adapter on which this plug-in is deployed).

mapObjectclass
Defines the objectclass value translation in the form of *OVD-objectclass=OIM-objectclass*, for example: inetorgperson=user. You can set the mapObjectclass configuration parameter multiple times to define translations for multiple objectclasses.

> **Note:** The mapObjectclass parameter for the UserManagement plug-in and the changelog plug-in must have identical values.

addAttribute
In the form of *attribute=value pairs*, this parameter identifies attributes to be added before returning the get operation result. You can prefix the attribute name with `objectclass,` to add the attribute and value to a specific objectclass. You can also surround a value with `%` to reference other attributes. For example, specifying the value `user,samaccountname=%cn%` assigns the value of cn to samaccountname when the entry objectclass=user. Specifying the value `samaccountname=jdoe` adds attribute samaccountname with value jdoe to all the entries.

4.5.2 Changelog Plug-Ins

Oracle Virtual Directory provides the following changelog plug-ins to standardize changelog information from source directories into a suitable format for Oracle Identity Manager:

- oidchangelog for use with Oracle Internet Directory
- sunonechangelog for use with Sun Java System Directory Server
- adchangelog for use with Microsoft Active Directory

4.5.2.1 Configuration Parameters

Each of the changelog plug-ins have the following configuration parameters:

oamEnabled
True or False: Indicates whether or not Oracle Access Manager is deployed with Oracle Identity Manager. By default, Oracle Access Manager is not deployed, therefore the default setting for this parameter is false.

> **Note:** The oamEnabled parameter for the UserManagement plug-in and the changelog plug-in must have identical values.

directoryType
Identifies the type of source LDAP directory server. Supported values are OID, ActiveDirectory, and SunOne. The default value is OID.

> **Note:** The directoryType parameter for the UserManagement plug-in and the changelog plug-in must have identical values.

mapObjectclass
Defines the objectclass value translation in the form of *Source-Directory-objectclass=OIM-objectclass*, for example: inetorgperson=user. You can set the mapObjectclass configuration parameter multiple times to define translations for multiple objectclasses.

> **Note:** The mapObjectclass parameter for the UserManagement plug-in and the changelog plug-in must have identical values.

sizeLimit
Identifies the maximum number of changelog entries to be returned with search filter of `changenumber >= integer` or `changenumber <= integer`. This parameter does not support changelog queries with complex filters using `AND | OR` of two or more filters.

mapAttribute
Defines the attribute translation in the form of *Source-Directory-attribute=OIM-attribute*, for example: orclGUID=objectGuid. You can set the mapAttribute configuration parameter multiple times to define translations for multiple attributes.

targetDNFilter
Identifies the container to retrieve changes from. This parameter can be set multiple times to identify multiple containers to retrieve changes from.

modifierDNFilter
Identifies the modifier DN whose changes will be filtered from the search result. You can set this parameter multiple times to exclude the changes made by the specified modifiers DN list from the search result. This parameter is not supported for use with Microsoft Active Directory.

requiredAttribute
Comma-separated list of attributes to always be retrieved from the source LDAP directory server, regardless of the return attributes list specified for changelog queries to Oracle Virtual Directory.

addAttribute
Comma-separated list of attributes to be added to the normalized changelog entry. For example, orclContainerOC=1, changelogSupported=1, where =1 indicates the changes retrieved from the source directory which support changelog.

mapUserState
True or False. This parameter enables or disables the mapping of the directory specific account attributes to Oracle Virtual Directory virtual account attributes.

4.6 Understanding the Oracle Access Manager Plug-Ins

Oracle Virtual Directory includes plug-ins to simplify the integration with Oracle Access Manager. This topic describes the plug-ins related to this integration and contains the following sections:

- OAMPolicyControl Plug-In

4.6.1 OAMPolicyControl Plug-In

For Oracle Virtual Directory-Oracle Access Manager integrations only, the OAMPolicyControl plug-in is for applications that use LDAP for authentication and want to use Oracle Access Manager policy controls, but cannot integrate with Oracle Access Manager.

4.6.1.1 Configuration Parameters

The OAMPolicyControl plug-in has the following configuration parameters:

> **Note:** All of the following configuration parameters—except for useAccessAuthPolicy—are required to deploy the OAMPolicyControl plug-in.

resourceIdOVD
Identifies the proxy resource for Oracle Virtual Directory that the Oracle Access Manager server configures. For example: //*host*:*port*/ovd_proxy_resource.

identityproxyid
Used for authentication against the Identity Server, the identityproxyid parameter identifies the value of the administrator's usernameAttribute.

install_dir
Identifies the AccessSDK installation directory containing the required libraries. For example: *AccessSDK_INSTALL_DIRECTORY*/AccessServerSDK/.

OrclOVDEncryptedproxypasswd
Administrator password for authentication against Identity Server.

identityEndpointAddress
Identifies the URL corresponding to the listening endpoint of the Identity Server's um_modifyUser web service. For example:
http://*host*:*port*/identity/oblix/apps/userservcenter/bin/userservcenter.cgi

usernameAttribute
Identifies the attribute configured to be the Login attribute of the Identity Server. For example, uid or genUserId.

useAccessAuthPolicy
An optional and case-insensitive parameter, useAccessAuthPolicy determines usage of the Oracle Access Manager server's authorization policies while accessing the proxy resource. Supported values are True and False. The default setting is False.

5

Understanding Oracle Virtual Directory Mapping

This chapter describes Oracle Virtual Directory mapping and includes the following topics:

- What is a Mapping?
- Understanding Mapping Templates
- Example Mapping Deployments
- Mapping Functions

> **Note:** The mapping information in this chapter is included for historical purposes. While existing default mapping scripts are supported, any new customization should be done using the Java plug-in API. This is because the Java API supports full access to all Oracle Virtual Directory functionality and it is also a generally easier environment to develop in.

5.1 What is a Mapping?

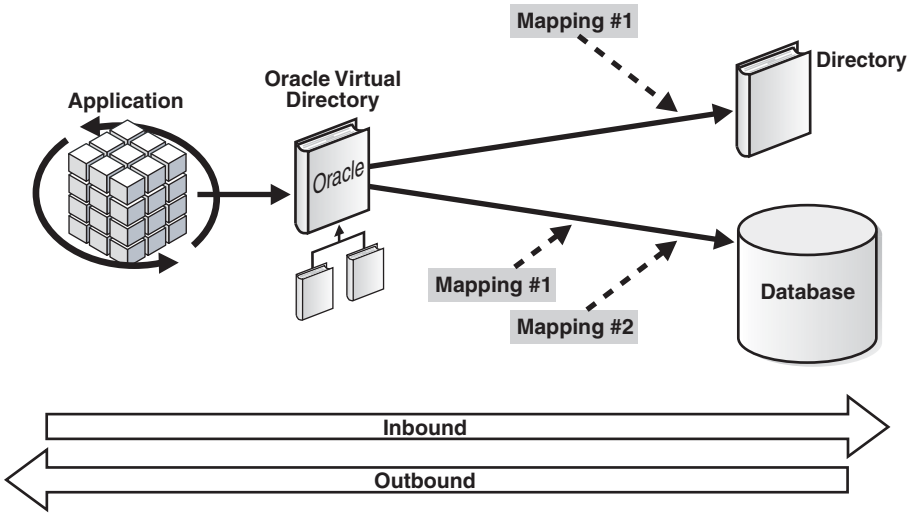
Oracle Virtual Directory includes a bidirectional mapping system based on the Python scripting language. A Mapping is a special Python script, file type .mpy, that processes inbound and outbound transactional data flow and allows Oracle Virtual Directory administrators to manipulate and map data as it passes through the Oracle Virtual Directory server. Based on the popular Python scripting language, Oracle Virtual Directory's mapping system allows you to perform complex data manipulation without learning a new, proprietary, or complicated programming language. Oracle Virtual Directory's mapping system provides enterprises with additional flexibility in supporting identity access from applications. Oracle Virtual Directory compiles mappings into executable byte code and runs it inline for maximum performance.

Integrators can develop easy-to-use mapping scripts that perform custom transformations when mapping information from one data source to another. These scripts can be installed on a running server and deployed without resetting the server. A mapping script can adjust requests as they enter the system on the way to data sources and transform responses on the return path to the client. For example, you can use a mapping to normalize schema, such as making Active Directory look like InetOrgPerson; attach data-type, such as {sha} to a hashed password; or create a virtual attribute based on values of attributes retrieved from a data store.

When you create a Mapping you can use a predefined mapping template to simplify its configuration or you can create a new custom mapping (refer to "Understanding

Mapping Templates" for more information on mapping templates). Typically, a Mapping is deployed to the Oracle Virtual Directory server as compiled Java code and runs inside a special type of plug-in known as a Mapper. As with Plug-ins, a Mapping may run globally or at an adapter level. Multiple mappings and adapters can be combined as a set of discrete functions performing an overall conversion service. Figure 5–1 shows a typical scenario where one Mapping is running on multiple adapters, while another Mapping is running only on a specific adapter.

Figure 5–1 Example Mapping Deployment on a Single Adapter and Multiple Adapters



Each Mapping has an inbound and outbound flow, allowing it to translate one way as a request is received and reverse that translation as results are returned to the requesting application. This programmatic reversal is important because it is not usually possible for the server to guess intent.

Oracle Virtual Directory provides a lot of flexibility in determining whether a Mapping should be executed globally or within the context of a single adapter. In some situations, you may need to further restrict the locations in the virtual tree where a Mapping is applied. For example, an adapter is set-up to proxy a Microsoft Active Directory domain and points to DC=VAN,DC=Oracle,DC=com. Under that point in the directory tree, there is a CN=Users container and a CN=Groups container. You can add a namespace filter to any Mapping to apply it to only one part of the tree.

The following is a list of notes to consider regarding Oracle Virtual Directory Mappings:

- Oracle Virtual Directory mappings make extensive use of the Python language with additional Oracle provided functions for LDAP data manipulation. For more information about Python, refer to the Python Programming Language Official Website at:

  `http://www.python.org/`

- If you rename attributes with Mappings, Oracle Virtual Directory supports search on the renamed attribute/value only if the custom code overrides the incoming filter object, as is in the DB_Groups Mapping. For example:

  During outbound processing, a Mapping renames the givenname attribute to cn. During inbound processing, an incoming LDAP search filter, such as cn=John must be converted to givenname=John by the Mapping custom code.

- If you deploy a Mapping and then run an ldapsearch command against Oracle Virtual Directory, the search base must be the namespace configured for the Mapping or any child of that namespace.

5.1.1 When to Use a Mapping and When to Use a Custom Plug-in

Most customers who use mappings use default mappings shipped with Oracle Virtual Directory to meet an application requirement. For customization needs, the Java plug-in API is typically used because the Mapper API only handles a subset of Java plug-in functionality and more developers know Java than Python, thus reducing time needed to develop the code.

5.1.2 Overview: Deploying Mappings

The following is an overview of the process for deploying Mappings at an adapter level and at a global server level:

1. Construct the Mapping using Mapping templates, compile it into a script, and deploy it to the Oracle Virtual Directory server so that it can be activated at either the global server level or at the adapter level.

   See: "Constructing Mappings Using Mapping Templates" on page 14-1 for more information.

2. Configure the Mapping at the global server level or at the adapter level by naming it, identifying its Mapping script file, determining the namespaces in the virtual directory tree where you want it to execute, and then activating it.

   See:

   - "Creating and Activating Server Mappings" on page 14-2 for information on configuring Mappings at the global server level.
   - "Applying Mappings to Adapters" on page 14-3 for information on configuring Mappings at the adapter level.

5.2 Understanding Mapping Templates

This topic describes each of the Oracle Virtual Directory Mapping templates and includes the following sections:

- Active_Directory_to_inetOrg
- Common_Name_to_Given_Name
- ConditionalPublish
- DB_Groups
- Map_DB_Password

5.2.1 Active_Directory_to_inetOrg

Maps Microsoft Active Directory user and group objects to the inetOrgPerson and groupOfUniquenames objects (respectively).

5.2.2 Common_Name_to_Given_Name

Creates a virtual common name attribute by combining values from two attributes, default sn and givenname. The Common_Name_to_Given_Name mapping is typically used with the Database Adapter, which may have only a first and last name, but no full name.

Note: This mapping does not support substring filters for common name attributes.

5.2.3 ConditionalPublish

Removes the attributes specified if the conditional value in another attribute is met. The ConditionalPublish mapping is useful to hide FERPA protected attributes in a higher education environment.

5.2.4 DB_Groups

Use this template to map a table that describes a group into a valid LDAP group. The first column is assumed to be cn, that is, the name of the group. The second column is assumed to be the uniquemember. In the case of uniquemember, the DN is stripped so that only the RDN value is used inside the table. For example, converting: (uniqueMember=cn=XXX,ou=testusers) to (uniqeMember=XXX).

5.2.5 Map_DB_Password

Maps inbound binary syntax passwords to IA5String passwords compatible with the database.

Note: If you associate the Map_DB_Password Mapping with a Database Adapter, then perform an LDAP modify with changetype Add and a binary attribute such as UserPassword with its value already existing in Oracle Virtual Directory, a duplicate row is added in the database if the primary key constraint is not present in the database table.

5.3 Example Mapping Deployments

This topic provides two examples for common mapping deployments and contains the following sections:

- Constructing Common Name Attributes from Givenname and Surname Attributes
- Mapping Microsoft Active Directory Schema

5.3.1 Constructing Common Name Attributes from Givenname and Surname Attributes

Overview

This example explains how to create a common name (cn) from a givenname and a surname (sn). This type of mapping deployment is useful when using a Database Adapter to provide an LDAP interface to a user data stored in a database. While LDAP directories generally store a cn, databases tend to store only a first name and last name. When performing a search, it could become very complicated when filtering on

common name. For example, the filter (cn=Marc Boorshtein) would need to read (&(givenName=Marc)(sn=Boorshtein)).

Mapping Requirements

The following is a list of hypothetical requirements for this example mapping:

- When data is retrieved from the adapter, you want to form a cn by combining givenname with sn.
- On the inbound side, you want to split cn into givenname and sn. If cn is present in the attribute request list, the list will be changed to include givenname and sn.
- If the inbound operation is a search operation, you want to check the search filter and convert the cn appropriately.

Mapping

```
def parceCN(val):
  return split(val,' ',2)

def inbound():
  #map the "cn" filters
  if operation == 'get':
     if haveAttribute('cn'):
         addAttribute('givenName')
         addAttribute('sn')

     cnFilters = findFilters('cn')
     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])

def outbound():
  #outbound stuff
  addAttributeValue('cn',getAttributeValue('givenName') + ' ' +
                    getAttributeValue('sn'))
```

Inbound Processing

In the `inbound()` function you want to convert any cn into separate givenname and sn attributes. For a search, you want to convert search filters for cn into a combined filter for givenname and sn so you will create a new function, `parceCN()`.

On the first line of the mapping, the split function is imported from the Python string module. The `parseCN()` Python function is defined to take a cn and split it into a first and last name based on detecting a space.

> **Note:** In reality, this is more complex, for example, when middle names are used. For the purposes of this example, we will consider this simple case to get started. Contact your Oracle Support representative for help with advanced mapping situations.

Next, you define the `inbound()` function. The inbound function could deal with any LDAP operation, but in this case, you are interested in looking at search operations. The first line after inbound is therefore an `if` block that tests the value of operation. The variable operation contains either add, bind, delete, get, modify, or rename.

`If operation = get`, the mapping proceeds by determining if the search request had cn in the attribute request list(). Since cn can only be formed by combining givenname and sn, you need to add givenname and sn to the search list using the `addAttribute()` function.

To process filter requests for cn, the mapping retrieves all filter elements whose target is the cn attribute(). For each filter, the mapping parses it, calculates the corresponding givenname and sn values by calling `parseCN`, and creates new givenName and sn filters. Lastly, the inbound function of the mapping replaces the filter term with cn with a combined filter including the givenName and sn.

Outbound Processing

The outbound function handles all transactions that are flowing from the adapter to the client. In this example, you want to form a cn from two other values and you use the `addAttributeValue` function to create a new cn value by combining givenname, a space, and the sn value. Notice how existing values are retrieved using the `getAttributeValue` function, which retrieves a specific attribute from the current entry returned to the client.

5.3.2 Mapping Microsoft Active Directory Schema

Overview

Frequently applications require the use of an LDAP directory using inetorgperson and groupofuniquenames schema objects. However, many organizations use Microsoft Active Directory which supports only user and group objects. This example mapping deployment illustrates how to use a mapping to make an Active Directory schema look like inetorg style schema using inetorgperson or groupofuniquenames.

Mapping Requirements

The following is a list of the translation and mapping requirements for this example mapping:

- Bidirectional mapping of attributes names. For example `uniquemember = member`, `uid = samaccountname`, and so on.
- Conversion of objectclass names. Not only do the basic objectclass names need to change, but you must also consider that Microsoft Active Directory does not use auxiliary objectclasses. For example, objectclass values of `interorgperson`, `organizationalperson`, or `person` must be collapsed to just `user`.
- Adding special attribute values. Microsoft requires the use of additional object type codes such as `groupType` or `userAccountControl`. Depending on the operation, special tags must be added to the request.
- RDN conversion. Microsoft typically uses cn as the relative distinguished name of user accounts. Many applications expect the use of uid.

Mapping

Using the following small script, an inetOrg application may use a Microsoft Active Directory:

```
def inbound():
      #first rename the attributes
      rename({'uniqueMember':'member','uid':'samaccountname','userpassword':
      'unicodepwd','ntgrouptype':'grouptype'})

      #map nessasary object class values
      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':
         if haveAttributeValue('objectClass','group'):
               addAttributeValue('groupType','-2147483646')
               if not haveAttribute('samaccountname'):
                        copy('cn','samaccountname')

         if haveAttributeValue('objectClass','user'):
               addAttributeValue('userAccountControl','66048')

     #collapse aux classes
      removeAttributeValue('objectClass','person')
      removeAttributeValue('objectClass','organizationalPerson')

      #set the rdn
      setRDN('samaccountname','cn')

def outbound():
    #first rename the attributes
    rename({'member':'uniqueMember','samaccountname':'uid','unicodepwd':
    'userpassword','grouptype':'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')
```

Inbound Processing

The first line of the `inbound()` function renames all inetorg attributes to Active Directory attributes. The rename function is called for all operations. For example, if the operation is a search, then all requested attributes will be renamed as well as all attributes in the filter. If the operation is an add or modify, then all attributes effected are renamed.

The second section of the inbound function replaces inetOrg object classes with InetAD object classes. Notice that you can use conditional statements to determine what actions should be performed.

The third section of the inbound function checks to see if the operation is an add, and if so, it adds the specific attribute information required by Active Directory.

In the fourth section of the inbound function all auxiliary object classes are removed because Active Directory does not allow for an auxiliary object class to be directly specified during an add.

In the last section of the inbound function the RDN is changed from uid to cn. Notice that the code converts `samaccountname` to `cn` because `uid` was already renamed to `samaccountname`. This does more than just change the rdn from a uid to cn, but it will deal with locating the cn if it's not specified (for example, in a modify or a search).

Outbound Processing

The `outbound()` function executes after a response is returned from Active Directory. The `outbound()` function reverses the inbound function by first renaming all applicable attributes, then mapping the object class names, and then changing the rdn of any results.

5.4 Mapping Functions

Oracle Virtual Directory Mappings are based on the Python language and can use any functions or subroutines available in Python. In addition to the Python functions supported by Oracle Virtual Directory, Oracle provides the library functions described in the following sections:

- Methods
- Data Objects

5.4.1 Methods

The following is a list of the methods available for Oracle Virtual Directory Mappings in addition to those of the Python language:

Note: Methods specifying `Map xxxxx` indicates that you can specify a list of values in the form:

```
{'uniqueMember':'member','uid':'samaccountname',[…] }
```

This is essentially an array of one or more mapped values. Use this construct for those methods that support it when a particular method is to be used multiple times for different named pair relationships (for example, `rename` in the "Mapping Microsoft Active Directory Schema" example). This syntax is good shorthand and also yields improved performance.

---

appendAttribute(source,destination)

operations: add, modify, get, entry

The `appendAttribute` function adds the values of the source attribute to the destination attribute. The source attribute remains in place. This function will effect a search filter.

```
Example: appendAttribute('sn','givenName')
add/entry:
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
becomes:
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
becomes:
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
givenName: First
givenName: Last

get:
(&(givenName=first)(sn=last))
becomes:
(&(|(sn=last)(givenName=last))(givenName=first))
```

copyAttribute(source,destination)

operations: add, modify, get, entry

The `copyAttribute` function copies attribute values from the source attribute to the destination attribute, overwriting the destination attribute if it already exists.

```
Example: copyAttribute('sn','givenName')
add/entry:
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
becomes:
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
becomes:
dn: cn=User
changetype: modify
add: sn
```

```
sn: Last
-
add: givenName
givenName: First
givenName: Last

get:
(&(givenName=first)(sn=last))
becomes:
(|(sn=last)(givenName=last))
```

renameAttribute(source,destination)

operations: add, modify, get, entry

Renames the source attribute to the destination attribute. If the destination attribute already exists, it is overwritten. If the source attribute does not exist, but the destination attribute does, the destination attribute is removed.

```
Example: renameAttribute('sn','givenName')
add/entry:
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
becomes:
dn: cn=User
objectClass: person
cn: User
givenName: name

modify:
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
givenName: First
becomes:
dn: cn=User
changetype: modify
add: givenName
givenName: Last

get:
(&(givenName=first)(sn=last))
becomes:
(givenName=last)
```

removeAttribute(attribute)

operations: add, modify, get, entry

Removes the named attribute, returning its values in a list. If the attribute is a part of an entry, the values are returned. If the value is part of a changelist, the EntryChange object is returned.

```
Example: removeAttribute('sn')
add/entry:
```

```
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
becomes:
dn: cn=User
objectClass: person
cn: User
givenName: User

modify:
dn: cn=User
changetype: modify
add: sn
sn: Last
-
add: givenName
givenName: First
becomes:
dn: cn=User
changetype: modify
add: givenName
givenName: First
givenName: Last

get:
(&(givenName=first)(sn=last))
becomes:
(givenName=last)
```

revalueAttribute(attribute,currentValue,newValue)

operations: add, modify, entry, get

```
Example: revalueAttribute('sn','name','newname')
add/entry:
dn: cn=User
objectClass: person
cn: User
givenName: User
sn: name
becomes:
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
becomes:
dn: cn=User
changetype: modify
add: sn
```

```
sn: newname
-
add: givenName
givenName: First
givenName: Last

get:
(&(givenName=first)(sn=name))
becomes:
(&(givenName=last)(sn=newname))
```

mapSyntax(value,newSyntax)

operations: add, modify, entry

Maps a syntax value to a new syntax. If the first argument is a `Syntax` object, the function will return an instance of `Syntax` as named by `newSyntax`. Valid syntaxes are: DirectoryString, IA5String, BinarySyntax, and BinarySyntax.

mapSyntax(attribute,newSyntax)

operations: add, modify, entry

Maps an attribute value to a new syntax. If the first argument is the name of an attribute, all instances of that attribute are mapped to the new syntax. Valid syntaxes are: DirectoryString, IA5String, BinarySyntax, and BinarySyntax.

splitValue(newNames,currentName,parseFunction,index,remove)

Splits the existing values in an attribute and the result of the split is added to the listed attributes in the `newNames` parameter. An example of when the splitValue functions are useful is if you encounter an adapter that only has cn values but you want to create givenname and sn values from the cn value. By default, the first value found is taken if the attribute is multi-valued, the split is based on space, and the initial values are left alone. The following is a list and explanation for the parameters for this method:

- newNames: an array of attributes that will be created from the split values.
- currentName: the existing attribute to create the value from.
- parseFunction: an optional parameter that identifies the function for parsing data if a parsing rule different than space is needed.
- index: for multi-valued attributes, this parameter identifies which value to split.
- remove: if true (1), the original data from the autoboot in the result is removed and source is left alone.

splitValue(newNames,currentName,parseFunction,index)

Splits the existing values in an attribute and the result of the split is added to the listed attributes in the `newNames` parameter. An example of when the splitValue functions are useful is if you encounter an adapter that only has cn values but you want to create givenname and sn values from the cn value. By default, the first value found is taken if the attribute is multi-valued, the split is based on space, and the initial values are left alone. The following is a list and explanation for the parameters for this method:

- newNames: an array of attributes that will be created from the split values.
- currentName: the existing attribute to create the value from.
- parseFunction: an optional parameter that identifies the function for parsing data if a parsing rule different than space is needed.
- index: for multi-valued attributes, this parameter identifies which value to split.

splitValue(newNames,currentName,parseFunction,remove)
Splits the existing values in an attribute and the result of the split is added to the listed attributes in the `newNames` parameter. An example of when the splitValue functions are useful is if you encounter an adapter that only has cn values but you want to create givenname and sn values from the cn value. By default, the first value found is taken if the attribute is multi-valued, the split is based on space, and the initial values are left alone. The following is a list and explanation for the parameters for this method:

- newNames: an array of attributes that will be created from the split values.
- currentName: the existing attribute to create the value from.
- parseFunction: an optional parameter that identifies the function for parsing data if a parsing rule different than space is needed.
- remove: if true (1), the original data from the autoboot in the result is removed and source is left alone.

splitValue(newNames,currentName,parseFunction)
Splits the existing values in an attribute and the result of the split is added to the listed attributes in the `newNames` parameter. An example of when the splitValue functions are useful is if you encounter an adapter that only has cn values but you want to create givenname and sn values from the cn value. By default, the first value found is taken if the attribute is multi-valued, the split is based on space, and the initial values are left alone. The following is a list and explanation for the parameters for this method:

- newNames: an array of attributes that will be created from the split values.
- currentName: the existing attribute to create the value from.
- parseFunction: an optional parameter that identifies the function for parsing data if a parsing rule different than space is needed.

splitValue(newNames,currentName,index,remove)
Splits the existing values in an attribute and the result of the split is added to the listed attributes in the `newNames` parameter. An example of when the splitValue functions are useful is if you encounter an adapter that only has cn values but you want to create givenname and sn values from the cn value. By default, the first value found is taken if the attribute is multi-valued, the split is based on space, and the initial values are left alone. The following is a list and explanation for the parameters for this method:

- newNames: an array of attributes that will be created from the split values.
- currentName: the existing attribute to create the value from.
- index: for multi-valued attributes, this parameter identifies which value to split.
- remove: if true (1), the original data from the autoboot in the result is removed and source is left alone.

splitValue(newNames,currentName,index)
Splits the existing values in an attribute and the result of the split is added to the listed attributes in the `newNames` parameter. An example of when the splitValue functions are useful is if you encounter an adapter that only has cn values but you want to create givenname and sn values from the cn value. By default, the first value found is taken if the attribute is multi-valued, the split is based on space, and the initial values are left alone. The following is a list and explanation for the parameters for this method:

- newNames: an array of attributes that will be created from the split values.
- currentName: the existing attribute to create the value from.
- index: for multi-valued attributes, this parameter identifies which value to split.

splitValue(newNames,currentName,remove)
Splits the existing values in an attribute and the result of the split is added to the listed attributes in the `newNames` parameter. An example of when the splitValue functions are useful is if you encounter an adapter that only has cn values but you want to create givenname and sn values from the cn value. By default, the first value found is taken if the attribute is multi-valued, the split is based on space, and the initial values are left alone. The following is a list and explanation for the parameters for this method:

- newNames: an array of attributes that will be created from the split values.
- currentName: the existing attribute to create the value from.
- remove: if true (1), the original data from the autoboot in the result is removed and source is left alone.

splitValue(newNames,currentName)
operations: add, modify, entry

```
Example: splitValue(['givenName','sn','cn',1)
add/entry:
dn: uid=User
objectClass: person
cn: First Last
uid: User
becomes:
dn: uid=User
objectClass: person
givenName: First
sn: Last
uid: User

modify:
dn: uid=User
changeType: modify
replace: cn
cn: First1 Last1
becomes:
dn: uid=User
changeType: modify
replace: givenName
givenName: First1
-
replace: sn
sn: :Last1

get:
(cn=First Last)
becomes:
(&(givenName=First)(sn=last))
```

addAttributeValue(name,value)
operations: add, modify, entry

Adds a value to the names attribute, or creates it if it does not exist. In the case of modify, if the attribute does not exist, then an `Add` modification item is created.

```
Example: addAttributeValue('myattrib','myval')
Example: addAttributeValue('noattrib','hasvalue')
add/entry:
dn: uid=User
objectClass: person
```

```
cn: First Last
uid: User
myattrib: noval
becomes:
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
becomes:
dn: uid=User
changeType: modify
delete: myattrib
myattrib: someval
myattrib: myval
-
changetype: add
add: noattrib
noattrib: hasValue
```

haveAttribute(attributeName)

operations: add, modify, entry, get

Returns 1 (true) or 0 (false) if the named attribute exists.

haveAttributeValue(attributeName,attributeValue,fetchFromServer)

operations: add, modify, entry, get

Returns 1 (true) or 0 (false) if the named attribute exists. If `fetchFromServer` is 1, then the entry is fetched from the server.

haveAttributeValue(attributeName,attributeValue)

operations: add, modify, entry, get

Returns 1 (true) or 0 (false) if the named attribute and associated value exists.

haveAttribute(attributeName,attributeValue,fetchFromServer)

operations: add, modify, entry, get

Returns 1 (true) or 0 (false) if the named attribute and associated value exists. If `fetchFromServer` is 1, then the entry is retrieved from the server for comparison.

removeAttributeValue(attributeName,attributeValue)

operations: add, modify, get, entry

Removes an attribute value, returning true if the value was removed.

Example: `removeAttributeValue('myattribute','myvalue')`

```
add/entry:
dn: cn=user
objectClass: person
cn: user
```

```
myattribute: myvalue
myattribute: myvalue2
becomes:
dn: cn=user
objectClass: person
cn: user
myattribute: myvalue2

modify:
dn: cn=User
changetype: modify
replace: myattribute
myattribute: myvalue
-
add: sn
sn: last
becomes:
dn: cn=User
changetype: modify
add: sn
sn: last

get:
(&(sn=last)(myattribute=myvalue))
becomes:
(sn=last)
```

setRDN(oldRDNAttribute,newRDNAttribute)

operations: add, modify, delete, bind, rename, entry

Changes the RDN of the current name (base for get) from the old RDN to a new RDN attribute.

```
Example: setRDN('cn','uid')
add/entry:
dn: cn=user
objectClass: inetOrgPerson
uid: userid
cn: user
becomes:
dn: uid=userid
cn: user
objectClass: inetOrgPerson

modify:
dn: cn=user
changetype: modify
add: sn
sn: last
becomes:
dn: uid=userid
changetype: modify
add: sn
sn: last

bind/get/delete:
dn: cn=user
becomes:
dn: uid=userid
```

addReturnAttribute(attributeName)
operations: get

Adds an attribute to the return attribute list during a search.

findFilters(attributeName)
operations: get

Returns a list of all filters that involve the specified attribute.

createfilter(target,operation,value)
operations: get

Creates a new filter object, with the target being the attribute being tested, the operation being one of the possible comparators and value being the value to filter on.

createAndFilter(filters)
operations: get

Creates an and filter from a list of filters

createOrFilter(filters)
operations: get

Creates an or filter from a list of filters

getAttributeValue(attributeName)
operations: add, entry

Returns the first value of the named attribute

getAttributeValues(entry,attributeName)
operations: any

Retrieves that values of the named attribute from the supplied entry.

createEntryChange(type,attribute,value)
operations: modify

Creates and returns a new `EntryChange` object.

addEntryChange(entryChange)
operations: modify

Adds `en` entry change to the list of entry changes.

getByName(dn)
operations: any

Returns the named entry.

convertBase(attributeName,oldBase,newBase)
operations: add, modify, entry, get

Replaces the `oldBase` with the `newBase` for the values of the named attribute.

removeAttributeValue(attributeName)
operations: add,modify,get,entry

Removes an attribute and returns its values if it exists (or EntryChange's if it is during a modify operation).

Example: `removeAttributeValue('myattribute')`
add/entry:

```
dn: cn=user
objectClass: person
cn: user
myattribute: myvalue
myattribute: myvalue2
becomes:
dn: cn=user
objectClass: person
cn: user
modify:
dn: cn=User
changetype: modify
replace: myattribute
myattribute: myvalue
-
add: sn
sn: last
becomes:
dn: cn=User
changetype: modify
add: sn
sn: last
get:
(&(sn=last)(myattribute=myvalue))
becomes:
(sn=last)
```

5.4.2 Data Objects

Data objects are variables that are made available from Oracle Virtual Directory to you in the Python environment. Use these variables to get handles to Oracle Virtual Directory data structures and to interpret various objects and status items.

operation
The current operation. Possible values are: add, modify, delete, rename, get, entry, and bind.

vsi
Retrieve a handle to Virtual Services Interface (VSI).

attributes
Attributes requested to be returned. Operations: get

base
The current search base. Operations: get

target, op, val = filter.contents and filter.contents = newfilter
Returns and sets the filter in the form of a tuple (target, operation, value).

boolean filter.isAnd
Returns TRUE if filter is an AND filter.

boolean filter.isOr
Returns TRUE if filter is an OR filter.

boolean filter.isNot
Returns TRUE if filter is a NOT filter.

changeEntries
Set of changes for a modify operation. Operations: modify

creds
The current credentials (DN) of the user. Operations: All

entry
The entry to be added or returned from a search. Operations: get, add

filter
The current search filter. Operations: get

name
The entry to be added, bound, modified or deleted. Available for all operations.

request
Retrieve a Value(s): val = request([String name])

Store a Value(s): request(['myname'])='myvalue'

Returns and sets the current request information object attribute specified. This object is used as a method for passing arbitrary information between different mappings or plug-ins that exist for the duration of a specific transaction. For example, during an inbound operation, you can store information that can be used for processing later during the outbound request.

results
Returns and sets result code if an error occurs. Operations: add, delete, modify

scope
The current search scope in the form of 0 (base), 1 (onelevel), or 2 (subtree). Operations: get

typesOnly
Whether the server is returning only types and not values. Operations: get

6

Understanding Oracle Virtual Directory Security

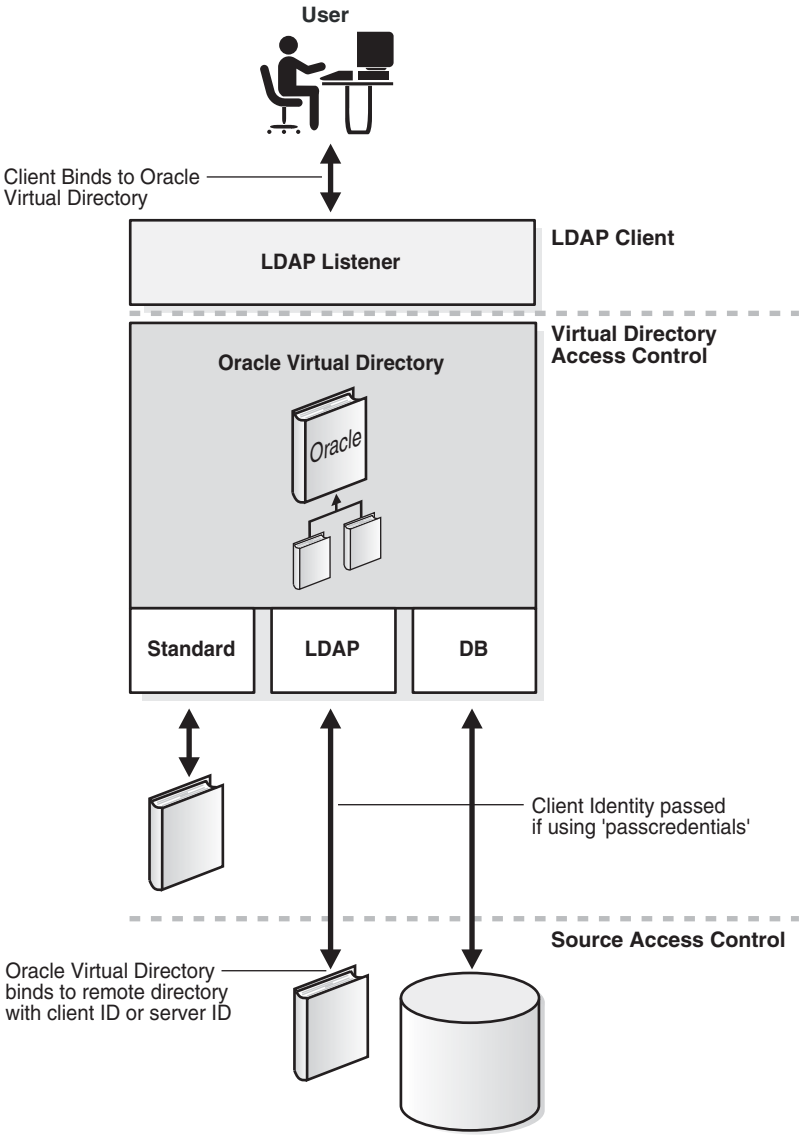
This chapter describes Oracle Virtual Directory security and includes the following topics:

- Overview
- Understanding Oracle Virtual Directory Authentication
- Understanding Oracle Virtual Directory Access Control
- Understanding Wallet and Certificate Management

6.1 Overview

Oracle Virtual Directory supports multiple tiers of access control and authentication. For remote directories accessed through adapters, Oracle Virtual Directory supports the security inherent in those systems. Depending on the adapter used and its capabilities, a passcredentials option can be set to determine if end-user binding credentials should be passed to the remote directory for authentication and access control enforcement, as shown in Figure 6–1:

Figure 6–1 Oracle Virtual Directory Multi-Layered Access Control and Authentication



6.2 Understanding Oracle Virtual Directory Authentication

This topic describes Oracle Virtual Directory authentication and contains the following sections:

- Pass-Through Authentication
- CRAM-MD5 and SASL Binding
- Proxy Account Authentication
- Client Certificate Authentication

6.2.1 Pass-Through Authentication

When an adapter has pass-through mode enabled and a user is to be authenticated to Oracle Virtual Directory, Oracle Virtual Directory will use the user-id and password credentials it receives to log in to the remote directory on the users behalf (for

password authentication only). If the authentication (bind) to the remote directory fails, Oracle Virtual Directory will fail the attempted bind by the user. In this mode, the remote directory is responsible for confirming a user's credentials.

When passcredentials is set to never (or is not supported by the selected adapter), Oracle Virtual Directory must perform the authentication of clients itself. In order for this to work, passwords in external directories must be stored in clear text or must use the CRYPT, SHA, or SSHA one-way encryption hash. For Oracle Virtual Directory to determine which encryption hash is being used, a prefix of the form {crypt} must be applied to the encrypted text. If the proxied source does not use this format, you will need to set up a mapping rule to define it. The mapping rule will add the prefix telling Oracle Virtual Directory how to handle a particular encryption format. If no prefix is present, a normal text comparison is made.

In passcredentials never mode, authentication is completed by Oracle Virtual Directory by performing the hash algorithm (specified in the value returned from the adapter) of the password provided by the user and comparing the result with the value returned from the adapter.

Note: When a client binds without the use of a clear text password (for example, with a certificate), the server cannot pass the user's credentials to the proxied directory. The Oracle Virtual Directory will use the configured adapter account to perform the bind verification and perform the LDAP service requested. This is equivalent to what happens when passcredentials is set to never.

6.2.2 CRAM-MD5 and SASL Binding

CRAM-MD5 is a challenge-response authentication mechanism (CRAM) based on the HMAC-MD5 MAC algorithm, a widely used cryptographic hash function with a 128-bit hash value (or "MD5"). CRAM-MD5 is a Simple Authentication and Security Layer (SASL) bind mechanism used to authenticate to Oracle Virtual Directory.

If the client supports CRAM-MD5, it can be used to keep passwords secret over the wire without using SSL. However, the CRAM-MD5 SASL mechanism requires that the server has a plain text version of the password that it will use to exchange information other than the password that will let the server determine whether a given password provided by an LDAP client is valid. If this mode is used, passwords must be stored in clear text in all local (standard) and proxied sources.

6.2.3 Proxy Account Authentication

Oracle Virtual Directory uses a proxy (or default) account when authenticating users for which no password is available, when proxying users whose bind DN is outside of the adapter's namespace, or when passcredentials is set to never. Oracle Virtual Directory will also use the adapter's proxy user-id and password to authenticate both the Oracle Virtual Directory Root Manager Account and Anonymous to the connected LDAP adapter directory. The default account is also used for users who authenticate using certificates. Therefore, when passcredentials mode is enabled, it is important to understand that the default account should be set to a non-privileged account (for example, anonymous) in remote directory as there are many conditions when the proxy account may be required to handle accounts that cannot be mapped to the current adapter.

6.2.4 Client Certificate Authentication

Oracle Virtual Directory supports the ability for clients to authenticate to the virtual directory using X.509 digital certificates. The LDAP clients must support SSL and SASL to authenticate to the virtual directory using X.509 digital certificates. The following are the two modes in which SSL authentication works:

- Using client certificates as a way to secure the connection but not to authenticate to the actual directory
- Using SASL to bind to the Oracle Virtual Directory using the certificate

The following is a list of guidelines for using client certificates for authentication:

- If using certificates to bind to Oracle Virtual Directory, it is only used to authenticate to Oracle Virtual Directory, not to any back-end data-store. Public Key Infrastructure (PKI) prevents authentication to any back-end data-store because only the LDAP client has access to its private key, which is required to do client certificate authentication. Therefore, all Oracle Virtual Directory operations to the back-end data-store are performed as the Proxy DN account when using the LDAP adapter.
- Certificates contain their own distinguished names (DNs), which sometimes do not match the DN of the user they are actually binding as. In these cases, you may need to map the DN of the certificate to the DN of an user in Oracle Virtual Directory for your Access Control Lists to work properly. You can use a plug-in to accomplish this mapping.
- Oracle Virtual Directory accepts any certificate issued by the root CAs stored in its keys.jks file.

6.3 Understanding Oracle Virtual Directory Access Control

Oracle Virtual Directory provides granular access controls that can be applied uniformly across all connected data stores and which are compliant with the Internet Engineering Task Force's RFC 2820, *Access Control Requirements for LDAP*. The access control rules are modeled on the IETF's internet draft titles *LDAP Access Control Model for LDAPv3*, (March 2, 2001 draft).

Note: Oracle Virtual Directory provides virtualized abstraction of one or more enterprise data sources into a single directory view. Accordingly, Access Control Lists (ACLs) and adapter namespaces are independent of each other. If you remove an entry, the ACLs associated with the entry are also removed. However, the ACLs associated with an entry are not affected if you change the root value of an adapter. ACLs and adapter namespaces must be configured independently of each other.

This topic describes Oracle Virtual Directory access control and contains the following sections:

6.3.1 Source Directory Access Control

As a client to a remote directory, Oracle Virtual Directory must conform to the authorization rules enforced in the remote directory. The rules applied will depend on what user context has been passed to the remote directory, that is, what account is Oracle Virtual Directory using to connect to the proxied directory with.

If the passcredentials option is enabled, the remote directory will enforce rules according to the user context Oracle Virtual Directory presents to the remote directory. Oracle Virtual Directory presents appropriately translated data results and errors back to the user. For example, if an access denied message is returned, Oracle Virtual Directory will return that message to the client. If data is filtered due to access control, Oracle Virtual Directory will take the filtered data, apply any configured translations and then present that data to the user.

If the passcredentials option is not enabled, the remote directory will perceive only the proxy user for all requests and the same authorization will be applied regardless of which user has bound to Oracle Virtual Directory.

Regardless whether the passcredentials option is enabled or disabled, Oracle Virtual Directory provides its own access control and authorization.

6.3.2 Oracle Virtual Directory Access Control

Oracle Virtual Directory supports access control across its entire virtual directory namespace by storing access control information. This information is maintained automatically by intercepting requests to modify the entry DN or subtree DN of the DIT. Because Oracle Virtual Directory Access Control Lists (ACLs) are defined in the namespace of the virtual directory and not in any of the directories connected via the adapters, a single ACL can be defined that governs access to data across several adapters.

When an entry belongs to a proxied LDAP directory using the same access control draft standard, it is impossible to modify access controls within that source directory using Oracle Virtual Directory because Oracle Virtual Directory will intercept the modification and apply it to its own ACI values for the entry. In this case, modification to these entries must be made directly to the source directory. Normally this is not an issue since directory servers from different vendors use different attributes for storing access control information.

6.3.3 Access Control and Groups

When using groups as subjects for access control it is important to consider where groups are located and how adapter translation will impact them. For each LDAP Proxy Adapter defined, make sure you have the DN Attribute List value defined. This defines the list of attributes that need to be mapped to the virtual directory tree in addition to the entry DN. In the case of an access control that depends on an attribute value containing a DN, the value must be correctly mapped.

To have a group that has members from multiple adapters, for example, from both a Local Store Adapter and an LDAP Adapter, place the group in the Local Store Adapter adapter namespace, that is, the local store of the virtual directory. If the group is placed within an external LDAP directory and it contains members that are not present in that directory, translation may not work as expected because the entries that are not in the external directories namespace have no context.

6.3.4 Oracle Virtual Directory Access Control Components

This section describes Oracle Virtual Directory access control components and contains the following sections:

- Overview
- Access Control Scope
- Access Control Rights
- Attribute Access Control
- Access Control Permissions
- Access Control Subjects

6.3.4.1 Overview

To create an Oracle Virtual Directory ACL you:

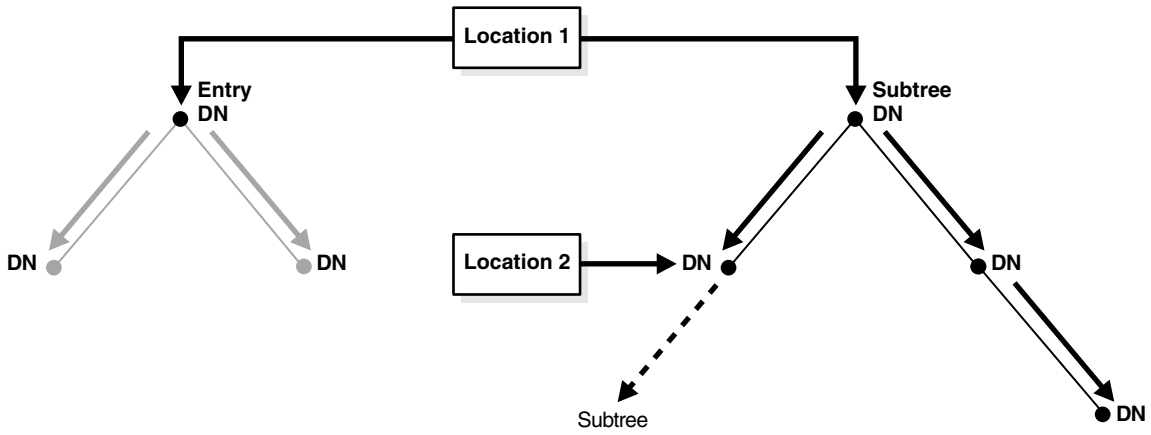
- Configure an Access Control Point, that is, identify the location where the ACL will be applied. Typically the Access Control is a distinguished name, but can also be root to stand for the base of the tree.
- Configure the policy for both Structural Access Items, that is, entire entries in the virtual directory tree, and for Content Access Items, that is, the attributes of the entry.

> **Note:** If an entry being accessed or modified on the Oracle Virtual Directory server does not equal or reside below the ACL Access Control Point, the given ACL will not be evaluated further. If the entry being accessed or modified does equal or reside below the ACL Access Control Point, the ACL scope setting is evaluated.

6.3.4.2 Access Control Scope

Oracle Virtual Directory defines two types of scope for access control: Entry and Subtree. Figure 6–2 illustrates how these scope components operate:

Figure 6–2 Oracle Virtual Directory Access Control Scopes



As shown in Figure 6–2, location and scope are related in that location indicates the position on the DIT where scope is evaluated. In the entry portion of Figure 6–2, the

ACL applies only when the directory entry (DN) being accessed or modified is the same DN indicated by Location 1. In the subtree portion of Figure 6–2, all DNs beginning from Location 1 and moving downward are affected by the ACL with subtree scope. The only endpoint for a subtree scope occurs when, for a given ACL, another ACL declared at a point below the first ACL (Location 2), alters the rules established by the first ACL.

Entry scope is often used with various subjects and deny settings. It is especially useful when a single entry contains more sensitive information than the entries around it or even below it and needs to be kept private. When two scope rules exist which differ only in their scope type, an entry scope will take precedence over a subtree scope.

6.3.4.3 Access Control Rights

There are two Oracle Virtual Directory access rights for each permission: grant and deny. The decision whether to grant or deny a client access to a particular piece of information is based on many factors related to the access control rules and the entry being protected. Throughout the decision making process, the following guiding principle are used:

- **Specificity**: more specific rules override less specific ones, for example, specific client DN in an ACL takes precedence over group reference.
- **Deny**: the default when access control is enabled and there is no access control information granting or denying the operation.
- **Grant**: the default when access controls are disabled in Oracle Virtual Directory.
- **Entry vs. Subtree:** The entry scope takes precedence over the subtree scope, given the subject and attributes have the same specificity.

The following is the order of precedence Oracle Virtual Directory uses in evaluating ACLs which differ only in the type of subject:

1. Specific DN or IP Address
2. This
3. Groups
4. Subtree
5. Public

> **Note:** If two ACLs differ only by their grant/deny property, the resulting permission will be a deny regardless of the order in which the ACLs are added. For example, the following two ACLs will result in a deny for Search(s) and Read(r) of all attributes for public:
>
> ```
> deny:s,r#[all]#public:
> grant:s,r#[all]#public:
> ```

6.3.4.4 Attribute Access Control

The attributes component is strongly linked to the permissions component because it determines whether permissions apply to directory entries as a whole, by selecting Entry, or to some or all of their attributes, by identifying specific attributes.

6.3.4.5 Access Control Permissions

The permissions that apply either to entire entries or to their attributes parallel the type of LDAP operations that can be performed. Each of the LDAP access permissions are discrete: one permission does not imply another permission. When configuring ACLs it is typical to have two ACLs relating to the same location because there is a need to separate permissions for the individual attributes (attribute permissions) from those of the entry itself (entry permissions). Attribute permissions require an attribute component of either * (meaning all attributes), or a list attributes for which the attribute permissions apply. Entry permissions require an attribute component of entry.

Structural Access Permissions

Structural Access permissions are for entire entries in the virtual directory tree. The following is a list and description of each Structural Access Item permission:

- **Add (a)**: permits an entry to be added below the given location. For a client application to add an entry, the make attribute permission must also be granted to add at least the mandatory attributes for each objectclass.
- **Delete (d)**: permits an entry to be deleted from the DIT, regardless of existing attribute permissions within the entry.
- **Rename DN (n)**: permits an entry to be renamed or moved.
- **Browse DN (b)**: permits an entry to be browsed. If granted, permits entries to be accessed using directory operations that do not explicitly provide the name of the entry.
- **Return DN (t)**: permits the entry's DN to be disclosed in the result of the LDAP operation.

Granting Rename DN is necessary for an entry to be renamed with a new Relative DN (RDN). However, it should be taken into account that there will be consequential changes to the distinguished names of its subordinate entries, if any. Renaming an entry does not require any prerequisite permissions of its contained attributes, including the attributes of RDN itself.

Further, note that there are two conditions under which a rename takes place: (1) the rename does not change the name of it's superior DN (thus, a literal renaming of the RDN) and (2) the rename relocates the DN and its subordinates in DIT, thus the entry gets a new superior DN (a move).

For any type of rename to take place, the DN to be moved must have the Rename DN permission; the target DN (under which the new entry will placed) must have both Add and Rename DN permission.

Unless other requirements forbid, it is common to specify entry permissions: Browse DN and Return DN for most locations to allow full retrieval of the data in the directory. Similarly, it is common to grant both write and obliterate permissions for locations that may require data modification.

Content Access Permissions

Content Access permissions are for entry attribute. The following is a list and description of each Content Access Item permission:

- **Read:** permits attribute values to be read in a read or search operation.
- **Write:** permits attribute values to be modified or added in a modify operation.
- **Obliterate:** permits attribute values to be deleted in a modify operation.

- **Search:** permits attributes to be searched for specified values in a search operation.
- **Compare:** permits attributes to be compared in a compare operation.
- **Make/Create:** permits new attributes to be made on a new entry below the identified entry.

The Make/Create attribute permission is required for all attributes in an entry when it is created; it is typically associated with the Add entry permission.

Unless other requirements forbid, it is common to specify attribute permissions: search, read, and compare for most locations to allow full retrieval of the data in the directory. Similarly, it is common to grant both write and obliterate permissions for locations that may require data modification.

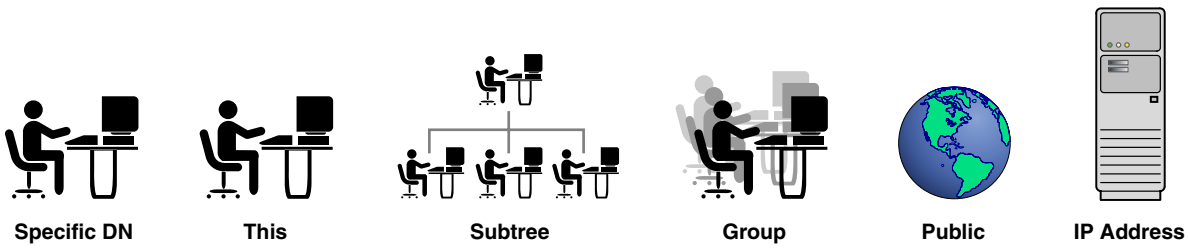
Write and obliterate have no bearing on an add operation; make has no bearing on a modify operation. Since a new entry does not yet exist, the add and make permissions needed to create it must be granted on the new entry's parent. This differs from write and obliterate, which must be granted on the entry being modified. The make permission is distinct and separate from the write and obliterate permissions so that there is no conflict between the permissions needed to add new children to an entry and the permissions needed to modify existing children of the same entry.

To replace attribute values by using the modify operation, the entry must have both write and obliterate permissions enabled for the attribute.

6.3.4.6 Access Control Subjects

The subject of an ACL determines whom it applies to. Oracle Virtual Directory utilizes a variety of possible subject types, which take the client credential, whether user-supplied or from the proxy account (for example, binddn), and apply it under the following conditions:

Figure 6–3 Oracle Virtual Directory ACL Subjects



- **Specific DN:** ACL applies when the client DN credential is compared and matches the value you identified in the ACL settings.
- **This:** ACL applies to the client whose credential matches that of the entry being accessed.
- **Subtree:** ACL applies when the client credential falls in or under the DN specified in the subject.
- **Group:** ACL applies when the DN specified by the subject component is looked up and the client credential is a member of the group determined by one of the three objectclasses: groupOfUniqueNames, groupOfNames, groupOfUniqueURLs. The first two types of groups specify static lists of users: for groupOfUniqueNames one of the `uniquemembers` attribute must match the client credential; for groupOfNames one of the member attributes must match the client credential. The third group type, groupOfUniqueURLs, deals with dynamic

groups and requires the client credential to match at least one of the DNs obtained by performing the search specified by the value of the memberurl attribute.

- **Public:** ACL applies to any client connected to the directory, whether they are authenticated or not.
- **IP Address:** ACL applies when the client IP address is compared and matches the value you identified in the ACL settings.

The following are notes and examples for each of the ACL subjects:

Specific DN

The Specific DN subject is typically used to grant or deny access to a specific client, such as proxy binddn, over part of the DIT. When a user binds anonymously, the proxy credentials are sent to the server and it would not be wise to allow the proxy binddn to view or modify sensitive entries or attributes in the DIT.

Example (if `binddn=cn=service`):

	ACL1	ACL2
location	ou=security, o=oracle.com	ou=security, o=oracle.com
scope	subtree	subtree
rights	deny	deny
permissions	read, search, make, write, obliterate	add, delete, renameDN, browseDN, returnDN
attributes	[all]	[entry]
subject	specificDN=cn=service	specificDN=cn=service

This

The This subject is generally used to permit any given client located somewhere in the DIT the ability to both read and modify their personal information, such as userpassword, postaladdress, telephonenumber, and so on. Another ACL is usually used to deny such access to the public at large

Example:

	ACL1	ACL2	ACL3
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	userpassw	userpassword, postaladdress
subject	public	this	this

Subtree

The Subtree subject is useful when the client credential may be found anywhere within a subtree of the DIT. Typically, this subtree might contain nothing but privileged accounts under closely related branches.

Example:

```
cn=jbowen, ou=eng_admins, ou=admins, ou=security, o=oracle.com
```

An admin with preceding DN needs to authenticate. One possible ACL implementation which would authenticate this admin, as well as other similarly-privileged admins, using subtree would be:

	ACL1	ACL2
location	o=oracle.com	o=oracle.com
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=oracle.com	subtree=ou=admins, ou=security, o=oracle.com

Group

The Group subject is most often chosen for granting specific rights and permissions to differing categories of users because it allows the greatest flexibility in specifying where in the DIT to look for their entries to perform the authentication.

For example, an admin with the credential: `cn=jbowen, ou=admins, ou=security, o=oracle.com` needs to authenticate. One possible ACL and entry combination which would authenticate this admin using the Group subject with objectclass groupOfURLs and attribute memberURL would be:

	ACL1	ACL2
location	o=oracle.com	o=oracle.com
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=oracle.com	group=ou=sales, ou=depts, o=oracle.com

The DN in the subject component which would be examined for objectclasses specifying groups is:

```
dn: ou=sales, ou=depts, o=oracle.com
objectclass: organizationalUnit
objectclass: groupofurls
ou=sales
memberurl:
ldap:///ou=admins,ou=security,o=oracle.com?sub?(objectclass=inetOrgPerson)
```

In the preceding DN, the LDAP search specified by the memberurl attribute would contain at least the following DN, which would then authenticate the client credential:

```
dn: cn=jbowen, ou=admins, ou=security, o=oracle.com
objectclass: inetOrgPerson
cn: jbowen
```

```
userpassword: xyz123
```

Public

An ACL which uses Public, usually with grant, wants to give access to the DIT to the widest potential client base. Generally, this is done to the [root] location with an entry scope and then done specifically to the base of the tree with another ACL with scope entry and an ACL with scope subtree. Usually, other ACLs are also added to restrict specific parts of the tree and specific attributes, such as `userpassword`, from viewing or modification by the public.

For example:

	ACL1	ACL2	ACL3	ACL4
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

IP Address

ACL applies when the client IP address is compared and matches the value you identified in the ACL settings. The IP Address value can be an IP mask.

For example:

	ACL1	ACL2
location	ou=security, o=oracle.com	ou=security, o=oracle.com
scope	subtree	subtree
rights	deny	deny
permissions	read, search, make, write, obliterate	add, delete, renameDN, browseDN, returnDN
attributes	[all]	[entry]
subject	specificIP=192.168.1.*	specificIP=192.168.1.*

6.3.5 Oracle Virtual Directory Access Control List Enforcement

The following is a summary of how Oracle Virtual Directory enforces ACLs:

1. The first ACL related to the attribute being examined is known as the *matched ACL*. The matched ACL assumes the highest priority of enforcement while Oracle Virtual Directory moves down the order and evaluates each and every subsequent ACL.
2. The enforcement of the matched ACL changes only in a few conditions, such as when the scope specified in a subsequent ACL is more specific, that is deeper to the DIT, or when the subject type is changed, for example, from public to this.

3. Oracle Virtual Directory denies access of an ACL if the scope of the matched ACL and a subsequent ACL are the same and either the matched or subsequent ACL denies permission to the attribute being considered.
4. Oracle Virtual Directory sequentially evaluates every ACL listed while descending through the last ACL listed until an enforcement decision is reached after all ACLs are evaluated.

6.4 Understanding Wallet and Certificate Management

In Oracle Virtual Directory 11*g* Release 1 (11.1.1), you can manage wallets and certificates using Oracle Enterprise Manager Fusion Middleware Control. Refer to the *Oracle Fusion Middleware Security Guide* for complete information on managing wallets and certificates for Oracle Virtual Directory.

7

Understanding Oracle Virtual Directory Fault Tolerance

This chapter describes Oracle Virtual Directory fault tolerance and contains the following topics:

7.1 Overview

Oracle Virtual Directory is extremely flexible when implementing fault-tolerant designs. Oracle Virtual Directory does not store data locally allowing duplicate copies of the data to be deployed and managed across multiple Oracle Virtual Directory instances. Additionally, Oracle Virtual Directory configuration files can be easily duplicated or shared on an appropriate Storage Area Network (SAN) configuration.

Oracle Virtual Directory's LDAP Adapter provides excellent support for managing connections to multiple source directory replicas and masters. Oracle Virtual Directory provides the ability to spread query loads across multiple directory replicas while directing add, modify, delete, and rename operations to designated directory master servers.

In a situation where one source directory does not have fault tolerance and the LDAP client application issues a query that spans all directories, LDAP RFCs require that all parts of the directory respond correctly or the entire result is invalid. This generally works well until one of the proxied directories becomes unavailable. If the source without a redundant directory link fails, global queries may begin to failover all directories even though only part of the user base is impacted. Oracle Virtual Directory allows you to control how it will respond when individual proxies fail and how it should impact the overall service.

In many scenarios the proxied directory is present to allow partner company users to access a host company's application. If the partner directory is offline or is unreachable, it is also likely that the company's users cannot get to the application anyway, so a failure could be deemed non-critical to the application. In this case, Oracle Virtual Directory can be configured to ignore the downed server connection, allowing the other partners to continue working.

The following is a list of the primary areas of Oracle Virtual Directory fail over, which are described in the subsequent topics in this chapter:

7.2 DNS and Network Fail Over

Depending on how you plan to implement fault tolerance for the Oracle Virtual Directory, you can consider several options for routing clients to available Oracle Virtual Directory systems.

The simplest method is to define DNS round robin where a particular DNS name has two `IN A records` in DNS management terms which causes a DNS server to give out a rotating address each time a request for a particular address is made (that is, ldap.corp.com alternates between 192.168.0.1 and 192.168.0.2). This approach is useful if you want to spread load between two available servers, but is less useful when one of those servers becomes unavailable because DNS is unaware of the failure and will continue to send clients to the server every time it rotates through the failed server's address.

You can also use a hardware load balancer such as Cisco's LocalDirector or F5's Big-IP. These types of products provide true load balancing while monitoring performance of each of the servers. There are many products that vary in cost and capability in this category.

Another method is to use a cluster configuration (for example, Veritas) capable of switching IP addresses between failed nodes in a cluster.

7.3 Oracle Virtual Directory Fail Over

Fail over Oracle Virtual Directory system fail over is relatively straightforward except when you are using a Local Store Adapter. Oracle Virtual Directory uses configuration files that are only read on start-up. In theory, two servers reading the same configuration data will automatically perform the same function.

7.3.1 Local Store Adapter Fail Over

When using the Local Store Adapter, you must consider a few additional issues, specifically replication. Replication is the process where one node updates the other node with changes to its local data store. If you are using the Local Store Adapter, you must set up a replication agreement between cluster nodes (and possibly other non-cluster nodes). When replication is configured, one node becomes the primary node and the other the node becomes a subordinate node to the primary node. For example, node 1 is the primary and node 2 is the subordinate. In this configuration, both nodes are equally functional, however, only node 1 may process writes. Once processed, node 1 will automatically update node 2.

In the event of a failure, you must configure your cluster failure handling scripts to take appropriate action. If node 2 fails, nothing is impacted as long as replication is restarted on the return of node 2. In contrast, if node 1 fails, node 2 must be promoted to primary, allowing node 2 to continue handling writes during node 1's absence. Before node 1 returns to operational status, the replication agreement will must be reversed.

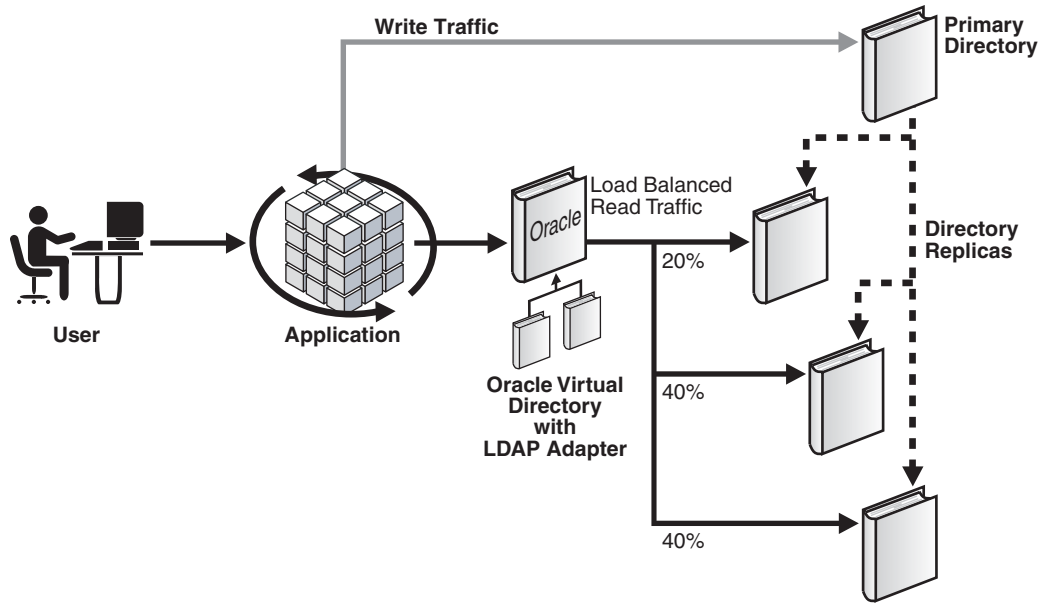
7.4 Proxied Sources Fail Over

Oracle Virtual Directory's LDAP P Adapter provides sophisticated fail over and load balancing management for all LDAP-compliant data repositories. For any proxied source or adapter you can define multiple remote host replicas and specify the following characteristics:

- Read/Write or Read Only (Master vs. Replica node)
- Percentage load distribution or switch only on failure

Figure 7–1 shows an example of how Oracle Virtual Directory's LDAP Adapter performs transaction load-balancing:

Figure 7–1 Oracle Virtual Directory LDAP Adapter and Transaction Load-Balancing



Oracle Virtual Directory includes configurable connection handling settings that allow you to specify the following:

- **Heartbeat Interval**: how often Oracle Virtual Directory will verify online status of a proxy. The heartbeat interval continually verifies availability of a server. If a proxy goes offline, Oracle Virtual Directory automatically removes it from its list of active servers and distributes load to other defined replicas. When the heartbeat interval verifies a server is available again, the server is put back on the available list.
- **Time-out Interval**: how long (in milliseconds) Oracle Virtual Directory will wait before determining a connection has failed. When a connection fails, Oracle Virtual Directory will automatically try the next server on its replica list. If no proxied servers are responding, the LDAP client will receive the DSA unavailable error.
- **Criticality**: how Oracle Virtual Directory determines when the proxy's results are critical to an overall query. If a query requires responses from multiple adapters, Oracle Virtual Directory responds with an error if any sources are unavailable (because all adapters could not be queried) and have been designated critical. In some situations you may wish to have Oracle Virtual Directory return results even if only some servers could be queried. To allow Oracle Virtual Directory to return

partial results, set adapters to non-critical if you are allowing missing results from those adapters.

Part II

Basic Administration

This part presents information about performing basic administration tasks for Oracle Virtual Directory. It contains the following chapters:

8

Getting Started with Administering Oracle Virtual Directory

Oracle Virtual Directory can be administered from both a graphical user interface and a command-line interface. This chapter describes those Oracle Virtual Directory management interfaces, and explains how to start and stop Oracle Virtual Directory.

> **Note:** This chapter assumes you have installed and configured Oracle Virtual Directory as described in: *Oracle Fusion Middleware Quick Installation Guide for Oracle Identity Management*.

This chapter includes the following topics:

- Getting Started After Installing 11g Release 1 (11.1.1)
- Basic Tasks for Configuring and Managing Oracle Virtual Directory
- Getting Started With Oracle Directory Services Manager
- Getting Started With Fusion Middleware Control
- Getting Started with WLST for Oracle Virtual Directory
- LDAP Tools Usage

8.1 Getting Started After Installing 11*g* Release 1 (11.1.1)

After installing 11*g* Release 1 (11.1.1), Oracle recommends:

- Reviewing Appendix A, "Comparing Oracle Virtual Directory 11g Release 1 (11.1.1) and 10g Releases (10.1.4.x)" to understand how fundamental items in Oracle Virtual Directory are implemented in 11*g* Release 1 (11.1.1) compared to legacy Oracle Virtual Directory 10*g* Releases (10.1.4.x).
- Reviewing Appendix B, "Starting and Stopping the Oracle Stack" to understand how to start and stop the components of the Oracle stack in 11*g* Release 1 (11.1.1).
- Reviewing Table 8–1 to understand the default URLs for various interfaces that can be used to manage Oracle Virtual Directory in 11*g* Release 1 (11.1.1):

Table 8–1 Default URLs for Management Interfaces

Interface	Default URL
Oracle Directory Services Manager	http://*host*:7005/odsm/
Fusion Middleware Control	http://*host*:7001/em/

Table 8–1 (Cont.) Default URLs for Management Interfaces

Interface	Default URL
Oracle WebLogic Server Administrative Console	http://*host*:7001/console/

- Reviewing Table 8–2 to understand various default ports for Oracle Virtual Directory in 11*g* Release 1 (11.1.1):

Table 8–2 Default Ports

Port Type	Default Port
LDAP	6501
LDAPS	7501
Admin Port (HTTPS)	8899

- Reviewing Table 8–3 to understand various environment variables for Oracle Virtual Directory 11*g* Release 1 (11.1.1):

Table 8–3 Environment Variables

Variable	Description
ORACLE_HOME	The location of non-writable files in your Oracle Identity Management installation.
ORACLE_INSTANCE	The location of writable files in your Oracle Identity Management installation.
PATH	Add the following directory locations to your PATH: ■ $ORACLE_HOME/bin ■ $ORACLE_HOME/ldap/bin ■ $ORACLE_INSTANCE/bin

8.2 Basic Tasks for Configuring and Managing Oracle Virtual Directory

The following provides an overview of the steps commonly used to configure and manage a basic Oracle Virtual Directory environment:

1. Configure Oracle Virtual Directory server by customizing its settings to be specific to your environment. For more information, refer to:
   - Configuring Oracle Virtual Directory Server Properties Using Fusion Middleware Control on page 9-1
   - Configuring Oracle Virtual Directory Server Settings Using WLST on page 9-7
2. Create and configure adapters for the target data repositories. For more information, refer to:
   - Chapter 2, "Understanding Oracle Virtual Directory Adapters"
   - Chapter 12, "Creating and Configuring Oracle Virtual Directory Adapters"
3. Configure plug-ins for your environment. For more information, refer to:
   - Chapter 4, "Understanding Oracle Virtual Directory Plug-Ins"
   - Chapter 13, "Managing Oracle Virtual Directory Plug-ins"
4. Configure Access Control Lists for Oracle Virtual Directory. For more information, refer to:

- Understanding Oracle Virtual Directory Access Control on page 6-4
- Chapter 16, "Configuring Oracle Virtual Directory Access Control"

8.3 Getting Started With Oracle Directory Services Manager

This topic explains how to set up the Oracle Directory Services Manager interface for use with Oracle Virtual Directory and contains the following sections:

- Understanding Oracle Directory Services Manager
- Invoking Oracle Directory Services Manager
- Logging in to the Directory Server from Oracle Directory Services Manager
- Managing Oracle Directory Services Manager's Key Store
- Configuring Oracle HTTP Server to Support Oracle Directory Services Manager in an Oracle WebLogic Server Cluster

8.3.1 Understanding Oracle Directory Services Manager

Oracle Directory Services Manager is the unified browser-based graphical user interface (GUI) for Oracle Virtual Directory and Oracle Internet Directory. Oracle Directory Services Manager simplifies the administration and configuration of Oracle Virtual Directory and Oracle Internet Directory by allowing you to use web-based forms and templates.

Notes:

- Only the superuser (usually `cn=orcladmin`) can log in to Oracle Directory Services Manager.
- The user name of the superuser used to log in to Oracle Directory Services Manager must be comprised of only ASCII characters. You cannot log in to Oracle Directory Services Manager using a user name of the superuser that contains non-ASCII characters.

Refer to the Oracle Identity Management Certification Information on the Oracle Technology Network web site for information about supported browsers for Oracle Directory Services Manager. You can access the Oracle Technology Network web site at:

`http://www.oracle.com/technology/index.html`

8.3.2 Invoking Oracle Directory Services Manager

You can invoke Oracle Directory Services Manager directly or from Oracle Enterprise Manager Fusion Middleware Control as follows:

- To invoke Oracle Directory Services Manager directly, enter the following URL into your browser's address field.

  `http://`*`host`*`:`*`port`*`/odsm`

  In the URL to access Oracle Directory Services Manager, *host* is the name of the managed server where Oracle Directory Services Manager is running. *port* is the managed server port number from the WebLogic server.

You can determine the exact port number by examining the following file, where *ORACLE_IDENTITY_MANAGEMENT_DOMAIN* represents the root directory of the Oracle WebLogic Server Domain for Oracle Identity Management components:

$*ORACLE_IDENTITY_MANAGEMENT_DOMAIN*/servers/*MANAGED_SERVER_NAME*/data/nodemanager/*MANAGED_SERVER_NAME*.url

- To invoke Oracle Directory Services Manager from Oracle Enterprise Manager Fusion Middleware Control, select one of the options from the **Directory Services Manager** entry in the **Oracle Virtual Directory** menu in the Oracle Virtual Directory target. A new browser window containing the Oracle Directory Services Manager Welcome screen appears.

8.3.3 Logging in to the Directory Server from Oracle Directory Services Manager

When the Oracle Directory Services Manager Welcome screen appears, you can connect to either an Oracle Internet Directory server or a Oracle Virtual Directory server. The following is a list of items to consider regarding logging in to a directory server from Oracle Directory Services Manager:

- The directory server must be running to connect to it from Oracle Directory Services Manager.
- Only the superuser (usually `cn=orcladmin`) can log in to Oracle Directory Services Manager.
- The user name of the superuser used to log in to Oracle Directory Services Manager must be comprised of only ASCII characters. You cannot log in to Oracle Directory Services Manager using a user name of the superuser that contains non-ASCII characters.
- After you have logged in to Oracle Directory Services Manager, you can connect to multiple Oracle Virtual Directory and Oracle Internet Directory components from the same Oracle Directory Services Manager session (that is, the same browser window). However, you should avoid using multiple browser windows of the same browser program to connect to different directories at the same time. Doing so can cause a `Target unreachable` error.
- You can use the same Oracle Directory Services Manager component with different browser programs, such as Internet Explorer and Firefox, and connect each to a different directory system component.
- If you change the browser language setting, you must update the session in order to use the new setting. To do update the session, either reenter the Oracle Directory Services Manager URL in the URL field and press **Enter** or quit and restart the browser.

This section contains the following topics:

8.3.3.1 Logging in to the Directory Server from Oracle Directory Services Manager

You log in to a directory server's non-SSL port from Oracle Directory Services Manager as follows:

1. Click **Connect to a directory** at the top of the Oracle Directory Services Manager Welcome screen. A menu containing the following appears:

- A list of live connections, which are current connections that you can return to.

  Note: To reconnect to a live connection, click it. You will see a short version of the Connect dialog box where you need to enter only a user name and password. To remove a live connection from the list, click it and then click **Remove** on the Connect dialog box.

- A **Create a New Connection** option, which is used to initiate a new connection.

  To initiate a connection to a new directory server, click **Create a New Connection**. The New Connection Dialog appears. Continue the log in process using the following steps.

2. Select **OID** or **OVD**.
3. Optionally, enter an alias name in the Name field to identify the connection. This name will appear in the list of live connections (as described in 1) to enable you to quickly reconnect to it after ending the current Oracle Directory Services Manager session.
4. Enter the name of server where Oracle Internet Directory or Oracle Virtual Directory is running in the Name field.
5. Enter the non-SSL port in the Port field. For Oracle Virtual Directory, enter the non-SSL port for the Admin Listener. For Oracle Internet Directory, enter the non-SSL LDAP port.
6. Deselect **SSL Enabled**.
7. Enter the superuser (usually `cn=orcladmin`) and password.

   Note: The user name of the superuser must be comprised of only ASCII characters.

8. Select the Start Page you want to go to after logging in.
9. Click **Connect**.

After you have logged in to an Oracle Internet Directory or Oracle Virtual Directory server, you can use the navigation tabs to select other pages.

The Oracle Directory Services Manager home pages for Oracle Internet Directory and Oracle Virtual Directory list version information about Oracle Directory Services Manager itself, as well as the directory and adapters. It also lists the existing configured adapters and listeners for Oracle Virtual Directory.

8.3.3.2 Logging in to the Directory Server from Oracle Directory Services Manager Using SSL

When you log in to the server's SSL port, you follow the procedure in "Logging in to the Directory Server from Oracle Directory Services Manager" on page 8-4, except that you specify the SSL port in Step 4 and select **SSL Enabled** in Step 6. Specifically, you enter the SSL port for the Admin Listener for Oracle Virtual Directory, or you enter enter the SSL LDAP port for Oracle Internet Directory. Then, after you click **Connect** in Step 9, you might be presented with a certificate, depending on the type of SSL authentication. The following sections provide information on handling the certificate for each supported SSL authentication type:

- SSL No Authentication
- SSL Server Only Authentication

8.3.3.2.1 SSL No Authentication If the directory server is using SSL No Authentication mode, you will not be presented with a certificate. SSL No Authentication provides data confidentiality and integrity, but no authentication using X509 certificates.

8.3.3.2.2 SSL Server Only Authentication If the directory server is using SSL Server Authentication Only Mode, which is the default for Oracle Virtual Directory, you will be presented with the server's certificate when you click **Connect** in Step 9. After manually verifying the authenticity of the server certificate, you can accept the certificate permanently, accept the certificate for the current session only, or reject the certificate. If you accept the certificate permanently, the certificate is stored in the Oracle Directory Services Manager's Java Key Store (JKS). From then on, you will not be prompted to accept the certificate when you connect to that server using that particular Oracle Directory Services Manager URL. If you accept the certificate only for the current session, you will be prompted to accept or reject the certificate every time you connect to the server. If you reject the certificate, Oracle Directory Services Manager closes the connection to the server.

Refer to "Managing Oracle Directory Services Manager's Key Store" for additional information.

8.3.4 Managing Oracle Directory Services Manager's Key Store

Oracle Directory Services Manager is integrated with the Credential Store Framework, a secure storage framework provided by Oracle. This section explains how to manage Oracle Directory Services Manager's credentials and contains the following topics:

- Understanding Oracle Directory Services Manager's Key Store
- Retrieving Oracle Directory Services Manager's Java Key Store Password
- Listing the Contents of odsm.cer Java Key Store
- Deleting the Trusted Certificate
- Expired Certificates Management

8.3.4.1 Understanding Oracle Directory Services Manager's Key Store

Oracle Directory Services Manager creates a Java Key Store file and assigns a random password to the JKS the first time Oracle Directory Services Manager is used. The JKS file has the name `odsm.cer`. It resides in a directory with a name of the form:

DOMAIN_HOME`/config/fmwconfig/servers/AdminServer/applications/odsm/conf`

Oracle Directory Services Manager stores this random password in the Credential Store Framework. The WebLogic server administrator can retrieve the Java Key Store password stored in the Credential Store Framework. Oracle Directory Services Manager also generates a self-signed certificate for itself and stores it in the Java Key Store.

See Also:

- The *Oracle Fusion Middleware Security Guide* for more information about the Credential Store Framework.
- JavaTM Cryptography Architecture API Specification & Reference, at `http://java.sun.com`
- keytool - Key and Certificate Management Tool, at `http://java.sun.com`

8.3.4.2 Retrieving Oracle Directory Services Manager's Java Key Store Password

To manage Oracle Directory Services Manager's Java Key Store, you must first retrieve Oracle Directory Services Manager's Java Key Store password. The WebLogic administrator can retrieve it using the WLST as follows:

1. Start the WLST shell:

   ```
   ORACLE_COMMON_HOME/common/bin/wlst.sh
   ```

2. Enter `connect()` and provide the username, password, and URL to the Admin Server.
3. Enter the following `listCred()` method to retrieve Oracle Directory Services Manager's Java Key Store password:

   ```
   listCred( map="ODSMMap", key="ODSMKey.Wallet" )
   ```

See Also: The "Managing Credentials with WLST Commands" section in the *Oracle Fusion Middleware Security Guide* for more information.

8.3.4.3 Listing the Contents of odsm.cer Java Key Store

After you retrieve the Java Key Store password, you can manage it using the `keytool` command.

To list contents of odsm.cer:

1. Move (`cd`) to the directory containing the odsm.cer, for example:

   ```
   cd DOMAIN/config/fmwconfig/servers/AdminServer/applications/odsm/conf
   ```

2. Use `keytool` to list the contents of odsm.cer, for example:

   ```
   ORACLE_HOME/jdk/jre/bin/keytool -list -keystore odsm.cer \
   -storepass "&M)S86)/RB" -v

   Keystore type: JKS
   Keystore provider: SUN

   Your keystore contains 2 entries

   Alias name: serverselfsigned
   Creation date: Dec 26, 2008
   Entry type: PrivateKeyEntry
   Certificate chain length: 1
   Certificate[1]:
   Owner: CN=OVD, OU=Development, O=Oracle, L=Redwood Shores, ST=California, C=US
   Issuer: CN=OVD, OU=Development, O=Oracle, L=Redwood Shores, ST=California, C=US
   Serial number: 495586b6
   Valid from: Fri Dec 26 17:36:54 PST 2008 until: Wed Jun 24 18:36:54 PDT 2009
   ```

```
Certificate fingerprints:
         MD5:  6C:11:16:F3:88:8D:18:67:35:1E:16:5B:3E:03:8A:93
         SHA1: F4:91:39:AE:8B:AC:46:B8:5D:CB:D9:A4:65:BE:D2:75:08:17:DF:D0
         Signature algorithm name: SHA1withRSA
         Version: 3


*******************************************
*******************************************

Alias name: cn=rootca, o=oracle, c=us (0)
Creation date: Dec 31, 2008
Entry type: trustedCertEntry

Owner: CN=RootCA, O=Oracle, C=US
Issuer: CN=RootCA, O=Oracle, C=US
Serial number: 0
Valid from: Tue Dec 30 02:33:11 PST 2008 until: Mon Jan 24 02:33:11 PST 2050
Certificate fingerprints:
         MD5:  72:31:7B:24:C9:72:E3:90:37:38:68:40:79:D1:0B:4B
         SHA1: D2:17:84:1E:19:23:02:05:61:42:A9:F4:16:C8:93:84:E8:20:02:FF
         Signature algorithm name: MD5withRSA
         Version: 1


*******************************************
*******************************************
```

8.3.4.4 Deleting the Trusted Certificate

To delete trusted certificates in odsm.cer:

1. Move (`cd`) to the directory containing the odsm.cer, for example:

   ```
   cd DOMAIN_HOME/config/fmwconfig/servers/AdminServer/
   ```

2. Use `keytool` to delete the contents of odsm.cer, for example:

   ```
   ORACLE_HOME/jdk/jre/bin/keytool -delete -keystore odsm.cer \
   -storepass PASSWORD_OBTAINED_FROM_CSF -alias "cn=rootca, o=oracle, c=us (0)"
   [Storing odsm.cer]
   ```

8.3.4.5 Expired Certificates Management

Oracle Directory Services Manager does not provide a web-based user interface for managing expired certificates. You must use `keytool` to find expired certificates and delete them. To find expired certificates, you must list the content of odsm.cer as described in Listing the Contents of odsm.cer Java Key Store. The keytool lists the validity of certificates, enabling you to find all expired certificates. Delete the expired certificates as described in Deleting the Trusted Certificate.

8.3.5 Configuring Oracle HTTP Server to Support Oracle Directory Services Manager in an Oracle WebLogic Server Cluster

Perform the following steps to configure Oracle HTTP Server to route Oracle Directory Services Manager requests to multiple Oracle WebLogic Servers in a clustered Oracle WebLogic Server environment:

1. Create a backup copy of the Oracle HTTP Server's httpd.conf file. The backup copy will provide a source to revert back to if you encounter problems after performing this procedure.
2. Add the following text to the end of the Oracle HTTP Server's httpd.conf file and replace the variable placeholder values with the host names and managed server port numbers specific to your environment. Be sure to use the `<Location /odsm/ >` as the first line in the entry. Using `<Location /odsm/faces >` or `<Location /odsm/faces/odsm.jspx >` can distort the appearance of the Oracle Directory Services Manager interface.

```
<Location /odsm/ >
SetHandler weblogic-handler
WebLogicCluster host-name-1:managed-server-port,host-name_2:managed_server_port
</Location>
```

3. Stop, then start the Oracle HTTP Server to activate the configuration change.

Note: Oracle Directory Services Manager loses its connection and displays a session time-out message if the Oracle WebLogic Server in the cluster that it is connected to fails. Oracle Directory Services Manager requests will be routed to the secondary Oracle WebLogic Server in the cluster that you identified in the httpd.conf file after you log back in to Oracle Directory Services Manager.

8.4 Getting Started With Fusion Middleware Control

This topic explains how to get started using Oracle Enterprise Manager Fusion Middleware Control with Oracle Virtual Directory. It contains the following sections:

- Invoking Fusion Middleware Control to Manage Oracle Virtual Directory
- Starting the Oracle Virtual Directory Server Using Fusion Middleware Control
- Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control
- Restarting the Oracle Virtual Directory Server Using Fusion Middleware Control
- Monitoring Oracle Virtual Directory Using Fusion Middleware Control Metrics

Note: If Oracle Virtual Directory is configured to listen on privileged ports, ensure OPMN was started as the superuser (root) before starting, stopping, or restarting Oracle Virtual Directory using Oracle Enterprise Manager Fusion Middleware Control as described in this topic. Refer to Chapter 10, "Managing Oracle Virtual Directory Server Processes" for more information.

8.4.1 Invoking Fusion Middleware Control to Manage Oracle Virtual Directory

Oracle Enterprise Manager Fusion Middleware Control is a graphical user interface that provides a comprehensive systems management platform for Oracle Fusion Middleware. Oracle Enterprise Manager Fusion Middleware Control organizes a wide variety of performance data and administrative functions into distinct, Web-based home pages for the farm, Oracle Fusion Middleware components, middleware system components, and applications.

Oracle Virtual Directory is a target type in Oracle Enterprise Manager Fusion Middleware Control. To use the Oracle Enterprise Manager Fusion Middleware Control interface to manage Oracle Virtual Directory:

1. Connect to Oracle Enterprise Manager Fusion Middleware Control using a web browser. The URL is of the form:

   `https://host:port/em`

2. In the left panel topology tree, expand the farm, then Identity and Access. Alternatively, from the farm home page, expand Fusion Middleware, then Identity and Access. Oracle Virtual Directory components are listed in both places.

   To distinguish one component from another, move the mouse over the component name and view the full name of the component in the tool tip.

3. Select the Oracle Virtual Directory component you want to manage.
4. Use the Oracle Virtual Directory menu to select tasks.

You can use the Oracle Virtual Directory menu to navigate to other Oracle Enterprise Manager Fusion Middleware Control pages for Oracle Virtual Directory and to navigate to Oracle Directory Services Manager pages for Oracle Virtual Directory.

See Also:

- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Concepts*

8.4.2 Starting the Oracle Virtual Directory Server Using Fusion Middleware Control

Perform the following steps to start an Oracle Virtual Directory server that is not running using Oracle Enterprise Manager Fusion Middleware Control. To restart an Oracle Virtual Directory server that is running, refer to Restarting the Oracle Virtual Directory Server Using Fusion Middleware Control.

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target you want to start.
2. Select **Control** from the Oracle Virtual Directory menu and then select **Start Up**. A dialog box appears listing messages and the status of the target.
3. Click **OK** on the message dialog box to close it.

8.4.3 Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control

Perform the following steps to stop a running Oracle Virtual Directory server using Oracle Enterprise Manager Fusion Middleware Control:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target you want to stop.
2. Select **Control** from the Oracle Virtual Directory menu and then select **Shut Down**. A confirmation dialog box appears asking you to confirm that you want to stop the Oracle Virtual Directory server.
3. Click **Yes** on the dialog box to stop the Oracle Virtual Directory server. A dialog box appears listing messages and the status of the target.
4. Click **OK** on the message dialog box to close it.

8.4.4 Restarting the Oracle Virtual Directory Server Using Fusion Middleware Control

Perform the following steps to restart an Oracle Virtual Directory server that is currently running using Oracle Enterprise Manager Fusion Middleware Control.

> **Note:** Restarting an Oracle Virtual Directory that is running reloads all the server configurations from the file system. Restarting an Oracle Virtual Directory that is running *will not stop* the server process.

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Control Directory target you want to restart.
2. Select **Control** from the Oracle Virtual Directory menu and then select **Restart**. A confirmation dialog box appears asking you to confirm that you want to restart the Oracle Virtual Directory server.
3. Click **Yes** on the dialog box to restart the Oracle Virtual Directory server. A dialog box appears listing messages and the status of the target.
4. Click **OK** on the message dialog box to close it.

8.4.5 Monitoring Oracle Virtual Directory Using Fusion Middleware Control Metrics

You can use Oracle Enterprise Manager Fusion Middleware Control to view multiple types of metrics for the Oracle Virtual Directory server. The Oracle Virtual Directory server must be running to view its metrics using Oracle Enterprise Manager Fusion Middleware Control. You can access the metrics from the following locations in Oracle Enterprise Manager Fusion Middleware Control:

- The Oracle Virtual Directory Home page
- The Oracle Virtual Directory Performance Summary page

Home Page

To view the metrics on the Oracle Directory Home page, log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target that you want to view metrics for. The Home page appears displaying the statistics.

Table 8–4 lists the statistics that are available on the Oracle Virtual Directory Home page:

Table 8–4 Metrics Available on the Home Page

Subject	Metric
Current Load	■ Open Connections: number of clients connected to Oracle Virtual Directory server. ■ Distinct Connected Users: number of unique users connected to Oracle Virtual Directory server. ■ Distinct Connected IP Addresses: number of unique IP addresses connected to Oracle Virtual Directory server.
Resource Usage	■ Percent of CPU being utilized on the Oracle Virtual Directory host ■ Percent of memory being utilized on the Oracle Virtual Directory host

Table 8–4 (Cont.) Metrics Available on the Home Page

Subject	Metric
Average Response Time and Operations	■ Average time to complete an LDAP search request. ■ Number of LDAP search requests.
Listeners	Displays a table of configured Oracle Virtual Directory Listeners, including: ■ Listener name ■ Whether the Listener is enabled or disabled ■ Listener type ■ Port the listener listens on
Adapters	Displays a table of configured Oracle Virtual Directory Adapters, including: ■ Adapter name ■ Whether the adapter is enabled or disabled ■ Adapter type ■ Number of searches performed by the adapter ■ Total number of operations performed by the adapter

Performance Summary Page

The Performance Summary page allows you to choose a variety of metrics to display in a time based context. You can customize the metrics displayed on the Performance Summary page using the Metric Palette. Refer to the *Oracle Fusion Middleware Administrator's Guide* for more information on using the Metric Palette.

To view the metrics on the Performance Summary page:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target that you want to view metrics for.
2. Select **Monitoring** and then **Performance Summary** from the Oracle Virtual Directory menu. The Performance Summary page appears.

Refer to Table D–1 on page D-6 for a list and description of the metrics that are available on the Performance Summary page.

8.5 Getting Started with WLST for Oracle Virtual Directory

You can use the WebLogic Scripting Tool (WLST) as the interface to perform several Oracle Virtual Directory administration and management tasks. While there are several tasks and procedures in this document that explain how to use WLST, you should refer to the following documents for complete information:

- The *Oracle Fusion Middleware Oracle WebLogic Scripting Tool* for information on how to use the WLST command-line tool.
- The *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* for information on syntax for the WLST command-line tool.

Important: After you install Oracle Virtual Directory or after you restart the Oracle WebLogic Server, you must execute the WLST `load()` method *before* you execute any other WLST command.

Additionally, Oracle recommends executing the WLST `load()` method before executing any WLST command on the Oracle Virtual Directory MBean. Executing the `load()` method refreshes the MBean to the current configuration.

8.6 LDAP Tools Usage

The LDAP tools (`ldapadd`, `ldapdelete`, `ldapbind`, and so on) for Oracle Virtual Directory have been modified to prevent exposing passwords. Use the `-q` option instead of the `-w` option for user passwords, and use the `-Q` option instead of the `-P` option for wallet passwords. Commands will prompt you for the password when you use the `-q` and `-Q` options.

You can disable the `-w` and `-P` password options by setting the `LDAP_PASSWORD_PROMPTONLY` environment variable to `TRUE` or `1`. Set this environment variable whenever possible.

9

Configuring and Managing the Oracle Virtual Directory Server

This chapter explains how to configure Oracle Virtual Directory server settings and includes the following topics:

- Configuring Oracle Virtual Directory Server Properties Using Fusion Middleware Control
- Configuring Oracle Virtual Directory Server Settings Using Oracle Directory Services Manager
- Configuring Oracle Virtual Directory Server Settings Using WLST
- Controlling the Maximum Heap Size Allocated to the Oracle Virtual Directory Server
- Controlling Orphan Connections Caused by Remote Client or Server Failure
- Managing Oracle Virtual Directory Libraries Using Oracle Directory Services Manager
- Copying Configuration Files Between Oracle Virtual Directory Servers Using syncovdconfig

9.1 Configuring Oracle Virtual Directory Server Properties Using Fusion Middleware Control

Oracle Virtual Directory provides the ability to regulate items such as the number of entries the server can return for an anonymous user or for an authenticated user. You can also limit inbound transaction traffic, which can be used to protect proxied sources from Denial Of Service attacks or to limit LDAP traffic to control access to a limited directory infrastructure resource. You can configure these properties and others on the Oracle Virtual Directory Server Properties page in Oracle Enterprise Manager Fusion Middleware Control.

There are two tabs on the Server Properties screen: **General** and **Change SuperUser Password**. The General tab contains options to configure general server properties, such as quotas on activity limits, search settings, and schema and access control checks. The Change SuperUser Password tab allows you to change the password for the Oracle Virtual Directory superuser.

The following are the procedures to configure the properties on each tab:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target that you want to configure server settings on.

2. Select **Administration** and then **Server Properties** from the Oracle Virtual Directory menu. The Server Properties screen appears.

To configure general Oracle Virtual Directory server properties:

1. Click the **General** tab on the Server Properties screen.
2. Enable quota enforcement on the server by selecting the **Enable Quota Enforcement** option and entering the following information:

   Note: You must select the **Enable Quota Enforcement** option to configure the Activity Limits parameters.

   - Enter the maximum number of client connections to allow in the Maximum Client Connections field.
   - Enter the maximum number of operations to allow for each connection in the Maximum Operations per Connection field.
   - Enter the maximum number of connections to allow for each authenticated subject in the Maximum Connections per Authenticated Subject field.
   - Enter the maximum number of connections to allow for each IP address connected to Oracle Virtual Directory in the Maximum Connections per IP Address field.
   - Enter the maximum length of time (in minutes) that a client connection can remain inactive before Oracle Virtual Directory closes the connection in the Maximum time period (minutes) field.
   - Add or delete IP addresses that are exempt from the quota checking in the Exempt IP addresses field. To add an IP address, enter the IP address in the Exempt IP Addresses field. To delete an IP address, select the IP address in the Exempt IP Addresses field and delete it.

     Note: Oracle Virtual Directory 11*g* Release 1 (11.1.1) supports IPv6. If your network supports IPv6 you can use literal IPv6 addresses in the Exempt IP Addresses field to identify IP addresses that are exempt from quota enforcement.

   - Add or delete subjects that are exempt from the quota checking in the Exempt Subjects field. To add a subject, enter the subject in the Exempt IP Subjects field. To delete a subject, select the subject in the Exempt IP Subjects field and delete it.

     Note: By default, the superuser (typically cn=orcladmin) is exempt from quota checking.

3. Enter the maximum number of entries to return for an anonymous client search in the Anonymous Search field. The default setting is 1000.
4. Enter the maximum number of entries to return for an authenticated user in the Authenticated User Search field. An authenticated user is defined as a user bound to Oracle Virtual Directory. The Oracle Virtual Directory root account is exempt from this quota and the default setting is 10,000.

5. Select the **Enable Access Control Check** option to enable Oracle Virtual Directory to enforce access controls as defined in the access control file.
6. Select the **Enable Persistent Search** option to enable Oracle Virtual Directory to support the persistent search control regardless of the adapters configured.
7. Select the **Enable Schema Check** option to enable Oracle Virtual Directory to check LDAP entries for conformance against the schema definitions contained in the files listed in the Schema Locations field.

   Oracle suggests disabling the **Enable Schema Check** option only when an external method for schema checking will be used.
8. If the **Enable Schema Check** option is selected, Oracle Virtual Directory uses the files that are listed in the Schema Locations field to verify that LDAP entries conform to schema definitions. Use this field to identify the files Oracle Virtual Directory uses to define its schema.

   Each file is applied in descending order from top to bottom, with each file overriding the previous one when conflicts occur. Typically, the last file identified is schema.user.xml. Any and all changes to schema are applied to the schema.user.xml file to ensure standard files, such as schema.core.xml, remain unchanged between releases, but can also be virtually modified by having the changes in schema.user.xml override default-shipped schema in schema.core.xml.

   If you are installing a manufacturer supplied schema (in DSML form), identify this file in the second to last file in the list of schema files. This will protect the distributed manufacturer file from modification while allowing local customization, which is then stored in schema.user.xml.

   The following is a list of the default schema files:

   - schema.core.xml
   - schema.cosine.xml
   - schema.inetorgperson.xml
   - schema.nis.xml
   - schema.dyngroup.xml
   - schema.java.xml
   - schema.diameter.xml
   - schema.eus.xml
   - schema.user.xml
9. Use the TLS Configuration section to:
   - Read the names of the adapter keystore and truststore. You cannot configure these values using Oracle Enterprise Manager Fusion Middleware Control.
   - Set the password for the adapter keystore and truststore.
10. Click **Apply** on the Server Configuration screen to apply your settings.

To change the password for the Oracle Virtual Directory superuser:

1. Click the **Change SuperUser Password** tab on the Server Properties screen.
2. Enter the existing superuser password in the Old Password field.
3. Enter the new superuser password in the New Password field.

4. Reenter the new superuser password in the Confirm Password field.
5. Click **Apply**.

9.2 Configuring Oracle Virtual Directory Server Settings Using Oracle Directory Services Manager

You can use Oracle Directory Services Manager to configure some Oracle Virtual Directory server settings, including settings related to the following:

- Schema files
- Access Control
- Server search limits
- Server activity limits
- Adapter SSL settings

Perform the following steps to configure Oracle Virtual Directory server settings using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Click the **Server Settings** entry in the Advanced navigation tree. The Server Settings entry expands and the Settings, Quotas, and Adapter SSL Settings groups appear in the navigation tree.
4. Click the group you want to configure. The following tables describe each setting in each group.

Note: After configuring the appropriate setting, click **Apply** in the main Oracle Directory Services Manager screen to save the settings to the Oracle Virtual Directory server.

Table 9–1 Configuration Parameters for Settings Group in ODSM

Category	Setting	Description
Schema	Schema Files	Use the Schema Files section to identify the files Oracle Virtual Directory uses to define its schema. The Available Files field lists all available schema files that contain schema definitions. The Selected Files field lists the files that Oracle Virtual Directory uses to verify that LDAP entries conform to schema definitions. Oracle Virtual Directory verifies LDAP entries against the files listed in the Selected Files field *only* when the **Enable Schema Checking** option is selected. To move files between the Available Files and Selected Files fields, select one or more files, then use the appropriate **Move** or **Remove** arrow buttons to move the file. Oracle Virtual Directory verifies LDAP entries against the files in the Selected Files field in the sequence, or order, in which they appear in the field. Each file is used for verification in descending order from top to bottom, with each file overriding the previous one when conflicts occur. You can change the sequence, or order, in which the files will be used for verification by selecting a file name in the Selected Files field and then using the up and down arrow buttons to the right of the Selected Files field to change the order. Typically, the last file identified is schema.user.xml. Any and all changes to schema are applied to the schema.user.xml file to ensure standard files, such as schema.core.xml, remain unchanged between releases, but can also be virtually modified by having the changes in schema.user.xml override default-shipped schema in schema.core.xml. If you are installing a manufacturer supplied schema (in DSML form), identify this file in the second to last file in the list of schema files. This will protect the distributed manufacturer file from modification while allowing local customization, which is then stored in schema.user.xml. The following is a list of the default schema files: ■ schema.core.xml ■ schema.cosine.xml ■ schema.inetorgperson.xml ■ schema.nis.xml ■ schema.dyngroup.xml ■ schema.java.xml ■ schema.diameter.xml ■ schema.eus.xml ■ schema.user.xml
	Enable Schema Checking	Select the **Enable Schema Check** option to enable Oracle Virtual Directory to check LDAP entries for conformance against the schema definitions contained in the files listed in the Schema Files section. Oracle suggests disabling the **Enable Schema Check** option only when an external method of schema checking will be used.

Table 9–1 (Cont.) Configuration Parameters for Settings Group in ODSM

Category	Setting	Description
Access Control	Enable Access Control	Select the **Enable Access Control** option to enable Oracle Virtual Directory to enforce access controls as defined in the access control file.
	Access Control File	Identify the file that stores Oracle Virtual Directory's Access Control Lists (ACL).
Server Root Adapter	Root DN	Allows you to relocate the Oracle Virtual Directory Root DSE entry (base="") to another location in the virtual directory tree. Relocating the DSE is most commonly performed when you need to proxy another server's root entry to replace Oracle Virtual Directory's root entry, usually when you want to make Oracle Virtual Directory appear to be another directory server. This can be useful when the application is making assumptions about the directory. After Oracle Virtual Directory's root entry is renamed from "", you can replace it by creating an LDAP Adapter with a remote base of "" and setting the local root as "". If you do this, you should also set Routing **Levels** to 0 for the LDAP Adapter so that Oracle Virtual Directory only tries to query the Root Entry of the remote server specifically when its root is queried. If you do not set Routing **Levels** to 0, the remote server will receive queries for all requests received by Oracle Virtual Directory.
Control	Persistent Search Control	Enables or disables Oracle Virtual Directory to support the persistent search control regardless of the adapters configured.

Table 9–2 Configuration Parameters for Quotas Group in ODSM

Category	Setting	Description
Search Limits	Anonymous	Enter the maximum number of entries to return for an anonymous client search. The default setting is 1000.
	Authenticated	Enter the maximum number of entries to return for an authenticated user. An authenticated user is defined as a user bound to Oracle Virtual Directory. The Oracle Virtual Directory root account is exempt from this quota and the default setting is 10,000.

Table 9–2 (Cont.) Configuration Parameters for Quotas Group in ODSM

Category	Setting	Description
Activity Limits	Enforce Quotas	Enables or disables quota enforcement on the Oracle Virtual Directory server. You must enable the **Enforce Quota** option to configure the Activity Limits parameters.
	Rate	Determines the time durations (in milliseconds) of quota enforcement. For example, if you set Rate to 50000, the quotas will be enforced for 50 seconds. After 50 seconds expires, the "count" of quota settings starts over at 0 and the quotas will be enforced for another 50 second duration. The default value is 30000, or 30 seconds.
	Max Connections	Enter the maximum number of client connections to allow.
	Max Ops/Con	Enter the maximum number of operations to allow for each connection.
	Max Cons/Subject	Enter the maximum number of connections to allow for each authenticated subject.
	Max Cons/IP Address	Enter the maximum number of connections to allow for each IP address connected to Oracle Virtual Directory.
	Inactive Connection Timeout	Enter the maximum length of time (in minutes) that a client connection can remain inactive before Oracle Virtual Directory closes the connection.
	Exempt Subjects	Add or delete subjects that are exempt from the quota enforcement. By default, the superuser (typically cn=orcladmin) is exempt from quota enforcement.
	Exempt IP Address	Add or delete IP addresses that are exempt from the quota enforcement.

Table 9–3 Configuration Parameters for Adapter SSL Settings Group in ODSM

Setting	Description
Keystore	Lists the names and locations of existing SSL keystores.
Keystore Password	Password for the keystore selected in the Keystore list.
Trust Store	Lists the names and locations of existing SSL trust stores.
Trust Store Password	Password for the trust store selected in the Trust Store list.
Adapters Key Alias	Lists the existing Java certificate aliases. Select an alias from the list to see its certificate details in the Selected Certificate Details table. This Adapter Key Alias control is for informational purposes only—it does not write any data.
Selected Certificate Details	Displays details about the Java certificate for the alias identified in the Adapter Key Alias list.

9.3 Configuring Oracle Virtual Directory Server Settings Using WLST

You can use the WebLogic Scripting Tool (WLST) at *ORACLE_COMMON_HOME*/common/bin/wlst.sh to set Oracle Virtual Directory server settings as follows:

1. Connect to the WebLogic Admin Server. For example:

```
connect('username', 'password','t3://host_name:Admin_Server_Port')
```

2. Move to the Oracle Virtual Directory Root Proxy MBean node and initialize the MBean. For example:

```
custom()
cd('oracle.as.management.mbeans.register')
cd('oracle.as.management.mbeans.register:type=component,name=OVD_COMPONENT_
NAME,instance=INSTANCE_NAME')
invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
g))
```

3. Move to the Oracle Virtual Directory Server configuration MBean. For example:

```
cd('../..')
cd('oracle.as.ovd/oracle.as.ovd:type=component.serverconfig,name=serverconfig,i
nstance=INSTANCE_NAME,component=OVD_COMPONENT_NAME')
```

4. Using the WLST ls() command, you can see a list of attributes for the Oracle Virtual Directory server configuration MBean. Use the `get('ATTRIBUTE_NAME')` command to retrieve the current value for an attribute. For example, to retrieve the current value for MaxConnections, which is the maximum number of client connections to allow, execute the following:

```
get('MaxConnections')
```

Use the set() command to update an attribute. For example, to update the value for the MaxConnections setting, execute the following:

```
set('MaxConnections', 3000)
```

Note: Using the set() command as shown in the preceding example saves the attribute setting to the MBean—you must perform step 5 in this procedure to save the changes to the Oracle Virtual Directory server.

The following is a list of each Oracle Virtual Directory server configuration MBean attribute and an example command for setting them:

- ACLCheck: `set('ACLCheck',true)`
- Anonymous: `set('Anonymous',2000)`
- Authenticated: `set('Authenticated',20000)`
- DoSActive: `set('DoSActive',true)`
- DoSRatePeriod: `set('DoSRatePeriod',20000)`
- ExemptIPAddresses:

  First (on one command-line):

  ```
  invoke('addExemptIPAddress',jarray.array([java.lang.String('127.0.0.1')],
  java.lang.Object),jarray.array(['java.lang.String'],java.lang.String))
  ```

  Then (on one command-line):

  ```
  invoke('deleteExemptIPAddress',jarray.array([java.lang.String('127.0.0.1')]
  ,java.lang.Object),jarray.array(['java.lang.String'],java.lang.String))
  ```

- ExemptSubjects:

  First (on one command-line):

  ```
  invoke('addExemptSubjects',jarray.array([java.lang.String('cn=myuser')],
  java.lang.Object),jarray.array(['java.lang.String'],java.lang.String))
  ```

  Then (on one command-line):

  ```
  invoke('deleteExemptSubjects',jarray.array([java.lang.String('cn=myuser')],
  java.lang.Object),jarray.array(['java.lang.String'],java.lang.String))
  ```

- InactiveConnectionTimeout: `set('InactiveConnectionTimeout',50)`
- MaxConnections: `set('MaxConnections',50)`
- MaxConnectionsPerIP: `set('MaxConnectionsPerIP',20)`
- MaxConnectionsPerSubject: `set('MaxConnectionsPerSubject',20)`
- MaxOperationsPerConnection:

  ```
  set('MaxOperationsPerConnection',10)
  ```

- PersistentSearch: `set('PersistentSearch',false)`
- TLSKeyStore: *Read-only attribute*
- TLSTrustStore: *Read-only attribute*
- TLSKeyStorePassword:

  ```
  set('TLSKeyStorePassword',java.lang.String('PASSWORD').toCharArray())
  ```

- TLSTrustStorePassword:

  ```
  set('TLSTrustStorePassword',java.lang.String('welcome1').toCharArray())
  ```

- SchemaCheck: `set('SchemaCheck',true)`
- SchemaLocations:

  Add (on one command-line):

  ```
  invoke('addSchemaLocation',jarray.array([java.lang.String('schema.myschema.
  xml')],java.lang.Object),jarray.array(['java.lang.String'],java.lang.String
  ))
  ```

  Delete (on one command-line):

  ```
  invoke('deleteSchemaLocation',jarray.array([java.lang.String('schema.mysche
  ma.xml')],java.lang.Object),jarray.array(['java.lang.String'],java.lang.Str
  ing))
  ```

5. Save the changes to the Oracle Virtual Directory server and then refresh the MBean. For example:

   ```
   cd('../..')
   cd('oracle.as.management.mbeans.register')
   cd('oracle.as.management.mbeans.register:type=component,name=OVD_COMPONENT_
   NAME,instance=asinst1')
   invoke('save',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
   g))
   invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
   g))
   ```

9.4 Controlling the Maximum Heap Size Allocated to the Oracle Virtual Directory Server

The `-Xmx` parameter in the opmn.xml file controls the maximum heap size allocated to the Oracle Virtual Directory server. The default value is `-Xmx512m`. Edit this parameter as needed to increase or decrease the maximum heap size allocated to the Oracle Virtual Directory server. The opmn.xml file is located in the *ORACLE_INSTANCE*/config/OPMN/opmn/ directory.

The following example shows the `-Xmx` parameter set to `-Xmx2048m`, which allocates 2 GB of heap size to the Oracle Virtual Directory Server:

```
<ias-component id="OVD_COMPONENT_NAME">
            <process-type id="OVD" module-id="OVD">
               <module-data>
                  <category id="start-options">
           <data id="java-options" value="-server -Xms512m -Xmx2048m
-Doracle.security.jps.config=$ORACLE_INSTANCE/config/JPS/jps-config-jse.xml
-Dvde.soTimeoutBackend=120"/>
                     <data id="java-classpath" value="$ORACLE_
HOME/ovd/jlib/vde.jar$:$ORACLE_HOME/jdbc/lib/ojdbc6.jar"/>
                  </category>
               </module-data>
               <stop timeout="120"/>
            </process-type>
         </ias-component>
```

9.5 Controlling Orphan Connections Caused by Remote Client or Server Failure

Oracle Virtual Directory supports two parameters that help detect and safely close orphan socket connections caused by remote client or server failure. These parameters will help if applications or directory sources are on different networks—in particular, outside of the same data-center—than Oracle Virtual Directory and the network is unstable.

Set each parameter to the amount of time in seconds that TCP should wait for a response from the client or server. The status and stability of your network will influence which parameters you set and also the amount of time you set. In an unstable network, you may want to set these parameters to a greater number of seconds than you would in a stable network environment.

> **Note:** If your operating system is reporting several connections in TIME_WAIT status and they do not close for an extended length of time, such as, five minutes or more, it is a good indication to use these parameters to control the orphan connections.

Controlling Orphan Client Connections:

The LDAP Listener's `SocketOptionsReadTimeout` parameter can be used to control orphan client connections. Use the WLST `set()` command to set the `SocketOptionsReadTimeout` parameter. For example:

```
set('SocketOptionsReadTimeout', 120)
```

Refer to "Updating Listener Settings" on page 11-11 for the complete procedure on updating Listener settings using WLST.

> **Note:** You can also use Oracle Enterprise Manager Fusion Middleware Control to set this parameter for the LDAP Listener. Refer to the Read Timeout parameter described in "Creating LDAP Listeners" on page 11-4 for more information.

Controlling Orphan Server Connections:

The `vde.soTimeoutBackend` Java Virtual Machine parameter located in the *ORACLE_INSTANCE*/config/OPMN/opmn/opmn.xml file can be used to control orphan server connections.

To set the `vde.soTimeoutBackend` parameter, edit opmn.xml and then restart Oracle Virtual Directory. The following is an example of the `vde.soTimeoutBackend` parameter set in the opmn.xml file:

```
<ias-component id="OVD_COMPONENT_NAME">
            <process-type id="OVD" module-id="OVD">
               <module-data>
                  <category id="start-options">
              <data id="java-options" value="-server -Xms512m -Xmx512m
-Doracle.security.jps.config=$ORACLE_INSTANCE/config/JPS/jps-config-jse.xml
-Dvde.soTimeoutBackend=120"/>
                     <data id="java-classpath" value="$ORACLE_
HOME/ovd/jlib/vde.jar$:$ORACLE_HOME/jdbc/lib/ojdbc6.jar"/>
                  </category>
               </module-data>
               <stop timeout="120"/>
            </process-type>
         </ias-component>
```

9.6 Managing Oracle Virtual Directory Libraries Using Oracle Directory Services Manager

This topic describes how to manage libraries used for Oracle Virtual Directory plug-ins and Join View Adapters. It contains the following sections:

- Viewing Oracle Virtual Directory Server Libraries
- Loading Libraries into the Oracle Virtual Directory Server

9.6.1 Viewing Oracle Virtual Directory Server Libraries

Perform the following steps to view the libraries, including plug-ins and Join View adapters, that reside on the Oracle Virtual Directory server:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Libraries** entry in the Advanced tree. A list of the library files that reside on the Oracle Virtual Directory server appears in the Libraries entry of the Advanced tree.

9.6.2 Loading Libraries into the Oracle Virtual Directory Server

Perform the following steps to load libraries into Oracle Virtual Directory using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Libraries** entry in the Advanced tree.
4. Click the **Upload New Library** button at the top of the Advanced tree. The Upload New Library dialog box appears.
5. Enter the path to the library you want to load into Oracle Virtual Directory or click **Browse**, navigate to the library and select it. Click **OK** on the Upload New Library dialog box to load the library into Oracle Virtual Directory. The library appears in the Libraries entry of the Advanced tree.

9.7 Copying Configuration Files Between Oracle Virtual Directory Servers Using syncovdconfig

You can use the `syncovdconfig` command to copy the following Oracle Virtual Directory configuration files between multiple Oracle Virtual Directory components:

- server.os_xml
- adapters.os_xml
- acls.os_xml
- schema.user.xml

The `syncovdconfig` command (.pl for UNIX/Linux and.bat for Windows) is located in the $*ORACLE_HOME*/ovd/bin/ directory. The following is the syntax for `syncovdconfig`:

```
syncovdconfig -srcHost source_host_name -srcPort source_port_number
-srcUserName source_user_name -dstHost destination_host_name
-dstPort destination_port_number -dstUserName destination_user_name
-configFile name_of_configuration_file -adapterName name_of_adapter
-isSrcAdminSSL [true |false] -isDstAdminSSL [true |false]
```

Notes:

- You will be prompted for the password for both the source and destination users.
- Set the `Oracle Home` variable before using the `syncovdconfig` command.

9.7.1 Options

The following is a list of the options for `syncovdconfig`:

srcHost
Required. String format. The host name of the source Oracle Virtual Directory server—that is, the Oracle Virtual Directory server that contains the configuration files you want to copy to a different Oracle Virtual Directory server.

srcPort
Required. Integer format. The listening port number of the source Oracle Virtual Directory server—that is, the Oracle Virtual Directory server that contains the configuration files you want to copy to a different Oracle Virtual Directory server.

srcUserName
Optional. String format. The superuser of the source Oracle Virtual Directory server—that is, the Oracle Virtual Directory server that contains the configuration files you want to copy to a different Oracle Virtual Directory server. If the srcUserName option is not specified, the default value of cn=orcladmin is used.

dstHost
Required. String format. The host name of the destination Oracle Virtual Directory server—that is, the Oracle Virtual Directory server where you want to copy the configuration files to.

dstPort
Required. Integer format. The listening port number of the destination Oracle Virtual Directory server—that is, the Oracle Virtual Directory server where you want to copy the configuration files to.

dstUserName
Optional. String format. The superuser of the destination Oracle Virtual Directory server—that is, the Oracle Virtual Directory server where you want to copy the configuration files to. If the dstUserName option is not specified, the default value of cn=orcladmin is used.

configFile
Optional. String format. The name of the configuration file on the source Oracle Virtual Directory server that you want to copy to the destination Oracle Virtual Directory server. You can use the configFile option multiple times in the same command to copy more than one configuration file.

If you do not include the configFile option, the server.os_xml, adapters.os_xml, acls.os_xml, and schema.user.xml files on the source Oracle Virtual Directory server are copied to the destination Oracle Virtual Directory server.

adapterName
Optional. String format. The name of the adapter on the source Oracle Virtual Directory server that you want to copy to the destination Oracle Virtual Directory server. You can use the adapterName option multiple times in the same command to copy more than one adapter.

If you do not include the adapterName option—but you include the configFile option and specify an adapters.os.xml file, you will overwrite the adapters.os.xml file on the destination Oracle Virtual Directory server.

Surround adapter names that contain space characters with quotation marks ("). For example:

```
ORACLE_HOME/ovd/bin/syncovdconfig.pl -srcHost sales.west.com -srcPort 8888 \
-dstHost sales.east.com -dstPort 8899 -configFile adapters.os_xml \
-adapterName "Sales Organizations"
```

isSrcAdminSSL
Optional. Boolean format. Indicates whether or not the administrative Listener on the source Oracle Virtual Directory component is SSL enabled. Supported values are true

and false. If the isSrcAdminSSL option is not specified, the default value of true is used.

isDstAdminSSL
Optional. Boolean format. Indicates whether or not the administrative Listener on the destination Oracle Virtual Directory component is SSL enabled. Supported values are true and false. If the isDstAdminSSL option is not specified, the default value of true is used.

9.7.2 Examples

The following are examples of the `syncovdconfig` command:

- To synchronize the server.os_xml, adapters.os_xml, acls.os_xml, and schema.user.xml files between two Oracle Virtual Directory components:

```
ORACLE_HOME/ovd/bin/syncovdconfig.pl -srcHost  sales.west.com -srcPort 8899 \
-srcUserName cn=orcladmin  -dstHost sales.west.com -dstPort 8888 -dstUserName \
cn=orcladmin  -isSrcAdminSSl true -isDstAdminSSL false
```

- To synchronize only the server.os_xml file between two Oracle Virtual Directory components:

```
ORACLE_HOME/ovd/bin/syncovdconfig.pl -srcHost  sales.west.com -srcPort 8899 \
-srcUserName cn=orcladmin -dstHost sales.west.com -dstPort 8888 \
-dstUserName cn=orcladmin -configFile server.os_xml
```

- To synchronize multiple files between two Oracle Virtual Directory components:

```
ORACLE_HOME/ovd/bin/syncovdconfig.pl -srcHost  sales.west.com -srcPort 8899 \
-srcUserName cn=orcladmin -dstHost sales.west.com -dstPort 8888 \
-dstUserName cn=orcladmin -configFile server.os_xml -configFile adapters.os_xml
```

- To synchronize a specific adapter between two Oracle Virtual Directory components:

```
ORACLE_HOME/ovd/bin/syncovdconfig.pl -srcHost  sales.west.com -srcPort 8899 \
-srcUserName cn=orcladmin -dstHost sales.west.com -dstPort 8888 \
-dstUserName cn=orcladmin -configFile server.os_xml \
-configFile adapters.os_xml -adapterName Sales
```

10

Managing Oracle Virtual Directory Server Processes

This chapter explains Oracle Virtual Directory process management using Oracle Process Manager and Notification Server and includes the following topics:

10.1 What is Oracle Process Manager and Notification Server?

The Oracle Process Manager and Notification Server (OPMN) is a daemon process that monitors Oracle Fusion Middleware components, including Oracle Virtual Directory. Oracle Enterprise Manager Fusion Middleware Control uses OPMN to stop or start Oracle Virtual Directory. From the command-line, you can use `opmnctl`, the command-line interface to OPMN, to perform the process management tasks for Oracle Virtual Directory that are documented in this chapter.

See Also: The *Oracle Process Manager and Notification Server Administrator's Guide* for complete information about OPMN and the `opmnctl` command.

10.2 Understanding the Default Oracle Virtual Directory Image

When you install Oracle Virtual Directory on a host computer, the Oracle Identity Management 11g Installer creates:

- An Oracle Fusion Middleware component of Type=OVD in a new or existing Oracle instance. The Oracle Virtual Directory component name is usually ovd1 and the Oracle instance name is usually asinst_1.
- File system directories under the Oracle instance directory. Some of the directory path names the installer creates are specific to the component name. For example, the path names under the Oracle instance on UNIX or Linux include:
  - *ORACLE_INSTANCE*/config/OVD/*ovd1*
  - *ORACLE_INSTANCE*/diagnostics/logs/OVD/*ovd1*

If you selected either the Create New Domain or Extend Existing Domain options during installation, the Oracle Virtual Directory component will be registered with a WebLogic domain. If you selected the None option during installation, the Oracle Virtual Directory component will not be registered with a domain. Oracle recommends registering the Oracle Virtual Directory component with a domain. You can register it from the command-line using opmnctl as described in this chapter.

If you install multiple Oracle Virtual Directory components on multiple nodes using the Extend Existing Domain option during installation, the second and subsequent nodes will have component names of ovd2, ovd3 and so on.

10.3 Creating an Oracle Virtual Directory Component Using OPMNCTL

You create an Oracle Virtual Directory component in an Oracle instance by using `opmnctl createcomponent`. The following is the syntax for creating an Oracle Virtual Directory component using `opmnctl createcomponent`:

```
$ORACLE_INSTANCE/bin/opmnctl createcomponent
   [-adminHost hostname]
   [-adminPort weblogic_port]
   [-adminUsername weblogic_admin]
   [-adminPasswordFile 'FILE_WITH_WEBLOGIC_ADMIN_PASSWORD']
   -componentType OVD
   -componentName componentName
   [-passwordFile 'FILE_WITH_OVD_ADMIN_PASSWORD']
   [-admin cn=orcladmin]
   [-isAdminSSL true | false ]
   [-ovdAdminPort OVD_ADMIN_GATEWAY_PORT]
   [-namespace dc=us,dc=oracle,dc=com]
   [-ldapPort LDAP_PORT]
   [-ldapSport  SSL_ENABLED_LDAP_PORT]
   [-httpPort HTTP_PORT]
   [-isHttpSSL true | false]
```

You can use several parameters with the opmnctl createcomponent command. The following is a list of parameters that are specific to Oracle Virtual Directory. Refer to the *Oracle Process Manager and Notification Server Administrator's Guide* to see all the parameters for the opmnctl createcomponent command.

-admin
Oracle Virtual Directory admin username, for example: cn=orcladmin. The default value is cn=orcladmin.

-passwordFile
Oracle Virtual Directory admin password file. You will be prompted for a password if you do not specify a file location.

-isAdminSSL
Enables and disables SSL on the Oracle Virtual Directory Admin Listener. Supported values are true and false. The default value is true.

-ovdAdminPort
Identifies the port for the Oracle Virtual Directory Admin Listener. The default values is 8899.

-namespace
Namespace value, for example: dc=us,dc=oracle,dc=com

-ldapPort
Identifies the port for Oracle Virtual Directory LDAP Listener. The default value is 6501.

-ldapSport
Identifies the SSL port for Oracle Virtual Directory LDAP Listener. The default value is 6502.

-httpPort
Identifies the port for Oracle Virtual Directory HTTP Listener. The default value is 8080.

-isHttpSSL
Enables and disables SSL on the Oracle Virtual Directory HTTP Listener. The default value is true.

Example 10–1 opmnctl createcomponent Command

The following example command creates an Oracle Virtual Directory component named ovd3:

```
$ORACLE_INSTANCE/bin/opmnctl createcomponent -adminHost sales.west.com \
-adminPort 7001 -adminUsername weblogic -componentName ovd3 -componentType OVD \
-admin cn=admin -isAdminSSL true -ovdAdminPort 8890 \
-namespace dc=us,dc=oracle,dc=com -ldapPort 5566 -ldapSport 4455 -httpPort 9090 \
-isHttpSSL true
```

10.4 Registering an Oracle Instance Using OPMNCTL

To register an Oracle instance and all the components in that Oracle instance, you use `opmnctl registerinstance`. The syntax is:

```
$ORACLE_INSTANCE/bin/opmnctl registerinstance
 [-adminHost hostname]
 [-adminPort weblogic_port]
 [-adminUsername weblogic_admin]
 [-adminPasswordFile 'FILE_WITH_WEBLOGIC_ADMIN_PASSWORD']
```

For example:

```
$ORACLE_INSTANCE/bin/opmnctl registerinstance \
 -adminHost myhost \
 -adminPort 7001 \
 -adminUsername weblogic \
```

The default administrative port on the WebLogic Administration Server is 7001.

10.5 Unregistering an Oracle Instance Using OPMNCTL

To unregister an Oracle Instance and all the components in that Oracle instance, you use `opmnctl unregisterinstance`. The syntax is:

```
$ORACLE_INSTANCE/bin/opmnctl unregisterinstance
 [-adminHost hostname]
 [-adminPort weblogic_port]
 [-adminUsername weblogic_admin]
 [-adminPasswordFile 'FILE_WITH_WEBLOGIC_ADMIN_PASSWORD']
```

For example:

```
$ORACLE_INSTANCE/bin/opmnctl unregisterinstance -adminHost myhost \
-adminPort 7001 -adminUsername weblogic \
```

The default administrative port on the WebLogic Administration Server is 7001.

10.6 Updating the Component Registration of an Oracle Instance Using OPMNCTL

To update the registration of an Oracle Virtual Directory component in a registered Oracle instance after changing the Oracle Virtual Directory component's registration, you use `opmnctl updatecomponentregistration`. The `opmnctl updatecomponentregistration` command updates the registration for the Oracle Virtual Directory component using the values in its listeners.os_xml and server.os_xml files.

The syntax for `opmnctl updatecomponentregistration` is:

```
$ORACLE_INSTANCE/bin/opmnctl updatecomponentregistration
   [-adminHost hostname]
   [-adminPort weblogic_port]
   [-adminUsername weblogic_admin]
   [-adminPasswordFile 'FILE_WITH_WEBLOGIC_ADMIN_PASSWORD']
   [-componentType OVD]
   -componentName componentName
   [-Host OVD_HOST_NAME]
```

Notes:

- If you do not use the -Host option, the value in listeners.os_xml will be used.
- Both the `componentName` and `componentType` parameters are required.

For example:

```
$ORACLE_INSTANCE/bin/opmnctl updatecomponentregistration -adminHost myhost \
-adminPort 7001 -adminUsername weblogic -componentType OVD -componentName ovd1
```

10.7 Deleting an Oracle Virtual Directory Component Using OPMNCTL

You remove an Oracle Virtual Directory component by using `opmnctl deletecomponent`. The syntax is:

```
$ORACLE_INSTANCE/bin/opmnctl deletecomponent
    [-adminHost hostname]
    [-adminPort weblogic_port]
    [-adminUsername weblogic_admin]
    [-adminPasswordFile 'FILE_WITH_WEBLOGIC_ADMIN_PASSWORD']
    [-componentType ovd]
    -componentName componentName
```

For example:

```
$ORACLE_INSTANCE/bin/opmnctl deletecomponent -adminHost myhost -adminPort 7001 \
-adminUsername weblogic -componentType OVD -componentName ovd1
```

10.8 Viewing Active Server Instance Information Using OPMNCTL

To view the status of components and processes using `opmnctl`, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl status -l
```

Note: Both HTTP endpoints (Admin and WebGateway) in Oracle Virtual Directory have the identical protocol name of `http`. However, you can differentiate between the two using the description reflected in the `opmnctl debug` command, not using the `opmnctl status -l` command.

Oracle Enterprise Manager Fusion Middleware Control does not show the description field while displaying port information of a server.

10.9 Starting the Oracle Virtual Directory Server Using OPMNCTL

Typically, the component name of the first Oracle Virtual Directory component is `ovd1`.

To start the first Oracle Virtual Directory component, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl startproc ias-component=ovd1
```

To start all Oracle Virtual Directory components, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl startproc process-type=OVD
```

To start all components, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl startall
```

10.10 Stopping the Oracle Virtual Directory Server Using OPMNCTL

To stop the first Oracle Virtual Directory component, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl stopproc ias-component=ovd1
```

To stop all Oracle Virtual Directory components, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl stopproc process-type=OVD
```

To stop all components, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl stopall
```

10.11 Restarting the Oracle Virtual Directory Server Using OPMNCTL

The `opmnctl restartproc` command performs a "soft" restart of the Oracle Virtual Directory server, that is, it reloads the Oracle Virtual Directory configuration, but does not kill the current the Oracle Virtual Directory server process.

To restart the first Oracle Virtual Directory component, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl restartproc ias-component=ovd1
```

To restart all Oracle Virtual Directory components, use the following:

```
$ORACLE_INSTANCE/bin/opmnctl restartproc process-type=OVD
```

11

Creating and Managing Oracle Virtual Directory Listeners

This chapter explains how to create Oracle Virtual Directory Listeners and includes the following topics:

11.1 What is a Listener?

Oracle Virtual Directory provides services to clients through connections known as Listeners. Oracle Virtual Directory supports the following two types of Listeners:

- LDAP: provides LDAPv2/v3 based services
- HTTP: provides one or more services such as DSMLv2, or basic white page functions provided by an XSLT enabled Web Gateway

An Oracle Virtual Directory configuration can have any number of Listeners or it can even have zero Listeners, thus restricting access to only the administrative gateway. Most Oracle Virtual Directory deployments will need no more than two HTTP Listeners and two LDAP Listeners, where one Listener is for SSL and one for non-SSL for each protocols.

Note: You must explicitly stop and start Oracle Virtual Directory—not Restart—to load Listener configurations to the Oracle Virtual Directory server. This includes after creating, updating, or deleting a Listener.

11.2 Understanding the Default Oracle Virtual Directory Listeners

Oracle Virtual Directory includes two Listeners by default: an HTTP Listener named *Admin Gateway* and an LDAP Listener named *LDAP SSL Endpoint*.

Admin Gateway

The HTTP Listener named Admin Gateway is the interface the Oracle Virtual Directory server uses to communicate with the Oracle Directory Services Manager and Oracle Enterprise Manager Fusion Middleware Control user interfaces. You cannot communicate with the Oracle Virtual Directory using the Oracle Directory Services Manager and Oracle Enterprise Manager Fusion Middleware Control user interfaces if you disable the Admin Gateway Listener. Refer to "Editing the Oracle Virtual Directory Administrative Listener Settings" for more information about editing the Oracle Virtual Directory Administrative Listener settings.

LDAP SSL Endpoint

The LDAP Listener named LDAP SSL Endpoint is the interface Oracle Virtual Directory uses to provide performance metrics in Oracle Enterprise Manager Fusion Middleware Control. LDAP SSL Endpoint should always be enabled and secured using SSL Server Authentication. Do not delete or disable LDAP SSL Endpoint. If you need an LDAP Listener that is secured using a different SSL mode, create a new Listener using Oracle Enterprise Manager Fusion Middleware Control.

11.2.1 Managing Communication Between Oracle Virtual Directory and Fusion Middleware Control

The communication between Oracle Virtual Directory and Oracle Enterprise Manager Fusion Middleware Control will be disrupted if you edit any of the following settings for the default Listeners (Admin Gateway and LDAP SSL Endpoint):

- Listener Host
- Listener Port
- Enable / Disable SSL

If you edit any of these settings for the default Listeners, you must update the Oracle Enterprise Manager Fusion Middleware Control target discovery information so Oracle Virtual Directory and Oracle Enterprise Manager Fusion Middleware Control can communicate.

To update the Oracle Enterprise Manager Fusion Middleware Control target discovery information, perform the following steps:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control.
2. Right-click the Farm entry in the navigation tree and select **Agent-Monitored Targets**. The Agent-Monitored Targets screen appears.
3. Click the **Configure** button for the appropriate Oracle Virtual Directory target in the Targets table. The Configure Target page appears.
4. Update the following settings according to your current Oracle Virtual Directory environment and click **OK** at the top of the Configure Target page:
   - Machine name
   - Virtual Directory Admin Port
   - Virtual Directory LDAP Port

See Also: The Troubleshooting appendix of the *Oracle Fusion Middleware Administrator's Guide*.

11.3 Configuring Oracle Virtual Directory to Listen on Privileged Ports

Perform the following steps to enable Oracle Virtual Directory 11g Release 1 (11.1.1.2.0) and higher on UNIX/Linux platforms to listen on privileged ports, that is, port numbers less than 1024:

1. As the same user that installed Oracle Virtual Directory, create the cap.ora file as follows:

   ```
   echo `id -ng`: bind  > /tmp/cap.ora
   ```

2. Using the Oracle Process Manager and Notification Server (OPMN) control command, stop all components:

   ```
   $ORACLE_INSTANCE/bin/opmnctl stopall
   ```

3. Change to root user permissions:

   ```
   su root
   ```

4. Update the *ORACLE_HOME*/bin/hasbind file by performing the following steps:

   a. Change ownership of the file to root:

      ```
      chown root $ORACLE_HOME/bin/hasbind
      ```

   b. Change the permissions on the file as follows:

      ```
      chmod 4755 $ORACLE_HOME/bin/hasbind
      ```

5. Copy the cap.ora file you created in step 1 to the /etc/ directory:

   ```
   cp /tmp/cap.ora /etc/cap.ora
   ```

6. Change the permissions on the /etc/cap.ora file as follows:

   ```
   chmod 644 /etc/cap.ora
   ```

7. As the same user that installed Oracle Virtual Directory, start Oracle Virtual Directory and enable it to listen on privileged ports by using the following command:

   ```
   $ORACLE_HOME/bin/hasocket $ORACLE_INSTANCE/bin/opmnctl startall
   ```

   Note: To enable Oracle Virtual Directory to listen on privileged ports, you must start it using only this command.

After performing the steps in this procedure, Oracle Virtual Directory listeners can listen on privileged ports. You can create new listeners and enter privileged port numbers, or edit existing listeners to use privileged port numbers.

11.4 Creating and Managing Listeners Using Fusion Middleware Control

This topic explains how to create and manage Oracle Virtual Directory Listeners using Oracle Enterprise Manager Fusion Middleware Control and contains the following sections:

- Creating LDAP Listeners
- Creating HTTP Listeners
- Managing Listeners

11.4.1 Creating LDAP Listeners

Perform the following steps to create an LDAP Listener using Oracle Enterprise Manager Fusion Middleware Control. Typically, when running secure and non-secure LDAP, there are at least two Listeners configured; one for regular LDAP (default port is 6501) and one for secure LDAP using SSL (default port is 7501).

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target where you want to create the LDAP Listener.
2. Select **Administration** and then **Listeners** from the Oracle Virtual Directory menu. The Listeners screen appears.
3. Click the **Create** button. The Add Listener screen appears.
4. Select **LDAP** from the Listener Type list and set values for the LDAP Listener configuration parameters as described in Table 11–1:

Table 11–1 LDAP Listener Configuration Parameters

Type	Parameter	Description
Basic	Listener Name	Name of the Listener. Use *only* ASCII characters in the value for the Listener Name parameter, as non-ASCII characters are not supported.
	Listener Host	Specify the IP address the Listener should use to listen for connections from clients. By default, Oracle Virtual Directory listens on all IP addresses if no value or 0.0.0.0 is specified for this parameter. **Note:** Do not use a a loopback IP address, including 127.0.0.1, :0:0:1, localhost, and so on, for the Listener Host setting. If you set this parameter to an IP address or host, the Listener will use that IP address or host to listen for connections from clients, regardless of whether the IP address or host is virtual or real.
	Listener Port	The port number the Listener will provide service on. Only one Listener per server can be active on a port at any given time. If Oracle Virtual Directory is installed on the same server as an existing server, for example, an Active Directory domain controller, enter a port that will not conflict with the existing service.
	Threads	The number of active worker threads the Listener will use to concurrently process incoming requests. The Listener will automatically increase the number of threads if you enter an insufficient amount. This initial setting serves only to indicate to Oracle Virtual Directory the expected amount of simultaneous clients so that it can pre-allocate resources. The default setting is 10, which should be sufficient for testing purposes. For production environments, Oracle recommends to increase this setting to 50.
	Listener Enabled	Enables (selected) and disables (not selected) the Listener for service.

Table 11–1 (Cont.) LDAP Listener Configuration Parameters

Type	Parameter	Description
LDAP Options	Anonymous Bind	Controls how Oracle Virtual Directory handles LDAP anonymous authentication. **Allow** will allow anonymous authentication; **Deny** will prevent anonymous operations; and **DenyDNOnly** will prevent empty password authentication. **Note**: According to the LDAP protocol specification, if an LDAP client connects to an LDAP server with a non-empty DN and an empty password, the LDAP server is expected to provide a successful anonymous bind. For applications that are using LDAP for authentication, this could allow end-users to log in to their applications without entering a password. Most LDAP-enabled applications prevent against this use case. However, as *added* security, you can configure Oracle Virtual Directory to prevent this from happening as an extra-safeguard.
	Work Queue Capacity	Specifies the maximum number of pending LDAP requests that can accumulate when all worker threads associated with LDAP Listener are busy processing requests. Once the specified capacity is reached, the LDAP Listener rejects new requests with `DSA is busy` error. The default value is 1024.
	Allow StartTLS	Determines whether or not LDAP clients can use StartTLS. If enabled, the LDAP Listener allows clients to use the StartTLS extended operation to initiate secure communication over an insecure channel.
Socket Options	Backlog	Determines the maximum number of pending connection requests that can accumulate before the server starts rejecting new connection attempts. Default setting is 128.
	Read Timeout	Enables and disables tolerance for idle client connections with the specified timeout period in milliseconds. If set to a non-zero time, client connections to the Oracle Virtual Directory server can remain idle only for the set amount of time. If the connection is idle for a period longer than the specified time, the client connection is terminated. A value of zero is considered an infinite timeout. The default value is 0.
	Reuse Address	Determines whether or not the LDAP Listener should reuse socket descriptors. If enabled, socket descriptors for clients in TIME_WAIT state can be reused.
	TCP Keep Alive	Determines whether or not the LDAP connection should use TCP keep-alive. If enabled, TCP keep-alive messages are periodically sent to the client to verify that the associated connection is still valid.
	TCP No Delay	Determines whether or not the LDAP connection should use TCP no-delay. If enabled, response messages to the client are sent immediately, rather than potentially waiting to determine whether additional response messages can be sent in the same packet.

5. Click the **OK** button on the Add Listener screen to save the LDAP Listener.
6. Stop Oracle Virtual Directory if it is running by referring to Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10. After it stops, start Oracle Virtual Directory by referring to Starting the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10.

Note: You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.4.2 Creating HTTP Listeners

Perform the following steps to create an HTTP Listener using Oracle Enterprise Manager Fusion Middleware Control:

> **See:** Appendix C, "HTTP Listener's Web Gateway Service" for more information about the HTTP Listener's Web Gateway settings.

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target where you want to create the HTTP Listener.
2. Select **Administration** and then **Listeners** from the Oracle Virtual Directory menu. The Listeners screen appears.
3. Click the **Create** button. The Add Listener screen appears.
4. Select **HTTP** from the Listener Type list and set values for the HTTP Listener configuration parameters as described in Table 11–2:

Table 11–2 HTTP Listener Configuration Parameters

Type	Parameter	Description
Basic	Listener Name	Name of the Listener. Use *only* ASCII characters in the value for the Listener Name parameter, as non-ASCII characters are not supported.
	Listener Host	Specify the IP address the Listener should use to listen for connections from clients. By default, Oracle Virtual Directory listens on all IP addresses if no value or 0.0.0.0 is specified for this parameter. **Note:** Do not use a a loopback IP address, including 127.0.0.1, :0:0:1, localhost, and so on, for the Listener Host setting. If you set this parameter to an IP address or host, the Listener will use that IP address or host to listen for connections from clients, regardless of whether the IP address or host is virtual or real.
	Listener Port	The port number the Listener will provide service on. Only one Listener per server can be active on a port at any given time.
	Threads	The number of active worker threads the Listener will use to concurrently process incoming requests. The Listener will automatically increase the number of threads if you enter an insufficient amount. This initial setting serves only to indicate to Oracle Virtual Directory the expected amount of simultaneous clients so that it can pre-allocate resources. The default setting is 10, which should be sufficient for testing purposes. For production environments, Oracle recommends to increase this setting to 50.
	Listener Enabled	Enables (selected) and disables (not selected) the Listener for service.
DSML V2 Service	Realm Name	Name of the realm used by Oracle Virtual Directory to protect the DSMLv2 service when the DSMLv2 service is security enabled. This realm name would appear in a HTTP browser challenge to the user.
Web Gateway Service Section	Allow Anonymous Access	Enables and disables anonymous access to the Web Gateway.
	Search Root	The root distinguished name (namespace) of the directory tree where the Web Gateway will start its sub-tree search for user identity names (UIDs) provided after a user authentication challenge.
	Search Attributes	The attribute the Web Gateway attempts to match when searching for a UID.
	User Object Classes	The objectclasses the Web Gateway uses when searching for users to authenticate.
	Result Cache Life (seconds)	Maximum time that Oracle Virtual Directory will wait before requerying a user credential stored in the directory source.

Table 11–2 (Cont.) HTTP Listener Configuration Parameters

Type	Parameter	Description
	HTDocs Path	The directory path, relative to the Oracle Virtual Directory root installation, where the XSLT and HTML files are located.
	Certificate Attributes	Indicates which attributes contain binary PKI certificate information. The default value is usercertificate.
	Photo/Image Attributes	Indicates which attributes contain graphical images. The default value is jpegphoto.
	Image Display Height	The height the Web Gateway scales photos to. The default value is 100.
	Image Display Width	The width the Web Gateway scales photos to. The default value is 100.

5. Click the **OK** button on the Add Listener screen to save the HTTP Listener.
6. Stop Oracle Virtual Directory if it is running by referring to Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10. After it stops, start Oracle Virtual Directory by referring to Starting the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10.

> **Note:** You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.4.3 Managing Listeners

This topic explains how to manage Oracle Virtual Directory Listeners using Oracle Enterprise Manager Fusion Middleware Control and contains the following sections:

- Editing Listener Settings
- Deleting Listeners

11.4.3.1 Editing Listener Settings

Perform the following steps to update settings for an existing Listener (LDAP or HTTP) using Oracle Enterprise Manager Fusion Middleware Control:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target where the Listener you want to edit resides.
2. Select **Administration** and then **Listeners** from the Oracle Virtual Directory menu. The Listeners screen appears displaying the exiting Listeners.
3. Select the Listener you want to edit by clicking on it.
4. Click the **Edit** button. The Edit Listener screen appears displaying the Listener's current settings.
5. Edit the settings as desired.

   Refer to Table 11–1, " LDAP Listener Configuration Parameters" for information about each LDAP Listener parameter.

   Refer to Table 11–2, " HTTP Listener Configuration Parameters" for information about each HTTP Listener parameter.

6. Click the **OK** button on the Add Listener screen to save the HTTP Listener.
7. Stop Oracle Virtual Directory if it is running by referring to Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10. After it stops, start Oracle Virtual Directory by referring to Starting the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10.

> **Note:** You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.4.3.1.1 Editing the Oracle Virtual Directory Administrative Listener Settings You can edit the settings for the Oracle Virtual Directory Administrative Listener in the same manner that you edit settings for LDAP or HTTP Listeners. However, if you disable the Admin Gateway Listener, you will not be able to communicate to the Oracle Virtual Directory using the Oracle Directory Services Manager and Oracle Enterprise Manager Fusion Middleware Control user interfaces. Refer to "Understanding the Default Oracle Virtual Directory Listeners" for more information about the Admin Listener.

Perform the following steps to edit settings for the Admin Gateway Listener using Oracle Enterprise Manager Fusion Middleware Control:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target.
2. Select **Administration** and then **Listeners** from the Oracle Virtual Directory menu. The Listeners screen appears displaying the exiting Listeners.
3. Select the Admin Gateway Listener by clicking on it.
4. Click the **Edit** button. The Edit Listener screen appears displaying the Admin Gateway Listener's current settings.
5. Edit the Administrative Listener settings as desired and click **Submit**. Each Administrative Listener setting is described below in the "Administrative Listener Settings" section.
6. Stop Oracle Virtual Directory if it is running by referring to Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10. After it stops, start Oracle Virtual Directory by referring to Starting the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10.

> **Note:** You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

Administrative Listener Settings

Listener Host

The name or IP address of the host where the Oracle Virtual Directory server is running. The default value is 0.0.0.0, which sets the Admin Listener to listen on all IP Addresses configured for the host.

Notes:

- Do not use a a loopback IP address, including 127.0.0.1, :0:0:1, localhost, and so on, for the Listener Host setting.
- If you edit the Host setting, you must immediately perform step 6 or you will not be able to communicate with Oracle Virtual Directory using the Oracle Enterprise Manager Fusion Middleware Control user interface.

Listener Port
The port on which Oracle Virtual Directory will provide administrative services on. This is the port is used by Oracle Directory Services Manager and Oracle Enterprise Manager Fusion Middleware Control user interfaces to communicate with the Oracle Virtual Directory server.

Note: If you edit the Listener Port setting, you must immediately perform step 6 or you will not be able to communicate with Oracle Virtual Directory using the Oracle Enterprise Manager Fusion Middleware Control user interface.

Threads
The number of active worker threads the Listener will use to concurrently process incoming requests.

Listener Enabled
Select to enable the Listener for service. If you disable the Admin Gateway Listener, you will not be able to communicate to the Oracle Virtual Directory using the Oracle Directory Services Manager and Oracle Enterprise Manager Fusion Middleware Control user interfaces. The default setting is Enabled.

Change SSL Settings
Displays the current SSL setting (Enabled or Disabled) for the Listener and provides a link to change the Listener's SSL settings. To edit the Listener's SSL Settings, click the link and refer to "Configuring SSL for Listeners Using Fusion Middleware Control" on page 11-21 for more information.

Note: If you edit the SSL setting (Enabled or Disabled), you must update the Oracle Virtual Directory component registration by referring to Updating the Component Registration of an Oracle Instance Using OPMNCTL on page 10-4. If you do not update the Oracle Virtual Directory component registration after editing the SSL setting, you will not be able to communicate with Oracle Virtual Directory using the Oracle Enterprise Manager Fusion Middleware Control user interface.

11.4.3.2 Deleting Listeners

Perform the following steps to delete an existing Listener (LDAP or HTTP) using Oracle Enterprise Manager Fusion Middleware Control:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target where the Listener you want to delete resides.

2. Select **Administration** and then **Listeners** from the Oracle Virtual Directory menu. The Listeners screen appears displaying the exiting Listeners.
3. Click the Listener you want to delete.
4. Click the **Delete** button. A dialog box appears asking you to confirm that you want to delete the Listener.
5. Click **OK** on the dialog box to delete the Listener. The Listener is removed from the list of existing Listeners.
6. Stop Oracle Virtual Directory if it is running by referring to Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10. After it stops, start Oracle Virtual Directory by referring to Starting the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10.

Note: You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.5 Managing Listeners Using WLST

This topic explains how to manage Oracle Virtual Directory Listeners using WLST and contains the following sections:

- Updating Listener Settings
- Deleting Listeners

See Also:

- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool* for information on how to use the WLST command line tool.
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* for information WLST command tool syntax.

11.5.1 Updating Listener Settings

You can use WLST to update the settings for an existing Listener as follows:

1. Launch the WLST command line tool shell.
2. Connect to the WebLogic Admin Server. For example:

```
connect('username', 'password','t3://host_name:Admin_Server_Port')
```

3. Move to the Oracle Virtual Directory Root Proxy MBean node and initialize the MBean. For example:

```
custom()
cd('oracle.as.management.mbeans.register')
cd('oracle.as.management.mbeans.register:type=component,name=ovd1,instance=asinst1')
invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.String))
```

4. Move to the MBean node for the Listener you want to update, for example, the Listener named *LDAP SSL Endpoint*:

```
cd('../..')
cd('oracle.as.ovd')
```

```
cd('oracle.as.ovd:type=component.Listenersconfig.sslconfig,name=LDAP SSL
Endpoint,instance=asinst_1,component=ovd1')
```

5. Using the WLST `set()` command, update the appropriate setting. The following example updates the Threads setting:

```
set('Threads', 20)
```

Notes:

- Do not use a a loopback IP address, including 127.0.0.1, :0:0:1, localhost, and so on, for the Host setting.
- If you edit the Host, Port, or SSL setting for the Admin Listener, you must update the Oracle Virtual Directory component registration by referring to Updating the Component Registration of an Oracle Instance Using OPMNCTL on page 10-4. If you do not update the Oracle Virtual Directory component registration after editing any of these settings for the Admin Listener, you will not be able to communicate with Oracle Virtual Directory using WLST.

See Also: The following sections to learn more about the Listener settings you can configure using WLST:

- Configuring Admin Listener Settings Using WLST
- Configuring LDAP Listener Settings Using WLST
- Configuring HTTP Listener Settings Using WLST

6. Save the changes and then refresh the MBean. For example:

```
cd('../..')
cd('oracle.as.management.mbeans.register')
cd('oracle.as.management.mbeans.register:type=component,name=ovd1,instance=asin
st1')
invoke('save',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
g))
invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
g))
```

7. Stop Oracle Virtual Directory if it is running. After it stops, start Oracle Virtual Directory.

Note: You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.5.1.1 Configuring Admin Listener Settings Using WLST

The following is a list and description of the Admin Listener settings you can configure using WLST:

See Also: "Understanding the Default Oracle Virtual Directory Listeners" for more information about the Admin Listener.

Active
Determines whether or not the Listener is enabled or disabled. Supported values are true and false. If you disable the Admin Listener, you will not be able to communicate to the Oracle Virtual Directory using the Oracle Directory Services Manager and Oracle Enterprise Manager Fusion Middleware Control user interfaces.

AuthenticationType
Determines the authentication mode for the Listener. Supported values are None, Server, and Mutual.

- None configures the Listener for SSL No-Authentication Mode
- Server configures the Listener for SSL Server Authentication Mode
- Mutual configures the Listener for SSL Mutual Authentication

BindAddress
The InetAddress representation of value for the Host setting. If you edit the BindAddress setting, the Host setting also changes. Conversely, if you edit the Host setting, the BindAddress setting also changes.

Ciphers
Configures cipher suite negotiation, which is part of the SSL handshaking used to initiate or verify secure communications. A cipher suite is a combination of cryptographic parameters that define the security algorithms and key sizes used for authentication, key agreement, encryption, and integrity protection. The default value is null. The following is a list of the supported values for the Ciphers setting:

- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA

GroupURL
An LDAP URL which defines a group of users with privileges to use the Admin Listener. These users will have near root privileges when accessing the Oracle Virtual Directory server through the Oracle Enterprise Manager Fusion Middleware Control and Oracle Directory Services Manager interfaces.

Host
The name or IP address of the host where the Oracle Virtual Directory server is running. The default value is 0.0.0.0, which sets the Admin Listener to listen on all IP Addresses configured for the host.

Note: Do not use a a loopback IP address, including 127.0.0.1, :0:0:1, localhost, and so on, for the Host setting.

KeyStore
The name of the JKS keystore containing the SSL artifacts.

Name
The name of the Listener.

Port
The port on which Oracle Virtual Directory will provide administrative services on. This is the port is used by Oracle Directory Services Manager and Oracle Enterprise Manager Fusion Middleware Control user interfaces to communicate with the Oracle Virtual Directory server.

Protocol
The protocol the Admin Listener uses to provide service. Supported values are HTTP and HTTPS.

SSLEnabled
Determines whether or not SSL is enabled on the Listener. Supported values are true and false.

SSLVersions
The supported protocols for SSL communication. The following is a list of the supported values:

- TLSv1
- SSLv2Hello

> **Note:** The SSLv2Hello value cannot be specified alone. If you specify SSLv2Hello, you must also specify at least one other supported version.

- SSLv3

Threads
The number of active worker threads the Listener will use to listen for connections on the port.

TrustStore
The name of the JKS keystore containing the SSL artifacts.

11.5.1.2 Configuring LDAP Listener Settings Using WLST

The following is a list and description of the LDAP Listener settings you can configure using WLST:

Active
Determines whether or not the Listener is enabled or disabled. Supported values are true and false.

AllowStartTLS
Determines whether or not LDAP clients can use StartTLS. If enabled, the LDAP Listener allows clients to use the StartTLS extended operation to initiate secure communication over an insecure channel. Supported values are true and false. The default value is false.

AnonymousBind
Controls how Oracle Virtual Directory handles LDAP anonymous authentication. Supported values are listed in Table 11–3:

Table 11–3 LDAP Anonymous Authentication Options

Option	Control
Allow	Allow anonymous authentication.
Deny	Prevent anonymous operations.
DenyDNOnly	Prevent empty password authentication. **Note**: According to the LDAP protocol specification, if an LDAP client connects to an LDAP server with a non-empty DN and an empty password, the LDAP server is expected to provide a successful anonymous bind. For applications that are using LDAP for authentication, this could allow end-users to log in to their applications without entering a password. Most LDAP-enabled applications prevent against this use case. However, as *added* security, you can configure Oracle Virtual Directory to prevent this from happening as an extra-safeguard.

AuthenticationType
Determines the authentication mode for the Listener. Supported values are None, Server, and Mutual.

- None configures the Listener for SSL No-Authentication Mode
- Server configures the Listener for SSL Server Authentication Mode
- Mutual configures the Listener for SSL Mutual Authentication

BindAddress
The InetAddress representation of value for the Host setting. If you edit the BindAddress setting, the Host setting also changes. Conversely, if you edit the Host setting, the BindAddress setting also changes.

Ciphers
Configures cipher suite negotiation, which is part of the SSL handshaking used to initiate or verify secure communications. A cipher suite is a combination of cryptographic parameters that define the security algorithms and key sizes used for authentication, key agreement, encryption, and integrity protection. The default value is null. The following is a list of the supported values for the Ciphers setting:

- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA

- TLS_RSA_WITH_AES_256_CBC_SHA

ExtendedOpsClass
In addition to the normal LDAP operations supported by the LDAP protocol, you can define your own LDAP operation using this setting. This setting is the full java class name that implements your user-defined LDAP operation.

ExtendedOpsOid
The unique name for your user-defined LDAP operation identified by the ExtendedOpsClass setting.

Host
The name or IP address of the host where the Oracle Virtual Directory server is running. The default value is 0.0.0.0.

Note: Do not use a a loopback IP address, including 127.0.0.1, :0:0:1, localhost, and so on, for the Host setting.

KeyStore
The name of the JKS keystore containing the SSL artifacts.

Name
The name of the Listener.

Port
The port number the LDAP Listener will provide service on. Only one Listener per server can be active on a port at any given time.

Protocol
The protocol the LDAP Listener uses to provide service. Supported values are LDAP and LDAPS.

SSLEnabled
Determines whether or not SSL is enabled on the Listener. Supported values are true and false.

SSLVersions
The supported protocols for SSL communication. The following is a list of the supported values:

- TLSv1
- SSLv2Hello

Note: The SSLv2Hello value cannot be specified alone. If you specify SSLv2Hello, you must also specify at least one other supported version.

- SSLv3

SocketOptionsBacklog
Determines the maximum number of pending connection requests that can accumulate before the server starts rejecting new connection attempts. Default setting is 128.

SocketOptionsKeepAlive
Determines whether or not the LDAP connection should use TCP keep-alive. If enabled, TCP keep-alive messages are periodically sent to the client to verify that the associated connection is still valid. Supported values are true and false. The default value is false.

SocketOptionsReadTimeout
Enables and disables tolerance for idle client connections with the specified timeout period in milliseconds. If set to a non-zero time, client connections to the Oracle Virtual Directory server can remain idle only for the set amount of time. If the connection is idle for a period longer than the specified time, the client connection is terminated. A value of zero is considered an infinite timeout. The default value is 0.

SocketOptionsReuseAddress
Determines whether or not the LDAP Listener should reuse socket descriptors. If enabled, socket descriptors for clients in TIME_WAIT state can be reused. Supported values are true and false. The default value is false.

SocketOptionsTcpNoDelay
Determines whether or not the LDAP connection should use TCP no-delay. If enabled, response messages to the client are sent immediately, rather than potentially waiting to determine whether additional response messages can be sent in the same packet. Supported values are true and false. The default value is true.

Threads
The number of active worker threads the Listener will use to concurrently process incoming requests. The Listener will automatically increase the number of threads if you indicate an insufficient amount. This initial setting serves only to indicate to Oracle Virtual Directory the expected amount of simultaneous clients so that it can pre-allocate resources. The default setting is 10, which should be sufficient for testing purposes. For production environments, Oracle recommends to increase this setting to 50.

TrustStore
The name of the JKS keystore containing the SSL artifacts.

WorkQueueCapacity
Specifies the maximum number of pending LDAP requests that can accumulate when all worker threads associated with LDAP Listener are busy processing requests. Once the specified capacity is reached, the LDAP Listener rejects new requests with `DSA is busy` error. The default value is 1024.

> **Note:** The `DSA is busy` error usually appears when a large number of requests are sent to the Oracle Virtual Directory server in a short period of time and the LDAP Listener is unable to support them.

11.5.1.3 Configuring HTTP Listener Settings Using WLST

The following is a list and description of the HTTP Listener settings you can configure using WLST:

Active
Determines whether or not the Listener is enabled or disabled. Supported values are true and false.

AuthenticationType
Determines the authentication mode for the Listener. Supported values are None, Server, and Mutual.

- None configures the Listener for SSL No-Authentication Mode
- Server configures the Listener for SSL Server Authentication Mode
- Mutual configures the Listener for SSL Mutual Authentication

BindAddress
The InetAddress representation of value for the Host setting. If you edit the BindAddress setting, the Host setting also changes. Conversely, if you edit the Host setting, the BindAddress setting also changes.

Ciphers
Configures cipher suite negotiation, which is part of the SSL handshaking used to initiate or verify secure communications. A cipher suite is a combination of cryptographic parameters that define the security algorithms and key sizes used for authentication, key agreement, encryption, and integrity protection. The default value is null. The following is a list of the supported values for the Ciphers setting:

- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA

CustomWebappContext
Base URL for the location of the customer developed custom web service.

CustomWebappSecurityRealm
Name of the realm used by Oracle Virtual Directory to protect the custom web service when the custom web service is security enabled.

CustomWebappWebapp
If you want to use your own web application to handle HTTP connections, instead of using the HTTP Listener's Web Gateway and/or DSMLv2 Gateway, use this setting to specify the path to the your custom web application war file.

Dsmlv2SecurityRealm
Name of the realm used by Oracle Virtual Directory to protect the DSMLv2 service when the DSMLv2 service is security enabled. This realm name would appear in a HTTP browser challenge to the user.

Host
The name or IP address of the host where the Oracle Virtual Directory server is running. The default value is 0.0.0.0.

Note: Do not use a a loopback IP address, including 127.0.0.1, :0:0:1, localhost, and so on, for the Host setting.

KeyStore
The name of the JKS keystore containing the SSL artifacts.

Name
The name of the Listener.

Port
The port number the HTTP Listener will provide service on. Only one Listener per server can be active on a port at any given time.

Protocol
The protocol the HTTP Listener uses to provide service. Supported values are HTTP and HTTPS.

SSLEnabled
Determines whether or not SSL is enabled on the Listener. Supported values are true and false.

SSLVersions
The supported protocols for SSL communication. The following is a list of the supported values:

- TLSv1
- SSLv2Hello

Note: The SSLv2Hello value cannot be specified alone. If you specify SSLv2Hello, you must also specify at least one other supported version.

- SSLv3

Threads
The number of active worker threads the Listener will use to concurrently process incoming requests. The Listener will automatically increase the number of threads if you indicate an insufficient amount. This initial setting serves only to indicate to Oracle Virtual Directory the expected amount of simultaneous clients so that it can pre-allocate resources. The default setting is 10, which should be sufficient for testing purposes. For production environments, Oracle recommends to increase this setting to 50.

TrustStore
The name of the JKS keystore containing the SSL artifacts.

WebgatewayAllowAnon
Enables and disables anonymous access to the Web Gateway. Supported values are true and false.

WebgatewayCertifiedAttributes
Indicates which attributes contain binary PKI certificate information. The default value is usercertificate.

WebgatewayHtDocsRoot
The directory path, relative to the Oracle Virtual Directory root installation, where the XSLT and HTML files are located.

WebgatewayMatchAttributes
The attribute the Web Gateway should attempt to match when searching for a UID. The default value is uid, mail, cn.

WebgatewayMatchObjectClasses
The objectclasses the Web Gateway should use when searching for users to authenticate. The default value is inetorgperson, user.

WebgatewayPhotoAttributes
Indicates which attributes contain graphical images. The default value is jpegphoto.

WebgatewayPhotoHeight
The height the Web Gateway scales photos to. The default value is 100.

WebgatewayPhotoWidth
The width the Web Gateway scales photos to. The default value is 100.

WebgatewaySearchRoot
The root distinguished name (namespace) of the directory tree where the Web Gateway will start its sub-tree search for user identity names (UIDs) provided after a user authentication challenge.

WebgatewaySecurityRealm
Name of the realm used by Oracle Virtual Directory to protect the Web Gateway service when the Web Gateway service is security enabled.

WebgatewayUserCacheLife
Maximum time (in seconds) that Oracle Virtual Directory will wait before requerying a user credential stored in the directory source.

11.5.2 Deleting Listeners

You can use WLST to delete an existing Listener as follows:

1. Launch the WLST command line tool shell.
2. Connect to the WebLogic Admin Server. For example:

```
connect('username', 'password','t3://host_name:Admin_Server_Port')
```

3. Move to the Oracle Virtual Directory Root Proxy MBean node and initialize the MBean. For example:

```
custom()
cd('oracle.as.management.mbeans.register')
cd('oracle.as.management.mbeans.register:type=component,name=ovd1,instance=asin
st1')
invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
g))
```

4. Move to the Oracle Virtual Directory Listeners configuration MBean. For example:

```
cd('../..')
cd('oracle.as.ovd/oracle.as.ovd:type=component.Listenersconfig,name=Listenersco
nfig,instance=asinst1,component=ovd1')
```

5. Delete the appropriate Listener, for example, the Listener named *test1*, as follows:

```
invoke('deleteListener',jarray.array([java.lang.String('test1')],java.lang.Obje
ct),jarray.array(['java.lang.String'],java.lang.String))
```

6. Save the changes and then refresh the MBean. For example:

```
cd('../..')
cd('oracle.as.management.mbeans.register')
cd('oracle.as.management.mbeans.register:type=component,name=ovd1,instance=asin
st1')
invoke('save',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
g))
invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.Strin
g))
```

7. Stop Oracle Virtual Directory if it is running. After it stops, start Oracle Virtual Directory.

> **Note:** You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.6 Securing Listeners with SSL

This topic explains how to secure Oracle Virtual Directory Listeners using SSL and contains the following sections:

- Configuring SSL for Listeners Using Fusion Middleware Control
- Configuring SSL for Listeners Using WLST
- Validating the SSL Connection

11.6.1 Configuring SSL for Listeners Using Fusion Middleware Control

Perform the following steps to secure Oracle Virtual Directory Listeners with SSL using Oracle Enterprise Manager Fusion Middleware Control:

> **Note:** If you are configuring the Listener for SSL No-Auth mode, *do not perform* step 2 and steps 3e through 3h in the following procedure.

> **See Also:** The information about enabling SSL for Oracle Virtual Directory Listeners in the *Oracle Fusion Middleware Administrator's Guide*.

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target of the Listener you want to secure with SSL.
2. Create a keystore if one does not already exist by selecting **Security** and then **Keystores** from the Oracle Virtual Directory menu. The Java Keystore screen appears. Refer to the information about creating a keystore using Oracle Enterprise Manager in the *Oracle Fusion Middleware Administrator's Guide* for additional information.
3. Configure the Listener by performing the following steps:

a. Select **Administration** and then **Listeners** from the Oracle Virtual Directory menu. The Listeners screen appears.

b. Select the Listener you want to secure with SSL by clicking on it and then click the **Edit** button. The Edit Listener: *Listener Name* screen appears.

c. Click the **Change SSL Settings** link.

d. Click the **Enable SSL** option to enable SSL on the Listener. If you are configuring the Listener for SSL No-Auth mode, skip to step i now.

e. Select the keystore you want to use from the Server Keystore Name field.

Note: If you select a different keystore or change the certificate in the keystore for the Admin Gateway Listener or the LDAP SSL Endpoint Listener, you must import the certificate into the Oracle Enterprise Manager Fusion Middleware Control Agent's wallet. If you do not import the certificate, Oracle Enterprise Manager Fusion Middleware Control cannot connect to Oracle Virtual Directory to retrieve performance metrics.

To import the certificate into the Oracle Enterprise Manager Fusion Middleware Control Agent's wallet:

1. Export the Oracle Virtual Directory server certificate by executing the following command:

```
ORACLE_HOME/jdk/jre/bin/keytool -exportcert \
-keystore OVD_KEYSTORE_FILE -storepass PASSWORD \
-alias OVD_SERVER_CERT_ALIAS -rfc \
-file OVD_SERVER_CERT_FILE
```

2. Add the Oracle Virtual Directory server certificate to the Oracle Enterprise Manager Fusion Middleware Control Agent's Wallet by executing the following command:

```
ORACLE_COMMON_HOME/bin/orapki wallet add -wallet \
$ORACLE_INSTANCE/EMAGENT/EMAGENT/sysman/config/monwallet \
-trusted_cert -cert OVD_SERVER_CERT_FILE -pwd WALLET_PASSWORD
```

---

f. Enter the password for the keystore in the Server Keystore Password field.

Note: The password for the keystore that is created during the Oracle Virtual Directory installation is the same as the password set for the Oracle Virtual Directory administrator during installation.

---

g. Select the truststore you want to use from the Server Truststore Name field.

h. Enter the password for the truststore in the Server Truststore Name field.

i. Click and expand the **Advanced SSL Setting** option.

j. Select one of the following authentication modes for the Listener from the Client Authentication field.

To configure the Listener for SSL No-Authentication Mode, select **No Authentication**.

To configure the Listener for SSL Server Authentication Mode, select **Server Authentication**.

To configure the Listener for SSL Mutual Authentication mode between the Oracle Virtual Directory server and the client, select **Mutual Authentication**.

Note: The **Optional Client Authentication** mode is not supported for Oracle Virtual Directory Listeners.

k. Select the appropriate option from the Cipher Suite field. You can select **All**, or a combination of individual options.

Note: If you are configuring the Listener for SSL No-Auth mode, you must select at least one DH_anon cipher. For all other SSL modes, you must select at least one RSA cipher.

l. Select the appropriate option from the SSL Protocol Version field.

Note: The **v2Hello** option is not supported by itself. That is, you cannot select the **v2Hello** option alone—you must select it in combination with at least one additional SSL Protocol Versions from the list.

m. Click the **OK** button.

4. Stop Oracle Virtual Directory if it is running by referring to Stopping the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10. After it stops, start Oracle Virtual Directory by referring to Starting the Oracle Virtual Directory Server Using Fusion Middleware Control on page 8-10.

Note: You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.6.2 Configuring SSL for Listeners Using WLST

To configure SSL for Oracle Virtual Directory using the WLST command line tool:

See Also:

- The WLST Reference for SSL information in the *Oracle Fusion Middleware Administrator's Guide*.
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool* for information on how to use the WLST command line tool.
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* for information WLST command tool syntax.

1. Launch the WLST command line tool shell.
2. Go to the custom tree using the following command:

```
custom()
```

3. Navigate to the root Oracle Virtual Directory mBean using the following commands:

```
cd('oracle.as.management.mbeans.register')
cd('oracle.as.management.mbeans.register:type=component,name=COMPONENT_
NAME,instance=INSTANCE_NAME')
```

4. Initialize the Oracle Virtual Directory configuration from the remote Oracle Virtual Directory server into the WebLogic server using the following command:

```
invoke('load',jarray.array([],java.lang.Object),jarray.array([],
java.lang.String))
```

5. Identify the Listeners for this Oracle Virtual Directory component by executing the following command:

```
listListeners('instName', 'compName')
```

   For example:

```
listListeners('instance1','ovd1')
```

   The command lists all the Listeners for the component named ovd1. In the list of Listeners returned, identify the Listener you want to secure using SSL. For example, imagine you want to secure the Listener named *LDAP SSL Endpoint*.

6. Display the existing SSL configuration for the Listener you want secure (*LDAP SSL Endpoint* in this example) using the following command:

```
getSSL('instance1','ovd1','ovd','LDAP SSL Endpoint')
```

7. Display the existing keystores using the following command:

```
listKeyStores('instance1','ovd1','ovd')
```

8. If you need to, create a new keystore and a self-signed certificate using the following commands.

   To create the new keystore, execute the following command:

```
createKeyStore('instance1','ovd1','ovd','NEW_KEYSTORE_NAME','PASSWORD_FOR_NEW_
KEYSTORE')
```

   To create a self-signed certificate in the new keystore, execute the following command:

```
generateKey ('instance1','ovd1','ovd','NEW_KEYSTORE_NAME','PASSWORD_FOR_NEW_
KEYSTORE', 'DN', 'keySize', 'alias')
```

9. Identify the name of the SSL MBean for the Oracle Virtual Directory Listener by executing the following command:

```
getSSLMBeanName('instance1','ovd1','ovd','LDAP SSL Endpoint')
```

10. Set the passwords for the keystore and truststore in the MBean by executing the following commands:

```
cd ('SSL_MBEAN_NAME')
set('KeyStorePassword',java.lang.String('PASSWORD').toCharArray())
set('TrustStorePassword',java.lang.String('PASSWORD').toCharArray())
```

11. Configure the SSL settings for the Listener using the following command and file.prop. An sample file.prop file is given for reference:

```
configureSSL ('instance1', 'ovd1', 'ovd', 'LDAP SSL Endpoint', 'PATH_TO_
file.prop')
```

Note: If you configure a different keystore or change the certificate in the keystore for the Admin Gateway Listener or the LDAP SSL Endpoint Listener, you must import the certificate into the Oracle Enterprise Manager Fusion Middleware Control Agent's wallet. If you do not import the certificate, Oracle Enterprise Manager Fusion Middleware Control cannot connect to Oracle Virtual Directory to retrieve performance metrics.

To import the certificate into the Oracle Enterprise Manager Fusion Middleware Control Agent's wallet:

1. Export the Oracle Virtual Directory server certificate by executing the following command:

   ```
   ORACLE_HOME/jdk/jre/bin/keytool -exportcert \
   -keystore OVD_KEYSTORE_FILE -storepass PASSWORD \
   -alias OVD_SERVER_CERT_ALIAS -rfc \
   -file OVD_SERVER_CERT_FILE
   ```

2. Add the Oracle Virtual Directory server certificate to the Oracle Enterprise Manager Fusion Middleware Control Agent's Wallet by executing the following command:

   ```
   ORACLE_COMMON_HOME/bin/orapki wallet add -wallet \
   $ORACLE_INSTANCE/EMAGENT/EMAGENT/sysman/config/monwallet \
   -trusted_cert -cert OVD_SERVER_CERT_FILE -pwd WALLET_PASSWORD
   ```

Example 11–1 Sample file.prop File

```
SSLEnabled=true
AuthenticationType=auth_type
SSLVersions=version
Ciphers=cipher
KeyStore=name_of_your_keystore
TrustStore=name_of_your_keystore
```

Important Notes Regarding the file.prop File:

- Replace the variable values in the Example 11–1 with the values for your environment.
- If you are configuring the Listener for SSL No-Auth mode, you must select at least one DH_anon cipher. For all other SSL modes, you must select at least one RSA cipher.
- You must specify the value of the KeyStore parameter when configuring SSL for server-auth and mutual-auth modes.
- If you specify only AES ciphers, the SSLVersions parameter must contain TLSv1.
- The text in the file.prop file is case sensitive.
- Do not use spaces after cipher entries in the file.prop file.
- Refer to the "Properties Files for SSL" section in the *Oracle Fusion Middleware Administrator's Guide* for more information about the contents of the file.prop file.

See Also: The following sections for information about the AuthenticationType, SSLVersions, and Ciphers you can configure in File.prop:

- Configuring Admin Listener Settings Using WLST on page 11-12
- Configuring LDAP Listener Settings Using WLST on page 11-14
- Configuring HTTP Listener Settings Using WLST on page 11-17

12. Stop Oracle Virtual Directory if it is running. After it stops, start Oracle Virtual Directory.

Note: You must explicitly stop and start Oracle Virtual Directory—not Restart—to load the Listener configuration to the Oracle Virtual Directory server.

11.6.3 Validating the SSL Connection

This topic explains how to validate SSL connections for each SSL mode and contains the following sections:

- SSL No-Authentication Mode
- SSL Server Auth Mode
- SSL Mutual Authentication Mode

Note: If you are using default settings after installing 11*g* Release 1 (11.1.1), you can use the following values for the following variables described in this section:

- For *OVD_KEY_STORE_FILE*, use:

  ORACLE_INSTANCE/config/OVD/*ovd1*/keystores/keys.jks

- For *OVD_SERVER_CERT_ALIAS*, use serverselfsigned
- For *PASSWORD* used for the `-storepass` and `-jkspwd` options, use the same password as orcladmin

11.6.3.1 SSL No-Authentication Mode

To validate a connection secured by SSL No-Authentication mode, execute the following command:

```
ORACLE_HOME/bin/ldapbind -D cn=orcladmin -q -U 1 -h HOST -p SSL_PORT
```

11.6.3.2 SSL Server Auth Mode

To validate a connection secured by SSL Server Authentication mode, perform the following steps:

1. Create an Oracle Wallet by executing the following command:

   ```
   ORACLE_COMMON_HOME/bin/orapki wallet create -wallet DIRECTORY_FOR_SSL_WALLET \
   -pwd WALLET_PASSWORD
   ```

2. Export the Oracle Virtual Directory server certificate by executing the following command:

```
ORACLE_HOME/jdk/jre/bin/keytool -exportcert -keystore OVD_KEYSTORE_FILE \
-storepass PASSWORD -alias OVD_SERVER_CERT_ALIAS -rfc \
-file OVD_SERVER_CERT_FILE
```

3. Add the Oracle Virtual Directory server certificate to the Oracle Wallet by executing the following command:

```
ORACLE_COMMON_HOME/bin/orapki wallet add -wallet DIRECTORY_FOR_SSL_WALLET \
-trusted_cert -cert OVD_SERVER_CERT_FILE -pwd WALLET_PASSWORD
```

4. Use the Oracle Wallet from step 3 while executing the following command:

```
ORACLE_HOME/bin/ldapbind -D cn=orcladmin -q -U 2 -h HOST -p SSL_PORT \
-W "file://DIRECTORY_FOR_SSL_WALLET" -Q
```

11.6.3.3 SSL Mutual Authentication Mode

To validate a connection secured by SSL Mutual Authentication mode, perform the following steps:

1. Create an Oracle wallet by executing the following command:

```
ORACLE_COMMON_HOME/bin/orapki wallet create -wallet DIRECTORY_FOR_SSL_WALLET \
-pwd WALLET_PASSWORD
```

2. Transform the Oracle Virtual Directory keystore file to an Oracle Wallet by executing the following command:

```
ORACLE_COMMON_HOME/bin/orapki wallet jks_to_pkcs12 \
-wallet DIRECTORY_FOR_SSL_WALLET -pwd WALLET_PASSWORD \
-keystore ORACLE_INSTANCE/config/OVD/OVD_COMPONENT/keystores/keys.jks \
-jkspwd PASSWORD
```

3. Export the client certificate in Base64 format by executing the following command:

```
ORACLE_COMMON_HOME/bin/orapki wallet export -wallet . -dn CLIENT_DN \
-cert ./b64certificate.txt
```

4. Import the client certificate you created in step 2 into the Oracle Virtual Directory keystore as a trusted entry by executing the following command:

```
ORACLE_HOME/jdk/jre/bin/keytool -importcert \
-keystore ORACLE_INSTANCE/config/OVD/OVD_COMPONENT/keystores/keys.jks
-storepass JKS_PASSWORD -alias ALIAS -file b64certificate.txt -noprompt
```

5. Verify the SSL connection using the bind DN of the client certificate by executing the following command:

```
ORACLE_HOME/bin/ldapbind -U 3 -h HOST -p SSL_PORT -W "file://DIRECTORY_FOR_SSL_
WALLET" -Q
```

12

Creating and Configuring Oracle Virtual Directory Adapters

This chapter explains how to create and configure Oracle Virtual Directory adapters and includes the following topics:

- Creating LDAP Adapters
- Creating Database Adapters
- Creating Local Store Adapters
- Creating Join View Adapters

12.1 Creating LDAP Adapters

This topic explains how to create and configure LDAP Adapters and includes the following sections:

- Configuring LDAP Adapters
- Configuring a Mutual Authentication SSL Connection Between Oracle Virtual Directory and Oracle Internet Directory

Perform the following steps to create LDAP Adapters using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the **Create Adapter** button. The New Adapter Wizard appears.
4. Perform the following steps to define the Type of adapter:
   a. Select **LDAP** from the Adapter Type list.
   b. Enter a unique name for the LDAP Adapter in the Adapter Name field. The adapter name value is used in other configuration fields that need to reference the adapter.
   c. Select an adapter template from the Adapter Template list by referring to "Understanding Adapter Templates" on page 2-28. Use the **Default** template if you are unsure which template to use.

Note: After selecting an adapter template, Oracle Directory Services Manager populates default values for some of the adapter settings. You should alter these default settings according to your environment.

---

d. Click **Next**. The Connection screen appears.

5. Select a DNS mode of operation from the **Use DNS for Auto Discovery** options to configure Oracle Virtual Directory to use DNS to automatically discover the appropriate LDAP hosts for the remote base defined (instead of configuring specific LDAP hosts in the Connection Details table). This is also referred to as serverless bind mode. The LDAP Adapter supports the following DNS modes of operation:

   Note: The DNS options are listed in the Oracle Directory Services Manager interface in English only, however the description for each DNS option is supported in localized language translations.

   - **No**: Use the Connection Details table configuration—no serverless bind.
   - **Standard**: Use standard DNS lookup for a non-Microsoft server. All servers are marked as read-write, so enabling the **Follow Referrals** setting is advised to allow for LDAP write support.
   - **Microsoft**: The DNS server is a Microsoft dynamic DNS and also supports load-balancing configuration. If proxying to a Microsoft dynamic DNS server, this is the recommended setting because of Oracle Virtual Directory's ability to auto-detect read/write servers compared to read-only servers.

     Note: Remote base should have a domain component style name when using this setting, for example, dc=myorg,dc=com. This enables Oracle Virtual Directory to locate the LDAP hosts within the DNS service by looking up `myorg.com`.

6. If you selected the **No** option for the **Use DNS for Auto Discovery** setting, add the proxy LDAP host information in the Connection Details table by clicking the **Add Host** button and then entering the following information. Each proxy LDAP host must provide equivalent content, that is, must be replicas.

   Note: Be careful when specifying only a single host for proxying. Without a failover host, the LDAP Adapter cannot automatically fail over to another host. A single host is suitable when Oracle Virtual Directory is connected to a logical LDAP service through a load balancing system.

   a. Enter the IP Address or DNS name of the LDAP host to proxy to in the Hosts field.

      Note: Oracle Virtual Directory 11*g* Release 1 (11.1.1) supports IPv6. If your network supports IPv6 you can use a literal IPv6 address in the Hosts field to identify the proxied LDAP host.

   b. Enter the port number the proxied LDAP host provides LDAP services on in the Port field.

   c. Enter a number between 0 and 100 in the Weight Value field to configure the load percentage to send to the host. If the combined percentages for all of the

hosts configured for the adapter do not total 100, Oracle Virtual Directory automatically adjusts the load percentages by dividing the percentage you entered for a host by the total percentage of all hosts configured for the adapter. For example, if you have three hosts configured for the adapter at 20 percent, 30 percent, and 40 percent, Oracle Virtual Directory adjusts the 20 to 22 (20/90), the 30 to 33 (30/90), and the 40 to 44 (40/90).

d. Select the Read-only option to configure the LDAP Adapter to only perform search operations on the LDAP host. The LDAP Adapter automatically directs all modify traffic to read/write hosts in the list.

7. Select the **Use SSL/TLS** option to secure the communication between the LDAP Adapter and the proxy LDAP hosts using SSL/TLS.

See: "Managing Certificate Authorities for LDAP Adapters Secured by SSL" on page 12-11 for information on Certificate Authorities.

If you select (enable) the **Use SSL/TLS** option, choose the SSL authentication mode to use for securing the adapter by selecting an option from the SSL Authentication Mode list. The SSL Authentication Mode setting is functional only when the **Use SSL/TLS** option is enabled.

8. Enter the default distinguished name for the LDAP Adapter to bind with when accessing the proxied directory in the Server proxy Bind DN field. Depending on the setting in the Pass Through Credentials field, this DN will be used for all operations, or only for exceptional cases such as pass-through mode. The form of the distinguished name must be in the form of the remote directory. The LDAP Adapter binds as Anonymous if the Server proxy Bind DN field is empty.

9. Enter the authentication password in clear text in the Proxy Password field to use with Server proxy Bind DN value. When loaded on the server, the value is automatically encrypted.

10. Click **Next**. Oracle Virtual Directory attempts to validate the connection(s) to the host(s) you defined in the Connection Details table. The Test Connection screen appears displaying the results of the connection validation process.

- Upon successful validations, a success message and the details for the connection appear. Click **Next**. The Name Space screen appears. Continue creating the New LDAP Adapter by advancing to step 11.
- Upon failed validations, a `Could not connect` message appears in the Connection column in the status table for the host connections that could not be validated. Click in the row for the host connection that could not be validated to see more information about why the connection failed. Resolve the failed connections by clicking the **Back** button, reviewing the settings for the host where the connection failed, and then editing the host settings as needed.

  The connection to the proxy LDAP host must be validated for the adapter to proxy the LDAP host. Click **Next** on the Test Connection screen of the New LDAP Adapter Wizard after resolving the failed connection. The Name Space screen appears. Continue creating the New LDAP Adapter by advancing to step 11.

11. Enter the location in the remote server directory tree structure to which the local Oracle Virtual Directory root suffix corresponds in the Remote Base field. This is the location in the remote directory under which Oracle Virtual Directory will execute all searches and operations for the adapter. The LDAP Adapter applies an automatic mapping of all entries from the remote base to the adapter root base.

12. Enter the namespace you want Oracle Virtual Directory clients to see for the proxied directory's namespace in the Mapped Namespace field. For example, if the DN in the proxied directory is dc=oracle, dc=com and you want Oracle Virtual Directory clients to see the namespace as dc=Oracle Corp, dc=com, you would enter `dc=Oracle Corp, dc=com` in the Mapped Namespace field.
13. Set the pass-through credentials for the LDAP Adapter by selecting one of the following options from the Pass Through Credentials list:

    Note: The pass-through options are listed in the Oracle Directory Services Manager interface in English only, however the description for each pass-through option is supported in localized language translations.

    - Select **Never** to use the Proxy DN credentials for all operations.
    - Select **BindOnly** to pass user credentials to the proxied LDAP server for bind only and use the default server credentials for all other operations.
    - Select **Always** to pass user credentials presented to Oracle Virtual Directory to the proxied LDAP server for all operations.

    Note: In some situations when pass-through mode is set to **Always**, the LDAP Adapter may still use the Proxy DN. This occurs when the user credential cannot be mapped, for example, from another adapter namespace, or if it is the root account.

    If defining multiple adapters to different domain controllers within a Microsoft Active Directory forest, you can program the LDAP Adapter to proxy credentials from other adapters (that is, two or more adapters pointing to the same Active Directory forest) by using the **Routing Bind-Include** setting.
14. Select the **Use Kerberos** option to configure the LDAP Adapter to perform LDAP bind operations using the Kerberos protocol. Oracle recommends using Java 1.6 or higher if you enable the Use Kerberos setting to resolve many known issues with the Microsoft Active Directory version of Kerberos.

    If you enable the **Use Kerberos** option:

    - The Pass Through option must be set to **BindOnly** because the Kerberos authentication can only be used to validate credentials and not passed to the back-end server for any other operation.
    - The RDN value must be the same as the Kerberos principal name, for example, sAMAccountName in Active Directory. This may mean that the bind DN for a Kerberos bind is not the actual user DN. For example, if the user DN is `cn=Jane Doe,cn=users,dc=mycompany,dc=com` but the sAMAccountName is `jdoe`, the bind DN with the Use Kerberos option enabled is `cn=jdoe,cn=users,dc=mycompany,dc=com`.
    - You must create a krb5.conf file and place it in the Oracle Virtual Directory's configuration folder. The krb5.conf has the following properties:

Table 12–1 Properties in the krb5.conf File

Property	Description
default_realm	The default domain used if not supplied by the mapping. For example, if a user binds as `uid=jsmith,ou=people,dc=myorg,dc=com`, this will be treated as `jsmith@myorg.com`. If the mapped namespace does not include a domain component (dc) based root, this value will be substituted instead.
domain_realm	Defines a mapping between a domain and a realm definition. For example: `.oracle.com = ORACLE.COM`
realms	Defines one or more realms, for example: `ORACLE.COM = {...}`
kdc	The DNS name of the server running the Kerberos service for a particular realm definition.

Kerberos binds use the Kerberos libraries provided in the standard Java package. The Kerberos libraries use the krb5.conf file, which is not currently synchronized with Oracle Virtual Directory LDAP Adapter settings. Kerberos fail-over is controlled by the default libraries. Refer to Sun Microsystem's Java documentation for more information on fail-over and advanced krb5.conf file configurations.

Note: If a Microsoft Active Directory server is in the process of shutting down (either stopping or rebooting) and Oracle Virtual Directory tries to connect to it, Active Directory may not validate the credential and may return a `Client not Found in Kerberos Database` error message instead of returning a Key Distribution Center (Domain Controller) connection error.

The end-user should attempt to login again and assuming that either the Active Directory server is available or Key Distribution Center fail-over is enabled, successful authentication should be returned.

---

15. If you enable the **Use Kerberos** option, you can use the **Kerberos Retry** option to control whether or not Oracle Virtual Directory should retry logging in after failed authentication attempts. If you enable the **Kerberos Retry** option and authentication fails, Oracle Virtual Directory reloads the kerb5.conf file and retries the log in.

Note: If you identified multiple Active Directory servers in a single Kerberos realm in the krb5.conf file, do not enable the **Kerberos Retry** option, as enabling the retry may disrupt fail-over functionality.

---

16. Click **Next** on the Name Space screen. The Summary screen appears listing the settings for the new LDAP Adapter.
17. Review the settings for the new LDAP Adapter and click **Finish** to create the LDAP Adapter. The new LDAP Adapter appears in the Adapter tree.

After you create the LDAP Adapter you can configure it using the procedures in Configuring LDAP Adapters.

12.1.1 Configuring LDAP Adapters

This section describes how to configure LDAP Adapter settings, including:

- Configuring LDAP Adapter General Settings
- Configuring Adapter Routing
- Configuring Adapter Plug-ins and Mappings
- Managing Certificate Authorities for LDAP Adapters Secured by SSL

12.1.1.1 Configuring LDAP Adapter General Settings

After you create the LDAP Adapter you can configure the general settings for the adapter by clicking the adapter name in the Adapter tree, clicking the **General** tab, setting values for the following fields, and clicking **Apply**:

Root

This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in this field will be the base DN value that returned entries will have. For example, if you enter dc=mydomain,dc=com in the field, all entries will end with dc=mydomain,dc=com.

Active

An adapter can be configured as active (enabled) or inactive (disabled). An adapter configured as inactive will not start during a server restart or an attempted adapter start. Use the inactive setting to keep old configurations available or in stand-by without having to delete them from the configuration. The default setting is active (enabled).

LDAP Server Details

Perform the following procedures to configure the proxy LDAP host information in the LDAP Servers table in the General tab. Each proxy LDAP host must provide equivalent content, that is, must be replicas.

Be careful when specifying only a single host for proxying. Without a failover host, the LDAP Adapter cannot automatically fail over to another host. A single host is suitable when Oracle Virtual Directory is connected to a logical LDAP service via a load balancing system.

Note: The information in the LDAP Servers table is used only if you set the Use DNS for Auto Discovery parameter to **No**.

To add a proxy LDAP host to the adapter:

1. Click the **Add Host** button.
2. Enter the IP Address or DNS name of the LDAP host to proxy to in the Hosts field.

Note: Oracle Virtual Directory 11*g* Release 1 (11.1.1) supports IPv6. If your network supports IPv6 you can use a literal IPv6 address in the Hosts field to identify the proxied LDAP host.

---

3. Enter the port number the proxied LDAP host provides LDAP services on in the Port field.

4. Enter a number between 0 and 100 in the Percentage field to configure the load percentage to send to the host. If the combined percentages for all of the hosts configured for the adapter do not total 100, Oracle Virtual Directory automatically adjusts the load percentages by dividing the percentage you entered for a host by the total percentage of all hosts configured for the adapter. For example, if you have three hosts configured for the adapter at 20 percent, 30 percent, and 40 percent, Oracle Virtual Directory adjusts the 20 to 22 (20/90), the 30 to 33 (30/90), and the 40 to 44 (40/90).
5. Select the Read-only option to configure the LDAP Adapter to only perform search operations on the LDAP host. The LDAP Adapter automatically directs all modify traffic to read/write hosts in the list.

To delete a proxy LDAP host from the adapter:

1. Click anywhere in the row of the host you want to delete in the Remote Host table.
2. Click the **Delete** button. A confirmation dialog box appears.
3. Click Confirm to delete the proxy LDAP host from the adapter.

To validate a proxy LDAP host connection:

1. Click anywhere in the row of the Remote Host table for the host you want to validate the connection for.
2. Click the **Validate** button. The connection to the proxy LDAP host must be validated for the adapter to proxy the LDAP host.

Use SSL/TLS

Enabling this option secures the communication between the LDAP Adapter and the proxy LDAP hosts using SSL/TLS.

> **See:** "Managing Certificate Authorities for LDAP Adapters Secured by SSL" on page 12-11 for information on Certificate Authorities.

SSL Authentication Mode

If you select (enable) the **Use SSL/TLS** option, choose the SSL authentication mode to use for securing the adapter by selecting an option from the SSL Authentication Mode list. The SSL Authentication Mode setting is functional only when the Use SSL/TLS option is enabled.

Failover Mode

If set to **Sequential**, the first host specified in LDAP Servers table will be used unless a failure occurs. If a failure occurs, the next host is tried. Sequential failover is often used for fail-over between geographies. In sequential failover, the LDAP Adapter attempts to use the designated host until it fails. At this point, it would fail-over to an equivalent host available in another data center or continent.

If set to **Distributed**, each new connection made will be load balanced through the list defined by the LDAP Servers table. Distributed failover is most often used when proxying a set of LDAP hosts that are typically in the same data center or are equally available in terms of network performance.

Note: If a remote host's network fails, a delay of several minutes may occur in Oracle Virtual Directory because of platform specific TCP socket timeout settings. However, Oracle Virtual Directory failover is operating properly and no data is lost during the delay.

Extended Trying

Enable this option to force the Oracle Virtual Directory server to continue trying to connect to the last host listed in the LDAP Servers table for new incoming requests on the adapter even after it has been determined that the connection to the host failed. When enabled, the adapter's **Heartbeat Interval** setting is ignored regardless if a connection to the host has failed and the host will not be removed from the LDAP Servers table. Some environments with distributed directories may prefer to disable the **Extended Trying** option in conjunction with the **Routing Critical** setting to quickly return partial results at that time. The default setting is enabled.

Heartbeat Interval

The LDAP Adapter periodically verifies the availability of each the hosts defined in the LDAP Servers table. Any currently disabled host can be resurrected or a currently active host that fails the TCP/IP connection test is labelled as **false** during this verification cycle. The Heartbeat Interval parameter specifies the number of seconds between verification passes. Setting a value too low can cause unnecessary connections to the remote directory. Setting a value too high can mean extended time for recovery detection in the event of a failure. For production environments, Oracle suggests starting with a value of 60 seconds, then making adjustments as needed.

Operation Timeout

The amount of time in milliseconds the server will wait for an LDAP request to be acknowledged by a remote host. If the operation fails, the LDAP Adapter automatically tries the next server in the Remote Host table. The minimum configurable value is 100. Settings that are too low can cause erroneous failures on busy servers. For production environments, Oracle suggests starting with a value of 5000, which is 5 seconds, then making adjustments as needed.

Max Pool Connections

A tuning parameter that allows you to control how many simultaneous connections can be made to a single server. For production environments, Oracle suggests starting with a value of 10 connections, then making adjustments as needed.

Max Pool Wait

The maximum amount a time in milliseconds that an LDAP operation will wait to use an existing connection before causing the LDAP Adapter to generate a new connection. For production environments, Oracle suggests starting with a value of 1000, which is 1 second, then making adjustments as needed.

Max Pool Tries

Maximum number of times an operation will wait for an LDAP connection before overriding the Max Pool Connections parameter to generate a new connection. Maximum time is a function of multiplying Max Pool Wait time by the number of tries. If pool wait is 1 second, and 10 is the maximum number of tries, then if after 10 seconds an LDAP connection is not available in the normal pool, the pool will be expanded to handle the extended load. To prevent pool expansion beyond Max Pool Connections, set the number of tries to a high number. For production environments, Oracle suggests starting with a value of 10, then making adjustments as needed.

Use Kerberos

Refer to step 14 on page 12-4 for information about the **Use Kerberos** option.

Kerberos Retry

If you enable the **Use Kerberos** option, you can use the **Kerberos Retry** option to control whether or not Oracle Virtual Directory should retry logging in after failed

authentication attempts. If you enable the **Kerberos Retry** option and authentication fails, Oracle Virtual Directory reloads the kerb5.conf file and retries the log in.

> **Note:** If you identified multiple Active Directory servers in a single Kerberos realm in the krb5.conf file, do not enable the **Kerberos Retry** option, as enabling the retry may disrupt fail-over functionality.

Use DNS For Auto Discovery

Instead of configuring specific proxy LDAP hosts in the LDAP Servers table, you can use this option to instruct Oracle Virtual Directory to use DNS to locate the appropriate LDAP servers for the remote base defined, also known as serverless bind mode. The LDAP Adapter supports the following modes of operation:

- **No**: Use the LDAP Servers table configuration—no serverless bind
- **Standard**: Use standard DNS lookup for a non-Microsoft server. All servers are marked as read-write, so enabling the **Follow Referrals** setting is advised to allow for LDAP write support.
- **Microsoft**: The DNS server is a Microsoft dynamic DNS and also supports load-balancing configuration. If proxying to a Microsoft dynamic DNS server, this is preferred setting because of Oracle Virtual Directory's ability to auto-detect read/write servers compared to read-only servers.

> **Note:** Remote base should have a domain component style name when using this setting, for example, dc=myorg,dc=com. This enables Oracle Virtual Directory to locate the LDAP hosts within the DNS service by looking up myorg.com.

The following fields appear in the Settings section of the General tab:

Remote Base

The location in the remote server directory tree structure to which the local Oracle Virtual Directory root suffix corresponds. This is the location in the remote directory under which Oracle Virtual Directory will execute all searches and operations for the current adapter. The LDAP Adapter applies an automatic mapping of all entries from the remote base to the adapter root base.

DN Attributes

List of attributes to be treated as DNs for which namespace translation will be required, such as member, uniquemember, manager. For example, when reading a group entry from a proxied directory, Oracle Virtual Directory will automatically convert the DN for the group entry itself as well as the uniquemember or member attributes if these attributes are in the DN Attributes list.

> **Note:** Translate only those attribute you know will need to be used by the client application. Entering all possible DN attributes may not be necessary and can consume some a small amount of additional CPU time in the proxy.

To add attributes to the DN Attributes list:

1. Click **Add**. The Select DN Attribute dialog box appears.

2. Select the attribute you want to add.
3. Click **OK**.

Escape Slashes

When a / character is encountered in a directory, Oracle Virtual Directory can optionally escape the slashes with back-slashes \ character. Some directory server products accept un-escaped slashes, while others will reject them. Selecting this setting enables escaping of slashes.

Follow Referrals

Enabling this setting causes the LDAP Adapter to follow (chase) referrals received from a source directory on the client's behalf. If disabled, the referral is blocked and not returned to the client.

The following list summarizes the LDAP Adapter's behavior with different settings in relation to the send managed DSA control in LDAP operations setting:

- If the LDAP Adapter's Follow Referrals is set to **Enabled (true)**, and Send Managed DSA Control in LDAP Operations is also set to **True**, Oracle Virtual Directory does not chase the referral entries, but it returns them back to the client.
- If the LDAP Adapter's Follow Referrals is set to **Enabled (true)**, but Send Managed DSA Control in LDAP Operations is set to **False**, Oracle Virtual Directory chases the referral entries.
- If the LDAP Adapter's Follow Referrals is set to **Disabled (false)**, but Send Managed DSA Control in LDAP Operations is set to **True**, Oracle Virtual Directory does not chase the referral entries, but it returns them back to the client.
- If the LDAP Adapter's Follow Referrals is set to **Disabled (false)**, and Send Managed DSA Control in LDAP Operations is also set to **False**, Oracle Virtual Directory does not chase the referral entries and does not return them back to client.

Proxied Page Size

If enabled, this setting allows the proxy to use the paged results control with a proxied directory. Enabling this setting is most often used when a directory limits the number of results in a query. This setting is used on behalf of and transparently to Oracle Virtual Directory's clients.

The following fields appear in the Credential Processing section of the General tab:

Proxy DN

The default DN the LDAP Adapter will bind with when accessing the proxied directory. Depending on the **Pass-through Mode** setting, this DN will be used for all operations, or only for exceptional cases such as pass-through mode. The form of the distinguished name should be in the form of the remote directory. Empty values will be treated as Anonymous.

Proxy Password

The authentication password to be used with the **Proxy DN** value. To set the password, enter a value in clear text. When loaded on the server, the value is automatically hashed with a reversible mask to provide additional security, for example, {OMASK}jN63CfzDP8XrnmauvsWs1g==.

Pass-through Mode

To pass user credentials presented to Oracle Virtual Directory to the proxied LDAP server for all operations, set to **Always**. To pass user credentials to the proxied LDAP

server for bind only and use the default server credentials for all other operations, set to **Bind Only**. To use the Proxy DN credentials for all operations, set to **Never**.

> **Note:** In some situations when pass-through mode is set to **Always**, the LDAP Adapter may still use the Proxy DN. This occurs when the user credential cannot be mapped, for example, from another adapter namespace, or is the root account.
>
> If defining multiple adapters to different domain controllers within a Microsoft Active Directory forest, you can program the LDAP Adapter to proxy credentials from other adapters (that is, two or more adapters pointing to the same Active Directory forest) by using the **Routing Bind-Include** setting.

The following fields appear in the Ping Protocol Settings section of the General tab:

The Ping Protocol Settings provide options for how to determine when a source LDAP directory server that is not responding becomes available. If multiple source directory servers are configured, Oracle Virtual Directory identifies the non-responsive servers and performs subsequent operations against the next available server.

Ping Protocol

Select either **TCP** or **LDAP** as the protocol Oracle Virtual Directory should use to ping source directory servers. Select **LDAP** if the source directory server is using SSL.

> **Note:** While the **TCP** protocol option is faster than the **LDAP** option, it may produce an inaccurate response from the source directory server if its network socket is available, but its LDAP server process is unavailable.

Ping Bind DN

If you select **LDAP** as the Ping Protocol, identify the DN to use for the LDAP bind.

Ping Bind Password

If you select **LDAP** as the Ping Protocol, identify the password for the DN specified in the Ping Bind DN setting.

12.1.1.2 Configuring Adapter Routing

After you create the adapter you can configure routing for the adapter by clicking the adapter name in the Adapter tree, clicking the **Routing** tab, and referring to "Understanding Routing Settings" on page 3-3.

12.1.1.3 Configuring Adapter Plug-ins and Mappings

After you create the adapter you can apply Plug-ins and Mappings to the adapter by clicking the adapter name in the Adapter tree, clicking the **Plug-Ins** tab, and referring to "Managing Adapter Plug-ins" on page 13-1 and "Applying Mappings to Adapters" on page 14-3.

12.1.1.4 Managing Certificate Authorities for LDAP Adapters Secured by SSL

In some situations, SSL connections from Oracle Virtual Directory to the SSL port of an LDAP Adapter can fail and the following message may appear:

```
Oracle Virtual Directory could not load certificate chain
```

Two examples of situations when this may happen are when:

- you create a new LDAP Adapter secured by SSL and use an untrusted Certificate Authority
- a certificate for an existing LDAP Adapter secured by SSL expires and the new certificate is signed by an untrusted Certificate Authority

To resolve this issue, import the LDAP server certificate *and* the Root Certificate Authority certificate used to sign the LDAP server certificate, into the Oracle Virtual Directory server so it knows the certificates are trusted.

Use the following `keytool` command and an appropriate alias **all on one command line**:

```
ORACLE_HOME/jdk/jre/bin/keytool -import -trustcacerts
-alias "NEW_CA" -file PATH_TO_CA_CERTIFICATE
-keystore ORACLE_INSTANCE/config/OVD/ovd1/keystores/adapters.jks
```

Using LDAP Adapters with Microsoft Active Directory and Microsoft Certificate Services

By default, Microsoft Certificate Services will automatically update expired Active Directory SSL certificates. However, client applications are not normally notified of this change. If this happens, the Oracle Virtual Directory LDAP Adapter connected to an updated Active Directory server will stop functioning. If this occurs, use Oracle Directory Services Manager to configure the LDAP Adapter to import trusted certificates and the adapter should begin to function again.

12.1.2 Configuring a Mutual Authentication SSL Connection Between Oracle Virtual Directory and Oracle Internet Directory

Perform the following steps to configure a mutual authentication SSL connection between Oracle Virtual Directory and Oracle Internet Directory:

1. Create and configure an LDAP Adapter for Oracle Internet Directory by referring to Creating LDAP Adapters and Configuring LDAP Adapters. When you configure the adapter, set it to use a non-SSL port number.
2. If *ORACLE_INSTANCE*/config/OVD/*ovd1*/adapters.jks does not exist, create it with a self-signed certificate to store the trusted certificates by using the following command:

   ```
   ORACLE_HOME/jdk/jre/bin/keytool -genkey \
   -keystore ORACLE_INSTANCE/config/OVD/ovd1/keystores/adapters.jks \
   -storepass password -alias alias -keyalg rsa -dname DN
   ```

   Note: The *DN* identified by the `-dname` option in the preceding command is the DN that Oracle Virtual Directory uses to act as a client to Oracle Internet Directory.

   A user entry corresponding to this DN must exist (or must be created) on Oracle Internet Directory in order for SSL mutual authentication to work.

3. Export the Oracle Internet Directory server certificate in Base64 format using the following command:

```
orapki wallet export -wallet LOCATION_OF_OID_WALLET \
-dn DN_FOR_OID_SERVER_CERTIFICATE -cert ./b64certificate.txt
```

> **Note:** If you use a certificate alias in the orapki command, you will receive an error if the alias is not in all lower case letters.

4. Import the Oracle Internet Directory server certificate created in step 2 to the Oracle Virtual Directory keystore as a trusted entry using the following command:

```
ORACLE_HOME/jdk/jre/bin/keytool -importcert \
-keystore ORACLE_INSTANCE/config/OVD/ovd1/keystores/adapters.jks \
-storepass password -alias alias -file b64certificate.txt -noprompt
```

5. Export the Oracle Virtual Directory server certificate in Base 64 format using the following command:

```
ORACLE_HOME/jdk/jre/bin/keytool -exportcert \
-keystore ORACLE_INSTANCE/config/OVD/ovd1/keystores/adapters.jks \
-storepass password -rfc -alias alias -file cert.txt
```

6. Import the Oracle Virtual Directory server certificate to the Oracle Internet Directory wallet as a trusted certificate. Execute the following command from the Oracle Internet Directory wallet directory:

```
orapki wallet add -wallet ./ewallet.p12 -cert cert.txt
-trusted_cert -pwd password
```

> **Note:** If you use a certificate alias in the orapki command, you will receive an error if the alias is not in all lower case letters.

7. Using Oracle Directory Services Manager, update the LDAP Adapter for Oracle Internet Directory as follows:
   - Select (enable) the **Use SSL/TLS** option
   - Change the port number to an SSL port number
   - Click the **Apply** button to save the changes to the adapter.
8. Restart the Oracle Virtual Directory server.

12.2 Creating Database Adapters

This topic explains how to create and configure Database Adapters and includes the following sections:

- Creating Database Adapters for Oracle RAC Database
- Creating Database Adapters for Oracle TimesTen In-Memory Database
- Configuring Database Adapters

Perform the following steps to create Database Adapters using Oracle Directory Services Manager:

Note: Before you create a Database Adapter for a non-Oracle database for the first time, you must first load the database's drivers into Oracle Virtual Directory. Refer to "Loading Libraries into the Oracle Virtual Directory Server" on page 9-12 for information on loading drivers into the Oracle Virtual Directory server.

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the **Create Adapter** button. The New Adapter Wizard appears.
4. Perform the following steps to define the Type of adapter:
   a. Select **Database** from the Adapter Type list.
   b. Enter a unique name for the Database Adapter in the Adapter Name field. The adapter name value is used in other configuration fields that need to reference the adapter.
   c. Select **Default** from the Adapter template list unless you are integrating Oracle Virtual Directory with Oracle Access Manager. Refer to "Understanding Adapter Templates" on page 2-28 for more information.

      Note: After selecting an adapter template, Oracle Directory Services Manager populates default values for some of the adapter settings. You should alter these default settings according to your environment.

   d. Click **Next**. The Connection screen appears.
5. Enter a valid base DN (in DN format) in the Adapter Suffix/Namespace field. This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in this field will be the base DN value that returned entries will have. For example, if you enter dc=mydomain,dc=com in the field, all entries will end with dc=mydomain,dc=com.
6. Select one of the following options from the URL Type list. Some of the steps to create a Database Adapter differ depending on which option you choose. After selecting one of the following options, continue this procedure by following the alphabetic numbered steps for each option.
   - **Use Predefined Database**: Select this option to connect to a predefined database. The predefined databases appear in the Database Type list after selecting Use Predefined Database from the URL Type list. If you are unsure if Oracle Virtual Directory has predefined your type of database, select Use Predefined Database from the URL Type list and check to see if your database is listed in the Database Type list. If your database is listed in the Database Type list, continue with the following steps. If your database is not listed, select **Use Custom URL** from the URL Type list and perform the steps for using a custom URL.
     a. Select the type of your of database from the Database Type list. After selecting the database type, the JDBC Driver Class and Database URL fields are populated with the appropriate information for the database.
     b. Enter the IP Address or DNS host name of the database in the Host field.

c. Enter the port number the database listens on in the Port field.

d. Enter the name of the database, for example, the Oracle SID, in the Database Name field.

e. Enter the user name that the Database Adapter should use to connect the database in the Database User field.

f. Enter the password for the user name you entered in the Database User field in the Password field. Oracle Virtual Directory replaces the value you enter in this field with a reversible masked value upon startup.

g. Click **Next**. The Map Database Tables screen appears. Continue this procedure by going to step 7 now.

- **Use Custom URL**: Select this option to connect Oracle Virtual Directory to a custom database.

  a. In the JDBC Driver Class field, enter the JDBC driver class name for the database.

  b. In the Database URL field, enter the URL that Oracle Virtual Directory should use to access the database.

  c. In the Database User field, enter the user name that the Database Adapter should use to connect the database.

  d. In the Password field, enter the password for the user name you entered in the Database User field. Oracle Virtual Directory replaces the value you enter in this field with a reversible masked value upon startup.

  e. Click **Next**. The Map Database Tables screen appears. Continue this procedure by going to step 7 now.

7. Identify the database tables the Database Adapter should use in the Map Database Tables field by entering the name of the table file, or by clicking **Browse**, navigating to the table file, selecting it, and clicking **OK**. Click **Next** on the Map Database Tables screen to proceed. The Map Object Classes screen appears.

   Note: If you do not define an object class in step 8, the information you entered in the Map Database Tables field will not be saved.

8. In the Map Object Classes field, define the object classes and their RDNs that map to the database tables. Click the **Create Object Class** button. The New Object Class Mapping dialog box appears allowing you to define the objectclass and their corresponding RDNs. Enter the following information:

   a. Select the appropriate object class for the Object Class list.

   b. Enter the RDN for the object class in the RDN field.

   c. Click **OK**. The object class and the RDN appear in the Object Class table.

   Note: You can create nested object classes by entering an existing object where the RDN of the nested class must be an attribute of the child object class. For example, you could create parent organization units for records in a table about people where location information is available that can be used to drive the organization unit (ou) information.

9. Map LDAP attributes for the object class and RDNs to the database table and fields. You must map LDAP attributes for the object class RDN value. You do not have to map every LDAP attribute required by the LDAP schema for the selected object class.

   Click the appropriate object class in the Object Class table and then click the **Add Mapping Attribute** button on the Attributes Mapping table. Enter the following information.

   - Select the LDAP attribute value for the object class from the LDAP Attribute list.
   - Select the appropriate database table and field from the Database Table:Field list.
   - Optionally, select a description for the attribute type from the Data Type list.

   Note: You must select BLOB from the Data Type list if you are mapping an attribute to a BLOB column in the database.

10. Click **Next** on the Map Object Class Mapping screen after defining all the object classes and attribute mappings. The Summary screen appears listing the settings for the Database Adapter.
11. Review the Database Adapter settings and click **Finish** to create the Database Adapter. The new Database Adapter appears in the Adapter tree.

When the adapter starts, Oracle Virtual Directory will connect to the database and retrieve all defined LDAP attributes and their corresponding table and column information to reconcile the attributes with the defined LDAP schema. If a mapped LDAP attribute is already defined, it will attempt to create a mapping from the database source format to the target LDAP schema format. If the LDAP attribute is not defined, the Database Adapter will temporarily add an attribute to the server schema that most closely maps to the database format (this definition is not added to the permanent Oracle Virtual Directory schema configuration).

After you create the Database Adapter, you can configure it using the procedures in Configuring Database Adapters.

12.2.1 Creating Database Adapters for Oracle RAC Database

To create a Database Adapter for use with Oracle RAC Database, perform the procedure in "Creating Database Adapters" on page 12-13, but when you configure the connection to the RAC database on the Connection screen:

- Select **Use Custom URL** from the URL Type list.
- In the Database URL field, enter the URL to connect to the RAC database, such as:

```
jdbc:oracle:oci:@(DESCRIPTION=(ADDRESS_LIST=(LOAD_
BALANCE=ON)(ADDRESS=(PROTOCOL=TCP)(HOST=host-name-1)(PORT=1521))(ADDRESS=
(PROTOCOL=TCP)(HOST=host-name-2)(PORT=1521)))(CONNECT_
DATA=(SERVER=DEDICATED)(SERVICE_NAME=database-service-name)))
```

Note: The Oracle Virtual Directory Database Adapter does not support Fast Connection Failover (FCF) for Oracle RAC. However, after a RAC instance failure, Oracle Virtual Directory reconnects to a surviving RAC instance.

12.2.2 Creating Database Adapters for Oracle TimesTen In-Memory Database

Perform the following steps to create a Database Adapter for use with Oracle TimesTen In-Memory Database:

1. If native Oracle TimesTen libraries are not accessible to Oracle Virtual Directory, you must install the Oracle TimesTen In-Memory Database client.
2. In Oracle Virtual Directory's opmn.xml file, add the location of the Oracle TimesTen libraries and add the location of the Oracle TimesTen JDBC driver to the class-path. The opmn.xml file is located in the following directory:

   ORACLE_INSTANCE/config/OPMN/opmn/

   To set the location of the Oracle TimesTen libraries:

   Add the LD_LIBRARY_PATH environment variable for UNIX and Linux platforms, or add the PATH environment variable on Windows.

   For example, on UNIX and Linux platforms, you add the LD_LIBRARY_PATH environment variable as follows, where *TIMESTEN_HOME* represents the directory where you installed the Oracle TimesTen software:

Example 12–1 Setting the Location of the Oracle TimesTen Libraries on UNIX/Linux

```
<ias-component id="ovd1">
   <process-type id="OVD" module-id="OVD">
      <environment>
         <variable id="TNS_ADMIN" value="$ORACLE_INSTANCE/config"/>
         <variable id="LD_LIBRARY_PATH" value="/TIMESTEN_HOME/lib" append="true"/>
      </environment>
```

   To add the location of the Oracle TimesTen JDBC driver to the class-path:

   Set the java-classpath to include the path to the TimesTen JDBC Driver as follows, where *TIMESTEN_HOME* represents the directory where you installed the Oracle TimesTen software:

Example 12–2 Adding the Location of the Oracle TimesTen JDBC Driver to the class-path

```
<module-data>
   <category id="start-options">
      <data id="java-bin" value="$ORACLE_HOME/jdk/bin/java"/>
      <data id="java-options" value="-server -Xms512m -Xmx512m
-Dvde.soTimeoutBackend=0 -Doracle.security.jps.config=$ORACLE_
INSTANCE/config/JPS/jps-config-jse.xml"/>
      <data id="java-classpath" value="$ORACLE_HOME/ovd/jlib/vde.jar$:$ORACLE_
HOME/jdbc/lib/ojdbc6.jar:/TIMESTEN_HOME/lib/ttjdbc6.jar"/>
   </category>
</module-data>
```

3. Reload the configuration to OPMN, and stop, then start Oracle Virtual Directory. For example:

To reload the configuration to OPMN, execute:

```
ORACLE_INSTANCE/bin/opmnctl reload
```

To stop Oracle Virtual Directory, execute:

```
ORACLE_INSTANCE/bin/opmnctl stopproc ias-component=NAME_OF_OVD_COMPONENT
```

To start Oracle Virtual Directory, execute:

```
ORACLE_INSTANCE/bin/opmnctl startproc ias-component=NAME_OF_OVD_COMPONENT
```

4. Create a Database Source Name (DSN) for Oracle TimesTen. Refer to the *Oracle TimesTen Operations Guide* on the Oracle Technology Network web site for more information.
5. Create the Database Adapter for Oracle TimesTen using Oracle Directory Services Manager. When you create the Database Adapter for Oracle TimesTen:

If the adapter is for an Oracle TimesTen client-only installation:

a. Select the **Use Custom URL** option from the URL Type list on the Connection screen of the New Database Adapter Wizard.

b. Enter the following in the JDBC Driver Class field:

```
com.timesten.jdbc.TimesTenDriver
```

c. In the Database URL field, enter the following and replace *DSN* with the Database Source Name you created in step 4:

```
jdbc:timesten:client:dsn=DSN
```

d. Continue creating the adapter by referring to the "Creating Database Adapters" section of the *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

If the adapter is for an Oracle TimesTen client and server installation:

a. Select the **Use Predefined Database** option from the URL Type list on the Connection screen of the New Database Adapter Wizard.

b. Choose **Oracle - Times-Ten** from the Database Type list.

c. Select the **Use Custom URL** option from the URL Type list.

d. In the Database URL field, enter the following and replace *DSN* with the Database Source Name you created in step 4:

```
jdbc:timesten:direct:dsn=DSN
```

e. Continue creating the adapter by referring to the "Creating Database Adapters" section of the *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

Note: The **Enable Case Insensitive Search** option, as described in the "Configuring Database Adapter General Settings" section of the *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*, can be used to improve Database Adapter performance during searches on case-insensitive LDAP attributes, such as uid, for Oracle TimesTen databases.

In addition to enabling the **Enable Case Insensitive Search** option, the linguistic indexes for the database columns used in the search must be created in the database. Refer to the *Oracle Database Globalization Support Guide* for information about Oracle TimesTen database linguistic indexes.

12.2.3 Configuring Database Adapters

This section describes how to configure Database Adapter settings, including:

- Configuring Database Adapter General Settings
- Configuring Adapter Routing
- Configuring Adapter Plug-ins and Mappings

12.2.3.1 Configuring Database Adapter General Settings

After you create the Database Adapter, you can configure the general settings for the adapter by clicking the adapter name in the Adapter tree, clicking the **General** tab, setting values for the following fields, and clicking **Apply**:

Root

This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in this field will be the base DN value that returned entries will have. For example, if you enter dc=mydomain,dc=com in the field, all entries will end with dc=mydomain,dc=com.

Active

An adapter can be configured as active (enabled) or inactive (disabled). An adapter configured as inactive will not start during a server restart or an attempted adapter start. Use the inactive setting to keep old configurations available or in stand-by without having to delete them from the configuration. The default setting is active.

The following fields appear in the Connection Settings section of the General tab:

URL Type

Select one of the following options from the URL Type list. Some of the fields for Database Adapter connection settings differ depending on which option you choose. After selecting one of the following options, continue configuring the Connection Settings by setting the fields listed for each option.

- **Use Custom URL**: Select this option to connect Oracle Virtual Directory a custom database.
  - Enter the JDBC driver class name for the database in the JDBC Driver Class field.
  - Enter the URL that Oracle Virtual Directory should use to access the database in the Database URL field.

  - Enter the user name that the Database Adapter should use to connect the database in the Database User field.
  - Enter the password for the user name you entered in the Database User field in the Password field. Oracle Virtual Directory replaces the value you enter in this field with a reversible masked value upon startup.

- **Use Predefined Database**: Select this option to connect to a predefined database. The predefined databases appear in the Database Type list after selecting Use Predefined Database from the URL Type list. If you are unsure if Oracle Virtual Directory has predefined your type of database, select Use Predefined Database from the URL Type list and check to see if your database is listed in the Database Type list. If your database is listed in the Database Type list, continue with the following steps. If your database is not listed, select **Use Custom URL** from the URL Type list and perform the steps for using a custom URL.
  - Select the type of your database from the Database Type list. After selecting the database type, the JDBC Driver Class and Database URL fields are populated with the appropriate information for the database.
  - Enter the IP Address or DNS host name of the database in the Host field.
  - Enter the port number the database listens on in the Port field.
  - Enter the name of the database, for example, the Oracle SID, in the Database Name field.
  - Enter the user name that the Database Adapter should use to connect the database in the Database User field.
  - Enter the password for the user name you entered in the Database User field in the Password field. Oracle Virtual Directory replaces the value you enter in this field with a reversible masked value upon startup.

The following fields appear in the Settings section of the General tab:

Ignore Modify Objectclass

Since objectclasses in the database are logical objects and do not map directly to a table column in the mapping, modifications to the objectclass attribute can cause errors. If the **Ignore Modify Objectclasses** option is enabled, the Database Adapter removes any references to the objectclass attribute so that errors will *not* be sent to the client application, that is, they will be ignored. If the **Ignore Modify Objectclasses** option is not selected, error messages *will* be sent to the client application

Include Object Class Super Classes

This setting causes the Database Adapter to list objectclass parent classes along with the main objectclass in the objectclass attribute. Disable this setting when you want to emulate Microsoft Active Directory server schema. For most scenarios, it is useful to enable this setting so that objectclass=xxx queries can be executed against parent objectclass values.

Enable Case Insensitive Search

Enabling (selecting) the **Enable Case Insensitive Search** option makes the search case insensitive for case insensitive LDAP attributes, such as uid. Oracle Virtual Directory uses UPPER in the SQL query when **Enable Case Insensitive Search** is enabled. If the database cannot maintain functional indexes, such as for Oracle TimesTen or MySQL databases, then you should disable the **Enable Case Insensitive Search** option. When the **Enable Case Insensitive Search** is disabled, Oracle Virtual Directory will perform case sensitive searches and will not use UPPER in the SQL query. The default value for **Enable Case Insensitive Search** is Enable.

Maximum Connections
This setting defines the maximum connections the Database Adapter may make with the database.

Connection Wait Timeout
This setting determines how much time (in seconds) the Database Adapter should wait before timing-out when trying to establish a connection with the database.

The following fields appear in the DB/LDAP Mapping section of the General tab:

Used Database Tables
This field displays the database tables the Database Adapter is set to use. To add a database table, click the **Add** button, navigate to the table file, select it and click **OK**.

The following fields appear in the Object Classes section of the General tab:

Object Classes
This field displays object classes and their RDNs that map to the database tables. To add an Object Class Mapping, click the **Create** button, select the appropriate object class from the Object Class list, enter an RDN value for the object class in the RDN field, and click **OK**.

12.2.3.2 Configuring Adapter Routing
After you create the adapter you can configure routing for the adapter by clicking the adapter name in the Adapter tree, clicking the **Routing** tab, and referring to "Understanding Routing Settings" on page 3-3.

12.2.3.3 Configuring Adapter Plug-ins and Mappings
After you create the adapter you can apply Plug-ins and Mappings to the adapter by clicking the adapter name in the Adapter tree, clicking the **Plug-Ins** tab, and referring to "Managing Adapter Plug-ins" on page 13-1 and "Applying Mappings to Adapters" on page 14-3.

12.3 Creating Local Store Adapters
This topic explains how to create and configure Local Store Adapters and includes the following sections:

- Configuring Local Store Adapters

Perform the following steps to create Local Store Adapters using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the **Create Adapter** button. The New Adapter Wizard appears.
4. Perform the following steps to define the Type of adapter:
   a. Select **Local Store** from the Adapter Type list.
   b. Enter a unique name for the Local Store Adapter in the Adapter Name field. The adapter name value is used in other configuration fields that need to reference the adapter.
   c. Select an adapter template from the Adapter Template list by referring to "Understanding Adapter Templates" on page 2-28. Use the **Default** template if you are unsure which template to use.

> **Note:** After selecting an adapter template, Oracle Directory Services Manager populates default values for some of the adapter settings. You should alter these default settings according to your environment.

d. Click **Next**. The Settings screen appears.

5. Enter a valid base DN (in DN format) in the Adapter Suffix/Namespace field. This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in this field will be the base DN value that returned entries will have. For example, if you enter dc=mydomain,dc=com in the field, all entries will end with dc=mydomain,dc=com.
6. Select the Create Adapter Suffix option to create a base entry in the Local Store Adapter using the value specified in the Adapter Suffix/Namespace field.

> **Note:** If you enable the Create Adapter Suffix option, an Objectclass screen appears after you click **Next** on the Settings screen. When the Objectclass screen appears, select an Objectclass for the base entry in the Local Store Adapter.

7. Enter the path, relative to the Oracle Virtual Directory installation, and a unique file name prefix for the Local Store Adapter data files in the Database File field. For example, a valid name may be data/localDB. If you are using more than one Local Store Adapter, this value must be unique for each adapter or data-corruption will occur.
8. Enter the size for the Local Store Adapter cache in the Cache Size field. The Cache Size option determines the number of entries the Local Store Adapter will cache, which always contains the last entries accessed or written. The size of the entries will determine how much memory you need.

> **Note:** Storing very large entries, for example, groups or binary objects, this may cause Oracle Virtual Directory to consume more memory than normal. You may need to increase the overall memory available to the Oracle Virtual Directory.

9. Select the password hash type by choosing an option from the Password Hash Mode list. The most secure algorithm is **SSHA**, however, others are available for compatibility purposes. Selecting **PLAIN** leaves the password valued un-hashed in the internal Local Store Adapter data store.
10. Enter the path, relative to the Oracle Virtual Directory installation, and a unique file name in the Backup File field in which automatic backups should be stored. For example, a valid backup file may be backup/localDB. The backup file name should be unique to the Local Store Adapter to prevent being over-written by another Local Store Adapter.
11. Enter the hour (0 to 23) in the Backup Time - Hour field to set the hour of the time at which the Local Store Adapter automatic backup should occur.
12. Enter the minute (0 to 59) in the Backup Time - Minute field to set the minute of the time at which the Local Store Adapter automatic backup should occur.

13. Enter the maximum number of backup files in the Max Backup Files field to keep in the backup file rotation for the Local Store Adapter.
14. Click **Next**. The Summary screen appears listing the settings for the Local Store Adapter.
15. Review the Local Store Adapter settings and click **Finish** to create the Local Store Adapter. The new Local Store Adapter appears in the Adapter tree.

After you create the Local Store Adapter you can configure it using the procedures in Configuring Local Store Adapters.

12.3.1 Configuring Local Store Adapters

This section describes how to configure Local Store Adapter settings, including:

- Configuring Local Store Adapter General Settings
- Configuring Adapter Routing
- Configuring Adapter Plug-ins and Mappings

12.3.1.1 Configuring Local Store Adapter General Settings

After you create the Local Store Adapter you can configure the general settings for the adapter by clicking the adapter name in the Adapter tree, clicking the **General** tab, setting values for the following fields, and clicking **Apply**:

Root
This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in this field will be the base DN value that returned entries will have. For example, if you enter dc=mydomain,dc=com in the field, all entries will end with dc=mydomain,dc=com.

Active
An adapter can be configured as active (enabled) or inactive (disabled). An adapter configured as inactive will not start during a server restart or an attempted adapter start. Use the inactive setting to keep old configurations available or in stand-by without having to delete them from the configuration. The default setting is active.

Read-Only
If you enable the Read-Only option the adapter does not accept modify transactions and is available for searching only. The default setting is disabled, that is, the adapter is in read/write mode.

The following fields appear in the Indexes section of the General tab:

Presence
The Presence field contains a list of attribute types whose presence in entries needs to be quickly identified, which is required for (attrname=*) style search filters to operate. To add an attribute to the list, click **Add**, select the attribute from the dialog box that appears, and click OK on the dialog box.

Exact
The Exact index field contains a list of attributes for supporting searches for exact match index, for example, sn=smith. When using the ordering index, this index is redundant. To add an attribute to the list, click **Add**, select the attribute from the dialog box that appears, and click OK on the dialog box.

Ordering
The Ordering field contains a list of attributes for enabling ordering searches, such as, sn<=Smith, exact searches, and initial substring searches, such as, sn=Smi*. LDAP filters allow only <= and >= ordering relationships. < and > are not supported in LDAPv3. To add an attribute to the list, click **Add**, select the attribute from the dialog box that appears, and click OK on the dialog box.

Substring
The Substring option is only necessary if final substring searches are necessary, for example, sn=*ith, in addition to the ordering index. Initial substring searches are often handled using the ordering index. To add an attribute to the list, click **Add**, select the attribute from the dialog box that appears, and click OK on the dialog box.

Search Un-indexed
Enables or disables low-performance searching of attributes that are not specifically indexed. If search un-indexed is disabled, searching an un-indexed attribute will return no results (that is, evaluates as false).

The following fields appear in the Security section of the General tab:

Enable Sensitive Attribute
Enables or disables sensitive attributes, which are attributes in the Local Store Adapter with encrypted values. If you enable the Enable Sensitive Attribute option, you must identify the attributes whose values will be encrypted using the Sensitive Attributes field.

Sensitive Attributes
If Enable Sensitive Attributes is selected, the values of the attributes listed in the Sensitive Attributes field will be encrypted.

The following fields appear in the Database section of the General tab:

Database File
The path relative to *ORACLE_INSTANCE*/ovd/*SYSTEM_COMPONENT_NAME* and a unique file name prefix for the Local Store Adapter data files. SYSTEM_ COMPONENT_NAME is usually ovd1. If you are using more than one Local Store Adapter, this value must be unique for each adapter or data-corruption will occur.

Password Hash Mode
Select the password hash type by choosing an option from the Password Hash Mode list. The most secure algorithm is SSHA, however, others are available for compatibility purposes. Selecting **PLAIN** leaves the password valued un-hashed in the internal Local Store Adapter data store.

Auto RDN
When adding an entry, the LDAP RFCs require that the relative distinguished name, RDN, or left most DN term, be present in the attribute list of the entry being added. Some directory product vendors ignore this and allow for the RDN value to be missing from the attribute list, which may lead to some compatibility problems with applications that depend on this behavior. By enabling Auto RDN, Oracle Virtual Directory will automatically create the missing attribute. The default setting is disabled.

Auto Compact
After a successful database backup, Oracle Virtual Directory can optionally compress the database files. If the Local Store Adapter data is being modified frequently, this will help to keep database size manageable. The default setting is disabled.

Note: On Windows platforms, it is highly recommended that the Auto Compact feature be disabled. There are some Windows scenarios where the ability to rename files is not guaranteed, which can result in corruption or loss of data.

Transaction Log Size

When a new entry is added or changed, it is first written to a transaction log to allow for faster application response, while ensuring that transactions are written to disk.

This option determines at what size (in bytes) the transaction log is truncated. Entries that have not been placed into the data store and indexed are never removed from the transaction log, even when the number of unprocessed transactions brings the log to a size that exceeds the size listed in this option.

Having a small transaction log that is continuously truncated can add considerable overhead if adding large quantities of entries. It may be better to make the transaction log as large as possible for an initial bulk load, but reduce its size afterwards, before going into production.

Cache Size

This option determines the number of entries that will be cached by the Local Store Adapter in memory. It will always contain the last entries accessed or written. The amount of memory needed is determined by the size of the entries.

Storing very large entries, for example, groups or binary objects, may cause Oracle Virtual Directory to consume more memory than normal. You may need to increase the overall memory available to the Oracle Virtual Directory.

The following fields appear in the Backup section of the General tab:

Backup File

The path relative to *ORACLE_INSTANCE*/ovd/*SYSTEM_COMPONENT_NAME* that points to a unique file name in which automatic backups should be stored. SYSTEM_COMPONENT_NAME is usually ovd1. The backup file name should be unique to the Local Store Adapter to prevent being over-written by another Local Store Adapter.

Backup Time - Hour

The hour (0 to 23) of the time at which the Local Store Adapter automatic backup should occur.

Backup Time - Minute

The minute (0 to 59) of the time at which the Local Store Adapter automatic backup should occur.

Max Backup Files

The maximum number of backup files to keep in the backup file rotation for the Local Store Adapter.

12.3.1.2 Configuring Adapter Routing

After you create the adapter you can configure routing for the adapter by clicking the adapter name in the Adapter tree, clicking the **Routing** tab, and referring to "Understanding Routing Settings" on page 3-3.

12.3.1.3 Configuring Adapter Plug-ins and Mappings

After you create the adapter you can apply Plug-ins and Mappings to the adapter by clicking the adapter name in the Adapter tree, clicking the **Plug-Ins** tab, and referring to "Managing Adapter Plug-ins" on page 13-1 and "Applying Mappings to Adapters" on page 14-3.

12.4 Creating Join View Adapters

This topic explains how to create and configure Join View Adapters and includes the following sections:

- Configuring Join View Adapters
- Configuring a Shadow Join View Adapter for Oracle Internet Directory

> **Note:** This topic assumes adapters that you want to join using a Join View Adapter already exist.

Prerequisites for Creating a Join View Adapter

Before you can create and deploy any type of Join View Adapter, you must create an adapter that will be the Join View Adapter's primary adapter. Refer to "Join View Adapter's Primary Adapter" on page 2-14 for more information.

Before you can create a Shadow Join View Adapter, in addition to creating a primary adapter, you must create either a LDAP Adapter connected to Oracle Internet Directory, or a Local Store Adapter to store shadow entries. If you use an LDAP Adapter and Oracle Internet Directory, the base DN of the LDAP Adapter must be in Oracle Internet Directory. If you use a Local Store Adapter, the base DN of the Local Store Adapter must be in Oracle Virtual Directory.

Creating Join View Adapters

After completing the prerequisites, perform the following steps to create Join View Adapters using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the **Create Adapter** button. The New Adapter Wizard appears.
4. Perform the following steps to define the Type of adapter:
   a. Select **Join** from the Adapter Type list.
   b. Enter a unique name for the Join Adapter in the Adapter Name field. The adapter name value is used in other configuration fields that need to reference the adapter.
   c. Select the **Default** template from the Adapter Template list.

      > **Note:** After selecting an adapter template, Oracle Directory Services Manager populates default values for some of the adapter settings. You should alter these default settings according to your environment.

   d. Click **Next**. The Settings screen appears.

5. Enter the root DN that the Join View Adapter provides information for in the Adapter Suffix/Namespace field. The DN defined and the child entries below it are the namespace of the adapter. The value entered in this field is the value that appears to clients of the virtual directory. The value should be specified as a comma separated distinguished name.

   Caution: Ensure that the root DN of the Join View Adapter is different from that of its primary adapter or any of the joined adapters, otherwise you can cause unexpected duplicate results.

6. Choose the primary adapter for the Join View Adapter by selecting it from the Primary Adapter list. The primary adapter is the primary driver of data in the Join View and is used by the Join View Adapter to construct its directory hierarchy. Entries in the Join View Adapter only exist if they exist in the primary adapter. The primary adapter can be any adapter. Refer to "Join View Adapter's Primary Adapter" on page 2-14 for more information.

   Note: After defining and debugging a Join View, you can set the primary adapter's Visibility routing setting to Invisible to hide un-joined entries from LDAP clients.

7. Enter the name of the adapter you want to perform bind verification with in the Bind Adapter field, or click **Browse** and select the adapter. While an LDAP client will bind with a DN based on the primary adapter, it may be that the password will be verified against a joined entry in another adapter. The Bind Adapter must be either the primary adapter or one of the joined adapters.
8. Click **Next**. The Summary screen appears displaying a summary of the Join View Adapter settings.
9. Review the Join View Adapter settings and click **Finish** to create the Join View Adapter. The new Join View Adapter appears in the Adapter tree.

After you create the Join View Adapter you can configure it using the procedures in Configuring Local Store Adapters.

12.4.1 Configuring Join View Adapters

This section describes how to configure Join View Adapter settings, including:

- Configuring Join View Adapter General Settings and Join Rules
- Configuring Adapter Routing
- Configuring Adapter Plug-ins and Mappings

12.4.1.1 Configuring Join View Adapter General Settings and Join Rules

After you create the Join View Adapter you can configure the general settings and Join Rules for the adapter by clicking the adapter name in the Adapter tree, clicking the **General** tab, setting values for the following fields, and clicking **Apply**:

Root

This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in this field will be the base DN value that returned entries will have. For

example, if you enter dc=mydomain,dc=com in the field, all entries will end with dc=mydomain,dc=com.

> **Caution:** Ensure that the root DN of the Join View Adapter is different from that of its primary adapter or any of the joined adapters, otherwise you can cause unexpected duplicate results.

Active

An adapter can be configured as active (enabled) or inactive (disabled). An adapter configured as inactive will not start during a server restart or an attempted adapter start. Use the inactive setting to keep old configurations available or in stand-by without having to delete them from the configuration. The default setting is active.

The following fields appear in the Settings section of the General tab:

DN Attributes

List of attributes to be treated as DNs for which namespace translation will be required, such as member, uniquemember, manager. For example, when reading a group entry from a proxied directory, Oracle Virtual Directory will automatically convert the DN for the group entry itself as well as the uniquemember or member attributes if these attributes are in the DN Attributes list.

> **Note:** Translate only those attribute you know will need to be used by the client application. Entering all possible DN attributes may not be necessary and can consume some a small amount of additional CPU time in the proxy.

To add attributes to the Map DN Attributes list:

1. Click **Add**. The Select DN Attribute dialog box appears.
2. Select the attribute you want to add.
3. Click **OK**.

Primary Adapter

The primary adapter is the primary driver of data in the Join View and is used by the Join View Adapter to construct its directory hierarchy. Entries in the Join View Adapter only exist if they exist in the primary adapter. The primary adapter can be any adapter. Refer to "Join View Adapter's Primary Adapter" on page 2-14 for more information.

Bind Adapter

A list of one or more adapter names that will be used for bind processing. By default, the primary adapter is used, however you can override this and list one or more other adapters. The Join View Adapter attempts to complete joins against the target adapter and process the bind. If the bind succeeds, processing stops and success is returned to the client. If the bind fails, the Join View Adapter continues trying each adapter in the Bind Adapter list. Only when all bind adapters have failed is a bind failure returned. This is useful when user identities exist in multiple directories and you want to give clients the opportunity to try password validation against more than one directory.

Join Rules

Perform the following steps to create join relationships for Join View Adapters:

1. Click the **Create** button. The Join Rule dialog box appears.

2. Select the adapter from the Adapter list that you want to join with the Join View adapter.
3. Select the type of join relationship for the Join View Adapter by choosing one of the join relationships from the Type list. Refer to "Join Relationships" on page 2-15 for more information on join relationships.
4. Enter a join condition in the Condition field as follows:
   - For Simple Joiners and OneToMany Joiners, enter a condition in the form `remoteattribute=primaryadapterattribute` where `remoteattribute` is an attribute in the target joined adapter and `primaryadapterattrinute` is an attribute from the primary adapter.
   - For Shadow Joiners, enter a unique key attribute name from the primary adapter, for example, `uid`, that can be used to locate records in case of a rename. For Shadow Joiners, the condition is not an equality condition as it is with other joiners.
   - For ConditionalSimpleJoiners, extend a Simple Joiner type of condition using the `;` character and an additional condition, such as `"employeenumber>0"` for which the join only occurs on.

     For example, a Simple Joiner condition could be:
     `employeenumber=employeenumber`

     Extend this condition for the ConditionalSimpleJoiner using the `;` character and an additional condition, for example:

     ```
     employeenumber=employeenumber;(&(employeenumber=101)(sn=Smith))
     ```
5. Click **OK** on the Join Rule dialog box to save the join relationship information. The join relationship information appears in the Join Rules table.
6. Click **Apply** at the top of the page on the General tab to deploy the join.

> **Note:** To join two different adapters with different keys to the primary adapter, create multiple Join Rules, each with single key. If you need multiple keys to create a single Join Rule, depending upon the specific criteria, you might be able to use the ConditionalSimpleJoiner or you may need to write a custom Join Rule.

Perform the following steps to modify join relationships for Join View Adapters:

1. Click the name of the join relationship in the Join Rules table that you want to modify. A split screen appears with the join relationship settings in the lower half of the screen.
2. Edit the join relationship as desired.
3. Click **Apply** in the lower half of the screen to save your changes.
4. Click **Apply** at the top of the page on the General tab to deploy the join.

Perform the following steps to delete join relationships for Join View Adapters:

1. Click the name of the join relationship in the Join Rules table that you want to delete.
2. Click the **Delete** button on the Join Rules table. A confirmation dialog box appears asking you to confirm that you want to delete the join relationship.

3. Click **Delete** on the confirmation dialog box to delete the join relationship. The join relationship is removed from the Join Rules table.

12.4.1.2 Configuring Adapter Routing

After you create the adapter you can configure routing for the adapter by clicking the adapter name in the Adapter tree, clicking the **Routing** tab, and referring to "Understanding Routing Settings" on page 3-3. Additionally, review the following information specific to configuring Join View Adapter routing:

- Primary Adapter Routing
- Local Store Adapter Routing as Join View Adapter's Local Store Directory

Primary Adapter Routing

Because the Join View Adapter's primary adapter is the primary driver of data in the Join View and is used by the Join View Adapter to construct its directory hierarchy you also must configure the primary adapter's routing.

Modify the primary adapter's **Retrievable Attributes** and **Storable Attributes** routing settings to control which attributes may be written to the primary adapter. If you do not want Oracle Virtual Directory to be able to write any modifications to the primary adapter, set **Storable Attributes** to `_never`.

Local Store Adapter Routing as Join View Adapter's Local Store Directory

If you are using a Local Store Adapter as the local store directory for the Join View Adapter you may want to adjust the Local Store Adapter's routing settings also.

Modify the **Storable Attributes** routing setting for the Local Store Adapter so that only the attributes that are to be written locally are listed. Include the unique key attribute used in the join rule and include the vdeprimaryref attribute. Optionally, set the **Visibility** routing setting to **Internal** for the if you do not want it to be seen by LDAP clients.

12.4.1.3 Configuring Adapter Plug-ins and Mappings

After you create the adapter you can apply Plug-ins and Mappings to the adapter by clicking the adapter name in the Adapter tree, clicking the **Plug-Ins** tab, and referring to "Managing Adapter Plug-ins" on page 13-1 and "Applying Mappings to Adapters" on page 14-3.

12.4.2 Configuring a Shadow Join View Adapter for Oracle Internet Directory

The following steps are an overview of the process for configuring a Join View Shadow Adapter for use with Oracle Internet Directory:

On Oracle Internet Directory:

1. Extend the Oracle Internet Directory schema to add support for shadow objects/attributes using the following steps:

   a. Create an LDIF file with the following information:

```
dn: cn=subschemasubentry
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.17119.1.0.1 NAME 'vdeprimaryref' EQUALITY
caseIgnoreMatch SYNTAX '1.3.6.1.4.1.1466.115.121.1.15' USAGE
userApplications )
```

```
dn: cn=subschemasubentry
changetype: modify
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.17119.1.1.1 NAME 'vdeShadowObject' SUP 'top'
STRUCTURAL MUST vdeprimaryref )
```

b. Use the Oracle Internet Directory `ldapmodify` tool to import the LDIF file, for example:

```
ldapmodify -h ORACLE_INTERNET_DIRECTORY_HOST
-p ORACLE_INTERNET_DIRECTORY_PORT -D bindDN -q -v -f PATH_TO_LDIF_FILE
```

2. Create a `cn=shadowentries` orclcontainer object to store the shadow entries in a branch that is separate from normal users to avoid confusing the shadow entries with any other normal user entries.

On Oracle Virtual Directory:

1. Create an LDAP Adapter that connects to the Oracle Internet Directory branch you created in Step 2 and set the visibility to **internal** because only the Shadow Join must access it.

2. Add `vdeprimaryref,`*`uid`* followed by comma separated list of attributes you want to store in the shadow entry to the Storeable Attributes field. Replace *`uid`* with the name of the attribute you can use to identify the entry if the DN changes in the primary adapter. An example may look like:

```
vdeprimaryref,uid,cn,obpasswordhistory
```

3. Set the primary adapter's visibility to **internal** as the Shadow Join will be the visible "entry" to LDAP clients.

4. Create a new Join View Adapter and set the bind adapter to be the primary adapter.

5. Create a new Shadow Join rule as follows:

a. Set the joined adapter to be the shadow LDAP adapter you created in Step 1.

b. Set *`uid`* as the condition value, replacing uid with proper value if you are using another attribute as the primary key attribute for the entry.

After completing these steps, when you update the entry exposed via the Join View:

- Oracle Virtual Directory determines which attributes must be written to the primary adapter and to the *Shadow LDAP*.
- When Oracle Virtual Directory writes to the *Shadow LDAP* it first checks to make sure the shadowed entry exists in the LDAP server (by checking for the `vderef` attribute and then the condition attribute value). If Oracle Virtual Directory does not find an entry, it creates the entry then updates the attributes.
- An LDAP client will see a complete entry with all of the attributes when it connects to Oracle Virtual Directory after the update is complete.

13

Managing Oracle Virtual Directory Plug-ins

This chapter explains how to manage Oracle Virtual Directory plug-ins and includes the following topics:

- Managing Adapter Plug-ins
- Managing Global Server Plug-ins

13.1 Managing Adapter Plug-ins

This topic explains various tasks for managing Adapter plug-ins and contains the following sections:

- Creating Adapter Plug-Ins
- Configuring Adapter Plug-Ins to Execute Only on Specific Operations
- Editing Adapter Plug-Ins
- Deleting Adapter Plug-Ins

13.1.1 Creating Adapter Plug-Ins

After you create an adapter you can create Plug-ins for the adapter.

> **Note:** The Performance Monitor plug-in is added to all adapters by default after the adapter is created. This enables Oracle Enterprise Manager Fusion Middleware Control to display performance metrics for each adapter.

Perform the following steps to create a plug-in for an adapter using Oracle Directory Services Manager:

> **See Also:** "Understanding Oracle Virtual Directory Plug-Ins" on page 4-1

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the name of the adapter in the Adapter tree that you want to create a plug-in for. The adapter's configuration appears.
4. Click the **Plug-Ins** tab

5. Click the **Create Plug-In** button. The Plug-In dialog box appears.
6. Enter a name for the Plug-in in the Name field.
7. Enter the Plug-in class name in the Class field, or click **Browse**, then select the plug-in from the Plug-In Selection box, and then click **OK**.
8. Add parameters and values to the Plug-in by clicking the **Create Parameter** button in the Parameters table, selecting a parameter from the Name list, and entering a value for the parameter in the Value field.
9. Add namespaces to the Plug-in by clicking the **Create Namespace** button in the Namespace table and entering a value for the namespace in the Namespace field.
10. Click **OK** to save the plug-in settings. The plug-in is added to the table of deployed plug-ins for the adapter on the adapter screen.
11. Optionally, you can configure the plug-in to execute on specific operations, such as Bind, Add, Get, and so on. Refer to "Configuring Adapter Plug-Ins to Execute Only on Specific Operations" for more information.
12. Click the **Apply** button on the adapter screen to associate the plug-in with the adapter.

13.1.2 Configuring Adapter Plug-Ins to Execute Only on Specific Operations

You can configure adapter plug-ins to execute only on specific operations, such as Bind, Add, Get, and so on. You can also configure the order (sequence) in which adapter plug-ins execute on specific operations.

Perform the following steps to configure adapter plug-ins to execute only on specific operations and in a specific order:

Note: The following steps assume the plug-in has already been created and deployed to the adapter. If this is not the case, refer to "Creating Adapter Plug-Ins" to create the plug-in before performing the following steps.

---

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the name of the adapter that contains the plug-in you want to configure. The adapter's configuration appears in the main screen.
4. Click the **Plug-Ins** tab.
5. Select the operation you want a plug-in to specifically operate on from the Select Operation list. For example, select **Get** to configure a plug-in to operate only on the get operation.
6. Click the **Edit Plug-Ins** (pencil) button in the Plug-ins for *Specific* Operation table. For example, if you selected the **Get** operation in the Select Operation list, the table is labeled Plug-ins for Get Operation.

Note: The label for this table changes according to the operation selected in the Select Operation list.

After you click the **Edit Plug-Ins** (pencil) button, the Select and Order the Plug-ins dialog box appears. The All Plug-ins field on the left of the dialog box lists all plug-ins deployed on the adapter. The Selected Plug-Ins field on the right lists plug-ins that will execute only on the specific operation selected in the Select Operation list.

7. Select the plug-ins you want to execute on the operation and use the **Move (>)** and **Remove (<)** buttons to move the plug-ins to and from the Selected Plug-Ins field on the right.

   If more than one plug-ins are listed in the Selected Plug-Ins field, you can configure the order in which they will execute.

8. To change the order in which the adapter plug-ins execute on the operation, select one plug-in at a time in the Selected Plug-ins field and use the **up** and **down** arrow buttons on the right side of the Selected Plug-ins field to change the order.

   Note: The plug-in at the top of the Selected Plug-ins field will execute on the operation first, and the remaining plug-ins in the field execute in a descending order.

9. Click **Apply** on the Select and Order the Plug-ins dialog box to save the changes. The plug-ins appear in the Plug-ins for *Specific* Operation table in the order they will execute.

10. Optionally, you can set namespaces for operation-specific plug-ins to control the location in the virtual tree where they execute, by performing the following steps:

    a. Click the expand/collapse icon to left of the plug-in name in the Plug-ins for *Specific* Operation table.

    b. Enter the namespace in the Namespace field.

    c. Add or delete namespaces using the **Add (+)** and **Delete (X)** buttons near the namespace field.

11. Click the **Apply** button at the top of the adapter screen to save the changes.

13.1.3 Editing Adapter Plug-Ins

Perform the following steps to edit an existing plug-in for an adapter using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the name of the adapter that contains the plug-in you want to edit in the Adapter tree. The adapter's configuration appears.
4. Click the **Plug-Ins** tab
5. Select the plug-in you want to edit from the table of existing plug-ins and click the **Edit Plugin/Mapping** button. The Plug-In dialog box appears displaying the plug-in's settings.
6. Edit the plug-in as desired and click **OK** on the Plug-In dialog box.
7. Click the **Apply** button at the top of the adapter screen to save the changes.

13.1.4 Deleting Adapter Plug-Ins

Perform the following steps to delete an existing plug-in for an adapter using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the name of the adapter that contains the plug-in you want to delete in the Adapter tree. The adapter's configuration appears.
4. Click the **Plug-Ins** tab
5. Select the plug-in you want to delete from the table of existing plug-ins and click the **Delete Plugin/Mapping** button. A confirmation dialog box appears asking you to confirm that you want to delete the plug-in from the adapter configuration.
6. Click the **Confirm** button on the confirmation dialog box to delete the plug-in. The plug-in is removed from the table of existing plug-ins.
7. Click the **Apply** button at the top of the adapter screen to save the changes.

13.2 Managing Global Server Plug-ins

This topic explains various tasks for managing server-level (Global) plug-ins and contains the following sections:

- Creating Global Server Plug-Ins
- Viewing Deployed Global Server Plug-ins
- Editing Global Server Plug-Ins
- Deleting Global Server Plug-Ins

13.2.1 Creating Global Server Plug-Ins

Perform the following steps to create a server-level plug-in using Oracle Directory Services Manager:

See Also: "Understanding Oracle Virtual Directory Plug-Ins" on page 4-1

---

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Global Plugins** entry in the navigation tree.
4. Click the **Create Plug-In** button. The Plug-In dialog box appears.
5. Enter a name for the Plug-in in the Name field.
6. Enter the Plug-in class name in the Class field, or click **Select**, then select the plug-in from the Plug-In Selection box, and then click **OK**.
7. Add parameters and their values to the Plug-in by clicking the **Create Parameter** button in the Parameters table, selecting a parameter from the Name list, and entering a value for the parameter in the Value field.

8. Determine where you want the plug-in to execute. The plug-in can execute at specific location in the virtual directory or at a global server level spanning the entire virtual tree.

   To execute the plug-in at a global server level, leave the Namespaces table empty and click **OK** to activate the plug-in for the entire virtual tree.

   To execute the plug-in at a specific location in the virtual tree, perform the following steps:

   a. Click the **Create Namespace** button in the Namespaces table.

   b. Enter the location of the virtual tree where you want the plug-in to execute in the Namespace field.

      Create multiple Namespaces to have the plug-in execute at multiple specific locations in the virtual tree.

   c. Click **OK** to activate the plug-in at the specific locations in the virtual tree.

9. Refer to "Viewing Deployed Global Server Plug-ins" to verify the plug-in was activated.

13.2.2 Viewing Deployed Global Server Plug-ins

You can view a list of the plug-ins that have been deployed at the server level—not adapter level—by performing the following steps:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Global Plugins** entry in the navigation tree. A list of the activated Mappings and plug-ins appears in the navigation tree.

13.2.3 Editing Global Server Plug-Ins

Perform the following steps to edit an existing server-level plug-in using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Global Plugins** entry in the navigation tree. A list of the activated Mappings and plug-ins appears in the navigation tree.
4. Click the name of the plug-in you want to edit from the navigation tree. The plug-in's configuration appears.
5. Edit the plug-in as desired and click the **Apply** button at the top of the plug-in screen to save the changes.

13.2.4 Deleting Global Server Plug-Ins

Perform the following steps to delete an existing server-level plug-in using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.

3. Expand the **Global Plugins** entry in the navigation tree. A list of the activated Mappings and plug-ins appears in the navigation tree.
4. Click the name of the plug-in you want to delete from the navigation tree. The plug-in's configuration appears.
5. Click the Delete button in the navigation tree. A confirmation dialog box appears asking you to confirm that you want to delete the plug-in.
6. Click the **Delete** button on the confirmation dialog box to delete the plug-in. The plug-in is removed from the list of existing plug-ins in the navigation tree.

14

Managing Oracle Virtual Directory Mappings

This chapter explains how to manage Oracle Virtual Directory mappings and includes the following topics:

- Constructing Mappings Using Mapping Templates
- Creating and Activating Server Mappings
- Applying Mappings to Adapters

14.1 Constructing Mappings Using Mapping Templates

Oracle Virtual Directory includes mapping templates that act as a macro and enable you to quickly construct Mappings. The following steps explain how to construct, compile, and deploy Mappings to the Oracle Virtual Directory server so they are available to be activated at both the adapter and server levels.

Perform the following steps to construct, compile, and deploy Mappings to the Oracle Virtual Directory server using Oracle Directory Services Manager's Mapping Templates feature:

> **Note:** If you are managing multiple Oracle Virtual Directory servers from multiple Oracle Directory Services Manager sessions that are running from the same Oracle Directory Services Manager component and you construct and then save different mapping templates to each Oracle Virtual Directory server continuously (meaning back-to-back, within 10 seconds of each other), the most recent mapping template will be saved to all servers.
>
> This issue occurs because the same Oracle Directory Services Manager work directory is shared between all of the Oracle Virtual Directory servers managed by a single Oracle Directory Services Manager component.
>
> To avoid this issue, be sure you wait more than 10 seconds before attempting to save the mapping templates to the different Oracle Virtual Directory servers.

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Mapping Templates** entry in the Advanced tree. The list of Mapping templates appear.

4. Determine which Mapping template you want to use to construct your Mapping by referring to "Understanding Mapping Templates" on page 5-3.

   After you determine which Mapping template you want to use, click on the name of the template. The Mapping Parameters for the Mapping template you selected appear in the main screen.

5. Enter values for the attributes in the Mapping template by performing the following steps:

   a. Click on the attribute in the Mapping Parameters table that you want to edit. The current value for the attribute appears.

   b. Enter a new value for the attribute in the Value field and click **OK**. The new value for the attribute appears in the Mapping Parameters table.

   Repeat this step until you have set values for all the desired attributes in the Mapping.

6. Click **Apply** at the top of the Mapping Templates screen.

   After you click **Apply**, Oracle Directory Services Manager complies the Mapping template into a Mapping script and sends it to the Oracle Virtual Directory server so that it is available to be activated at the adapter or global server level.

7. Refer to "Viewing Deployed Mappings" to verify the Mapping was deployed to the Oracle Virtual Directory server.

14.1.1 Viewing Deployed Mappings

You can view a list of the Mappings that have been deployed to the Oracle Virtual Directory server and that can be activated for use by performing the following steps:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Deployed Mappings** entry in the Advanced tree. A list of the Mappings that have been deployed to the Oracle Virtual Directory server and that can be activated for use appears in the Advanced tree.

14.2 Creating and Activating Server Mappings

This section describes how to create and activate Mappings at a global server level. Refer to "Applying Mappings to Adapters" for information on activating Mappings at an adapter level.

Perform the following steps to create and activate a Mapping at a global server level using Oracle Directory Services Manager:

> **Note:** Before you can create and activate a Mapping, the Mapping file must reside on the Oracle Virtual Directory server. Refer to "Constructing Mappings Using Mapping Templates" for information on constructing and deploying Mappings.

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.

3. Expand the **Global Plugins** entry in the Advanced tree.
4. Click the **Create Mapping** button at the top of the Global Plugins entry in the Advanced tree. The Mapping dialog box appears.
5. Enter a name in the **Name** field to describe the Mapping. This name is used to identify and describe the Mapping, not to name the actual Mapping script file.
6. Enter the path to the Mapping script file in the Mapping File field, or click **Select**, navigate to the Mapping script file, select it, and then click **OK**.
7. Determine where you want the Mapping to execute. The Mapping can execute at specific location in the virtual directory or at a global server level spanning the entire virtual tree.

   To execute the Mapping at a global server level, leave the Namespaces table empty and click **OK** to activate the Mapping for the entire virtual tree.

   To execute the Mapping at a specific location in the virtual tree, perform the following steps:

   a. Click the **Create Namespace** button in the Namespaces table.
   b. Enter the location of the virtual tree where you want the Mapping to execute in the Namespace field.

      Create multiple Namespaces to have the Mapping execute at multiple specific locations in the virtual tree.

   c. Click **OK** to activate the Mapping at the specific locations in the virtual tree.
8. Refer to "Viewing Activated Server Mappings" to verify the Mapping was activated.

14.2.1 Viewing Activated Server Mappings

You can view a list of the Mappings that have been activated at the server level—not adapter level—by performing the following steps:

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Global Plugins** entry in the Advanced tree. A list of the activated Mappings and plug-ins appears in the Advanced tree.

14.3 Applying Mappings to Adapters

Perform the following steps to apply a mapping to an adapter using Oracle Directory Services Manager:

> **Note:** Before you can apply a Mapping to an adapter, the Mapping file must reside on the Oracle Virtual Directory server. Refer to "Constructing Mappings Using Mapping Templates" for information on constructing and deploying Mappings to the Oracle Virtual Directory server.

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.

3. Click the name of the adapter in the tree that you want to apply the mapping to. The adapter's settings screen appears.
4. Click the **Plug-ins** tab. The adapter's plug-ins screen appears.
5. Click the **Create Mapping** button. The Mapping dialog box appears.
6. Enter a name in the **Name** field to describe the Mapping. This name is used to identify and describe the Mapping, not to name the actual Mapping script file.
7. Enter the path to the Mapping script file in the Mapping File field, or click **Select**, navigate to the Mapping script file, select it, and then click **OK**.
8. Determine where you want the Mapping to execute. The mapping can execute at a specific location under the adapter namespace or at the adapter namespace itself, thus spanning the entire adapter.

   To execute the Mapping at the adapter level, leave the Namespaces table empty and click **OK** to activate the Mapping for the entire adapter.

   To execute the Mapping at a specific location under the adapter, perform the following steps:

   a. Click the **Create Namespace** button in the Namespaces table.
   b. Enter the location of the virtual tree where you want the Mapping to execute in the Namespace field.

      Create multiple Namespaces to have the Mapping execute at multiple specific locations in the virtual tree.

   c. Click **OK** to activate the Mapping at the specific locations in the virtual tree.
9. Click **Apply** on the adapter's plug-ins screen to apply the mapping to the adapter.

15

Managing Oracle Virtual Directory Entries and Schema

This chapter explains how to manage Oracle Virtual Directory entries and schema using Oracle Directory Services Manager. It contains the following topics:

- Managing Oracle Virtual Directory Entries Using Data Browsers
- Managing Oracle Virtual Directory Schema Using Oracle Directory Services Manager

15.1 Managing Oracle Virtual Directory Entries Using Data Browsers

This topic describes Oracle Virtual Directory data browsers and how to use them to manage Oracle Virtual Directory entries. This topic contains the following sections:

- Understanding Oracle Virtual Directory Data Browsers
- Managing Oracle Virtual Directory Entries Using the Client View Data Browser
- Managing Oracle Virtual Directory Source Entries Using the Adapter Browser

15.1.1 Understanding Oracle Virtual Directory Data Browsers

Oracle Virtual Directory provides the following types of data browsers:

- Client View browser
- Adapter browser

Both the Client View and Adapter browsers are automatically created when you define a new Oracle Virtual Directory server. Oracle Virtual Directory uses DSMLv2 over its administrative gateway to retrieve the data presented by the browsers.

Client View Browser

The Client View browser allows you to search and view the entire virtual directory tree (defined by all configured adapters) after Oracle Virtual Directory has performed all data mapping and transformation. Think of the Client View as the *after* view—what the data looks like after it is virtualized by Oracle Virtual Directory.

You can also import and export LDIF files to and from the Oracle Virtual Directory using the Client View data browser. LDIF is an industry standard textual interchange format designed for exchanging data between LDAP servers. LDIF files are typically used to import and export batch data and schema configuration changes.

Adapter Browser

The Adapter Browser allows you to view data as it exists in both LDAP and Database Adapter connected repositories. Think of the Adapter Browser view as the *before* view—what the data in LDAP and database repositories looks like before it is virtualized by Oracle Virtual Directory. When using the Adapter Browser to view databases, tables and fields appear as they exist in the original database, including sample table rows to assist in data modeling.

Notes:

- When you click the name of an existing adapter in the Adapter Browser, the configuration of the adapter appears in the main Oracle Directory Services Manager screen. This adapter configuration information is read only—you cannot edit an adapter's configuration using the Adapter Browser.
- Data from Join View and Local Store Adapters is not visible from the Adapter Browser.

15.1.2 Managing Oracle Virtual Directory Entries Using the Client View Data Browser

The Client View browser allows you to view and search the entire virtual directory tree (defined by all configured adapters) after Oracle Virtual Directory has performed all data mapping and transformation. You can use the Client View browser to import and export LDIF files to and from the virtual directory. You can also modify and delete attributes of the virtual tree entries using the Client View Browser.

This topic explains how to perform the following Client View browser tasks:

- Searching the Virtual Directory Tree
- Viewing Oracle Virtual Directory Entries
- Modifying Attributes of Virtual Directory Tree Entries
- Importing an LDIF File
- Exporting an LDIF File

15.1.2.1 Searching the Virtual Directory Tree

You can search the virtual directory tree using the Client View data browser. There are two types of searches: simple and advanced. A simple search will search only the cn, uid, sn, givenname, mail, and initials attributes. An advanced search allows you to specify the search scope depth and other detailed search parameters.

To perform a simple search, perform the following steps:

1. Log in to Oracle Directory Services Manager.
2. Select **Data Browsers** from the task selection bar. The Data Tree appears.
3. Select the **Client View** entry in the Data Tree.
4. Enter the keyword you want to search for in the search field at the top of the Data Tree and click the Simple Search > icon.

To perform an advanced search, perform the following steps:

1. Log in to Oracle Directory Services Manager.
2. Select **Data Browsers** from the task selection bar. The Data Tree appears.

3. Select the **Client View** entry in the Data Tree.
4. Click the **Advanced** button at the top of the Data Tree. The Search Dialog box appears.
5. Enter the starting point for the search in the Root Of The Search field.
6. Enter the maximum number of entries for the search to return in the Max Results (entries) field.
7. Select the depth scope for the search by selecting one of the following options from the Search Depth list:

   Base: searches only the entries at the location specified by the Root Of The Search field.

   One Level: searches all entries one level under the location specified by the Root Of The Search field.

   Subtree: searches the location specified by the Root Of The Search field and includes all entries under that location.
8. Enter in the maximum number of seconds for the search to execute in the Max Search Time (seconds) field.
9. Enter the Search Criteria as follows:
   a. Select the attribute to search for by selecting the attribute name from the list of attributes.
   b. Select a matching rule from the list of matching rules.
   c. Enter a value for the matching rule in the Specify Matching Value field.

      You can delete a search criterion by clicking the **Delete** button next to it.

      Note: To search for customized (extended) criteria, select the **Show LDAP filter** option and enter a custom search filter, such as (`objectclass=*`), in the LDAP Query field.
10. Click **Search** to execute the search.

15.1.2.2 Viewing Oracle Virtual Directory Entries

Perform the following steps to view entries in the Oracle Virtual Directory using the Client View data browser:

1. Log in to Oracle Directory Services Manager.
2. Select **Data Browsers** from the task selection bar. The Data Tree appears.
3. Expand the **Client View** entry in the Data Tree. The namespaces of the entries in the virtual directory appear.
4. Navigate to the content you want to view by expanding the appropriate namespace.
5. Click the entry you want to view. The properties screen appears displaying the attributes and objectclasses for that entry. You can adjust which attributes are shown and which attributes are hidden in the properties screen by clicking the **Show All** or **Hide Empty Values** option at the top-right of the screen.

15.1.2.3 Modifying Attributes of Virtual Directory Tree Entries

You can modify and delete attributes of the virtual directory tree entries using the Client View Browser. You *cannot* add entries using the Client View Browser.

Perform the following steps to modify attributes of virtual directory tree entries using the Client View Browser:

1. Log in to Oracle Directory Services Manager.
2. Select **Data Browsers** from the task selection bar. The Data Tree appears.
3. Expand the **Client View** entry in the Data Tree. The namespaces of the entries in the virtual directory appear.
4. Navigate to the entry you want to modify by expanding the appropriate namespace and then click the entry. The details for that entry appear in the main screen and are organized by context-sensitive tabs, such as Attributes, Person, and Groups, depending upon the type of entry.

The following are common procedures for modifying entries. Regardless of the specific procedure you perform, after modifying an entry, click **Apply** to save your changes or **Revert** to discard them.

Notes:

- To modify the attributes for all types of entries, click the **Attributes** tab and make the desired changes. By default, only non-empty attributes are shown. You can switch between **Managed Attributes** and **Show All** by using the **Views** list.
- To change the list of attributes shown as managed attributes, click the icon under **Optional Attributes**. Select attributes you want to move from the All Attributes list to the Shown Attributes lists and use the **Move** and **Move All** arrows to move the attributes. Select attributes you want to move from the shown Attributes list to the All Attributes lists and use the **Remove** and **Remove All** arrows to move the attributes. Click **Add Attributes** to make your changes take effect or click **Cancel** to discard your changes. After you click **Add Attributes**, only the attributes that were on the Shown Attributes list are shown in the Managed Attributes view.

To add an object class:

1. Click the **Attributes** tab.
2. Click the **Add** icon next to `objectclass` and use the Add Object Class dialog to select object class entries. Optionally, use the search box to filter the list of object classes. To add the object class, click it and then click **OK**.

To delete an object class:

1. Click the **Attributes** tab.
2. Select the object class you want to delete.
3. Click the **Delete** icon next to `objectclass`. The Delete Object Class dialog lists the attributes that will be deleted with that class.
4. Click **Delete** to proceed or **Cancel** to cancel the deletion.

To modify person entries:

1. Click the **Person** tab.
2. Modify the information as needed. To upload a photograph for the person entry, click **Browse**, navigate to the photograph, then click **Open**. To update the photograph, click **Update** and follow the same procedure. Click the **Delete** icon to delete the photograph.

To modify group entries:

1. Click the **Group** tab.
2. Click **Add** or **Delete** in the appropriate text box to add or delete a group owner or member.

15.1.2.4 Importing an LDIF File

Perform the following steps to import LDIF files into the Oracle Virtual Directory using the Client View data browser:

1. Verify the LDIF file you want to import has a valid version number in the first line in the file. Oracle Virtual Directory requires all LDIF files that will be imported to contain this version number at the beginning of the file. If the file does not have a version number in the first line, add `version: 1` to the beginning of the file.
2. Log in to Oracle Directory Services Manager.
3. Select **Data Browsers** from the task selection bar. The Data Tree appears.
4. Expand the **Client View** entry in the Data Tree.
5. Select the location where you want to import the LDIF file to by clicking the appropriate namespace in the Client View entry in the tree.
6. Click the **Import LDIF** button at the top of the tree. The Import File dialog box appears.
7. Enter the path of the LDIF file you want to import in the Select an LDIF File field, or click the **Browse** button and navigate to the file.
8. Click the **OK** button on the Import File dialog box to import the LDIF file.

15.1.2.5 Exporting an LDIF File

Perform the following steps to export LDIF files from the Oracle Virtual Directory using the Client View data browser:

1. Log in to Oracle Directory Services Manager.
2. Select **Data Browsers** from the task selection bar. The Data Tree appears.
3. Expand the **Client View** entry in the Data Tree.
4. Select the location where you want to export the LDIF file from by clicking the appropriate namespace in the Client View entry in the tree.
5. Click the Export LDIF button at the top of the Data Tree. The Download LDIF File dialog box appears.

> **Note:** The maximum number of entries in an LDIF File that can be exported is 1000. If there are more than 1000 entries in the namespace that you attempted to export, only the first 1000 entries are exported.

6. Open the LDIF file in your browser by clicking the **Click here to open the LDIF file** link in the Download LDIF File dialog box.

   Note: Clicking the OK button in the Download LDIF File dialog box *does not* export the LDIF file.

   After clicking the **Click here to open the LDIF file** link in the Download LDIF File dialog box, the LDIF File appears in a new, separate browser window.

7. Use your browser's Save command to save the LDIF file.

15.1.3 Managing Oracle Virtual Directory Source Entries Using the Adapter Browser

The Adapter Browser allows you to view data as it exists in both LDAP and Database Adapter connected repositories. The Adapter Browser allows you to see what data looks like before it is virtualized by Oracle Virtual Directory. You can also modify and delete attributes of the source entries using the Adapter Browser.

This topic explains how to perform the following Adapter Browser tasks:

- Viewing Source Repository Entries
- Modifying Attributes of Source Repository Entries in Oracle Virtual Directory

Notes:

- When you click the name of an existing adapter in the Adapter Browser, the configuration of the adapter appears in the main Oracle Directory Services Manager screen. This adapter configuration information is read only—you cannot edit an adapter's configuration using the Adapter Browser.
- Data from Join View and Local Store Adapters is not visible from the Adapter Browser.

15.1.3.1 Viewing Source Repository Entries

Perform the following steps to view data as it exists in the remote, underlying repositories for each adapter defined using the Adapter Browser:

1. Log in to Oracle Directory Services Manager.
2. Select **Data Browsers** from the task selection bar. The Data Tree appears.
3. Expand the **Adapter Browser** entry in the Data Tree. The names of the adapters that are connected to data repositories appear.
4. Expand the entry for the adapter that contains the source entries you want to view. The entries for the adapter appear.
5. Click the entry you want to view. The source data for that entry appears in the properties screen. By default, the properties screen displays only the attributes for the entry that have values. Select the **Show All** option to view all attributes for the entry.

15.1.3.2 Modifying Attributes of Source Repository Entries in Oracle Virtual Directory

You can modify and delete attributes of the source repository entries in Oracle Virtual Directory using the Adapter Browser. You *cannot* add source entries using the Adapter Browser.

Perform the following steps to modify attributes of the source repository entries in Oracle Virtual Directory using the Adapter Browser:

1. Log in to Oracle Directory Services Manager.
2. Select **Data Browsers** from the task selection bar. The Data Tree appears.
3. Expand the **Adapter Browser** entry in the Data Tree. The names of the adapters that are connected to data repositories appear.
4. Expand the entry for the adapter that contains the source entries you want to modify. The entries for the adapter appear.
5. Click the entry you want to modify. The details for that entry appear in the main screen and are organized by context-sensitive tabs, such as Attributes, Person, and Groups, depending upon the type of entry.

The following are common procedures for modifying entries. Regardless of the specific procedure you perform, after modifying an entry, click **Apply** to save your changes or **Revert** to discard them.

Notes:

- To modify the attributes for all types of entries, click the **Attributes** tab and make the desired changes. By default, only non-empty attributes are shown. You can switch between **Managed Attributes** and **Show All** by using the **Views** list.
- To change the list of attributes shown as managed attributes, click the icon under **Optional Attributes**. Select attributes you want to move from the All Attributes list to the Shown Attributes lists and use the **Move** and **Move All** arrows to move the attributes. Select attributes you want to move from the shown Attributes list to the All Attributes lists and use the **Remove** and **Remove All** arrows to move the attributes. Click **Add Attributes** to make your changes take effect or click **Cancel** to discard your changes. After you click **Add Attributes**, only the attributes that were on the Shown Attributes list are shown in the Managed Attributes view.

To add an object class:

1. Click the **Attributes** tab.
2. Click the **Add** icon next to `objectclass` and use the Add Object Class dialog to select object class entries. Optionally, use the search box to filter the list of object classes. To add the object class, click it and then click **OK**.

To delete an object class:

1. Click the **Attributes** tab.
2. Select the object class you want to delete.
3. Click the **Delete** icon next to `objectclass`. The Delete Object Class dialog lists the attributes that will be deleted with that class.

4. Click **Delete** to proceed or **Cancel** to cancel the deletion.

To modify person entries:

1. Click the **Person** tab.
2. Modify the information as needed. To upload a photograph for the person entry, click **Browse**, navigate to the photograph, then click **Open**. To update the photograph, click **Update** and follow the same procedure. Click the **Delete** icon to delete the photograph.

To modify group entries:

1. Click the **Group** tab.
2. Click **Add** or **Delete** in the appropriate text box to add or delete a group owner or member.

15.2 Managing Oracle Virtual Directory Schema Using Oracle Directory Services Manager

This topic explains how to manage Oracle Virtual Directory schema and contains the following sections:

- Managing Oracle Virtual Directory Schema Attributes
- Managing Oracle Virtual Directory Schema Object Classes

Note: This topic explains how to manage Oracle Virtual Directory schema using Oracle Directory Services Manager. If you use `ldapmodify` to modify Oracle Virtual Directory schema, be aware of the following items:

- Oracle Virtual Directory expects schema keywords (such as `name`) to be in all capital letters (`NAME`).
- Oracle Virtual Directory does not support the `ldapmodify replace` operation when modifying schema.

15.2.1 Managing Oracle Virtual Directory Schema Attributes

This section explains how to manage Oracle Virtual Directory schema attributes and contains the following tasks:

- Searching for Schema Attributes
- Creating New Schema Attributes
- Creating "Like" Schema Attributes
- Modifying Schema Attributes
- Deleting Schema Attributes

15.2.1.1 Searching for Schema Attributes

Oracle Directory Services Manager provides search functionality to simplify the process of navigating schema attributes. Perform the following steps to search for schema attributes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.

2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the **Attribute Types** entry. The Attribute Type controls, including search field, and a list of the existing schema attributes appear.
4. Enter a string to search for in the search field. Two pattern matching characters are supported, `*` and `?`. Use the `*` character as a wildcard to match zero or more characters. Use the ? character to match one single character. For example, the search string `auth????????` returns the attribute authPassword.
5. Click the **Go (>)** icon to start the search. The attributes that match the search criteria appear in the navigation tree.

15.2.1.2 Creating New Schema Attributes

Perform the following steps to create new Oracle Virtual Directory schema attributes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Attribute Types entry. A list of the existing schema attributes appears.
4. Click the **Create** button. The New Attribute Type dialog box appears.
5. Enter the following information in the New Attribute Type dialog box fields:
   - Enter the name of the attribute in the Name field.
   - Enter a unique object identifier specified by ICANNS in the Object ID field. If not registered, any unique value will suffice. Oracle recommends registering all custom attributes by using a unique object identifier.
   - Optionally, enter a description for the attribute in the Description field.
   - Select the format for the attribute value by selecting one of the options in the Syntax list. Oracle Virtual Directory uses parent syntax values only.
   - Enter the bytes length of the attribute in the Size (bytes) field. 0 or no value (empty) implies unlimited. Oracle Virtual Directory does not enforce this attribute definition.
   - Select a standard from the Usage list for how the attribute can be used.
   - Enter an Object ID matching rule in the Ordering field for ordered searching. Oracle Virtual Directory does not use this attribute definition.
   - Enter a matching rule Object ID in the Equality field for equality. Oracle Virtual Directory does not use this attribute definition.
   - Enter a matching rule Object ID in the Substring field for substring searching. Oracle Virtual Directory does not use this attribute definition.
   - Enable the Single Value option if the attribute may hold only a single value at a time. If this option is not enable, the attribute may hold multiple values.
   - Optionally, select a parent attribute for the new attribute by selecting one of the existing attributes from the Superior list.
6. Click **OK** on the New Attribute Type dialog box to create the attribute. The new attribute appears in the Attribute Types tree.

15.2.1.3 Creating "Like" Schema Attributes

Oracle Directory Services Manager provides the ability to create new Oracle Virtual Directory schema attributes that are similar— or "like"—an existing attribute. This ability is known as "Create Like." When you create a new attribute like an existing attribute, you select an existing attribute to base the new one on and then you modify the base attribute's definitions to make it unique.

Perform the following steps to create a new attribute like an existing attribute using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Attribute Types entry. A list of the existing schema attributes appears.
4. Click the existing attribute that you want to base the new attribute on.
5. Click the **Create Like** button at the top of the tree. The base attribute's definitions appear.
6. Modify the base attribute's definitions as desired to create the new attribute. You must modify the base attribute's Name and Object ID definitions to create a valid new attribute.

> **Note:** Refer to step 5 in "Creating New Schema Attributes" on page 15-9 for a description of each field for the attribute definition.

7. Click **OK** on the dialog box to create the new attribute. The new attribute appears in the Attribute Types tree.

15.2.1.4 Modifying Schema Attributes

Perform the following steps to modify existing Oracle Virtual Directory schema attributes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Attribute Types entry. A list of the existing schema attributes appears.
4. Click the attribute in the list that you want to modify. The attribute's definitions appear.
5. Modify the attribute's definitions as desired. Refer to step 5 in "Creating New Schema Attributes" on page 15-9 for more information on attribute definitions.
6. Click **Apply** to save the changes.

15.2.1.5 Deleting Schema Attributes

Perform the following steps to delete existing Oracle Virtual Directory schema attributes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Attribute Types entry. A list of the existing schema attributes appears.

4. Click the attribute in the list that you want to delete. The attribute's definitions appear.
5. Click the **Delete** button at the top of the Attribute Types tree. A dialog box appears asking you to confirm that you want to delete the attribute.
6. Click the **Delete** button on the confirmation dialog box to delete the attribute. The attribute is removed from the list of existing attributes in the Attribute Types tree.

15.2.2 Managing Oracle Virtual Directory Schema Object Classes

This section explains how to manage Oracle Virtual Directory schema object classes and contains the following tasks:

- Searching for Schema Object Classes
- Creating New Schema Object Classes
- Creating "Like" Schema Object Classes
- Modifying Schema Object Classes
- Deleting Schema Object Classes

15.2.2.1 Searching for Schema Object Classes

Oracle Directory Services Manager provides search functionality to simplify the process of navigating schema object classes. Perform the following steps to search for schema object classes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the **Object Classes** entry. The Object Class controls, including search field, and a list of the existing schema object classes appear.
4. Enter a string to search for in the search field. Two pattern matching characters are supported, `*` and `?`. Use the `*` character as a wildcard to match zero or more characters. Use the ? character to match one single character. For example, the search string `inet???person` returns the object class inetOrgPerson.
5. Click the **Go (>)** icon to start the search. The object classes that match the search criteria appear in the navigation tree.

15.2.2.2 Creating New Schema Object Classes

Perform the following steps to create new Oracle Virtual Directory schema object classes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Object Classes entry. A list of the existing schema object classes appears.
4. Click the **Create** button. The New Object Class dialog box appears.
5. Enter the following information in the New Object Class dialog box fields:
   - Enter the name of the new object class in the Name field.

- Optionally, enter a description for the object class in the Description field. Oracle Virtual Directory does not enforce this object class definition.
- Enter a unique object identifier string in the Object ID field. Oracle recommends registering all custom object classes by using a unique object identifier.
- Enable the **Obsolete** option to mark the object class as obsolete for administrative purposes. Oracle Virtual Directory does not enforce this object class definition.
- Select the type of object class by selecting one of the following options from the Type list. Oracle Virtual Directory does not enforce this object class definition.
  - Select **Abstract** if the object class represents object classes that will be inherited by another class and not intended to be used directly by an object.
  - Select **Auxiliary** if the object class will be used to add additional attributes to an existing object (based on a structural object class).
  - Select **Structural** if the object class can form an entry.
- Select a parent object class for the new object class by selecting one of the existing object classes from the Superior list. If you do not select a parent object class the new object class must be descendant from top.
- Add attributes that must be present in the object class by clicking the **Add** button in the Mandatory Attributes field, selecting an attribute from the list of existing attributes in the Mandatory Attribute Selector dialog box, and clicking **OK**. You can delete Mandatory Attributes by selecting the attribute and clicking the **Delete** button.
- Add attributes that may optionally be supplied in the object class by clicking the **Add** button in the Optional Attributes field, selecting an attribute from the list of existing attributes in the Optional Attribute Selector dialog box, and clicking **OK**. You can delete Optional Attributes by selecting the attribute and clicking the **Delete** button.

6. Click **OK** on the New Object Class dialog box to create the object class. The new object class appears in the Object Classes tree.

15.2.2.3 Creating "Like" Schema Object Classes

Oracle Directory Services Manager provides the ability to create new Oracle Virtual Directory schema object classes that are similar— or "like"—an existing object class. This ability is known as "Create Like." When you create a new object class like an existing object class, you select an existing object class to base the new one on and then you modify the base object class's definitions to make it unique.

Perform the following steps to create a new object class like an existing object class using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Object Classes entry. A list of the existing schema object classes appears.
4. Click the existing object class that you want to base the new object class on.

5. Click the **Create Like** button at the top of the tree. The base object class's definitions appear.
6. Modify the base object class's definitions as desired to create the new object class. You must modify the base object class's Name and Object ID definitions to create a valid new object class.

> **Note:** Refer to step 5 in "Creating New Schema Object Classes" on page 15-11 for more information on object class definitions.

7. Click **OK** on the dialog box to create the new object class. The new object class appears in the Object Classes tree.

15.2.2.4 Modifying Schema Object Classes

Perform the following steps to modify existing Oracle Virtual Directory schema object classes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Object Classes entry. A list of the existing schema object classes appears.
4. Click the object classes in the list that you want to modify. The object classes's definitions appear.
5. Modify the object classes's definitions as desired. Refer to step 5 in "Creating New Schema Object Classes" on page 15-11 for more information on object class definitions.
6. Click **Apply** to save the changes.

15.2.2.5 Deleting Schema Object Classes

Perform the following steps to delete existing Oracle Virtual Directory schema attributes using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Schema** from the task selection bar. The Attribute Types and Object Classes navigation tree appears.
3. Expand the Object Classes entry. A list of the existing schema object classes appears.
4. Click the object class in the list that you want to delete. The object classes' definitions appear.
5. Click the **Delete** button at the top of the Object Classes tree. A dialog box appears asking you to confirm that you want to delete the object class.
6. Click the **Delete** button on the confirmation dialog box to delete the object class. The object class is removed from the list of existing object classes in the Object Classes tree.

16

Configuring Oracle Virtual Directory Access Control

This chapter explains how to configure access control for Oracle Virtual Directory and includes the following topics:

- Creating Access Control Lists Using Oracle Directory Services Manager
- Managing Access Control Lists Using Oracle Directory Services Manager

16.1 Creating Access Control Lists Using Oracle Directory Services Manager

Perform the following steps to create a new ACL using Oracle Directory Services Manager:

> **Note:** If two ACLs differ only by their grant/deny property, the resulting permission will be a deny regardless of the order in which the ACLs are added. For example, the following two ACLs will result in a deny for Search(s) and Read(r) of all attributes for public:
>
> ```
> deny:s,r#[all]#public:
> grant:s,r#[all]#public:
> ```

1. Log in to Oracle Directory Services Manager.
2. Select **Security** from the task selection bar. The Access Control Point navigation tree appears listing the existing Access Control Points.
3. Click the **Create** button. The new ACL dialog box appears.
4. Identify the Access Control Point for the new ACL by entering the DN where you want to apply the new ACL in the DN field.
5. Configure the scope of the new ACL by selecting either **entry** or **subtree** from the Scope list. Selecting **entry** applies the new ACL only at the Access Control Point DN entry in the virtual tree. Selecting **subtree** applies the new ACL at the Access Control Point DN entry and all the entries in the subtree below it.
6. Click the **Create** button in the Structural Access Items (Entry Level Operations) area to create access policy for the entries in the virtual directory tree. The Structural Access configuration dialog box appears.
7. Click the **Permissions** tab and perform the following to set the entry permissions for the access policy:

- To explicitly grant access for an entry permission, select **Grant** from the Access Type list and select the permissions you want to grant access to.
- To explicitly deny access for an entry permission, select **Deny** from the Access Type list and select the permissions you want to deny access to.

8. Click the **By Whom** tab and perform the following to set to whom the entry access policy applies:
   - Select the subject of the ACL from the By Whom list.
   - Enter the DN or IP address of the in the DN or IP Address field if you chose **Specific DN** or **IP Address** from the By Whom list.

   Click the **OK** button to save the Structural Access Items (Entry Level Operations) settings. The new entry access policy appears in the Structural Access Items (Entry Level Operations) table.
9. Click the **Create** button in the Content Access Items (Attribute Level Operations) area to create access policy for the attributes of the entry. The Content Access configuration dialog box appears.
10. Click the **Target** tab and select the attributes from the Attribute list that the access policy applies to. Selecting * applies the access policy to all attributes.
11. Click the **Permissions** tab and perform the following to set the attribute permissions for the access policy:
    - To explicitly grant access for an attribute permission, select **Grant** from the Access Type list and select the permissions you want to grant access to.
    - To explicitly deny access for an attribute permission, select **Deny** from the Access Type list and select the permissions you want to deny access to.
12. Click the **By Whom** tab and perform the following to set to whom the attribute access policy applies:
    - Select the subject of the ACL from the By Whom list.
    - Enter the DN or IP address of the in the DN or IP Address field if you chose **Specific DN** or **IP Address** from the By Whom list.
13. Click the **OK** button to save the Content Access Items (Attribute Level Operations) settings. The new attribute access policy appears in the Content Access Items (Attribute Level Operations) table.

16.2 Managing Access Control Lists Using Oracle Directory Services Manager

This topic explains how to manage ACLs using Oracle Directory Services Manager and contains the following sections:

16.2.1 Updating Access Control Lists

Perform the following steps to edit an existing ACL using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.

2. Select **Security** from the task selection bar. The Access Control Point navigation tree appears listing the existing ACLs.
3. Click the ACL you want to edit in the tree. The settings for the ACL appear.
4. Click the attribute you want to edit, edit the value as desired, and then click the **OK** button to save the changes.

16.2.2 Deleting Access Control Lists Entries

Perform the following steps to delete an existing Access Control List (ACL) using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Security** from the task selection bar. The Access Control Point navigation tree appears listing all the existing ACLs.
3. Click the ACL in the tree that contains the entry you want to delete. The settings for the ACL appear.
4. Click the entry in the ACL you want to delete.
5. Click the **Delete** button. The Delete dialog box appears asking you to confirm that you want to delete the entry. Click the **Delete** button on the Delete dialog box to delete the entry.
6. Click the **Apply** button on the ACL settings screen to apply the updated ACL.

17

Managing Oracle Virtual Directory Logging and Auditing

This chapter provides information about managing Oracle Virtual Directory logging and auditing. It contains the following topics:

- Managing Oracle Virtual Directory Logging
- Managing Oracle Virtual Directory Auditing

17.1 Managing Oracle Virtual Directory Logging

You can use Oracle Enterprise Manager Fusion Middleware Control and the Oracle WebLogic Scripting Tool (WLST) as the interface to manage Oracle Virtual Directory logging. This topic includes the following sections on managing Oracle Virtual Directory logging:

- Managing Oracle Virtual Directory Logging Using Oracle Enterprise Manager
- Managing Oracle Virtual Directory Logging Using WLST
- Managing Granular Logging

17.1.1 Managing Oracle Virtual Directory Logging Using Oracle Enterprise Manager

Oracle Enterprise Manager Fusion Middleware Control allows you to list, search, and configure log files across Oracle Fusion Middleware components. You can view log files from Oracle Enterprise Manager Fusion Middleware Control or download log files and view them using another tool.

> **See:** The *Oracle Fusion Middleware Administrator's Guide* for complete information on logging using Oracle Enterprise Manager Fusion Middleware Control.

Logging Considerations Specific to Oracle Virtual Directory

The following items must supplement the information in the *Oracle Fusion Middleware Administrator's Guide* as they are specific to Oracle Virtual Directory logging:

- When setting log levels for Oracle Virtual Directory using Oracle Enterprise Manager Fusion Middleware Control, the following log levels do not apply to and have no effect on Oracle Virtual Directory:
  - NOTIFICATION: 16 (CONFIG)
  - TRACE: 16 (FINER)

- Log messages are written to the access.log file only when logging is set to NOTIFICATION:1 (INFO) level. You can increase the log level to ERROR:1 (SEVERE) or WARNING:1 (WARNING) to disable information from being written to the access.log file.
- As a general guideline, Oracle recommends setting the log level for Oracle Virtual Directory to the *least* amount of information as possible for your environment.

17.1.2 Managing Oracle Virtual Directory Logging Using WLST

You can use WLST to perform Oracle Virtual Directory logging management tasks, including:

- List loggers and levels using `listLoggers`
- Get (view) the level for a logger using `getLogLevel`
- Set the level for a logger using `setLogLevel`
- List log handlers using `listLogHandlers`
- Configure log handlers using `configureLogHandler`
- Lists known logs using `listLogs`
- Search and display the contents of logs using `displayLogs`

> **See:** For complete information about managing Oracle Virtual Directory logging using WLST, refer to the following documents:
>
> - *Oracle Fusion Middleware Administrator's Guide*
> - *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

17.1.3 Managing Granular Logging

Message logging can be controlled using the `java.util.logging.Filter` implementation class and through the logLevel attributes specified for each adapter configuration in the in adapters.os_xml file. A default implementation of `java.util.logging.Filter StringMatchFilter` is included in Oracle Virtual Directory. This default implementation supports the following two parameters:

- `StringToBeMatched`: Allows you to specify one or more DIT/Strings.
- `AcceptOnMatch`: This boolean allows you to include or exclude the log messages based on matching the list of DIT/Strings specified.

A `java.util.logging.Filter` implementation and its parameters can be specified in the server.os_xml file. The following is an example `logFilter` configuration for `StringMatchFilter` class to exclude the log messages containing string `c=us`:

```
<logFilters>
    <filter className="com.octetstring.vde.util.StringMatchFilter">
        <param name="StringToBeMatched"  value="c=us" />
        <param name="AcceptOnMatch" value="false" />
    </filter>
</logFilters>
```

To include the log messages, set `AcceptOnMatch` to `true` and the log messages will contain the DIT specified in the `logFilter` configuration. To exclude the log messages, set `AcceptOnMatch` to `false` and the log messages will not contain the

DIT specified. To enable `StringMatchFilter`, configure it as a filter for either Logger or Handler defined in the ovd-logging.xml file. The following is an example configuration to specify the filter for `LogHandler`:

```
<logging_configuration>
   <log_handlers>
   <log_handler name='OVDHandler'
class='oracle.core.ojdl.logging.ODLHandlerFactory'
filter='com.octetstring.vde.util.StringMatchFilter'>
       <property> ... </property>
   </log_handler>
   </log_handlers>
      <loggers> ... </loggers>
<logging_configuration>
```

Note: The server.os_xml, ovd-logging.xml, and adapters.os_xml files are located in the following directory:

ORACLE_INSTANCE/config/OVD/config/*COMPONENT_NAME*/

17.2 Managing Oracle Virtual Directory Auditing

Oracle Virtual Directory utilizes the Common Audit Framework of the Oracle Application Server 11*g* infrastructure for compliance, monitoring, and analytics purposes. You can use Oracle Enterprise Manager Fusion Middleware Control and WLST as the interface to the Common Audit Framework to manage Oracle Virtual Directory auditing. This topic contains the following sections on managing Oracle Virtual Directory auditing:

- Managing Oracle Virtual Directory Auditing Using Fusion Middleware Control
- Managing Oracle Virtual Directory Auditing Using WLST

17.2.1 Managing Oracle Virtual Directory Auditing Using Fusion Middleware Control

You can use Oracle Enterprise Manager Fusion Middleware Control to perform Oracle Virtual Directory auditing tasks, including managing:

- Audit policies
- Audit data collection and storage
- Audit reports

The auditing procedure for most Oracle Fusion Middleware components, including Oracle Virtual Directory, is similar and explained in detail in the *Oracle Fusion Middleware Security Guide*. The following is an overview of the procedure for auditing Oracle Virtual Directory using Oracle Enterprise Manager Fusion Middleware Control:

1. From the Oracle Virtual Directory menu, select **Security**, then **Audit Policy Settings**.
2. From the Audit Policy list, select **Custom** to configure your own filters, or one of the filter presets, **None**, **Low**, or **Medium**.
3. If you want to audit only failures, click **Select Failures Only**.
4. To configure a filter, click the **Edit** icon next to its name. The Edit Filter dialog for the filter appears.

5. Specify the filter condition using the buttons, selections from the menus, and strings that you enter. Condition subjects include Initiator, Target, Remote IP, and Resource. Condition tests include -contains, -contains_case, -endswith, -endswith_case, -eq, -matches, -ne, -startswith, and -startswith_case. Enter values for the tests as strings. Parentheses are used for grouping and AND and OR for combining.
6. To add a condition, click the **Add** icon.
7. When you have completed the filter, click **OK**.

> **See:** The *Oracle Fusion Middleware Security Guide* for complete information about auditing Oracle Virtual Directory using Oracle Enterprise Manager Fusion Middleware Control.

17.2.2 Managing Oracle Virtual Directory Auditing Using WLST

You can use WLST to perform Oracle Virtual Directory auditing tasks, including:

- Getting (viewing) audit policy using `getAuditPolicy`
- Setting audit policy using `setAuditPolicy`
- Listing (viewing) audit events using `listAuditEvents`

> **See:** For complete information about managing Oracle Virtual Directory auditing using WLST, refer to the following documents:
>
> - *Oracle Fusion Middleware Security Guide*
> - *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

For components that manage their audit policy locally, such as Oracle Virtual Directory, you must include an MBean name as an argument to the command. The name for an Audit MBean is of the form:

```
oracle.as.ovd:type=component.auditconfig,name=auditconfig,instance=INSTANCE,
component=COMPONENT_NAME
```

> **Note:** The Audit MBean in the preceding example should be one continuous string. It is shown on two lines in this document because of space/width limitations in this document.

You must ensure the MBean has the current server configuration before you make any changes to attributes. To do that, you must use the wlst `invoke()` command to load the configuration from Oracle Virtual Directory server to the MBean. After making changes, you must use the `invoke()` command to save the MBean configuration to the Oracle Virtual Directory server and then use the `invoke()` command to load the updated configuration from Oracle Virtual Directory server to the MBean. This process ensures the Oracle Virtual Directory server and the MBean are in sync. In order to use `invoke()` in this way, you must navigate to the Root Proxy MBean in the tree. The name for a Root Proxy MBean is of the form:

```
oracle.as.management.mbeans.register:type=component,name=COMPONENT_
NAME,instance=INSTANCE
```

For example:

```
oracle.as.management.mbeans.register:type=component,name=ovd1,instance=instance1
```

Here is an example of a `wlst` session using `setAuditPolicy()` and `invoke()`:

```
java weblogic.WLST
connect('username','password','t3://WEBLOGIC_HOST:WEBLOGIC_ADMIN_PORT')
custom()
cd(’oracle.as.management.mbeans.register’)
cd(’oracle.as.management.mbeans.register:type=component,name=ovd1,instance=
instance1’)
invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.String)
setAuditPolicy(filterPreset='None',addSpecialUsers="cn=user2,cn=users,dc=oracle,dc
=com",removeSpecialUsers='cn=user1,cn=users,dc=oracle,dc=com',on='oracle.as.ovd:ty
pe=component.auditconfig,name=auditconfig,instance=instance1,component=ovd1')
custom()
cd('oracle.as.management.mbeans.register')
cd
('oracle.as.management.mbeans.register:type=component,name=ovd1,instance=
instance1')
invoke('save',jarray.array([],java.lang.Object),jarray.array([],java.lang.String)
invoke('load',jarray.array([],java.lang.Object),jarray.array([],java.lang.String)
```

Part III

Advanced Administration

This part presents information about advanced administration tasks for Oracle Virtual Directory and contains the following chapters:

- Chapter 18, "Customizing Oracle Virtual Directory"
- Chapter 19, "Configuring Oracle Virtual Directory for Integrated Directory Solutions"
- Chapter 20, "Oracle Communications Universal User Profile"

18

Customizing Oracle Virtual Directory

This chapter explains how to customize Oracle Virtual Directory and contains the following topics:

- Setting Localized Languages for Oracle Directory Services Manager
- Creating and Configuring Custom Adapters
- Developing Custom Java Plug-Ins

18.1 Setting Localized Languages for Oracle Directory Services Manager

Oracle Virtual Directory includes localized translations for the Oracle Directory Services Manager interface in the following languages:

- French
- Italian
- German
- Spanish
- Brazilian Portuguese
- Japanese
- Traditional Chinese
- Simplified Chinese
- Korean

You can set the language for the Oracle Directory Services Manager interface using your web browser's language settings. Refer to your web browser's documentation for specific information on setting languages.

Notes:

- Only the superuser (usually `cn=orcladmin`) can log in to Oracle Directory Services Manager.
- The user name of the superuser used to log in to Oracle Directory Services Manager must be comprised of only ASCII characters. You cannot log in to Oracle Directory Services Manager using a user name of the superuser that contains non-ASCII characters.

18.2 Creating and Configuring Custom Adapters

Oracle Virtual Directory supports the ability to create custom adapters using plug-ins that can connect to almost any data source with a defined API. For example, you can use custom adapters to abstract information available through web services. A custom adapter is an adapter that has no functionality itself—it is a place holder where adapter level plug-ins can be configured to implement its functions instead. By default, Custom Adapters do not map to any data source. Plug-ins, such as the Diameter plug-in, that are added to Custom Adapters on the Plug-In tab in Oracle Directory Services Manager provide data to Custom Adapters. Typically, Custom Adapters are written by customers that must connect Oracle Virtual Directory to non-LDAP or non-database services, such as Web Services.

This topic contains the following sections:

- Creating Custom Adapters
- Configuring Custom Adapters

18.2.1 Creating Custom Adapters

Perform the following steps to create a Custom Adapter using Oracle Directory Services Manager:

1. Log in to Oracle Directory Services Manager.
2. Select **Adapter** from the task selection bar. The Adapter navigation tree appears.
3. Click the **Create Adapter** button. The New Adapter Wizard appears.
4. Perform the following steps to define the Type of adapter:
   a. Select **Custom** from the Adapter Type list.
   b. Enter a unique name for the Custom Adapter in the Adapter Name field. The adapter name value is used in other configuration fields that need to reference the adapter.
   c. Select the **Default** template from the Adapter Template list.
   d. Click **Next**. The Settings screen appears.
5. Enter a valid base DN (in DN format) in the Adapter Suffix/Namespace field. This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in the Adapter Suffix field will be the base DN value that returned entries will have. For example, if you enter dc=mydomain,dc=com in the Adapter Suffix/Namespace field, all entries will end with dc=mydomain,dc=com.
6. Click **Next**. A summary of the Custom Adapter settings appears. Review the settings and click **Finish** to create the Custom Adapter. The Custom Adapter appears in the Adapter tree.

After you create the Custom Adapter you can configure it using the procedures in Configuring Custom Adapters.

18.2.2 Configuring Custom Adapters

This section describes how to configure Custom Adapter settings, including:

- Configuring Custom Adapter General Settings
- Configuring Adapter Routing

- Configuring Adapter Plug-ins and Mappings

18.2.2.1 Configuring Custom Adapter General Settings

After you create the Custom Adapter you can configure the general settings for the adapter by clicking the adapter name in the Adapter tree, clicking the **General** tab, setting values for the following fields, and clicking **Apply**:

Root
This field defines the root DN that the adapter provides information for. The DN defined, and the child entries below it, comprise the adapter's namespace. The value you enter in this field will be the base DN value that returned entries will have. For example, if you enter dc=mydomain,dc=com in the field, all entries will end with dc=mydomain,dc=com.

Active
An adapter can be configured as active (enabled) or inactive (disabled). An adapter configured as inactive will not start during a server restart or an attempted adapter start. Use the inactive setting to keep old configurations available or in stand-by without having to delete them from the configuration. The default setting is active.

18.2.2.2 Configuring Adapter Routing

After you create the adapter you can configure routing for the adapter by clicking the adapter name in the Adapter tree, clicking the **Routing** tab, and referring to "Understanding Routing Settings" on page 3-3.

> **Note:** Enable the Bind Support routing setting when defining Custom Adapters that may or may not support a bind operation.

18.2.2.3 Configuring Adapter Plug-ins and Mappings

After you create the adapter you can apply Plug-ins and Mappings to the adapter by clicking the adapter name in the Adapter tree, clicking the **Plug-Ins** tab, and referring to "Managing Adapter Plug-ins" on page 13-1 and "Applying Mappings to Adapters" on page 14-3.

18.3 Developing Custom Java Plug-Ins

This topic explains how to develop custom Java plug-ins for Oracle Virtual Directory and contains the following section:

- Overview
- Understanding the Chain System
- Plug-In Implementation Points
- Creating EntrySets
- Understanding Filter Processing
- Understanding Classes

18.3.1 Overview

Oracle Virtual Directory allows you to create and deploy custom Java plug-ins that can process and manipulate LDAP operations as they pass through the Oracle Virtual Directory. Plug-ins can be positioned at either a global level, where they see and affect

all requests, or at an adapter level, where they see and affect only requests for a particular adapter. You can also create and deploy plug-ins to run on particular operations and for certain namespaces.

> **Note:** If you rename attributes using custom Java plug-ins, Oracle Virtual Directory supports search on the renamed attribute/value only if the custom code overrides the incoming filter object, as is in the DB_Groups Mapping.

Each Oracle Virtual Directory plug-in has a specific implementation point, as listed in Table 18–1:

Table 18–1 Plug-In Implementation Points

Implementation Point	Description
Configuration	Plug-in configuration data. The custom portion of the configuration consists of name and value pairs of initialization parameters
Startup / Shutdown	The `init(PluginInit initParams, String name)` and `destroy()` methods are called on plug-in initialization and de-initialization.
Availability	The `available(Chain chain, DirectoryString base)` method is called before execution of the plug-in to determine if the plug-in will be executed.
Operations	The various operational methods that will be called.

This chapter demonstrates how to create a custom plug-in by explaining the implementation points listed in Table 18–1. The chapter provides information for a *fictitious example* plug-in called the Bad Password Count plug-in which would detect if a bind operation has failed or succeeded. If the operation succeeded, then the count would be cleared and if the bind fails, then the count would increase. The fictitious Bad Password Count plug-in would also make sure that the bad password count cannot be changed from outside the directory.

> **Note:** The Bad Password Count plug-in described in this chapter is a fictitious example used to demonstrate how Oracle Virtual Directory plug-ins and its chain system operate. Oracle Virtual Directory does not include a Bad Password Count plug-in, though it could support one if you created it.

18.3.2 Understanding the Chain System

Oracle Virtual Directory plug-ins follow an implementation based on the Java Servlet 2.3 Filter model where a single method is used to handle the pre-operation and post-operation, and to determine if an operation should continue. Multiple plug-ins are combined to form a chain of plug-ins. To demonstrate this chain implementation, consider the following situation where the fictitious example Bad Password Count plug-in determines if the bad password count attribute should be added to an entry being added to the directory.

You have the ability to manipulate the request when the add method is called, which allows you to manipulate the passed-in attributes and their values (for example, to change objectclass value `inetOrgPerson` to `user` if you were masking `ActiveDirectory` as a standard LDAP directory) or to handle the storage of the data

into your non-directory or database system (such as in a custom adapter). If you want to allow the virtual directory to have a chance to further process the request through other plug-ins, you would call the `chain.nextAdd` method. Most plug-in methods have corresponding `chain.next<XXX>` methods. If you do not want to allow further processing of the request by plug-ins, you can omit the `chain.next<XXX>` call.

> **Note:** The adapter type is irrelevant to the plug-ins, as plug-ins can be added to any type of adapter.

18.3.3 Plug-In Implementation Points

Before you can build a custom plug-in, you must decide whether to implement the `com.octetstring.vde.chain.Plugin` interface or extend the `com.octetstring.vde.chain.BasePlugin` class. The `BasePlugin` class is a convenience that allows a plug-in developer to only implement the methods for operations to be handled by the plug-in. The example plug-in in this chapter extends the BasePlugin class to simplify the implementation.

The sections in this topic describe the Oracle Virtual Directory plug-in implementation points, including:

- Configuration, Startup, and Shutdown Plug-In Implementation Points
- Availability Plug-In Implementation Point
- Operation Plug-In Implementation Point

18.3.3.1 Configuration, Startup, and Shutdown Plug-In Implementation Points

Configuration is the first plug-in implementation point. Plug-ins are configured using a set of simple name and value pairs provided by the Oracle Virtual Directory configuration system. The pairs are provided to the plug-in developer through the `params` argument to the `init` method of the plug-in. The example plug-in in this chapter includes the following configuration options:

- `countAttribute`: An attribute that will be attached to all user entries that will store the bad password count.
- `addOnCreate`: Boolean value set to true if the plug-in will add this attribute when a user is created.
- `objectClassForAdd`: The object classes that represent users who will have the attribute added to.
- `ignoreOnModify`: Boolean value, set to true if modify requests on the `countAttribute` should be ignored.

The configuration options listed above will be picked-up at the life cycle methods, which is the second implementation point. The `init` method is called on the initialization of the plug-in at server startup and the destroy method is called when the plug-in is being shutdown. Example 18–1 shows an example `init` method:

Example 18–1 Example init Method

```
/**
 * 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");
```

The method in Example 18–1 checks the initialization parameters to setup the plug-in. If there is not enough configuration information, then the plug-in throws an exception, causing the plug-in to not be configured for operational use by the server. The destroy method is not required to be implemented unless there is a need to release any connections or shutdown any services.

18.3.3.2 Availability Plug-In Implementation Point

The available implementation point follows the configuration and startup and shutdown implementation points. The available method is called before each plug-in can be called for a particular LDAP operation. If the available method returns as true, then the plug-in is executed. In Example 18–2, the available method checks for the

existence of the `ignoreOnModify` option in the `Request` object. If it is defined, then the plug-in will be skipped. Similarly, if the `addonCreate` option is set to false, the plug-in will be skipped.

Example 18–2 Example Method Checking for ignoreOnModify Option

```
/**
 * 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;
       }
}
```

If the available method returns as true, the operation portion of the request will be executed.

18.3.3.3 Operation Plug-In Implementation Point

The final implementation point is operation implementations. Consider the following code implementation of a bind operation in Example 18–3:

Example 18–3 Example Bind Operation Implementation

```
/**
 * 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);
              }
        }
}

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);

}
```

The method in Example 18–3 shows an example where password failure counts are being maintained within the directory as a form of password policy. Notice that the method does not perform any pre-processing of the operation, nor does it attempt to take over the bind operation. The plug-ins bind method immediately calls the `chain.nextBind` method and waits for the bind to complete before moving forward with its own logic. Once the bind is complete, that is, control is returned from `chain.nextBind`, the plug-in checks to see if the bind was successful or not. If the bind was successful, the plug-in sets the failure count attribute to zero for the user. If the bind failed, then the current failure count is retrieved and an increased value is set.

The bind method uses the Virtual Services Interface (VSI) to modify records for the binding user. You can use the VSI interface throughout Oracle Virtual Directory as a consistent way to access directory information regardless of whether a plug-in is deployed globally or within the context of an adapter. VSI does this by always calling into Oracle Virtual Directory by starting with the next plug-in in the chain after the current plug-in. For example, if there is a mapper before the plug-in, and a cache after the plug-in, then the call to VSI will go only through the cache.

Because the plug-in is now logically in charge of maintaining the bind failure count, the plug-in modify method must be implemented so that any attempt by an LDAP client to modify the count is blocked. The plug-in modify method in Example 18–4 is implemented to throw an exception if the count attribute is included in the modify change list.

Example 18–4 Example modify Method

```
/**
 * 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);


}
```

A `DirectoryException` is thrown with both a status code and a message. If this exception is not caught by another plug-in, then both the message and the code will make it back to the client. For this example, you do not need to check and see if the `ignoreOnModify` has been configured because you have delegated that decision to the available method. If it was set, the plug-in modify method in Example 18–4 would not have been called.

18.3.3.3.1 Searches Searches are different then the other operations because there is not one, but three methods that may be implemented. The first method, get, acts as the other operation methods do and allows for the plug-in developer to pre-process the search request and post process the returns. While you could process each Entry returned, it would be very inefficient in terms of memory utilization to do so. Processing results in the get method means that all results must be obtained in memory before they can be returned to the client. For this reason there is a `postSearchEntry` method, which is executed for every Entry that will be returned to the client where attributes can be changed, added or modified in an efficient matter. There is also the `postSearchComplete` method that marks that the search operation is complete.

In order to effectively use the `postSearchEntry` processing, Oracle Virtual Directory tends to use a special class known as an `EntrySet` to handle result set processing. A

get method returns results by returning a Vector, which includes one or more EntrySets (refer to "Creating EntrySets" for more information).

18.3.4 Creating EntrySets

Each object in a directory is represented in Oracle Virtual Directory by a `com.octetstring.vde.Entry` object. Each entry contains the name of the object and attributes with attribute values. All entry objects are processed in Oracle Virtual Directory using an implementation of the `com.octetstring.vde.EntrySet` interface. Entry sets store or handle all entries returned by a particular data source. During normal Oracle Virtual Directory processing, each adapter called during the course of a search request will add its own EntrySet implementation to the list of results to be returned by Oracle Virtual Directory. Additionally, it is also possible that a plug-in could insert additional EntrySet objects into the results vector array. After all adapters have been queried to fulfill the search request, each EntrySet is traversed with its entries sent to the client.

While all adapters produce EntrySet implementations, a plug-in may also create an instance of the EntrySet interface and use it to return entries to the client during a search request.

The following are the two means a plug-in can use to create an EntrySet:

- ExtensibleEntrySet
- Custom EntrySet

18.3.4.1 ExtensibleEntrySet

The simplest means a plug-in can use to create an EntrySet is by using the `com.octetstring.vde.backend.extensible.ExtensibleEntrySet` class to create an EntrySet based on a `java.util.Vector` of `Entry` objects. The following is the procedure to do so:

1. Create a new java.util.Vector array.
2. Add all of the Entry objects to the vector.
3. Create a new instance of ExtensibleEntrySet passing the above Vector in the constructor.

Example 18–5 shows an example of plug-in using a Web Service to retrieve a stock price based on a stock symbol. The plug-in is designed to implement the concept of a Custom Adapter, which is an adapter that has no functionality itself and is a place holder where adapter level plug-ins can be configured to implement its functions instead. In this stock service example, the plug-in would be configured against a custom adapter. The plug-in is then responsible for handling all events, which means that you would expect that the stock service plug-in would not call the `chain.getNext()` method.

The get method of the plug-in adds a list of Entry objects (that is, stock prices entries) to a Vector and creates an ExtensibleEntrySet based on that Vector:

Example 18–5 Example EntrySet Creation Using ExtensibleEntrySet

```
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));
}
```

18.3.4.2 Custom EntrySet

While the use of ExtensibleEntrySet is the simplest means to create an EntrySet, it is not the most efficient because it requires that all results be compiled before processing is returned to the client. In this case, a call is made to the service for each term in the filter. A better way to process this request would be to create an EntrySet in such a way as to retrieve new entries as they are requested from the stock service, as in the LDAP Adapter operation.

When the LDAP Adapter EntrySet is asked for the next Entry, the system retrieves the next entry from the remote server one at a time—the way LDAP protocol is intended to work. This approach is more efficient as it allows the client to begin retrieving entries before all entries have been processed. This approach also allows the client to stop retrieval of entries and abort the query.

An implementation of `com.octetstring.vde.EntrySet` must created for a plug-in to create an EntrySet that returns entries as requested. Each EntrySet must implement the following methods:

- `boolean hasMore()`

Returns true if there are more entries in this EntrySet. This method must be non-destructive.

- `Entry getNext()`

  Returns the next entry in the EntrySet or null if there are no more entries.

- `void cancelEntrySet()`

  This method is called when an EntrySet will not be run to completion, allowing a custom EntrySet implementation to release any system resources it was holding.

Example 18–6 is the same plug-in implementation as Example 18–5 that creates an adapter out of a stock ticker Web Service, however, in Example 18–6, the get method only creates a list of symbols which gets passed off to the custom EntrySet. The get method in Example 18–6 adds a list of Entry objects (that is, stock prices entries) to a Vector and creates an ExtensibleEntrySet based on that Vector:

Example 18–6 Example get Method That Passes to a Custom EntrySet

```
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));

}
```

In Example 18–6, a list of stock symbols is created by iterating over the **or** in the search filter. The compiled list is passed to the custom EntrySet implementation as shown in Example 18–7:

Example 18–7 Example of Data Passed to Custom EntrySet

```
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
    }
}
```

The EntrySet implementation in Example 18–7 uses a java.util.Iterator to track which symbol is currently being processed. The `StockEntrySet` class does not call out to the Web Service to create entry results until an Entry is requested by Oracle Virtual Directory on behalf of the client.

Since the plug-in supports the searching of one or more stocks, it is possible that not all searches will return valid results. Consider that if `getNext` returns a `NULL` result to Oracle Virtual Directory before the list of stocks is exhausted, Oracle Virtual Directory prematurely will assume the results are exhausted. To handle this situation, an extra block of code is added to `getNext` after the call to `getStockEntry`. If `getStockEntry` returns a `NULL` and the iteration through the requested stocks has not finished, `getNext` will call itself in order to process the next candidate. This recursion will continue until at least one valid result is returned or all queries are exhausted.

18.3.5 Understanding Filter Processing

LDAP filter possessing can be complicated. In the context of a Oracle Virtual Directory plug-in, there are two instances when it may be useful to parse a filter: pre-process or post-process. Each method offers it's own advantages and disadvantages and is not always mutually exclusive.

Post-Process Filtering

In post-process filtering, the `com.octetstring.vde.util.FilterUtils.evalFilter(Entry e, Filter f)` method is used to see if an entry being returned as a result matches a required filter. This is the simplest way to handle filters and is useful when you are dealing with a small predefined data set that can remain in memory as a collection of Entry objects. This method is not generally the best solution when a filter needs to be translated into another format, for example, into a SQL `WHERE` clause or a special object model for an external API.

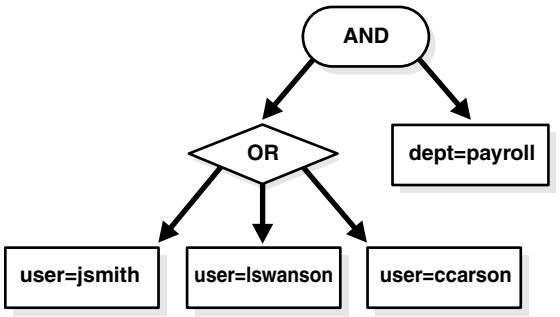
Pre-Process Filtering

Pre-processing filters are used to parse a filter and to apply it to a modified search or transform it to another format that the target of the search can understand. Think of pre-processing filters as converting an LDAP filter to an SQL `WHERE` clause, which to do so, you must traverse the filter object. For example, consider the conversion of the following LDAP filter to an SQL `WHERE` clause:

```
(&(|(user=jsmith)(user=lswanson)(user=ccarson))(dept=payroll))
```

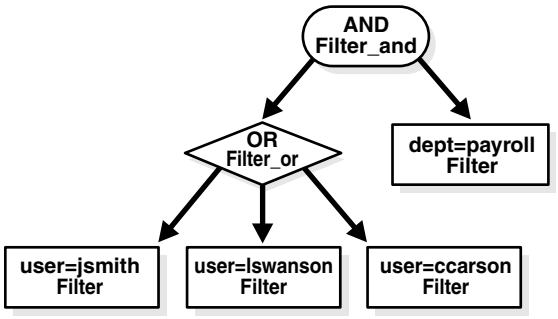
The preceding LDAP filter states *All records where the user is jsmith, lswanson or ccarson and whose department is payroll*. Figure 18–1 shows a visual representation of this LDAP filter:

Figure 18–1 Visual Representation of an Example LDAP Filter



To translate the filter shown in Figure 18–1 into an SQL `WHERE` clause, you use a recursive function that traverses the tree. The example filter is represented in Oracle Virtual Directory as a hierarchy of Filter objects, which contain collections of other Filter objects to create a traversable tree. The filter shown in Figure 18–1 will have the object model shown in Figure 18–2, where the name of the class used to represent the filter element is below the operation or operand:

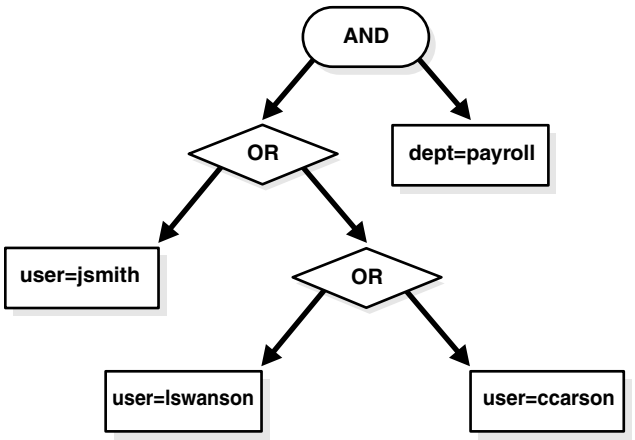
Figure 18–2 Example Object Model of a Filter



To traverse the tree shown in Figure 18–2, a recursive method is used that queries the `getSelector()` method of the filter to determine what type of filter it is. After the type for the filter is determined, its value needs to be extracted by using a `getFilterType` method. For example, if the filter is an equality filter, such as `user=jsmith`, the value of the filter object would come from `currentFilter.getEqualityMatch()`. In this case the return value is an `AttributeValueAssertion`, which stores the attribute name and value as an Oracle. Once retrieved, the values can be converted into String objects. `Filter_and` and `Filter_or` objects return `java.util.Iterator` classes for iterating through the child filters that are being operated on.

LDAP filters do not limit you to two terms per relation. The `OR` portion has three operands. Since SQL only allows two operands per operation, the tree in Figure 18–2 must be converted to a binary tree.

Figure 18–3 Breaking an OR Function



In Figure 18–3, the `OR` operation is broken up into two separate `OR` operations. The final `WHERE` clause from the filter is `((user=jsmith) OR ((user=lswanson) OR (user=ccarson))) AND (dept=payroll)`. The LDAP prefix notation has been transformed into a SQL like infix notation with only two operands per operation. Example 18–8 shows the source code for the transformation:

Example 18–8 Example Source Code for Transforming an LDAP Prefix Notation to SQL Notation

```
import com.octetstring.vde.util.*;
import com.octetstring.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};
       }
}
```

18.3.6 Understanding Classes

The sections in this topic provide a high-level introduction to the Oracle Virtual Directory classes that are available. Refer to the *Oracle Fusion Middleware Java API Reference for Oracle Virtual Directory* Javadoc for complete information on Oracle Virtual Directory classes. This topic contains the following sections:

- Virtual Service Interface
- Global Service Interface
- Adapter Service Interface
- Joiner
- Utility Classes
- Data Classes
- Data Types
- Exceptions

18.3.6.1 Virtual Service Interface

The Virtual Service Interface (VSI) provides methods to make LDAP-like calls into the Oracle Virtual Directory. VSI works the same way regardless of whether the context is a global plug-in or an adapter level plug-in.

If there are three plug-ins in a chain and the first plug-in in the chain calls into VSI, then the context of the call is Oracle Virtual Directory as it appears through the second

and third plug-in. If the call comes from the second plug-in, then it would only go through the third plug-in. If the call originates from the third plug-in, the call will not go through any plug-ins, regardless if the plug-in is global or local. If the plug-in is global, then the call will continue out of the global chain, through the Oracle Virtual Directory routing system into adapter level chains (depending on whether one or more adapters are selected by the router), which guarantees that not only will calls into Oracle Virtual Directory be consistent, but it also protects against infinite loops of a plug-in calling itself. The VSI is retrieved by using the chain object which is passed into every plug-in method.

18.3.6.2 Global Service Interface

The Global Service Interface (GSI) provides methods to make LDAP-like calls into the Oracle Virtual Directory as though they were coming from a end client. Each call is processed through the access control system (if enabled) and offers the ability to let the router select appropriate adapters for an operation. The GSI is the same interface that the LDAP Listener and Web Gateway use to communicate.

GSI can be retrieved by using the VSI by calling `chain.getVSI().getGSI()`. With this handle, the `add`, `bind`, `delete`, `get`, `getByDN`, `modify`, and `rename` methods can be called.

Warning: With the GSI, it is possible for a plug-in to be caught in an infinite loop if it calls to a context above the current plug-in. Doing this can cause a scenario where the plug-in code is called repeatedly causing unanticipated results. Unless you intend for this to happen, be careful of scenarios where plug-ins call up the stack where looping might occur. In general, unless you need to call a specific adapter, it is always safest to use VSI.

Oracle Virtual Directory provides no loop detection mechanisms. If you find that Oracle Virtual Directory has crashed with a custom plug-in due to a stack overflow or memory exhaustion, this is the most likely cause.

18.3.6.3 Adapter Service Interface

The Adapter Service Interface (ASI) provides methods to make LDAP-like calls into the Oracle Virtual Directory at the router level or directly to a specific adapter. The Oracle Virtual Directory Join View Adapter and its Joiners use ASI to communicate with adapters that are being searched and joined. The ASI interface is useful when you want to obtain information from an internal adapter, such as when configured to provide look-up information for a plug-in class.

ASI is retrieved via the VSI by calling `chain.getVSI().getASI()`. With this handle, the `add`, `bind`, `delete`, `get`, `getByDN`, `modify`, and `rename` methods can be called. Each method has two variations: one that provides a parameter for an adapter name, and another without. Use the method with adapter names to select specific adapters or use the other, nameless method to let the router select the appropriate adapters for you based on routing logic and routing configuration.

Warning: With the ASI, it is possible for a plug-in to be caught in an infinite loop if it calls to a context above the current plug-in. Doing this can cause a scenario where the plug-in code is called repeatedly causing unanticipated results. Unless you intend for this to happen, be careful of scenarios where plug-ins call up the stack where looping might occur. In general, unless you need to call a specific adapter, it is always safest to use VSI.

Oracle Virtual Directory provides no loop detection mechanisms. If you find that Oracle Virtual Directory has crashed with a custom plug-in due to a stack overflow or memory exhaustion, this is the most likely cause.

VSI, GSI and ASI all share a common interface, with certain interfaces providing extra functionality. For more information, refer to the *Oracle Fusion Middleware Java API Reference for Oracle Virtual Directory*.

The following list describes the supported ASI methods:

- `add()`

  Performs an LDAP add operation. Two versions of this method allow either the Oracle Virtual Directory Router to select the target adapter, or a specific adapter can be selected.

- `bind()`

  Performs an LDAP bind operation, either letting the Oracle Virtual Directory Router choose the adapter or applying to a specific adapter.

- `delete()`

  Performs an LDAP delete operation either letting the Oracle Virtual Directory Router choose the adapter or applying to a specific adapter.

- `get()`

  Performs an LDAP get operation letting the Oracle Virtual Directory Router choose eligible adapters. The get method returns a java.util.Vector of EntrySet values. An EntrySet is included for each adapter that was queried.

- `getbyDN()`

  A convenience method that performs an LDAP base search using a specific DN. The caller may choose to specify a specific adapter or may let the Oracle Virtual Directory Router choose.

- `modify()`

  Performs an LDAP modify operation. The caller may specify a specific adapter or may elect to have the Router choose automatically.

- `rename()`

  Performs an LDAP rename operation. The caller may specify specific from and to adapters or may elect to have the Router choose automatically.

18.3.6.4 Joiner

The Oracle Virtual Directory Join View Adapter uses Joiners to join entries from a specific adapter and to merge them with entries from a primary adapter. A Joiner is an abstract class that defines the basic operations and methods required to implement a

new Joiner. Joiners are called by the Join View Adapter whenever operations need to be performed against a joined entry. Joiners define pre-action operations to allow manipulation of data prior to any LDAP operation. Joiners also define `mapOperationTargetByEntry` methods that allow the it to select a target entry in the target joined adapter depending on the operation being called.

A Joiner is instantiated with a primary adapter and a target adapter. The Join View Adapter always works in the context of the primary adapter and calls Joiner methods when mapping and when manipulations must be performed on a target joined adapter.

The get operation of the Join View Adapter builds a `JoinEntrySet` based solely on results from the primary adapter. As the Oracle Virtual Directory client subsequently polls for results from the Oracle Virtual Directory, the `JoinEntrySet` class calls the joiner `JoinByEntry` method to make a call to the joined adapter and merge the entry results. If more than one join relationship is configured, the entry set processing will loop through all of the joins until the entry is fully joined based on all defined relationships.

The Joiner constructor method is called when the Joiner is instantiated by the Join View Adapter. This does not happen until the first LDAP operation is processed by the Join View Adapter (a form of lazy construction). The constructor is passed the configuration parameters for the joiner from the configuration file along with the associated target adapter name.

The `createJoinFilter` method is usually a local method called by the `JoinByEntry` method to create a search filter for a subsequent call to the `AdapterServiceInterface`.

18.3.6.5 Utility Classes

Oracle Virtual Directory supports the following utility classes:

- `PluginUtils`

  This class is a basic toolbox of mapping functions including `renameAttribute`, `copyAttribute`, and others. These classes are normally used in mapping scripts but are available for use in Java Plug-ins.

- `FilterTools`

  This class provides methods for creating and manipulating LDAP search filters.

- `ParseFilter`

  The `ParseFilter` class provides ability to convert a String to a Filter and back again.

- `DNUtility`

  This class provides DN manipulation routines such as explode and create dn allowing manipulation of individual DN name components.

- `LDAPResult`

  The `LDAPResult` is a utility class that can be used to compare the results returned from any method that returns an `Int8` value. These constants help you translate result codes into LDAP error codes.

- `VDELogger`

  The Logger class provides an interface into the Oracle Virtual Directory logging facility (`Log4J`). Use this class to integrate your console or audit messages with those of Oracle Virtual Directory.

- `PasswordEncryptor`

  This class provides various methods to encrypt string values into various hashed formats including Crypt, SHA, SSHA.

18.3.6.6 Data Classes

Oracle Virtual Directory supports the following utility classes:

- `Attribute`

  `Attribute` is a basic object used in conjunction with the Entry class. An attribute defines a type (as in the attribute name) and contains its values. Methods are also provided for cloning and equivalence testing.

- `Credentials`

  A basic object holding the credentials of a session. The IP address, binddn, and password if needed, are in this object. Normally, for most operations relating to the `AdapterServiceInterface`, only `binddn` is relevant.

- `Entry`

  This object is used to hold an LDAP entry and it is used to contain partial entries such as with an LDAP modify request. The `FilterTool` utilities often work with these objects to test filters.

- `EntryChange`

  This object contains an LDAP modify item. When handling modification requests, usually a Vector of EntryChange objects are passed to the `AdapterServiceInterface`. Each EntryChange contains a single modification to a single entry.

- `EntrySet`

  An EntrySet contains a set of query results from an adapter. When a method first receives an EntrySet, the entire result set may not be in memory. Unique Entry objects are returned from an EntrySet by calling its `getNext` method. Each time `getNext` is called, the relevant adapter or plug-in class code is called to retrieve the next Entry if there is one. To test the availability of another entry, use the `hasMore()` method.

  > **Tip:** Unless you intend to process an entire result set, you should avoid calling `getNext()` directly. It is always better to let the LDAP client do this. In the case of a Oracle Virtual Directory plug-in class, a special method, `postSearchEntry()`, is provided giving the ability to modify each entry as it is returned to the client. Needlessly calling `getNext()` can cause excessive memory use and performance loss as Oracle Virtual Directory will be required to load an entire result set at once, rather than process entries as they arrive from the adapters.

- `Filter`

  The `Filter` object is a representation of a standard LDAP filter. This object provides useful methods for setting, testing, and comparing LDAP filters. The Filter object may contain a hierarchy of other filter objects (for example, `Filter_and`, `Filter_or`).

- `LDAPURL`

This class provides methods to parse a standard `LDAPURL` or to create one.

18.3.6.7 Data Types

Oracle Virtual Directory supports the following data types:

- `BinarySyntax`

  Any binary value such as a password or user certificate.

- `DirectoryString`

  A case-insensitive string.

- `IASString`

  A case-sensitive string.

- `IntegerSyntax`

  An integer value

- `DistinguishedName`

  A distinguished name value (comparisons will follow DN equality rules).

18.3.6.8 Exceptions

Oracle Virtual Directory supports the following exceptions:

- `DirectoryBindException`

  Exception thrown when a bind is unsuccessful.

- DirectoryException

  A general exception thrown during any directory error. Check `getMessage()` for more information, or `getLDAPErrorCode()` to determine the LDAP error code. This exception may be generated by an adapter or by another plug-in.

- `DirectorySchemaViolation`

  A schema violation occurs when an attempted add or modify of an entry that violates either remote schema or local schema.

- `InvalidDNException`

  An `InvalidDNException` is thrown by objects and utilities whenever an invalid DN is passed as a parameter.

19

Configuring Oracle Virtual Directory for Integrated Directory Solutions

This chapter explains how to configure Oracle Virtual Directory for integration with commonly used directory and identity management technologies and contains the following topics:

- Configuring Oracle Virtual Directory for Oracle Access Manager
- Integrating with Oracle's Enterprise User Security
- Integrating with Oracle's Net Services

> **Note:** Oracle Virtual Directory can be used with most LDAP-enabled technologies. The information in this chapter highlights Oracle Virtual Directory features and capabilities that simplify common integrations. Contact your Oracle support representative for assistance with other Oracle Virtual Directory integrations.

19.1 Configuring Oracle Virtual Directory for Oracle Access Manager

Perform the following steps to configure Oracle Virtual Directory for integration with Oracle Access Manager (OAM) using Oracle Directory Services Manager's Setup for Oracle Access Manager Quick Config Wizard. The Setup for Oracle Access Manager Quick Config Wizard walks you through the steps to create the required Local Store Adapter and also the appropriate adapter type, either LDAP, Database, or Custom, for the data repository that Oracle Access Manager uses.

1. Log in to Oracle Directory Services Manager.
2. Select **Advanced** from the task selection bar. The Advanced navigation tree appears.
3. Expand the **Quick Config Wizards** entry in the Advanced tree.
4. Click **Setup for Oracle Access Manager** in the tree. The Setup for Oracle Access Manager screen appears.
5. Enter the namespace for the Local Store Adapter in DN format in the Namespace used for creating Local Store Adapter (LSA) field and click **Apply**. The Adapters screen appears.
6. Create an adapter for the data repository that Oracle Access Manager uses. Perform one of the following procedures that is appropriate for the data repository that Oracle Access Manager uses:

To create an LDAP Adapter for Oracle Access Manager, perform the following steps:

a. Click the **Create OAM LDAP Adapter** button. The Preparing OVD for OAM - Create LDAP Adapter dialog box appears.

b. Enter a unique name for the LDAP Adapter in the Adapter Name field. Select the appropriate template for the LDAP Adapter by choosing an option from the Adapter Template list. Choose **Default** if you are not integrating with Microsoft Active Directory or Sun Java System Directory Server. Refer to "Understanding Adapter Templates" on page 2-28 for more information. Click **Next.** The Connection screen of the Preparing OVD for OAM - Create LDAP Adapter dialog box appears.

c. Perform steps 5–16 in "Creating LDAP Adapters" on page 12-1 to configure the LDAP Adapter for OAM.

d. Review the summary of settings and click **Finish** to create the LDAP Adapter for OAM. The new LDAP Adapter for OAM appears in the list of adapters on the Setup for Oracle Access Manager screen.

To create a Database Adapter for Oracle Access Manager, perform the following steps:

a. Click the **Create OAM Database Adapter** button. The Preparing OVD for OAM - Create Database Adapter dialog box appears.

b. Enter a unique name for the Database Adapter in the Adapter Name field. Select the appropriate template for the Database Adapter by choosing an option from the Adapter Template list. Refer to "Understanding Adapter Templates" on page 2-28 for more information. Click **Next.** The Connection screen of the Preparing OVD for OAM - Create Database Adapter dialog box appears.

c. Perform steps 5–10 in "Creating Database Adapters" on page 12-13 to configure the Database Adapter for OAM.

d. Review the summary of settings and click **Finish** to create the Database Adapter for OAM. The new Database Adapter for OAM appears in the list of adapters on the Setup for Oracle Access Manager screen.

To create a Custom Adapter for Oracle Access Manager, perform the following steps:

a. Click the **Create OAM Custom Adapter** button. The Preparing OVD for OAM - Create Custom Adapter dialog box appears.

b. Enter a unique name for the Custom Adapter in the Adapter Name field.

c. Enter a valid base DN in the Adapter Suffix/Namespace field.

d. Click **Next** on the Preparing OVD for OAM - Create Custom Adapter dialog box. The Configure plug-in screen appears.

e. Enter a name for the Plug-in in the Name field.

f. Enter the Plug-in class name in the Class field, or click **Browse**, then select the plug-in from the Plug-In Selection box, and then click **OK**.

g. Add parameters and values to the Plug-in by clicking the **Create** button in the Parameters table, selecting a parameter from the Name list, and entering a value for the parameter in the Value field.

h. Click the **Next** on the Configure plug-in screen.

i. Review the summary of settings and click **Finish** to create the Custom Adapter for OAM. The new Custom Adapter for OAM appears in the list of adapters on the Setup for Oracle Access Manager screen.

7. Configure the adapter for the data repository that Oracle Access Manager uses by selecting **Adapter** from the Oracle Directory Services Manager task selection bar and then clicking the name of the adapter that you want to configure in the Adapter tree.

See Also: The following sections for more information on configuring each type of adapter:

- "Configuring LDAP Adapters" on page 12-6
- "Configuring Database Adapters" on page 12-19
- "Configuring Custom Adapters" on page 18-2

19.1.1 Modifying Oracle Access Manager Adapter Settings

To modify settings for an Oracle Access Manager integration adapter:

1. Click the name of the adapter you want to modify on the Setup for Oracle Access Manager page. The adapter's settings appear at the bottom of the page.
2. Modify the appropriate adapter settings. Refer to Chapter 12, "Creating and Configuring Oracle Virtual Directory Adapters" for more information on adapter settings.
3. Click **Apply** at the bottom of the adapter settings screen to apply the changes.

19.2 Integrating with Oracle's Enterprise User Security

Integrating Oracle Virtual Directory and Enterprise User Security enhances and simplifies your authentication and authorization capabilities by allowing you to leverage user identities stored in an external LDAP repository without any additional synchronization.

This topic describes how to integrate Oracle Virtual Directory with Oracle's Enterprise User Security and contains the following sections:

- Preparing Oracle Virtual Directory for the Enterprise User Security Integration
- Integrating Oracle Virtual Directory with External Directories
- Configuring Access Control Lists for the Enterprise User Security Integration
- Configuring Oracle Virtual Directory to Support Multiple Enterprise User Security Domains
- Enabling User Account Lockout
- Integration Limitations

19.2.1 Preparing Oracle Virtual Directory for the Enterprise User Security Integration

Regardless of which external directory you are storing your user identities in, you must perform the steps in this section first. After you complete the steps in this section, proceed with the integration by referring to Integrating Oracle Virtual Directory with External Directories.

Perform the following steps to prepare Oracle Virtual Directory for integration with with Enterprise User Security:

1. Create a backup copy of the *ORACLE_HOME*/ovd/eus/ directory. All the configuration files required for the Enterprise User Security integration are in the eus directory. Making a backup copy of the eus directory allows you to edit the template-like files in the original eus directory based on your environment, and still keep copies of the original files.
2. If one does not already exist, create an LDAP listener that is secured with SSL No Authentication Mode by referring to Chapter 11, "Creating and Managing Oracle Virtual Directory Listeners."
3. Create and add the subschemasubentry and Dynamic Groups plug-ins as global server plug-ins. Refer to "Managing Global Server Plug-ins" on page 13-4 for steps on creating server plug-ins.

Important: The steps for integrating Oracle Virtual Directory with Enterprise User Security from this point forward differ depending on which external directory you are storing your user identities in.

Continue the integration with Enterprise User Security by referring to Integrating Oracle Virtual Directory with External Directories.

19.2.2 Integrating Oracle Virtual Directory with External Directories

This section contains instructions for integrating Oracle Virtual Directory with Enterprise User Security for use with specific external directories. Perform the steps in the appropriate section that are specific to the external directory you are storing your user identities in. This sections contains the following sections:

- User Identities in Microsoft Active Directory
- User Identities in Microsoft Active Directory and Metadata in Oracle Internet Directory
- User Identities in Sun Java System Directory Server
- User Identities in Novell eDirectory
- User Identities in Oracle Internet Directory

19.2.2.1 User Identities in Microsoft Active Directory

Perform the following procedures to integrate Oracle Virtual Directory with Enterprise User Security for user identities stored in Active Directory:

- Configuring Active Directory for the Integration
- Configuring Oracle Virtual Directory for the Integration

19.2.2.1.1 Configuring Active Directory for the Integration Perform the following steps to configure Active Directory for the integration:

Note: If you are using Kerberos authentication in the integration, do not perform steps 3 and 4 in the following procedure.

---

1. Make a backup copy of your Active Directory image. The schema extensions inside of Active Directory are permanent and cannot be cancelled. The backup image allows you to restore all your changes if required.
2. Load the Enterprise User Security required schema, extendAD, into Active Directory using the Java classes included in Oracle Virtual Directory by executing the following command. The extendAD file is located in the $*ORACLE_HOME*/ovd/eus/ directory. You can use the java executable in the *ORACLE_HOME*/jdk/bin directory.

   ```
   java extendAD -h Active_Directory_Host_Name -p Active_Directory_Port
   -D Active_Directory_Admin_DN -w Active_Directory_Admin_Password
   -AD Active_Directory_Domain_DN
   ```

   Note: An example of a valid Active Directory domain DN is: `dc=oracle,dc=com`

3. Install the Oracle Internet Directory Password Change Notification plug-in, oidpwdcn.dll, by performing the following steps:
   a. Copy the $*ORACLE_HOME*/ovd/eus/oidpwdcn.dll file to the Active Directory WINDOWS\system32 directory.
   b. Use regedt32 to edit the registry and enable the oidpwdcn.dll. Start regedt32 by entering regedt32 at the command prompt.
   c. Add oidpwdcn to the end of the Notification Packages entry in the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\ registry, for example:

      ```
      RASSFM
      KDCSVC
      WDIGEST
      scecli
      oidpwdcn
      ```

   d. Restart the Active Directory system after making these changes.
4. Verify the Oracle Internet Directory Password Change Notification plug-in by performing the following steps:
   a. Change the password of an Active Directory user.
   b. Search Active Directory for the user you changed the password for. Verify the orclCommonAttribute attribute contains the generated hash password value.
   c. Reset the password for all the Active Directory users, allowing the plug-in to acquire the password changes and generate and store password verifiers.
5. If you are using Kerberos authentication on Windows 2000 or Windows 2003 with Oracle Database Advanced Security, you must configure it now by referring to the *Oracle Database Advanced Security Administrator's Guide*.

   After you configure the Kerberos authentication, make sure you can log in to the database using your Active Directory user credential before proceeding to the next steps.

19.2.2.1.2 Configuring Oracle Virtual Directory for the Integration Perform the following steps to configure Oracle Virtual Directory for the integration:

1. Ensure you have performed all steps in "Preparing Oracle Virtual Directory for the Enterprise User Security Integration" on page 19-3 before proceeding with this procedure.
2. Start the Oracle Virtual Directory server, then start Oracle Directory Services Manager, and then connect to the Oracle Virtual Directory server.
3. Create three new Local Store Adapters using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.
   - Use the **Local_Storage_Adapter** template for each adapter.
   - The **Adapter Suffix** for one of the Local Store Adapters must be `cn=OracleContext`; the **Adapter Suffix** for another of the Local Store Adapters must be `cn=OracleSchemaVersion`; and the **Adapter Suffix** for the other the Local Store Adapters must be `dc=com`, unless your Active Directory domain is something like `dc=example,dc=net`, in which case the **Adapter Suffix** must be `dc=net`.
   - The **Database File** and **Backup File** fields for each of the adapters must be unique.
4. Update and load the entries into the Local Store Adapters by performing the following steps:
   a. Extend the Oracle Virtual Directory schema with the loadOVD.ldif file using the following command. The loadOVD.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f loadOVD.ldif
      ```

      Note: The loadOVD.ldif file contains entries for Oracle Context and schema version that Enterprise User Security queries.

   b. Update realmRoot.ldif to use your namespaces, including the dn, dc, o, orclsubscriberfullname, and memberurl attributes in the file. If you have a DN mapping between Active Directory and Oracle Virtual Directory, use the DN that you see from Oracle Virtual Directory. The realmRoot.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      Note: The realmRoot.ldif file contains core entries in the directory namespace that Enterprise User Security queries. The realmRoot.ldif file also contains the dynamic group that contains the registered Enterprise User Security databases to allow secured access to sensitive Enterprise User Security related attributes, like the user's Enterprise User Security hashed password attribute.

   c. Load your domain root information in the realmRoot.ldif file into Oracle Virtual Directory using the following command:

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f realmRoot.ldif
      ```

5. Create an LDAP Adapter for Enterprise User Security using the following settings and by entering the Active Directory host information, including the appropriate

Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.

- Use the **EUS_ActiveDirectory** template for the adapter.
- Enable the **Use SSL/TLS** option.

6. Configure the Enterprise User Security plug-ins by performing the following steps:

   a. Click the **Advanced** tab, click the EUS_ActiveDirectory entry under Mapping Templates, and then click the **Apply** to deploy the mapping.

   b. Access the LDAP Adapter for Enterprise User Security and click the **Plug-ins** tab.

   c. Select the ObjectclassMapper plug-in, click the **Edit** button, click the **Create Namespace** button, enter `cn=OracleContext,<YOUR DOMAIN NAME>` in the Namespace field, and then click the **OK** button.

   d. Select the ActiveDirectory Password plug-in, click the **Edit** button, click the **Create Namespace** button, and enter `cn=OracleContext,<YOUR DOMAIN NAME>` in the Namespace field.

      Click the **Create Namespace** button again, enter `cn=users,<YOUR DOMAIN NAME>` in the Namespace field, and then click the **OK** button.

   e. Optionally, if you want to enforce account lockout for the users in Active Directory according to Active Directory's account lockout policies, you must set the ADLockoutDuration plug-in parameter in the EUSActiveDirectory plug-in.

      Query the Active Directory domain to determine the value for that specific domain using the following command:

      ```
      ORACLE_HOME/bin/ldapsearch -h Active_Directory_Host_Name -D bindDN \
      -q -s base -b Active_Directory_Domain objectclass="*" lockoutDuration
      ```

      Click the **EUSActiveDirectory** plug-in, then click the **Create New Parameter** button, then select **ADLockoutDuration** and enter the lockout duration value in the Parameter field, and click the **OK** button. Be sure you use the exact same value returned from Active Directory. For example, if the value returned from Active Directory is lockoutDuration=-18000000000, enter `-18000000000` in the Parameter field regardless of the value being negative.

   f. Click the **Create Mapping** button, then select **EUSActiveDirectory.py**, then enter a unique mapping name, then click the **Create Namespace** button, then enter `cn=users,<YOUR DOMAIN NAME>` in the Namespace field, and then click the **OK** button.

   g. Click the **Apply** button.

7. Configure the Access Control Lists (ACLs) for the integration. Refer to "Configuring Access Control Lists for the Enterprise User Security Integration" on page 19-21 for details about each ACL. After you configure the ACLs, continue the integration by proceeding to step 8.

8. Create an LDAP Adapter for the Enterprise User Security administrative group using the following settings and by entering the Active Directory host information. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.

   - Use the **Active_Directory** template for the adapter.

- Use `cn=OracleContextAdmins,cn=users, <YOUR Active_Directory_Domain_DN>` as the Remote Base.
- Use the following for the Mapped Namespace:

```
cn=OracleContextAdmins,cn=Groups,cn=OracleContext,<YOUR Mapped DOMAIN DN in Oracle Virtual Directory>
```

9. Configure the mappings and plug-ins for the Enterprise User Security administrative group adapter by performing the following steps:
   a. Click the **Advanced** tab, then click **Active_Directory_to_inetOrg**, and then click the **Apply** button to deploy the mapping.
   b. Click the **Adapter** tab, then click the adapter for the Enterprise User Security administrative group, then click the **Plug-ins** tab, then click the **Create Mapping** button, then select **ActiveDirectorytoinetOrg.py**, then enter a unique mapping name, and then click **OK**.
   c. Click the **Create Plugin** button, then click the **Select** button, then select the **EUSMemberDNMapping** plug-in, then click **OK**, then enter a unique plug-in name, then create the `localDomainDN` and `remoteDomainDN` parameters, and then click **OK**. Note that the `localDomainDN` and `remoteDomainDN` may be different if you have DN mapping configured.
   d. Click the **Apply** button.

   Note: You may not see the group membership changes immediately after your changes in Active Directory. This is because of Active Directory's group membership refresh interval configuration.

10. Update the realm information with Root Oracle Context by performing the following steps:
    a. Edit the modifyRealm.ldif file to use your Active Directory domain name. If you use DN mappings between Oracle Virtual Directory and Active Directory, use the mapped DN in Oracle Virtual Directory.
    b. Update the realm information using the following command:

```
ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p port \
-D bindDN -q -v -f modifyRealm.ldif
```

    Note: To update the Active Directory-Oracle Virtual Directory configuration, edit the modifyRealm.ldif file and execute ldapmodify with the updated modifyRealm.ldif file.

The steps to configure Oracle Virtual Directory for integration with Enterprise Security and for use with Microsoft Active Directory are complete. Continue the integration process and configure Enterprise User Security by referring to the *Oracle Database Enterprise User Administrator's Guide*.

19.2.2.2 User Identities in Microsoft Active Directory and Metadata in Oracle Internet Directory

Perform the following steps to integrate Oracle Virtual Directory with Enterprise User Security when user identities are stored in Active Directory and to store metadata in Oracle Internet Directory:

Note: If you are using Kerberos authentication in the integration, do not perform steps 6 and 7 in the following procedure.

1. Create a backup copy of the *ORACLE_HOME*/ovd/eus/ directory. All the configuration files required for the Enterprise User Security integration are in the eus directory. Making a backup copy of the eus directory allows you to edit the template-like files in the original eus directory based on your environment, and still keep copies of the original files.
2. If one does not already exist, create an LDAP listener that is secured with SSL by referring to Chapter 11, "Creating and Managing Oracle Virtual Directory Listeners.".
3. Create and add the Dynamic Groups plug-ins as global server plug-ins. Refer to "Managing Global Server Plug-ins" on page 13-4 for steps on creating server plug-ins.
4. Make a backup copy of your Active Directory image. The schema extensions inside of Active Directory are permanent and cannot be cancelled. The backup image allows you to restore all your changes if required.
5. Load the Enterprise User Security required schema into Active Directory using the Java classes included in Oracle Virtual Directory by executing the following command. You can use the java executable in the *ORACLE_HOME*/jdk/bin directory.

   ```
   java extendAD -h Active_Directory_Host_Name -p Active_Directory_Port
   -D Active_Directory_Admin_DN -w Active_Directory_Admin_Password
   -AD Active_Directory_Domain_DN -commonattr
   ```

   Note: An example of a valid Active Directory domain DN is: `dc=oracle,dc=com`

6. Install the Oracle Internet Directory Password Change Notification plug-in, oidpwdcn.dll, by performing the following steps:
   a. Locate the oidpwdcn.dll file and copy it to the Active Directory WINDOWS\system32 directory.
   b. Use regedt32 to edit the registry and enable the oidpwdcn.dll. Start regedt32 by entering regedt32 at the command prompt.
   c. Add oidpwdcn to the end of the Notification Packages entry in the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\ registry, for example:

      ```
      RASSFM
      KDCSVC
      WDIGEST
      scecli
      oidpwdcn
      ```

   d. Restart the Active Directory system after making these changes.
7. Verify the Oracle Internet Directory Password Change Notification plug-in by performing the following steps:
   a. Change the password of an Active Directory user.

b. Search Active Directory for the user you changed the password for. Verify the orclCommonAttribute attribute contains the generated hash password value.

c. Reset the password for all the Active Directory users, allowing the plug-in to acquire the password changes and generate and store password verifiers.

8. If you are using Kerberos authentication on Windows 2000 or Windows 2003 with Oracle Database Advanced Security, you must configure it now by referring to the *Oracle Database Advanced Security Administrator's Guide*.

   After you configure the Kerberos authentication, make sure you can log in to the database using your Active Directory user credential before proceeding to the next steps.

9. Extend the Oracle Internet Directory LDAP attribute and objectclass using the following command:

```
ORACLE_HOME/bin/ldapmodify -h OID_Host_Name -p OID_Port -D bindDN \
-q -v -f OIDSchema.ldif
```

10. Create four new LDAP Adapters using the following settings and by entering the Oracle Internet Directory host information. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.

    For the first **three** new LDAP Adapters:

    - Use the **Oracle_Internet_Directory** adapter template.
    - The Adapter **Remote Base** and **Mapped Namesapce** for the first adapter must be `cn=OracleContext`.
    - The Adapter **Remote Base** and **Mapped Namesapce** for the second adapter must be `cn=OracleSchemaVersion`
    - The Adapter **Remote Base** and **Mapped Namesapce** for the third adapter must be `dc=subschemasubentry`.

    For the **fourth** new LDAP Adapter:

    - Use the **EUS_OID** adapter template.
    - The Adapter **Remote Base** and **Mapped Namesapce** for the fourth adapter must be `cn=oraclecontext,`*`your_OID_realm`*.

11. Create a new Local Store Adapter using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.

    - Use the **Local_Storage_Adapter** template.
    - The **Adapter Suffix** must be `dc=com`, unless your Oracle Internet Directory realm is something like `dc=example,dc=net`, in which case the **Adapter Suffix** must be `dc=net`.

12. Update realmRoot.ldif to use your namespaces, including the dn, dc, o, orclsubscriberfullname, and memberurl attributes in the file. If you have a DN mapping between Active Directory and Oracle Virtual Directory, use the DN that you see from Oracle Virtual Directory.

> **Note:** The realmRoot.ldif file contains core entries in the directory namespace that Enterprise User Security queries. The realmRoot.ldif file also contains the dynamic group that contains the registered Enterprise User Security databases to allow secured access to sensitive Enterprise User Security related attributes, like the user's Enterprise User Security hashed password attribute.

13. Load your domain root information in the realmRoot.ldif file into Oracle Virtual Directory using the following command:

```
ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
-D bindDN -q -v -a -f realmRoot.ldif
```

14. Create a new LDAP Adapter for the user search base in Active Directory using the following settings and by entering the Active Directory host information, including the Remote Base. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.
    - Use the **EUS_ActiveDirectory** template for the adapter.
    - For Remote Base, enter the container in Active Directory, for example: `cn=users,dc=adrealm,dc=com`
15. Check if the EUSActiveDirectory.py mapping is already deployed. If it is, go to step 16 now.

    If the EUSActiveDirectory.py mapping is not deployed, you must create a mapping for the Active Directory user search base adapter by clicking the **Create Mapping** button, then select **EUSActiveDirectory.py**, then enter a unique mapping name, then click the **OK** button, and then click the **Apply** button.
16. Add the Mapped Namespace to the orclcommonusersearchbase under `cn=Common,cn=Products,cn=oraclecontext,<OID realm>`. You can use an LDIF file such as:

```
dn: cn=Common,cn=Products,cn=oraclecontext,dc=oracle,dc=com
changetype: modify
add: orclcommonusersearchbase
orclcommonusersearchbase: cn=users,dc=adrealm,dc=com
```

17. Create the following ACLs. Refer to "Creating Access Control Lists Using Oracle Directory Services Manager" on page 16-1 for information on creating ACLs. If you have customized your ACLs after installing Oracle Virtual Directory, you must adjust the following ACL settings to include your customizations.

Target DN	`cn=subschemasubentry`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=subschemasubentry`
Scope	subtree
Applies To	All Attributes

Grant	Search and Read
Access	Public

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`dc=com`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`dc=com`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read

Access	Public

Target DN	`dc=com`
Scope	subtree
Applies To	authpassword
Deny	All operations
Access	Public

Note: The following ACL must be the last ACL in the ACL list for dc=com.

Target DN	`dc=com`
Scope	subtree
Applies To	authpassword
Grant	Search and Read
Access	Group with DN of: `cn=EUSDBGroup,<Your Mapped OID domain>`.

18. Set the ACLs in Oracle Virtual Directory to support the OracleContextAdmins administrative group as follows:

Target DN	`cn=OracleContext,<YOUR DOMAIN>`
Scope	subtree
Applies To	Entry
Grant	All
Access	Group with DN of: `cn=OracleContextAdmins,cn=Groups,cn=OracleContext,<YOUR DOMAIN>`

Target DN	`cn=OracleContext,<YOUR DOMAIN>`
Scope	subtree
Applies To	All Attributes
Grant	All
Access	Group with DN of: `cn=OracleContextAdmins,cn=Groups,cn=OracleContext,<YOUR DOMAIN>`

19. Set the ACLs in the Oracle Internet Directory to protect the data under `cn=OracleContext,<YOUR DOMAIN>`.

19.2.2.3 User Identities in Sun Java System Directory Server

Perform the following procedures to integrate Oracle Virtual Directory with Enterprise User Security for user identities stored in Sun Java System Directory Server:

- Configuring Sun Java System Directory Server for the Integration
- Configuring Oracle Virtual Directory for the Integration

19.2.2.3.1 Configuring Sun Java System Directory Server for the Integration Perform the following steps to configure Sun Java System Directory Server for the integration:

1. Extend the iPlanet LDAP attribute and objectclass using the following command:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./iPlanetSchema.ldif
```

2. Create a realm in iPlanet by performing the following steps:
   a. Open the realmiPlanet.ldif file and replace all instances of the `dc=us,dc=oracle,dc=com` string with the name of your domain.
   b. Run the following command to create a realm in iPlanet using the realmiPlanet.ldif file:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./realmiPlanet.ldif
```

3. Configure the user and group containers by either creating new user and group containers, or using existing user and group containers.

   Creating New User and Group Containers

   a. Open the iPlanetContainers.ldif file and replace all instances of the `dc=us,dc=oracle,dc=com` string with the name of your domain.
   b. Run the following command to create user and group containers in iPlanet using the iPlanetContainers.ldif file:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./iPlanetContainers.ldif
```

   Using Existing User and Group Containers

   a. Open the useiPlanetContainers.ldif file.
   b. Replace all instances of the `cn=users,dc=us,dc=oracle,dc=com` string with the name of your user container.
   c. Replace all instances of the `cn=groups,dc=us,dc=oracle,dc=com` string with the name of your group container.

   Note: Make sure the user and group containers are in the same domain and realm you are creating. For example, if your domain is `dc=superdemo,dc=net`, then `ou=people,dc=ultrademo,dc=org` is not a valid user container.

   d. Run the following command to create a realm in iPlanet using the useiPlanetContainers.ldif file:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./useiPlanetContainers.ldif
```

19.2.2.3.2 Configuring Oracle Virtual Directory for the Integration Perform the following steps to configure Oracle Virtual Directory for the integration:

1. Ensure you have performed all steps in "Preparing Oracle Virtual Directory for the Enterprise User Security Integration" on page 19-3 before proceeding with this procedure.
2. Start the Oracle Virtual Directory server, then start Oracle Directory Services Manager, and then connect to the Oracle Virtual Directory server.
3. Create three new Local Store Adapters using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.
   - Use the **Local_Storage_Adapter** template for each adapter.
   - The **Adapter Suffix** for one of the Local Store Adapters must be `cn=OracleContext`; the **Adapter Suffix** for another of the Local Store Adapters must be `cn=OracleSchemaVersion`; and the **Adapter Suffix** for the other the Local Store Adapters must be `dc=com`, unless your Sun Java System Directory domain is something like `dc=example,dc=net`, in which case the **Adapter Suffix** must be `dc=net`.
   - The **Database File** and **Backup File** fields for each of the adapters must be unique.
4. Update and load the entries into the Local Store Adapters by performing the following steps:
   a. Extend the Oracle Virtual Directory schema with the loadOVD.ldif file using the following command. The loadOVD.ldif file contains entries for Oracle Context and schemaversion that Enterprise User Security queries. The loadOVD.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f loadOVD.ldif
      ```

   b. Update realmRoot.ldif to use your namespaces, including the dn, dc, o, orclsubscriberfullname, and memberurl attributes in the file. If you have a DN mapping between Sun Java System Directory Server and Oracle Virtual Directory, use the DN that you see from Oracle Virtual Directory. The realmRoot.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      Note: The realmRoot.ldif file contains core entries in the directory namespace that Enterprise User Security queries. The realmRoot.ldif file also contains the dynamic group that contains the registered Enterprise User Security databases to allow secured access to sensitive Enterprise User Security related attributes, like the user's Enterprise User Security hashed password attribute.

   c. Load your domain root information in the realmRoot.ldif file into Oracle Virtual Directory using the following command:

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f realmRoot.ldif
      ```

5. Create an LDAP Adapter for Enterprise User Security using the following settings and by entering the Sun Java System Directory Server host information, including the appropriate Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.
   - Use the **EUS_Sun** template for the adapter.

- The proxy DN user must be able to read the `userPassword` attribute in the Sun Java System Directory Server.

After creating the LDAP Adapter for Enterprise User Security, DBCA adds a user under `cn=oraclecontext, <YOUR DOMAIN NAME>`. Make sure this user can read the `userPassword` attribute in the Sun Java System Directory Server.

6. Configure the Enterprise User Security plug-ins by performing the following steps:
   a. Click the **Advanced** tab, click the EUS_Sun entry under Mapping Templates, and then click the **Apply** to deploy the mapping.
   b. Access the LDAP Adapter for Enterprise User Security and click the **Plug-ins** tab.
   c. Select the ObjectclassMapper plug-in, click the **Create Namespace** button, enter `cn=OracleContext,<YOUR DOMAIN NAME>` in the Namespace field, and then click the **OK** button.
   d. Click the **Create Mapping** button, then select **EUS_Sun.py**, then enter a unique mapping name, then click the **Create Namespace** button, then enter the name of your domain in the Namespace field, and then click the **OK** button.
   e. Click the **Apply** button.
7. Configure the Access Control Lists (ACLs) for the integration. Refer to "Configuring Access Control Lists for the Enterprise User Security Integration" on page 19-21 for details about each ACL. After you configure the ACLs, continue the integration by proceeding to step 8.
8. Update the realm information with Root Oracle Context by performing the following steps:
   a. Edit the modifyRealm.ldif file to use your Sun Java System Directory Server domain name. If you use DN mappings between Oracle Virtual Directory and Sun Java System Directory Server, use the mapped DN in Oracle Virtual Directory.
   b. Update the realm information using the following command:

```
ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p port \
-D bindDN -q -v -f modifyRealm.ldif
```

Note: To update the Sun Java System Directory Server-Oracle Virtual Directory configuration, edit the modifyRealm.ldif file and execute ldapmodify with the updated modifyRealm.ldif file.

The steps to configure Oracle Virtual Directory for integration with Enterprise Security and use with Sun Java System Directory Server are complete. Continue the integration process and configure Enterprise User Security by referring to the *Oracle Database Enterprise User Administrator's Guide*.

19.2.2.4 User Identities in Novell eDirectory

Perform the following procedures to integrate Oracle Virtual Directory with Enterprise User Security for user identities stored in Novell eDirectory:

- Configuring Novell eDirectory for the Integration
- Configuring Oracle Virtual Directory for the Integration

19.2.2.4.1 Configuring Novell eDirectory for the Integration Perform the following steps to configure Novell eDirectory for the integration:

1. Extend the eDirectory LDAP attribute and objectclass using the following command:

```
ORACLE_HOME/bin/ldapmodify -h eDirectory_Host_Name -p eDirectory_Port \
-D bindDN -q -v -f eDirSchema.ldif
```

2. Modify X-NDS_CONTAINMENT in the groupofuniquenames objectclass by executing the following command:

```
ORACLE_HOME/bin/ldapmodify -h eDirectory_Host_Name -p eDirectory_Port \
-D bindDN -q -v -f eDirgoun.ldif
```

3. Create a realm in Novell eDirectory by performing the following steps:
   a. Open the eDirRealm.ldif file and replace all instances of the `dc=oracle,dc=com` string with the name of your domain.
   b. Run the following command to create a realm in eDirectory using the eDirRealm.ldif:

```
ORACLE_HOME/bin/ldapmodify -h eDirectory_Host_Name -p eDirectory_Port \
-D bindDN -q -v -f eDirRealm.ldif
```

4. Configure the user and group containers by performing the following steps:
   a. Open the eDirUserContainer.ldif file.
   b. Replace all instances of the `ou=users,dc=oracle,dc=com` string with the name of your user container.
   c. Replace all instances of the ou=groups,dc=oracle,dc=com string with the name of your group container.

   Note: Make sure the user and group containers are in the same domain and realm you are creating. For example, if your domain is `dc=superdemo,dc=net`, then `ou=people,dc=ultrademo,dc=org` is not a valid user container.

   d. Run the following command to configure the user and group containers:

```
ORACLE_HOME/bin/ldapmodify -h eDirectory_Host_Name -p eDirectory_Port \
-D bindDN -q -v -f eDirUserContainer.ldif
```

5. Enable Universal Password in eDirectory and allow the administrator to retrieve the user password. Refer to Novell's eDirectory documentation on Password Management for more information.

19.2.2.4.2 Configuring Oracle Virtual Directory for the Integration Perform the following steps to configure Oracle Virtual Directory for the integration:

1. Ensure you have performed all steps in "Preparing Oracle Virtual Directory for the Enterprise User Security Integration" on page 19-3 before proceeding with this procedure.
2. Download the NMAS toolkit from the Novell Developer Community website.
3. Copy the NMAStoolkit.jar file to the *ORACLE_HOME*/ovd/jlib/ directory.

   Restart the Oracle Virtual Directory server.

4. Start Oracle Directory Services Manager and connect to the Oracle Virtual Directory server.
5. Create three new Local Store Adapters using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.
   - Use the **Local_Storage_Adapter** template for each adapter.
   - The **Adapter Suffix** for one of the Local Store Adapters must be `cn=OracleContext`; the **Adapter Suffix** for another of the Local Store Adapters must be `cn=OracleSchemaVersion`; and the **Adapter Suffix** for the other Local Store Adapter must be `dc=com`, unless your eDirectory domain is something like `dc=example,dc=net`, in which case the **Adapter Suffix** must be `dc=net`.
   - The **Database File** and **Backup File** fields for each of the adapters must be unique.
6. Update and load the entries into the Local Store Adapters by performing the following steps:
   a. Extend the Oracle Virtual Directory schema with the loadOVD.ldif file using the following command. The loadOVD.ldif file contains entries for Oracle Context and schemaversion that Enterprise User Security queries. The loadOVD.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f loadOVD.ldif
      ```

   b. Update realmRoot.ldif to use your namespaces, including the dn, dc, o, orclsubscriberfullname, and memberurl attributes in the file. If you have a DN mapping between Novell eDirectory and Oracle Virtual Directory, use the DN that you see from Oracle Virtual Directory. The realmRoot.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      Note: The realmRoot.ldif file contains core entries in the directory namespace that Enterprise User Security queries. The realmRoot.ldif file also contains the dynamic group that contains the registered Enterprise User Security databases to allow secured access to sensitive Enterprise User Security related attributes, like the user's Enterprise User Security hashed password attribute.

   c. Load your domain root information in the realmRoot.ldif file into Oracle Virtual Directory using the following command:

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f realmRoot.ldif
      ```

7. Create an LDAP Adapter for Enterprise User Security using the following settings and by entering the Novell eDirectory host information, including the appropriate Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.
   - Use the **EUS_eDirectory** template for the adapter.
   - Enable the **Use SSL/TLS** option.
8. Configure the Enterprise User Security plug-ins by performing the following steps:

a. Click the **Advanced** tab, click the EUS_EDir entry under Mapping Templates, and then click the **Apply** to deploy the mapping.

b. Access the LDAP Adapter for Enterprise User Security and click the **Plug-ins** tab.

c. Select the ObjectclassMapper plug-in, click the **Create Namespace** button, enter `cn=OracleContext,<YOUR DOMAIN NAME>` in the Namespace field, and then click the **OK** button.

d. Click the **Create Mapping** button, then select **EUS_EDir.py**, then enter a unique mapping name, and then click the **OK** button.

e. Click the **Apply** button.

9. Configure the Access Control Lists (ACLs) for the integration. Refer to "Configuring Access Control Lists for the Enterprise User Security Integration" on page 19-21 for details about each ACL. After you configure the ACLs, continue the integration by proceeding to step 10.

10. Update the realm information with Root Oracle Context by performing the following steps:

a. Edit the modifyRealm.ldif file to use your Novell eDirectory domain name. If you use DN mappings between Oracle Virtual Directory and Novell eDirectory, use the mapped DN in Oracle Virtual Directory.

b. Update the realm information using the following command:

```
ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p port \
-D bindDN -q -v -f modifyRealm.ldif
```

The steps to configure Oracle Virtual Directory for integration with Enterprise Security and use with Novell eDirectory are complete. Continue the integration process and configure Enterprise User Security by referring to the *Oracle Database Enterprise User Administrator's Guide*.

19.2.2.5 User Identities in Oracle Internet Directory

Perform the following procedures to integrate Oracle Virtual Directory with Enterprise User Security for user identities stored in Oracle Internet Directory:

- Configuring Oracle Internet Directory for the Integration
- Configuring Oracle Virtual Directory for the Integration

19.2.2.5.1 Configuring Oracle Internet Directory for the Integration To configure Oracle Internet Directory for the integration, extend the Oracle Internet Directory LDAP attribute and objectclass using the following command:

```
ORACLE_HOME/bin/ldapmodify -h OID_Host_Name -p OID_Port -D bindDN -q -v \
-f ./OIDSchema.ldif
```

19.2.2.5.2 Configuring Oracle Virtual Directory for the Integration Perform the following steps to configure Oracle Virtual Directory for the integration:

1. Ensure you have performed all steps in "Preparing Oracle Virtual Directory for the Enterprise User Security Integration" on page 19-3 before proceeding with this procedure.

2. Start the Oracle Virtual Directory server, then start Oracle Directory Services Manager, and then connect to the Oracle Virtual Directory server.

3. Create three new Local Store Adapters using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.
   - Use the **Local_Storage_Adapter** template for each adapter.
   - The **Adapter Suffix** for one of the Local Store Adapters must be `cn=OracleContext`; the **Adapter Suffix** for another of the Local Store Adapters must be `cn=OracleSchemaVersion`; and the **Adapter Suffix** for the other the Local Store Adapters must be `dc=com`, unless your Oracle Internet Directory domain is something like `dc=example,dc=net`, in which case the **Adapter Suffix** must be `dc=net`.
   - The **Database File** and **Backup File** fields for each of the adapters must be unique.
4. Update and load the entries into the Local Store Adapters by performing the following steps:
   a. Extend the Oracle Virtual Directory schema with the loadOVD.ldif file using the following command. The loadOVD.ldif file contains entries for Oracle Context and schemaversion that Enterprise User Security queries. The loadOVD.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f loadOVD.ldif
      ```

   b. Update realmRoot.ldif to use your namespaces, including the dn, dc, o, orclsubscriberfullname, and memberurl attributes in the file. If you have a DN mapping between Oracle Internet Directory and Oracle Virtual Directory, use the DN that you see from Oracle Virtual Directory. The realmRoot.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

      Note: The realmRoot.ldif file contains core entries in the directory namespace that Enterprise User Security queries. The realmRoot.ldif file also contains the dynamic group that contains the registered Enterprise User Security databases to allow secured access to sensitive Enterprise User Security related attributes, like the user's Enterprise User Security hashed password attribute.

   c. Load your domain root information in the realmRoot.ldif file into Oracle Virtual Directory using the following command:

      ```
      ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
      -D bindDN -q -v -a -f realmRoot.ldif
      ```

5. Create an LDAP Adapter for Enterprise User Security using the **EUS_OID** adapter template and by entering the Oracle Internet Directory host information, including the appropriate Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.
6. Configure the Access Control Lists (ACLs) for the integration. Refer to "Configuring Access Control Lists for the Enterprise User Security Integration" on page 19-21 for details about each ACL. After you configure the ACLs, continue the integration by proceeding to step 7.
7. Update the realm information with Root Oracle Context by performing the following steps:

a. Edit the modifyRealm.ldif file to use your Oracle Internet Directory domain name. If you use DN mappings between Oracle Virtual Directory and Oracle Internet Directory, use the mapped DN in Oracle Virtual Directory.

b. Update the realm information using the following command:

```
ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p port -D \
bindDN -q -v -f modifyRealm.ldif
```

Note: To update the Oracle Internet Directory-Oracle Virtual Directory configuration, edit the modifyRealm.ldif file and execute ldapmodify with the updated modifyRealm.ldif file.

The steps to configure Oracle Virtual Directory for integration with Enterprise Security and use with Oracle Internet Directory are complete. Continue the integration process and configure Enterprise User Security by referring to the *Oracle Database Enterprise User Administrator's Guide*.

19.2.3 Configuring Access Control Lists for the Enterprise User Security Integration

This section describes the Access Control Lists (ACLs) you must configure in Oracle Virtual Directory for the Enterprise User Security integration regardless of which external repository you are using to store user identities in. If you have customized your ACLs after installing Oracle Virtual Directory, you must adjust the following ACL settings to include your customizations.

Perform the following steps to configure Oracle Virtual Directory ACLs for the Enterprise User Security integration:

1. Create the following ACLs. Refer to "Creating Access Control Lists Using Oracle Directory Services Manager" on page 16-1 for information on creating ACLs:

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN

Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`dc=com`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`dc=com`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`dc=com`
Scope	subtree
Applies To	authpassword
Deny	All operations
Access	Public

Note: The following ACL must be the last ACL in the ACL list for dc=com.

Target DN	`dc=com`
Scope	subtree
Applies To	authpassword
Grant	Search and Read
Access	Group with DN of: `cn=EUSDBGroup,dc=dbdemo,dc=orion,dc=com`. **Note**: Replace `dc=dbdemo,dc=orion,dc=com` with the DN of your namespace

2. Set the ACLs in Oracle Virtual Directory to support the OracleContextAdmins administrative group as follows:

Target DN	`cn=OracleContext,<YOUR DOMAIN>`
Scope	subtree
Applies To	Entry
Grant	All
Access	Group with DN of:
	`cn=OracleContextAdmins,cn=Groups,cn=OracleContext,<YOUR DOMAIN>`

Target DN	`cn=OracleContext,<YOUR DOMAIN>`
Scope	subtree
Applies To	All Attributes
Grant	All
Access	Group with DN of:
	`cn=OracleContextAdmins,cn=Groups,cn=OracleContext,<YOUR DOMAIN>`

3. Set the ACLs in the external directory to protect the data under `cn=OracleContext,<YOUR DOMAIN>`.
4. Give write permission to the `cn=OracleContextAdmins,cn=Groups,cn=OracleContext,<YOUR DOMAIN>` group.

19.2.4 Configuring Oracle Virtual Directory to Support Multiple Enterprise User Security Domains

Perform the following steps to configure Oracle Virtual Directory to allow Enterprise User Security users contained in multiple domains to authenticate to a database:

1. Configure the first domain using the instructions in Integrating Oracle Virtual Directory with External Directories on page 19-4.

> **Note:** If you are not using Enterprise Roles, you may use any directory server for the first domain. However, if you plan to use Enterprise Roles, Oracle recommends using Oracle Internet Directory as the first domain. Microsoft Active Directory and Novell eDirectory have DN syntax validation and if the second domain's DN does not exist in the first domain, you cannot complete this configuration.

2. If it does not already exist, add the realm root for the second domain by performing the following steps:
   a. Create an LDIF file using the following example. Replace the *VARIABLES* with the appropriate DN and orclsubscriberfullname of the second domain:

```
dn: DN_OF_SECOND_DOMAIN
dc: DC_OF_SECOND_DOMAIN
o: O_OF_SECOND_DOMAIN
objectclass: domain
```

```
objectclass: organization
objectclass: orclSubscriber
orclsubscriberfullname: ORCLSUBSCRIBERFULLNAME_OF_SECOND_DOMAIN
orclVersion: 90400
```

b. Update Oracle Virtual Directory with the new LDIF file. For example:

```
ORACLE_HOME/bin/ldapadd -h Oracle_Virtual_Directory_Host -p Port \
-D bindDN -q -v -f LDIF_File
```

3. Create a new LDAP Adapter for the second domain using the **EUS_*Directory-Type*** adapter template that is specific to the directory type. Enter the host name, port number, proxy DN, and password of the second domain. Be sure to configure the Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.
4. Configure the Mappings for the second domain LDAP Adapter by clicking the **Create Mapping** button on the Plug-Ins tab for the adapter. The Mapping you use depends on the type of directory you are using.

> **Note:** Do not configure a Mapping if you are using Oracle Internet Directory.

- Use EUS_EDir.py for Novell eDirectory
- Use EUS_ActiveDirectory.py for Active Directory
- Use EUS_Sun.py for Sun Java System Directory Server

5. Update Access Control Lists to protect the user entry and to allow the database account to access the password. You may skip this step if the Access Control Lists that were configured for the first domain cover the second domain's mapped namespace.

a. Create the following ACLs. Refer to "Creating Access Control Lists Using Oracle Directory Services Manager" on page 16-1 for information on creating ACLs:

Target DN	*Mapped Namespace for second domain*
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	*Mapped Namespace for second domain*
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	*Mapped Namespace for second domain*

Scope	subtree
Applies To	authpassword
Deny	All operations
Access	Public

Note: The following ACL must be the last ACL in the ACL list for the *Mapped Namespace for second domain*.

Target DN	*Mapped Namespace for second domain*
Scope	subtree
Applies To	authpassword
Grant	Search and Read
Access	Group with DN of: `cn=EUSDBGroup,dc=dbdemo,dc=orion,dc=com`. **Note**: Replace `dc=dbdemo,dc=orion,dc=com` with the DN of your first domain.

6. Update the Oracle Context with the newly added namespace by performing the following steps:

   a. Create an LDIF file like the following example and replace dc=dbdemo,dc=orion,dc=com with the DN of your first domain:

   ```
   dn: cn=Common,cn=Products,cn=OracleContext,dc=dbdemo,dc=orion,dc=com
   changetype: modify
   add: orclcommonusersearchbase
   orclcommonusersearchbase: Mapped_Namespace_for_Second_Domain
   ```

   b. Update Oracle Virtual Directory using the LDIF file. For example:

   ```
   ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p Port \
   -D bindDN -q -v -f LDIF_File
   ```

7. Repeat steps 2-6 to support additional domains.

19.2.5 Enabling User Account Lockout

LDAP servers can lock a user account after several bind attempts fail. The Oracle Virtual Directory-Enterprise User Security integration can utilize this lockout feature and enforce the back-end LDAP server's password lockout policy as follows:

- An incorrect login to the Oracle Database records a login failure to the back-end LDAP server
- A correct login to the Oracle Database resets the login failure count in the back-end LDAP server

Note: This functionality is not available for integrations that use Active Directory.

- A locked user account cannot be used to log in to the Oracle Database

After performing the Oracle Virtual Directory-Enterprise User Security integration, you can enable user account lockout by performing the following steps:

> **Note:** If you are using Oracle Internet Directory as the back-end LDAP server, skip steps 1 and 2 in the following procedure.

1. Create and configure the euslockout plug-in for the Enterprise User Security integration LDAP Adapter by referring to Managing Adapter Plug-ins on page 13-1. When you configure the euslockout plug-in, you must:
   - Create a **directoryType** parameter with a value according to your back-end LDAP server, such as **ActiveDirectory** for Active Directory, **iPlanet** for Sun Java System Directory Server, or **eDirectory** for Novell eDirectory.
   - Create a namespace using the name of your user container.
2. If you are using Active Directory or Sun Java System Directory Server as a back-end LDAP server, you must configure an additional plug-in parameter on the Enterprise User Security integration LDAP Adapter. If you are using Novell eDirectory as a back-end LDAP server, go to step 3.

   For Active Directory

   a. Query the Active Directory domain to determine the `ADLockoutDuration` value for that specific domain. For example:

   ```
   ORACLE_HOME/bin/ldapsearch -h Active_Directory_Host_Name -D bindDN \
   -q -s base -b Active_Directory_Domain objectclass="*" lockoutDuration
   ```

   b. Set the **ADLockoutDuration** parameter in the EUSActiveDirectory plug-in using the value returned from the query. Click the **EUSActiveDirectory** plug-in, then click the **Create New Parameter** button, then select **ADLockoutDuration** and enter the lockout duration value in the Parameter field, and click the **OK** button. Be sure you use the exact same value returned from Active Directory. For example, if the value returned from Active Directory is lockoutDuration=-18000000000, enter `-18000000000` in the Parameter field regardless of the value being negative.

   For Sun Java System Directory Server

   a. Query the Sun Java System Directory Server to determine its `passwordMaxFailure` value. For example:

   ```
   ORACLE_HOME/bin/ldapsearch -h Sun_Java_System_Directory_Server_Name \
   -D bindDN -q -s base -b "cn=password policy,cn=config" objectclass="*"
   passwordmaxfailure
   ```

   b. Set the **passwordMaxFailure** parameter in the EUSiPlanet plug-in using the value returned from the query. Click on the EUSiPlanet plug-in, then click the **Create New Parameter** button. Select **passwordMaxFailure** and enter the value in the Parameter field. Click **OK**.
3. Create the following Access Control Lists. Refer to "Creating Access Control Lists Using Oracle Directory Services Manager" on page 16-1 for information on creating ACLs:

Target DN	*Your_User_Container*
Scope	subtree

Applies To	orclaccountstatusevent
Deny	All operations
Access	Public

Target DN	*Your_User_Container*
Scope	subtree
Applies To	orclaccountstatusevent
Grant	Write
Access	Group with DN of: `cn=EUSDBGroup,dc=dbdemo,dc=orion,dc=com`. **Note**: Replace `dc=dbdemo,dc=orion,dc=com` with the DN of your namespace.

4. For Oracle Internet Directory, Sun Directory Server, and Novell eDirectory, ensure the proxy user configured for the Enterprise User Security LDAP Adapter has permission to modify the account lockout related attributes.

> **Note:** Resetting the account lockout counter after a correct login is not available for Oracle Virtual Directory-Enterprise User Security integrations with Active Directory. Alternatively, Active Directory has the ability to reset the account lockout counter after a specified period of time has elapsed. This option can be utilized to prevent the lockout counter from accumulating indefinitely.

19.2.6 Integration Limitations

The following is a list of Oracle Virtual Directory-Enterprise User Security integration known limitations:

- The following functionality is not supported in the integration:
  - DN mapping between Microsoft Active Directory and Oracle Virtual Directory if the Active Directory domain containing the domain DN is mapped to Oracle Virtual Directory. For example, if the Active Directory DN is `dc=us,dc=oracle,dc=com` and you try to map it to `dc=oracle,dc=com` in Oracle Virtual Directory, this type of DN mapping is not supported.
  - Administrative Groups except for OracleContextAdmins
  - Enterprise Security Manager console to Oracle Internet Directory Delegated Administration Services
  - Password Policy
  - Client certificate authentication
  - Kerberos authentication when integrating for use with Sun Java System Directory Server and Oracle Internet Directory
  - User Migration Utility (UMU)
  - Multiple Domain environments
  - JDBC Thin Driver—you must use the OCI driver

  - Combined Microsoft Active Directory and Sun Java System Directory Server environments
- In the Enterprise Security Manager interface:
  - Listed databases may sometimes include an Active Directory tombstone entry.
  - Database and Oracle Internet Directory version information is not available.

19.3 Integrating with Oracle's Net Services

This topic describes how to integrate Oracle Virtual Directory with Oracle Database Net Services to centralize name services with Oracle Internet Directory, Microsoft Active Directory, and Sun Java System Directory Server. This topic contains the following sections:

19.3.1 Overview

Oracle Virtual Directory has the ability to integrate with Oracle's Net Services database product. Integrating Oracle Virtual Directory and Net Services enhances and simplifies your name service capabilities by allowing you to leverage service entries stored in an external LDAP repository without any additional synchronization.

19.3.2 Starting the Integration

This section lists the common steps required for all Oracle Virtual Directory-Net Services integrations. Perform the steps in this section first to start the integration, then proceed to one of the subsequent sections specific to Oracle Internet Directory, Microsoft Active Directory, and Sun Java System Directory Server. Different steps are presented depending on whether you are integrating Oracle Virtual Directory with Net Services for use with Oracle Internet Directory, Microsoft Active Directory, or Sun Java System Directory Server. Only perform the steps appropriate for your environment.

Perform the following steps to start the Oracle Virtual Directory-Net Services integration process:

1. Create a backup copy of the *ORACLE_HOME*/ovd/eus/ directory.
2. Create the subschemasubentry plug-in as global server plug-in. Refer to "Managing Global Server Plug-ins" on page 13-4 for steps on creating server plug-ins.

19.3.3 Integrating for Use with Microsoft Active Directory

Perform the following steps to integrate Oracle Virtual Directory with Net Services for use with Microsoft Active Directory. Perform these only after you have completed the steps in the "Starting the Integration" section. The procedure for integrating Oracle Virtual Directory with Net Services for use with Microsoft Active Directory includes the following tasks:

- Configuring Active Directory for the Integration
- Configuring Oracle Virtual Directory for the Integration

19.3.3.1 Configuring Active Directory for the Integration

Perform the following steps to configure Active Directory for the integration:

1. Make a backup copy of your Active Directory image. The schema extensions inside of Active Directory are permanent and cannot be cancelled. The backup image allows you to restore all your changes if required.
2. Load the Net Services required schema into Active Directory using the Java classes included in Oracle Virtual Directory by executing the following command. You can use the java executable in the *ORACLE_HOME*/jdk/bin directory.

```
java extendAD -h Active_Directory_Host_Name -p Active_Directory_Port
-D Active_Directory_Admin_DN -w Active_Directory_Admin_Password
-AD Active_Directory_Domain_DN
```

> **Note:** An example of a valid Active Directory domain DN is: `dc=oracle,dc=com`

19.3.3.2 Configuring Oracle Virtual Directory for the Integration

Perform the following steps to configure Oracle Virtual Directory for the integration:

1. Start the Oracle Virtual Directory server, then start Oracle Directory Services Manager, and then connect to the Oracle Virtual Directory server.
2. Create two new Local Store Adapters using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.
   - Use the **Local_Storage_Adapter** template for each adapter.
   - The **Adapter Suffix** for one of the Local Store Adapters must be `cn=OracleContext` and the **Adapter Suffix** for the other of the Local Store Adapters must be `cn=OracleSchemaVersion`.
   - The **Database File** and **Backup File** fields for each of the adapters must be unique.
3. Update and load the entries into the Local Store Adapters by extending the Oracle Virtual Directory schema with the loadOVD.ldif file using the following command. The loadOVD.ldif file contains entries for Oracle Context and schemaversion that Net Services queries. The loadOVD.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

```
ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
-D bindDN -q -v -a -f loadOVD.ldif
```

4. Create an LDAP Adapter for Net Services using the following settings and by entering the Active Directory host information, including host name, non-SSL port number, proxy DN and password, and the appropriate Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.
   - Use the **ONames_ActiveDirectory** adapter template.
   - Select the **BindOnly** Pass Through Credential option.

5. Update the Access Control Lists by performing the following steps. If you have customized your ACLs after installing Oracle Virtual Directory, you must adjust the following ACL settings to include your customizations.

   a. Create the following ACLs. Refer to "Creating Access Control Lists Using Oracle Directory Services Manager" on page 16-1 for information on creating ACLs:

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	Your Mapped Namespace in Oracle Virtual Directory, for example: `dc=example,dc=com`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	Your Mapped Namespace in Oracle Virtual Directory, for example: `dc=example,dc=com`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

b. Set the ACLs in Oracle Virtual Directory to support the OracleNetAdmins administrative group as follows:

Target DN	`cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`
Scope	subtree
Applies To	Entry
Grant	All
Access	Group with DN of: `cn=OracleNetAdmins,cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`

Target DN	`cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`
Scope	subtree
Applies To	All Attributes
Grant	All
Access	Group with DN of: `cn=OracleNetAdmins,cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`

6. Create an LDAP Adapter for the OracleNetAdmins administrative group using the following settings and by entering the Active Directory host information, including port number, proxy DN, and password. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.

- Use the **Active_Directory** adapter template.
- Enter `cn=OracleNetAdmins,cn=users, <YOUR Active_Directory_Domain_DN>` as the Remote Base.
- Enter `cn=OracleNetAdmins,cn=OracleContext,<YOUR MAPPED DOMAIN DN in Oracle Virtual Directory>` as the Mapped Namespace.

7. Configure a mapping and plug-in for the OracleNetAdmins administrative group adapter by performing the following steps:

a. Click the **Advanced** tab, then click **Active_Directory_to_inetOrg**, and then click the **Apply** button to deploy the mapping.

b. Click the **Adapter** tab, then click the adapter for the OracleNetAdmins administrative group, then click the **Plug-ins** tab, then click the **Create Mapping** button, then select **Active_Directory_to_inetOrg.py**, then enter a unique mapping name, and then click **OK**.

c. Click the **Create Plug-in** button, then click the **Select** button, then select the **EUSMemberDNMapping** plug-in, then click **OK**, then enter a unique plug-in name, then create the `localDomainDN` and `remoteDomainDN` parameters, and then click **OK**. Note that the `localDomainDN` and `remoteDomainDN` may be different if you have DN mapping configured.

d. Click the **Apply** button.

Note: You may not see the group membership changes immediately after your changes in Active Directory. This is because of Active Directory's group membership refresh interval configuration.

The steps to configure Oracle Virtual Directory for integration with Net Services and for use with Microsoft Active Directory are complete. Continue the integration process and configure Oracle Net Services by referring to the *Oracle Database Net Services Administrator's Guide*.

19.3.4 Integrating for Use with Sun Java System Directory Server

Perform the following steps to integrate Oracle Virtual Directory with Net Services for use with Sun Java System Directory Server. Perform these only after you have completed the steps in the "Starting the Integration" section. The procedure for integrating Oracle Virtual Directory with Net Services for use with Sun Java System Directory Server includes the following tasks:

- Configuring Sun Java System Directory Server for the Integration
- Configuring Oracle Virtual Directory for the Integration

19.3.4.1 Configuring Sun Java System Directory Server for the Integration

Perform the following steps to configure Sun Java System Directory Server for the integration:

1. Extend the iPlanet LDAP attribute and objectclass using the following command:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./iPlanetSchema.ldif
```

2. Create a realm in iPlanet by performing the following steps:
   a. Open the realmiPlanet.ldif file and replace all instances of the `dc=us,dc=oracle,dc=com` string with the name of your domain.
   b. Run the following command to create a realm in iPlanet using the realmiPlanet.ldif file:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./realmiPlanet.ldif
```

3. Configure the user and group containers by either creating new user and group containers, or by using existing user and group containers.

   Creating New User and Group Containers

   a. Open the iPlanetContainers.ldif file and replace all instances of the `dc=us,dc=oracle,dc=com` string with the name of your domain.
   b. Run the following command to create user and group containers in iPlanet using the iPlanetContainers.ldif file:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./iPlanetContainers.ldif
```

   Using Existing User and Group Containers

   a. Open the useiPlanetContainers.ldif file.

b. Replace all instances of the `cn=users,dc=us,dc=oracle,dc=com` string with the name of your user container.

c. Replace all instances of the `cn=groups,dc=us,dc=oracle,dc=com` string with the name of your group container.

Note: Make sure the user and group containers are in the same domain and realm you are creating. For example, if your domain is `dc=superdemo,dc=net`, then `ou=people,dc=ultrademo,dc=org` is **not** a valid user container.

---

d. Run the following command to create a realm in iPlanet using the useiPlanetContainers.ldif file:

```
ORACLE_HOME/bin/ldapmodify -h iPlanet_Host_Name -p iPlanet_Port \
-D cn="directory manager" -q -v -a -f ./useiPlanetContainers.ldif
```

19.3.4.2 Configuring Oracle Virtual Directory for the Integration

Perform the following steps to configure Oracle Virtual Directory for the integration:

1. Start the Oracle Virtual Directory server, then start Oracle Directory Services Manager, and then connect to the Oracle Virtual Directory server.
2. Create two new Local Store Adapters using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.
   - Use the **Local_Storage_Adapter** template for each adapter.
   - The **Adapter Suffix** for one of the Local Store Adapters must be `cn=OracleContext` and the **Adapter Suffix** for the other of the Local Store Adapters must be `cn=OracleSchemaVersion`.
   - The **Database File** and **Backup File** fields for each of the adapters must be unique.
3. Update and load the entries into the Local Store Adapters by extending the Oracle Virtual Directory schema with the loadOVD.ldif file using the following command. The loadOVD.ldif file contains entries for Oracle Context and schemaversion that Net Services queries. The loadOVD.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

   ```
   ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
   -D bindDN -q -v -a -f loadOVD.ldif
   ```

4. Create an LDAP Adapter for Net Services using the following settings and by entering the Sun Java System Directory Server host information, including host name, non-SSL port number, proxy DN and password, and the appropriate Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.
   - Use the **ONames_Sun** adapter template.
   - Select the **BindOnly** Pass Through Credential option.
5. Update the Access Control Lists by performing the following steps. If you have customized your ACLs after installing Oracle Virtual Directory, you must adjust the following ACL settings to include your customizations.

a. Create the following ACLs. Refer to "Creating Access Control Lists Using Oracle Directory Services Manager" on page 16-1 for information on creating ACLs:

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	Your Mapped Namespace in Oracle Virtual Directory, for example: `dc=example,dc=com`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	Your Mapped Namespace in Oracle Virtual Directory, for example: `dc=example,dc=com`
Scope	subtree
Applies To	All Attributes

Grant	Search and Read
Access	Public

b. Set the ACLs in Oracle Virtual Directory to support the OracleNetAdmins administrative group as follows:

Target DN	`cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`
Scope	subtree
Applies To	Entry
Grant	All
Access	Group with DN of: `cn=OracleNetAdmins,cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`

Target DN	`cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`
Scope	subtree
Applies To	All Attributes
Grant	All
Access	Group with DN of: `cn=OracleNetAdmins,cn=OracleContext,<YOUR MAPPED ORACLE VIRTUAL DIRECTORY NAMESPACE>`

c. Set the ACLs in the external directory to protect the data under `cn=OracleContext,<YOUR DOMAIN>`.

The steps to configure Oracle Virtual Directory for integration with Net Services and for use with Sun Java System Directory Server are complete. Continue the integration process and configure Oracle Net Services by referring to the *Oracle Database Net Services Administrator's Guide*.

19.3.5 Integrating for Use with Oracle Internet Directory

Perform the following steps to integrate Oracle Virtual Directory with Net Services for use with Oracle Internet Directory. Perform these only after you have completed the steps in the "Starting the Integration" section.

1. Start the Oracle Virtual Directory server, then start Oracle Directory Services Manager, and then connect to the Oracle Virtual Directory server.
2. Create two new Local Store Adapters using the following settings. Refer to "Creating Local Store Adapters" on page 12-21 for information on creating Local Store Adapters.
   - Use the **Local_Storage_Adapter** template for each adapter.
   - The **Adapter Suffix** for one of the Local Store Adapters must be `cn=OracleContext` and the **Adapter Suffix** for the other of the Local Store Adapters must be `cn=OracleSchemaVersion`.

- The **Database File** and **Backup File** fields for each of the adapters must be unique.

3. Update and load the entries into the Local Store Adapters by extending the Oracle Virtual Directory schema with the loadOVD.ldif file using the following command. The loadOVD.ldif file contains entries for Oracle Context and schemaversion that Net Services queries. The loadOVD.ldif file is located in the *ORACLE_HOME*/ovd/eus/ directory.

```
ORACLE_HOME/bin/ldapmodify -h Oracle_Virtual_Directory_Host -p OVD_Port \
-D bindDN -q -v -a -f loadOVD.ldif
```

4. Create an LDAP Adapter for Net Services using the **ONames_OID** adapter template and by entering the Oracle Internet Directory host information, including host name, non-SSL port number, proxy DN and password, and the appropriate Remote Base and Mapped Namespace. Refer to "Creating LDAP Adapters" on page 12-1 for information on creating LDAP Adapters.

5. Update the Access Control Lists by performing the following steps. If you have customized your ACLs after installing Oracle Virtual Directory, you must adjust the following ACL settings to include your customizations.

   a. Create the following ACLs. Refer to "Creating Access Control Lists Using Oracle Directory Services Manager" on page 16-1 for information on creating ACLs:

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleContext`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	`cn=OracleSchemaVersion`
Scope	subtree
Applies To	All Attributes

Grant	Search and Read
Access	Public

Target DN	Your Mapped Namespace in Oracle Virtual Directory, for example: `dc=example,dc=com`
Scope	subtree
Applies To	Entry
Grant	Browse DN and Return DN
Access	Public

Target DN	Your Mapped Namespace in Oracle Virtual Directory, for example: `dc=example,dc=com`
Scope	subtree
Applies To	All Attributes
Grant	Search and Read
Access	Public

The steps to configure Oracle Virtual Directory for integration with Net Services and for use with Oracle Internet Directory are complete. Continue the integration process and configure Oracle Net Services by referring to the *Oracle Database Net Services Administrator's Guide*.

20

Oracle Communications Universal User Profile

This chapter describes Oracle Communications Universal User Profile and contains the following topics:

> **Note:** Throughout this chapter, the term "Oracle Communications Universal User Profile server" refer to an Oracle Virtual Directory server configured with a Diameter Adapter.

20.1 What is Oracle Communications Universal User Profile?

Oracle Communications Universal User Profile is a stateless subscriber identity service that aggregates and normalizes subscriber data from the various systems that may exist in a telecommunications organization. As telecommunications organizations have grown from land-line to ISP to mobile systems, the number of repositories have also grown exponentially. To take advantage of the new business opportunities enabled by these new services—as well as to reduce the cost of delivering service to customers—a single view of subscriber identity must be provided to applications.

To provide a single view of subscriber identity to applications, several challenges must be addressed, including:

- Due to mergers, acquisitions and subsidiaries, subscriber data may be distributed across multiple systems with different keys.
- Internal organizational data-politics, external regulations, and because the data likely will be out of date if it must be synchronized, it is impossible to merge all data into a single universal repository.
- Applications require data in application specific views.
- Multiple aspects related to protecting identity data.
- The process of duplicating all the storage, security, and back-ups of the existing data while copying it to a different location is an enormous task—one that typically is not a viable option for most organizations.

Oracle Communications Universal User Profile solves the challenges described in the preceding list by providing a stateless, carrier-grade solution that delivers a standard, single point of contact for subscriber identity information. Instead of copying data into a centralized repository, Oracle Communications Universal User Profile retrieves the required data on demand and presents it in a format required by the application.

Additionally, because Oracle Communications Universal User Profile is stateless and integrates with key related technologies such as Oracle's TimesTen In-Memory Database, Service Delivery Platform, and SOA Suite, it can be used to enable "smart-routing" to meet the demands of a telecommunications solution.

The standard interfaces to Oracle Communications Universal User Profile include LDAP and SOAP, and by default, it can access data in LDAP, Relational Database, or Diameter HSS repositories. Oracle Communications Universal User Profile also provides a custom Java API to leverage data in XCAP, Web Services, and any other system Java can connect to.

20.2 Example Oracle Communications Universal User Profile Use Cases and Deployment Scenarios

This topic describes example use cases and deployment scenarios for Oracle Communications Universal User Profile, including:

- Enabling Double or Triple Play Services
- Improving Organic Social Networking
- Mid-Call Move
- Smart Package Routing

Enabling Double or Triple Play Services

Many telecommunications companies provide multiple types of service, including mobile/land-line, ISP, and television (cable or fiber) services. As a result, many of these telecommunications companies attempt to "cross-sell" their multiple services to customers that do not currently subscribe to all of them.

In this type off situation, Oracle Communications Universal User Profile can be used to aggregate data from various systems to help build both a marketing campaign and a customer portal to improve the user experience when dealing with the telecommunications company.

For example, to enable the marketing campaign, Oracle Communications Universal User Profile can be used with a BPEL work-flow to:

1. Find all mobile subscribers that are not television subscribers
2. Send them a Short Message Service (SMS) message that provides a coupon for a reduced price if they enroll in a television service within a certain time period, say, with in the next four hours

To accomplish this, Oracle Communications Universal User Profile pulls data from:

- Both mobile and television billing systems to determine which customers have which service
- The contact database to determine customer phone numbers and phone capabilities
- The HSS information repository to determine if customers are on-line and capable of meeting the four hour deadline

Using BPEL allows the workflow to be segregated for performance, reliability, and also so that the marketing group can not see customer names and contact information, thereby protecting subscriber's identity information.

Improving Organic Social Networking

While the popularity of social networks is rapidly growing online, social networks have always naturally occurred in our physical, non-electronic communities. In reality however, it is often easier to communicate with the thousands of people in your electronic social network that you have never met face-to-face, than it is to communicate with 10-15 people that you informally associate with by circumstance, say, the parents of the children on a youth athletic team.

Imagine you coach a youth athletic team and one day during practice a rain storm begins unexpectedly. If you wanted to cancel the practice because of the rain it can be very hard, if not impossible for you to contact all the parents of the children in a timely fashion.

However, using Oracle's Web Center, Service Delivery Platform, and Oracle Communications Universal User Profile, telecommunication companies can build a mobile Web application that can address a situation like this and contact the parents of the all children quickly. If you decide to cancel the practice, you could leave a voice or text message on a custom, billable service which would then relay the message to the parents of the children using the team's address book.

In this situation, Oracle Communications Universal User Profile provides:

- access to the coach's credentials to authenticate to the application
- the attribute that authorizes the call
- the team address book, which could have information such as the primary contact number and whether the phone can receive Short Message Service (SMS) or voice only messages

Mid-Call Move

The concept of "Mid-Call Move" refers to a service that allows a person on the phone to have the current call re-directed to different phone. For example, imagine you are participating in a conference call at your desk at work, but you need to leave the office. A Mid-Call Move would allow you to press a certain phone number combination on the desk phone, for example, star then 7, and transfer the conference call to your mobile phone. In this situation, Oracle Communications Universal User Profile can be used to determine whether you have enrolled in the Mid-Call Move service, and if so, to then look up your mobile phone number before initiating the transfer.

Smart Package Routing

Imagine while at work, you ordered 1,000 parts and had them sent to Facility A in Atlanta, Georgia. Later you realize you made a mistake and the 1,000 parts are actually needed at Facility B in Boston, Massachusetts. Unfortunately, you do not know the address at Facility B in Boston and you do not know who to notify at either Facility A or B to communicate the situation.

To address a situation like this, a shipping service could be integrated with your office phone service so that a mobile web application could be used to track the package, select a different route, and also notify the correct people—all by selecting the proper office name. In this scenario, Oracle Communications Universal User Profile provides cost-center information for the billing of the package, authorization allowing you to change the package routing, and addresses of the various facilities.

20.3 Oracle Communications Universal User Profile Diameter Adapters

In today's telecommunications environments, subscriber information may be contained in a IMS-compliant Home Subscriber Service (HSS). One of the features of Oracle Communications Universal User Profile is the Diameter adapter. The Diameter adapter enables Oracle Communications Universal User Profile to:

- Connect to 3GPP IP Multimedia Subsystem (IMS) Home Subscriber Systems that run over the Diameter protocol
- Present a standardized service layer for subscriber identity information regardless of the information's source

---

Note: The Client View browser is unable to browse the namespace of Diameter adapter because it does not support subtree search scope with a `objectclass=*` filter. Similarly, the Client View browser is unable to browse a Join View adapter configured with a Diameter adapter as its primary adapter.

This topic contains the following sections:

20.3.1 Enabling Support for Diameter Adapters

Perform the following steps to enable Oracle Virtual Directory to support Diameter Adapters:

1. Use WLST to add `schema.diameter.xml` to the Oracle Virtual Directory server settings by referring to the "SchemaLocations" entry in "Configuring Oracle Virtual Directory Server Settings Using WLST" on page 9-7.
2. In the *ORACLE_INSTANCE*/config/OPMN/opmn/opmn.xml file, add the following, as one continuos string, to the end of the `java-classpath` property for the Oracle Virtual Directory server entry:

   ```
   :$ORACLE_HOME/ovd/diameter/lib/wlssdiameter.jar:$ORACLE_
   HOME/ovd/diameter/lib/weblogic.jar:$ORACLE_HOME/ovd/diameter/lib/xbean.jar
   ```

3. Reload the OPMN configuration by executing the following command:

   ```
   $ORACLE_INSTANCE/bin/opmnctl reload
   ```

4. Restart the Oracle Virtual Directory server.

20.3.2 Creating and Configuring Diameter Adapters

Perform the following steps to create and configure a Diameter Adapter:

1. Create a new Custom Adapter. Refer to "Creating and Configuring Custom Adapters" on page 18-2 for more information.
2. Select Diameter as Adapter Template in the New Custom Adapter dialog box and enter the appropriate Adapter Suffix/Namespace. Click the **Finish** button.
3. Go to the Plug-ins tab for the new Diameter adapter. Select DiameterAdapterPlugin and click the **Edit** button. The Edit Plug-in: DiameterAdapterPlugin dialog box appears.

4. Add the following configuration parameters (**Name**: Value) to the Parameters table by selecting a parameter and clicking the **Edit** button. The **remoterealm** and **remotePeers** parameters are required.

   - **interface**: The type of Diameter interface used to connect to HSS. The default value is `Sh`.

     Note: Currently, `Sh` is the only supported value for the **interface** configuration parameter.

   - **listener**: The hostname of the Oracle Communications Universal User Profile server and the port to listen on to receive messages from the HSS server, which is a requirement of the HSS protocol. The listener value must be in the form of `aaa://host:port`, for example `aaa://ovd.oracle.com:3689`. The port must be accessible through the firewall from the HSS server. 3869 is the default port for HSS client listener ports.

     Notes:

     - The `aaa` string in the listener value is not a replaceable variable—it must be explicitly included.
     - Do not confuse the listener parameter for an Oracle Virtual Directory listener.

   - **nosctp**: Determines whether or not to disable SCTP transport. Supported values are `true` and `false`. The default setting is `true`.

     Note: Refer to "Enabling SCTP Transport" for more information about using the **nosctp** option and enabling SCTP Transport.

   - **localrealm**: The realm of this custom Diameter adapter and can be any valid value for the HSS server. The default is us.oracle.com. The localrealm parameter is used as a key to map records for Oracle Communications Universal User Profile to the HSS server.
   - **remoterealm** (required): Supplied by the HSS provider, it is the remoterealm value for the HSS server.
   - **remotePeers** (required): The remotePeers value is the hostname and port to the HSS server. Both values are provided by the HSS server. The remotePeers value must be in the format of `aaa://host:port`, for example, `aaa://hss.oracle.com:3868`. The default port is 3868.

     Note: Multiple values can be specified by separating each with the semi-colon character (`;`). For example, `aaa://127.0.0.1:3868; aaa://different-host:3939`.

   - **remoteHost**: Hostname of the remote HSS server. The default value is 127.0.0.1.
   - **connectionWaitTimeout**: The amount of time (in milliseconds) for the HSS service to be available while the adapter starts. The default value is 30000.

- **requestTimeout**: The maximum waiting time (in milliseconds) for pending HSS transactions to complete. An empty value will indicate no timeout. The default value is 5000.

5. Click the Finish button on the Edit Plug-in: DiameterAdapterPlugin dialog box.

20.3.2.1 Enabling SCTP Transport

The Diameter Adapter supports SCTP Transport only on Solaris SPARC10 and Linux. Perform the following steps to enable SCTP transport for the Diameter Adapter:

1. Enable SCTP on the server where Oracle Communications Universal User Profile is running.
2. In Linux and Solaris SPARC 10 environments only, in the *ORACLE_INSTANCE*/config/OPMN/opmn/opmn.xml file, update `LD_LIB_PATH` by adding the following to the end of the `java-options` property for the Oracle Virtual Directory server entry:
   - For Linux, add the following as one line:

     -Djava.library.path=$*ORACLE_HOME*/ovd/diameter/native/linux32/libsctpwrapper.so
   - For Solaris SPARC10, add the following as one line:

     -Djava.library.path=$*ORACLE_HOME*/ovd/diameter/native/solaris10_sparc64/libsctpwrapper.so
3. Add the following to the end of the `java-classpath` property for the same Oracle Virtual Directory server entry in the *ORACLE_INSTANCE*/config/OPMN/opmn/opmn.xml file:

   :$*ORACLE_HOME*/ovd/diameter/lib/sctp.jar
4. Set the Diameter Adapter's **nosctp** configuration parameter to `false`.
5. Set the Diameter Adapter's **listener** configuration parameter to the fully qualified domain name of the Oracle Communications Universal User Profile server.
6. Set the Diameter Adapter's **remotePeers** configuration parameter to the fully qualified domain name and port of the HSS server.
7. Set the Diameter Adapter's **remoteHost** configuration parameter to the fully qualified domain name of the HSS server.
8. Restart the Oracle Communications Universal User Profile server.

20.4 Mapping IMS 3GPP Schema to LDAP Schema

Oracle Communications Universal User Profile supports version 6 Sh user profile as defined in the 3GPP TS 29.328 specification (which Oracle recommends becoming familiar with to understand IMS user profiles).

`schema.diameter.xml` is the Oracle Communications Universal User Profile mapped LDAP schema and defines an LDAP `orclHSSProfile` objectclass containing the following attributes:

- `orclHSSPublicId`: Maps to the Public User Identity IMS user profile as defined in the 3GPP TS 23.008 specification. `orclHSSPublicId` is the filter attribute typically used to query the HSS Sh profile.
- `orclHSSServiceProfile`: In `ldapsearch` results orclHSSServiceProfile contains the binary value of the XML document contained in the `User-Data`

attribute-value pair of the Diameter Response message. This XML document conforms to version 6 Sh XSD.

In `ldapmodify` requests orclHSSServiceProfile contains the RepositoryData serviceIndicator to be updated in the following format:

<serviceIndicator>:<serviceIndicator_value>

- `orclHSSMSISDN`: An alternative to `orclHSSPublicId`, `orclHSSMSISDN` maps to the Mobile Station ISDN Number as defined in the 3GPP TS23.008 specification.

Example Entry

Generally, the Diameter data will be linked to a core record via the Join View adapter. For example, the subscriber's core record exists in the billing database but the phone's actual capabilities are stored in the HSS server. The following is an example of what the data may appear like when retrieved and printed out as an LDIF file:

```
uid=user1,ou=uup2,dc=imc,dc=com
givenname=Jane
sn=Doe
telephonenumber=15551234567
objectclass=top
objectclass=person
objectclass=organizationalperson
objectclass=inetorgperson
objectclass=orclHSSProfile
uid=user1
cn=Jane Doe
title=Associate
authpassword;oid={SASL/MD5}CPvyoxkufZJ69n0YBwfPsw==
authpassword;oid={SASL/MD5-DN}JnDX4y0mf8vdN1dAHVutDw==
authpassword;oid={SASL/MD5-U}/a3vgMwtjQKe8dEcAdxQwQ==
orclHSSPublicId=user1
orclHSSServiceProfile=<?xml version = '1.0' encoding = 'UTF-8'?>
<Sh-Data>
<RepositoryData>
  <PublicIdentifiers>
    <IMSPublicIdentity>
   sip:test.user@test.company.com
    </IMSPublicIdentity>
  </PublicIdentifiers>
  <ServiceIndication>DualRingDiameter</ServiceIndication>
  <SequenceNumber>0</SequenceNumber>
</RepositoryData>
</Sh-Data>
vdejoindn=Diameter:orclhsspublicid=user1,ou=Custom,dc=hsscontractid=2001-A57
```

Note: Notice the `orclHSSServiceProfile` attribute value is stored as XML, which allows for a complex data value to be stored for this attribute without needing to reference another LDAP object.

The `orclHSSServiceProfile` value contains the result value for the DataReference type for the IMSPublicIdentity. The DataReference type is specified by a numeric value in the communication, however each value corresponds to a type. The result values for the DataReference type are listed in Table 20–2:

Table 20–1 Result Values for DataReference Type

Type	Numeric Value
REPOSITORY DATA	0
IMS PUBLIC IDENTITY	10
IMS USER STATE	11
SCSCF NAME	12
INITIAL FILTER CRITERIA	13
LOCATION INFORMATION	14
USER STATE	15
CHARGING INFORMATION	16
MSISDN	17
PSIActivation	Currently not supported

By default, Oracle Communications Universal User Profile sends the value corresponding to IMS PUBLIC IDENTITY with All-Identities SH UDR to the HSS and retrieves the complete user identity set.

If clients must send different UDR requests, an LDAP control defined as 2.16.840.1.23008.2.9.28 can be used. The data in the control takes the numeric value followed by a semicolon (`;`) and Reference Identifier, for example: `0;DualRingTone`. The numeric value corresponds to one of the UDR values listed in Table 20–2:

Table 20–2 Result Values for DataReference Type

Type	Numeric Value
REPOSITORY DATA	0;*serviceIndication*
IMS PUBLIC IDENTITY	10;*identitySet* *identitySet* supports integer values of 0, 1, and 2, as follows: ■ 0 = All-Identities ■ 1 = Registered-Identities ■ 2 = Implicit-Identities
IMS USER STATE	11
SCSCF NAME	12
INITIAL FILTER CRITERIA	13;*serverName*
LOCATION INFORMATION	14;*requestedDomain* *requestedDomain* supports integer values of 0 and 1 as follows: ■ 0 = CS-Domain ■ 1 = PS-Domain
USER STATE	15;*requestedDomain* *requestedDomain* supports integer values of 0 and 1 as follows: ■ 0 = CS-Domain ■ 1 = PS-Domain

Table 20–2 (Cont.) Result Values for DataReference Type

Type	Numeric Value
CHARGING INFORMATION	16
MSISDN	17
PSIActivation	Currently not supported.

The Diameter adapter supports writes and reads. When updating the data stored in `orclHSServiceProfile`, the data must be supplied in the following format:

<serviceIndicator>:*<serviceIndicator_value>*

For example: `DualRingTone:30`

Part IV

Appendixes

This part contains the following appendixes:

- Appendix A, "Comparing Oracle Virtual Directory 11g Release 1 (11.1.1) and 10g Releases (10.1.4.x)"
- Appendix B, "Starting and Stopping the Oracle Stack"
- Appendix C, "HTTP Listener's Web Gateway Service"
- Appendix D, "Troubleshooting Oracle Virtual Directory"

A

Comparing Oracle Virtual Directory 11g Release 1 (11.1.1) and 10g Releases (10.1.4.x)

This appendix compares the implementation of fundamental items in Oracle Virtual Directory between 11*g* Release 1 (11.1.1) and legacy Oracle Virtual Directory 10*g* Releases (10.1.4.x). The information in this appendix is provided to give you an overview of implementation changes between the releases and to provide orientation after you upgrade to 11*g* Release 1 (11.1.1).

This appendix contains the following topics:

Note: The following sections include directory paths such as:

ORACLE_INSTANCE/config/OVD/*ovd1*

Be aware that *ORACLE_INSTANCE* and *ovd1* are documented using italic font because they represent directory names that may differ depending on your specific environment.

A.1 Default Super User

In 10*g* Releases (10.1.4.x):

- cn=admin

In 11*g* Release 1 (11.1.1):

- cn=orcladmin

A.2 Process Management

In 10*g* Releases (10.1.4.x):

- Start, stop, restart, and other processes controlled using vde.sh on UNIX/Linux platforms and OViDserver.exe on Windows platforms.
- One Oracle Virtual Directory process for each Oracle Virtual Directory installation

In 11*g* Release 1 (11.1.1):

- Start, stop, restart, and other processes controlled using `opmnctl`, the command-line interface to Oracle Process Manager and Notification Server.
- Multiple Oracle Virtual Directory instances can be created using opmnctl.

  See: Chapter 10, "Managing Oracle Virtual Directory Server Processes" for more information.

A.3 Location of Configuration Files

In 10*g* Releases (10.1.4.x):

- Configuration files are located in: *ORACLE_VIRTUAL_DIRECTORY_HOME*/conf

In 11*g* Release 1 (11.1.1):

- Configuration files are located in: *ORACLE_INSTANCE*/config/OVD/*ovd1*

A.4 Location of Plug-In Files

In 10*g* Releases (10.1.4.x):

- Plug-in files are located in: *ORACLE_VIRTUAL_DIRECTORY_HOME*/plugins/lib

In 11*g* Release 1 (11.1.1):

- Files for the default plug-ins included with Oracle Virtual Directory are located in: *ORACLE_HOME*/ovd/plugins/lib
- Custom plug-in files are located in: *ORACLE_INSTANCE*/OVD/*ovd1*/plugins/lib

A.5 Location of Deployed Mapping Files

In 10*g* Releases (10.1.4.x):

- Deployed Mapping files are located in:

  ORACLE_VIRTUAL_DIRECTORY_HOME/mappings

In 11*g* Release 1 (11.1.1):

- Deployed Mapping files are located in:

  ORACLE_INSTANCE/OVD/*ovd1*/mappings

A.6 Location of Log Files

In 10*g* Releases (10.1.4.x):

- Log files are located in:
  - *ORACLE_VIRTUAL_DIRECTORY_HOME*/log/vde.log
  - *ORACLE_VIRTUAL_DIRECTORY_HOME*/log/vde.log.exception

In 11*g* Release 1 (11.1.1):

- Log files are located in:

  ORACLE_INSTANCE/diagnostics/logs/OVD/*ovd1*/diagnostic.log

A.7 Location of Local Store Adapter Data Store

In 10*g* Releases (10.1.4.x):

- The Local Store Adapter data store is located in:

  ORACLE_VIRTUAL_DIRECTORY_HOME/data

In 11*g* Release 1 (11.1.1):

- The Local Store Adapter data store is located in:

  ORACLE_INSTANCE/OVD/*ovd1*/data

A.8 Location of Schema Files

In 10*g* Releases (10.1.4.x):

- Schema files are located in: *ORACLE_VIRTUAL_DIRECTORY_HOME*/conf

In 11*g* Release 1 (11.1.1):

- Schema files are located in:
  - *ORACLE_HOME*/ovd/schema
  - *ORACLE_INSTANCE*/config/OVD/*ovd1*/schema.user.xml

A.9 Location of Oracle Virtual Directory Server Libraries

In 10*g* Releases (10.1.4.x):

- Oracle Virtual Directory server libraries are located in:

  ORACLE_VIRTUAL_DIRECTORY_HOME/server/lib

In 11*g* Release 1 (11.1.1):

- Oracle Virtual Directory server libraries are located in:

- *ORACLE_HOME*/ovd/jlib
- *ORACLE_HOME*/ovd/adapters/lib

Note: Oracle Virtual Directory Adapter libraries are located in separate JAR files.

A.10 Enabling Oracle Virtual Directory Server Debugging

In 10*g* Releases (10.1.4.x):

- To enable Oracle Virtual Directory server debugging, you set the log level to `Debug` or `Dump` using the Oracle Virtual Directory Manager interface.

In 11*g* Release 1 (11.1.1):

- To enable Oracle Virtual Directory server debugging, set the log level to `TRACE:1` for debug or `TRACE:32` for Dump using Fusion Middleware Control or WLST.

  See: "Managing Oracle Virtual Directory Logging" on page 17-1 for more information.

A.11 Graphical User Interfaces

In 10*g* Releases (10.1.4.x):

- Oracle Virtual Directory Manager (DME)

In 11*g* Release 1 (11.1.1):

- Fusion Middleware Control
- Oracle Directory Service Manager

  See: The following topics for more information about the graphical user interfaces listed above:

  - "Fusion Middleware Control" on page 1-9
  - "Oracle Directory Services Manager" on page 1-9

A.12 Command-Line Tools

In 10*g* Releases (10.1.4.x):

- vde.sh on UNIX/Linux platforms and OViDserver.exe on Windows platforms.

In 11*g* Release 1 (11.1.1):

- opmnctl
- WebLogic Scripting Tool (WLST)
- syncovdconfig

See: The following topics for more information about the command-line tools listed above:

- Chapter 10, "Managing Oracle Virtual Directory Server Processes" on page 10-1
- "Getting Started with WLST for Oracle Virtual Directory" on page 8-12
- "Copying Configuration Files Between Oracle Virtual Directory Servers Using syncovdconfig" on page 9-12

A.13 Updating Classpaths

In 10*g* Releases (10.1.4.x):

- vde.sh on UNIX/Linux platforms and OViDserver.lax on Windows platforms.

In 11*g* Release 1 (11.1.1):

- *ORACLE_INSTANCE*/config/OPMN/opmn/opmn.xml to update the `java-classpath` of the *ovd1* component.

A.14 Synchronizing the Configuration of Two Oracle Virtual Directory Components

In 10*g* Releases (10.1.4.x):

- Manually copy all configuration and data files

In 11*g* Release 1 (11.1.1):

- syncovdconfig command-line tool

See: "Copying Configuration Files Between Oracle Virtual Directory Servers Using syncovdconfig" on page 9-12 for more information.

A.15 Audit Configurables

In 10*g* Releases (10.1.4.x):

- Access log file path

In 11*g* Release 1 (11.1.1):

- Filter Preset
- Audit Log Rotation
- Purge

See: "Managing Oracle Virtual Directory Auditing" on page 17-3 for more information.

A.16 Audit Log Location

In 10*g* Releases (10.1.4.x):

- *ORACLE_VIRTUAL_DIRECTORY_HOME*/log/access.log (default)

In 11*g* Release 1 (11.1.1):

- *ORACLE_INSTANCE*/auditlogs/OVD/*ovd1*/audit.log

B

Starting and Stopping the Oracle Stack

You must start and stop the components of the Oracle stack in a specific order. This appendix describes that order and contains the following topics:

- Starting the Stack
- Stopping the Stack

> **Note:** When executing the scripts to start and stop WebLogic managed components (respectively, startManagedWebLogic.sh and stopManagedWebLogic.sh), as described in the following topics:
>
> - The default value for *DOMAIN_NAME* is IDMDomain
> - *SERVER_NAME* represents the name of the Oracle WebLogic Managed Server. Its default value is wls_ods1.
> - You will be prompted for values for *USER_NAME* and *PASSWORD* if you do not provide them as options when you execute the script.
> - The value for *ADMIN_URL* will be inherited if you do not provide it as an option when you execute the script.

Starting the Stack

Start the stack components in the following order:

1. Start the Oracle WebLogic Administration Server by executing the following command:

   MW_HOME/user_projects/domains/*DOMAIN_NAME*/bin/startWebLogic.sh

   > **Note:** When you start the Oracle WebLogic Administration Server from the command line, it runs in the foreground and prints output to the screen.

2. Ensure the Node Manager is running. Normally, the Oracle WebLogic Administration Server starts the Node Manager. If the Node Manager is not running, start it by executing the following command:

   MW_HOME/user_projects/domains/*DOMAIN_NAME*/bin/startNodeManager.sh

3. Start system components, such as Oracle Internet Directory and Oracle Virtual Directory, by executing the following command:

   ORACLE_INSTANCE/bin/opmnctl startall

You can verify that the system components have started by executing the following command:

```
ORACLE_INSTANCE/bin/opmnctl status -l
```

4. Start WebLogic managed components, such as Oracle Directory Services Manager, by executing the following command:

```
MW_HOME/user_projects/domains/DOMAIN_NAME/bin/startManagedWebLogic.sh \
SERVER_NAME {ADMIN_URL}
```

Note: You can use the Oracle WebLogic Administration Console to start managed components in the background. See *Oracle Fusion Middleware Introduction to Oracle WebLogic Server* for more information.

Stopping the Stack

You can stop the Oracle WebLogic Administration Server and all the managed servers by using Oracle WebLogic Administration Console. See *Oracle Fusion Middleware Introduction to Oracle WebLogic Server* for more information.

To stop the stack components from the command line, perform the following steps:

1. Stop WebLogic managed components, such as Oracle Directory Services Manager, by executing the following command:

```
MW_HOME/user_projects/domains/DOMAIN_NAME/bin/stopManagedWebLogic.sh \
{SERVER_NAME} {ADMIN_URL} {USER_NAME} {PASSWORD}
```

2. Stop system components, such as Oracle Internet Directory and Oracle Virtual Directory, by executing the following command:

```
ORACLE_INSTANCE/bin/opmnctl stopall
```

3. Stop the Oracle WebLogic Administration Server by executing the following command:

```
MW_HOME/user_projects/domains/DOMAIN_NAME/bin/stopWebLogic.sh
```

4. If you want to stop the Node Manager, you can use the `kill` command:

```
kill -9 pid
```

C

HTTP Listener's Web Gateway Service

Oracle Virtual Directory's HTTP Listener provides an HTML-based gateway, known as the Web Gateway, that provides DSML and XSLT-rendered directory reporting. The Web Gateway's DSML functionality is based on Oracle's URL-based query gateway that returns DSML formatted XML results based on URL based operations. A DSML*v2* service is offered separately through the HTTP Listener's DSML V2 Service.

This appendix describes the HTTP Listener's Web Gateway and contains the following subsections:

C.1 Web Gateway Functionality and Features

In addition to providing DSML and XSLT-rendered directory reporting, the Web Gateway also provides the ability to serve static web content, allowing you to construct sophisticated directory white pages, delegated administration, and self-service functions.

The following is a list of Web Gateway features:

- dynamic integration with the Oracle Virtual Directory directory service.
- Apache-like security with LDAP integration providing authentication and authorization.
- dynamic XSLT translation providing output rendered in XHTML, DSML, WML, or any XML derivative form.
- support for static content, for example, typical HTML files.
- extended support for form-based input supporting all LDAP operations, including add, modify, and delete.
- extended support for retrieving binary attributes such as jpegphoto or usercertificates.

- integrated support of Oracle Virtual Directory's denial of service protection mechanism.

C.2 Demonstration Directory Browser

Oracle Virtual Directory includes a demonstration directory browser in the htdocs directory that highlights the capabilities of the XSLT Listener's XML services. This demonstration directory browser is not designed to be a sophisticated tool for production purposes, but is provided as example code that developers may review and extend for their specific purposes.

Note: Oracle does not provide coding level support for Oracle Virtual Directory. However, Oracle can provide assistance through its consulting services programs to provide development assistance.

C.3 Web Gateway Architecture

The Web Gateway is based on the Jetty server used in the popular open source Apache AXIS and Tomcat servers. The Jetty server is an extensible server designed for embedding into special purpose products. The following is a description of the components in the Web Gateway architecture, including:

- DSML Serverlet
- XSLT Serverlet
- Handlers

C.3.1 DSML Serverlet

The DSML serverlet receives request and calls Oracle Virtual Directory's internal API so that the request is treated as if it came from the Listener. The directory can be queried for standard DSML content by using the /dsml prefix with an appropriate query, for example:

```
http://localhost:8080/dsml/directory?base=o=myorg.com&scope=base&filter=objectclass=*
```

When the URL is received by the Listener, the query is processed in the context of the HTTP session. If no user security principal is present, the query is treated as an anonymous query. After the query is processed, results are returned to the HTTP client in DSML standard form.

Security context can be established by replacing "directory" with a server directory context. A security context for DSML can be established by creating an .htaccess file in an appropriate subdirectory of htdocs. For example, to set a restrictive query domain, place a .htaccess file in the /htdocs/dsml/administrators directory, causing queries to http://localhost:8080/dsml/administrators to meet the access requirements specified in the /htdocs/dsml/administrators/.htaccessaccess file.

C.3.2 XSLT Serverlet

The XSLT serverlet allows you to control how the HTML appears. The servlet provides the ability to query the directory for content and to have it translated according to a specified XSLT translation file.

The XSLT servlet supports both directory queries and specialized commands to support specialized processing. For example:

```
http://localhost:8080/search/results.xsl?params=...
```

The .xsl file suffix indicates the XSLT servlet and /search/results.xsl specifies the XSL translation file used to render the results and /search/results.xsl is automatically mapped to /htdocs/search/results.xsl.

C.3.3 Handlers

- **DoS Handler**: Every request is tracked by subject name and IP address against overall DoS Manager quotas (as defined in vde.prop). During a quota exceeded event, the server becomes unresponsive to the offending client, which protects the web service in the event of an over-active client or denial of service attack.
- **LDAPUserRealm**: The LDAPUserRealm works with the HTAccessHandler and integrates the Web Gateway with the directory service, that is, Oracle Virtual Directory. The LDAPUserRealm performs user authentication against the directory and tracks user context for other operations, including providing uid to distinguished name mapping.
- **HTAccessHandler**: This handler provides Apache-like web server access control by examining each resource requested and looking for an .htaccess file in the resources directory or parent directory. If a .htaccess file is not found, the default access control specified in /conf/htaccess.prop is used. If the current user context is not authorized to see the current resource, HTAccessHandler returns an access denied response to the HTTP client. When the browser responds with a user-id and password, the LDAPUserRealm service automatically creates an LDAP User Principal, which can be referenced by the HTAccessHandler for authorization purposes.

  Refer to "Security Contexts" on page C-7 for more information.
- **Servlet Handler**: The servlet handler directs URL requests ending in .xsl to the Web Gateway servlet and URL requests beginning with /dsml to the DSML servlet. The Web Gateway servlet provides XSLT formatted results and support both directory query and entry modification operations. The DSML servlet provides simple directory query and DSML formatted response functions.
- **Resource Handler**: All other requests are treated as document resource requests. If the resource is located in the htdocs document root of the server, the document is returned to the HTTP client. Static content, for example, html and gif files, may be served from the Web Gateway via the Resource Handler just as on a standard web server. Access control on static content is enforced according to the presence of a .htaccess security configuration file in the document directory or parent directory. Static documents found in the htdocs directory of the Oracle Virtual Directory installation are mapped directly to the main server root. For example, /htdocs/index.html maps to http://localhost:8080/index.html.

C.4 DSML and XSLT LDAP Query Parameters

The Web Gateway supports the following DSML and XSLT LDAP query parameters:

base=[dn]
The base distinguished name where the search begins. If empty, null is assumed.

filter=[searchfilter]]
An LDAP standard search filter. If empty, "(objectclass=*)" is assumed.

scope=[base|onelevel|sub]]
The scope of the search. Base specifies that only the entry specified in the base is searched (useful for directly addressing an entry). `onelevel` specifies that searches will look only at the immediate child entries below the base dn. Subtree specifies that all descendants of the basedn will be searched. By default, scope is set to base.

binddn=[userDN]
The user distinguished name the search is to execute under. If empty, the default assumed is anonymous.

password=[password]
The password associated with the binddn account. This parameter is normally specified in conjunction with a binddn.

xsl=[xsl-file] (Deprecated)
This parameter applies only to the XSLT serverlet and is ignored by the DSML serverlet. When specified, the parameter references a file relative to the root installation directory of Oracle Virtual Directory. HTML files are relative to the htdocs directory). If empty, the default value is conf/html.xsl

[any]=[value]
The XSLT serverlet accepts any additional parameters you want to provide. This is useful if you want to pass a parameter to an XSL style sheet to provide contextual information, for example, `parentname="Home"`.

C.5 Web Gateway Commands

Web Gateway commands provide additional features such as binary attribute retrieval, form-based searching, and form-based entry manipulation. The following is a description of the each of the Web Gateway commands, including:

- Binary Attribute Retrieval Commands
- Form-Based Searching Commands
- Form-Based Entry Manipulation Commands
- HTTP POST
- HTTP GET

C.5.1 Binary Attribute Retrieval Commands

The getcert and getphoto commands are used to retrieve binary attributes in their native binary forms, as opposed to encoded in DSML. A call to getcert returns http content with mime type application/x-x509-email-cert. Similarly, a call to getphoto will return content with mime type image/jpeg.

cmd=getcert
Retrieves a certificate specified by the dn and attr parameters. For example:

```
http://localhost:8080/xslt/search/?cmd=getcert&dn=uid=jsmith,ou=people,o=airius.com&attr=usercertificate
```

cmd=getphoto
Retrieves a jpegphoto as specified by the dn and attr parameters.

dn=[distinguished name]
Specifies the distinguished name of an entry. Typically used with a command parameter.

attr=[attribute name]
Specifies the name of an attribute in relation to a getcert or getphoto command.

C.5.2 Form-Based Searching Commands

While it is possible to do an LDAP-like XSLT/DSML style search, the `cmd=search` option allows more flexibility for supporting user-friendly search forms. This command takes form values and converts the content into a standard directory query.

cmd=search
Provides support for a form-based search with related parameters: base, scope, kind, searchvalue, and filterattrs. This command provides similar functionality to the XSLT query form except that it is designed to make it easier to program search forms where the search filter is automatically constructed from the form parameters. Example:

```
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]
Specifies the search base for a form-based search command.

scope=[base|one|sub]
Specifies the scope of the search. Valid values are base, one, and sub.

kind=[all|attrname]
Specifies whether the search filter will be constructed based on a specific attribute or based on all attributes specified by the filterattrs parameter

filterattrs=[attr1,attr2,…]
The list of attributes to search against.

searchvalue=[user searchstring value]
The value entered by the user as the searchstring value.

The action of a search form can be constructed as a GET or a POST. The following is an example of a GET request generated by a search form:

```
http://localhost:8080/xslt/search/results.xsl?cmd=search&base=o=airius.com&scope=sub&kind=all&filterattrs=cn,sn,givenname&searchvalue=smith
```

This command would be interpreted by the server and processed as equivalent to:

```
http://localhost:8080/xslt/search/results.xsl?base=o=airius.com&scope=sub&filter=(|(cn=*smith*)(sn=*smith*)(givename=*smith*))
```

After processing the query, the results would be translated using the XSL file /xslt/search/results.xsl.

C.5.3 Form-Based Entry Manipulation Commands

The modify command allows for construction of html forms that can be used to modify directory entries. The commands provide the ability to add, delete, or modify virtually any LDAP directory entry.

Some form parameters can be provided as hidden fields, for example, attrs, changetype, and objectclass. When adding new entries, remember to supply the objectclass attribute, which is usually supplied as a hidden field.

Note: The term attrx in the following descriptions represents any valid LDAP attribute.

cmd=modify
Indicates the client has requested modification of a directory entry. The type of modification is specified in the changetype field.

changetype=[add|modify|delete]
Specifies the operation on the directory entry to be performed: add, modify, or delete. Only one directory entry can be modified in any operation.

When adding a new entry using changetype=add only the attrx_modify=[value] values is processed. When deleting an entry, all attrx parameters are ignored. You can delete multiple LDAP entries by providing multiple dn form values.

dn=[distinguished name]
The distinguished name of the entry to be added, modified, or deleted.

modify_attrs=[attr1,attr2,attr3]
The list of attributes to be processed from the form. This list will be used to locate all other attribute parameters on the form, for example, attr1_type, attr1_action, attr1_new, attr1_confirm.

xslerror=[/dir/error.xsl]
Specifies an xsl file to direct errors to if a processing error occurs. In the event of an error, the XSL form will pass all form parameters provided and the result and message parameters. These can be parsed in error.xsl to determine the appropriate error message to display to the end user.

attrx_type=[single|multi|photo|cert]
Defines the type of attribute represented by attrx. If it is single, only the first value of attrx is accepted. If it is cert or photo, the binary value is located as part of a multi-part form. One value can be added at a time.

attrx_required=[true|false]
If set to true, the processor verifies a value is present for attrx.

attrx_delete=[value]
A value to be deleted from attrx. One or more values can be specified. An empty value is ignored.

attrx_modify=[value]
The value used to modify the attrx attribute. Depending on the value of attrx_action, this value will be added to attrx or will replace all values in attrx.

attrx_action=[add|replace]
Specifies whether the value in attrx_modify is added to attrx or will replace all existing values. When changetype=add, attrx_action is forced to be add.

rdn_attr=[attr]
During an `add` operation, `dn` is assumed to be the parent entry, and `rdn_attr` informs the processor which attribute to use for the `rdn` value.

C.5.4 HTTP POST

You can use HTTP POST interchangeably with HTTP GET. There is a maximum input file size in the Listeners.prop file that can be set to prevent denial of service or buffer overflow attacks. Use multi-part forms as per RFC 1867 when binary attributes, for example jpegphotos and certificates, are to be uploaded.

C.5.5 HTTP GET

When using HTTP GET operations, all parameters are passed as a URL, allowing you to bookmark the URLs. However, confidential information, such as the `binddn`, is available in the URL, so you may want to use HTTP POST for non-anonymous queries.

You must use escape ampersands (&) and spaces appropriately to conform with URL requirements when using GET. This may require the addition of javascript to modify field parameters prior to submission. In general, ampersands (&) must be replaced with `&`.

C.6 Security Contexts

The HTAccessHandler allows you to define security contexts within the server. A Security Context defines which IP addresses, subjects, or groups may access a particular resource directory on the server. You can create security contexts by creating a .htaccess security configuration file in the htdocs directory or any sub-directory. Each .htaccess file applies to directory it resides in and any child directories below it. The .htaccess file in the immediate directory or closest parent is the file applied to any resource. You can define a default security context by editing the /conf/htaccess.prop file.

The following information provides more detail on security contexts, including:

- Requirements for .htaccess Files
- Directives for .htaccess Files
- Resource Restrictions
- Example Security Context Files

C.6.1 Requirements for .htaccess Files

The following is a list of requirements for .htaccess files:

- The .htaccess file must be a plain text file.
- Any content that is not a valid directive must be prefixed with a comment (#) character.
- Directives in a .htaccess file apply to the current or child directories only.
- If a sub or child directory contains a .htaccess file, it will override the .htaccess files of its parent directories.

C.6.2 Directives for .htaccess Files

The following is a list of supported .htaccess file directives:

AuthUserFile [filename]
Indicates users will be authenticated using a text file that contains a list of users and passwords encrypted with the CRYPT hash. Refer to Example C–2 to see an example.

AuthUserLDAP [base]
Indicates users will be authenticated from the Oracle Virtual Directory server. base specifies the distinguished name of the search base for authenticated users.

AuthGroupFile [filename]
Indicates users must be a member of the group file specified by filename. Refer to Example C–3 for an example.

AuthName [name]
Sets the name of the authentication realm. This is the text displayed in the challenge to the user, that is, in the Directory Browser.

<Limit>
[restrictions]

Refer to "Resource Restrictions" on page C-8 for more information.

</Limit>
Defines a resource restrictions block. Restrictions are placed within the tokens <Limit> and </Limit>.

C.6.3 Resource Restrictions

Resource restrictions are placed between <Limit> statements. The following is a list of the supported restrictions:

satisfy [all|any]
Specifies where all or any requirements must be satisfied.

require [user|valid-user|group|ldapgroup] [entities]
Specifies that the user must be a member of one of the following:

- user: either a member of AuthUserFile or AuthUserLDAP
- valid-user: any validated user
- group: a member of AuthUserGroup
- ldapgroup: a member of an ldap group

entities specifies either an ldapgroup enclosed in round brackets(), or an explicit list of user names or groups.

order [allow,deny|deny,allow|mutual-failure]
Specifies rule precedence for allow and deny rules as follows:

- allow,deny: Must be on the allow list and cannot be on the deny list.
- deny,allow or mutual-failure: Allow if not on the deny list or allow if on the allow list.

allow from [ip address list]
Specifies a space-separated list of IP addresses to allow requests from.

deny from [ip address list]
Specifies a space-separated list of IP addresses to deny requests from.

C.6.4 Example Security Context Files

The following are example security context files. Example C–1 demonstrates an example .htaccess file:

Example C–1 Example .htaccess file

```
# 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>
```

Example C–2 demonstrates an example htpasswd file:

Example C–2 Example htpasswd file

```
# <PRE>
# HTPassword example.
#
# Passwords must be in CRYPT format
#
admin: adpexzg3FUZAk
tom: tofn8Rh1L.xhQ
dick: diF1tp4K2rvw.
harry: haVyKPUXAHTjA
```

Example C–3 demonstrates an example htgroup file:

Example C–3 Example htgroup file

```
# HT Group example.
# Place uid values after group name
#
Users: tom dick harry jturner
TestGroup: tom harry
AdminGroup: admin
```

C.7 Using XSL Stylesheets

The following information describes techniques for using XSL stylesheets, including:

- Using XSLT Serverlet Queries to Create Dynamic Groups
- Setting Content-Type Serverlet Based on Media Type Attribute
- Support for XSL Document() and Import/Include Commands

- Passing Parameters to XSL Stylesheets
- Example XSL

C.7.1 Using XSLT Serverlet Queries to Create Dynamic Groups

You can use an XSLT query combined with a stylesheet to create an edit form for a client, for example:

```
http://localhost:8080/manage/edit.xsl?base=ou=Atlanta,o=myorg.com&scope=base&filte
r=objectclass=*&mode=edit
```

This URL causes the server to retrieve the directory entry `ou=Atlanta,o=myorg.com` and pass it to the `/manage/edit.xsl` xsl form.

Inside the xsl form, look for the `mode` parameter to determine how to render the result. In this case, `edit` might mean that the form should be rendered so that the object returned by the query can be edited. The `addchild` mode might be used to prepare a form that could be used to add a child entry to the organization.

You can create a new `mode` parameter. In the case of XSL, `mode` is a special parameter often used in templates to cause different logic to occur for different rendering subroutines. In the default shipping demonstration application, `htdocs/xslt/secure/admin.xsl` provides multi-mode editing (that is, display, add, modify, delete) within a single style sheet.

C.7.2 Setting Content-Type Serverlet Based on Media Type Attribute

When using an XSLT stylesheet to process data into HTML, the Oracle Virtual Directory Web Gateway examines the `xsl:output` tag for a media-type attribute. If one is present, the Web Gateway sets the content-type serverlet to the same setting as the media-type attribute. If a media-type attribute does not exist in the `xsl:output` tag, the content-type serverlet uses `text/html` by default.

C.7.3 Support for XSL Document() and Import/Include Commands

The XSLT servlet processor in Oracle Virtual Directory supports the `document()` command. You can specify either a complete document URL or a file specification. If you specify just a directory or document, it is treated as relative to the htdocs document root location.

For example, the vdexsl.xml file is used to translate LDAP attribute names into more readable names:

```
<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=$lname]/vdexsl:
displayname"/>
```

Files retrieved using the `document()` command are automatically cached for performance reasons. Similarly, the `xsl:import` and `xsl:include` commands are also supported and can be used to import global subroutines into your XSL templates. As with the `document()` command, references can be to any URL or file. By default, file references are relative to the htdocs document root location. The following are two examples:

```
<xsl:import href="http://192.168.0.2:8050/lib/globals.xslt"/>

<xsl:import href="lib/globals.xslt"/>
```

Note: If you are using HTTP to retrieve a stylesheet from an Oracle Virtual Directory, using the .xsl file suffix causes the XSLT servlet to intervene and render the xsl file as if it were part of a query. If you want to retrieve an unprocessed XSL file from another Oracle Virtual Directory, it must have a suffix name other than .xsl. In the example above, the .xslt directory is used instead.

C.7.4 Passing Parameters to XSL Stylesheets

The XSLT servlet makes all parameters that are passed to it available to the XSL style sheet. This allows XSL stylesheets to receive contextual information that may not be present in the DSML search results. Parameters can be part of the valid search parameters or other parameters defined for this purpose. In the following example code, the `parentname` and `base` parameters are retrieved and made available as variables within the XSL code:

```
…
<xml:output method="html" indent="yes"/>
<xsl:param name="parentname" select="'Root'"/>
<xsl:param name="base" select="'base'"/>
<xsl:template match="/">
<html>
…
```

The select part of the `xsl:param` statement represents the default value if it is not present.

C.7.5 Example XSL

The following Example C–4 is designed to build a directory navigation bar. You can consult any guide on XSLT for more information on XSL programming. Example C–4 pulls in two parameters in order to establish context. The `parentname` and `base` parameters are used to remember the parent entries name so that navigation up the tree as well as down the tree is possible. The parent's parent DN is assumed by dropping the RDN from the current `base` query parameter.

Example C–4 Example XSL

```
<?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={$parentdn}&amp;
xsl=htdocs/Browser-navbar.xsl" target="leftFrame">
<FONT FACE="Verdana,Arial" size="-1"><xsl:value-of select="$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
={.}&amp;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/res
```

```
ults.xsl" 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;parentna
me={$evalue}&amp;xsl=htdocs/Browser-navbar.xsl" target="leftFrame"><FONT
FACE="Verdana,Arial" size="-1"><xsl:value-of select="$evalue"/></FONT></a>
<FONT FACE="Verdana,Arial" size="-1">[<a
href="/xslt?base={@dn}&amp;scope=base&amp;filter=(objectclass=*)&amp;xsl=htdocs/r
esults.xsl" target="mainFrame">-></a>]</FONT></li>
</td></tr>
</xsl:if>
</xsl:template>
```

D

Troubleshooting Oracle Virtual Directory

This appendix describes common problems that you might encounter when using Oracle Virtual Directory and explains how to solve them and also provide information on diagnosing problems. It contains the following topics:

- Problems and Solutions
- Diagnosing Oracle Virtual Directory Problems
- Need More Help?

D.1 Problems and Solutions

This section describes common problems and solutions. It contains the following sections:

- Cannot Invoke Oracle Directory Services Manager
- Cannot Invoke Oracle Directory Services Manager from Fusion Middleware Control in Multiple NIC and DHCP Enabled Environment
- Cursor Problems When Accessing Oracle Directory Services Manager in Accessibility Mode Using Internet Explorer 7
- Oracle Directory Services Manager Failover Using Oracle HTTP Server is Not Transparent
- Oracle Directory Services Manager Loses Connection to Oracle Virtual Directory-Oracle RAC Database Configuration
- Error Returned After Querying Oracle Virtual Directory Configured with LDAP Adapters
- Error Returned After Querying Oracle Virtual Directory Configured with Database Adapters

D.1.1 Cannot Invoke Oracle Directory Services Manager

Problem

Attempting to invoke Oracle Directory Services Manager using a web browser fails.

Solution

- Verify the Oracle Virtual Directory server is running. The Oracle Virtual Directory server must be running to connect to it from Oracle Directory Services Manager.

- Verify you entered the correct credentials in the Server, Port, User Name and Password fields. You can execute an ldapbind command against the target Oracle Virtual Directory server to verify the server, user name, and password credentials.
- Verify you are using a supported browser. Refer to the Oracle Identity Management Certification Information on the Oracle Technology Network web site for information about supported browsers for Oracle Directory Services Manager. You can access the Oracle Technology Network web site at:

  `http://www.oracle.com/technology/index.html`
- Verify you specified the port of the Admin Listener—not the LDAP port—in the URL for Oracle Directory Services Manager.

D.1.2 Cannot Invoke Oracle Directory Services Manager from Fusion Middleware Control in Multiple NIC and DHCP Enabled Environment

Problem

The WebLogic Managed Server where Oracle Directory Services Manager is deployed has multiple Network Interface Cards or is DHCP enabled. Attempts to invoke Oracle Directory Services Manager from Oracle Enterprise Manager Fusion Middleware Control fail and return `404 errors`.

Solution

Use the WebLogic Server Administration Console to change the listen address of the Managed WebLogic Server so that the IP address or hostname in the URL for Oracle Directory Services Manager is accessible.

Perform the following steps:

1. Using a web browser, access the WebLogic Server Administration Console.
2. In the left pane of the WebLogic Server Administration Console, click **Lock & Edit** to edit the server configuration.
3. In the left pane of the WebLogic Server Administration Console, expand **Environment** and select **Servers**.
4. On the Summary of Servers page, click the link for the WebLogic Managed Server where Oracle Directory Services Manager is deployed.
5. On the Settings page for the WebLogic Managed Server, update the Listen Address to the host name of the server where Oracle Directory Services Manager is deployed.
6. Click **Save** to save the configuration.
7. Click **Activate Changes** to update the server configuration.

D.1.3 Cursor Problems When Accessing Oracle Directory Services Manager in Accessibility Mode Using Internet Explorer 7

Problem

In Internet Explorer 7, when you access Oracle Directory Services Manager in accessibility mode using only the keyboard, the cursor loses focus. This behavior has been observed under the following circumstances:

- You access the directory in SSL-enabled mode and the server certificate appears.

- You type an invalid password and the error dialog appears.

Solution

Press the Tab key nine times, then press the Enter key.

D.1.4 Oracle Directory Services Manager Failover Using Oracle HTTP Server is Not Transparent

Problem

When you perform an Oracle Directory Services Manager failover using Oracle HTTP Server, the failover is not transparent. You will see this behavior when you perform the following steps:

1. Oracle Directory Services Manager is deployed in a High Availability active-active configuration using Oracle HTTP Server.
2. Display an Oracle Directory Services Manager page using the Oracle HTTP Server name and port number.
3. Make a connection to an Oracle Virtual Directory server.
4. Work with the Oracle Virtual Directory server using the current Oracle Directory Services Manager Oracle HTTP Server host and port.
5. Shut down one managed server at a time using the WebLogic Server Administration Console.
6. Go back to the Oracle Directory Services Manager page and port, and the connection which was established earlier with Oracle Virtual Directory. When you do, a message is displayed advising you to re-establish a new connection to the Oracle Directory Services Manager page.

Solution

If you encounter this problem, perform the following steps:

1. In your web browser, exit the current Oracle Directory Services Manager page.
2. Launch a new web browser page and specify the same Oracle Directory Services Manager Oracle HTTP Server name and port.
3. Re-establish a new connection to the Oracle Virtual Directory server you were working with earlier.

See Also:

- "Configuring Oracle HTTP Server to Support Oracle Directory Services Manager in an Oracle WebLogic Server Cluster" on page 8-8.
- *The Oracle Fusion Middleware High Availability Guide* for more information about Oracle Directory Services Manager in High Availability configurations.

D.1.5 Oracle Directory Services Manager Loses Connection to Oracle Virtual Directory-Oracle RAC Database Configuration

Problem

Oracle Directory Services Manager temporarily loses its connection to an Oracle Virtual Directory component that is using an Oracle RAC Database. Oracle Directory Services Manager *might* display a message such as `Failure accessing Oracle database (oracle errcode=`*`errcode`*`)`, where *`errcode`* is one of the following values: `3113`, `3114`, `1092`, `28`, `1041`, or `1012`.

Solution

This error can occur during failover of the Oracle Database that the Oracle Virtual Directory component is using. The connection will be reestablished in less than a minute, and you will be able to continue without logging in again.

D.1.6 Error Returned After Querying Oracle Virtual Directory Configured with LDAP Adapters

Problem

Oracle Virtual Directory is configured with LDAP Adapters and the entries are visible in the proxied LDAP servers. An `LDAP err=1` operation error message returns after querying Oracle Virtual Directory from an LDAP client.

Solution

Verify the proxied LDAP servers are running. If they are not running, start them and query Oracle Virtual Directory again from the LDAP client.

D.1.7 Error Returned After Querying Oracle Virtual Directory Configured with Database Adapters

Problem

Oracle Virtual Directory is configured with Database Adapters and the entries are visible in the proxied databases. An `LDAP err=1` operation error message returns after querying Oracle Virtual Directory from an LDAP client.

Solution

Verify the proxied databases are running. If they are not running, start them and query Oracle Virtual Directory again from the LDAP client. If the operation error message returns again, verify the correct drivers for each database have been loaded into Oracle Virtual Directory as described in "Loading Libraries into the Oracle Virtual Directory Server" on page 9-12.

D.2 Diagnosing Oracle Virtual Directory Problems

This topic provides information on how you can diagnose Oracle Virtual Directory problems and contains the following sections:

- Increasing the Log Level to DEBUG
- Examining the Exceptions Logged to the Diagnostic Log

- Using the Dump Transactions Plug-In to Gather Information About Data Transformation Errors
- Monitoring the Oracle Virtual Directory Server Using Fusion Middleware Control Metrics

D.2.1 Increasing the Log Level to DEBUG

When an Oracle Virtual Directory error occurs, you can gather more information about what caused the error by performing the following steps:

1. Increase the log level to DEBUG by referring to Managing Oracle Virtual Directory Logging on page 17-1.
2. Repeat the task or procedure where you originally encountered the error.
3. Examine the log information generated using the DEBUG level.

D.2.2 Examining the Exceptions Logged to the Diagnostic Log

Examining the exceptions logged to the Oracle Virtual Directory log file will help you identity errors in target directories and in custom plug-ins and adapters. For example, if you receive an `LDAP error=1` message, you may examine the diagnostic log and find the cause of the error by a messages like `host not found` or `out of memory`.

You can access the diagnostic log in the following directory:

$*ORACLE_INSTANCE*/diagnostics/logs/OVD/*COMPONENT_NAME*/

D.2.3 Using the Dump Transactions Plug-In to Gather Information About Data Transformation Errors

The Dump Transactions plug-in generates a record of all transactions for each LDAP operation and logs the record to the Oracle Virtual Directory console log. You can configure the Dump Transactions plug-in to run on any log level. The Dump Transactions plug-in is particularly useful for diagnosing mapping and integration efforts while logic flows through the Oracle Virtual Directory system. You can use the Dump Transaction plug-in to analyze issues on a specific adapter without setting the entire server log level to a more verbose level. Think of the Dump Transactions plug-in as a protocol analyzer for Oracle Virtual Directory.

> **See Also:** "Dump Transactions Plug-In" on page 4-12 for more information.

D.2.4 Monitoring the Oracle Virtual Directory Server Using Fusion Middleware Control Metrics

Oracle Virtual Directory's Performance Summary page in Oracle Enterprise Manager Fusion Middleware Control allows you to view a variety of metrics of the Oracle Virtual Directory Server in a time based context. You can use these metrics to monitor Oracle Virtual Directory and to help diagnose problems.

> **Note:** You can customize the metrics displayed on the Performance Summary page using the Metric Palette. Refer to the *Oracle Fusion Middleware Administrator's Guide* for more information on using the Metric Palette.

To view the metrics on the Performance Summary page:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control and navigate to the Oracle Virtual Directory target that you want to view metrics for.
2. Select **Monitoring** and then **Performance Summary** from the Oracle Virtual Directory menu. The Performance Summary page appears.

Table D–1 lists the metrics that are available for the Oracle Virtual Directory Server on the Performance Summary page:

Table D–1 OVD Metrics on the Fusion Middleware Control Performance Summary Page

Metric	Description
CPU Usage (%)	Specifies the percentage of the CPU that Oracle Virtual Directory is using.
Other CPU Usage (%)	Specifies the percentage of the CPU that components other than Oracle Virtual Directory are using.
CPU Idle Time (%)	Specifies the percentage of the CPU Idle Time on the Oracle Virtual Directory host.
Memory Usage (MB)	Specifies the amount of memory consumed (in MB) by Oracle Virtual Directory Server.
Memory Usage (%)	Specifies the percentage of memory consumed by the Oracle Virtual Directory Server.
Other Memory Usage (MB)	Specifies the amount of memory consumed (in MB) by components other than Oracle Virtual Directory Server.
Other Memory Usage (%)	Specifies the percentage of memory consumed by components other than the Oracle Virtual Directory Server.
Free Memory (MB)	Specifies the amount of free memory (in MB) on the Oracle Virtual Directory Server host.
Free Memory (%)	Specifies the percentage of free memory on the Oracle Virtual Directory Server host.
Total Memory (MB)	Specifies the total available memory (in MB) on the Oracle Virtual Directory Server host.
Heap Usage (MB)	Specifies the JVM heap usage (in MB) of the Oracle Virtual Directory Server.
Up Time (ms since Epoch)	Specifies the amount of time (in milliseconds) that the Oracle Virtual Directory Server has been up and running.
Start Time (ms since Epoch)	Specifies the start time (milliseconds since Epoch) of the Oracle Virtual Directory Server.
UpDown Status	Specifies whether Oracle Virtual Directory is up and running or down and unavailable.
Total No of Operations	Specifies the total number of all LDAP operations that have been completed since the data last collection.
Total No of Open Connections	Specifies the total number of open connections to Oracle Virtual Directory Server.
Total No of Users Currently Connected	Specifies the total number of users that are currently connected to the Oracle Virtual Directory Server.
Total No of IPs Currently Connected	Specifies the total number of distinct IP Addresses that are currently connected to the Oracle Virtual Directory Server.
Current Connections (User)	Specifies the number of connections that are currently open for a particular user.

Table D–1 (Cont.) OVD Metrics on the Fusion Middleware Control Performance

Metric	Description
Total Connections (User)	Specifies the total number of connections opened by a particular user.
Current Connections (IP)	Specifies the number of current open connections from a particular IP Address.
Total Connections (IP)	Specifies the total number of connections from a particular IP Address.
Minimum Time to complete a search request	Specifies the minimum length of time Oracle Virtual Directory took to complete a search request since its last start.
Maximum Time to complete a search request	Specifies the maximum length of time Oracle Virtual Directory took to complete a search request since its last start.
Average time to complete an LDAP search request	Specifies the average length of time Oracle Virtual Directory took to complete an LDAP search request since its last start.
Number of LDAP Search Requests	Specifies the number of LDAP search requests since the last data collection.
Number of LDAP Add Requests	Specifies the number of LDAP add requests since the last data collection.
Number of LDAP Binds Requests	Specifies the number of LDAP bind requests since the last data collection.
Number of LDAP Delete Requests	Specifies the number of LDAP delete requests since the last data collection.
Number of LDAP Modify Requests	Specifies the number of LDAP modify requests since the last data collection.
Number of LDAP Rename Requests	Specifies the number of LDAP rename requests since the last data collection.
Total Operations	Specifies the total number of all LDAP operations since the last data collection.
Enabled (Adapter)	Specifies if an Oracle Virtual Directory Adapter is enabled or disabled.
Operational Version (Adapter)	Specifies the operational version of the Oracle Virtual Directory Adapters.
Provisioned Version (Adapter)	Specifies the provisioned version of the Oracle Virtual Directory Adapters.
Type (Adapter)	Specifies the type of the Adapter.
Total No of Operations Renamed (Adapter)	Specifies the total number of operations renamed by a particular Adapter.
Total No of Operation Binds (Adapter)	Specifies the total number of bind operations by a particular Adapter.
Total No of Connections Reused (Adapter)	Specifies the total number of connections reused by a particular Adapter.
Total No of Connections Processed (Adapter)	Specifies the total number of connections processed by a particular Adapter.
Minimum Time taken to complete a search request (Adapter)	Specifies the minimum length of time since the last Oracle Virtual Directory start that a particular Adapter took to complete an LDAP search request.
Maximum Time taken to complete a search request (Adapter)	Specifies the maximum length of time since the last Oracle Virtual Directory start that a particular Adapter took to complete an LDAP search request.

Table D–1 (Cont.) OVD Metrics on the Fusion Middleware Control Performance

Metric	Description
Average Time taken to complete a search request (Adapter)	Specifies the average length of time since the last Oracle Virtual Directory start that a particular Adapter took to complete an LDAP search request.
Operations Count (Adapter)	Specifies the total number of all operations performed by a particular Adapter.
Add Operations (Adapter)	Specifies the total number of add operations performed by a particular Adapter since the last data collection.
Modify Operations (Adapter)	Specifies the total number of modify operations performed by a particular Adapter since the last data collection.
Search Operations (Adapter)	Specifies the total number of search operations performed by a particular Adapter since the last data collection.
Delete Operations (Adapter)	Specifies the total number of delete operations performed by a particular Adapter since the last data collection.
Open Connections (Adapter)	Specifies the total number connections opened by a particular Adapter since the last data collection.

D.3 Need More Help?

You can find more solutions on My Oracle Support (formerly MetaLink) at http://metalink.oracle.com. If you do not find a solution for your problem, log a service request.

See Also:

- *Oracle Application Server Release Notes*, available on the Oracle Technology Network:

  http://www.oracle.com/technology/documentation/index.html