Oracle® COREid Access and Identity

Deployment Guide

10*g* Release 2 (10.1.2)
Part No. B19011-01

May 2005

ORACLE

Copyright © 1996-2005, Oracle. All rights reserved. US Patent Numbers 6,539,379; 6,675,261; 6,782,379; 6,816,871.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Oracle COREid Access and Identity products includes RSA BSAFE™ cryptographic or security protocol software from RSA Security. Copyright © 2003 RSA Security Inc. All rights reserved. RSA and RC4 are trademarks of RSA Data Security. Portions of Oracle Internet Directory have been licensed by Oracle Corporation from RSA Data Security. This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). Copyright © 1999-2003 The Apache Software Foundation. All rights reserved. Copyright © 2003 The Apache Software Foundation.

This program contains third-party code from Apache. Under the terms of the Apache Software License, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Apache software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Apache.

* The Apache Software License, Version 1.1
*
* Copyright (c) 2000 The Apache Software Foundation. All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
* distribution.
*
* 3. The end-user documentation included with the redistribution,
* if any, must include the following acknowledgment:
* "This product includes software developed by the
* Apache Software Foundation (http://www.apache.org/)."
* Alternately, this acknowledgment may appear in the software itself,
* if and wherever such third-party acknowledgments normally appear.

* 4. The names "Apache" and "Apache Software Foundation" must
 *   not be used to endorse or promote products derived from this
 *   software without prior written permission. For written
 *   permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache",
 *   nor may "Apache" appear in their name, without prior written
 *   permission of the Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * ==
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Software Foundation. For more
 * information on the Apache Software Foundation, please see
 * <http://www.apache.org/>.
 *
 * Portions of this software are based upon public domain software
 * originally written at the National Center for Supercomputing Applications,
 * University of Illinois, Urbana-Champaign.
 */

Contents

Preface

The *COREid Deployment Guide* provides information for people who plan and manage the environment in which COREid will run. This guide covers capacity planning, network topologies and system tuning.

Note: Oracle *COREid* was previously known as Oblix *Netpoint*. All legacy references to Oblix and NetPoint, for example, in screen shots, illustrations, and documentation titles, should be understood to refer to Oracle and COREid, respectively.

This Preface covers the following topics:

Intended Audience

This guide targets the knowledge and skill requirements of system, network, or COREid administrators responsible for optimizing the COREid implementation.

This document assumes that you are familiar with your network architecture, your LDAP directory, and firewall and internet security.

COREid Documentation

The manuals that are available for this release include:

Introduction to COREid—Provides an introduction to COREid, a road map to COREid manuals, and a COREid glossary of terms.

COREid Release Notes—Provides up-to-the minute details about the latest COREid release.

COREid Installation Guide—Explains how to install and configure the COREid components.

COREid Upgrade Guide—Explains how to upgrade earlier versions of COREid to the latest version of COREid.

COREid Administration Guide—Explains how to configure COREid applications to display information stored in the directory, how to assign view and modify permissions for data displayed on the COREid applications, and how to assign access controls to users.

COREid Deployment Guide—Provides information for people who plan and manage the environment in which COREid runs. This guide covers capacity planning, system tuning, failover, load balancing, caching, and migration planning.

COREid Customization Guide—Explains how to change the appearance of COREid applications and how to control COREid by making changes to operating systems, Web servers, directory servers, directory content, or by connecting CGI files or JavaScripts to COREid screens. This guide also describes the Access Server API and the Authorization and Authentication Plug-in APIs.

COREid Developer Guide—Explains how to create AccessGates and how to develop plug-ins. This guide also provides information to be aware of when creating CGI files or JavaScripts for COREid.

COREid Integration Guide—Explains how to set up COREid to run with third-party products such as BEA WebLogic, the Plumtree portal, and IBM Websphere.

COREid Schema Description—Provides details about the COREid schema.

Online Help is available from each COREid screen.

Typographical Conventions

COREid manuals use the following typographical conventions:

- When you are instructed to select elements sequentially, the actions are separated with angle brackets, as shown below:

  Click System Admin > System Configuration > View Server Settings.

- Paths to a file are shown using syntax for either the Unix or Windows platform:

  /COREid_install_dir/identity/oblix/logs/debugfile.lst
  COREid_install_dir\identity\oblix\logs\debugfile.lst

  where *COREid_install_dir* refers to the directory where the component, in this case, the COREid Server, is installed.

Contact Information

For a list of contacts including corporate offices world wide, sales, and other details, visit the Oracle Web site at:

`http://www.oracle.com`

You can contact Oracle with questions or comments as follows:

Customer Care—http://www.oracle.com/support/contact.html

Corporate Headquarters

Oracle maintains offices world wide. Oracle corporate headquarters is located at:

500 Oracle Parkway
Redwood Shores, CA 94065
Phone: (650) 506-7000

Before Contacting Customer Care

Before contacting Customer Care, please have available the following:

- Oracle product name and version number
- Type of computer and operating system you are using

Accessing the Customer Care Knowledge Base

For more information about using COREid, see the Oracle Customer Care Knowledge Base. To access the Knowledge Base, you need a login name and password, which you can obtain from your Oracle sales representative.

To access the Knowledge Base:

1. Enter the following URL in your browser and press Return.

   ```
   http://www.oracle.com/support/contact.html
   ```

2. Click the phrase, Login to the Oracle PremiumCare Online Portal.
3. Enter your user name and password in the box that appears, then click Login.
4. Under Oracle Support Tools, click Case Manager.
5. In the next screen, click Find Answers to gain access to the Knowledge Base.

1 Capacity Planning

Capacity planning consists of selecting the right equipment for a NetPoint deployment based on anticipated usage. For most deployments, NetPoint functions well using standard off-the-shelf equipment.

When planning for a NetPoint installation, you want to be sure that your equipment is adequate for handling peak loads.

This chapter provides information on a medium-sized deployment of 20,000 users and a large deployment of 100,000-2,000,000 users.

This chapter includes the following topics:

For an overview of NetPoint, see the *Introduction to NetPoint 7.0* manual.

About Capacity Planning

Capacity planning is the process of determining your hardware and memory requirements. The goal of capacity planning is to maintain good system performance during times of peak load. There are no rules when it comes to capacity planning. A quote from an anonymous author on the Web states: *The reasonable answer to any performance question begins with* "*it depends.*"

Having said that, it is important to use the right equipment for the task. In general, it is cost-effective to have more equipment than you need than it is to try to *make do* with inadequate hardware. The cost of extra hardware is low compared to the effort required to maintain an under-powered system. The guidelines provided in "Typical Machine Requirements" on page 19 are a good starting point for most environments. NetPoint components tend to function well on standard hardware such as the machines listed in this section. In fact, if you use the information in this section as the basis for your capacity planning, this may be entirely adequate for your environment.

If you want to take a more methodical approach to equipment planning, a general guideline for capacity planning is to estimate your peak load and to purchase equipment that is capable of handling peak loads. This chapter presents methods of estimating peak usage.

Note: You can spend a considerable amount of time calculating your requirements. However, following the recommendations for typical hardware configurations in this chapter may ultimately be the simplest and best choice.

A key point: Remember that it is hard to predict all of the variables in your network. As a result, most calculations of peak usage are error-prone and may ultimately be less reliable than following a few simple guidelines.

Estimating Peak Load

To ensure that you have adequate equipment for your environment, your machines need to be able to handle the maximum number of operations that can be expected in a particular time interval. The equipment that you use in your NetPoint installation should be able to accommodate your users during times of peak load.

The information in this section is divided into the following topics:

- General guidelines for estimating peak load

  These guidelines are somewhat reductionist, but they are reasonable in light of the fact that network performance tends to be unpredictable. If you can estimate your peak system throughput, you generally can predict what class of server you need.

- Specific methods

  These are more labor-intensive methods of predicting requirements by analyzing estimated system usage. These more complex methods, however, may or may not result in better overall predictions of hardware requirements.

General Guidelines for Estimating Peak Load

A simple method for estimating peak usage is as follows:

- Measure usage over at least a day, and preferably over several weeks.

  If usage tends to spike during particular weeks of the year, try to obtain measurements from the busiest weeks.

- Use the highest value seen in production.
- Estimate the what a typical busy load is.

  To estimate a typical busy load, multiply the value of an average heavy load by a small integer such as 2 or 3. This allows for usage patterns that are two or three standard deviations higher than an average heavy load, assuming a Gaussian distribution (bell curve) of loads.

If you have a multi-site deployment, you may want to create a chart of peak usage for all sites and estimate peak load based on total estimated usage across the sites. For instance, you can record the number of logged-in users at different times of the day. The following table illustrates this method of estimation:

Peak hours GMT	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Mexico	0	0	0	3	75	150	175	225	225	225	225	225	225	225	225	175
Spain	35	50	75	50	25	35	50	75	75	75	75	35	20	0	0	0
Egypt	45	45	50	50	50	50	50	55	55	45	30	10	0	0	0	0
U.S.	0	2	5	10	30	80	95	95	95	95	86	80	80	90	75	60
Colum-bia	0	2	7	15	30	45	45	50	50	50	55	55	55	55	45	35
Costa Rica	0	0	0	0	2	5	10	10	10	12	12	12	12	12	12	12
Indo-nesia	60	45	30	10	4	2	0	0	0	0	0	0	0	0	3	5
Taiwan	125	115	100	60	30	5	0	0	0	0	0	0	0	10	25	75
Total	265	269	267	198	372	425	425	510	510	502	483	417	392	392	385	362

If you can estimate the number of transactions per user during your peak hours, you have an estimate of your overall system capacity. You can compare your estimates of transactions-per-second to your equipment manufacturer's estimates for your hardware. Based on these comparisons, you can determine if the machines you already have are adequate for supporting the estimated load, and if not, you can base your equipment choices in part on your throughput requirements.

Note: Even this method of estimation may be more rigorous than is required for a deployment of fewer than 20,000 users. The hardware estimates provided in "Typical Machine Requirements" on page 19 are adequate for most deployments.

Complex Estimates of Peak Load

As an alternative to taking simple measurements of peak usage patterns, you can use an estimation method. This may be practical if there is no simple way to measure usage—for instance, if you have a geographically distributed environment. For a geographically distributed environment, you can identify the hours of peak usage and estimate the number of users who are present during those times. It may be more practical to make these estimates rather than trying to gather statistics on who is logged in at each office.

For instance, suppose your company has offices in a variety of countries. You can create a chart of regular office hours plotted against the GMT time, as illustrated below. The table below allows you to estimate the times (GMT) when the majority of your offices are busy, as indicated by the shaded area in the above table. This is a good indication of peak load hours. :

GMT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Mexico	19	20	21	22	23	24	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Spain	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1
Egypt	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2
U.S.	20	21	22	23	24	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Colum-bia	20	21	22	23	24	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Costa Rica	19	20	21	22	23	24	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Indo-nesia	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	6	7	8
Taiwan	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	6	7	8

Using Human Resources data on the number of users per office, you can estimate the total number of users accessing your resources during peak hours.

Country	Full time	Part time	Contract	Total
Mexico	3021	496	35	3552
Spain	755	356	5	1116
Egypt	329	275	0	604
U.S.	134	25	55	214
Columbia	1290	245	11	1546
Costa Rica	175	130	0	305

Again, you can create a simple estimate of throughput based on an estimate of the maximum number of users that you expect to be logged in at the same time, the estimated number of transactions per user for a given time period, and ultimately the estimated transactions-per-second that need to be supported. You can compare your transactions-per-second estimates with claims made by your hardware vendors. Estimates for the class of machines that you require are also provided in "Typical Machine Requirements" on page 19.

Predictions of Concurrency and Throughput

You can calculate peak requirements by estimating either concurrency or throughput. NetPoint is a stateless system. As a result, a discussion of concurrency is not entirely meaningful and estimates of throughput are more useful. For example, a user can log in but let their account remain idle until they time out. This user is accessing the account concurrently with other users, but their effect on throughput may be minimal.

However, you may want to use estimates of concurrency as a partial method of predicting system requirements. Based on an estimate of concurrency and an estimate of average number of transactions per user per minute, you can estimate system load.

For example, the Response-Time Law states that the number of requests per second (X_o) is equal to the number of concurrently logged-in users divided by response time and think time:

$$X_o = N/R+Z$$

For example, for a community of 5,000 users with an estimated response time of 3 seconds and an estimated user wait or think time of 2 seconds, the estimated requests per second would be 1,000.

Little's Law states that the number of simultaneous connections is equal to the throughput times the response time. That is:

N (concurrency) = X_o (requests per second) * R_{site} (response time for the web site)

Based on a network response time of 1 second, the Web site time would be 2 seconds (3-1), and the estimated concurrency would be 1,000 * 2 or 2,000 users. Assuming this is an estimate of concurrency during average load, you would want to multiply this concurrency by 2 or 3, for a peak load estimate of 6,000 users. This can be the basis for estimating your hardware requirements. Typical machine capacities are discussed in "Typical Machine Requirements" on page 19.

Note: Little's Law is a simple equation to calculate. However, it may also be relatively simple to base the calculation on an incorrect assumption.

You can also use information about the number of simultaneously logged in users to construct a poisson distribution of the estimated number of concurrent users. A poisson distribution is often used as a model for the number of events, such as the number of telephone calls at a business or the number of accidents at an intersection, in a specific time period.

The following are input parameters for a poisson distribution of concurrent users:

- The total number of users who may be logged in during peak hours.
- The time period for critical usage.
- The estimated duration of a user session.

Information on calculating a poisson distribution is readily available on the Web and in textbooks and will not be covered further in this chapter.

Note: Keep in mind that although a complex analysis may help you feel more confident about system requirements estimates, there may be a number of unpredictable elements on your network that interfere with the accuracy of your estimates. As a result, using the simpler guidelines and recommendations in this chapter may, in fact, be the most efficient method for estimating peak usage.

Typical Machine Requirements

When planning for a deployment of up to 20,000 users, it is safe to assume that NetPoint works well with standard equipment. The main rule of thumb when selecting hardware is to be sure you have enough. Using dedicated machines is preferable to using shared equipment, and server-class machines are preferable to desktop units.

The following estimates are for deployments of 20,000 - 2,000,000 users.

Note: Oblix recommends that you deploy NetPoint using adequate hardware, including memory and disk space. It can be a false economy to install a NetPoint component on a machine that is not optimal. The cost of hardware is sufficiently low to offset any cost savings from using inadequate hardware.

Access Server and COREid Server Requirements

For medium-sized deployments of up to 20,000 users, the following equipment is probably adequate for Access Servers and COREid Servers:

- Any supported server-class machine and operating system for the Access Server.

  The *NetPoint 7.0 Installation Guide* provides a complete list of supported platforms. Failover requirements will double the number of machines that you require. Use a minimum of two Access Servers for redundancy.

- Any supported server-class machine and operating system for the COREid Server.

  Failover requirements will double the number of machines that you require. Use a minimum of two COREid Servers for redundancy.

- 2-4 GB of memory for each server.

  2 GB of memory is a minimum.

- Use two CPUs per machine.

- For the administration console, a single dedicated Web server could be deployed in one of the two main COREid and Access Servers so that no additional hardware is required.

For deployments of 25,0000-100,000 users, each server should have 5 GB of disk storage.

For deployments of 100,000-2,000,000 users, the following is recommended:

- The Access Server should have 2 x 17 GB of disk storage
- For each Access Server a 4 x 400 MHz CPU is recommended.

- For delegated identity management, Oblix recommends separate COREid Servers.
- Server size mainly depends on the number of concurrent users.

  On an E420R class machine, a 4 x 400 MHz CPU with 4 GB of memory, and 2 x 17 GB of mirrored storage is recommended for the COREid Server. For a deployment of 17 million users, 64 GB of memory has proven to be adequate in lab tests.

Note: On Wintel machines, use of more than 2 CPUs does not seem to improve performance.

NetPoint COREid and Access servers use "SmartHeap for SMP" for fast memory management. SmartHeap retains unused memory for a longer time period, assuming NetPoint servers may reuse it in the near future. Memory is infrequently returned back to the operating system in large chunks. While this method is faster and more efficient, it can lead to confusion when looking at memory consumption of NetPoint servers. Please refer to the following sites for more technical details:

SmartHeap for SMP—http://www.microquill.com/smartheapsmp/index.html

SmartHeap FAQ—http://www.microquill.com/kb/faq_qxs.htm

Directory Server Requirements

For medium-sized deployments of up to 20,000 users, plan for the following for your directory server:

- Any supported server-class machine with 2 GB of memory is probably adequate for the directory server.

  Most directory servers currently on the market are likely to be adequate for medium-sized deployments.
- Use disks that are RAID 01 (striped and mirrored) for the directory server.
- Use two replicated directory servers for failover.
- Have enough memory to hold two times the LDAP data in memory.
- The LDAP cache should be large enough to hold the entire database.

  For optimum performance, a copy of the data should be stored in the LDAP cache.

For deployments of 25,000-100,000 users, each directory server should have 5 GB of disk storage.

For deployments of 100,000-2,000,000 users the following is recommended:

- 2 x 17 GB of disk storage.

- CPUs of 4 x 400 MHz.

  As a rough estimate, a 400 MHz CPU on any recent-model directory server can handle 250 logins per second, so that four CPUs should handle 1,000 logins per second. Note that with a potential user base of 2,000,000 users, you may wish to increase the footprint of the server to 8 GB of RAM and 4 x 400 MHZ CPU.

- Allocate at least 2 GB of RAM for a 100,000 user directory and at least 4 GB for a very large directory.

- A JBOD usually provides the same or higher performance as a high end disk array unless the disk array has enough cache (battery backed up and mirrored) to hold the database.

WebGate Web Server Requirements

The number of WebGate Web servers required depends on usage patterns. Web servers that already exist in your environment can be used to host WebGate. However, this depends on the amount of dynamic versus static content generation and retrieval. As a rough rule of thumb, you can expect a 10% to 20% degradation with a WebGate on the Web server.

Note: This degradation estimate means that you need to plan for one additional Web server for every five existing Web servers that will be hosting WebGate.

Other Requirements

Plan for the following:

- Using lab tests that simulate realistic usage in an Active Directory environment, controllers with 64 GB of memory have been measured to be sufficient to hold up to 17 million users in memory.

- All components should be redundant for load balancing and failover.

- As a worst case scenario, an approximate ratio of 1:1:5 for directory servers:COREid or Access Servers:Web servers will suffice.

  In most cases, you can double or triple the number of Web servers that can be protected. The estimate of 1:1:5 assumes that the Web servers are handling a very high volume of authorizations and authentications and are serving very little content. Actual site usage and performance testing is needed to determine your requirements.

Geographically Distributed Environments

If you have a geographically distributed environment, the estimates provided in this chapter can serve the requirements for each geographical area. Additional Access Servers and Identity Servers can be deployed locally, along with the corresponding WebGates and WebPass instances, which should be configured to point at the local servers as primaries and the global servers as secondaries.

You can also implement a geographically sensitive failover scheme, where for a particular location the servers can be secondary servers for a geographically close location. A close location should be defined by bandwidth more than physical distance.

Directory Performance

NetPoint performance is closely tied to the performance of the LDAP server. Be aware of performance differences among classes of directory servers. In addition, you should estimate directory requirements based on the amount of space being consumed in the directory and added to the network load.

In terms of network load, directory entries generally average around 1 KB. Every directory lookup adds about 1 KB to the network load. If directory users perform approximately ten directory lookups a day, for every directory user you will see an increased network load of around 10,000 bytes per day. This is a conservative estimate.

Typically, the network load is not an issue. However, if you have a heavily loaded or unreliable LAN, you may need to replicate the directory tree to a local server.

Capacity Planning for Multiple Environments

A standard NetPoint deployment consists of several instances of NetPoint installed for different purposes:

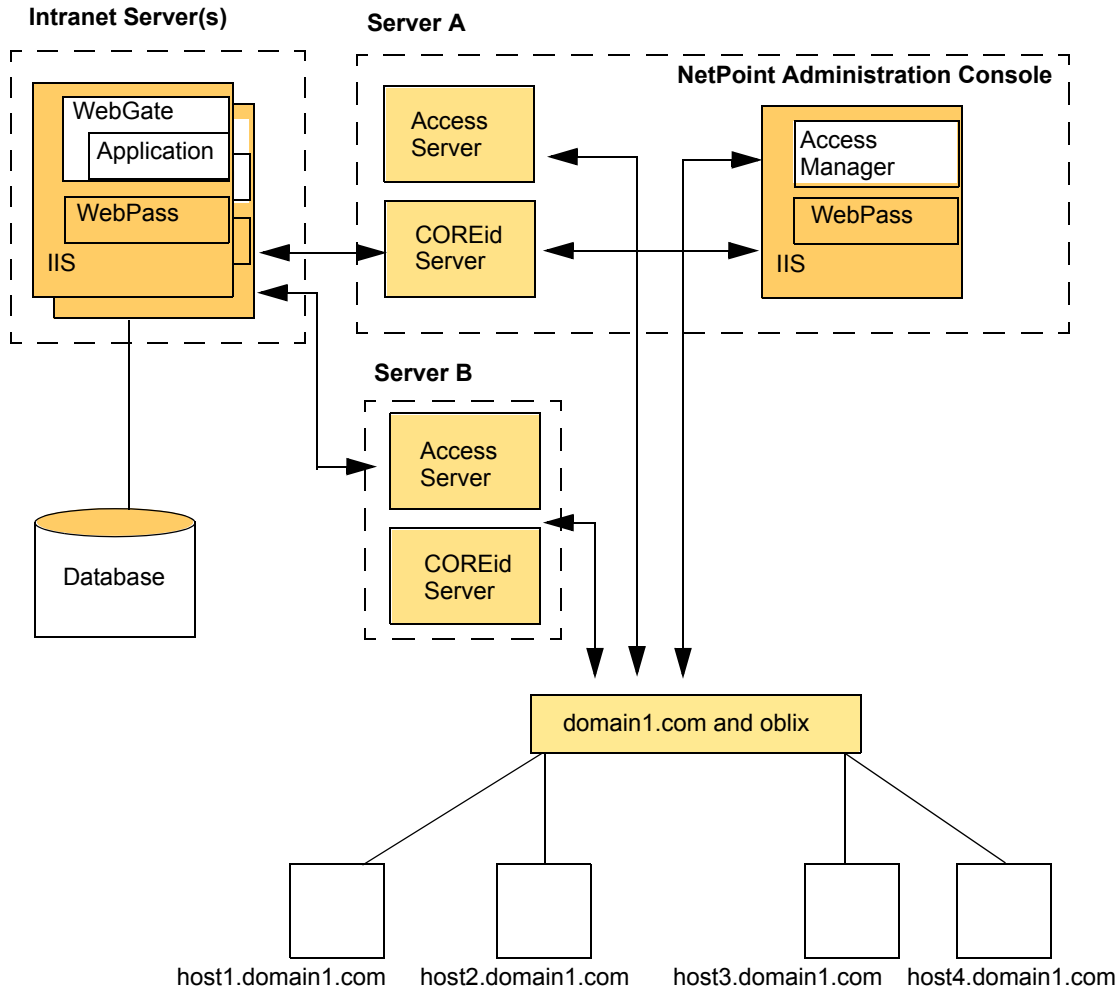
Development and Test—This is an area where you configure and test a fully deployable system. On a development or test server, a smaller amount of RAM may be required. For development and test systems, you can run the Web server on the same system as the COREid Server, assuming that only a small development team is using the machines rather than the whole user population.

Staging—This is an area where new application rollouts, software upgrades, and performance benchmarking can be performed without affecting your production and development environments. This environment can closely resemble the production environment, though it may be a scaled down image of it. For instance, a smaller amount of RAM may be used.

Production—This is an area where your end users can access the deployed system. Machine estimates provided in “Typical Machine Requirements” on page 19 assume a production environment. Oblix recommends you use a replicated directory and NetPoint’s native load-balancing features. You may also want to configure the COREid Servers for failover and load balancing and have separate machines for your Web servers. As an alternative, you can install multiple Web server instances on a single machine.

Example from an Actual Deployment

The following diagram illustrates hardware and software choices in an actual mid-sized (10,000-20,000 user) deployment. Using a poisson distribution, the estimated concurrency was predicted to be 74 users, with a 0.01% probability of having 100 or more users.:



For the production environment:

- The servers are running Windows and Active Directory, IIS, and NetPoint.
- Each server has two CPUs and 2 GB of RAM, and about 10 GB of disk space.
- There are two COREid Servers running on different machines.

- Multiple WebPass instances are installed in several intranet IIS servers, with at least one server acting as the administration console for the COREid System.
- There are two Access Servers running on different machines.
- There is one WebGate installed on each IIS intranet server.
- The administrative console for the Access System is installed on an IIS server on one of the two Access and COREid Server boxes.

This environment could be scaled as follows:

- The number of Access and COREid Servers can be increased to spread the load and offer higher redundancy.
- The number of CPUs, memory, and disk space can be increased to improve server performance.
- The Active Directory environment supports the addition of new servers for cases where this is desired to improve performance and reliability.

2 Performance Tuning

Once you have installed and configured NetPoint, you want to be sure to get the best possible performance out of the product.

This chapter provides information for experienced NetPoint administrators on how to optimize product performance. This chapter includes the following topics:

Note: For details about Access Manager tuning factors for Apache Web servers, see the *NetPoint 7.0 Installation Guide*.

Guidelines for Directory Tuning

NetPoint stores its data in an LDAP directory. Many performance issues can be resolved by avoiding trips to the directory, especially for write operations.

The following sections discuss how to minimize trips to the directory.

Storing Workflow Tickets in the Directory

As described in the *NetPoint 7.0 Administration Guide Volume 1*, by default each workflow step generates a ticket and NetPoint writes the ticket to the directory.

One way to avoid unnecessary directory write operations is to minimize the number of steps in a workflow.

Another way to avoid unnecessary directory write operations is to change NetPoint's default behavior of creating tickets for every step in a workflow.

Tickets are of little value when:

- The information in a step has little value for the next step.
- A participant triggers the workflow but there are no participants for other steps.

Workflow Example

Suppose you create a workflow consisting of the following steps:

1. **Initiate**—The user initiates a self-registration.
2. **External action**—The request is passed to an external process.
3. **Enable**—The workflow is enabled.

If no one participates in steps 2 and 3, the workflow may not require a ticket.

Writing Workflow Tickets to the Directory

The WFInstanceNotRequired flag determines whether every workflow step generates a ticket. This flag is set to false by default, which means all workflow instances are written to the directory. When set to true, workflow instances are only written to the directory server when:

- A user action is required.
- Any errors are encountered (for example, the commit action fails).
- Any subflows are triggered.

Setting the WFInstanceNotRequired flag to true reduces the directory size and speeds up the workflow process. The disadvantage of setting this flag to true is that, because the instances are not being stored, you will not able to see all of the workflows that were executed from NetPoint's Workflow Monitor.

To avoid create tickets for every workflow step

1. Set the WFInstanceNotRequired flag to true in the following file:

   COREid_install_dir/identity/oblix/data/common/workflowdbparams.xml

2. Restart the COREid Server.

Indexing Attributes in the Directory

Indexing improves search performance in the directory, although at the cost of slower database modification and creation operations and disk space. NetPoint automatically indexes key NetPoint attributes when you run the setup program. The index files are specific to your directory server type, and are stored in:

COREid_install_dir/identity/oblix/data/common

where *COREid_install_dir* is the directory where COREid is installed.

Index types include the following:

- An equality index improves searches for directory entries that contain a specific attribute value.
- A presence index improves searches for directory entries that contain a specific attribute.
- A substring index improves searches for entries that contain specific text strings.

You can enhance performance if you index attributes that are read during various operations. These include:

- Attributes used in searches that are triggered indirectly by user interaction.
- Attributes used in searches that are explicitly invoked by users.
- Attributes used in mapping filters for authentication schemes.
- Attributes in filters for dynamic member definitions in Group Manager.

Your directory documentation describes how to index attributes.

NetPoint provides an index file fragment showing how NetPoint-specific attributes can be indexed for each supported directory server. For example, the Oblix attributes that can be indexed to improve performance in a directory managed by Sun (formerly iPlanet) directory server are listed in the file:

COREid_install_dir/identity/oblix/data/common/
iPlanet5_oblix_index_add.ldif

An example of an index entry in this file is:

```
index obclass pres,eq,sub
```

This means that the attribute obclass can be indexed for presence (`pres`), equality (`eq`), or substring (`sub`).

Note: The iPlanet5_oblix_index_add.ldif file needs to be loaded to the directory server using ldapmodify.

Limitations of Indexing

You should carefully weigh the benefits of indexing directory attributes for search operations against the performance hit incurred when these attributes are modified. When an attribute value is modified, indexes for that attribute may need to be rebuilt by the directory server. However, this is accomplished varies by directory server.

You should read the manufacturer's guidelines for indexing. Avoid over-indexing attributes.

Indexing and User Deactivation

After you deactivate a significant number of users, COREid System performance can start to degrade. If you experience this problem but want to maintain all deactivated users in the directory, use an equality index for the following attributes:

- ObIndirectManager
- ObModifyAccessUID
- ObViewAccessUID
- ObNotifyUID
- ObAccessUID
- ObUniqueMember

An equality index allows you to search efficiently for entries containing a specific attribute value.

Indexing and Workflows

If it takes a long time to delete a workflow, index any attributes that the system checks during a delete operation. Performance issues with the delete workflow function usually result from attributes that need to be indexed.

Index the following attributes using the types of equality, presence, and substring:

- ObWorkflowID
- ObContainerID
- ObWFStepID
- ObWFTargetID
- ObIsWorkflowProvisioned
- ObLocationUID
- OblixGID

- Manager
- Secretary

An equality index type improves searches for directory entries that contain a specific attribute value. A presence index type improves searches for entries that contain a specific attribute. A substring index type improves searches for entries that contain specific strings.

For example, if you search for an attribute with a substring index type as follows:

```
cn=*lane
```

The search would match common names containing these strings:

```
John Lane
Jane Tulane
```

If you conducted this search:

```
telephonenumber= *123*
```

The search would return all entries with telephone numbers that contain 123.

Indexing and Groups

The following attributes are used for group expansion and could be indexed to improve performance:

- Any attribute configured with the obSDynamicMember semantic type
- The obGroupExpandedDynamic attribute of the oblixAdvancedGroup object class
- All user attributes used in the dynamic filters of the groups to be expanded.

Indexing and Search Constraints

You can limit the performance impact of searches by forcing users to use a minimum number of characters for a search. This is controlled through the searchStringMinimumLength parameter. It is set to 3 by default because searching for fewer than 3 characters typically forces the directory not to use its indexes.

To set a minimum number of search characters

1. Locate seachStringMinimumLength in:

   COREid_install_dir/identity/oblix/apps/common/bin/oblixappparams.xml

2. Set its value to the minimum number of characters for a search.

   For instance, if you set it to 6, users could not search for “Smith” because it has only 5 characters. The constraint only applies to the longest search string supplied by the user. For example, if the search is on both surname and job title, and the user enters “manager” for job title, the longest string (“manager”) has 7 characters, the search is allowed. A value of 0 turns off checking.

The searchStringMinimumLength constraint is enforced for searches that are conducted using the NetPoint user interface and for clients using other interfaces to NetPoint (for example, Identity XML clients).

To enforce a per-attribute minimum number of characters

1. Set the searchStringMinimumLength to 0 in the catalog.
2. In the JavaScript code, find the function validateSearchAndSubmit() in the file misc.js.
3. Modify the JavaScript code to handle the per-attribute checking you require.

Note: This technique does not enforce the constraint on users of Identity XML clients.

As with other parameters, you can set searchStringMinimumLength to one value in the global oblixappparams.xml catalog, and to a different value in one or more of the application-specific catalogs. For example, to override the global value and set searchStringMinimumLength to a different value for User Manager, specify the parameter and its value for User Manager only in:

COREid_install_dir/identity/oblix/apps/userservcenter/bin/userservcenterparams.xml

Changing the Number of Access Server-to-Directory Server Connections

By default, each Access Server opens only one connection to a primary directory server and one connection to a failover directory server (if failover is configured). This may not be optimal for systems with heavy loads.

To configure additional Access Server-to-directory server connections for NetPoint configuration and policy data, use the command line tool configureAAAServer. The *NetPoint 7.0 Administration Guide Volume 2* provides information on this tool. In environments with very high loads (perhaps 1,000 hits/second on the Access Server), creating twenty connections has significantly improved performance.

Note: If your directory is running on a low-powered system, the directory itself may be the limiting factor. In this case, increasing the number of connections may have little benefit. If the machine running the directory is the problem, increase the number of directory server instances and configure load balancing among them.

Deleting and Archiving Workflows

To prevent the Oblix configuration tree from getting too large, periodically delete or archive workflows. Deleting a workflow instance removes the directory entries associated with that instance. Archiving a workflow instance keeps a record of the instance in an LDIF file and deletes the instance from the directory.

The frequency for archiving or deleting depends on how frequently your workflows are used.

Archived workflows are stored in LDIF format. The default archive file is:

COREid_install_dir/identity/oblix/data/common/wfinstance.ldif

Multiple archive operations add information to this file. You can change the archive file by changing entries in the following files:

User Manager—*COREid_install_dir*/identity/oblix/apps/userservcenter/bin/usc_wf_params.xml

Group Manager—*COREid_install_dir*/identity/oblix/apps/groupservcenter/gsc_wf_params.xml

Org Manager—*COREid_install_dir*/identity/oblix/apps/objservcenter/osc_wf_params.xml

The entry to change is similar to the following:

```
<NameValPair ParamName="archiveFileName" Value="wfinstance.ldif"/>
```

To delete or archive a workflow

1. In User, Group, or Organization Manager, click Requests.
2. Click Monitor Requests.

   For subflows, if the first step has not been processed, the Date Processed field will be empty.
3. In the Search fields, select your search criteria
4. Click Go.

   The results appear below the search fields.
5. Click individual check boxes or the Select All button.
6. Click Next or Previous as necessary to see other results.
7. Click the Delete or the Archive button.

Setting Read and Write Permissions for Administrators

By default, administrators can view all attributes in the directory so that they can perform initial setup of NetPoint. Because end users only have read permissions for specific attributes, there is a difference in performance for an administrator and an end user.

The parameter BypassAccessControlForDirAdmin controls whether the NetPoint Master Identity Administrator or a delegated administrator is permitted to view all attributes in the directory. By default, the value of the parameter BypassAccessControlForDirAdmin is set to true. You can set the BypassAccessControlForDirAdmin parameter to false in the following file:

COREid_install_dir/identity/oblix/apps/common/bin/globalparams.xml

If the BypassAccessControlForDirAdmin flag is set to false, performance is the same for non-administrative and administrative users. This is because attribute read and write permissions are applied to the administrative users.

The following is an example of the parameter entry:

```
<SimpleList>
   <NameValPair
      ParamName="BypassAccessControlForDirAdmin"
      Value="true">
   </NameValPair>
</SimpleList>
```

Configuring the Searchbase

Searchbase configuration can affect NetPoint performance. Guidelines for searchbase configuration include:

- Set the searchbase for the user object class at a point in the people branch where all users can be found, for example, the root.
- Use the default searchbase filter objectclass=* whenever possible.
- Instead of defining searchbase filters, configure read and write permissions for attributes.
- Configure workflow rules and roles to control the user's ability to view and modify attribute values.

Setting a Searchbase Filter

When configuring a searchbase, if you add a filter and enter a filter other than the default (objectclass=*), the Resource Filter Search scope takes effect. The Resource Filter Search scope has no effect unless you define a filter.

The default searchbase filter (objectclass=*) instructs NetPoint to apply the user's search criteria to the entire section of the directory tree that has been selected as the searchbase. If you specify other criteria, for example, (telephone=408*), NetPoint retrieves all users who satisfy that criteria, then applies your search criteria to the subset specified by the filter instead of to the entire tree.

This can degrade performance, especially if the filter retrieves a large number of entries. For example, if you specify (objectclass=wwmOrgPerson) in the filter, NetPoint may recover all users (assuming all users satisfy this filter) under the specified tree before applying your search criteria. This can seriously degrade performance. You do not have to specify (objectclass=wwmOrgPerson) because the searchbase is already set for that object class. In general, setting read and write privileges for *attributes* is a better way of controlling user access than setting a searchbase filter. To optimize directory performance, avoid defining complex filters for the searchbase. In the case of searchbases for a tab, only objects of the class which that tab applies to are searched. There is no need to mention (objectclass=*) explicitly.

If you must enter a filter, you can limit the performance impact by setting the resourceSearchFilter scope parameter to 1.

Applying Search Constraints

The larger the number of entries that a search actually or potentially returns, the longer the search takes. For an interactive application such as NetPoint COREid System, large result sets may be unmanageable for the end user.

Your directory may allow you to limit the time that the directory server spends on a search, the size of the result, or both.

Increasing Connections to the Directory in the COREid System

The LDAP Database Instance Maximum Connections is the maximum number of connections allowed from the COREid Server to the directory servers. This value defaults to 1, but you may see a performance improvement by setting this value to more than 1. (With an SQL profile, the default is 5.)

There is no optimal value for the maximum connections. There are variables to consider such as directory configuration and hardware. You may want to consider setting the maximum directory connections no higher than the number of threads set for a COREid Server. For example, if the COREid Server is configured with only 20 threads, there is no benefit in having more than 20 connections because no COREid worker threads can take advantage of the additional connections.

You may want to increase the maximum connections by increments of 5, and monitor your system performance. If performance is worse, decrease the number of connections. The Initial Connections and the Maximum Connections settings work together. When a COREid server starts, it opens the number of connections to the directory as specified in the Directory Profile Initial Connections. Those connections are then pooled and used by the COREid Server.

Note: Additional connections can introduce overhead that in turn can hurt performance. For example, if you restart the directory server, the COREid Server has to reconnect the configured number of connections. Or if the directory server is down, the COREid Server has to try each of its configured connections before recognizing that the directory server is down.

Changing Directory Content

This section describes modifications that can be made to the directory content to affect NetPoint operation. For a discussion of the tools needed to make these changes, see “LDAP Tools” on page 41.

Ordering the Columns in a Search Results List

A search of the Access Manager for policy domain names or policy names returns a default set of columns in a default order. You can display different columns or change the order of columns. While this does not affect performance in terms of response times, it may improve your satisfaction with the results returned from a search.

To modify results for a policy or policy domain names search

1. Locate the DN, for example:

   `obname=SDSearchColumnList, obapp=WRSC, o=Oblix, o=Company, c=US2`

2. Under this DN, find the current column list.

   The column list consists of the values for obsearchresultcolumns.

   Possible values for a search of policy names are:

   `WRORname`—Policy Name

   `AuthentPolicyName`—Authentication Rule Name

   `AuthorPolicyName`—Authorization Rule Name

   `AdminPolicyName`—Auditing Rule Name

   `URLPrefix`—URL for the domain controlled by the policy

   Possible values for a search of *policy domain* names are:

   `SDName`—Policy Domain Name

   `AuthentPolicyName`—Authentication Rule Name

   `AuthorPolicyName`—Authorization Rule Name

   `AdminPolicyName`—Auditing Rule Name

   `AbsPathPattern`—Path to the controlled domain

For example, the following is an LDIF extract that shows the policy name, authentication rule name, authorization rule name, and URL for the domain controlled by the policy:

```
dn: obname=SDSearchColumnList, obapp=WRSC, o=Oblix,
o=Company, c=US
  objectclass: top
  objectclass: OblixWRSSearchResultColumns
```

```
obname: SDSearchColumnList
obSearchResultColumns: WRORName
obSearchResultColumns: AuthentPolicyName
obSearchResultColumns: AuthorPolicyName
obSearchResultColumns: URLPrefix
```

To change the order or content of the displayed search results, modify this information and store it back to the directory.

Changing the Bind DN

Using Directory Manager as the bind DN bypasses search limits such as *look through limit*, *size limit*, and *time limit*. Large searches can tie up the directory server and increase the amount of processing that is required by the COREid Server.

You can create another user to use as a bind DN.

Note: This procedure illustrates the technique for Sun (formerly Netscape and iPlanet) directories. Consult your directory server's administration manual for instructions on how to create a directory user with the appropriate permissions.

Create your new user, for example oblixadmin. Use the LDAP directory to give the user permissions to act as an administrator for NetPoint. For the searchbase, configuration base, and all branches underneath them, this user needs the following permissions:

- Read
- Write
- Add
- Delete
- Search
- Compare
- Selfwrite

To change bind DN permissions

1. From the Sun LDAP Server Admin Console, navigate to the directory server instance and open it.
2. Choose the Directory tab, then locate and right-click the branch for which you want to set permissions.

   For example:

   for o=Company, c=US, you would find the Company node and right-click to get a menu.
3. Choose Set Access Permissions in the menu.
4. Add a new access permission, or edit the one already in place.

   There should be two lines when you are done. The first is a default deny for everyone, the second is the allow statement.
5. Choose Allow, then search users and groups to find the new administrative user.
6. Set the rights for this user.

If you make this change after installing the COREid System, change the bind DN for the COREid Server to match the value you created in the directory.

Adjusting Cache Settings

Caching avoids the latency of unnecessary lookups or calculations. If they keep the results of recent requests close to the consumer, producers can respond quickly by returning cached results rather than recalculating them. The cost of a cache is usually extra RAM or disk space.

A directory that manages large entries will likely hit a *maximum cache size limit* first, while a directory of small entries might hit the *maximum number of entries limit* first. If you only care about one of these two criteria, set the other to an unrealistically large number, or match their limits by setting the maximum cache size first, then dividing that number by the size of each entry to get the number you should use for maximum entries in cache.

Deleting ObSyncRecord Entries from the Directory

Every time a change is made to a user entry in the COREid System, NetPoint creates a directory entry called ObSyncRecord. This directory entry enables the user cache for different COREid and Access Servers to stay in sync. If many updates are made to user entries over a short time interval, the number of ObSyncRecord entries that are written can cause a directory performance issue.

You may want to delete ObSyncRecord entries at a regular interval. If you do this, you may want to check each entry before deleting it to ensure that the entry has been in the directory for a time period that is greater than the cache-flush interval.

Database Connection Pool Size

If you are running a large number of time-consuming searches, you can increase database connection pool size to improve performance for user data profiles.

To increase the connection pool size

1. Navigate to COREid System Console > System Admin > System Configuration > Configure Directory Options.
2. Click the name of a user data directory server instance in the Configure LDAP Directory Server Profiles list on the Configure Profiles page.
3. Click the name of a directory server instance in the Database Instances list of the Modify Directory Server Profiles page.
4. On the Modify Database Instance page, modify the two parameters below:
   - **Maximum Connections**—Increases the number of available connections in the pool. The default value is 1. No upper limit is enforced; you can experiment with this setting to find a suitable value.
   - **Initial Connections**—Specifies how many connections are opened at database initialization. The default value is 1. There is no upper limit.

The directory server starts by opening the number of connections specified in Initial Connections. As needed, more connections are opened until the maximum number of connections specified in Maximum Connections is reached. Once opened, connections remain open until the COREid Server shuts down or the directory server stops responding.

LDAP Tools

Directory applications use Lightweight Directory Access Protocol (LDAP) as a standard tool to create, modify, and report data stored in the directory. Tools are available to allow easy manipulation of this data. This section introduces these tools. More detail is available from the manufacturer of your server application.

Viewing Directory Content in LDIF Files

The structure of a directory, and the data contained within it, is represented by the content of an LDAP Data Interchange Format (LDIF) file. The file can be *output*, the formatted result of a request made to the directory by an LDAP reporting tool, such as LDAPSEARCH. It can also be *input*, data that is intended for insertion to the directory, either as entirely new data, or as an update to existing data, using an updating tool such as LDAPMODIFY.

The following is an example, part of an LDIF file taken from a NetPoint Demo Directory:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: companyOrgPerson
cn: John Kramer
sn: Kramer
telephonenumber: 415-555-5269
facsimiletelephonenumber: 415-333-1005
title: Account Manager
departmentnumber: 1204
employeetype: Fulltime
employeenumber: 521-321-4560
givenname: John
.
.
Reporting Directory Content with LDAPSEARCH
```

LDAPSEARCH is one possible tool that can be used to report directory content. There are others, which use a different syntax, but the concepts are the same.

LDAPSEARCH can be used in either a command line or interactive mode. The command line approach is preferable, as it allows users to provide the text of the report request by means of a input file. It is easy to verify the content of this file before making the request. You can correct errors by changing a few characters in the file rather than retyping the full request, which would be necessary in the interactive mode.

LDAPSEARCH Command-Line Format

The command-line format for LDAPSEARCH is:

`ldapsearch <params> <filter> <attr_list>`

Note: Each of the three categories shown between <> is optional; if all are omitted, LDAPSEARCH drops into interactive mode, which is not discussed here.

The categories are as follows:

- **<params>**—*Parameters* tell LDAPSEARCH how to operate. One of them, `-f`, is used to specify a filter file. If instead the search filter is provided on the command line, all parameters must be stated before the filter is stated.
- **<filter>**—The *filter* instructs LDAPSEARCH to provide a subset of the data that would otherwise be provided. For example, a filter could require that only names beginning with N be reported. A filter provided on the command line must be enclosed in quotes.
- **<attr_list>**—The *attribute list*, if included on the command line, overrides the default attribute listing. The default list shows all attributes belonging to the directory entry, except operational attributes. If you wish to see only some of these attributes listed, provide their names in the command line, following the filter, and separated by spaces. If you want to see operational attributes, provide their names in the command line. If you follow the operational attributes with a `*` you get the default list of attributes as well.

LDAPSEARCH Command-Line Parameters

Parameters are always provided in the form:

`-p pdata`

where *p* is the parameter, preceded by a dash, and *pdata* is the information (data) required for the parameter, if any. If the data contains one or more spaces, it must be enclosed in double quotes:

`-p "pdata with spaces"`

Following is a list of commonly used parameters. See the reference document for your version of LDAPSEARCH, or use the parameter `/?` to see them listed.

`-A`	Retrieve the attribute names only, not the attribute values.
`-b`	Searchbase, the starting point for the search. The value specified here must be a distinguished name that is in the directory. Data provided for this parameter MUST be in double quotation marks. For example: `-b "cn=Barbara Jensen, ou=Development, o=Oblix.com"`

`-D`	Distinguished name of the server administrator, or other user authorized to search for entries. This parameter is optional if anonymous access is supported by your server. For example: `-D "uid=j.smith, o=Oblix.com"`
`-f`	Specifies the file containing the search filters to be used in the search. For example: `-f filterfile`
`-h`	Hostname or IP address of the machine on which the directory server is installed. This entry is optional; if no hostname is provided, LDAPSEARCH uses the localhost. For example: `-h myserver.com`
`-H`	This generates a list of all possible LDAPSEARCH parameters.
`-p`	Port number that the directory server listens at. For example: `-p 1049`
`-s`	Scope of the search. The data provided for the scope must be one of the following:
`base`	Search only the entry specified in the -b option.
`one`	Search only the immediate children of the entry specified in the -b parameter; do not search the actual entry specified in the -b parameter.
`sub`	Search the entry specified in the -b parameter, and all of its descendants. That is, perform a subtree search starting at the point identified in the -b parameter. This is the default, if the -s parameter is not used.
`-S`	Designates the attributes to use as sort criteria, controlling the order in which the results are displayed. You can use multiple -S arguments if you want to sort on multiple criteria. The default behavior is not to sort the returned entries. In the following example, the search results are sorted first by surname and then, within surname, by first name: `-S sn -S firstname`
`-w`	Password associated with the distinguished name that is specified in the -D option. If you do not specify this parameter, anonymous access is used. For example: `-w password`

`-x`	Specifies that the search results are sorted on the server rather than on the client. This is useful if you want to sort according to a matching rule, as with an international search. In general, it is faster to sort on the server than on the client.
`-z`	Specifies the maximum number of entries to return in response to a search request. For example: `-z 1000`

LDAPSEARCH Examples

To get the surname (sn), common name (cn), and given name for every employee named John in the Sales organization, from the directory server listening at port 392, entirely from the command line, you could provide the following information:

```
ldapsearch -p 392 -b “ou=sales, o=company, c=US” -s sub
“givenname=John” sn cn givenname
```

Results could be something like:

```
dn: cn=John Jackson, ou=Sales, o=Company, c=US
sn: Jackson
cn: John Jackson
givenname: John
dn: cn=John Kramer, ou=Sales, o=Company, c=US
sn: Kramer
cn: John Kramer
givenname: John
```

You can get the same results by using a filter file. For example, a file called namejohn containing the filter:

```
givenname=John
```

can be used, with the following command line:

```
ldapsearch -p 392 -b “ou=sales, o=company, c=US” -s sub -f namejohn
sn cn givenname
```

Changing Directory Content with LDAPMODIFY

The LDAPMODIFY tool changes or adds directory content. There are other tools, but the concepts are similar. LDAPMODIFY opens a connection to the specified server, using the distinguished name and password you supply, and modifies the entries based on LDIF update statements contained in a specified file. LDAPMODIFY can also be run in interactive mode, a method that is not discussed here.

If schema checking is active when you use LDAPMODIFY, the server performs schema checking for the entire entry before applying it to the directory. If the directory server detects an attribute or object class in the entry that is not known to the schema, then the entire modify operation fails. Also, if a value for a required attribute is not present, the modify operation fails. Failure occurs even if the value for the problem object class or attribute is not being modified.

Note: Keep schema checking turned on at all times to avoid accidently adding data to the directory that could be unusable or cause schema violations when schema checking is turned back on. Schema checking is controlled at the directory administration server console and is generally on by default.

LDAPMODIFY Command-Line Format

The command line format for LDAPMODIFY is:

`ldapmodify <params>`

Note: The params category is optional; if it is omitted, LDAPMODIFY drops into interactive mode, which is not discussed here.

`<params>` These *parameters* tell LDAPMODIFY how to operate. One of them, `-f`, can be used to specify a file describing modifications to be made to the directory.

LDAPMODIFY Command-Line Parameters

Parameters are always provided in the form:

`-p pdata`

where *p* is the parameter, preceded by a dash and followed by a space, and *pdata* is the information required for the parameter, if any. If the data contains one or more spaces, it must be completely enclosed in double quotes:

`-p "pdata with spaces"`

Following is a partial list of commonly used parameters. Use the parameter `/?` to see all of them.

`-a`	Allows you to add LDIF entries to the directory without requiring the `changetype:add` LDIF update statement, which is necessary in the interactive mode. This provides a simplified method of adding entries to the directory; in particular, this allows you to directly add a file created by LDAPSEARCH and modified to make changes.
`-c`	Forces the utility to run in continuous operation mode. Errors are reported, but the utility continues with modifications. Default is to quit after reporting an error.
`-D`	Distinguished name of the server administrator or other user authorized to change directory entries. This parameter is optional if anonymous access is supported by your server. For example: `-D "uid=j.smith, o=Oblix.com"`
`-f`	Provides the name of the file containing the LDIF update statements used to define the directory modifications. For example: `-f changestomake.txt`
`-h`	Hostname or IP address of the machine on which the directory server is installed. This entry is optional; if no hostname is provided, LDAPSEARCH uses the localhost. For example: `-h mozilla`
`-H`	Lists all possible LDAPMODIFY parameters.
`-p`	Port number that the directory server uses. For example: `-p 1049`
`-w`	Password associated with the distinguished name that is specified in the -D option. If you do not specify this parameter, anonymous access is used. For example: `-w password`

LDAPMODIFY Examples

Suppose you want to change the stored given name of John Kramer, as reported under the discussion of LDAPSEARCH. The data reported back was:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
sn: Kramer
cn: John Kramer
givenname: John
```

This output can be used to derive an input file, ToHarvey, whose content might be:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
changetype:modify
replace:givenname
givenname: Harvey
```

The command line would then be:

```
ldapmodify - p 392 -f ToHarvey
```

If you were to now search the directory with the command line:

```
ldapsearch -p 392 -b “ou=sales, o=company, c=US” -s sub
“givenname=Harvey” sn cn givenname
```

The response would be:

```
dn: cn=John Kramer, ou=Sales, o=Company, c=US
sn: Kramer
cn: John Kramer
givenname: Harvey
```

Access Server Performance Tuning

This section describes ways to tune the performance of the Access Server, including:

- Allowing the Access Server to validate user passwords, as described in “Configuring Password Validation by the Access Server” on page 48
- Changing the number of request threads and queues, as described in “Changing the Number of Request Queues and Threads” on page 49
- Using ObMyGroups, as described in “Performance Considerations when Using ObMyGroups” on page 50

Configuring Password Validation by the Access Server

By default, when NetPoint validates a user password as part of the validate_password plug-in, it passes the request to the user directory server. The directory server validates the password and returns the result to NetPoint. This operation is slow. In an environment with many authentications, it degrades Access Server performance.

You can control whether to use the Access Server or the directory server for password validation on a scheme-by-scheme basis.

Process overview: When using Access Server password validation

1. The first time a user's password is validated, the Access Server goes to the directory server for validation.

   If the password is valid, the Access Server caches an MD5 hash of the password. The Access Server never caches a clear text password.

2. The next time the user's password needs to be validated, the Access Server creates an MD5 hash of the supplied password and compares it to the hash of the cached password.

   - If the two hashes match, the user's password is considered valid.
   - If the two hashes don't match, the Access Server validates the password against the directory. If the directory validates the password, the Access Server hashes it and replaces the old hash in the cache.

The ObCredValidationByAS Parameter

To configure an authentication scheme so that the Access Server validates passwords, add the parameter ObCredValidationByAS with a value of true to the validate_password plug-in parameter list. To control the timeout of the cached password on a scheme-by-scheme basis, use the Time To Live parameter. The default value of the Time To Live parameter is 1800 seconds (30 minutes). To control the interval at which the Access Server goes to the directory for password validation, add the parameter obPwdHashTTL with a value equal to the number of seconds required.

The following is an example of the out-of-the-box validate_password plugin:

```
validate_password obCredentialPassword="password", obAnonUser="cn=anonymous,
o=Company, c=US"
```

The following is an example of validate_password configured to use Access Server password validation with the default Time To Live parameter:

```
validate_password obCredentialPassword="password", ,obCredValidationByAS="true',
obAnonUser="cn=anonymous,o=Company, c=US"
```

The following is an example of validate_password set to use the Access Server password validation with Time To Live set to 100 seconds:

```
validate_password obCredentialPassword="password", ,obCredValidationByAS="true"
, obAnonUser="cn=anonymous,o=Company, c=US", obPwdHashTTL="100"
```

Changing the Number of Request Queues and Threads

The Access Server uses request queues to create a sequence of incoming requests that are processed by worker threads. By default, there is a single request queue and 60 worker threads per queue. If you create two request queues and one hundred threads per queue, the Access System spawns a total of two hundred worker threads.

You can tune the performance of the Access Server using a parameter that controls the number of request queues used in the Access Server. If you change the number of request queues, you should also change the number of threads per queue as recommended below.

Estimating the Number of Threads and Queues

Having many queues with a relatively small number of threads per queue can cause problems if the requests come primarily on one queue. It may be helpful to configure more than the minimum number of threads per queue. You can control the number of worker threads from the command line or from the Access Server configuration page.

If your Access Server handles more than 800 requests per second, you may need to change the number of request queues. A good rule of thumb:

- Set the number of queues equal to the number-of-peak-requests-per-second/ 800.
- Set the number threads per queue to 60/number-of-request-queues.

These recommendations are based on benchmark and performance tests. For example, if the peak request rate for an Access Server is expected to be 2000 requests per second, the number of queues should be 2000/800, or approximately 2. The number of threads per queue should be 60/2 = 30.

Avoid having too few or too many threads per queue. Four threads is adequate for benchmarking, but if the directory server is slow in responding you might need more. The optimal number for the total number of threads may be closer to 100 for a modest improvement in performance on a large, fast system. However, performance is not excessively sensitive to the number.

For example, an environment with 16 queues and 16 threads apiece (256 total threads) produced about 9,000 TPS during a lab test with directory access turned off. An environment with 16 queues and 4 threads apiece (64 total threads) produced about 9,400 TPS on the same configuration. For even the largest deployments, 8 queues should be enough, and 2-4 queues may be sufficient.

To change the number of request queues

1. When starting the Access Server from the command line, type the following:
   `aaa_server -i install_dir -Q n`

   where *n* is the number of request queues. The number of queues must be an integer to a maximum of 1024.

   When using Access Server service on Windows NT or Win2K the -Q option must be specified as a startup parameter to the service. Alternatively, you can modify the following script to include this parameter:

   AccessServer_install_dir/access/oblix/apps/common/bin/start_access_server script

2. Restart the Access Server for this parameter to take effect.

Performance Considerations when Using ObMyGroups

There are several situations where the use of obmygroups proves to be a powerful approach for further integrating the Access System with applications to provide role-based information on the given user. For example, a given application could be modeled as the same set of URLs whose access is restricted only to the members of a few groups in LDAP. However to drive the navigation or presentation of the application, the application would need to know which groups the particular user belongs to because specific menu items or functions would be presented based on group membership. In this case, setting an action with ObMyGroups would be a seamless way to supply such information to the application, thus isolating the application from having to query LDAP directly or resolve whether the group is static, nested, or dynamic.

Using ObMyGroups as an authentication or authorization action requires you to consider the performance implication that resolving user group membership could have in the system for the LDAP directory and Access Server, and consequentially the Web server and application being accessed.

Guidelines for ObMyGroups

Following are some guidelines and considerations that apply to this feature:

When using ObMyGroups, by default, NetPoint searches for all group objects within the searchbase and then builds a user/group relationship cache in the Access Server, called the *Group Query Cache*.The searchbase used is the Access System searchbase. No COREid ACLs would apply to the query. For example, all groups the user belongs to are supplied, regardless of whether or not the user has read rights to those groups' class attribute.

The user/group relationship cache is built by checking group membership in this order:

1. Static Groups Membership
2. Dynamic Group Membership
3. Nested Group Membership

The Group Query Cache expires every 10 minutes. This timeout is *not* currently configurable. Contrary to user entries in the Access Server cache, there is no flush mechanism for the Group Query Cache other than waiting for the cache timeout or restarting the Access Servers.

Note: In general the user/group evaluation is an expensive operation from an Access Server perspective. Oblix strongly recommends that these always be configured with a qualifying LDAP filter. For example, enter obmygroups:ldap:///o=company,c=us??sub?(group_type=role).

Depending on the number of groups being searched, ObMyGroups processing may take a significant amount of time. It is best to specify obmygroups in an authentication rule rather than an authorization action and, if possible, have the action be a cookie so that the data is available to other applications under the same Web server without incurring an additional toll.

Note: When ObMyGroups is used in an authorization rule, limit its use to as few resources (URLs) as practical.

Even though user/group relationship information is in the Group Query Cache, the entire cache must be evaluated for each resource protected by a policy that contains an authorization action. The entire cache must be evaluated because all groups must be checked to see if the user is in that group. So even though it's a cache lookup, it's an expensive lookup. You need to consider the Access Server performance impact of using ObMyGroups for authorization actions.

Note: Oblix strongly recommends that a thorough performance test, with the specific use cases relevant to ObMyGroups actions, be designed and executed prior to rollout. This allows you to benchmark, tune, and understand the actual performance and response times involved in your specific environment.

Tuning the Caches

You can tune three groups of Access System caches:

- The cache for policy domains and policies
- The cache for user data
- The cache for URL Prefix

For additional information, see "Caching and Cloning" on page 87.

Tuning the Policy Cache

A policy cache contains information about policy domains, excluding URLs, policy descriptions, and display names. A policy cache element contains the following:

- Rules
- Actions associated with rules
- Filters, groups, and roles associated with rules
- Policy conditions for rules
- All policy domains
- All policies
- All authentication schemes

You tune the policy cache by setting the Maximum Elements in Policy Cache and Policy Cache Timeout parameters.

Note: These parameters are also documented in the *NetPoint 7.0 Administration Guide Volume 2*.

Calculating Maximum Elements in a Policy Cache

To estimate the requirements, calculate the total number of authorization rules in all policy domains and policies. To do this, search the Oblix directory tree using the following filter:

"(objectclass=oblixpolicyrule)"

When determining the number of elements in a policy cache, be sure to account for future growth of the number of rules.

Calculating Memory Requirements for the Policy Cache Elements

To calculate the memory requirements for the Access Server's policy cache elements, check the memory requirements of the directory used to store the policies.

This value is roughly the value you should provide on the Access Server.

Calculating Policy Cache Timeout

When a policy domain, rule, or policy is created or modified, the administrator has the ability on each screen to Update Cache. When these screens are saved, this forces an immediate flush of the relevant policy cache, ensuring that the change takes effect immediately.

You can have the caches time out on a regular basis as a security precaution or to clean up if the administrator does not press the Update Cache button during a policy change. The *NetPoint 7.0 Administration Guide Volume 2* discusses automatic flushing of the Access System cache.

User Cache Tuning

The tuning parameters available for this are Maximum Elements in User Cache and User Cache Timeout.

Calculating the User Cache Timeout

The user cache contains all user attributes and values used in audit and HTTP actions. You need to determine the shortest time acceptable for not changing bulk user attribute values. If the value is too large, values that are considered important from a risk-management perspective may not make their way into the NetPoint system until the cache is flushed. This can be a problem if the administrator forgets to update the cache after making changes. The administrator, delegated identity administrator, or user may change a user value in the directory and think it has been implemented when, in fact, there will be a delay until the value in the cache is flushed.

On the other hand, you should avoid setting the timeout or the cache size so small that data is flushed while a user is accessing a site, since this forces another trip to the directory.

To estimate a reasonable interval for updating user values, you can track average session times over a 24-hour period. The User Cache Timeout value can probably be set to this value or slightly higher than this value.

Calculating Maximum Elements in the User Cache

If you know the value for User Cache Timeout, you should next measure the number of users who access resources during this time interval.

When you know the maximum number of concurrent users during the time interval, the number represents the maximum number of cache elements. This is because each element contains a user DN and their attributes and values used in the audit logs and HTTP actions.

Calculating Memory Requirements for User Caches

If you know the user cache timeout and the maximum number of elements in the user cache, you can calculate hardware requirements.

To begin, calculate the requirements per cache element. An element in the user cache refers to all user attributes and their values used in all the rules. The formula for this is:

> cache element size = size of DN + size of all attribute values to be cached + 50 bytes for overhead.

You can now calculate total memory requirements as follows:

Max. elements in user cache memory requirements = Max. elements x cache element size

Tuning the URL Prefix Cache

The URL prefix cache sets the interval for flushing a URL prefix. If a URL prefix is added to a policy domain or policy, the administrator can click the Update Cache button to force an automatic cache flush. The URL Prefix reload period is an additional safeguard that provides automatic flushing.

Choose a time interval that you feel comfortable with between automatic cache flushes. The default value is 7200 seconds, or two hours.

WebGate Cache Tuning

WebGate caches information on authentication and on whether or not a resource is protected. WebGate cache tuning refers to the total number of unique URLs expected over the timeout interval.

The chances of an authentication scheme changing quickly are very low. The chances of a URL prefix being changed or added are low. If an administrator adds, changes, or deletes a URL prefix, the screens contain an Update Cache button for forcing an immediate cache flush. As a result, the default value for the WebGate cache timeout is zero. This means that the cache is not automatically updated and is flushed only when the administrator clicks the Update Cache button. If you are not comfortable with the default, choose an appropriate interval before the URL is automatically flushed from the cache.

Sizing the Maximum Elements in Cache

WebGate can cache URLs to avoid trips to the Access Server or directory server when determining if a URL requires protection. The default value for Maximum Elements is 100,000. To estimate an appropriate value for this parameter, examine the number of URLs that the Web server is protecting and the HTTP operations associated with them.

Web browsers place an internal limit of 4096 bytes on URLs. Most URLs are smaller than 4096 bytes. You may want to determine upper limits for the cache and choose the same size or a smaller size than 4096, depending on what you believe is an average URL size.

To calculate memory requirements per cache element:

memory per cache element = URL (4096 bytes) + overhead (128 bytes) = 4224 bytes

To calculate the memory requirements for maximum cache elements:

memory required = 100,000 elements x 4224 bytes/element = 422,400,000 bytes = 422 MB of memory.

Memory requirements may be smaller if the maximum number of elements is downsized according to the needs of the Web server that the WebGate is on and the average size of the URLs.

Tuning the COREid System

It is a good practice to use at least two COREid servers running in a *pooled primary* configuration. Pooled primary means using multiple COREid servers that run as primary servers, with one or more WebPass instances connecting to the primary COREid servers.

You can use separate COREid servers as secondary servers when using a pooled primary approach. If you have only two servers, a pooled primary configuration is recommended over using one primary and one secondary server. When running a pooled primary configuration, it is best to use identical but separate hardware for the COREid servers.

Advantages of pooled primary mode

- Increased performance through load balancing
- Increased availability through multiple servers
- Automatic failover

Disadvantages of pooled primary mode

- The cost of additional hardware.

  If there are no secondary servers, each primary server needs to be sized to handle the total expected load if the other primary servers are unavailable.

- Additional system configuration.

Note: COREid Server configuration and stylesheet files must be identical on all servers. This applies to all configurations that use multiple COREid Servers. You should configure all COREid Servers from a file system level, that is, ensure that all directory and file system structures are identical.

Tuning the Group Manager

The Group Manager is a COREid application. The Group Manager has unique performance tuning requirements. In particular, group size can adversely affect performance. For instance, if you have groups of more than 100,000 users, operations involving these groups may be slow. Similarly, nested groups can result in slow performance during evaluation due to the large number of members in the nested groups. Where possible, use dynamic groups instead of nested groups.

In addition, three Group Manager pages can be tuned to optimize performance:

- My Groups page
- Group Expansion page
- View Members page

Tuning the My Groups Page

You can tune the performance of the My Groups page as follows.

To tune the My Groups page

1. In the COREid System Console, navigate to COREid System Configuration > Group Manager > Group Manager Configuration > Configure Group Manager Options.

   On this screen are several Boolean flags that you can set by clicking Modify. The *NetPoint 7.0 Administration Guide Volume 1* describes these settings. As an example of how these flags might influence the performance of your system, consider the setting labeled Show nested groups. Showing nested groups can be a time-consuming operation, depending on the complexity of the group hierarchy. Setting this option to false limits group display to a simpler format, but can significantly improve the performance of the page.

   The types of attributes in step 2 are used in the My Groups profile.

2. To improve directory performance, index the following:

   - Attributes configured with the ObSDynamicMember semantic type
   - Attributes configured as the ObSStaticMember semantic type
   - All user attributes used in group dynamic filters

3. Group Manager can use an optional filter—the extra group filter—to further qualify results shown in the My Groups profile.

   The filter can be used to further qualify results under any searchbase, but would most likely be used in a FAT tree scenario where all entries are located under one container. The parameters that control the use of this filter are extra_group_filter and use_extra_group _filter_mygroups, found in the groupdbparams.xml catalog. The extra filter can be any LDAP filter and, in addition, may contain an Oblix rule substitution ($ $). When using a rule substitution, the substituted value comes from the user entry. This provides a means to link the values of attributes in a user entry with those in group entries. For example, a filter such as ou=ou would specify, that in order for a group entry to qualify, the ou of the group must be the same as that for the user.

4. The default stylesheet for the My Groups page uses DHTML and layers to render the groups in a browseable tree format. If the My Groups page has to show many groups, the browser may take a long time to render the page. You can replace the default stylesheet, gsc_myprofile.xsl, with gsc_myprofile_simple.xsl. This version of the stylesheet has a simpler, non-browseable interface and does not use DHTML and layers as much. Both stylesheets are located in the directories:

   COREid_install_dir/identity/oblix/shared (wrapper stylesheet)
   COREid_install_dir/identity/oblix/lang/en-us/style0 (stylesheet template)

For details about stylesheets and styles, see the *NetPoint 7.0 Customization Guide*.

To use gsc_myprofile_simple.xsl

1. Change the XML registry settings in the registry file:

   COREid_install_dir/identity/oblix/apps/groupservcenter/bin/groupservcenterreg.xml

2. In this file, change the following line:

   ```
   <ObStyleSheet name="gsc_myprofile.xsl"/>
   ```

   to

   ```
   <ObStyleSheet name="gsc_myprofile_simple.xsl"/>
   ```

3. Then restart the COREid Server and Web server.

Tuning the View Members Page

You can tune the performance of the View Members page in the following ways:

- You can control the behavior of the View Members page using three COREid System Console options.

  These options can be turned on and off in the COREid System Configuration > Group Manager > Group Manager Configuration > Configure Group Manager Options screen. See the *NetPoint 7.0 Administration Guide Volume 1* for information on these flags.

- You can constrain the search that can be done in the View Members page. Locate the groupMemberSearchStringMinimumLength parameter in the catalog:

  COREid_install_dir/identity/oblix/apps/groupservcenter/bin/groupservcenterparams.xml

  This parameter controls the minimum number of characters that the user must type to be able to search for members of a group. Other than being restricted to group membership searches, its usage is identical to that of the searchStringMinimumLength parameter (see "Indexing and Search Constraints" on page 31).

- You can index any user attributes used in group dynamic filters.

Tuning the Group Expansion Page

The following attributes are used in group expansion and should be indexed to improve performance:

- Any attributes configured with the *ObSDynamicMember* semantic type.
- The obgroupexpandeddynamic attribute of the oblixadvancedgroup objectclass.
- All user attributes used in the dynamic filters of the groups to be expanded.
- As discussed under "Tuning the My Groups Page" on page 57, you should use the Set Searchbase feature to localize access to sub-domains appropriately. This may also improve the performance of the Group Expansion page.

The performance of the Group Expansion page may be improved by using the extra group filter feature, discussed under "Tuning the My Groups Page" on page 57.

Other Tuning Issues

The following are issues you should examine to optimize performance.

Be Aware of Resource-Intensive Operations

Resource-intensive operations include login forms, password management, and plug-ins.

Login Forms

Login forms increase the overall load on Web servers. This may mean that you need more Web servers, depending on the size of the server and the form content. Dynamic contents and graphics adversely affect performance. An extremely high number of logins can cause network latency.

Password Management

Password management causes many directory write operations. When you implement password management, be aware that there are performance implications if you expect it to be used frequently.

Plug-Ins

NetPoint plug-ins and .exe files can affect performance. When you develop customizations for NetPoint, be aware of the impact of these customizations. To minimize performance impact:

- When creating an action for the Identity Event API, because an action incurs overhead, you should identify all the possible events to attach the action to and choose the least frequently used one that yields the desired result.
- Evaluate the sequence in which actions are executed.

  For example, suppose that you develop an exe to send an email message through the SMTP server when a user performs a particular action. Depending on how you write this program, the COREid Server may be unable to perform further actions until it receives a reply from the SMTP server. If it is the case that the SMTP server is down, there may be a large impact on performance.
- When comparing EXEs and LIBs, note that LIB actions execute quickly because they are binary code modules compiled from C or C++ source code. However, their perceived speed depends on the function they perform.

Be Sure Your Machines Are Working Properly

If you experience performance issues, you may want to check system CPU usage. For example, if a domain controller is not working well, performance will suffer.

3 Failover and Load Balancing

Failover and load-balancing are vital considerations when you performance-tune your NetPoint environment. This chapter contains the following topics:

About Load Balancing with NetPoint

NetPoint uses *load balancing* to maximize performance by distributing server requests across multiple physical servers. You configure load balancing among the components listed below.

Load balancing between NetPoint Web components and NetPoint servers includes:

- WebPass requests to one or more COREid servers
- WebGate requests to one or more Access servers

  You cannot use hardware load balancers to load balance requests from WebPass and/or WebGates to the COREid and/or Access Servers because the hardware load balancers do not understand the proprietary NetPoint protocols.

Hardware load balancers can load balance only at the connection level, not the request level. The connections to the NetPoint servers are persistent connections.

Load balancing among NetPoint servers and directory servers includes:

- COREid Server to one or more directory servers
- Access Server to one or more directory servers

Configuring Load Balancing for Web Components

NetPoint supports both simple round-robin and weighted round-robin load balancing of requests from its Web components (WebPass and WebGate) to their associated NetPoint servers (COREid and Access). You configure load balancing of Web component requests to NetPoint servers from the perspective of the web component using two key fields:

- **Maximum Connections**—The maximum connections you want opened for a given instance of the Web component. This is the total number of live connections you want established to one ore more primary NetPoint Servers at any given time. If the Web component cannot establish a connection to one of its primary servers, it will try to establish a connection to a secondary server.
- **Initial Connections**—The initial connections from the Web component to its associated NetPoint Servers. This applies to both primary and secondary NetPoint servers.

There are several procedures in this section, to use depending upon your environment:

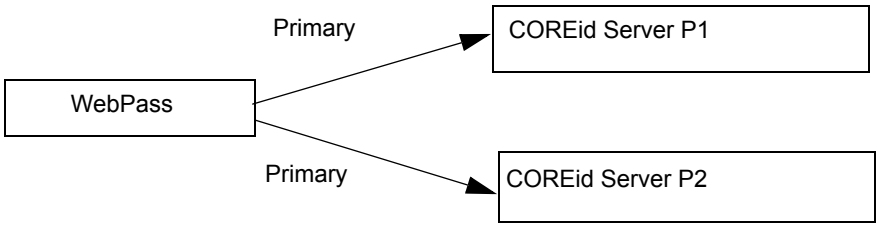
- "Configuring Simple Round-Robin Load Balancing" on page 64
- "Configuring Weighted Round-Robin Load Balancing" on page 66

Configuring Simple Round-Robin Load Balancing

Configuring simple round-robin load balancing of Web component requests means that a Web component opens a single connection to each of its associated primary NetPoint servers in the order they are listed, and distributes the requests evenly among them.

For example, assume that you have a single WebPass and two primary COREid Servers as shown in Figure 1. In this case, WebPass opens a connection to each COREid Server in the order they are listed. WebPass sends request1 to COREid P1, request2 to COREid P2, request 3 to COREid P1 and so on.

Figure 1 Simple Round-Robin Load Balancing of Web Component Requests



Max Connections:2
Initial Connections of WebPass to P1: 1
Initial Connections of WebPass to P2: 1

To configure simple round-robin load balancing

1. Access the Web component configuration whose requests you want to load balance.

   For example:

   - From the COREid System Console, select System Admin > System Configuration > Configure WebPass.
   - From the Access System Console, select Access System Configuration > AccessGate Configuration.

   See the *NetPoint 7.0 Administration Guide Volumes 1* and *2* for more information about Web component configuration.

2. Enter the Maximum Connections you want opened for this WebPass or WebGate.

   This is the total number of live connections you want established to one or more primary NetPoint COREid or Access servers at any given time. If the Web component cannot establish a connection to one of its primary servers, it tries to establish a connection to a secondary server.

3. Leave the Initial Connections to each associated NetPoint Server at the default, which is 1.

   This applies to both primary and secondary NetPoint servers.

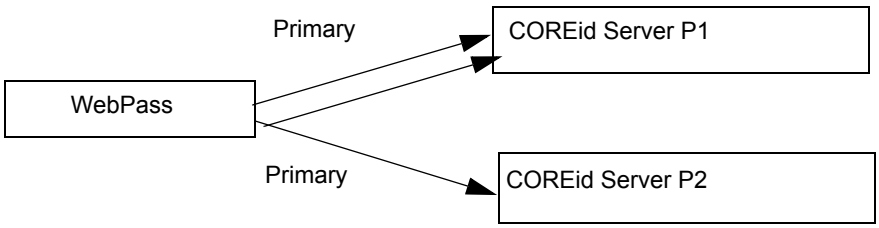
Configuring Weighted Round-Robin Load Balancing

You may want to configure weighted round-robin load balancing of Web component requests if your NetPoint Servers have varying performance capacities. The primary difference when configuring weighted load balancing is that you adjust the initial connections you want established to each NetPoint Server.

Figure 2 provides an example. Assume you have two primary COREid Servers as shown in Figure 2. However, COREid P1 can handle additional load. Then you may want to configure WebPass to open two connections to COREid P1, and one connection to COREid P2. The Maximum Connections for that WebPass would be 3: COREid P1, request2 to COREid P2, request 3 to COREid P1 and so on.

Note: When you associate an AccessGate with an Access Server cluster, NetPoint automatically configures the number of connections between the AccessGate and all the Access Servers in the cluster based on the maximum number of connections that is specified for the cluster. Load balancing is dynamically configured and NetPoint ensures that the AccessGate routes requests to the most lightly loaded Access Servers in the cluster.

Figure 2 Weighted Load Balancing Layout Using Two Servers and no Failover



Max Connections:3
WebPass to P1: 2
WebPass to P2: 1

To configure weighted round-robin load balancing of Web component requests

1. Access the Web component where you will configure load balancing.

   For example:

   - From the COREid System Console, select System Admin > System Configuration > Configure WebPass.
   - From the Access System Console, select Access System Configuration > AccessGate Configuration.

2. Enter the Maximum Connections you want opened for a given WebPass.

   Again, this is the total number of live connections you want established to one or more primary COREid Servers at any given time. If WebPass cannot establish a connection to one of its primary servers, it tries to establish a connection to a secondary server.

3. Enter the Initial Connections to each associated NetPoint server to reflect that server's capacity.

   Again, this applies to both primary and secondary COREid Servers. When you associate an AccessGate with an Access Server cluster, NetPoint automatically configures the number of connections between the AccessGate and all the Access Servers in the cluster based on the maximum number of connections specified for the cluster. Load balancing is dynamically configured and NetPoint ensures that the AccessGate routes requests to the most lightly loaded Access Servers in the cluster.

Configuring Load Balancing among NetPoint and Directory Servers

NetPoint supports simple round-robin load balancing of NetPoint server requests to two or more directory servers.

Important: The COREid System generally does not support load balancing for Oblix (configuration) data because many functions are dependent on data carried over from the previous request. In a load balanced environment, this data may not yet be available to the replicated server.

The instructions for configuring load balancing for directory servers vary depending on the type of component (COREid Server, Access Server, Access Manager), and whether you are configuring load balancing for user data or configuration data. See Table 1, "Supported Load Balancing Configurations to Directory Servers," on page 68.

Previous versions of NetPoint managed directory connection information solely through XML and LST configuration files. Recently, NetPoint provided the ability to manage this information through the interface using the Directory Profile screen in the COREid System Console and the Access System Console. However, some Oblix and Policy data is still managed through the XML and LST files. Therefore, you will find yourself using combined methods for configuring load balancing.

Table 1 Supported Load Balancing Configurations to Directory Servers

Component	Data Store	Operation	Where Configured
COREid Server	User	READ	Directory Profile See “Configuring Load Balancing for User Data” on page 69.
		WRITE	*Not supported*
	Oblix	READ	*Not Supported*
		WRITE	*Not Supported*
Access Server	User	READ	Directory Profile
		WRITE	*Not supported* **Note**: Write operations apply to the Access Server only if password policy is enabled.
	Oblix	READ	ConfigureAAAServer command line tool See “Configuring Load Balancing of Oblix & Policy Data” on page 70.
		WRITE	*n/a*
	Policy	*READ	ConfigureAAAServer command line tool **Note**: If you turn on the Access Management Service for that Access Server, you cannot load balance requests from an Access Server to the data store containing policy information.
		WRITE	n/a
Access Manager	User	READ	Directory Profile

This section includes the following procedures:

- “Configuring Load Balancing for User Data” on page 69
- “Configuring Load Balancing of Oblix & Policy Data” on page 70
- “Adjusting Connection Pooling for a Directory Server Instance” on page 72

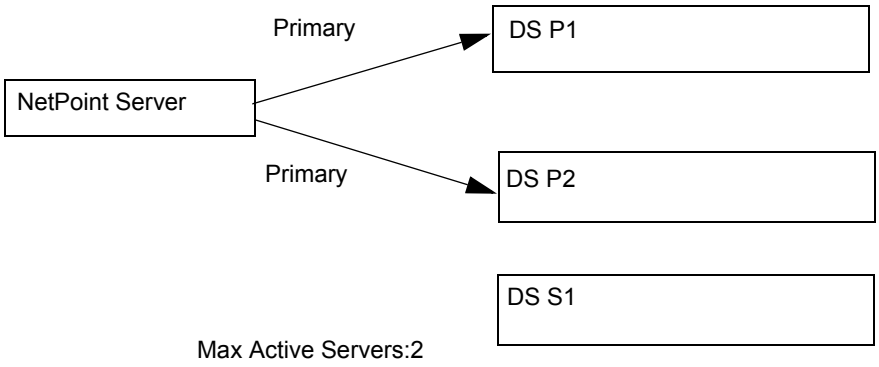
Configuring Load Balancing for User Data

By default, NetPoint creates a directory profile for each installed component. When configuring load balancing for NetPoint requests to directory servers containing user data, you use the Directory Profile page. For more information on directory profiles, see the *NetPoint 7.0 Administration Guide Volume 1*.

Maximum Active Servers—The total number of live primary directory servers you want up and running at all times. Requests are distributed evenly across these servers.

As shown in Figure 3, assume you have 2 primary directory servers and one secondary directory server. You want to load balance between the primary directory servers, so you should enter Max Active Servers = 2. The secondary directory server only comes into play during failover and does not impact this setting.

Figure 3 Load Balancing of NetPoint Requests to Directory Servers



To configure load balancing for user data

1. Access the Directory Profile page.

   For example:

   - From the COREid System Console, select System Admin > System Configuration > Configure Directory Options.
   - From the Access System Console, select System Admin > System Configuration > View Server Settings > Directory Options.

2. Select the directory profile that contains connection information for the component and data where you want load balancing.
3. Enter the Maximum Active Servers available for load balancing.

Configuring Load Balancing of Oblix & Policy Data

When you configure load balancing of Access Server requests to directory servers containing Oblix and/or policy data, use the configureAAAserver tool. The following instructions assume that you have two or more primary directory servers set up.

Important: Load balancing for Oblix data is generally *not* supported for the COREid Server. These instructions address the Access Server only. The ConfigureAAAServer is an older tool that does not have the most current naming conventions. Maximum Connections as shown in the tool is equivalent to the Maximum Active Servers as explained earlier in "Configuring Load Balancing among NetPoint and Directory Servers" on page 67

To configure load balancing of configuration and policy data for the Access Server

1. From the command prompt, access the configureAAAServer tool located in *AccessServer_install_dir*/access//oblix/tools/configureAAAServer
2. Run configureAAAServer utility using the reconfig <*install_dir*> option.

   where <*install_dir*> is the name of the directory where your Access Server is installed.

   For example:

   ```
   configureAAAServer reconfig "c:\Program
   Files\NetPoint_70\access"
   ```

3. Enter the number that corresponds to the Access Server security mode. These are the Access Servers that will connect to the directory servers.

   1) Open

   2) Simple

   3) Cert

   You are then be asked if you want to specify failover information for Oblix or Policy.

4. Select Yes (Y).
5. Specify whether the data is stored in:

   1) Oblix tree

   2) Policy tree

6. Enter 1 to add a failover server at the following prompt.

   1) Add a failover server

   2) Modify a failover server

   3) Delete a failover server

   4) Modify common parameters

7. Enter the following information:

   - Directory server name
   - Directory server port

     Note: For LDAP in an Active Directory forest environment, use port 3268 for Open mode and port 3269 for SSL mode. These two are the global catalog ports.

   - Directory server login DN
   - Directory server password
   - Directory Server security mode

     1) Open

     2) SSL

8. Enter 1 as the priority since you are configuring load balancing

   1) Primary

   2) Secondary

9. Enter 4 to modify common parameters.
10. Enter 1 to specify the Maximum Active Servers.

    Note: Maximum Connections as shown in the tool is equivalent to the Maximum Active Servers as explained earlier in "Configuring Load Balancing among NetPoint and Directory Servers" on page 67

11. Enter the total number of primary directory servers available for load balancing.
12. Enter 5 to Quit.
13. Enter 1 to commit changes.

Adjusting Connection Pooling for a Directory Server Instance

In addition to specifying load balancing of requests evenly across directory server instances using the Max Active Servers parameter, you can also adjust connection pooling within a specific directory server instance by entering the Initial and Maximum Connections for that specific server instance. The request is sent on the connection with the least load. Similar to configuring load balancing, you must adjust directory pool connections from the following places:

- **Directory Profile Page**—See "To adjust directory connection pooling from the directory profile" on page 72.
- **ConfigureAAAServer Tool**—See "To adjust directory connection pooling using the ConfigureAAAServer tool" on page 73.

To adjust directory connection pooling from the directory profile

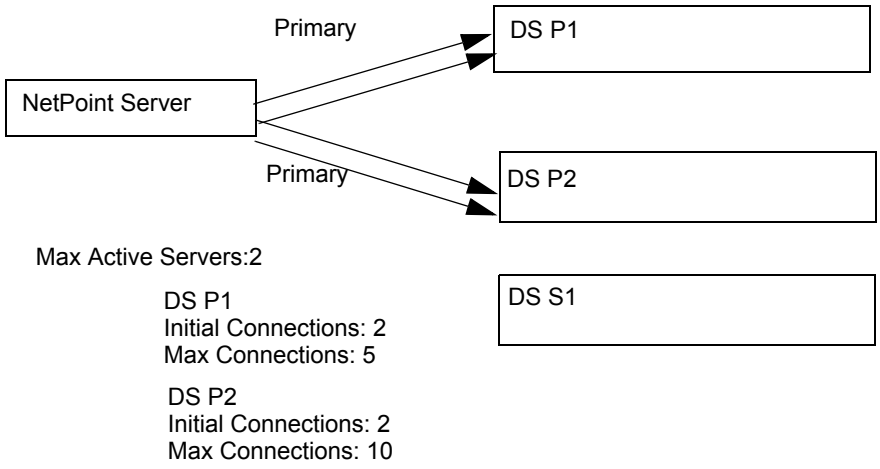
1. From the Directory Profile page, select a specific DB (directory) instance.
2. Enter the Initial Connections you want opened to this directory server.

   The default is 1.
3. Enter the Maximum Connections allowed to this directory server instance.

   If any of the connections to the directory server are lost, then NetPoint can attempt to open connections up to the maximum entered.

   When the first request is sent to a directory server, the initial number of connections is opened. When connections are lost, or when there are more service threads than connections, new connections are opened (up to the maximum). See Figure 4 for an example.

Figure 4 Adjusting Connection Pooling for Directory Server Instances



For example, assume there are 5 initial connections and 10 max connections to a directory server. When a service thread makes a request to the directory server, then 5 initial connections are created. Assume now, that there are 5 concurrent active service threads that require information from the directory server. They will use all of the existing 5 connections.

If the concurrent service threads increase beyond 5 more connections, NetPoint can create up to 10 max connections to that directory server. When the limit of 10 connections is reached, and there are 11 or more concurrent service threads, the 11th service thread will get one of the least loaded connections from the pool of 10 connections. The connection is then shared by more than one service thread.

Note: There is no general *optimal* value for the initial and maximum connections, given variables such as directory configuration, hardware, and so on. However, you should *not* set the number of connections higher than the number of threads for a given NetPoint server.

To adjust directory connection pooling using the ConfigureAAAServer tool

1. See "To configure load balancing of configuration and policy data for the Access Server" on page 70.
2. Specify the Modify Common Parameters option when prompted.

About Failover with NetPoint

NetPoint uses *failover* to provide uninterrupted service. Failover involves re-directing requests to another server when the original request destination fails.

Failover is accomplished by configuring primary and secondary servers and identifying specific parameters for the failover process. Failover can be configured between:

- **NetPoint Web components to NetPoint servers**
  - WebPass requests to secondary COREid servers
  - WebGate requests to secondary Access Servers
- **NetPoint servers to directory servers**
  - COREid Server to secondary directory servers
  - Access Server to secondary directory servers
- **Access Manager to directory servers**
  - NetPoint 7 now provides the ability to perform failover from the Access Manager to secondary directory servers.

Note: Access Manager failover for Oblix and policy data is not supported.

Primary Versus Secondary Servers

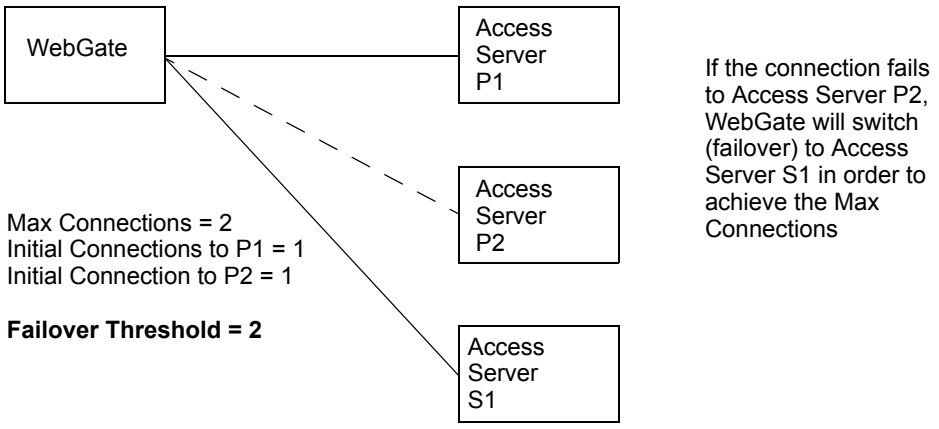
NetPoint components first attempt to connect to a primary server.

If the primary server is unavailable, a connection attempt may be made to a secondary server. NetPoint continues to attempt to connect to the primary server, and when the connection is re-established, the connection to the secondary server is dropped. Any server can be configured as a primary or a secondary server. For example, you could designate less powerful or geographically distant servers as secondary.

Configuring Failover of Web Components

Configuring failover enables a WebPass or WebGate to check the health of its connections, and failover to secondary NetPoint Servers in case one or more primary servers go down. You configure failover from the perspective of the Web component. See Figure 5 for an example.

Figure 5 Basic Failover Scenario between a WebGate and its Access Servers



Failover Threshold—The key to configuring failover is the Failover Threshold field in the Web component configuration. This specifies the minimum number of live primary connections required. If the number of live connections drops below the failover threshold, then the Web component attempts to establish connections to its secondary servers in the order they are listed. The default is the maximum number of connections.

Sleep For Interval—The default interval is 60 seconds. After this interval, the WebPass or WebGate will check to see if the number of valid connections equals the maximum number of connections configured. If the number of valid connections does not equal the maximum number of connections (drops below the failover threshold), the Web component tries to establish connections to its secondary servers in the order they are configured. Then, at the interval specified, the Web component tries establishing a connection to the primary servers. When it establishes the connection, it drops the connections to the secondary servers after finishing the request it is servicing.

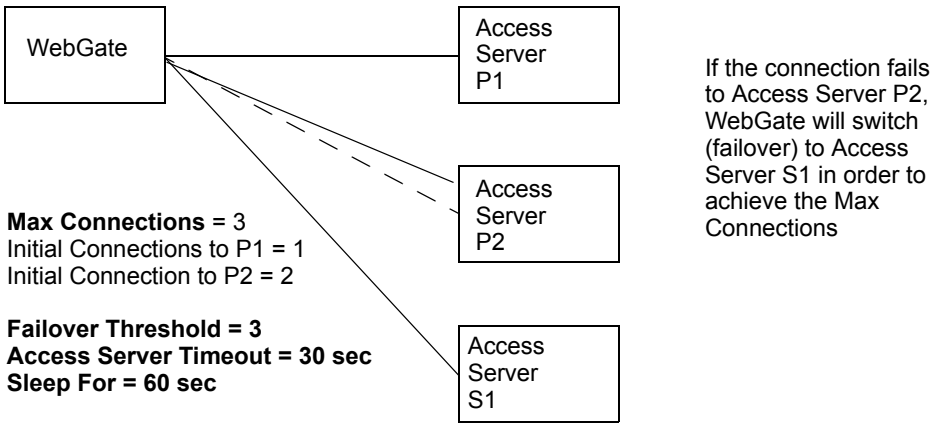
Timeout Threshold—Specifies how long (in seconds) the Web component waits for a non-responsive NetPoint server before it considers it unreachable and attempts to contact another. No value indicates there is no timeout, and the Web component will wait endlessly for a response from the NetPoint server. Leaving no timeout can potentially result in a hung session or default to a "TCP" timeout. For example:

- For an existing WebPass, enter a value for the COREid Server Timeout Threshold.
- For an existing AccessGate, enter a value for the Access Server Timeout Threshold.

Note: Do *not* configure two servers to provide mutual failover for each other. In the example above, Access Server P1 should not be configured to provide failover for Access Server P2 or any other server that provides failover for Access Server P1.

In Figure 6, a WebGate communicates with two primary Access Servers Access Server P1 and Access Server P2.

Figure 6 Failover Scenario between a WebGate and its Associated Access Servers



- The **Maximum Connections** is 3. Assume that Access Server P2 can handle additional load, and therefore two initial connections are established to it.
- The **Failover Threshold** is 3. If the number of valid connections drops below the failover threshold, WebGate tries to establish a connection to its secondary server, Access Server S1. Assume that Access Server P2 dropped its second connection.
- The **Access Server Timeout Threshold** parameter is set for 30 seconds. If the WebGate does not receive a response from the Access Server P2 in 30 seconds,

it considers the server unreachable and attempts to failover to Access Server S2.

- The **Sleep For** parameter is set for 60 seconds. Every 60 seconds this WebGate checks whether the number of valid connections to primary servers equals the specified number of maximum connections. If not, WebGate continues its attempts to re-establish connections to failed primary Access Server P2.

To configure failover for Web component requests

1. Access the WebPass or WebGate Web component where you will configure failover.

   For example:

   - From the COREid System Console, select System Admin > System Configuration > Configure WebPass > *Name* > Modify.
   - From the Access System Console, select Access System Configuration > AccessGate Configuration > All > Go > *Name*.

   See the *NetPoint 7.0 Administration Guide Volumes 1* and *2* for more information.

2. In the Failover Threshold field, enter the required number of live connections from the Web component to its primary NetPoint server.
3. Enter the Sleep For interval in seconds.
4. Enter a Timeout Threshold to specify how long (in seconds) the Web component waits for a non-responsive NetPoint server before it considers it unreachable and attempts to contact another:

   - **WebPass**—Enter a value for the **COREid Server Timeout Threshold.**
   - **AccessGate**—Enter a value for the **Access Server Timeout Threshold**.

5. Save your changes.

About Failover Between NetPoint and Directory Servers

Failover for NetPoint servers occurs when the number of live primary directory servers drops below the number in the Failover Threshold field. The instructions for configuring failover from NetPoint components to directory servers vary depending on the type of component (COREid Server, Access Server, Access Manager), and whether you are configuring failover for user data or Oblix data.

Table 2 Supported Failover Configurations to Directory Servers

Component	Data Store	Operation	Where Configured
COREid Server	User	READ/ WRITE	Directory Profile
	Oblix	READ/ WRITE	Directory Profile and XML configuration files
Access Server	User	READ/ * WRITE	Directory Profile **Note**: Write is only applicable when if password policy is enabled.
	Oblix	READ	ConfigureAAAServer command line tool
	Policy	READ	ConfigureAAAServer command line tool
Access Manager	User	READ	Directory Profile
	Oblix		Access Manager failover for Oblix and policy data is not supported.
	Policy	READ	LST configuration files

Note: Previous versions of NetPoint managed directory connection information solely through XML and LST configuration files. Recently, NetPoint provided the ability to manage this information through the interface using the Directory Profile screen in the COREid System Console and the Access System Console. However, some Oblix and policy data is still managed through the XML and LST files. Therefore, you will find yourself using the combined methods for configuring failover.

Configuring Directory Failover for User Data

You use the Directory Profile screen when configuring failover of NetPoint requests to directory servers containing user data.

Failover Threshold—The required number of live primary directory servers. If the number of primary servers drops below the failover threshold, NetPoint fails over to a secondary directory server.

When the NetPoint Server sends a request on one of its directory connections, and the LDAP SDK returns a connection or *server-down* error, the directory server is assumed not to be available. If the number of primary directory servers drops below the failover threshold, then the NetPoint attempts to establish connections to its secondary servers in the order they are listed.

If there is a primary server available when failover occurs, the NetPoint Server fails over to the primary server first.

Sleep For—The number of seconds before the watcher thread *wakes up* and attempts to re-establish connections and create new connections if the connection was down. The watcher thread maintains a pool of good connections at all times.

By default, NetPoint creates a directory profile for each installed component. You need to access this page to configure directory failover. For more information on directory profiles, see the *NetPoint 7.0 Administration Guide Volume 1*.

To configure directory failover for user data

1. Access the Directory Profile page:

   For example:

   - From the COREid System Console, select System Admin > System Configuration > Configure Directory Options.
   - From the Access System Console, select System Admin > System Configuration > View Server Settings > Directory Options.

2. Select the directory profile that contains connection information for the component and data where you want failover.
3. Enter the Failover Threshold.
4. In the Sleep For field, enter the number of seconds before the watcher thread wakes up and attempts to re-establish connections and create new connections if the connection was down.
5. Add the Database Instances and indicate their status as secondary servers.

Configuring Directory Failover for Oblix and Policy Data

The instructions for configuring failover from NetPoint components to directory servers vary depending on the type of component (COREid Server, Access Server, Access Manager), and whether you are configuring failover for user data or Oblix data. See Table 2, “Supported Failover Configurations to Directory Servers,” on page 78.

Note: Access Manager failover for Oblix and policy data is not supported.

Task overview: Configuring directory failover for Oblix and policy data

1. “Configuring COREid Server Failover for Oblix Data” on page 80
2. “Configuring Access Server Directory Failover for Oblix and Policy Data” on page 83

Configuring COREid Server Failover for Oblix Data

For the COREid Server, most Oblix (configuration) data is still managed through the XML configuration files. However, multi-language and referential-integrity data is managed through the Directory Profile page.

When the primary configuration data directory server is down, there is no way for the COREid Server to read any configuration entries. Therefore, the COREid Server reads failover.xml to obtain *bootstrap* secondary directory server information. See Figure 7 Sample Failover.xml on page 81 for an example.

Task overview: Configuring COREid Server failover for Oblix data

1. “To configure COREid Server directory failover for Oblix data” on page 80
2. “To create the encrypted password for the bind DN” on page 81
3. “To create failover.xml” on page 81

To configure COREid Server directory failover for Oblix data

1. Using the Directory Profile page, enter failover specifications for the directory profile containing the Oblix branch of the tree, as described in “Configuring Directory Failover for User Data” on page 79.
2. Create a file called failover.xml and add it to *COREid_install_dir*/identity/oblix/config/ldap directory.

To create the encrypted password for the bind DN

1. Locate the obencrypt tool in:

   AccessServer_install_dir/access/oblix/tools/ldap_tools/

2. Run obencrypt *password*.

3. Copy and paste the encrypted password into your failover file, as described in "To create failover.xml" on page 81.

To create failover.xml

1. Copy and paste the existing sample_failover.xml template into the directory:

   COREid _install_dir/identity/oblix/config/ldap

2. Open the copy with a text editor and add failover information for secondary servers using Figure 7 as a guide.

3. Copy and paste the encrypted password into your failover file.

4. Rename the copy to failover.xml.

5. Repeat as necessary for each applicable COREid Server.

Figure 7 Sample Failover.xml

```
?xml version="1.0" encoding="ISO-8859-1"?>
<CompoundList xmlns="http://www.oblix.com"
ListName="failover.xml">
<!-- # Max number of connections allowed to all the active ldap
servers -- note this is the same as Max Active Servers>
<SimpleList>
<NameValPair ParamName="maxConnections" Value="1"></
NameValPair>
</SimpleList>
<!-- # Number of seconds after which we switch to a secondary or
reconnect to a restarted primary ldap server -->
<SimpleList>
<NameValPair ParamName="sleepFor" Value="60"></NameValPair>
</SimpleList>
<!-- # Max amount of time after which a connection to the ldap
server will expire -->
<SimpleList>
<NameValPair ParamName="maxSessionTime" Value="0"></
NameValPair>
</SimpleList>
<!-- # Minimun number of active primary ldap servers after which
failover to a secondary server will occur -->
```

```
<SimpleList>
NameValPair ParamName="failoverThreshold" Value="1"></
NameValPair>
</SimpleList>
<!-- # Specify the list of all secondary ldap servers here -->
<ValList xmlns="http://www.oblix.com"
ListName="secondary_server_list">
<ValListMember Value="sec_ldap_server"></ValListMember>
</ValList>
<!-- # Specify the details of each secondary ldap server here -->
<ValNameList xmlns="http://www.oblix.com"
ListName="sec_ldap_server">
      <NameValPair ParamName="ldapSecurityMode" Value="Open"></
NameValPair>
        <NameValPair ParamName="ldapServerName"
Value="instructor.oblix.com"></NameValPair>
        <NameValPair ParamName="ldapServerPort" Value="9002"></
NameValPair>
       <NameValPair ParamName="ldapRootDN" Value="cn=Directory
Manager"></NameValPair>
        <NameValPair ParamName="ldapRootPasswd"
Value="000A0259585F5C564C"></NameValPair>
        <NameValPair ParamName="ldapSizeLimit" Value="0"></
NameValPair>
        <NameValPair ParamName="ldapTimeLimit" Value="0"></
NameValPair>
    </ValNameList>
</CompoundList>
```

Configuring Access Server Directory Failover for Oblix and Policy Data

These procedures describe configuring failover from the Access Server to one or more directory servers containing Oblix and/or policy data.

- "To configure Access Server failover for Oblix and policy data" on page 83
- "To add a failover directory server using the ConfigureAAAServer tool" on page 83

To configure Access Server failover for Oblix and policy data

1. Using the Directory Profile page, enter the failover for the directory profile containing the Oblix branch of the tree.

   See "Configuring Directory Failover for User Data" on page 79 for general instructions.

2. Repeat the above procedure for the directory server containing policy data, if applicable.

   Note: For the Access Server, most Oblix data is still managed through the .LST configuration files. However, multi-language and referential-integrity data is managed through the Directory Profile page.

3. Add failover information using the configureAAAServer tool, as described next.

To add a failover directory server using the ConfigureAAAServer tool

1. From the command line, navigate to the folder where configureAAAserver tool is located.

   For example, the default location is:

   AccessServer_install_dir\access\oblix\tools\configureAAAServer

   where *AccessServer_install_dir* is the directory where the Access Server is located.

2. Run configureAAAServer tool using the following arguments:

   ```
   configureAAAServer reconfig AccessServer_install_dir
   ```

   For example:

   ```
   configureAAAServer reconfig "c:\Program
   Files\NetPoint_70\access"
   ```

3. Enter the number that corresponds to the Access Server security mode for Access Servers that will connect to the directory servers.

   1) Open

   2) Simple

   3) Cert

   You are then be asked if you want to specify failover information for Oblix or Policy.

4. Select Yes (Y).
5. Specify whether the data is stored in the:

   1) Oblix tree

   2) Policy tree

6. Enter 1 to add a failover server at the following prompt.

   1) Add a failover server

   2) Modify a failover server

   3) Delete a failover server

   4) Modify common parameters

   5) Quit

7. Enter the following information:

   - **Directory server name**
   - **Directory server port**

     Note: For LDAP in an Active Directory forest environment, use port 3268 for Open mode and port 3269 for SSL mode. These two are the global catalog ports.

   - **Directory server login DN**
   - **Directory server password**
   - **Directory Server security mode**

     1) Open

     2) SSL

   - **Priority** Enter 2 as the priority

     1) Primary

     2) Secondary

8. Enter 5 to Quit.

   You are prompted to commit the changes.

9. Select Y to commit your changes.

ConfigureAAAServer automatically creates the following .lst files in *AccessServer_install_dir*/access/oblix/config/ldap

- AppDBfailover.lst
- ConfigDBfailover.lst
- WebResrcDBfailover.lst

4 Caching and Cloning

Caching information and cloning play key roles when you performance tune your NetPoint environment. This chapter contains the following topics:

Cloned and Synchronized Components

Instead of using the command line or the installation GUI to install a NetPoint component, you can automatically install a component by *cloning* the configuration of an already installed component. Cloning creates a copy of a component on a remote system using an already-installed component as a template.

Synchronizing allows you to harmonize two installations of the same NetPoint component when one is more up-to-date than the other. Synchronization can be used to upgrade or repair installations on similar platforms.

See the *NetPoint 7.0 Installation Guide* for more information on cloning and synchronizing.

About Caching Recent Information

A cache is in-memory storage that keeps a copy of recently used information. NetPoint retrieves information that is used frequently from a cache instead of the directory. This allows information to be retrieved quickly. NetPoint uses several caches to improve performance.

The COREid and Access Systems contain different caches. The COREid System caches configuration settings and group information. The Access System caches information on password policies, policy domains, user credentials, and authentication schemes.

Some operations performed in the COREid System impact the evaluation of access policies in the Access System. For example, if you deactivate a user in the COREid System, that change must be reflected in the Access System so the user does not have access to the resources that the Access System protects.

The optimum number of elements in a cache is a balance between:

- The number of elements required to be in the cache. This depends on the information being cached and system usage.
- The RAM available for caching.

The higher the number of elements in a cache, the greater the probability of finding the requested information and improving performance.

In a worst case scenario, if the cache uses a lot of memory and the machine on which the component runs does not have enough RAM, the operating system could spend too much time swapping pages in and out of memory. This would decrease performance.

The minimum cache size for optimal performance would be:

N = XR

where:

N = the number of entries in the cache.
X = the number of new sessions per second.
R = the average length of a session in seconds.

For example, if:

X = 100 sessions per second, and
R = 600 seconds per session, then
N = 60,000 number of entries in the cache

The default cache size for NetPoint components is sufficient for most installations.

Triggering COREid-to-COREid Cache Flush Events

COREid Servers can communicate with one another. They do so primarily for cache flush requests. When a cache is updated on one server, that server tells the other servers to update their caches. The COREid Server has several caches. Each can receive a cache flush request if data is modified:

- **Configuration Data**—Object classes, attributes, tabs, and panels. The Oblix-specific data (OSD) cache is flushed automatically. In addition, you can request a cache flush as described in "To clear the OSD cache" on page 90.
- **Group Objects**—Automatic when you modify a group, or you can explicitly request a cache flush as described in "Caching COREid Group Objects" on page 91.
- **DNs of User Objects**—This is the name cache. It is flushed automatically.
- **Read and Write Privileges for Attributes**—Automatic cache flush.
- **Delegated COREid Administration and Auditing**—Automatic cache flush.
- **Workflow Definitions and participants**—Automatic cache flush.
- **Basic Configuration Data Entered in Setup**—Oblix base (configuration DN) and searchbase. Automatic cache flush.
- **User-Defined Portal Inserts**—Invoked explicitly through a URL, as described in the *NetPoint 7.0 Customization Guide*.

Cache Timeout

The cache timeout specifies how long an element is held in the cache. When the time expires, the element is removed from the cache and must be retrieved from the directory.

If the information changes, but the cache does not, the NetPoint component uses outdated information until the information expires from the cache. Updating caches when you change the information is one way to avoid this problem. If updating caches is not possible, such as when a user's information is changed through other software, then configuring a reasonable timeout for the cache is another solution.

A shorter timeout means information about the user is more recent, but the COREid Server and the Access Server must fetch data more often from the directory. This may reduce performance. Setting a long timeout means out-of-date information could remain in the cache for a longer period of time.

Note: In general, setting a lower cache timeout value means the data is more recent. However, a cache timeout value of 0 means the cache never expires.

Caching COREid System Information

The COREid System caches configuration information for Oblix-specific data such as object classes, attributes, panels, and tabs used by COREid applications. This information is automatically cached during startup.

You must be a NetPoint Administrator to view, reload, or clear the OSD cache.

To view the OSD cache contents

1. Launch the COREid System Console.
2. Click the System Configuration tab and select View Server Settings.
3. In the View Server Settings page, click Cache.
4. In the Cache page, click View Cache Contents.

   The cached OSD information is displayed.

To load the OSD cache

1. Launch the COREid System Console.
2. Click the System Configuration tab and select View Server Settings.
3. In the View Server Settings page, click Cache.
4. In the Cache page, click Load memory cache.

   The latest information is loaded into the cache.

   To update existing information, you must first clear the cache and then reload the information.

To clear the OSD cache

1. Launch the COREid System Console.
2. Click the System Configuration tab and select View Server Settings.
3. In the View Server Settings page, click Cache.
4. In the Cache page, click Clear memory cache.

   The cache is cleared.

See also:

- “Caching COREid Group Objects” on page 91
- “Turning Off the Credential Mapping Cache” on page 93

Caching COREid Group Objects

The COREid System provides a cache for group objects to boost performance for all group functions, especially those involving computation of parent (isMemberOf) groups, and children or nested groups.

A group is cached only when NetPoint makes its first request for that group. Subsequent requests for the group are directed to the cache. When you modify a group, it is removed (flushed) from the cache along with any parent, child, or nested groups. In a multi-server configuration, when a group is modified on one COREid Server, all other COREid Servers are notified so that they remove the group and all its dependent groups from their caches. When NetPoint receives another request for the modified group, it re-stores the group in the cache. This ensures that the cache has the most recent information for the group.

When you search for a group, NetPoint searches the directory, not the cache.

You manage the group cache using the groupdbparams.xml configuration file.

To configure group cache parameters

1. Navigate to the file groupdbparams.xml located in:

   COREid_install_dir\identity\oblix\data\common

   where *COREid_install_dir* is the directory where COREid is installed.

2. Configure values for the parameters in the groupdbparams.xml file that control the cache-related functions described in Table 3.

Table 3 Parameters in groupdbparams.xml

Parameter	Description
GroupCacheTimeout	The time period (in seconds) for which the object is valid. By default, the Group Cache never times out. Default = 0.
GroupCacheMaxNumElements	The maximum number of group objects that can be stored in the cache. Default = 10000. If the cache is at its maximum size, objects that are not accessed often are replaced by those that are more frequently accessed. Consider the memory limitations of your system before setting the value for this parameter.

Table 3 Parameters in groupdbparams.xml

GroupCacheDisabled	Indicates whether the cache can be used or not. Default = false. A value of false indicates that the cache can be used. A value of true indicates that the cache cannot be used. If a cache is disabled, all reads of group objects will go to the directory.
GroupCacheReadFromMaster	Forces reads of groups so that the cache can do reads from the master replica in a replicated environment. This ensures that the cache does not contain old information in case a read from a consumer replica occurs before the consumer replica has received the updated information from the master replica. Default = false If value = true, it indicates that the cache should read from the master replica.

On occasion, it may be necessary to remove a group from the cache to maintain cache integrity. For example, you may have to remove a group that has been modified using an application other than NetPoint to ensure that the updated group is read when the cache receives a read request for the group.

A NetPoint Master Identity Administrator can clear the group cache through the COREid System Console, or with an Identity XML function named *flushGroupCache*. See the IdentityXML chapter in the *NetPoint 7.0 Developer Guide* for more information.

To clear group caches from the COREid System Console

1. Launch the COREid System Console.
2. Click the Group Manager Configuration tab.
3. In the Group Manager Configuration page, click Group Cache.
4. In the Group Cache page, click the Clear group cache link.

   The group cache is cleared and a message appears confirming whether or not the operation was successful.

Turning Off the Credential Mapping Cache

If a user is deactivated in the COREid System, you can deny that user access to protected resources. Because the COREid Server does not automatically flush the Access Server credential mapping cache when a user is deactivated, you must manually turn off the credential mapping cache.

When the cache is turned off, the credential mapping plug-in communicates directly with the LDAP directory whenever it must map a user to a user profile (DN). Otherwise, the plug-in uses the cached credentials for the user.

By default, the credential mapping cache is enabled.

To disable use of this cache, set the obEnableCredentialCache parameter to false in the credential_mapping plug-in. If you deactivate a user who is logged in, the user will still have access to resources based on policy information and prior authentication. However, if obEnableCredentialCache is set to false, when the user's session token expires or the user logs out, the user will not be allowed access to a protected resource the next time the user is authenticated.

Table 4 shows the possible values for the parameter.

Table 4 Parameters for obEnableCredentialCache

Value	Interpretation
no value	Credential mapping cache turned on
true	Credential mapping cache turned on
false	Credential mapping cache turned off

Here is an example of the credential_mapping authentication plug-in with the credential mapping cache turned off:

```
credential_mapping
obMappingBase="%domain%",obMappingFilter="
(&(&(objectclass=user)(samaccountname=%userid%))
(|(!(obuseraccountcontrol=*))
(obuseraccountcontrol=ACTIVATED)))",
obdomain="domain",obEnableCredentialCache="false"
```

To set the obEnableCredentialCache parameter

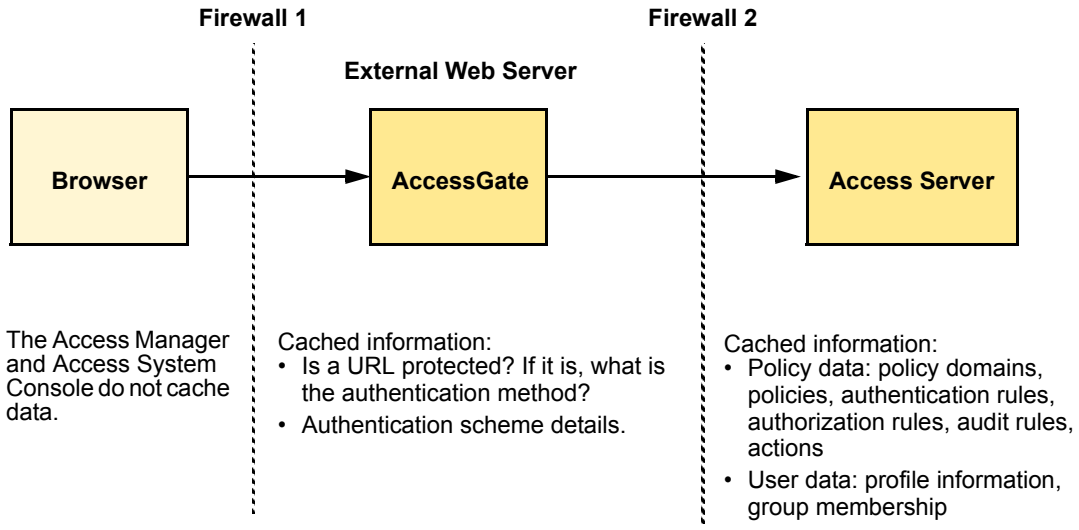
1. In the Access System Console, select Access System Configuration > Authentication Management.
2. Select the authentication scheme you want to modify, then click Modify.
3. Click the Plugins tab.
4. Add the obEnableCredentialCache="false" parameter to the credential_mapping plug-in.

Caching Access System Information

The Access System caches information on password policies, policy domains, user credentials, and authentication schemes.

Figure 8 illustrates how caching works in the NetPoint Access System. It is not necessarily a deployment recommendation.

Figure 8 Illustration of Caching in NetPoint



You can specify the maximum number of elements in a cache. The Access System ensures that the total number of elements in a cache never exceeds the maximum specified for that cache.

For example, suppose you have set the Access Server's user cache to contain a maximum of 100,000 elements. If a user is added to the system when the cache is full, the Access System removes the user element that has not been used in the longest time and adds information about the new user to the cache.

Note: WebGate and Access client configurations are cached in the Access Server. To reduce off-time network traffic between WebGate and Access Server and between Access Server and LDAP directory server, you can change the default configuration cache timeout. For details about reducing network traffic between components, see the *NetPoint 7.0 Administration Guide Volume 2*.

Access Server Cache Configuration

The Access Server caches information that includes data on policy domains, policies, users, host identifiers, and password policies. When information is updated for any of these, ensure that the Update Cache box is selected to immediately update the Access Server caches.

Information in a Policy Cache

An Access Server's policy cache contains information about policy domains, policies, authentication, authorization, audit rules, and actions.

An Access Server caches policy information for efficient runtime lookup. If policy information is not in the cache, the Access Server must obtain it from the directory. Obtaining data from the directory is slow when compared to the AccessGate obtaining information from the Access Server, so preferably all policy information should be cached.

When policy information is changed through Access Manager or the NetPoint System Console, Access Server caches can immediately be updated by selecting Update Cache. Policy cache timeout becomes important when this feature is not used. Cache timeout should be set to how quickly the policy information needs to be updated when Update Cache is not or cannot be used. The default policy cache timeout is two hours.

If the Access Server machine has sufficient memory to hold all the configured policy information, the maximum elements should be set according to whatever is the maximum of following:

- Total number of policy domains
- Total number of authentication rules, default or otherwise
- Total number of authorization rules, default or otherwise
- Total number of audit rules, default or otherwise

This configuration works for a typical deployment.

Caching User Information

An Access Server's user cache includes information about:

- User Profile, which is required both in actions and rule-based access evaluation
- User's group-membership status, such as whether a user is member of a group or not

The Access Server caches a user's profile information and group membership information. The profile information includes both the information required in actions and the information required for authorization. For example, if the authorization rule for a policy allows access to anyone whose userLevel attribute is set to greater than 3, Access Server caches will include the userLevel attribute.

Similarly, an Access Server caches information about whether or not a user is a member of a group.

User and group information can be updated through the COREid Server or other applications. As user and group information changes, the NetPoint caches become out of date. For this reason, user cache timeout is crucial.

The timeout of the user cache should be proportional to the average time interval in which part of User Profile—those relevant to Access System—changes. Relevant parts of a user profile are:

- Attributes returned to AccessGate in actions
- Attributes used to control access
- Attributes used in dynamic group-membership definitions that may in turn be used to control access

The maximum elements should be equal to the number of different users that may access the system in the *Access Server user cache timeout* period of time.

You can configure the COREid Server to notify the Access Server automatically when user or group information changes. The Access Server's caches are then automatically flushed and updated with new information.

See also:

- "Automatically Flushing Access Server Caches" on page 97
- "Manually Flushing Access Server Caches" on page 98
- "Cache Configuration Using Replicated Directories" on page 99
- "AccessGate Cache Configuration" on page 101

Automatically Flushing Access Server Caches

The COREid System and the Access System use different user and group caches. An administrator may perform any of the following COREid System operations:

- Deactivating a user
- Changing user attributes
- Deleting a group
- Changing a password policy
- Changing redirect URLs

After any of the above operations take place, the directory server is updated with the new information. However, the Access Server may be unaware of these changes. For example, if you deactivate a user in the COREid System, that change should be reflected in the Access System so the user does not have access to its protected resources. To ensure that the Access Server is informed of changes in the COREid System, you can manually flush the Access Server's user cache. Or, you can configure the COREid Server to notify the Access Server of changes to user and group information. The Access Server caches are then automatically flushed and replaced with the latest information.

Note: The user cache is not automatically flushed when changes are made to group membership through a dynamic filter or through static membership.

To flush the Access Server cache automatically

1. Navigate to *COREid_install_dir*/identity/oblix/data/common/basedbparams.xml file

   where *COREid_install_dir* is the directory where COREid is installed.

2. In the basedbparams.xml file, locate the doAccessServerFlush parameter and set it to true.
3. Restart the COREid Server.
4. Add a dummy AccessGate using the configureAccessGate command line tool, as follows:

   ```
   configureAccessGate -i COREid_install_dir/identity/
   AccessServerSDK -t AccessGate
   ```

5. Be sure the Access Management Service for the AccessGate is turned on:

   a) From the Access System Console, click Access System Configuration > AccessGate Configuration > click a link for the appropriate AccessGate > Modify.

   b) Set the Access Management Service to On.

6. Be sure the Access Management Service for the Access Server is turned on:
   a) From the Access System Console, click Access System Configuration > Access Server Configuration > click a link for the appropriate Access Server > Modify.
   b) Set the Access Management Service to On.
7. Restart the Access Server.

Manually Flushing Access Server Caches

If you choose not to implement automatic cache flushing for the Access Server, the Access Server's caches will contain outdated information until the cache timeout occurs. You can, however, manually flush the Access Server's user and password policy caches in the Access System Console. You can flush stored information on a specific password policy or on all password policies from the Access Server cache. You can also flush cached information on all redirect URLs.

To flush the Access Server's user cache manually

1. Launch the Access System Console.
2. Navigate to Access System Configuration > User Access Configuration and click Flush User Cache.
3. To flush a specific user's profile and group information, click Select User.

   The Selector page appears.
4. To search for a user, enter the name and click Go.

   The search results are listed under Selector.
5. Click Add to select a user, or to select all listed users, click Add All.

   The user name is listed under Selected.
6. Click Done to leave the screen.

   The selected user's name appears on the Flush User Cache screen.
7. Click Flush Cache.

   A dialog box requesting confirmation appears.
8. Click OK to remove the user's cached information; click Cancel if you do not want to remove the user's cached information.

To flush the password policy cache manually

1. Launch the Access System Console.
2. Navigate to Access System Configuration > Common Information Configuration and then click Flush Password Policy Cache.

3. If you changed any information for a password policy, select that policy from the drop-down list under Flush All Cached Information for a Specified Password Policy, then click Flush Cache.

   For information about the order of password policy evaluation, see the *NetPoint 7.0 Administration Guide Volume 1*.

4. To flush all of the password policies, or if you deleted a password policy from the COREid System, then click Flush Cache to delete cached information for all password policies.

5. If you changed any of the redirect URLs on the Password Policy Management screen, click Flush Redirect URL.

   For information about configuring the password redirect URLs, see the *NetPoint 7.0 Administration Guide Volume 1*.

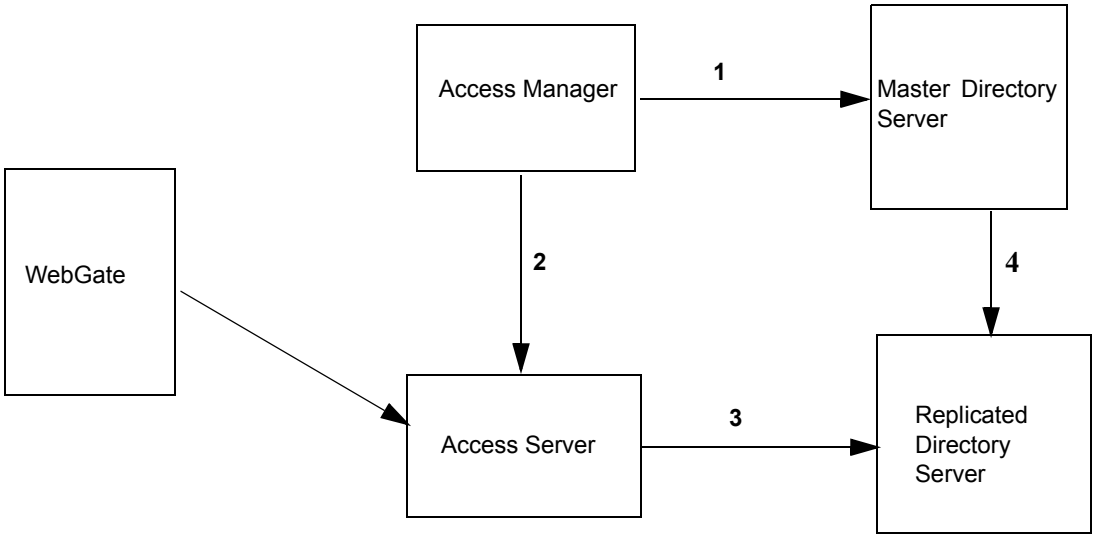
6. Click OK to confirm your decision.

Cache Configuration Using Replicated Directories

When you change data using the Access Manager, it modifies data in the directory and notifies the Access Server to flush its cache. The Access Server flushes its cache and reads the updated information from the directory server.

In a replicated directory environment, modified data is passed from the master directory server to replicated directory servers. There is a time lag before the modified data is reflected in the replicated directory servers. If the Access Manager communicates with the master directory server and the Access Server communicates with a replicated directory server, the Access Server may flush its cache and read data before the modified data is reflected in the replicated server. As a result, the Access Server may cache old data. Thus, in a replicated environment, policy evaluation by the Access Server can be incorrect if it is based on old data. Figure 9 illustrates a replicated environment.

Figure 9 A Replicated Environment



Process overview: Replication

1. The Access Manager modifies data in the master directory server.
2. The Access Manager sends a signal to the Access Server to flush its cache.
3. The Access Server flushes its cache and reads data from the replicated directory server.
4. Modified data is replicated from the master directory server to the replicated directory server.

Timeouts That Ensure Correct Behavior

To ensure that policy evaluation is accurate and that the Access Server reads the latest data, use the splTimeout parameter. The splTimeout parameter enables you to specify a timeout on the element being modified. It allows for the time lag between replication from the master directory server to the replicated directory server. The splTimeout parameter takes precedence over the cache timeouts specified in the Access Server Configuration page of the Access System Console.

The splTimeout value is specified in seconds.

If you do not specify a value for the splTimeout parameter, the Access Server flushes its cache and reads the data as soon as it receives a flush signal from the Access Manager. The splTimeout parameter is in the file located in:

AccessServer_install_dir/access/oblix/apps/common/bin/globalparams.lst

where *AccessServer_install_dir* is the directory where the Access Server is installed.

AccessGate Cache Configuration

When you configure an AccessGate, you can specify the maximum number of elements in the cache as well as the cache timeout.

Each AccessGate caches the following information:

- Details about an authentication scheme, such as:
  - How to challenge a user for credentials
  - Level of the scheme.

    For information about creating authentication schemes and their levels, see the *NetPoint 7.0 Administration Guide Volume 2*.
  - Redirection URL, if any
- Protection information about every resource—including the authentication scheme, URL, and operation—that is passed to the AccessGate, such as:
  - The authentication scheme key
  - The protected resource's URL
  - Whether the resource is protected in the system or not
  - If it is protected, the authentication method required for the resource.
- Details about an authorization scheme, such as:
  - The LDAP user attribute
  - The parameters specified in the authorization rules of the policy domain

When you configure or modify an authentication or authorization scheme, select Update Cache to update the AccessGate cache immediately with the updated scheme.

The number of URLs for which information is cached can be configured for each AccessGate. With out-of-the-box configuration, URL elements timeout every 30 minutes. When you make any policy changes that may change a URL's protection status or authentication method, select Update Cache to update the AccessGate cache immediately. This means the AccessGate's cache timeout period need not be short.

Assuming the system has sufficient memory, the maximum number of URLs cached should be at least equal to the number of distinct URLs processed by AccessGate in the amount of time specified in the Cache Timeout field.

For example http://www.myserver.com and http://myserver are treated as distinct URLs. If information about a URL is not in the cache, AccessGate makes a request to the Access Server. Fetching information from the Access Server takes longer than fetching information from a local cache, but not significantly longer.

5 Migration

After installing and setting up NetPoint for the first time, and configuring the product in a test environment, you need to deploy the product to a wider audience. This process is referred to as migration. Migration consists of moving all of your NetPoint configuration files and data to new servers, installing various NetPoint components on machines in the new environment, and ensuring that the newly created NetPoint system works in the same way as the old one.

This chapter includes the following topics:

Preparing for Migration

Migration refers to moving NetPoint from one area to another, for example from Staging to Production. For migration, you should preserve any customizations that you have made to your NetPoint stylesheets, JavaScripts, and other setup files. You also need to attend to changes that may be required in host IDs, administrators, and other system configuration settings. That way, once NetPoint is running in the new location, it behaves as expected. Note that attention to detail is important. The greater the difference between the source and the target environments, the more details there are to attend to.

The following sections describe the considerations that are required when you are migrating from one environment to another—for example, from Staging to Production.

Task overview: Preparing for migration includes

4. “Copying the Source Environment to the Target” on page 107
5. “Other Migration Considerations” on page 107
6. “Using the NetPoint GUI” on page 109

Preparing Customized Data

This section provides details:

For customized authentication and authorization plug-ins

- Standardize the layout of the file system in all of your environments.

  It is recommended that you locate all NetPoint-specific files in one directory path across all environments.
- Be sure that Web server and directory server versions are the same across all environments.
- Create a /custom directory for all custom code.

  This directory should not be in the installation directory of the NetPoint component. During a re-installation or uninstallation of the component, all subdirectories are deleted.
- Document your changes.

For JavaScript files

- Avoid modifying the main misc.js and miscsc.js files located in the WebPass directory.

  These files are used for client-side processing and are common to all NetPoint components. Any modification can adversely affect all components. If you do modify these files, and you are migrating across environments, remember that the XSL stylesheets include these files. When migrating across environments, you need to modify the newer versions of the files to reflect the changes made to the existing files.
- Create a separate stylesheet that incorporates all of the JavaScript hooks and include that stylesheet in other affected stylesheets.
- Document your changes.

For Identity Event API (PPP) hooks

- Be sure that you are running the same version of the development software in all environments. For example, if you are using a Java or PERL executable, make sure that the versions of the PERL interpreter and JVM are the same. This is also true for C compilers.
- You cannot migrate oblixpppcatalog.lst because it contains references to specific workflow IDs that may or may not be present in the new environment.
- Document the changes.

For XSL stylesheets

- Create a separate instance of the stylesheets before modifying anything.

  To do this, define a new stylesheet that contains all of your new styles.
- If you are modifying the client-side images, styles, and so on, all of these changes must go into all of the affected WebPass instances.

See the *NetPoint 7.0 Customization Guide* for details.

Preparing for Directory Changes

When migrating across environments, you need to be sure that directory data is updated in the new environment:

- Be sure you have at least one WebPass, one COREid Server, and one Access Server running in the new environment.
- Create and export the LDIF of the configuration and policy branches of the source directory.

  If you already have an existing NetPoint deployment in the target environment, you can use a tool to compare the differences between the directories and manually configure the new directory. There are tools on the market that can help you compare differences between LDIFs.
- Clean up the LDIF.

  NetPoint creates sync records to synchronize the Access System with the latest changes. In general you can delete records of type oblixSyncRecord under the NetPoint node. Do not remove records that were created since the last policy cache timeout.
- Compare all aspects of your old and target directories.

  Be aware of differences in DIT structure, schemas, and replication methods for the Oblix subtree. “Directory Data Details” on page 113 describes the schema objects that you may want to review.

Some guidelines for comparing LDIFs between the source and target environments:

- Be sure that all of the differences between source and target LDIFs are appropriate (for example, name changes).
- Compare isolated blocks (for example, by OUs) to simplify making the comparisons.

  This can indicate what blocks of data need to be updated. For example, you may find that it is not useful to migrate Access Server configuration information, but it is useful to migrate policy domains and authentication schemes.

Noting Differences Between Source and Target Environments

When migrating from one environment to another, be aware of the following differences between the two environments:

- Differences in the sharing of configuration information between the Access System and the COREid System
- Differences in who is given administrative privileges, including NetPoint Administrator, Delegated Access Administrator, and Delegated Identity Administrator
- Names and implementation details of the COREid Servers
- Names and implementation details of the WebPass instances
- Names and implementation details of the Access Servers
- Names and implementation details of the WebGates, including changing what Access Server the WebGate points to
- Definitions for Host Identifiers and ipValidationExceptions
- Definitions for authentication schemes, including Challenge Redirect parameters.
- Definitions for authorization schemes
- Definitions for policy domains, including all redirect URLs defined in authentication and authorization actions
- Directory details such as machine name and port number
- Users and groups involved in policy domains

If there are differences in any of these items, make the necessary changes before you migrate the NetPoint configuration.

Note: The section "Directory Data Details" on page 113 provides details on how to identify directory entries for NetPoint components.

Copying the Source Environment to the Target

Once you are done synchronizing the two environments, you should:

- **On Unix**—tar the directories that hold the custom NetPoint files, codes, and updated styles, keeping the directory structure and permissions intact.
- **On Windows**—Zip all the source files while keeping the relative paths intact.

You can then untar or unzip the files and install them in the new environment.

Note: You should use the default style when you first migrate the styles. After copying all the stylesheets, you can switch to the custom style. This prevents a problem with the COREid System pointing to the default style, and the migrated stylesheets pointing to a custom style.

When these steps are done, you can restart all the servers, in this order:

- Directory servers
- COREid and Access Servers
- Web servers

Other Migration Considerations

Be aware of these other points when migrating:

- If you are running in Simple mode, by default, once a year both the COREid and Access Servers need to refresh the self-signed certificate that NetPoint generates to provide SSL encryption.

  The frequency of updates can be configured.
- Update the Simple certificates for the Web server running the WebPass.
- Update the certificate required for Access Manager to work in Simple mode.
- Be sure the obSSOCookie is not shared between machines in the two different environments.

  This would allow a user who exists in both environments to use single sign-on from the source environment. Updating the obSSOCookie is an important security precaution when migrating across environments.

Important: Other ways of breaking SSO between different environments include changing domains, for instance, changing from dev.company.com to staging.company.com and updating the shared secret among different environments.

- Be careful if you decide to recycle a COREid Server Instance Name

  Under certain circumstances, you may want to use an existing COREid Server name. For example, if you are rolling NetPoint out from a test environment to a production environment or if you need to remove a COREid Server for some reason.

  If you do not delete the COREid Server name from the System Console, a login following setup may result in the message "Application has not be set up". Special steps must be taken to ensure you can set up the application and login. The steps below presume that you have another COREid Server and WebPass setup within the same installation.

To recycle a COREid Server instance name

1. Delete the COREid Server name in the directory server under:

   Oblix > Policies > WebResrcDB > name

2. Re-run COREid Server setup, as described in "Setting Up the COREid System" on page 113.

3. Go to the COREid System Console, delete the inoperable COREid instance.

   For example:

   COREid System Console > Sys Admin > System Configuration > Configure COREid Server

   name > Delete

4. From the List all COREid Servers page, re-create the instance using the same ID as described in the *NetPoint 7.0 Administration Guide Volume 1*.

   For example:

   Add
   Name
   Host name
   Port
   Transport Security

Using the NetPoint GUI

As an alternative to cleaning up the LDIF manually, you can also delete key entries from the directory and re-configure your migrated NetPoint system using the COREid and Access System GUIs. Whether you choose to migrate by editing the data in the directory or by using the NetPoint GUI is, ultimately, a matter of personal preference.

The following sections describe a migration methodology that consists of a minor amount of directory tweaking and using the NetPoint GUI to accomplish most of the migration-related updates.

Preparing LDAP Data

The following procedure describes how to migrate LDAP data from a source environment to a target environment.

To prepare LDAP data in the source environment

1. Disable the source system's WebGate so that it no longer protects the COREid and Access Systems.

   Go to the Access System Console > Access System Configuration > AccessGate Configuration to perform this task.

2. Create an LDIF of the NetPoint configuration and policy data in the source system.

   Depending on how you configured your NetPoint installation, these might be separate LDIF files.

3. Enable the WebGate you disabled in step 1.

Preparing the Target Environment

To prepare the target environment, you need to back up data and remove specific entries from the directory.

To prepare the target environment

1. Shut down all NetPoint components that you installed on the target environment.
2. Back up any directories containing NetPoint configuration, user, and policy data.
3. Remove the top node of the Oblix directory tree.
4. Import the LDIF files you created in your source environment to the corresponding locations in the target environment.

5. Remove the following entries under WebResrcDB from the configuration data in your LDAP directory on the target environment:
   - In the object class obPolicyContainerId=WebResrcDB, locate the entry for WebPass. The relative distinguished name (RDN) for this entry is the ID supplied when the WebPass was installed. The object class to look for is oblixWebPassConfigInfo. Delete all of these entries.
   - Under WebResrcDB, locate the entry for the COREid Server. The cn for this entry is the ID that was supplied when installing the COREid Server. The object class to look for is oblixOISServerConfigInfo. Delete all of these entries.
   - Under WebResrcDB, locate an entry with a time-stamp for obName, for example, obName=2003072115T16221897. This entry connects the WebPass and COREid components. The object class is oblixOISServerIDNode. Delete all of these entries. For every WebPass instance that you deleted there will be a corresponding entry with a time-stamp.
6. Locate the entry for obContainerID=DBAgents.

   The entries under DBAgents represent directory server profiles. Delete all of these entries.
7. Remove the entry for cn=cookieEncryptionKey,obContainerId=encryptionKey
8. Remove all workflow instances located under obContainerId=workflowInstances
9. Back up the LDAP directory in your target environment.

Configuring the Target Environment

Configuring the target environment consists of ensuring that each NetPoint component has the correct information to run properly on new hosts.

To reconfigure COREid Server and WebPass

1. If a WebGate has been installed in the target environment, make a copy of the server configuration file.

   For example, on Apache you would archive the http.conf file, and on Sun (formerly iPlanet) you would archive the obj.conf file.
2. Reconfigure the COREid Server, as follows:
   - Back up ois_server_config.xml.bak if it exists.
   - Back up the following files if they exist: setup.xml, configInfo.xml, ois_server_config.xml and remove them from their original directory.
3. Stop the Web server running WebPass.

4. Restart the COREid Server.
5. Restart the Web server running WebPass.
6. Re-run WebPass setup from the COREid System Console.
   a) Go to *WebPass_server:port*/identity/oblix/ and select COREid System Console.
   b) Select Setup and follow the setup script.
   c) De-select any auto-configure options.

Note: When configuring attributes, select Next on all screens. This information should already be filled in.

To reconfigure the Access System

1. Reconfigure the Access Manager.
   - If it exists, copy the setup.lst file to setup.lst.bck and remove the original setup.lst file.
   - Restart the Web server running the Access Manager.
2. Set up the Access Manager.
   a) Got to *WebPass_server:port*/access/oblix and select the Access System Console.
   b) Select Setup and complete the setup script.
   c) Be sure you uncheck the auto-configure options.

Note: When configuring the attributes, click Next on all steps. This information should already be filled in.

Once setup is complete, you can configure additional components such as COREid Servers, WebPass instances, AccessGates, and Access Servers. You may want to test to be sure that all components are working as expected. You may also want to back up your directory data.

To configure the WebGate

1. *WebPass_server:port*/access/oblix and select Access System Console > Access System Configuration.
2. Add a new Access Server instance.

   If you receive an error stating the connection to the requested server failed, you can ignore it for now.
3. Select AccessGate Configuration and create AccessGate and WebGate instances.

To clean up configuration data

1. If needed, reconfigure the other WebGates to point to the new Access Server.
2. Remove the old Access Server instance using the Access Server Console.
3. Disable all policies.
   a) Go to *WebPass_server:port*/access/oblix/.
   b) Select the Access Manager.
4. From the Access System Console, select Access System Configuration, and complete the tasks below:
   a) Authentication Management > *link*, then change the Challenge Redirect parameters to the appropriate host names and ports in all authentication schemes.
   b) Select Host Identifiers, then ensure the host identifier names map to the correct host name and port in the target environment.

   See the *NetPoint 7.0 Administration Guide Volume 2* for details.
5. From the Access Manager, check all policy domains, including redirect URLs defined in all authentication and authorization actions
6. Enable all policy domains used in your environment.
7. Copy any files required for the server, such as login forms used for form-based authentication or custom authorization and authentication plug-ins.
8. Configure additional Access Servers and AccessGates for your target environment.

To complete COREid System configuration

1. Create additional COREid Servers, as needed.
2. Create additional WebPass instances, as needed.

To finish migration

1. Install any additional components needed, including Access Servers, WebGates, WebPass, and so on.
2. Start the Access Servers.
3. Start the Web servers running WebPass and WebGate.
4. Install any other components, such as Access Server SDK.

Directory Data Details

When migrating directory information across different environments—for instance, from development to production—certain types of data must change. This data includes server names, IP addresses, administrative user information, and other data that is different in the new environment. Simple LDIF imports and exports solve most of the directory migration issues. However, some manual configuration is necessary to ensure proper operation after migration. The following sections outline the types of directory data that are migrated.

The following sections give a concise overview of the objects and attributes of interest when you are migrating environments. For details on each of these objects and attributes, see the *NetPoint 7.0 Schema Description*.

Note: Workflow records must be removed from the LDIF before you import directory data to the new environment. These records are written to the oblixWorkflowInstance object class.

The following discussions will be of interest:

- "Unchanged Data" on page 113
- "Encryption Keys" on page 114
- "COREid Data" on page 115
- "Access System Data" on page 124

Unchanged Data

A number of objects need to be copied *as is* from the source to the target system. These unchanged objects belong to the following object classes (obclass=):

- oblixOrgPerson
- oblixLocation
- oblixAuxLocation
- oblixAdvancedGroup
- oblixSiteDomain
- oblixAuthenticationPolicy
- oblixChallengeScheme
- oblixResourceOperationRule
- oblixWebResourceAuxClass
- oblixUrlPrefix

- oblixPolicyRule
- oblixWRSCAction
- oblixAuditPolicy
- oblixCustomAuthzCondition
- All object classes under obapp=WRSC
- All object classes under obname=SDSearchColumnList
- All object classes under obname=WRORSearchColumnList

Encryption Keys

Encryption keys must be updated when you migrate across environments. These can be initially transferred *as is*, and modified from the NetPoint Access System Console. For more information on shared secret and encryption keys, see the *NetPoint 7.0 Administration Guide Volume 1*.

The following is an example LDAP URL for extracting the encryption keys:

```
ldap://server:port/
obContainerId=encryptionKey,o=Oblix,ou=Apps,o=oblix.net??
sub?(o=objectclass=*)
```

The following is an example encryption key object:

```
dn: obContainerId=encryptionKey,o=Oblix,out=Apps,o=oblix.net
objectClass: oblixContainer
objectClass: top
obContainerId: encryptionKey


dn: cn=cookieEncryptionKey,obContainerId=encryptionKey,
o=Oblix,ou=Apps,o=oblix.net
cn: cookieEncryptionKey
objectClass: top
objectClass: oblixEncryptionKey
obSharedSecret: m9+XBn7aVladIkLiJ/
qfIPKjoPcS5KlpFnA7ClPECi69KXUB3EUaSU4myFYD1UmXHFJsgcC+tVBBCa9HmWc1
HumonYoCqA1cTbXwtIr8S7aEgmY7K9zGhegV6Cjwa6RGsMWYSqfIPKjoPcS5KlpFnA
7ClPECi69KXUB3EUaSU4myFYD1UmXHFJsgcC+tVBBCa9HmWc1HumonYofr1CqA1cTb
XwtIr8S7aEgmY7K9zGhegV6Cjwa6RGsMWYSYwuf9ycJYOaw==
obSecretSize: 256
```

COREid Data

COREid data consists of three types of directory entries:

- Configuration data
- Run-time data
- Dynamic data

Table 5 summarizes the different types of directory entries:

Table 5 Different Types of Directory Entries

Configuration Data	Runtime Data	Dynamic Data
• Object class definitions • COREid Servers • WebPasses • Directory Options • Administrators • Server Settings • Auditing Policies	• Workflow configurations • Attribute-level ACLs • searchbases • User Manager profiles • Group Manager profiles • Organization Manager profiles	• Workflow tickets

Object Class Definitions

COREid object class definitions consist of two or more objects. The first object is the top object. For instance, the following is the definition for inetOrgPerson:

```
dn: obclass=inetOrgPerson,o=Oblix,ou=Apps,ob=oblix.net
obReady: True
obVer: 7.0
obClass: inetOrgPerson
objectClass: top
objectClass: OblixClass
obClassKind: Structural
obClassType: personClass
obClassAttr: cn
```

The following is an example LDAP URL for extracting the top object:

```
ldap://server:port/
o=Oblix,ou=Apps,o=oblix.net??one?(objectclass=OblixClass)
```

An object class definition consists of objects with meta-definition of the object, including its display types, semantic types, and so on. The following is an example LDAP URL for extracting the object class meta definition:

```
ldap://server:port/
obClass=CLASS_NAME,o=Oblix,ou=Apps,o=oblix.net??sub?(!(objectclass
=oblixClass))
```

An example of an object class definition:

```
dn: obattr=cn,obclass=inetorgperson,o=oblix,ou=apps, o=oblix.net
obAttr: cn
obVer: 7.0.0
objectClass: top
objectClass: oblixMetaAttribute
obDisplayName: Full Name
obSemanticType: obSName
obCardinality: multi
obDisplayType; obDTextS
obSearchable: True
obVisible: True

dn: obattr=givenName,obclass=inetorgperson,o=oblix,ou=apps,
o=oblix.net
obAttr: givenName
obVer: 7.0.0
objectClass: top
objectClass: oblixMetaAttribute
obDisplayName: First Name
obSemanticType: obSName
obCardinality: multi
obDisplayType; obDTextS
obSearchable: True
obVisible: True
```

The following attributes are unchanged (obpanelid=) in the COREid System:

- monitorTable
- ticketTable
- wfProfileTopPanel
- wfProfileLowerPanel
- ticketInfoTable

COREid Servers

COREid Server information is stored in the class oblixOISServerConfigInfo. The following is an example LDAP URL for extracting the COREid Server information:

```
ldap://server:port/
obPolicyContainerId=WebResrcDB,obContainerId=Policies,
o=Oblix,ou=Apps,o=oblix.net??one?
(objectClass=oblixOISServerConfigInfo)
```

WebPass Instances

WebPass instances consist of two or more objects:

- The first is the WebPass definition, stored in the object class oblixWebPassConfigInfo.
- The second aspect of the definitions consists of the primary and secondary failover configuration. Pointers to this information are stored in the attributes obOISPrimaryServerID and obOISSecondaryServerID. The object class oblixOISServerIDNode contains this information.

The following is an example LDAP URL for extracting the WebPass definition object:

```
ldap://server:port/
obPolicyContinerId=WebResrcDB,obContainerId=Policies,
o=Oblix,ou=Apps,o=oblix.net??one?
(objectClass=oblixWebpassConfigInfo)
```

The following is an example LDAP URL for extracting the WebPass primary and secondary failover information:

```
ldap://server:port/
obPolicyContainerId=WebResrcDB,obContainerId=Policies,
o=Oblix,ou=Apps,o=oblix.net??one?(&(obname=OBJECT_NAME)
(objectClass=oblixOISServerIDNode))
```

where *OBJECT_NAME* is the WebPass identifier.

Directory Options

There are three data elements for directory options:

- Default directory server information is stored in oblixDBProfile.
- Directory profiles stored with a different obname than the default server in the oblixDBProfile object class.
- Disjoint searchbases are simple attribute values on a specific object, such as groupOfUniqueNames.

The following is an example LDAP URL for extracting the default directory server information:

```
ldap://server:port/
obName=default,obContainerId=DBAgents,o=Oblix,ou=Apps,
o=oblix.net??base?(objectClass=oblixDBProfile))
```

The following is an example LDAP URL for extracting the directory profiles:

```
ldap://server:port/
obName=certificateServer,obContainerId=DBAgents,o=Oblix,
ou=Apps,o=oblix.net??one?(&(objectClass=oblixDBProfile)
(!obname=default)))
```

The following is an example LDAP URL for extracting the disjoint searchbases:

```
ldap://server:port/
o=Oblix,ou=Apps,o=oblix.net??base?(objectClass=*)
```

Administrators

There will be directory entries for your directory administrators and Web masters.

The following is an example LDAP URL for extracting the directory administrators:

```
ldap://server:port/cn=Directory
Administrators,o=Oblix,ou=Apps,o=oblix.net??base? (objectClass=*)
```

The following is an example LDAP URL for extracting the Web Masters:

```
ldap://server:port/cn=Web
Masters,o=Oblix,ou=Apps,o=oblix.net??base?(objectClass=*)
```

Server Settings

These include settings for email feedback accounts, SMPT server settings, and session timeout values. A single object defines all of these items.

The attributes obFeedbackEmail, obWebMasterEmail, and obUserSessionTimeout are stored on this object.

The following is an example LDAP URL for extracting information for all server settings:

```
ldap://server:port/
o=Oblix,ou=Apps,o=oblix.net??base?(objectClass=*)
```

Auditing Policies

Audit policies are stored in the object classes oblixMasterAuditPolicy and oblixAuditPolicy. The following is an example LDAP URL for extracting information for auditing policies:

```
ldap://server:port/
obname=MasterAuditPolicy,obPolicyContainerId=WebResrcDB,
obContainerId=Policies,o=Oblix,ou=Apps,o=oblix.net??base?
(objectClass=*)
```

Password Policies

Password policies are stored in the object class oblixPasswordPolicy. The following is an example LDAP URL for extracting information for password policies:

```
ldap://server:port/
obContainerId=password,o=Oblix,ou=Apps,o=oblix.net??one?
(objectClass=oblixPasswordPolicy)
```

Workflow Configurations

Workflow configurations are as follows:

- Workflow definitions are stored in the object class oblixWorkflow.
- Workflow definition details are stored in the object classes oblixWorkflowTarget and oblixWorkflowStep.
- Workflow participant definitions consist of a placeholder for the participant rule, with a flag to indicate if it is enabled or disabled (in the obPolicyRuleEnabled attribute). These entries contain oblixPolicyContainerId=WorkflowDB in their DN and have attributes to link back to the workflow *definition* entry.
- Workflow participant condition entries store the criteria for who can participate in the workflow. These entries contain oblixPolicyContainerId=WorkflowDB in their DN.
- Workflow particpant rules—These can be one or more entries that describe what workflow the rule applies to. Each entry contains an attribute called obWorkflowName to point to the workflow definition entry for this rule. Each entry also contains an obPolicyReulName to point to the workflow participant *condition* entry. The object class for these rules is oblixWorkflowResourceAuxClass.

The following is an example LDAP URL for extracting information for workflow definition entries:

```
ldap://server:port/
obContainerId=workflowDefinitions,o=Oblix,ou=Apps,
o=oblix.net??one?(objectClass=oblixWorkFlow)
```

The following is an example LDAP URL for extracting information for workflow definition *detail* entries:

```
ldap://server:port/
obWorkflowId=WF_ID,obContainerId=workflowDefinitions,
o=Oblix,ou=Apps,o=oblix.net??sub?(objectClass=*)
```

where *WF_ID* is the value of the attribute obWorkflowId from the *parent* workflow definition entry.

The following is an example LDAP URL for extracting information for workflow *participant definition* entries:

```
ldap://server:port/
obPolicyContainerId=WorkflowDB,obContainerId=Policies, o=Oblix,
ou=Apps,o=oblix.net??one? (objectClass=oblixPolicyRule)
```

The following is an example LDAP URL for extracting information for workflow *participant condition* entries:

```
ldap://server:port/obName=PD_ID,obPolicyContainer=WorkflowDB,
obContainerId=Policies, o=Oblix,ou=Apps,o=oblix.net??sub?
(objectClass=oblixPolicyCondition)
```

where *PD_ID* is the value of the obName attribute in the workflow *participant definition* entry, above.

The following is an example LDAP URL for extracting information for workflow *participant operation rule* entries:

```
ldap://server:port/obName=PD_ID,obPolicyContainer=WorkflowDB,
obContainerId=Policies, o=Oblix,ou=Apps,o=oblix.net??sub?(&
(objectClass=oblixResourceOperationRule)
(obWorkflowName=obWorkflow=WF_ID,
obContainerId=workflowDefinitions,o=oblix,ou=Apps, o=oblix.net))
```

Attribute Read and Write Privileges

Attribute-level access control rules are stored as a number of entries in the directory.

- All attribute access controls are defined in entries with obPolicyContainerId in their DNs. The possible values for obPolicyContainerId are obGroupDB, obObjectDB, and UserDB. Example:

  ```
  dn: obname=C233409809898,obPolicyContainerId=UserDB,
  obContainerId=Policies,o=Oblix,ou=Apps,o=oblix.net
  ```

  ```
  objectClass: Top
  ```

```
objectClass: oblixPolicy Rule
obName: P222222
obPolicyRuleConditionListType: 2
obPolicyRuleEnabled: True
obPolicyRulePriority: 1
```

- For each access control, in addition to the definition entry there is a condition entry that stores the criteria for determining who the ACL applies to. This entry contains obPolicyContainerId of obGroupDB, obObjectDB, or UserDB, and and the value of the obName attribute in the attribute access control definition entry. Example:

  ```
  dn: obname=P222222,obname=C233409809898,
  obPolicyContainerId=UserDB,obContainerId=Policies,
  o=Oblix,ou=Apps,o=oblix.net
  objectClass: oblixPolicyCondition
  obName: P222222
  obPolicyConditionOrder: 1
  obPolicyConditionUsage: Allow
  obPolicyConditionRole: ob_self
  ```

For each access control, there is an operation rule that define the attribute that the access control applies to. These are defined in the object class oblixResourceOperationRule.

The following is an example LDAP URL for extracting information for an ACL definition entry:

```
ldap://server:port/
obPolicyContainerId=CONTAINER_TYPE,obContainerId=Policies,
o=Oblix,ou=Apps,o=oblix.net??one? (objectClass=oblixPolicyRule)
```

where *CONTAINER_TYPE* may be obObjectDB, UserDB, or ObGroupDB.

The following is an example LDAP URL for extracting information for an ACL condition:

```
ldap://server:port/
obName=ACL_ID,obPolicyContainerId=CONTAINER_TYPE,
obContainerId=Policies,o=Oblix,ou=Apps,o=oblix.net??sub?
(objectClass=oblixPolicyCondition)
```

where *CONTAINER_TYPE* may be obObjectDB, UserDB, or ObGroupDB, and *ACL_ID* is the value of the attribute obName in the ACL definition entry described above.

The following is an example LDAP URL for extracting information for an ACL operation rule:

```
ldap://server:port/
obPolicyContainerId=CONTAINER_TYPE,obContainerId=Policies,
o=Oblix,ou=Apps,o=oblix.net??one?(&
(objectClass=oblixResourceOperationRule)
(obPolicyRuleName=obName=ACL_ID,
obPolicyContainerId=CONTAINER_TYPE,obContainerId=Policies,
o=Oblix,ou=Apps,o=oblix.net))
```

where *CONTINER_TYPE* may be obObjectDB, UserDB, or obGroupDB, and *ACL_ID* is the value of the attribute obName in the ACL definition entry.

Searchbases

Searchbases are defined by two types of LDAP entry:

- A single entry that serves as a high-level searchbase definition. This entry is defined using the object classes oblixResourceOperationRule and oblixUserResourceAuxClass.
- The actual searchbase conditions are stored in entries containing a DN with the value of the attribute obPolicyRuleName of the high-level searchbase definition.

The following is an example LDAP URL for extracting information for the high-level searchbase definition:

```
ldap://server:port/
obPolicyContainerId=UserDB,obContainerId=Policies,o=Oblix,
ou=Apps,o=oblix.net??one?(&
(objectClass=oblixResourceOperationRule)
(objectClass=oblixUserResourceAuxClass))
```

The following is an example LDAP URL for extracting the actual searchbase conditions:

```
ldap://server:port/DISTINGUISHED_NAME??sub?(objectClass=*)
```

where *DISTINGUISHED_NAME* is the value of the attribute obPolicyRuleName in the high-level searchbase definition.

Workflow Tickets

A workflow ticket is defined by two or more entries:

- The ticket definition created in the oblixWorkflowInstance object class.
- Additional workflow ticket data, such as attribute values and status. These entries are child entries to the ticket definition.

The following is an example LDAP URL for extracting the workflow ticket definition:

```
ldap://server:port/
obContainerID=workflowInstances,o=Oblix,ou=Apps,o=oblix.net??one?(
objectClass=oblixWorkflowInstance)
```

The following is an example LDAP URL for extracting the additional workflow ticket data:

```
ldap://server:port/
obWFInstanceId=WF_INSTANCE,obContainerId=workflowInstances,o=oblix
,ou=Apps,o=oblix.net??sub?(objectClass=*)
```

where *WF_INSTANCE* is the value of the attribute obWorkflowId in the workflow ticket definition object.

User, Group, and Organization Manager Panels

These entries define the panels in the COREid applications. They are components of the obApp entries for userServCenter, groupServCenter, and objServCenter. They are identified by the following object classes:

- oblixPanel
- oblixTabPanel
- oblixLocation

Additionally, some entries may be in the oblixMetaAttribute object class with these attributes:

- obPanelType
- obLocationName
- obLocationTitle
- obPanelID
- obParentLocationDN
- obRectangle
- obPhoto

User Manager Example

The following is an example LDAP URL for extracting the information on User Manager panels:

```
ldap://server:host/
obApp=userServCenter,o=Oblix,ou=Apps,o=oblixl.net??sub?
(objectClass=*)
```

Group Manager Example

The following is an example LDAP URL for extracting the information on Group Manager panels:

```
ldap://server:host/
obApp=groupServCenter,o=Oblix,ou=Apps,o=oblixl.net??sub?
(objectClass=*)
```

Organization Manager Example

The following is an example LDAP URL for extracting the information on Organization Manager panels:

```
ldap://server:host/
obApp=objServCenter,o=Oblix,ou=Apps,o=oblixl.net??sub?
(objectClass=*)
```

Access System Data

The Access System LDAP entries are categorized in Table 6:

Table 6 Access System LDAP Entries

Configuration Data	Run-time Data	Dynamic Data
• Master Web resource administrators • Resource type definitions • Host identifiers • Access Servers • Access clients • Authentication schemes • Authorization schemes • Auditing policies	• Policy domain	• Cache update requests

Master Web Resource Administrators

These entries describe the Access Administrators. Example:

```
dn: cn=Master Web Resource Admins,obapp=PSC,o=Oblix,
ou=apps,o=oblix.net
objectClass: groupOfUniqueNames
objectClass: top
objectClass: oblixGroupOfUniqueNames
obUniqueMember: uid=jdoe,ou=People,o=oblix.net
obVer: 7.0
cn: Master Access Administrator
```

The following is an example LDAP URL for extracting the information on Master Web Resource Administrators:

```
ldap://server:port/cn=Master Web Resource
Admins,obApp=PSC,o=Oblix,ou=Apps,o=oblix.net??base?
(objectClass=*)
```

Resource Type Definitions

These entries belong to the object class oblixResourceType. The following is an example LDAP URL for extracting the information on resource type definitions:

```
ldap://server:port/obContainerId=URI
Resources,obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?
(objectClass=oblixResourceType)
```

Host Identifiers

These entries belong to the object class oblixHostId. The following is an example LDAP URL for extracting the information on host identifiers:

```
ldap://server:port/obapp=PSC,o=Oblix,ou=apps,o=oblix.net??one?
(objectClass=oblixHostId)
```

Access Servers

These entries belong to the object classes oblixAAAServerConfigInfo and oblixAAAEngineConfig. The following is an example LDAP URL for extracting the information on Access Servers:

```
ldap://server:port/obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?(&
(objectClass=oblixAAAServerConfigInfo)
(objectClass=oblixAAAEngineConfig))
```

Access Clients

Access clients consist of two or more objects:

- The actual AccessGate, defined by the object class oblixWebgateConfigInfo.
- Primary and secondary failover information. The attributes obAAAPrimaryServerId and obAAASecondaryServerid in the Access client definitions will point at these objects. They will belong to the oblixAAAServerIDNode object class.

The following is an example LDAP URL for extracting the information on the Access Client definition object:

```
ldap://server:port/obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?
(objectClass=oblixWebGateConfigInfo)
```

The following is an example LDAP URL for extracting the information on the primary and secondary failover configuration:

```
ldap://server:port/obApp=PSC,o=Oblix,ou=Apps,o=oblix.net??one?(&
(obName=OBJECT_NAME)(objectClass=oblixAAAServerIDNode))
```

where *OBJECT_NAME* is any name that corresponds to the primary and secondary server IDs.

Authentication and Authorization Schemes

These belong to the oblixChallengeScheme object class. The following is an example LDAP URL for extracting the information on authentication schemes:

```
ldap://server:port/obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?
(objectClass=OblixChallengeScheme)
```

The following is an example LDAP URL for extracting the information on authorization schemes:

```
ldap://server:port/obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?
(objectClass=OblixAuthzPluginScheme)
```

Auditing Policies

These belong to the oblixAuditPolicy and oblixMasterAuditPolicy object classes.

The following is an example LDAP URL for extracting the information on auditing policies:

```
ldap://server:port/obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?(&
(objectClass=oblixAuditPolicy)
(objectClass=oblixMasterAuditPolicy))
```

Cache Update Requests

These belong to the oblixGSN object class. Cache update messages should not be synchronized, except for the *last* sequence number. There are two objects that should be considered when migrating:

- The last cache update sent in the system is stored on a special object, but it needs to have a shadow record with the description of the update. This record is in the oblixSynchRecord object class.
- The individual cache update notification message created as an entry of the oblixSynchRecord object class

The following is an example LDAP URL for extracting the information on the last cache update:

```
ldap://server:port/obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?
(objectClass=oblixGSN)
```

The following is an example LDAP URL for extracting the information on the cache update notification message:

```
ldap://server:port/
cn=PSCMgmt,obApp=PSC,o=Oblix.ou=apps,o=oblix.net??one?(&
(obSyncRequestNo=*)(objectClass=oblixSynchRecord))
```

Policy Domains

Policy Domains consist of multiple objects organized in a single two-level LDAP branch where the policy domain definition is the top of the branch, and the child objects define the behavior of the policy domain. These objects belong to the oblixSiteDomain object class.

The following is an example LDAP URL for extracting the information on the policy domain:

ldap://*server*:*port*/obApp=PSC,o=Oblix,ou=apps,o=oblix.net??one?(objectClass=oblixSiteDomain)

For each policy domain, the following rule outputs the elements that make up the domain:

```
ldap://server:port/
obName=SITE_DOMAIN_NAME,obApp=PSC,o=Oblix,ou=apps,
o=oblix.net??sub?(objectClass=*)
```

where *SITE_DOMAIN_NAME* is the value of the attribute obName that serves as the unique identifier for the policy domain parent object.

Upgrading NetPoint

Upgrading refers to bringing your older NetPoint environment up to a more recent version of the NetPoint product. The *NetPoint 7.0 Upgrade Guide* provides a detailed procedure for running upgrade scripts.

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X