

CA Identity Manager

High Availability Guide

r12



This documentation and any related computer software help programs (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the Documentation for their own internal use, and may make one copy of the related software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the Product are permitted to have access to such copies.

The right to print copies of the Documentation and to make a copy of the related software is limited to the period during which the applicable license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

EXCEPT AS OTHERWISE STATED IN THE APPLICABLE LICENSE AGREEMENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of any product referenced in the Documentation is governed by the end user's applicable license agreement.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2008 CA. All rights reserved.

CA Product References

This document references the following CA products:

- CA Identity Manager
- eTrust Directory

Contact Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at <http://ca.com/support>.

Contents

Chapter 1: High Availability Overview	9
Purpose of High Availability	9
Identity Manager Server Architecture	10
Provisioning Components Architecture	12
 Chapter 2: Installation, Upgrade, and Migration	 17
How to Install Identity Manager for High Availability	17
How to Upgrade a High Availability Installation	18
Upgrade an Identity Manager Cluster	18
Upgrade Highly Available Provisioning Components	19
How to Migrate a Standalone Installation to a Cluster	22
 Chapter 3: Policy Server Cluster	 25
Configure a Policy Server Cluster	25
Migrate the Policy Store	25
 Chapter 4: Identity Manager Cluster	 29
Application Server Information	29
WebLogic Clusters	30
WebLogic 8.1 Clusters	30
WebLogic 9.2 Clusters	37
JBoss Clusters	47
Test the Default Multicast Address	47
Create the First Cluster Node	48
Add Cluster Nodes	51
WebSphere Clusters	52
WebSphere Cluster Prerequisites	54
Create a WebSphere Cluster	54
Configure the Cluster with the First Node	56
Add Nodes to the Cluster	56
Install Identity Manager	57
Create Core Group Policies	59
Update Cluster Members	60
Create a Message Store	62
SiteMinder High Availability for the Application Server Cluster	63

Modify Policy Server Connection Settings	64
Add More Policy Servers	65
Select Load Balancing or Fail Over	65
Load-Sharing Limitations	65
Policy Server and Application Server Communication	66
Migrate Workflow Process Definitions	67
Export process definitions	68
Import process definitions	68
Web Server Clusters	69
Configure the Proxy Plug-In (WebLogic)	69
Configure the JK Connector (JBoss)	70
Configure the Proxy Plug-In (WebSphere)	71
Cluster Startup	72
Start the Servers for WebLogic	72
Start the Servers for JBoss	73
Start the Cluster for WebSphere	74
Verify the Clustered Installation	75
Update the Migrated Environments	75

Chapter 5: Provisioning Components **77**

Load-Sharing Limitations	77
Preconfiguration Considerations	78
Planning	78
DSA Configuration Considerations	79
Global Naming Convention	80
Consistent Port Assignments	85
Sample Provisioning Configuration	86
DSA Configuration Example	86
Port Assignment Example	87
Primary and Alternate Provisioning Servers	88
Provisioning Domain Specifications	88
Client-Tier	89
Server Tier	89
Provisioning Directory Tier	89
ROOT Domain Installation	90
Domain Prerequisites	91
How to Make Provisioning Clients Highly Available	93
Enable Client-Tier Failover for Provisioning Manager	94
Test the Client-Tier Failover Configuration	95
How to Make Provisioning Servers Highly Available	95
Use SiteMinder for High Availability	95
How to Use DXLink Routers for High Availability	96

Provisioning Server Disaster Recovery	101
How to Make Provisioning Directories Highly Available	101
Replicate the ROOT Domain Provisioning Directory	103
How to Reconfigure DSAs on the Primary Domain Server	106
Configure the Alternate Domain Server Provisioning Directory on ROOT02	114
Optimize Performance for the Provisioning Directory	118
Provisioning Directory Security Settings	119
How to Make Connector Servers Highly Available	121
Install Connector Servers	121
Deployment of the Connector Server Framework	121
Deployment Examples	123
Multi-Platform Installations	124
Configuring the Connector Server Framework	125
Load-Balancing and Failover	131

Chapter 6: Provisioning Directory Monitoring 135

Determine Monitoring Thresholds	135
Monitoring Log Files	136
Perform Backup and Restore	136
Observe Router Traffic Using DXconsole	136
Stop Local Read-Only DSAs to Synchronize Local Multiwrite DSA	137
Enable SSL Encryption	138

Appendix A: DSA Port Assignments 139

Provisioning Server DSA	139
Provisioning Directory DSA	139
Router DSAs	140
Read-Only DSAs for Provisioning Directory	140
Provisioning Directory Operations DSAs	140

Index 143

Chapter 1: High Availability Overview

This section contains the following topics:

[Purpose of High Availability](#) (see page 9)

[Identity Manager Server Architecture](#) (see page 10)

[Provisioning Components Architecture](#) (see page 12)

Purpose of High Availability

High availability is the possibility to handle more users and systems over many different network operating systems, security systems, databases, and groupware products in a timely fashion.

Identity Manager offers high availability solutions for mission-critical systems that need constant data accessibility.

These solutions make use of the following:

- Failover—Automatic backup operation that switches to another system if the primary system fails or is temporarily offline for any reason.
- Load balancing—Distributing processing and communications activity evenly across a computer network so that no single device is overwhelmed.
- Various deployment tiers that provide the flexibility to serve dynamic business requirements.

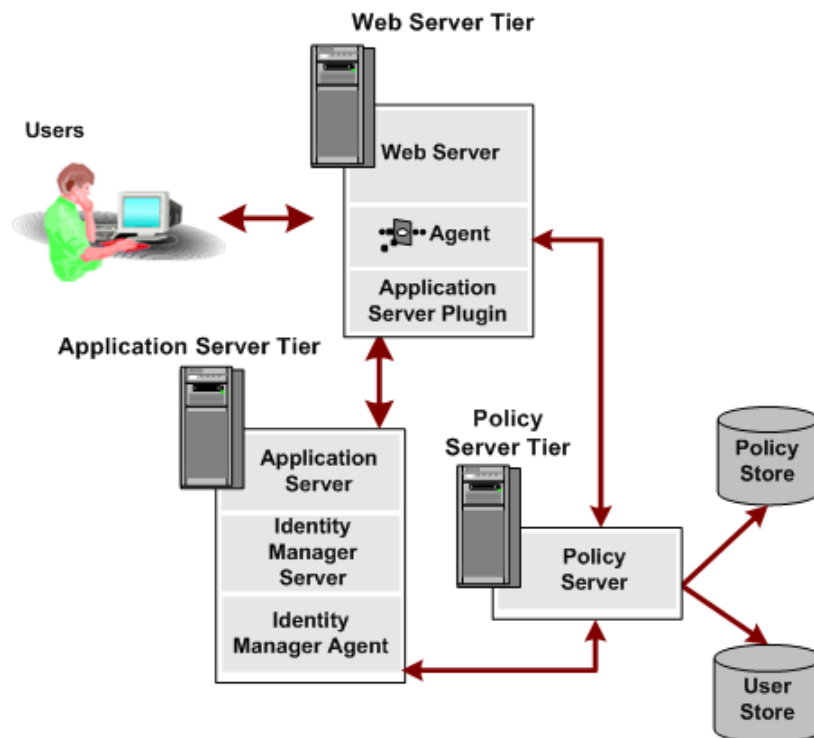
The high availability solutions take full advantage of the routing, failover and load sharing capability of eTrust Directory routers.

Identity Manager Server Architecture

An Identity Manager deployment may span a multi-tiered environment that includes a combination of hardware and software. Each tier may use a cluster. The following figure shows the tiers used by Identity Manager:

- Web Server tier
- Application Server tier
- Policy Server tier

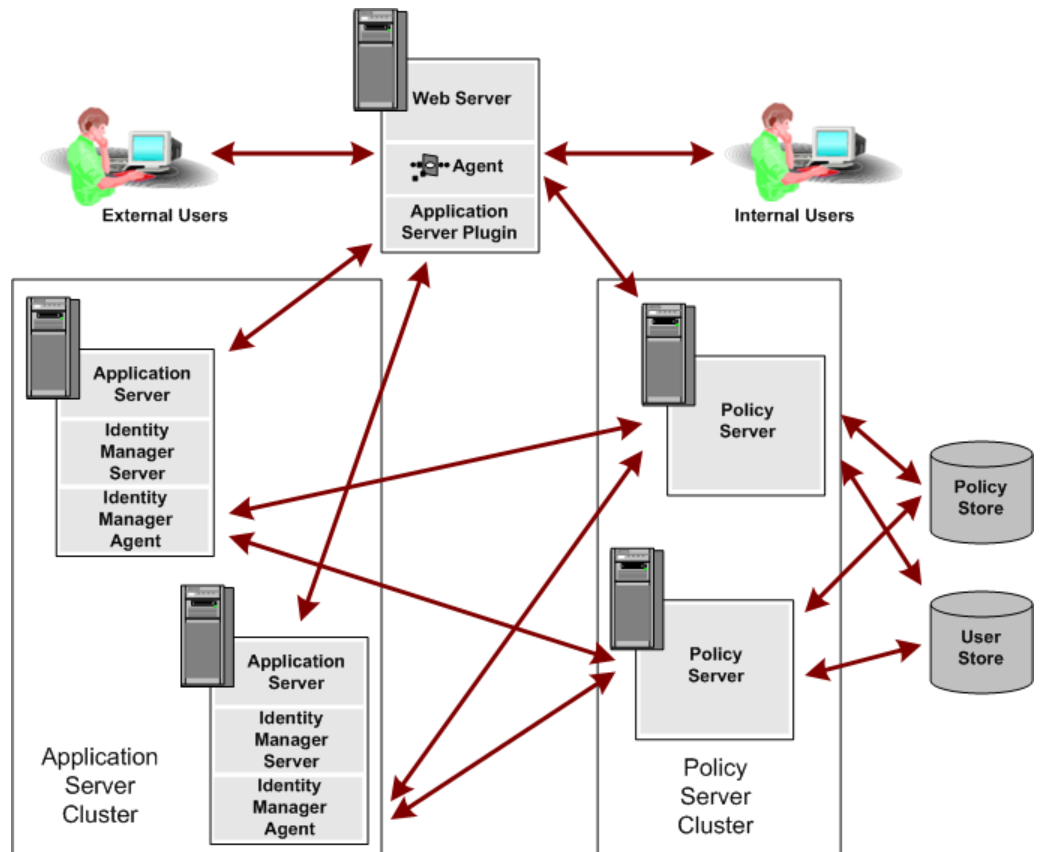
Note: A Policy Server tier is optional. If you have not installed CA SiteMinder, you can base your architecture as in the diagrams, but without the Policy Server tier.



Each tier may contain a cluster of servers that perform the same function to share the workload for that tier. You configure each cluster separately, so that you can add servers only where they are needed.

For example, in a clustered Identity Manager deployment, a group of several systems may all have an Identity Manager Server installed. These systems share the work that is performed by the Identity Manager Server.

The following figure shows an Identity Manager deployment in which the application server and Policy Server tiers are clustered.



In the previous figure, the application server tier and the Policy Server tier each contain a cluster of two systems that share the load and provide failover support. These systems are called *nodes*.

Note: Nodes from different clusters may exist on the same system. For example, an application server node can be installed on the same system as a Policy Server node.

Provisioning Components Architecture

Provisioning provides high availability solutions in three tiers:

- Client tier

You can group clients together based on their geographic locations, organizational units, business functions, security requirements, provisioning workload, or other administration needs. Configure each client with a collection of Provisioning Servers, in order of their failover preference. Client requests continue to be submitted to the first server until that server fails, that is, the connection stays active until the server fails.

In case of a failure, the client checks the list of configured servers, in order of preference, to find the next available server. You can also configure the eTrust Directory Router DXlink so that Provisioning clients can communicate with one router and failover to another. Each Router DXlink can further provide server tier high availability configurations. You can introduce additional routers for various routing purpose, if needed.

- Provisioning Server tier

If you have CA SiteMinder Web Access Manager, you can create multiple server definitions in JIAM to communicate with the Provisioning Server for provisioning operations. The JIAM API supplied with Identity Manager supports native failover in its communication layer. You create these definitions in the smuserdirectory object associated with the Identity Manager directory for your environment.

Without CA SiteMinder Web Access Manager, you can route client requests through the eTrust Directory Router DXlink to servers. The Router DXlink knows which servers are configured for sharing the workload and which are configured for failover.

- In load-sharing mode, the router forwards incoming client requests to the server that is the least loaded.
- In failover mode, client requests are sent to a failover server only when the servers cannot be contacted.

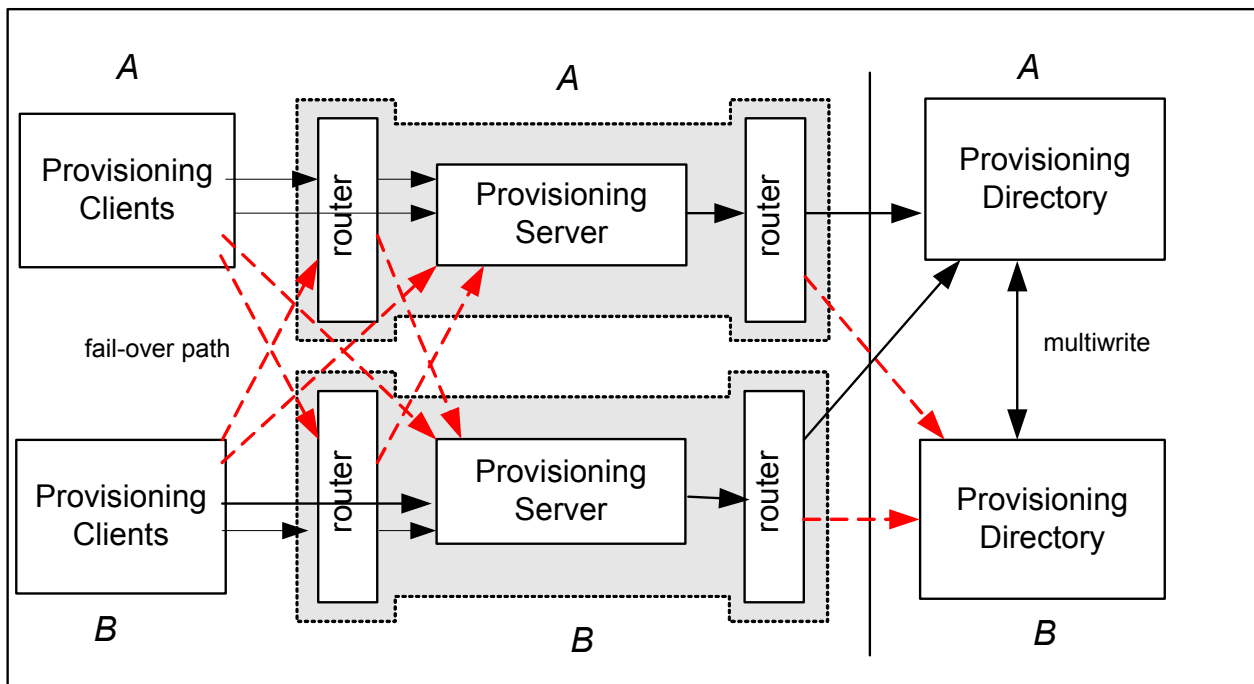
- eTrust Directory Repository tier (Provisioning Directory)

You can use another eTrust Directory router to send server requests to provisioning directories. You can replicate multiple provisioning directories for load-balancing, failover, or both. You can also configure the multiwrite mode to ensure data coherence in replicated provisioning directories. We recommend a multiwrite configuration in which you designate one provisioning directory as the primary provisioning directory. All Provisioning Servers send updates to the primary, and failover to the secondary.

To further enforce data coherence during multi-write recovery, DSAs can be configured to hold off services during the recovery process by using the DSA no-service-while-recovering flag.

An example of high availability architecture follows. Assume that there are two Provisioning Servers, A and B, that serve the same provisioning domain. The provisioning directories are identical. The data coherence of the two provisioning directories is maintained using the eTrust Directory multiwrite technology. The following diagram illustrates the Provisioning failover architecture for this sample domain.

In this diagram, each tier in the architecture provides its own failover mechanism: client failover, Provisioning Server failover, and Provisioning Directory failover. You can configure failover for any tier or any combination of tiers.

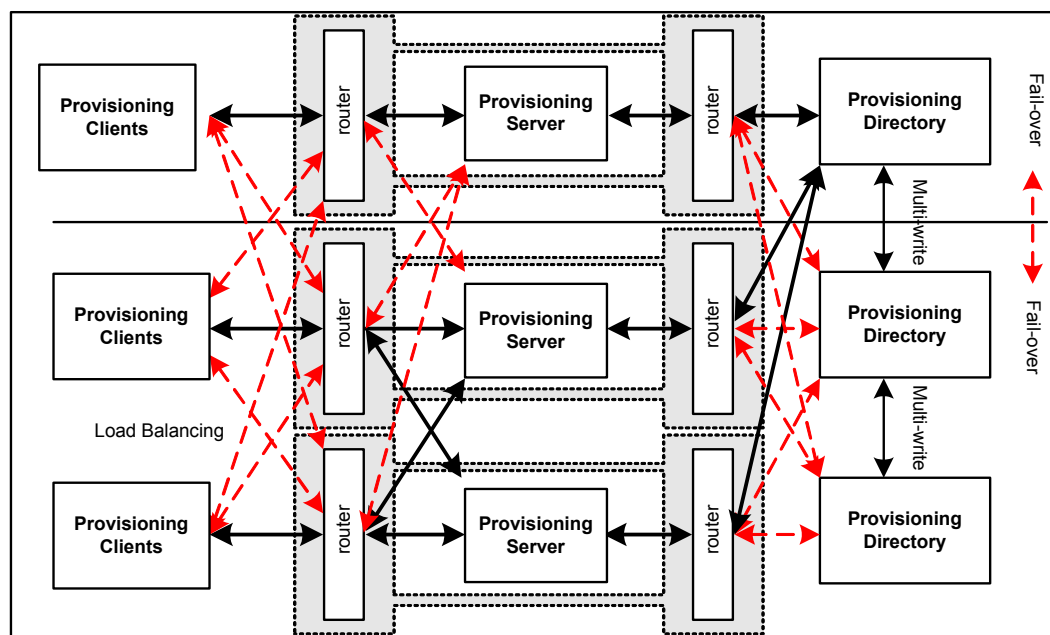


Solid lines indicate the normal control flow. The dashed lines indicate the failover control paths. In normal operational mode, clients, such as Provisioning Manager instances, communicate with their respective Provisioning Servers. The Provisioning Servers in turn communicate with their respective provisioning directories using X.500 router Directory Service Agents (DSAs). The Provisioning Manager, Provisioning Server, provisioning directory, and the routers can reside on the same machine or on different machines across a network of computing systems.

Important! We recommend that you use the multiwrite replication scheme with careful operations monitoring. Multi-write-disp-recovery, an operation mode supported by eTrust Directory, is not recommended because of its lack of value-level reconciliation and its inability to recover large databases after long outages.

For clients, such as Provisioning Manager, that know how to failover from one server to another, we suggest that clients be configured to communicate with Provisioning Servers directly. For those clients that are unable to fail over from one server to another, we recommend that clients send requests to a Router DXLink. This is set during server-tier failover configuration.

The following diagram illustrates a Provisioning failover and load balancing architecture in a single domain with more servers than appeared in the previous diagram:



In this configuration, three Provisioning Servers serve the same Provisioning domain. The bottom two servers are configured for load balancing, that is, the two routers for the bottom servers balance the loads from their respective clients between the two servers. The heavy black lines depict the normal control flow. In contrast, the top-most Provisioning Server is configured as a failover server to the two other servers. The bottom two servers also act as failover for the top server. The dashed red lines show the failover traffic flows. All of the Provisioning Servers have solid lines to one primary provisioning directory and failover to others.

Note: For performance reasons, we recommend that Provisioning clients communicate with Provisioning Servers directly and failover to other Provisioning Servers, if needed.

Chapter 2: Installation, Upgrade, and Migration

This section contains the following topics:

[How to Install Identity Manager for High Availability](#) (see page 17)

[How to Upgrade a High Availability Installation](#) (see page 18)

[How to Migrate a Standalone Installation to a Cluster](#) (see page 22)

How to Install Identity Manager for High Availability

Use the following check list to configure a clustered environment and migrate Identity Manager to it. You may find it useful to print this check list and check off tasks as you complete them.

X	Step	Refer to...
	1. (Optional) Create a Policy Server cluster with the SiteMinder Policy Server and the Identity Manager Extensions.	Policy Server Cluster (see page 25)
	3. Create an application server cluster and install the Identity Manager Server on it.	Identity Manager Cluster (see page 29)
	4. Configure a web server cluster.	Web Server Clusters (see page 69)
	5. Start all the servers in the cluster.	Cluster Startup (see page 72)
	6. Configure Provisioning Components for high availability.	Provisioning Components (see page 77)

How to Upgrade a High Availability Installation

If you have previously installed Identity Manager in a clustered environment for the Identity Manager Server or with provisioning components that provide failover and load balancing, you can upgrade that installation.

Upgrade an Identity Manager Cluster

You can upgrade a previous Identity Manager 8.1 release where it was installed on an application server cluster.

To upgrade Identity Manager on a WebLogic cluster

1. On each node, delete the EAR content Identity Manager from the staging area:
`bea\weblogic92\common\nodemanager\servers\Server_name\stage\IdentityMinder\identityMinder.ear*`
2. Use the Identity Manager installation program from the cluster domain using the same steps you would use for a new installation.

To upgrade Identity Manager on a JBoss cluster

1. Run the Identity Manager installer from the server where the 8.1 Identity Manager installation program was run.
2. Use the Identity Manager installation program from the cluster domain using the same steps you would use for a new installation.

To upgrade Identity Manager on a WebSphere cluster

1. Export the previous version of Identity Manager from WebSphere 6.0.1.17 to WebSphere 6.1.0.15.

Note: Do not create any resources by running `ims6SetupClusterMember.jacl` at this point.

2. Make sure all JMS resources, including IMSBus, are removed under WebSphere Admin Console, Resources, JMS section.

Perform this step for each server in the cluster.

3. Use the Identity Manager installation program from deployment manager using the same steps you would use for a new installation.

Upgrade Highly Available Provisioning Components

The upgrade process introduces a notification DSA with the following DSA definition and port assignments. All the deployed high availability DSAs should not have any port and DSA conflict with the notification DSA.

```
set dsa "etaops_notify" =
{
  prefix = <dc etaops><dc notify>
  dsa-name = <dc etaops><cn notify>
  dsa-password = "secret"
  address = tcp "<hostname>" port 20404
  disp-psap = DISP
  cmip-psap = CMIP
  snmp-port = 20405
  # To enable the console port, use the 'dxpassword' utility to generate the password-hash
  # and enter it below, uncommenting both console-port and console-password lines.
  # console-port = 20406
  # console-password = "{encoding-method}password-hash"
  ssl-port = 20392
  auth-levels = anonymous, clear-password
  dsp-idle-time = 3600
  trust-flags = allow-check-password, trust-conveyed-originator
};
```

To meet prerequisites for the upgrade

Ensure that your high availability installation meets these requirements before the upgrade:

- You can upgrade Identity Manager to r12 from eTrust Admin 8.1 SP2. If you have a previous release of eTrust Admin, upgrade it to eTrust Admin 8.1 SP2 first.

- The notification DSA knowledge file will be added to the %DXHOME%\config\knowledge\etrustadmin.dwg file.

For all high availability DSAs to have the knowledge of the notification DSAs, the DSA DXI files need to be configured to source the above etrustadmin.dwg group knowledge file. If a high availability installation has removed the etrustadmin.dwg file, it can be reintroduced to source other group knowledge files.

- The schema group knowledge file is %DXHOME%\config\schema\etrustadmin.dwg. A DSA that sources the etrustadmin.dwg schema file is considered a Provisioning Directory DSA.
- The notification DSA brings in a default etaops_notify.pem file which may not match high availability DSA personality files generated using the dxcertgen utility. Therefore, the etrustadmin_trusted.pem file should retain the default IM PD root CA certificate. The root CA certificate generated using dxcertgen in a high availability deployment should be appended to etrustadmin_trusted.pem.
- If the notification DSA does not have the knowledge of all the deployed high availability DSAs, adjustments need to be made after the upgrade to make sure that all DSAs know about each other.
- The multi-write queues of the DSAs to be upgraded must be drained before upgrade. Otherwise, the updates in the queue may be lost during the upgrade process. In addition, the DSAs in the upgrade process should not be the target of any write requests.

You may want to perform the high availability upgrade in a lab environment first. If the upgrade is successful, back up your production deployment and run upgrade installer in the production environment. If the high availability upgrade encounters issue in a lab environment, follow the manual process to upgrade your high availability deployment.

To avoid down time during the upgrade

We recommend that an Identity Manager r12 Provisioning Server installation uses additional Provisioning Servers and Provisioning Directories. All Provisioning Servers communicate with their respective local Provisioning Directory routers, which communicate with the Provisioning Directory Servers.

In an Identity Manager r12 Provisioning Server upgrade scenario, use these steps to avoid down time:

1. Designate one Provisioning Server and one Provisioning Directory Server for the upgrade of eTrust Admin 8.1 SP2 to Identity Manager r12 Provisioning Server.
2. Disconnect the Provisioning Server and Provisioning Directory from the rest of the Provisioning Servers and Provisioning Directory Servers.

The disconnected Provisioning Server and Provisioning Directory will not be servicing requests from clients from this point on.

3. Back up the Provisioning Directory.
4. Upgrade the disconnected Provisioning Server and Provisioning Directory using the Identity Manager r12 Provisioning Server Upgrade program.
5. Redirect all the provisioning clients to submit provisioning requests to the Identity Manager r12 Provisioning Server.

6. Back up the content of the Provisioning Directory used for eTrust Admin 8.1 SP2.

The changes between the backups done in steps 3 and 6 reflect the Provisioning Directory changes during the upgrade process.

7. Apply the Provisioning Directory changes above to the Identity Manager r12 Provisioning Directory.
8. Use the Upgrade program on the remaining Provisioning Servers and Provisioning Directories.

All Provisioning Server and Provisioning Directories are upgraded to Identity Manager r12.

9. Replicate the Identity Manager r12 Provisioning Directory content on all newly upgraded Provisioning Directories.
10. Restore the high availability configuration in which all Provisioning Servers communicate with their respective local Provisioning Directory routers.

How to Migrate a Standalone Installation to a Cluster

We recommend that you install, customize, and debug Identity Manager in a development environment before you install it in a production environment. A common scenario is to install Identity Manager on one system in a development environment and migrate that installation to a cluster of systems in a production environment.

Note: When you use the Identity Manager Management Console during the migration, only one Identity Manager node can be enabled. Stop all but one Identity Manager node before you create or modify an Identity Manager environment or an Identity Manager directory.

Once the development environment has been established and tested, you can migrate Identity Manager to a production environment. Migration includes:

- Making deployment-specific changes according to the requirements of the HTTP Servers, J2EE Application Servers, firewalls, hardware load-balancers, and other applications in the environment
- Creating a clustered environment for Identity Manager
- Migrating the Identity Manager environments — Includes objects that are stored in the Identity Manager object policy store, such as role and task definitions, and Identity Manager directories.
- Modifying custom code — Includes features that you created with Identity Manager APIs, such as logical attribute handlers, business logic task handlers, and event listeners. Custom code may also include email templates and skins for branding Identity Manager environments.
- Migrating Workflow scripts — Include custom workflow activity definitions

Note: After migrating Identity Manager to a production environment, you may need to update that environment periodically when the development environment changes. For details, see the *Configuration Guide*.

Chapter 3: Policy Server Cluster

This section contains the following topics:

[Configure a Policy Server Cluster](#) (see page 25)

[Migrate the Policy Store](#) (see page 25)

Configure a Policy Server Cluster

To configure a policy server cluster

1. Install the SiteMinder Policy Server on each node that should be in the Policy Server cluster. For details, see the *CA SiteMinder Web Access Manager Installation Guide*.

Note: Each Policy Server in the cluster uses the same policy store.

2. Install the Identity Manager Extensions to the Policy Server.

After installing the Policy Server and the Identity Manager Extensions on each node, continue with the next section, [Migrate the Policy Store](#) (see page 25).

Migrate the Policy Store

If you creating a highly available installation by migrating standalone systems to a cluster, you migrate the policy store. Migrating the policy store involves exporting all objects from the policy store in the development environment and importing them into the policy store in the production environment.

To migrate the policy store

1. Export data from the policy store in the development environment, as follows:

- a. Navigate to the *siteminder_installation/bin* directory.

- b. Enter the following command:

```
smobjexport -ofile-name.smdif-c -dadmin-name  
-wadmin-pw-v -t export-log
```

file-name.smdif is the name of the output file

-c exports sensitive data as clear text to allow you to migrate a policy store that uses one encryption key to a policy store that uses a different encryption key

-d*admin-name* identifies the SiteMinder Administrator who can manage all SiteMinder objects in the policy store being exported

-w*admin-pw* is the password for the SiteMinder Administrator

-v enables verbose logging

-t enables low level tracing, which can help to troubleshoot problems

export-log is a file that records output from the command

Note: Running smobjexport also creates a .cfg file in the same location where the .smdif file is created.

2. Check the *export-log* file for any errors that you need to address.

3. Delete the policy store data by entering this command:

```
smdapsetup remove aadmin_name badmin_pw
```

-a*admin-name* identifies the SiteMinder Administrator who can manage all SiteMinder objects in the policy store being exported

-b*admin-pw* is the password for the SiteMinder Administrator

4. Run the following command to set the super user password:

```
smreg -su password
```

5. Modify development environment settings as follows:

- a. Edit *file-name.cfg* to change the development environment settings to production environment settings.

To edit this tab-delimited file, use Microsoft Excel, Microsoft Word, or another word processor.

- b. Replace all development environment references including IP addresses, agent names, shared secrets, and URL references.

6. Import data into the production policy store as follows:

a. Copy the .sml dif and .cfg files to the production server.

b. Execute the following command:

```
smobjimport -i file-name.sml dif -d admin-name  
-w admin-pw -v -t -f -c
```

-i *file-name.sml dif* is the name of the file generated from running smobjexport

-d *admin-name* is the name of the SiteMinder administrator

-w *admin-pw* is the password of the SiteMinder administrator

-v enables logging

-t enables tracing for troubleshooting

-f indicates that existing objects with the same OID are overwritten.

-c indicates the input file contains sensitive data in clear-text, so that you can import policy data from a SiteMinder deployment that uses one encryption key to another SiteMinder deployment that uses a different encryption key.

7. Follow the instructions in the next section to configure an application server cluster for Identity Manager.

Chapter 4: Identity Manager Cluster

This section contains the following topics:

[Application Server Information](#) (see page 29)

[WebLogic Clusters](#) (see page 30)

[JBoss Clusters](#) (see page 47)

[WebSphere Clusters](#) (see page 52)

[SiteMinder High Availability for the Application Server Cluster](#) (see page 63)

[Policy Server and Application Server Communication](#) (see page 66)

[Migrate Workflow Process Definitions](#) (see page 67)

[Web Server Clusters](#) (see page 69)

[Cluster Startup](#) (see page 72)

[Verify the Clustered Installation](#) (see page 75)

[Update the Migrated Environments](#) (see page 75)

Application Server Information

For complete information about WebLogic clusters, see the application server documentation at one of these locations, based on the software release you are using:

<http://e-docs.bea.com/wls/docs81/cluster/>

<http://e-docs.bea.com/wls/docs92/cluster/>

For complete information about JBoss clusters, see the application server documentation at:

<http://www.jboss.org/>

For complete information about WebSphere clusters, see the application server documentation at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>

Note the following about Identity Manager clusters:

- In a clustered environment, an Identity Manager task may fail (depending on the execution state) when a server in the cluster fails. In this case, users see a task failed error in the user console and will need to resubmit the task. When the task submits again, the request is forwarded to the next server in the cluster.
- Identity Manager does not support HTTP session persistence in a clustered environment.
- Before installing Identity Manager to a production environment, be sure that you have an active support contract for the application server that will host Identity Manager.

WebLogic Clusters

A WebLogic Server cluster consists of multiple WebLogic Server instances that work together to provide increased scalability and reliability.

WebLogic clusters are configured differently for WebLogic 8.1 and WebLogic 9.2.

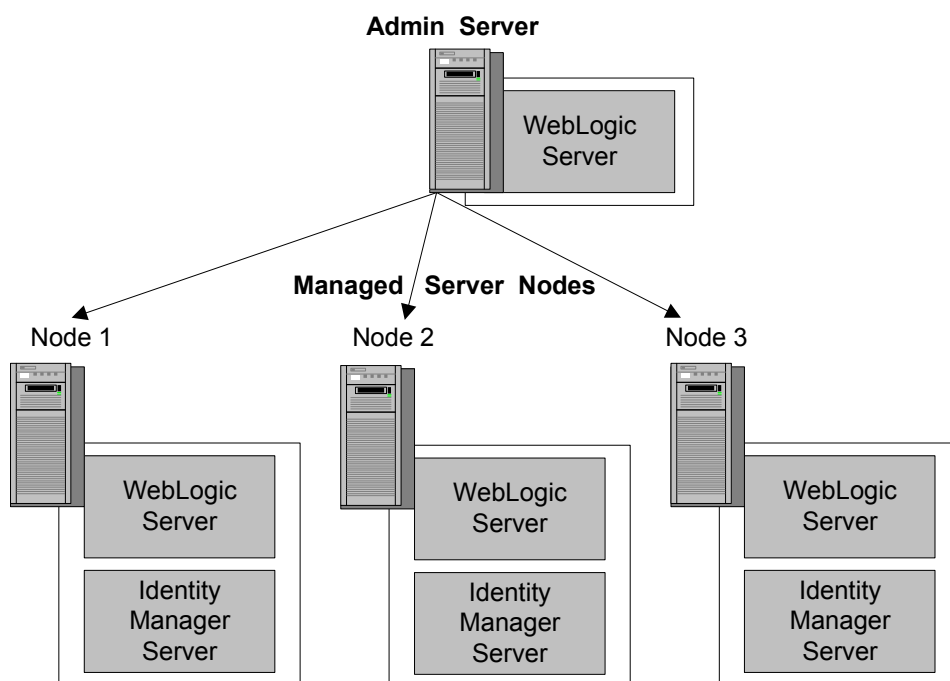
WebLogic 8.1 Clusters

Note: Before you configure a WebLogic cluster, you should be familiar with WebLogic clusters and JMS objects. See BEA's documentation at: <http://edocs.bea.com/wls/docs81/cluster/>

Configuring a WebLogic cluster for Identity Manager involves configuring two types of servers:

- One Admin Server, which manages the nodes in a cluster
- Managed Server nodes, each of which run the Identity Manager Server

The following figure shows the relationship between the Admin Server and three Managed Server nodes in a WebLogic cluster.



Note: This figure shows the Identity Manager Server on each Managed Server node. However, to configure a cluster for Identity Manager, you install the Identity Manager Server on the Admin Server node and deploy it to the Managed Server nodes.

Install Identity Manager and WebLogic

To install the software needed for the cluster

1. On each node that will host a WebLogic server (Admin Server or Managed Server), install the WebLogic Application Server.
2. On the node where you create the Admin Server, create a cluster domain.
This domain can serve as the source location from where the application will be deployed to the managed server nodes.
3. On each managed server node, create a directory that will serve as a staging area for Identity Manager directories.
4. On the node with the Admin Server, use the Identity Manager installation program as described in WebLogic version of the *Installation Guide*.
5. Copy the startManagedWebLogic.cmd or startManagedWebLogic.sh file from `wl_home\user_projects\domains\wl_domain` to the staging folder on each Managed Node.
6. Continue with the next procedure to create the cluster.

Create a WebLogic 8.1 Cluster

Once you have installed WebLogic and Identity Manager, create the WebLogic cluster.

To create a WebLogic 8.1 cluster

1. Follow the steps in the section "Example: Creating a Domain with Administration Server and Clustered Managed Servers" in the BEA documentation:
<http://edocs.bea.com/wls/docs81/adminguide/createdomain.html#CreateClusteredDomain>
Note: For ease of reference, consider printing the BEA document.
2. Follow the steps to "Start the WebLogic Server Cluster" in this BEA document:
<http://edocs.bea.com/wls/docs81/cluster/setup.html#718051>
3. Follow the instructions in [Create JMS Objects](#) (see page 33).

Create JMS Objects

After configuring a WebLogic cluster for Identity Manager, you create the JMS objects needed by the cluster. Create JMS objects using one of these approaches:

- **Single Server Approach** — One node contains the JMS objects and other servers in the cluster communicate with it for JMS functions.
- **Distributed Server Approach** — One JMS server exists for each node in the cluster. This approach provides better performance, because JMS functions occur on the local node.

Note: Some of the JMS objects may already exist. You can modify the existing objects or delete them and create new objects.

Configure JMS Objects on a Single Server

To configure the JMS objects on a single server

1. Decide which Managed Server node you will use to run JMS services.
2. On that node, create an IM_JMS directory, for example:
c:\bea\user_projects\IM_JMS
3. Start the WebLogic console by pointing your browser to:
WebLogic Admin Server address:port/console
4. In the Services area, configure the following JMS objects:
 - **JMS File Store:**
Name: *jms_store_name* (any meaningful name)
Synchronous Write Policy: Cache-Flush
Directory: *pathname/IM_JMS*
 - **JMS Server:**
Server Name: *jms_server_name* (any meaningful name)
Persistent Store: *jms_store_name* (the JMS File Store that you configured)
5. In the left pane under Services, select *jms_server_name*, the JMS Server that you created.

6. For that server, configure Destinations with the following values:
 - JMS Queue
 - Name:** IMS Events
 - JNDIName:** com.netegrity.ims.msg.queue
 - JMS Topic
 - Name:** General Monitor Messages
 - JNDIName:** com/netegrity/idm/GeneralMonitorActions
 - Connection Factory
 - Name:** General Monitor CF
 - JNDIName:** com/netegrity/idm/GeneralMonitorCF

Note: After creating the Connection Factory, set the Target to a server in the cluster.
7. Target the server where the JMS service will run:
 - a. Under Services, select *jms_server_name*.
 - b. Set Target to the Managed Server node where you created the IM_JMS directory.
8. In the Services area, configure the following JMS object for WorkPoint workflow:
 - JMS Server for WorkPoint workflow:
 - Server Name:** *jms_wf_server_name* (any meaningful name, such as wpJMSServer)
9. In the left pane under Services, select *jms_wf_server_name*, the JMS Server that you created for workflow.

10. For the JMS Server for WorkPoint workflow, configure Destinations with the following values:
 - JMS Queue
Name: wpServAutoActQueue
JNDIName: queue/wpServAutoActQueue
 - JMS Queue
Name: wpUtilQueue
JNDIName: queue/wpUtilQueue
 - Connection Factory
Name: wpConnectionFactory
JNDIName: jms/wpConnectionFactory
XAConnectionFactoryEnabled: true
11. Target the server where the JMS service for workflow will run:
 - a. Under Services, select *jms_wf_server_name*.
 - b. Set Target to the Managed Server node where you created the IM_JMS directory.
12. To continue the configuration, go to [Deploy the Identity Manager Server to Clustered Nodes](#) (see page 36).

Configure a Distributed JMS Server

To configure a distributed JMS server, perform these steps once for each node in the cluster.

To configure a distributed JMS server

1. Create an IM_JMS directory on this node, for example:
c:\bea\user_projects\IM_JMS
2. Start the WebLogic console by pointing your browser to:
WebLogic Admin Server address:port/console

3. In the Services area, configure the following JMS objects:
 - JMS File Store
 - Name:** *jms_store_name* (any meaningful name)
 - Synchronous Write Policy:** Cache-Flush
 - Directory:** *pathname/IM_JMS*
 - JMS Server
 - Server Name:** *jms_server_name* (any meaningful name)
 - Persistent Store:** *jms_store_name* (the JMS File Store that you configured)
4. Target the server where the JMS service will run:
 - a. Under Services, select *jms_server_name*.
 - b. Set Target to the appropriate cluster node server.
5. Under Services, configure Distributed Destinations with the following values:
 - Distributed Queue
 - Name:** IMS Events
 - JNDIName:** com.netegrity.ims.msg.queue
 - Distributed Topic
 - Name:** General Monitor Messages
 - JNDIName:** com/netegrity/idm/GeneralMonitorActions
6. For each Distributed Queue and Topic, use the Auto Deploy feature to add cluster nodes as members of the destination.
7. To continue the configuration, [deploy the Identity Manager Server to clustered nodes](#) (see page 36).

Deploy the Identity Manager Server to Clustered Nodes

Use the following steps to move an Identity Manager (which you installed and configured in a development environment) to a WebLogic cluster.

Note: In a clustered environment, an Identity Manager tasks may fail (depending on the execution state) when a server in the cluster fails. In this case, users see a task failed error in the user console and will need to resubmit the task. When the task submits again, the request is forwarded to the next server in the cluster.

To deploy the Identity Manager Server to clustered nodes

1. Copy the IdentityMinder.ear from the development environment to a location available to the WebLogic Administration Server for the cluster in the production environment.

The IdentityMinder.ear is located in:

development_wl_install\bea\user_projects\wl_domain\applications\IdentityMinder.ear

2. Use the WebLogic Admin console to deploy the IdentityMinder.ear to the cluster:
 - a. Start the Admin Server and Managed Server Nodes.
 - b. Deploy the Identity Manager application to the cluster. See the following BEA documentation:

<http://edocs.bea.com/wls/docs81/cluster/index.html>

Note: Deploying the application may take several minutes. To check your progress, click Activate application Identity Manager on *yourclustername* and select the Details tab.

If you see errors indicating that WebLogic cannot find resources on the managed server nodes, make sure that the paths in the StartManagedWebLogic.cmd/.sh file on each managed node in the cluster include references to the deployed IdentityMinder.EAR on that server.

For example, make sure the following paths are correct:

```
set PATH=C:\bea\user_projects\domains\domain\applications\IdentityMinder.ear\library;%PATH%
```

```
set IDM_OPTS=-Djava.rmi.server.codebase=
```

```
file:///C:/bea/user_projects/domains/domain/applications/IdentityMinder.ear/library/wpMonitorAdaptorUtil.jar
```

- c. Restart the WebLogic cluster.

WebLogic 9.2 Clusters

A WebLogic Server cluster consists of multiple WebLogic Server instances that work together to provide increased scalability and reliability.

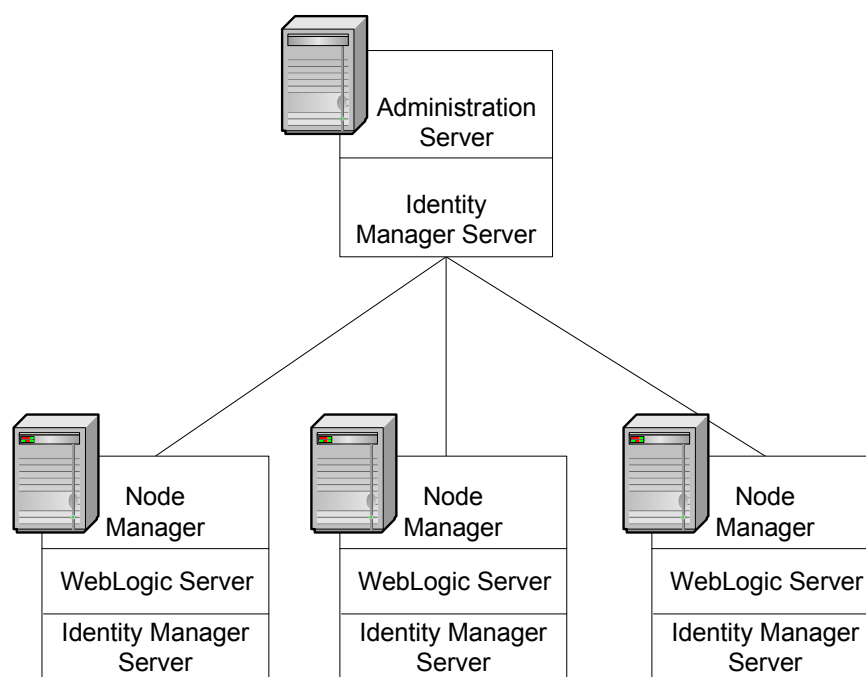
Note: Before you configure a WebLogic cluster, you should be familiar with WebLogic clusters and JMS objects. See BEA's documentation at: <http://edocs.bea.com/wls/docs92/cluster/>

WebLogic 9.2 clusters include the following components:

- One Administration Server, which configures and manages the WebLogic Server instances in its domain
- Managed Server nodes, each of which run the Identity Manager Server
- Node Manager, a WebLogic Server utility that enables you to start, shut down, and restart Administration Server and Managed Server nodes from a remote location.

You install the Node Manager on each WebLogic server in the cluster.

The following figure shows the relationship between the Administration Server and three Managed Server nodes in a WebLogic cluster.



Create a WebLogic 9.2 Cluster

The instructions in this section describe how to configure a cluster based on the multi-tier architecture described in [BEA's clustering documentation](#).

To create a WebLogic 9.2 cluster

1. Decide which system in the cluster will be the Administration Server, and which systems will be the managed nodes.
2. Install WebLogic on the Administration Server and the managed nodes.

3. Use the WebLogic Configuration Wizard to create the WebLogic domain on the Administration Server.

For instructions, see BEA's documentation on [creating a domain](#).

4. Customize the WebLogic domain for the cluster.

For instructions, see BEA's documentation on [customizing a WebLogic domain for a cluster](#).

5. On the node where you create the Administration Server, create a cluster domain.

This domain can serve as the source location from where the application will be deployed to the managed server nodes.

6. On each managed server node, create a directory that will serve as a staging area for Identity Manager directories.

Register Node Manager

Node Manager is a WebLogic utility that allows you to start the Administration Server and managed nodes from a remote location. BEA recommends using the Node Manager in a high availability cluster configuration.

For information on registering Node Manager on each managed node with the WebLogic Administration Server, see BEA's Node Manager documentation http://e-docs.bea.com/wls/docs92/server_start/nodemgr.html.

Verify the Cluster

Verify that you can start and stop each managed node.

To verify the cluster

From the WebLogic Admin Console:

1. Set up user and password for each managed node:
 - a. In an internet browser, go to `http://admin server machine:7001/console`
 - b. Choose Environment, servers, *Server Name*
 - c. Go to Server Start tab.
 - d. Provide the user and password that you used to set up the managed node

2. On each node manager system,
 - a. Start the WebLogic Node Manager.

On Windows, go to Start, Programs, BEA Products, Tools and choose Run Node Manager.

On Solaris use the script: NodeManager.sh
 - b. Start each of the managed servers/machines. In the WebLogic Admin Console, go to:

Environment->servers, *Server Name*, control, start/stop
3. Stop the Admin server and all managed servers in preparation for installing CA Identity Manager.

Install Identity Manager

Before deploying Identity Manager to the cluster, deploy it in a cluster domain in the Administration Server to verify that it installs correctly. The cluster domain serves as a staging area where you can test Identity Manager and then deploy it to other nodes in the cluster.

When you supply a cluster name during the installation, these primary resources will be configured:

- Distributed queues/topics with Round-Robin as the default load-balancing algorithm, all targeted to cluster name provided.
- Connection factories targeted to cluster name provided.
- Data sources also targeted to cluster name provided.
- Two JMS modules: imjmsmodule and wpjmsmodule.

During the installation, the following EARs are installed to that cluster domain at the following root C:\bea\user_projects\domains\<domain name>\applications

- IdentityMinder.ear
- ca-stylesr5.1.1.ear

To install Identity Manager on the Administration Server

1. On the system that hosts the WebLogic Admin Server, run the Identity Manager installation.
2. For details on the installation, see the WebLogic version of the *Installation Guide*.

- When you reach the WebLogic section, complete it as follows:

WebLogic Binary Folder

The location of the application server home directory.

Domain Folder

The name of the WebLogic domain you created for Identity Manager.

Server Name

The name of the WebLogic server on which the domain is configured.

Cluster Name

The name of the cluster.

App Server URL and port

The application URL and port number of the system that will host the application server used by the Identity Manager Server.

WebLogic Binary Folder:	C:\bea\weblogic92\	Restore Default	Choose...
Domain Folder:	C:\bea\user_projects\domains\domain7001\	Restore Default	Choose...
Server Name:			
Cluster Name:	new_Cluster_1		
App Server URL and port:	http://wl9-manager.ca.com:7001		

Configure Managed Nodes

To configure the managed nodes in a cluster, you copy files from the cluster domain on the Administration Server to each managed node, and then configure the JDBC drivers for the cluster.

To configure managed nodes

- Create a lib directory in the nodemanager directory on each managed node in the cluster.

By default, the nodemanager directory is located in:

- For Windows: C:\Program Files\weblogic92\common\nodemanager
- For Solaris: /bea/weblogic92/common/nodemanager

- Copy the JDBC drivers such as sqljdbc.jar or ojdbc14.jar to C:\bea\weblogic92\server\lib on the Administration Server to the same location on each managed node for the task persistence, workflow, auditing, and reporting databases.

3. Copy agentfilter.jar under C:\bea\user_projects\domains\<domain name>\lib to each of the managed nodes at location C:\bea\weblogic92\server\lib
4. Copy the following DLLs from the library directory in the clustered domain on the Administration Server to the nodemanager\lib directory on each managed node:
 - smagentapi.dll
 - smcommonutil.dll
 - smerrlog.dll
 - smjavaagentapi.dll
5. Specify parameters for running Identity Manager on each managed node in the commEnv.cmd/.sh file as follows:
 - a. Create a backup copy of the commEnv file, which is located in *wl_home\common\bin*.
 - b. Copy the sample commEnv file from *im_admin_tools_dir\samples\Cluster\Weblogic* to *wl_home\common\bin*
 - c. Verify that the PATH and IDM_OPTS parameters point to the [staging path](#) (see page 38) for the managed server node.
 - d. Locate the following parameters in the commEnv file and modify them for your environment:
 - set IM_BEA**

The installed location of WebLogic 9.2. For example:

```
set IM_BEA=C:\bea921
```
 - set IM_JAVA**

The installed location of the JDK required by Identity Manager on WebLogic 9.2. This is the location you specified during installation.

For example: set IM_JAVA=C:\Program Files\Java\jdk1.5.0_11
 - set IM_MANAGED_SERVER**

The name of WebLogic server where the commEnv file is located.

For example: set IM_MANAGED_SERVER=imServer1
 - e. Save the file.

Note: If you have made changes to the default commEnv file and want to continue using that file instead of the sample commEnv file, there are additional parameters that you need to set.

6. On a Windows-based cluster, edit commEnv.bat as follows:
 - a. On each managed node, append the JARs in bold to the weblogic classpath:

```
set
WEBLOGIC_CLASSPATH=%PATCH_CLASSPATH%;%JAVA_HOME%\lib\tools.jar;
%WL_HOME%\server\lib\weblogic_sp.jar;%WL_HOME%\server\lib\weblogic.jar;
%WL_HOME%\server\lib\webservices.jar;
%WL_HOME%\server\lib\ojdbc14.jar;%WL_HOME%\server\lib\sqljdbc.jar;
%WL_HOME%\server\lib\agentfilter.jar
```

- b. Insert this if you are using Workflow:

```
set IDM_OPTS=
Djava.rmi.server.codebase=file:///C:\bea\weblogic92\common\nodemanager\
servers\server_name\stage\IdentityMinder\IdentityMinder.ear\library\
wpMonitorAdaptorUtil.jar
set IDM_OPTS=%IDM_OPTS% -Djava.security.manager=weblogic.rmi.RMISecurityManager
set IDM_OPTS=%IDM_OPTS% -
Dworkpoint.classpath.url=file:///C:\bea\weblogic92\common\nodemanager\
servers\server_name\stage\IdentityMinder\IdentityMinder.ear\workflow_rar

REM The provisioning servlets have an implementation of JCE that conflicts with the libraries
REM that come with the VM, so make sure the VM's JCE jars are not in the classpath.
set IDM_OPTS=%IDM_OPTS% -Djava.ext.dirs=
set JAVA_OPTIONS=%JAVA_OPTIONS% %IDM_OPTS% -
Dweblogic.security.SSL.trustedCAKeyStore=C:\bea\weblogic92\server\lib\cacerts
```

7. On a Solaris-based cluster, edit commenv.sh as follows:

a. On each managed node, add the following content:

```
# JDBC driver class paths.  
DB_CLASSPATH=${WL_HOME}/server/lib/msbase.jar:${WL_HOME}/server/lib/  
mssqlserver.jar:${WL_HOME}/server/lib/msutil.jar:${WL_HOME}/server/  
lib/ojdbc14.jar:${WL_HOME}/server/lib/sqljdbc.jar  
  
# IM Modified CLASSPATH  
CLASSPATH="${CLASSPATH}:${DB_CLASSPATH}"  
export CLASSPATH
```

b. Insert this if you are using Workflow:

```
IDM_OPTS=  
Djava.rmi.server.codebase=file:///export21/bea921/user_projects/domains/  
mile18b/applications/IdentityMinder.ear/library/  
wpMonitorAdaptorUtil.jar  
# IDM_OPTS="$IDM_OPTS -Djava.security.manager=weblogic.rmi.RMISecurityManager"  
IDM_OPTS="$IDM_OPTS -Dworkpoint.classpath.url=file:///export21/bea921/user_projects/domains/  
/mile18b/applications/IdentityMinder.ear/workflow_rar"  
  
JAVA_OPTIONS="$IDM_OPTS"  
  
export JAVA_OPTIONS POST_CLASSPATH MEM_ARGS LD_LIBRARY_PATH
```

8. Set the following parameters in the Server Start configuration for each managed node in the WebLogic console:

User Name

The name of a user account that can access the Administration Server.

Password

The password for the Administration Server user account.

Arguments

The memory allocation for Identity Manager on the WebLogic server.

The memory allocation should be at least 768M. For example:

```
-Xms768m -Xmx768m -XX:MaxPermSize=768m
```

9. Restart WebLogic.

Start the Cluster Domain

After you configure managed nodes, you start the WebLogic cluster.

To start the WebLogic cluster

1. Start the Node Manager on each managed node.
 - For Windows: Navigate to Start, Programs, BEA Products, Tools, Node Manager
 - For Solaris: Use the startNodeManager.sh script.

A sample startNodeManager.sh script is installed in `WL_HOME\server\bin`. Edit this script for your environment.

For more information, see BEA's documentation on [starting Node Manager](#).

2. Use the WebLogic Administration Console to start the managed nodes:
 - a. Navigate to Environment, servers, *managed node*, control, start/stop.
 - b. Repeat for each managed node in the cluster.

Note: If errors stating that Identity Manager cannot connect to SiteMinder occur, set the `-Djava.library.path` in the Arguments field in the WebLogic console for each node in the cluster. The `-Djava.library.path` must be set to the [directory that contains the SiteMinder library files](#) (see page 41). Make sure that this path is not overwritten by the default path.

Create a Distributed JMS Server for WebLogic 9.2

To configure a WebLogic cluster for high availability, configure a distributed JMS server to manage queues and topics in JMS Modules for the cluster.

To create a distributed JMS server

1. Create an `IM_JMS_filestore` directory on the managed node, for example:
`c:\bea\user_projects\IM_JMS_filestore`
2. Under Admin Console, go to Services, Messaging, JMS Servers, New JMS Server.

3. Configure the following JMS objects in the Persistent Store section:
 - JMS File Store

Name: IM_JMS_filestore

Directory: *pathname/IM_JMS_filestore*
 - JMS Server

Server Name: *jms_server_name* (any meaningful name)

Persistent Store: *jms_store_name* (the JMS File Store that you configured)

Target: Set the target for these objects to the server where the JMS service will run. Use the imjmsmodule created during the installation.

Note: The required distributed queues and topics were already created targeted to the cluster during installation.
4. Repeat steps 1 to 3 using wpjmsmodule created during the installation.
5. Repeat this procedure for one managed node per server in the cluster.
6. To continue the configuration, [configure a distributed JMS Server for workflow](#) (see page 46).

Configure a Distributed JMS Server for Workflow

If you are using workflow in your development environment, create additional JMS resources for the WebLogic cluster.

To create a distributed JMS server for workflow

1. Create a WP_JMS_filestore directory on each managed node, for example:
c:\bea\user_projects\WP_JMS_filestore
2. Create a JMS Server for each managed node in the WebLogic Administration Console.

Set the target for each JMS Server to the managed node name, and configure each JMS Server to use the file store you created in step 1.
3. Create a wpJMSSubDeployment.
4. Set wpJMSSubDeployment to each of Workflow JMS servers you created.

wpSubDeployment associates with resources
{queue/wpServAutoActQueue, queue/wpUtilQueue},
wpJMSSubDeployment associates with jms/wpConnectionFactory.
5. Create the Workflow Data Source for the workflow database.

When you create the data source for the cluster, set the target to the cluster, and enable Logging Last Resource.

6. Repeat this procedure for one managed node per server in the cluster.
7. To continue the configuration, [deploy the Identity Manager Server to clustered nodes](#) (see page 36).

JBoss Clusters

The following procedures describe how to set up multiple JBoss application servers with the same Identity Manager application on each server. In this type of cluster, each JBoss application server acts independently of the other application servers. Failover and load balancing are handled by the JK connector.

Use these steps:

1. [Test the Default Multicast Address](#) (see page 47)
2. [Create the First Cluster Node](#) (see page 48)
3. [Add Cluster Nodes](#) (see page 51)
4. [Configure the JK Connector](#) (see page 70)
5. [Start the Servers](#) (see page 73)

Test the Default Multicast Address

The `run_idm` script uses a multicast address, either the default address or an alternative address supplied by your network administrator.

To test the default multicast address

1. Run sender on first node as follows:
 - a. Navigate to `jboss-home-1/server/all/lib`.
 - b. Run: `java -cp jgroups.jar org.jgroups.tests.McastSenderTest -mcast_addr 224.10.10.10 -port 5555`
 - c. Enter any text.
 - d. Confirm that a reply appears to acknowledge the text was sent.

2. Run receivers on other nodes in the cluster.
 - a. Navigate to `jboss-home-N/server/all/lib`.
 - b. Run: `java -cp jgroups.jar org.jgroups.tests.McastReceiverTest -mcast_addr 224.10.10.10 -port 5555`
 - c. Confirm that a message appears, the text that you sent from the first node.
3. If either the send or receive test fails, ask your network administrator to provide a multicast address that works and repeat this test.

Create the First Cluster Node

You begin creating the JBoss cluster by creating the first node. IPv6 is not supported for a JBoss cluster with the current release of the JDK. Each node must be an IPv4 system or part of an IPv4/IPv6 stack.

To create the first cluster node

1. Install the Identity Manager server on one of the systems intended to be part of the cluster.

Note: To avoid contention over `workpoint.log` in `jboss_home/bin` directory, each node on one system needs a separate `jboss_home`. For Identity Manager installation instructions, see the *Installation Guide*.

2. On Windows systems, edit `run_idm.bat` in the `jboss_home\bin` directory:
 - a. Uncomment the cluster-related parts of `run_idm.bat`.
 - b. Replace *host-name* at the end of this section with name of the system:

```
set ARGS=
set SERVER=default/deploy
:loop
if [%1] == [] goto endloop
set ARGS=%ARGS% %1
if [%1] == [-c] set SERVER=%2/fam
shift
goto loop
:endloop
ARGS=%ARGS% -b host-name -g IdmPartition -u multicast-address
```

3. For Solaris systems, edit `run_idm.sh` in the `jboss_home\bin` directory:

- a. Uncomment cluster-related parts of this script.
- b. Replace *host-name* at the end of this section with name of the system:

```
SERVER=default/deploy
ARGS=
    until [ -z "$1" ]
    do
        ARGS="${ARGS} $1"
        if [ $1 = '-c' ]
        then
            SERVER=$2/farm
        fi
        shift
    done
ARGS="${ARGS} -b host-name -g IdmPartition -u multicast-address"
```

4. Copy the following files from `jboss_home\server\default\deploy` to `jboss_home\server\all\farm`:

- `castylesr5.1.1.ear`
- `IdentityMinder.ear`
- `imauditdb-ds.xml`
- `imtaskpersistencedb-ds.xml`
- `imworkflowdb-ds.xml`
- `objectstore-ds.xml`
- `reportsnapshot-ds.xml`
- `SiteMinderAgent.ear` (this is true even if SiteMinder is not installed)

5. Copy `login-config.xml` from `jboss_home\server\default\conf` to `jboss_home\server\all\conf`.

6. Copy the following files from `jboss_home\server\default\lib` to `jboss_home\server\all\lib`:

- `sqljdbc.jar` (If MS-SQL is the object store)
- `agentfilter.jar`
- `ojdbc14.jar` (If Oracle is the object store)

7. Set up JMS connection factory, topics and queues definitions as follows:

- a. Copy `samples\Cluster\JBoss\deploy-hasingleton\jms\workflow-service.xml` to `jboss_home\server\all\deploy-hasingleton\jms`.
- b. Remove `jboss_home\server\all\farm\IdentityMinder.ear\META-INF\workflow-service.xml`.
- c. Locate the `jbossmq-destinations-service.xml` in `jboss_home\server\all\deploy-hasingleton\jms`.
- d. Add JMS topic and queue definitions to this file by copying in the content of the file:

```
jboss_home\server\all\farm\IdentityMinder.ear\META-INF\jbossmq-destinations-service.xml
```

Important! Copy the content only in between the server tags.

- e. Remove these files:

```
jboss_home\server\all\farm\IdentityMinder.ear\META-INF\jbossmq-destinations-service.xml.  
jboss_home\server\all\farm\IdentityMinder.ear\identityminder_ejb.jar\META-INF\jbossmq-  
destinations-service.xml
```

8. Copy `samples\Cluster\JBoss\IdentityMinder.ear` over `jboss_home\server\all\farm\IdentityMinder.ear`

If Workflow Designer is installed, copy `samples\Cluster\JBoss\IdentityMinder.ear\config\workpoint-client.properties` to `<CA ROOT>\IAM Suite\Identity Manager\tools\Workpoint\conf`

9. Make these changes for HTTP load balancing with sticky sessions:

- Modify `jboss_home/server/all/deploy/jbossweb-tomcat55.sar\META-INF/jboss-service.xml` by changing the `UseJK` attribute to `true`.
- Modify `jboss_home\server\all\deploy\jbossweb-tomcat55.sar\server.xml` by adding a tag in this format:
`jvmRoute="worker1"` to `<Engine ...>`

Important! If you are installing a JBoss cluster on an IPv6/IPv4 stack, modify the sample files on each system and replace `jnp://localhost` with `jnp://host-name` used in `run_idm.bat` or `run_idm.sh`. Modify all `jboss.xml` files.

Add Cluster Nodes

To configure additional cluster nodes

If you need to run additional nodes of JBoss on the same system, modify the port assignments of JBoss based on the vendor instructions. You will need to modify the JNDI names in the procedure to create the first cluster node, so that it uses the correct port number.

Also, modify files from step 6 in the procedure to create the first cluster node; include the changed port number for this node.

The location of the new JBoss node is *jboss_home_n*. It may or may not reside on the same system as the first node of the cluster.

1. Copy the following files from *jboss_home\bin* to the additional cluster node location: *jboss_home_n\bin*:
 - `workpoint_client.policy`
 - `run_idm.bat`
Remember to change the *host-name* used by the `-b` argument to the name of the system where you copy this file.
 - `run_idm.ico`
2. Copy the following files from *jboss_home\server\all\deploy-hasingleton\jms* to *jboss_home_n\server\all\deploy-hasingleton\jms*:
 - `jbossmq-destinations-service.xml`
 - `workflow-service.xml`
3. Migrate native libraries under Program Files\CA\IAM Suite\Identity Manager\agentframework or a UNIX location to each system in the cluster.
4. For HTTP load balancing with sticky sessions:
 - Modify *jboss_home_n/server/all/deploy/jbossweb-tomcat55.sar/META-INF/jboss-service.xml* to set the `UseJK` attribute to `true`.
 - Modify *%JBOSS_HOME_N%\server\all\deploy\jbossweb-tomcat55.sar\server.xml*, by adding this tag:
`jvmRoute="workerN"` to `<Engine ...>`
5. Copy `login-config.xml` from *jboss_home\server\default\conf* to *jboss_home_n\server\all\conf*.
6. Copy the following files from *jboss_home\server\default\lib* to *%JBOSS_HOME_N%\server\all\lib*:
 - `sqljdbc.jar` (MS-SQL)
 - `agentfilter.jar`
 - `ojdbc14.jar` (Oracle)

7. Copy the following files of *jboss_home*\server\all\farm to %JBOSS_HOME_N%\server\all\farm:
 - castylesr5.1.1.ear
 - IdentityMinder.ear
 - SiteMinderAgent.ear (this applies even if SiteMinder is not installed)
8. To create a data source, place the data source file under all/farm on one cluster node.

Important! If you are installing a JBoss cluster on an IPv6/IPv4 stack, modify the sample files on each system and replace `jnp://localhost` with `jnp://host-name` used in `run_idm.bat` or `run_idm.sh`. Modify all `jboss.xml` files.

WebSphere Clusters

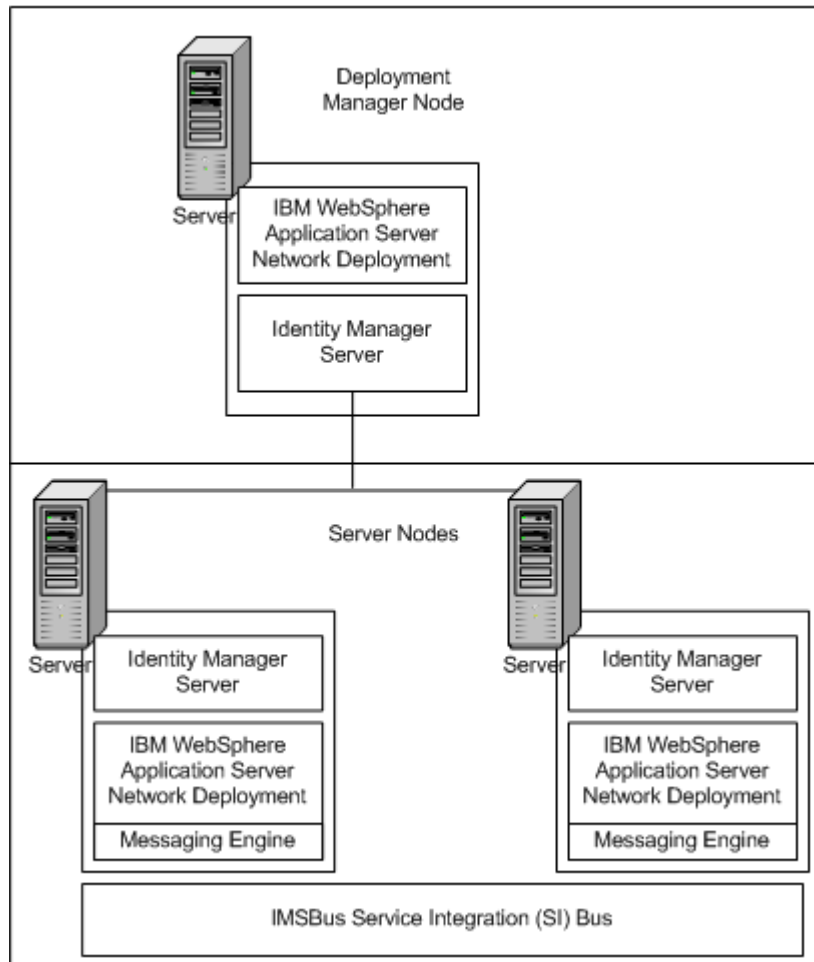
When you install software for a WebSphere 6.0 cluster, you set up the following:

- One WebSphere Deployment Manager—Manages the nodes in a cluster. The Deployment Manager deploys the Identity Manager Server to all nodes in a cluster.
- One or more server nodes—Each server node in a cluster runs the Identity Manager Server.
- Service Integration Bus—Groups resources in WebSphere 6.0 to simplify administration. The WebSphere cluster is added as a member of the bus.
- Messaging Engine—Provides messaging functionality for members of the service integration bus. Each node in the cluster has a messaging engine.
- Message Store—Stores messages and transaction status for a messaging engine. Each message engine requires a message store.

Note: For more information about these components, see the [WebSphere Application Server V6 System Management and Configuration Handbook](#).

The following figure shows the relationship between the Deployment Manager and the server nodes in a WebSphere cluster. The Identity Manager Server is installed on the Deployment Manager system and is automatically deployed to each node in the cluster.

WebSphere 6 Application Server Cluster



WebSphere Cluster Prerequisites

Before you configure Identity Manager on a WebSphere 6.1 cluster, you should be familiar with the concepts and procedures for creating a WebSphere 6.1 cluster.

See the following documents on the IBM web site for more information about WebSphere clusters:

- [WebSphere Application Server V6 System Management and Configuration Handbook](#)
- [WebSphere Application Server V6 Scalability and Performance Handbook](#)

Create a WebSphere Cluster

You set up a WebSphere 6.1 cluster in the WebSphere Administrator Console.

To create a WebSphere 6.1 cluster for Identity Manager

1. Decide which systems you plan to use for the cluster.
 - a. Select a system for the WebSphere Deployment Manager. For best performance, the system should not be used as a server node.
 - b. Determine the server nodes in the cluster.
2. Install the WebSphere Deployment Manager using the [IBM WebSphere documentation](#) for the most recent instructions.

During the installation:

 - Install the Deployment Manager.
 - Note the directory where you install the Deployment Manager.

The directory is represented as *directory_manager_dir* in later portions of the installation.
3. In the WebSphere Admin console, set the IM JMS Msg Queue Max Sessions to 5 in this location:

Application Servers, server1, Message Listener Service, Listener Ports, Maximum Sessions
4. Install the IBM WebSphere Application Server Network Deployment V6 software on the Deployment Manager machine.

When the installation completes, you are prompted to configure a *profile*, a WebSphere runtime environment.
5. Run the Profile Creation Wizard to create the profile for the Deployment Manager machine. When you are prompted to select a profile type, select the Deployment Manager profile.

6. Start the Deployment Manager using one of the following methods:
 - Run the StartManager.bat (Windows) or StartManager.sh (Solaris) from a command prompt.

These scripts are located in the bin directory where WebSphere is installed.
 - If you registered the Deployment Manager as a Windows Service, use Windows Services to start the Deployment Manager.
7. Install the IBM WebSphere Application Server Network Deployment V6 software on each server node in the cluster.
8. Use the Profile Creation Wizard to create a Custom profile for each node.

A Custom profile allows you to configure a connection to the Deployment Manager.
9. Start each node as follows:
 - a. Navigate to `was_home\WebSphere\AppServer\bin` on the system where the managed node is located.

Execute the `startNode.bat\sh` command.

Note: Identity Manager does not support HTTP session persistence in a clustered environment.
10. Confirm that a single cell has all the nodes associated with it at this location: :
`<WAS_HOME>/profiles/Deployment_Manager_Profile/config/cells/<Cell_Name>/Nodes/`

You should see all federated nodes displayed as folder names.

The servers associated with the profiles created should be unique names within the cell, which automatically applies to cluster, since cell has a higher scope than cluster.

Creation of profiles may sometimes fail if the bootstrap ports (default: 2809) are not unique.]You can check for an error message in the `pctLog.txt` file in the created profiles' logs folder. For example:
(Oct 10, 2007 6:45:55 PM), Install, com.ibm.ws.install.ni.ismp.actions.ISMPWSPProfileLaunchAction, err, INSTCONFFAILED: Cannot complete required configuration actions after the installation. The configuration failed. The installation is not successful. Refer to C:\Program Files\IBM\WebSphere\AppServer\logs\wasprofile\wasprofile_create_CustomIMFromNode.log for more details.

Inspecting the `wasprofile_create_CustomIMFromNode.log` shows this failure was due to Bootstrap ports that is not unique.

Configure the Cluster with the First Node

You now configure the cluster with a single node. The other nodes are added in a subsequent procedure after you install Identity Manager.

To configure the cluster with the first node

1. In the Administrative Console for the Deployment Manager, go to System Administration, Nodes.
2. Verify that the nodes shows a Synchronized status.
3. Select Servers, Clusters and click New.
4. In Step 1 of the Create New Cluster wizard, enter a name for the cluster in the Cluster Name field—for example, `im_cluster`.
5. Note the cluster name, which is represented as `cluster_name` in subsequent procedures.
6. In Step 2 of the Create New Cluster wizard, enter a name for the server node (for example, `im_cluster_server1`) and select the node from the list box. Click Apply.
7. Click Next, Finish, Save.
8. Stop all servers.

You may leave the Deployment Manager running.

Add Nodes to the Cluster

You can now add nodes to the cluster using the server template from the first node.

To add nodes to the cluster

1. In the Administrative Console for the Deployment Manager, go Servers, Clusters.
2. Add nodes, selecting the default server template.
3. Repeat this procedure for any additional nodes you wish to add to cluster.

Install Identity Manager

You install Identity Manager as described in the following procedure. During the installation, the following EARs are installed on the cluster domain:

- IdentityMinder.ear
- ca-stylesr5.1.1.ear

When you supply a cluster name during the installation, these primary resources are configured:

- Distributed queues/topics targeted to cluster name provided
- Connection factories targeted to server name provided
- Data sources also targeted to cluster name provided
- MSBus, the Service Integration Bus for Identity Manager

To install Identity Manager on the first server

1. On the system that hosts the WebSphere Deployment Manager, run the Identity Manager installation.
2. Complete the WebSphere section of the installation as follows:

Server Name

Use the name of one of the servers in the WebSphere cluster.

Profile Name

The deployment manager profile.

Cell Name

The deployment manager's cell which can be found at path:
was_home/profiles/Deployment_Manager_Profile/config/cells/Cell_Name/Nodes/

Node Name

A node that contains the Server Name you supplied on this screen.

Cluster Name

The name of the cluster.

App Server URL and port

The application URL and port number of the system that will host the application server used by the Identity Manager Server.

WebSphere Install Folder:	C:\WebSphere\AppServer	Restore Default	Choose...
Server Name:	Member1		
Profile Name:	<Deployment_Manager_Profile_Name>		
Cell Name:	<Deployment_Manager_Cell_Name>		
Node Name:	Node1		
Cluster Name:	imR12Cluster		
App Server URL and port:	http://iam-fw-was6.ca.com:9080		

To install Identity Manager on each additional server

1. From the deployment manager, navigate to *was_home/profiles/Deployment_Manager_Profile/bin*.
2. Use execute wsadmin as follows for each additional server in the cluster:

```
wsadmin -f ims6SetupClusterMember.jacl NodeName ServerName ClusterName  
MEDataSourceJndiName
```

3. Repeat step 2 once for each additional server in the cluster.

At this point, Identity Manager is installed in the cluster.

To verify that Identity Manager starts

1. Use stopNode on each node under Custom_Profile/bin.
2. Run stopManager on the Deployment Manager system under deployment manager profile/bin.
3. Use startManager followed by startNode on each node.
4. Start each server in the cluster from the Admin Console.
5. Verify that Identity Manager starts by checking that this URL displays the Management Console.

```
http://im_server_hostname:port/ldmmanage
```

Create Core Group Policies

The final step in creating the WebSphere 6.1 cluster is creating core group policies that control the distribution of the message engines across the servers in the cluster.

A core group policy allows you to define a preferred server for the messaging engine. If that server fails, then the messaging engine fails over to other servers in the cluster, but will return to the preferred server as soon as it becomes available. This type of policy enables high availability and workload management within the WebSphere cluster.

For detailed information on core group policies, see the section on setting up preferred servers in the IBM WebSphere Application Server V6 System Management and Configuration Handbook, which is available from the [IBM Web site](#).

To create a core group policy

1. In the WebSphere Console, create a new policy for the DefaultCoreGroup with the following settings:

- Policy Type: One of N
- Options: Select the Fall Back and Preferred Servers Only

Note: Do not delete or modify the default policies.

2. Create a new match criteria for the policy you created with the following properties:

- Name: type
- Value: WSAF_SIB

3. Create another match criteria with the following properties.

- Name: WSAF_SIB_MESSAGING_ENGINE
- Value: *name of the first messaging engine on the IMSBus*

WebSphere automatically generates the name of the messaging engine when you create it. The name has the following format:

cluster_name.00n-IMSBus

where *cluster_name* is the name of the cluster you are configuring, and *n* represents a unique number for the messaging engine, which is automatically incremented each time a messaging engine is created for the cluster.

For example, if the cluster name is mycluster, and there are two messaging engines, the names would be:

mycluster.001-IMSBus

mycluster.002-IMSBus

4. Return to the configuration page for the policy you are creating.
5. Select the servers you want to configure as preferred servers for the new policy.

You can select as many servers as needed from the servers in the cluster where the messaging engine is defined. Do not select node agents or the Deployment Manager.

The first server in the list is the server that the messaging engine will use by default. Move the servers up or down in the list until they appear in the order in which they should be used.

6. Click OK to save the changes.
7. Use the instructions in steps 1-6 of this procedure to create a policy for each node in the cluster.

Update Cluster Members

From the Deployment Manager system where you installed Identity Manager, make these changes.

To update cluster members

1. From the Deployment Manager system, copy these binary folders to same location on each cluster member.
 - Custom
 - Config
 - User_console.war
 - Management_console.war

Important! Do not delete folders on the cluster members; overwrite the folders instead.

2. Copy the folders from the following location on the Deployment Manager system to the corresponding IdentityMinder.ear folder on the cluster member:

`<WAS_HOME>\profiles\ManagerProfile\installedApps\.\IdentityMinder.ear`

3. Confirm that the deployment.xml file exists here for each node:

`<WAS_NODE_HOME>\profiles\<Profile Name>\config\cells\<Cell Name>\applications\IdentityMinder.ear\deployments\IdentityMinder\deployment.xml`

4. Locate the following section of the file:

```
<classloader xmi:id="Classloader_1202404294500" mode="PARENT_FIRST"/>

<modules xmi:type="appdeployment:WebModuleDeployment"
xmi:id="WebModuleDeployment_1202404294500" deploymentId="1" startingWeight="4000"
uri="user_console.war

```

5. Change this section, where **user_console.war** is shown in bold, so that it appears as follows:

```
<classloader xmi:id="Classloader_1202404294500" mode="PARENT_FIRST"/>

    <modules xmi:type="appdeployment:WebModuleDeployment"
xmi:id="WebModuleDeployment_1202404294500" deploymentId="1" startingWeight="4000"
uri="user_console.war" classloaderMode="PARENT_LAST">

    <targetMappings xmi:id="DeploymentTargetMapping_1202404294501"
target="ServerTarget_1202404294500"/>

    <classloader xmi:id="Classloader_1202404294501"/>

</modules>
```

6. Update the Workpoint-client.properties file with the port for the BOOTSRAP_ADDRESS for the WebSphere profile.
 - a. Start the WebSphere Admin Console.
 - b. Navigate to Application Servers, *server_name*, Communications, Expand Ports.
 - c. Edit Workpoint-client.properties file under IdentityMinder.ear/config.
 - d. Change the default port 2809 in the WebSphere section to the profile's port for the BOOTSRAP_ADDRESS.
7. Repeat this procedure for each member node in the cluster.
8. Restart the nodes.

Create a Message Store

The Service Integration Bus includes a messaging engine, which manages the communications for the Bus. The messaging engine stores information such as messages, subscription information, and transaction states, in a message store.

The message store can be any database that WebSphere can access remotely, but it must be a non-XA provider implementation.

To create a message store

1. You use IBM's `sibDDLGenerator` command to generate the DDL statements needed to create the Database Management System (DBMS) resources used by the messaging engine.

For instructions on using this utility, see IBM's documentation for the [sibDDLGenerator](#).

2. After you create the DDL statements, you import them into the message store.
3. To enable the messaging engine to use the database, you configure a JDBC Driver for the database type, and a data source for the cluster in the WebSphere Console.

Note the jndi name when you create the data source.

Configure a message store for each node in the cluster.

4. Before testing the connection to the data source, restart the node agent on the node where you created the data source. If you do not restart the node agent, the connection test fails.

Messaging engine names are created in the format: *Cluster Name.000-Bus Name*, so if your cluster name is `imR12Cluster`, you will see a single messaging engine under Service Integration -> Buses -> IMSBus -> Messaging engines:

`imR12Cluster.000-IMSBus`

5. Create additional messaging engines for each server added to the cluster. Go to Service Integration, Buses, IMSBus, Bus members, Messaging engines and click on Add messaging engines.

The installation appends the node and server names for the server specified during installation as the Data source JNDI name under Service Integration, Buses, IMSBus, Messaging engines, *message_engine_name*, Data store.

This jndi name may be changed to whatever is desired as long as it is reflected in the relevant message store.

After you create the message store, [create core group policies](#) (see page 59) which determine the distribution of messaging engines among servers in the cluster.

SiteMinder High Availability for the Application Server Cluster

If you have created a SiteMinder Policy Server cluster, you can configure the application server cluster to use it for load balancing and failover.

You edit the ra.xml file in one of these locations:

For WebLogic:

wl_domain/applications/IdentityMinder.ear/policyserver_rar/META-INF

For JBoss:

jboss_home/server/default/IdentityMinder.ear/policyserver_rar/META-INF

For WebSphere:

was_im_tools_dir/WebSphere-ear/IdentityMinder.ear/policyserver_rar/META-INF

Note: After making changes in the IdentityMinder.ear directory in *was_im_tools_dir/WebSphere-ear*, you must package and deploy a new IdentityMinder EAR file for changes to take effect.

The following sections explain how to modify:

- Connection settings for the Policy Server
- The number of Policy Servers
- The selection of load balancing or failover for the cluster

Modify Policy Server Connection Settings

The Policy Server connection information should reflect the primary server for the production environment. This information consists of the ConnectionURL, the user name and password for the SiteMinder Admin account, and the name and shared secret for the Agent.

In the following example, boldface shows the values to edit.

```
<config-property>
  <config-property-name>ConnectionURL</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>developmentserver.company.com,44441,44442,44443</config-
    property-value>
</config-property>

<config-property>
  <config-property-name>UserName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>SiteMinderAdminName</config-property-
    value>
</config-property>

<config-property>
  <config-property-name>Password</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>encryptedpassword</config-
    property-value>
</config-property>

<config-property>
  <config-property-name>AgentName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>DevelopmentAgentName</config-property-
    value>
</config-property>

<config-property>
  <config-property-name>AgentSecret</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>encryptedagentsecret</config-
    property-value>
</config-property>
```

Note: For the values that require encrypted text, use the Identity Manager password tool. See the *Configuration Guide*.

Add More Policy Servers

To add more Policy Servers to the Identity Manager installation instance, edit the FailoverServers entry in the ra.xml file.

Note: Include the primary Policy Server and all failover servers in the FailoverServers entry.

For each Policy Server, enter an IP address followed by port numbers for authentication, authorization, and accounting services. Use a semi-colon to separate entries as shown here:

```
<config-property>
  <config-property-name>FailoverServers</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>
    172.123.123.123,44441,44442,44443;172.123.123.124,33331,
    33332,33333
  </config-property-value>
</config-property>
```

Select Load Balancing or Fail Over

The default behavior of Identity Manager is to use round-robin load balancing using the servers identified by the ConnectionURL and FailoverServers. Load balancing occurs if you leave FailOver set to false.

To select failover, change FailOver to true:

```
<config-property>
  <config-property-name>FailOver</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>true</config-property-value>
</config-property>
```

Load-Sharing Limitations

Server-tier load sharing has the following limitations:

- Changes to log settings are not propagated to all servers
- Queries to a server for operation details of operations initiated by another server may not obtain the intended results
- Load sharing is done at the LDAP operation level

We strongly recommend that you use the failover configuration techniques rather than load sharing because of these limitations.

Policy Server and Application Server Communication

You need to enable communication between SiteMinder and a WebSphere, WebLogic 9.2, or a JBoss cluster.

To enable communication with a WebSphere cluster

1. Copy ETPKI_setup from Node Manager machine where Identity Manager r12 installer was installed to each cluster node.

This file is in C:\Program Files\CA\IAM Suite\Identity Manager (on Windows) or /CA/IAM_Suite/Identity_Manager/ (on Solaris).

2. Run ETPKI_setup using from command line or shell as follows:
ETPKI_setup install caller=IAMFW

3. Create a new PATH variable (on Windows) or LD_LIBRARY_PATH (on Solaris) for each node from the admin console at:
Application servers, *node_name*, Process Definition, Environment Entries

4. Add a path to the DLLs for SiteMinder and ETPKI:

On Windows:

C:\ProgramFiles\IBM\WebSphere\AppServer\profiles\profile_name\installedApps\cellname\IdentityMinder.ear\library;C:\Program Files\CA\SC\ETPKI\lib

On Solaris:

/opt/WebSphere/AppServer/profiles/profile_name/installedApps/profile_name/IdentityMinder.ear/library
:/opt/CA/SharedComponents/ETPKI/lib/

5. Restart the system so changes will take effect.

To enable communication with a WebLogic cluster

1. Copy ETPKI_setup from Node Manager machine where Identity Manager r12 installer was installed to each cluster node.

This file is in C:\Program Files\CA\IAM Suite\Identity Manager (on Windows) or /CA/IAM_Suite/Identity_Manager/ (on Solaris).

2. Run ETPKI_setup using from command line or shell as follows:
ETPKI_setup install caller=IAMFW

3. Edit commenv.bat/sh and add as follows:

On Windows:

```
set
IM_SM_PATH=C:\bea\weblogic92\common\nodemanager\servers\<SERVER_NAME>\stage\IdentityMind
er\IdentityMinder.ear\library;C:\Program
Files\CA\SC\ETPKI\lib
set PATH=%IM_SM_PATH%;%PATH%
```

On Solaris:

Append the following to LD_LIBRARY_PATH section under

```
LD_LIBRARY_PATH=/bea/weblogic92/common/nodemanager/servers/serv
er_name/stage/IdentityMinde/IdentityMinder.ear/library:/opt/CA/SharedC
omponents/ETPKI/lib/
```

This will be before the export LD_LIBRARY_PATH call.

4. Restart the system so changes will take effect.

To enable Siteminder communication with a JBoss cluster

1. Copy ETPKI_setup from Node Manager machine where Identity Manager r12 installer was installed to each cluster node.

This file is in C:\Program Files\CA\IAM Suite\Identity Manager (on Windows) or /CA/IAM_Suite/Identity_Manager/ (on Solaris).

2. Run ETPKI_setup using from command line or shell as follows:
ETPKI_setup install caller=IAMFW
3. Copy the run_idm.bat/sh from the machine where the Identity Manager r12 installer was run, and make sure the correct path to IdentityMinder.ear/library is referenced in variable PATH (on Windows) or LD_LIBRARY_PATH (on Solaris).
4. Restart the system so changes will take effect.

Migrate Workflow Process Definitions

If you are creating a cluster as part of a migration procedure and you used workflow in the development environment, you need to export the workflow definitions in the development environment and import them into the production environment. Then, configure workflow in each of the server nodes.

Export process definitions

On the development environment system, you export the workflow process definitions.

To export process definitions

1. Make sure the application server is running.
2. Go to *im_admin_tools_dir*\Workpoint\bin\ and run Archive.bat (for Windows) or Archive.sh (for UNIX) as follows:
 - a. In the Import dialog, select the root object.
 - b. Click Add.
 - c. Specify the name of the file to generate.
 - d. Click Export.
 - e. Click Go.
3. Follow the instructions in the next section, [To import process definitions](#) (see page 68).

Import process definitions

On the production environment system, import the workflow process definitions.

To import process definitions

1. Restart the application server.
2. Optionally, create a backup copy of your current definitions by exporting the definitions using the preceding procedure.
3. Go to *im_admin_tools_dir*\Workpoint\bin\ and run the Archive script as follows:
 - a. In the Import dialog, select all items to import.
 - b. Supply the name of the file generated by the export.
 - c. Click Go.
4. Follow the instructions in the next section to configure the web server cluster for the application server.

Web Server Clusters

If you are using a Web Server cluster, you install a SiteMinder Web Agent to protect access to Identity Manager resources and configure a proxy plug-in.

Create a SiteMinder Web Agent instance that will be used only to protect Identity Manager resources. This Web Agent needs its own configuration objects and should not protect other applications or resources.

Important! If Identity Manager is installed to support IPv6 addresses, the web server plug-in for the application server must also support IPv6.

Configure the Proxy Plug-In (WebLogic)

After configuring the SiteMinder Web Agent, you modify the proxy plug-in file that corresponds to the Web Server on which you installed the Web Agent. Use one of the following procedures to modify the plug-in file.

Modify the Plug-in for an IIS Web Server

To modify the plug-in for an IIS Web Server

Make these changes to the iisproxy.ini file:

1. Remove or comment out the WebLogicHost and WebLogicPort entries. For example, comment out these entries:

```
WebLogicHost=localhost
```

```
WebLogicPort=7001
```

2. Add a WebLogicCluster entry in this format:

```
WebLogicCluster="wl_host:port,wl_host:port,..."
```

For example, this cluster has two nodes:

```
WebLogicCluster=north.com:7101,south.com:7201
```

Modify the Plug-in for an iPlanet Web Server

To modify the plug-in for an iPlanet Web Server

Make these changes to the obj.conf file:

1. Locate each line that includes WebLogicHost and WebLogicPort parameters. For example:

```
Service fn="wl-proxy"
```

```
WebLogicHost="north.com" WebLogicPort="7001" PathTrim="/weblogic"
```

2. Replace these parameters with a WebLogicCluster parameter in this format:

```
WebLogicCluster="wl_host:port,wl_host:port,..."
```

For example, this cluster has two nodes:

```
Service fn="wl-proxy" WebLogicCluster="north.com:7001,south.com:7001" PathTrim="/weblogic"
```

Configure the JK Connector (JBoss)

To configure the JK connector

1. Install a JK connector based on these instructions:

http://tomcat.apache.org/connectors-doc/generic_howto/quick.html

Note: If you installed SiteMinder, choose one that corresponds to the Web Server on which you installed the SiteMinder Web Agent that will protect Identity Manager resources.

2. Configure JK load balancing as follows:
 - a. Locate ConnectorConfiguration in the sample directory appropriate for your Web Server/operation system.
 - b. Use property files in samples\Cluster\JBoss\ConnectorConfiguration instead of the files shipped with the ConnectionConfiguration sample.

Configure the Proxy Plug-In (WebSphere)

For WebSphere 6.1, you install the proxy plug-in and then configure it with the GenPluginCfg command.

To install the proxy plug-in

1. Install the proxy plug-in from the WebSphere Launch Pad.
2. Add the Web Server to the WebSphere cell by running the configurewebserver1.bat command:
 - a. Edit `was_home\Plugins\bin\configurewebserver1.bat/.sh` in a text editor.
 - b. Add a user name and password to after `wsadmin.bat/.sh` as follows:

```
wsadmin.bat -user wsadmin -password password -f  
configureWebserverDefinition.jacl
```
 - c. Run `configurewebserver1.bat/.sh`.

Note: See the IBM WebSphere documentation for more information about the configurewebserver command.

3. From the command line, navigate to `was_home\bin`, where `was_home` is the installed location of WebSphere.

For example:

- For Windows:

`C:\Program Files\WebSphere\AppServer\bin\`

- For UNIX:

`/<home_dir>/WebSphere/AppServer/bin`

4. Run the GenPluginCfg.bat or GenPluginCfg.sh command.

Running this command generates a plugin-cfg.xml file in `was_home\Plugins\config\webserver1\config\cells`.

5. If the application server is on a separate system from the one with the Web server, copy the plugin-cfg.xml to the `was_home\Plugins\config\webserver1\config\cells` directory on the system where you installed the proxy plug-in.

6. Restart the Web server to activate the plug-in.

- For IIS Web Servers—In the master WWW service, be sure that the WebSphere plug-in (sePlugin) appears after the SiteMinder Web Agent plug-in and that the WebSphere plug-in started successfully.
- For iPlanet Web Servers—Be sure that the WebSphere plug-in (libns41_http.so) is loaded after the SiteMinder Web Agent plug-in (NSAPIWebAgent.so)

For iPlanet 6.0 Web Servers, check the order of plug-ins in
<iplanet_home>/https-instance/config/magnus.conf.

For iPlanet 5.x Web Servers, copy the following lines from
<iplanet_home>/https-instance/config/magnus.conf to
<iplanet_home>/https-instance/config/obj.conf

```
Init fn="load-modules" funcs="as_init,as_handler,as_term"  
shlib="/export/WebSphere/AppServer/bin/libns41_http.so"  
Init fn="as_init" bootstrap.properties="/export/WebSphere/AppServer/config/cells/plugin-cfg.xml"
```

Add the following after AuthTrans fn="SiteMinderAgent" in the obj.conf file:

```
Service fn="as_handler"
```

- For Apache Web Servers— In the Dynamic Shared Object (DSO) Support section of *Apache_home*/config/httpd.conf, be sure that the SiteMinder Web Agent plug-in (mod2_sm.so) is loaded before the WebSphere plug-in (mod_ibm_app_server_http.so).

Cluster Startup

After you configure the application server cluster, the Policy Server cluster, and the Web server cluster, you can start the application server cluster nodes.

Start the Servers for WebLogic

To start the servers for WebLogic

Once all configuration is complete, start all servers in this order:

1. If necessary, start one of the SiteMinder Policy Servers that supports Identity Manager.

Note: If you have a Policy Server cluster, only one Policy Server should be running while you create Identity Manager directories, create or modify Identity Manager environments, or change WorkPoint settings.

2. If necessary, start the WebLogic Administration Server:
 - a. Change to the domain directory you created for the cluster.
 - b. Start WebLogic.
3. Do one of the following:
 - Start the clustered server instances by running one of the following commands on each of the nodes you created:
 - For WebLogic 8.1:
`startManagedWeblogic.cmd ServerName AdminServerAddress:port`
 - For WebLogic 9.2:
`startManagedWeblogic.cmd ServerName AdminServerAddress:port`

Note: Before using the `startManagedWebLogic.cmd` file to start each node in the WebLogic cluster, update the paths in the file to point to the deployed `IdentityMinder.ear` on the managed server node. If these paths are not set correctly, errors occur when the cluster tries to start.

 - Use the WebLogic Node Manager to start the clustered Managed Server nodes from the Admin Server. This requires that the Node Manager is installed and running on each of the physical systems that hosts the clustered servers.
4. Start the Web Server where you installed the SiteMinder Web Agent and the application server proxy plug-in.

Start the Servers for JBoss

Once all configuration is complete, start all servers in the correct order.

To start the servers for JBoss

1. Start the Policy Server on all the clustered server nodes (if you have SiteMinder installed).
2. From a command line, navigate to:
`jboss_home/bin`
For example, the Windows default is:
`C:\CA\CA Identity Manager\jboss-4.0.5\bin`

3. Enter the following:
 - For Windows:
`run_idm.bat -c all`
 - For UNIX:
`./run_idm.sh -c all`
4. Start the Web Server where you installed the SiteMinder Web Agent and the application server proxy plug-in.

Start the Cluster for WebSphere

To start a WebSphere 6.1 cluster, you start the Deployment Manager and then start each managed node.

To start the WebSphere 6.1 cluster

1. Make sure the `agentfilter.jar` file is in the correct location on the Deployment Manager and each managed node.
2. Run the Deployment Manager.
3. On the first managed node, complete the following steps:
 - a. Navigate to `was_home\WebSphere\AppServer\bin`.
 - b. Execute the `startNode.bat\sh` command.The first managed node starts.
4. Repeat step 2 on each node in the cluster.
5. Start each cluster member in *Servers, Clusters, cluster name*, Cluster Members in the WebSphere Administrative Console on the Deployment Manager.
6. Make sure that the messaging engines for the cluster is running in *Service integration, Buses, IMSBus, Messaging Engines* in the WebSphere Admin Console on the Deployment Manager.

Verify the Clustered Installation

When you have completed all steps using an application server cluster, check that the installation was successful.

To verify the clustered installation

1. Start the Identity Manager Management Console as follows:

`http://domain_name:port/idmmanage`

host_name

Defines the fully-qualified host name for the server where Identity Manager is installed

port

Defines the application server port.

2. If you migrated an Identity Manager environment, access the Identity Manager environment as follows:
 - a. Enter the URL for the Identity Manager environment.
 - b. Verify that you are prompted for the appropriate credentials.
 - c. Log in using the account with the System Manager role.
 - d. Verify the correct roles are assigned to this account.
3. If these steps succeeded, start any extra Policy Servers and Identity Manager nodes that you stopped.

Note: If you still need to make changes to an Identity Manager environment, omit this step until you are done.

Update the Migrated Environments

If you are using a different user directory in migrated Identity Manager environments, assign the system manager role to a new user.

To update a migrated environment

1. If you have a Policy Server cluster, check that only one Policy Server is running.
2. Stop all but one Identity Manager node.
3. Log into Identity Manager Management Console.
4. Select the Identity Manager environment.
5. Select System Manager.

6. Enter the unique identifier of the user with the System Manager role and click Validate.

If no problems occurred for the preceding steps, the installation is successful. You can now use the production environment for standard Identity Manager tasks.

Chapter 5: Provisioning Components

Based on the guidelines in this chapter, you implement high availability for provisioning components by installing more than one instance of a component. However, the installation details exist elsewhere. For the Java Connector Server, see the *Java Connector Server Implementation Guide*. For other components, see the *Installation Guide*.

This section contains the following topics:

- [Load-Sharing Limitations](#) (see page 77)
- [Preconfiguration Considerations](#) (see page 78)
- [Planning](#) (see page 78)
- [Sample Provisioning Configuration](#) (see page 86)
- [Primary and Alternate Provisioning Servers](#) (see page 88)
- [Provisioning Domain Specifications](#) (see page 88)
- [ROOT Domain Installation](#) (see page 90)
- [Domain Prerequisites](#) (see page 91)
- [How to Make Provisioning Clients Highly Available](#) (see page 93)
- [How to Make Provisioning Servers Highly Available](#) (see page 95)
- [How to Make Provisioning Directories Highly Available](#) (see page 101)
- [How to Make Connector Servers Highly Available](#) (see page 121)

Load-Sharing Limitations

Server-tier load sharing has the following limitations:

- Changes to log settings are not propagated to all servers
- Queries to a server for operation details of operations initiated by another server may not obtain the intended results
- Load sharing is done at the LDAP operation level

We strongly recommend that you use the failover configuration techniques rather than load sharing because of these limitations.

Preconfiguration Considerations

Let us assume that you are starting with a new installation and configuration of Identity Manager. You should start with the type of high availability configuration that best fits your business needs. You can also select to create a high availability configuration in one or more tiers for your enterprise.

Before beginning your configuration, note the following important points:

- The Provisioning high availability solutions rely on advanced configuration techniques in eTrust Directory. We recommend that the persons involved with the configuration receive formal eTrust Directory training before attempting to configure failover and load balancing.
- You can use eTrust Directory routers to route Provisioning Server directory traffic using the X.500 protocol, and the Provisioning Server LDAP provisioning traffic with the Router DXlink.
- Identity Manager r12 provides the Connector Server Framework, which supports per directory or per managed system C++ Connector Server configuration. Multiple Connector Servers can be configured for each Provisioning domain, which increases the bandwidth to the managed systems. Each Connector Server is also an LDAP server, similar to the Provisioning Server.

Planning

A high availability deployment requires manual changes to the Provisioning Server and provisioning directory configurations, which potentially involves many provisioning directory configuration files, the use of a large number of network ports, and the naming of DSA and knowledge files. Advanced planning in the following areas is strongly advised:

- DSA configuration
- Global naming convention
- Consistent port assignments
- Monitoring and maintenance

DSA Configuration Considerations

All the provisioning directories for an Identity Manager installation present one single logical Directory Information Tree (DIT) to all Provisioning Servers. This DIT can be serviced by one or more DSAs on Windows or UNIX platforms. You can configure many DSAs for the Provisioning Server: one main DSA, a separate operations DSA, one primary multiwrite DSA and multiple read-only DSAs for searches. You can also specify write precedence amongst the DSAs using the set precedence statements in the appropriate DSA configuration files.

For DSA configuration, perform the following:

- If the amount of free physical memory is sufficient to cache all the data, configure DXCache DSA for performance. Please see below on eTrust Directory best practices. The sample configuration assumes that multiple read-only DSAs are still needed to provide additional bandwidth to the provisioning directory databases.
- Introduce one multiwrite DSA and multiple read-only, load-share DSAs for a provisioning directory, if the free physical memory is insufficient to cache all the data. In your deployment, decide in advance the number of DSAs for the provisioning directory. The sample configuration illustrates how to configure multiple DSAs, both multiwrite and read-only DSAs. You can remove the read-only DSA configuration files if they are not needed. In addition, you can adjust DSA configuration based on the eTrust Directory best practices to fit your enterprise business needs.
- Make the operations DSAs independent of the main provisioning directory DSA, as the operations DSA stores transient operation detail which may fragment the main provisioning directory database. Note that the in-memory cache-only DSA can also be introduced to service the transient operation detail.
- Set each Provisioning Server to search first through local provisioning directory DSAs, and then failover to remote DSAs.
- Designate one multiwrite DSA as the primary. All updates are sent to the primary and replicated to its peers.

eTrust Directory Best Practices:

- Consolidate eTrust Directory if it is deployed by multiple CA products.
- Configure one DXCache DSA for performance if the amount of free physical memory is sufficient to cache all the data. Please refer to eTrust Directory for further detail on DXCache.
- Decouple provisioning server installation from provisioning directory configuration.

- Deploy three directory servers at each region for disaster recovery. A single server should be able to accommodate 100% of the LDAP client load allowing two servers to be brought down concurrently. Introduce one or more null-prefix router DSAs prefixing each server.
- Deploy a single DXcache data DSA per database, if possible, for performance. CA Directory r12 mandates one DSA per database. The use of single DSA per database facilitates easier migration beyond Directory r12.
- Use synchronous multi-write between the servers within a region and asynchronous multi-write groups between regions.
- Define a preferred master data DSA per multi-write group, i.e. a write master at each region.
- Define a load-share-group for each region such that the router DSAs will forward search requests to one of the three servers at that local region only.
- The use of 64-bit machines is advised as this removes any limitation on the amount of data that can be stored in DXCache. The 32-bit directory server is restricted to 4GB RAM per DXcache DSA.

Global Naming Convention

The Provisioning high availability solutions involve the configuration of many eTrust Directory DSAs. All DSAs must know about all other DSAs. You achieve this goal by sharing the DSA knowledge files of one installation with another. A global naming convention allows each DSA to have a unique name. This makes it possible to share the DSA knowledge files without creating naming conflicts.

Using a global naming convention provides the following advantages:

- Allows all provisioning directory DSAs to source (refer to) the knowledge files of other DSAs when needed
- Prevents any naming conflict
- Clarifies the DSA configuration, and simplifies the DSA knowledge file management

In a high availability deployment, you must select a naming convention that fits your needs. For example, the default Provisioning Server installation configures a Provisioning directory with three DSAs named *etrustadmin*, *etadmintemp*, and *etaops_notify*. When the provisioning directory DSA is used for failover and load balancing, you should rename these DSAs to avoid naming conflicts.

DSA and Router Names Rule

The global naming convention for DSAs and routers, adopted in this document, is as follows:

`[imps | impd]_kindofdsa_name`

Name

Defines the host name

[imps | impd]

Represents the DXlink DSA for provisioning server or provisioning directory DSA.

kindofdsa

If provided, specifies the type of DSA, as one of the following:

- **not provided** - Represents the provisioning directory DSA.
- **temp** - Represents the provisioning directory DSA for transient operations details.
- **RO** - Represents the read-only provisioning directory DSA.
- **router** - Represents either a provisioning server DXlink router when prefixed by imps or a provisioning directory router when prefixed by impd.
- **notify** - Represents the provisioning notification directory DSA

Note: The limit on DSA names is 31 characters.

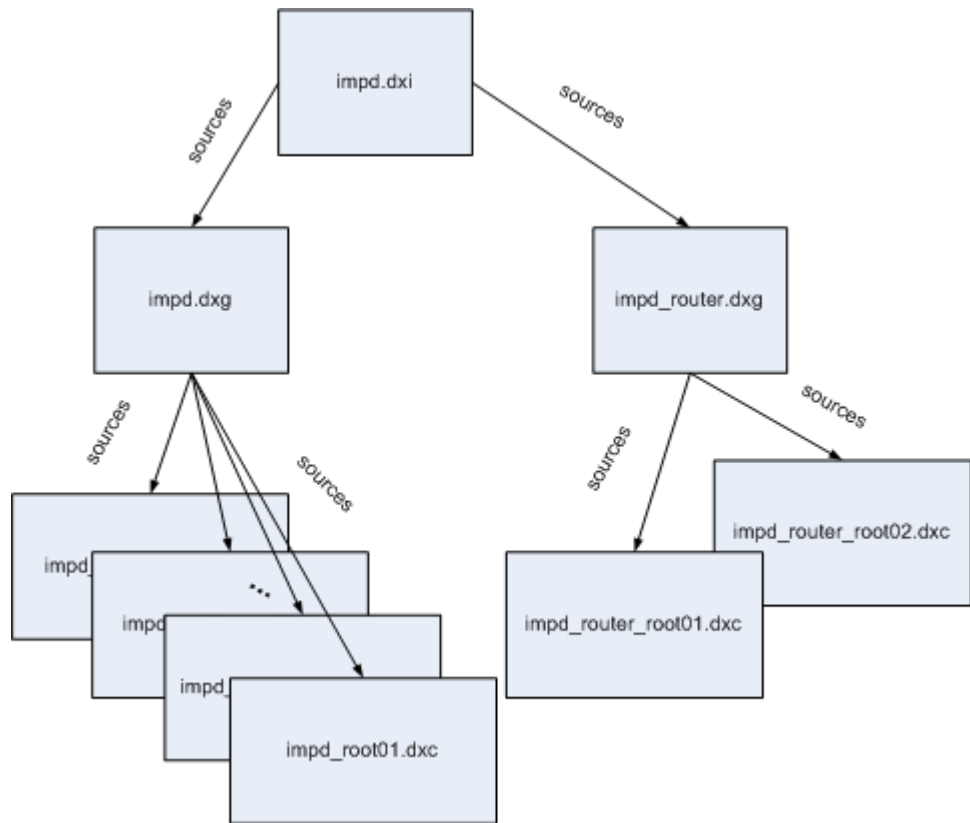
Examples:

- The provisioning directory DSA name on ROOT01 is impd_root01.
- The router name in charge of routing the provisioning directory traffic on ROOT02 is impd_router_root02.

Sample Configuration Files

Note: Optimal high availability solutions depend on enterprise information technology infrastructures and business requirements. Configurations described in this document are examples for your reference. You can adapt them for your specific requirements.

The DSAs source (reference) each other according to the following hierarchy of files:



The file names reflect the global naming convention used for each DSA. Using such a naming convention ensures that the file names are consistent.

- For simplicity, each DSA knowledge file receives the same name as its DSA. In addition, group DSA knowledge files are introduced to streamline the configuration.
- The DSA initialization file (.dxi) file always sources both impd.dxc and impd_router.dxc group knowledge files.
- The impd_router.dxc file sources all the provisioning directory routers in all the installations.
- The impd.dxc group knowledge file sources all the DSA knowledge files, including both primary and alternate server installation.
- The DXlink DSA for the Provisioning Server on ROOT01 is named imps_root01.
- The impd.dxc group knowledge file sources all the DSA knowledge files, including both primary and alternate server installations.

```

#
# Provisioning Directory group knowledge file
#
# The DSA group knowledge contains all the DSA knowledge
#
# The ordering of the DSAs follows the rules:
# (1) searches to the local read-only, load-share, DSAs, then remote ones
# (2) updates to the primary multi-write DSA, and replicate to peers
#
# Note that the ordering depends on which host the group knowledge file resides.
# Adjust the DSA search order based on ROOT01 and ROOT02 installations

source "impd_RO1_root01.dxc";
source "impd_RO2_root01.dxc";
source "impd_RO1_root02.dxc";
source "impd_RO2_root02.dxc";

# read-only, load-share, operations DSAs (optional)
source "impd_tempRO1_root01.dxc";
source "impd_tempRO1_root02.dxc";

# Update DSAs Preference
# Always update the primary multiwrite DSA and replicate to peers

source "impd_root01.dxc";
source "impd_root02.dxc";

source "impd_temp_root01.dxc";
source "impd_temp_root02.dxc";

# notification DSAs
source "impd_notify_root01.dxc";
source "impd_notify_root02.dxc";

```

Note: Installations may deploy only the write DSAs for operations details. In this case, there is no need to introduce read-only DSAs for transient operations detail. Furthermore, for performance reasons, a memory-resident DSA can be introduced for storing operations detail.

- The preference order of DSAs for a particular provisioning server installation is defined by setting the precedence and using a load-share-group. The advantage is that all group knowledge files contain the same content. Setting precedence is the recommended method for handling preference order. This approach also makes it easier to debug, as there is no need to open up group knowledge files one by one to find out the preference order.

Note: eTrust Directory provides the Set Precedence command, which can be used to set the DSA precedence.

You can also define the preference order by shuffling the order of DSA knowledge files in the group knowledge file. For example, the `impd_root.dxc` file on `root02` sources the same DSA knowledge files, except that the order of DSAs is rearranged to fit the needs of the `root02` server installation:

Note: Though this approach is possible, it is not the recommended method, as it can make debugging more difficult.

```
#
# Provisioning Directory Group knowledge file for Root Domain on root02
#
# Search DSAs Preference
# Adjust the DSA search order based on ROOT01 and ROOT02 installations

source "impd_RO1_root02.dxc";
source "impd_RO2_root02.dxc";
source "impd_RO1_root01.dxc";
source "impd_RO2_root01.dxc";

# read-only, load-share, operations DSAs (optional)
source "impd_tempRO1_root02.dxc";
source "impd_tempRO1_root01.dxc";

# Update DSAs Preference
# Always update the primary multiwrite DSA and replicate to peers

source "impd_root01.dxc";
source "impd_root02.dxc";

#notification DSAs

source "impd_notify_root01.dxc";
source "impd_notify_root02.dxc";

source "impd_temp_root01.dxc";
source "impd_temp_root02.dxc";
```

Note: Installations may deploy only the write DSAs for operations details. In this case, there is no need to introduce read-only DSAs for transient operations detail. Furthermore, for performance reasons, a memory-resident DSA can be introduced for storing operations detail.

Sample DSA Names

The folder in which the eTrust Directory software is installed is usually `\Program Files\CA\eTrust Directory\dxserver`. This folder is referenced as follows:

- `%DXHOME%` for Windows installations
- `$DXHOME` for Solaris (UNIX) installations

There are many different ways to establish global names. A sample configuration that uses the sample global naming convention might include the following:

- The Provisioning Directory DSA name on ROOT01 is `impd_root01`. On a Provisioning Server installation, you can rename this DSA from the default DSA of the provisioning directory bundle, or you can create the DSA manually.
- The read-only (RO) Provisioning Directory DSA name on ROOT02 is `impd_RO_root02`. When you configure multiple, read-only DSAs, the DSA names are numbered, for example, `impd_RO1_root02` and `impd_RO2_root02`.
- The router name in charge of routing the database (Provisioning Directory) traffic on ROOT02 is `impd_router_root02`.
- The DSA for storing transient operations on ROOT01 is named `impd_temp_root01`.
- The local server-tier Router DXlink is named `imps_router`.

Consistent Port Assignments

Each DSA needs the following port assignments:

- LDAP/DAP port
- SNMP port
- Console port
- SSLD port

A high availability deployment potentially involves many DSAs, such as read-only, load-share, multi-write, operations and Provisioning Directory DSAs. It is a good practice to assign port numbers consistently across the entire deployment, as this simplifies the maintenance.

Note: Only one SSLD port is needed for all DSAs, since the SSL daemon listens on a single port. If you choose to configure more than one SSL daemon, you must use a separate port number for each.

Sample Provisioning Configuration

The Provisioning domain has two provisioning directories installed: one for the primary server, the other for an alternate server. During the installation, the provisioning directory is installed followed by the provisioning server on both primary and alternate servers. The provisioning directory installation on the alternate server provides the needed provisioning DSA schema and configuration files. The alternate server is configured to use the primary server's provisioning directory at installation. After the installation, you must manually perform a series of steps to replicate the provisioning directory of the primary server to an alternate server, and to configure multi-write replication for both provisioning directories.

The sample configurations use the provisioning directories on a Windows server.

When setting the system to suit your needs, you must consider the following:

- DSA Configuration
- Port Assignment

A set of sample files is installed in a samples folder where you install the provisioning directory.

DSA Configuration Example

Each provisioning directory installation is configured as follows for provisioning directory tier, high-availability solutions:

- One local provisioning directory router is introduced. A Provisioning domain server communicates with its local Provisioning Directory router, which then routes Provisioning Directory traffic.
- An operations DSA is introduced. (Optionally, read-only DSA can be introduced.)
- A notification DSA is configured for the communication between Provisioning Server and Identity Manager Server.
- One multi-write DSA is designated as the primary update DSA for a domain. All updates are sent to the primary update DSA and replicate to the peers.

- Two read-only provisioning directory DSAs are introduced for search performance.
- Search requests are performed on the local read-only and load-share DSAs, and then on the remote read-only and load-share DSAs.
- A local Router DXlink is configured to communicate with local CA Identity Manager - Provisioning Server service and failover to the remote Provisioning Server.

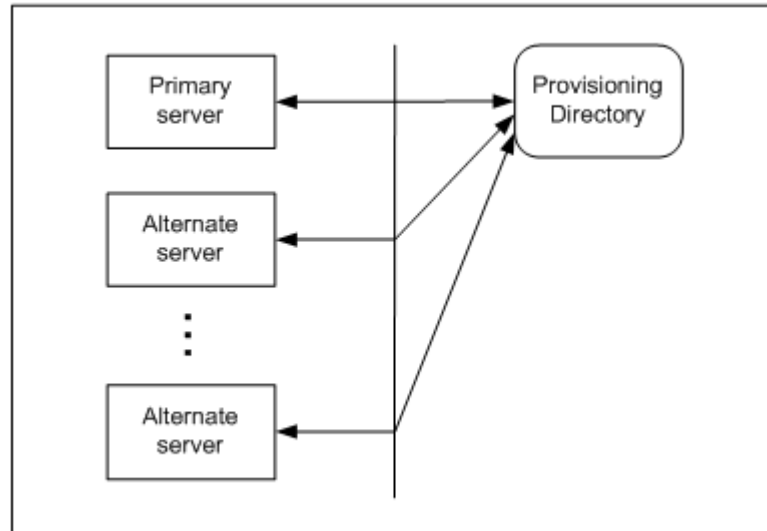
Port Assignment Example

Based on the DSA configuration and global naming conventions, the following table illustrates port assignment examples:

DSA Name	Address	SNMP Port	Console Port	SSLD Port
<i>impd_hostname</i>	20391	20395	20396	20392
<i>impd_temp_hostname</i>	20393	20397	20398	20392
<i>impd_RO1_hostname</i>	22391	22395	22396	20392
<i>impd_RO2_hostname</i>	23391	23395	23396	20392
<i>impd_tempRO1_hostname</i>	22393	22397	22398	20392
<i>impd_router_hostname</i>	21399	21400	21401	20392
<i>imps_router</i>	20399	20400	20401	20392
<i>impd_notify_hostname</i>	23404	23405	23406	20392
<i>imps_hostname</i>	<i>provisioning server port or tls port</i>	-	-	-

Primary and Alternate Provisioning Servers

In a Provisioning domain, you can configure the Provisioning Server as primary or alternate, as shown in this illustration. You can configure multiple alternate servers for one primary server.



In the above illustration, multiple Provisioning Servers are configured to serve the provisioning domain. All servers are configured to use the Provisioning Directory of the primary server installation. The Provisioning Server uses eTrust Directory as its provisioning directory.

You perform a series of manual steps to replicate additional provisioning directories and to configure multi-write replication.

Note: The installation of the provisioning directory on an alternate server installation provides the necessary DSA schema and configuration files which can be used for the replicated provisioning directory on the alternate server.

Provisioning Domain Specifications

A case study of a domain deployment named ROOT on Windows is presented. This deployment includes the following specifications:

- The two Provisioning Servers, primary and alternate, serve the same domain named "root."
- The provisioning directory is installed on hosts, ROOT01 and ROOT02, respectively.
- The primary and alternate servers are installed on hosts, ROOT01 and ROOT02, respectively.

- The CA Identity Manager - Provisioning Server services on Windows are reconfigured to depend on their respective local provisioning directory router services.
- The local Provisioning Directory DSAs and knowledge files are renamed to conform to the global naming convention.
- The CA Identity Manager - Provisioning Server services on Windows communicate with their respective local provisioning directory routers, which route traffic to provisioning directory DSAs, regardless of local or remote repositories.
- The client-tier, server-tier, and directory-tier have high availability solutions that you can configure to suit the needs of your enterprise.

Client-Tier

With the client-tier failover, Provisioning Manager communicates directly with its designated Provisioning Server and fails over to another server, bypassing the Router DXlink, for performance reasons.

Server Tier

The following applies to the server-tier failover:

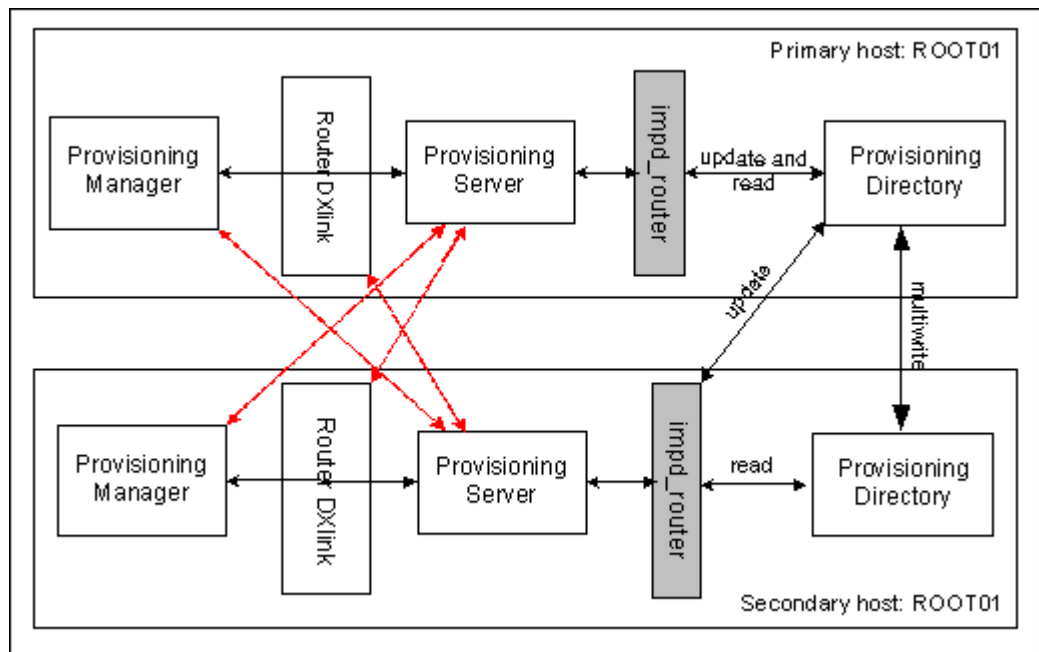
- Router DXlink routes provisioning requests to one Provisioning Server and fails over to another server. (Router DXlink is used for clients, other than Windows clients, such as Provisioning Manager.)
- Provisioning Manager can also communicate with Router DXlink, if needed.

Provisioning Directory Tier

The following applies to the Provisioning Directory-tier high availability:

- Provisioning Server communicates with a local eTrust Directory Router DSA, which routes provisioning directory traffic.
- Provisioning Directories are replicas of each other.
- Multiple DSAs are configured for query-streaming for performance of Provisioning Directory-tier high availability.
- One multiwrite DSA is designated as the primary write DSA. All updates are made to the primary write DSA and replicate to the peer write DSA.
- Provisioning Directory data coherency is maintained using eTrust Directory multiwrite replication technology.

The illustration that follows shows the high availability configuration that is deployed for the ROOT domain.



Solid lines indicate normal control flow, while dashed lines show failover control flow.

ROOT Domain Installation

Two Provisioning Servers on Windows are configured during installation so that they serve the same ROOT domain. The two Windows servers, primary and alternate, are installed on ROOT01 and ROOT02, respectively.

Domain Prerequisites

The prerequisites are as follows:

Primary Domain Server: ROOT01

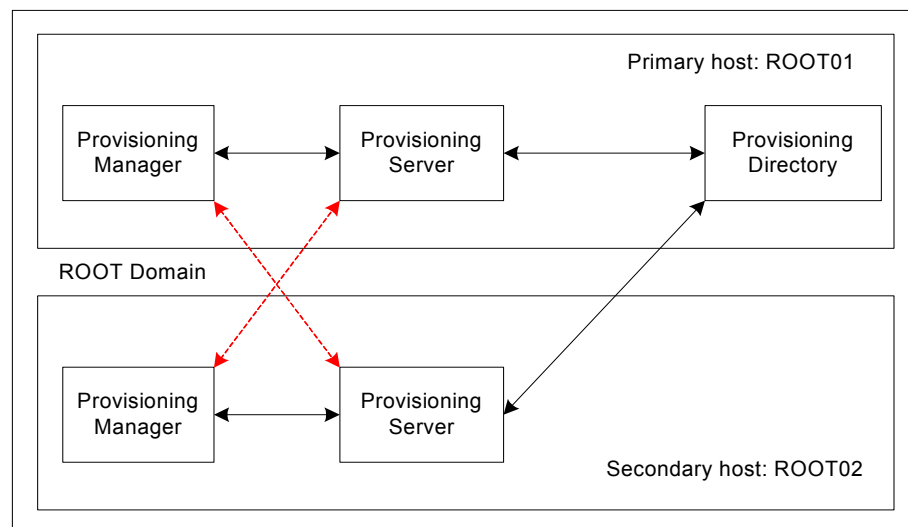
- Install the latest build of eTrust Directory r8.1. eTrust Directory r8.1 requires the Java 2 Runtime Environment v1.4.2_04, which is bundled with the CA Identity Manager r12 installation package.
- Install Provisioning Directory on Windows.
- Install Provisioning Server on Windows.
- Install Provisioning Client on Windows.
- Configure a primary Windows root domain server with the domain name, ROOT.

Secondary (Alternate) Domain Server: ROOT02

- Install the latest build of eTrust Directory r8.1.
- Install Provisioning Directory which provides DSA schema and configuration files for replication purpose.
- Configure an alternate server for the ROOT domain, specifying the use of provisioning directory on ROOT01.
- Install Provisioning Client on Windows.

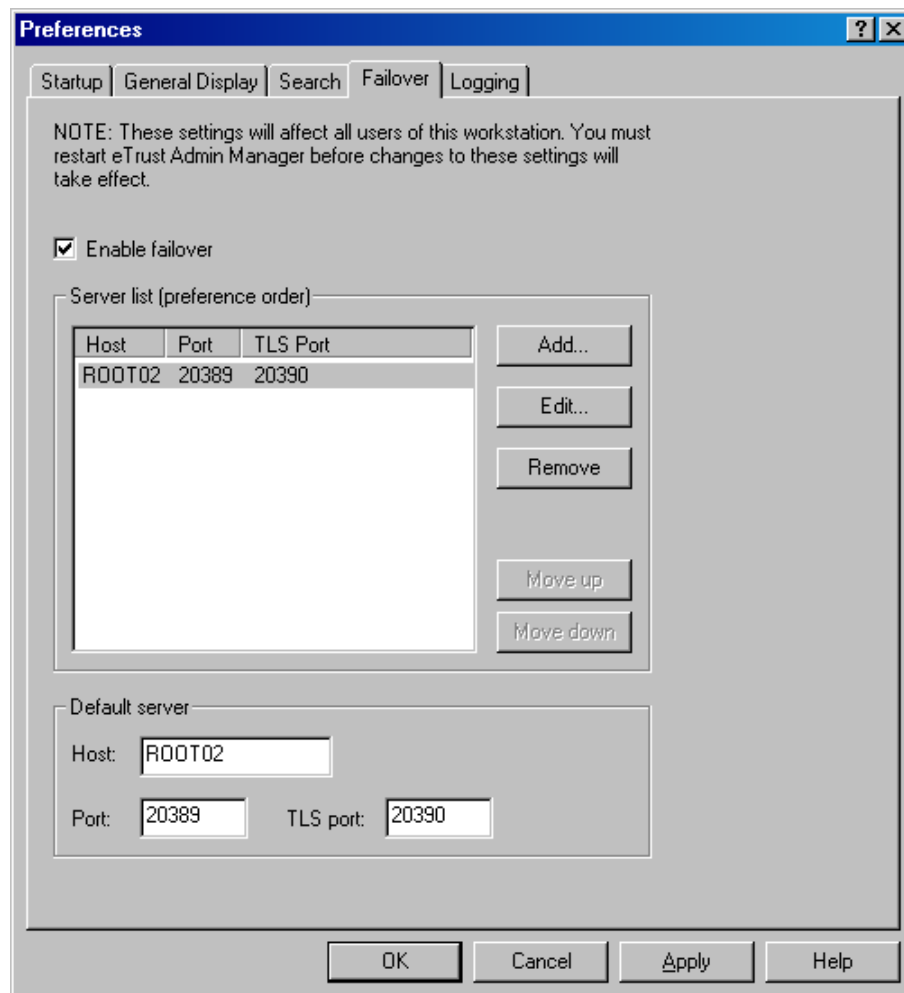
Important! You must install both domain servers with the same connectors.

After you complete this installation, both provisioning Windows servers ROOT01 and ROOT02 use the same provisioning directory located on the primary server, ROOT01 as shown in the following illustration:



Both provisioning servers are installed with failover enabled by default, as shown in the next illustration. This means that the identity provisioning requests from Windows clients will be sent directly to the servers specified on the Failover tab in the preference order.

If failover is not enabled for an alternate server, Windows clients attempt to resolve the domain name into a host name for identity provisioning tasks. Since only the primary domain server is registered, all requests are always sent to the primary domain servers.



The provisioning server always configures failover to direct identity provisioning traffic to the intended alternate server. From Provisioning Manager, select File, Preferences, and the Failover tab. Check Enable failover.

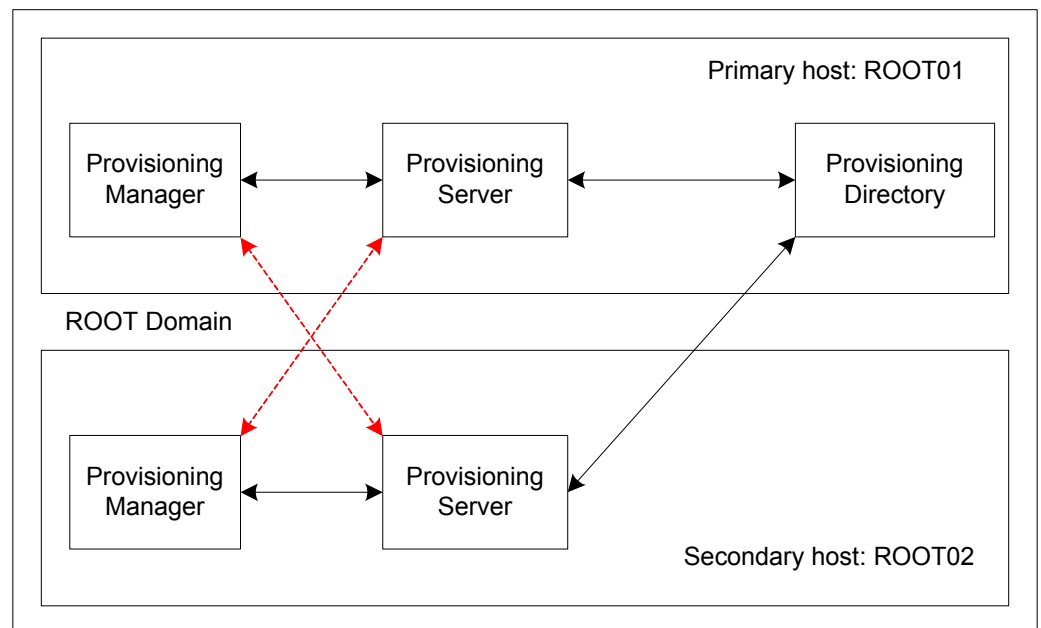
How to Make Provisioning Clients Highly Available

Client-tier configuration includes the following tasks:

- Configure the Windows client-tier failover
- Configure the Provisioning Manager on ROOT01 and ROOT02 to communicate with their local Provisioning Servers, and fail over to the remote Provisioning Server

You use the same Provisioning Manager dialog to accomplish both of these tasks, on each server in turn.

The configuration shown in the following illustration lets Provisioning Manager submit identity provisioning requests to one Provisioning Server and fail over to another server:



The Provisioning Manager sends requests to the default Provisioning Server and fails over to another server.

Enable Client-Tier Failover for Provisioning Manager

You can enable client-tier failover on both the primary and secondary host servers, ROOT01 and ROOT02. When this procedure is complete, you will have configured each server for failover to the other.

To enable the client-tier failover

1. Launch the Provisioning Manager.
2. Select File, Preferences, and select the Failover tab.
3. Mark the Enable Failover check box, if this is not already done. By default, the local Provisioning Server is already defined.
4. Click Add.
5. Enter the host name of the remote Provisioning Server. On ROOT01, enter the server host for ROOT02. On ROOT02, enter the server host for ROOT01.
6. Enter 20389 for the LDAP port value and 20390 for the LDAP/TLS port value, respectively.

Additionally, you can enter a Router DXlink as a failover server. In this case, enter 20399 for both LDAP and LDAP/TLS ports. You must modify the LDAP and LDAP/TLS ports for Router DXlink if they are changed for a Provisioning Server installation.

7. Adjust the preference order by moving the entries up and down in the list.
8. Click OK.
9. Restart the Provisioning Manager to enable your changes.

Important! The client-tier failover applies to the Provisioning Manager and etautil. Other provisioning clients, such as the LDAP search, should be able to submit requests to the Router DXlink and allow the router to handle failover.

Test the Client-Tier Failover Configuration

You can test your client failover configuration by performing the following procedure:

To test the client failover configuration

1. Stop the CA Identity Manager - Provisioning Server service on one domain server.
2. Issue one or more operations using Provisioning Manager for this server installation.

Since you stopped the CA Identity Manager - Provisioning Server service locally, the traffic flows to the failover domain server. If it does not, check your configuration and try the test again.

How to Make Provisioning Servers Highly Available

Multiple provisioning servers share the workload of a provisioning domain, providing performance, scalability, and high availability. The first provisioning server installed is called the primary server. Additional servers are called alternate servers.

Two methods exist for making Provisioning Servers highly available. If you have CA SiteMinder, you can use it, since it is the recommended method. Otherwise, you can use DXlink routers.

Use SiteMinder for High Availability

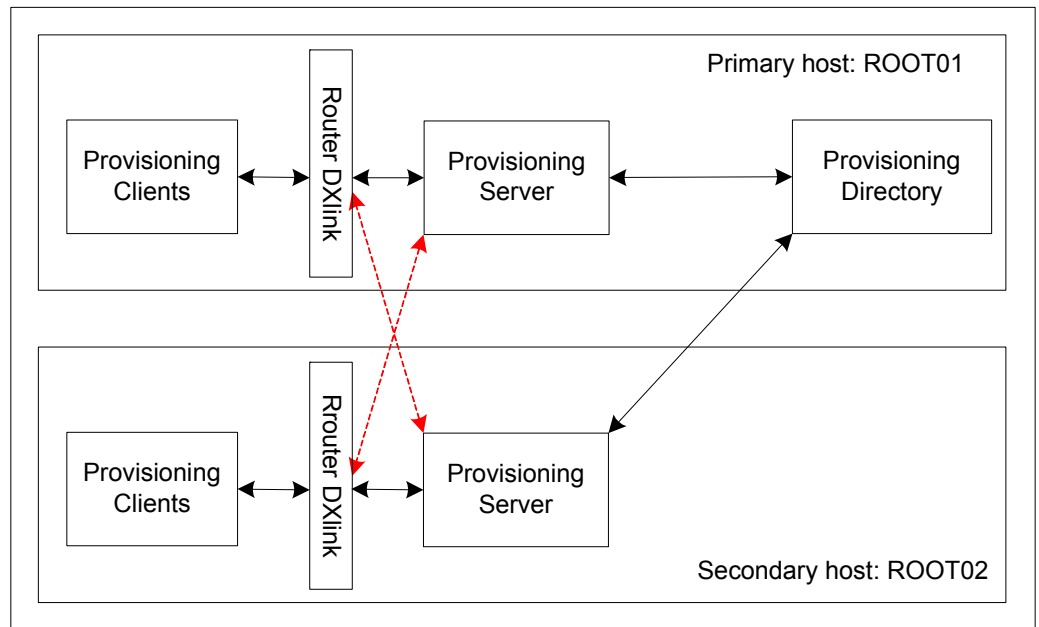
If you have CA SiteMinder Web Access Manager, you can create multiple server definitions in JIAM to communicate with the Provisioning Server for provisioning operations. The JIAM API supplied with Identity Manager supports native failover in its communication layer. You create these definitions in the smuserdirectory object associated with the Identity Manager directory for your environment.

During initialization, Identity Manager obtains the connection information from the Identity Manager directory's smusedirectory object. Identity Manager reads any failover server definitions defined in that object, adding them to the JIAM failover server definitions. Also during the Policy Server startup, it reads these server definitions, adding them to its failover server definitions. So regardless of the interface, they both support failover to the same servers.

Note: For details on setting up server definitions, see the CA SiteMinder Web Access Manager *Configuration Guide*.

How to Use DXLink Routers for High Availability

If you use DXlink routers for server failover, the Provisioning clients submit requests to the router and let the router direct the Provisioning Server traffic to the servers behind the router, as shown in the following illustration:



Clients submit identity provisioning requests to Router DXlink which sends them to the local Provisioning Server and fails over to the remote server.

The following conventions are used in the configuration of server-tier high-availability solutions:

- A Provisioning Server is named `imps_hostname`, where *hostname* is the name of the host where Provisioning Server resides.
- The knowledge file for a Provisioning Server is named `imps_hostname.dxc`.
- The DSA name of a Provisioning Server is `imps_hostname`.
- The dsa-name of the Provisioning Server DXlink knowledge file is `dsa-name = <dc eta><cn imps_hostname>`.
- The prefix of a Provisioning Server is always `<dc eta>`.
- The local Provisioning Router DXlink is named `imps_router`.

The local Router DXlink knowledge file is named `imps_router.dxc`.

Configure Local and Remote Failover Servers

Configuring the local and remote failover servers involves the following steps:

- Understanding knowledge file relationships
- Creating the local router DSA knowledge file
`imps_router.dxc`
- Creating the local server DSA knowledge file
`imps_localhostname.dxc`
- Creating the remote failover server knowledge file
`imps_remotehostname`
- Configuring the server-tier router DSA group file, `imps_router.dxc`
- Configuring the server-tier router DSA, `imps_router.dxi`

Knowledge File Relationships

The `imps_router.dxc` file includes the knowledge of the router local provisioning server, and remote failover server.

Note: Because of its limitations, server-tier load-sharing is not recommended.

The `imps_router.dxc` knowledge file defines the Provisioning Router DXlink DSA. The `imps_<hostname>.dxc` knowledge file defines the Provisioning Server DSA on hostname.

The knowledge file for the Router DXlink is named `imps_router.dxc`. Each Provisioning Router DXlink listens on port 20399, for both LDAP and LDAP/TLS connections. TLS/SSL connections are forwarded to the Provisioning SSL daemon listening on port 20392. You must remove the localhost address if this knowledge file is to be sourced (referenced) by other installations.

The Provisioning router to server communication is facilitated by the eTrust Directory DXlink technology that enables X.500 to LDAP communication. This is set using the statement `link-flags = dsp-ldap` in the `imps_<hostname>.dxc`, e.g., `imps_root01.dxc` or `imps_root02.dxc` file.

The communication between two servers in a local system does not have to be encrypted. However, the communication between two systems across a network must be encrypted. The `link-flags = ssl-encryption` parameter specifies explicitly that a communication link must be encrypted.

Note: When two DSAs are on the same system, communication can be done via a localhost connection.

Create Local Server DSA Knowledge File

Execute the following procedure to create a local server DSA knowledge file.

To create the knowledge file for the local server DSA

1. Copy the `imps_<localhostname>.dxc` from the samples high-availability folder to the eTrust Directory knowledge folder.
2. Edit the `imps_localhostname.dxc` file using a text editor.
3. Make sure that the set `dsa` line becomes `set dsa imps_localhostname`.
4. Make sure `dsa-name` is `imps_localhostname`.
5. Make sure that the address line uses `localhost` and port number `20389`. Since the local Provisioning Server is on the same host as the Router DXlink, there is no need to encrypt the communication traffic.
6. Make sure that the `link-flags` defines `dsp-ldap`, for DXlink X500 to LDAP protocol switch.

Note: In this procedure, replace *localhostname* with either `ROOT01` or `ROOT02` depending on the Provisioning domain server that you are configuring. For example, you must specify `ROOT01` as the local host name when you are configuring a Provisioning Server on `ROOT01`, and conversely.

Configure Server-Tier Router DSA Group File

Execute the following procedure to create a server-tier router DSA group file.

To create the DSA group file for the server-tier router

1. Copy and edit the `%DXHOME%\config\knowledge\imps_router.dxc` file in samples folder
2. Make sure that router group knowledge file sources the local router knowledge file, `imps_router.dxc` and the two provisioning server DXlink knowledge files, `imps_localhostname.dxc` and `imps_remotehostname.dxc`.
3. Verify that the `imps_localhostname.dxc` file is sourced before *imps_remotehostname.dxc*.

Note: In this procedure, replace *localhostname* and *remotehostname* with either `ROOT01` or `ROOT02` depending on the Provisioning domain server that you are configuring. For example, you must specify `ROOT01` as the remote host name when you are configuring a Provisioning Server on `ROOT02`, and conversely.

Create Remote Failover Server Knowledge File

Execute the following procedure to create a remote failover server knowledge file.

To create the knowledge file for the remote failover server

1. Copy `imps_remotehostname.dxc` in `samples\high-availability\root\localhostname\DXHOME\config\knowledge` to the deployment eTrust Directory knowledge folder.
2. Edit the `imps_remotehostname.dxc` file using an ASCII text editor, such as Microsoft Notepad.
3. Make sure that the set `dsa` becomes `set dsa imps_<remotehostname>`.
4. Make sure that the `dsa-name` is set to `imps_<remotehostname>`.
5. Replace `localhost` in the address line with `remotehostname`, if needed
Since a failover server is on a remote host, the port number specified must be the TLS/SSL port number that the remote Provisioning Server listens on. The default value is 20390.
6. Set the `link-flags` to include both `DXlink` and encryption:

```
link-flags =dsp-ldap, ssl-encryption-remote
```


The parameter `ssl-encryption-remote` encrypts the communication channel as the traffic is sent across the network.
7. Make sure that the `auth-levels` value is set as `auth-levels = clear-password`. You may need to remove `ssl-auth` if this is specified in the knowledge file.

Note: In this procedure, replace *remotehostname* with either `ROOT01` or `ROOT02` depending on the Provisioning domain server that you are configuring. For example, you must specify `ROOT01` as the remote host name when you are configuring a Provisioning Server on `ROOT02`, and conversely.

When you complete this procedure on each domain server, each server of `ROOT` is configured to be the failover of the others.

Note: To observe router traffic, you can configure the `DXconsole` from the eTrust Directory bundle to listen on the console port defined in the router knowledge file. For example, the default Provisioning router uses port 20401. For more information, see *eTrust Directory Administrator Guide*.

Configure Server-Tier Router DSA

Execute the following procedure to create a server-tier router DSA

To create the router DSA DXI file for the server-tier router

1. Copy and edit the `imps_router.dxi` file from the samples high-availability `config\servers` folder to the eTrust Directory `config\servers` folder.
2. Make sure that `imps_router.dxi` file sources the aforementioned `imps_router.dxi` file in the knowledge folder, which includes the DSA knowledge of the local router, local provisioning server, and remote provisioning server.
3. Make sure that `imps_router.dxi` file does not include the use of any database.
4. Start up the `imps_router` using "`dxserver start imps_router`" command.

Test the Server Failover

You can test the configuration by performing the following steps from either ROOT01 or ROOT02.

To test the server failover configuration

1. Using Provisioning Manager, select File, Preferences, Failover tab.
2. Make a note of the current settings.
3. Enable failover and configure only one local Router DXlink.
Note: You must specify 20399 as both the LDAP and LDAP/TLS default port numbers or modify these values if you have changed your configuration.
4. Stop the CA Identity Manager - Provisioning Server service on the local system.
5. Ensure that the remote CA Identity Manager - Provisioning Server service is running.
6. Open a new Provisioning Manager window.
7. Connect to the ROOT domain.

If the connection is successful, the Provisioning Manager is connected to the remote failover server through the local Router DXlink.

Note: To observe router traffic, you can configure the DXconsole from the eTrust Directory bundle to listen on the console port defined in the router knowledge file. For example, the default Provisioning router uses port 20401. For more information, see *eTrust Directory Administrator Guide*.

Important! After this test, reset the failover server preference list back to its previous setting, and restart the CA Identity Manager - Provisioning Server service.

Provisioning Server Disaster Recovery

The difference between the primary and alternate servers is that the primary server installation populates the provisioning directory container entries, default provisioning directory entries, and bootstrapping provisioning administrator entries. Also, uninstalling a primary server removes those entries.

Apart from installation and uninstallation, the primary and alternate servers function in the same manner. Also, you can convert a primary server to an alternate server or an alternate server to a primary server by using the Identity Manager Provisioning Server installation.

For provisioning server disaster recovery, we recommend that a primary server is re-configured to be an alternate server after installation. As a result, uninstalling a primary server does not remove Provisioning Directory entries that other alternate servers depend on.

You can recover a corrupted provisioning server by installing an alternate server. If you need to remove the primary server that is reconfigured as an alternate server, the server can be reconfigured as a primary server before it is uninstalled.

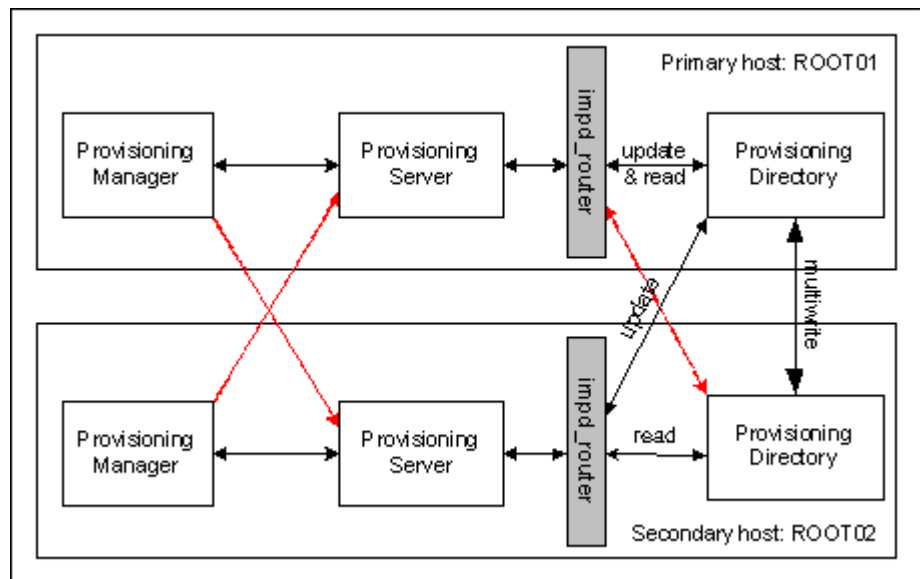
How to Make Provisioning Directories Highly Available

The following major tasks are needed to configure Provisioning Directory-tier high-availability solutions:

Task	Refer to...
Replicate the Provisioning Directory on the primary server to the alternate server installation	Replicate the ROOT Domain Provisioning Directory (see page 103)
Reconfigure Provisioning Directory-tier high-availability DSAs on the primary server installation on ROOT01	How to Reconfigure DSAs on the Primary Domain Server (see page 106)

Reconfigure Provisioning Directory-tier high-availabilityDSAs on the alternate server installation on ROOT02	Configuring the Alternate Domain Server Provisioning Directory on ROOT02 (see page 114)
Share the DSA knowledge files between ROOT01 and ROOT02	Import Files from the Primary Domain (see page 116)
Configure the DSA certificates and personality files for TLS/SSL communication	Provisioning Directory Security Settings (see page 119)
Configure the Provisioning Directory-tier replication	Import Files from the Primary Domain (see page 116)
Configure Provisioning Servers to communicate with their respective Provisioning Directory routers	Configure CA Identity Manager - Provisioning Server service to Direct Provisioning Directory Requests to Local Routers (see page 113)

The targeted Provisioning Directory-tier high availability configuration is shown in the following illustration:



The Provisioning Directory-tier high availability solution uses the following conventions:

- Global naming convention for DSA names and knowledge files.
- A local Provisioning Directory-tier router DSA is introduced for a Provisioning Server. A Provisioning Server communicates with the local router DSA which routes Provisioning Directory traffic.
- All Provisioning Directory X.500 DSAs have knowledge of each other.
- Provisioning Directory update DSAs are configured to multiwrite to each other.
- Each Provisioning Directory update DSA is named `impd_hostname`, where *hostname* is the name of the host where the Provisioning Directory is installed.
- Each operation DSA is named `impd_temp_hostname`, where *hostname* is the name of the host where an operations DSA is installed.
- A Provisioning Directory X.500 router DSA is named `impd_router_hostname`, where *hostname* is the name of the host where a Provisioning Server is installed.
- The local Router DXlink keeps the name `imps_router`.
- The address line in the X.500 DSA knowledge file does not include `localhost`. This is important because the same set of knowledge files is sourced by multiple DSAs.

Replicate the ROOT Domain Provisioning Directory

To configure high availability for the provisioning directory tier, you must first replicate the existing provisioning directory on ROOT01 to ROOT02.

To replicate the ROOT domain provisioning directory

1. Dump the provisioning directory contents in ROOT01 using the eTrust Directory bulk-load tool, `dxdumpdb`.
2. Load the content to the provisioning directory created on ROOT02 during the installation of the alternate domain server, using the bulk-load tool, `dxloaddb`.

Dump the Primary Provisioning Directory Contents

The following procedure can be used to dump the provisioning directory contents.

To dump the contents of the primary server's provisioning directory

1. Start a Command Prompt on ROOT01.
2. Stop the eTrust Directory - etrustadmin service using the command:

```
net stop dxserver_etrustadmin
```

Answer Y to continue the operation. In this case, CA Identity Manager - Provisioning Server service is also stopped, based on dependencies between the two processes.

When the provisioning server and provisioning directory are installed on the same host, a Windows service dependency is created. The "net stop dxserver_etrustadmin" command ensures that provisioning service is stopped before the provisioning directory service is stopped.

Note: The default database/DSA name for the provisioning directory is etrustadmin. Adjust the database/DSA name according to your requirements.

3. Dump the provisioning directory contents using the command:

```
dxdumpdb -S etrustadmin -p "dc=etadb" -f impd.ldf etrustadmin
```

4. Sort the generated LDAP Data Interchange Format (LDIF) file using the command:

```
ldifsort impd.ldf impd_sorted.ldf
```

5. Remove the etrustadmin.ldf file.
6. Restart the services that you previously stopped using the command:

```
net start im_ps
```

Dependencies between the processes cause the eTrust Directory services to restart as well.

Load the Replicated Provisioning Directory Data

The installation of the provisioning directory on ROOT02 provides the necessary DSA configuration to load the replicated data.

To load the data into the ROOT02 provisioning directory

1. Transfer the impd_sorted.ldf file from ROOT01 to ROOT02. You may want to specify the %TEMP% directory.
2. Remove the impd_sorted.ldf file on ROOT01.
3. Start a Command Prompt on ROOT02.

4. Change directories with the command:

```
cd %TEMP%
```

5. Stop the eTrust Directory - etrustadmin service:

```
net stop dxserver_etrustadmin
```

Answer Y to continue the operation. Process dependencies cause the CA Identity Manager - Provisioning Server service to stop as well.

If the provisioning directory is on a remote system, you can stop your etrustadmin DSA by running the following command:

```
dxserver stop etrustadmin
```

Adjust the DSA name, if required.

6. Load the data in the provisioning directory using the command:

```
dxloaddb -S etrustadmin -p "dc=etadb" impd_sorted.ldf etrustadmin
```

Note: The -I option with the list of nonSearchable attributes prevents the database SEARCH table from being populated with unwanted attributes from the beginning. The list of nonSearchable attributes is stored in the %DXHOME%\config\database\etrustadmin.dxc file.

7. Remove impd_sorted.ldf.

Create DSA Indexes

The Provisioning Server DSA's indexes (wide-, reverse-, and notSearchable) are specified in the %DXHOME%\config\database\etrustadmin.dxc file. The indexes should match those created in the database. When the provisioning directory is copied using the dxloaddb command, indexes are not created in the database. You must execute the dxindexdb command to create indexes in the database, matching those specified in the database configuration. You can safely ignore the warning messages generated in the process.

If you used the -I option, specifying the notSearchable attributes when you ran dxloaddb (covered in [Load the Replicated Provisioning Directory Data](#) (see page 104)), you only need to run dxindexdb for wide-indexes and reverse-indexes.

Note: CA Identity Manager Provisioning Directory installation provides a sample script file, %<Provisioning Directory Home Folder>%\index\impdindexdatabase.bat, which you can use as a reference for creating the indexes for provisioning directory DSAs. You can run the script if you have not introduced additional attributes that need further indexing.

To create DSA indexes

1. Create the reverse and wide indexes using this command. Adjust attribute names according to the aforementioned database configuration file:

```
dxindexdb etrustadmin +wide eTExitReference eTUserAdminProfile eTwfManager eTwfBusinessApprover  
eTwfTechnicalApprover eTwfDelegate eTRoleDN eTSuperiorClassEntry eTSubordinateClassEntry  
eTCorrelationPolicyDN eTDefaultPolicyDN eTPolicyDN eTAccountDirectory eTNDSTGroupMember  
+reverse objectClass
```

2. If you did not run dxloaddb with the -I option, create the notSearchable index using this command. Adjust attribute names according to the aforementioned database configuration file:

```
dxindexdb etrustadmin -notSearchable eTConfigParamDescription eTConfigParamProperties  
eTConfigParamValue eTConfigParamOrigin eTUPOExitPayload eTUPOExitCustomData
```

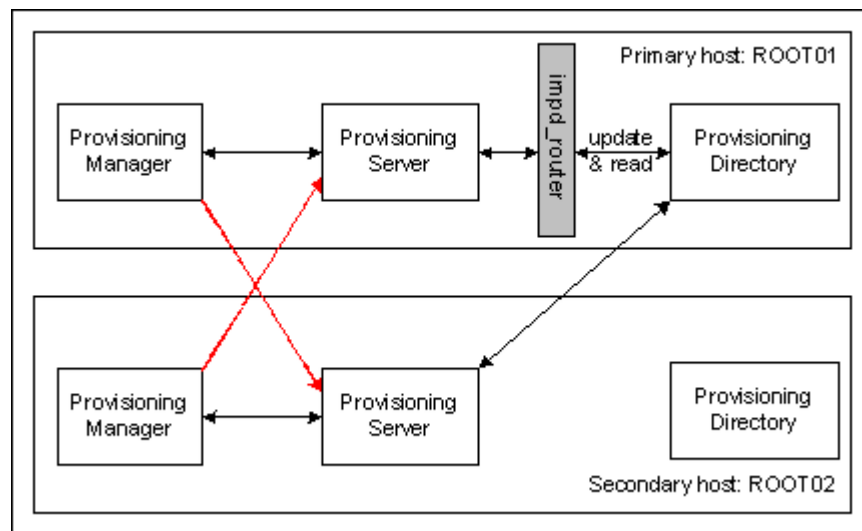
3. Restart the services that were previously stopped:

```
net start im_ps
```

If the provisioning directory is on a remote system, you can run the dxserver start etrustadmin command to start your provisioning directory server. Adjust the DSA name, if required.

How to Reconfigure DSAs on the Primary Domain Server

Once a provisioning directory is replicated, the next step is to reconfigure DSAs on both the primary and alternate server installations.



The primary Provisioning Server is configured to communicate with a local provisioning directory router DSA which routes provisioning directory traffic. The provisioning directory on the alternate installation has a replicated copy of data (Intermediate stage).

The process is as follows:

1. Configure a local provisioning directory router, which is aware of all the DSAs on both the primary and alternate directory installation.
2. Redirect the Provisioning Server to communicate with the local provisioning directory router DSAs.

Note: A Provisioning Server uses a password to authenticate to the provisioning directory. The password is locked down to the host/port/tls-port of the default provisioning directory. The provisioning directory password must be reconfigured and locked down to the provisioning directory router before the provisioning server can be redirected to communicate with the provisioning directory router. You can use the Password Manager, `pwdmgr.exe`, for this task. The password lock-down needs to be done with a working provisioning server. In addition, the new DSA personality files must be configured before the provisioning server can communicate with the provisioning directory via TLS/SSL.

How to Introduce and Configure Local X.500 Provisioning Directory and Router DSAs

The following steps introduce and configure local X.500 provisioning directory and router DSAs.

Note: Before you reconfigure provisioning directory DSAs, you must use Password Manager to lock down the provisioning directory password to the provisioning directory router, 21399 for both LDAP and LDAP/TLS ports, in the sample configuration, that the provisioning server will eventually communicate with. This step must be done before any reconfiguration, as Password Manager needs to communicate with provisioning server in order to lock down password. If this step is not done first, the provisioning server will not be able to authenticate to the provisioning directory router. Furthermore, Password Manager will not be able to authenticate to the provisioning server to lock down the password to the provisioning directory router.

To introduce and configure local X.500 provisioning directory and router DSAs

1. Introduce provisioning directory router DSAs on ROOT01 and ROOT02.
2. Create and organize the knowledge files of all the DSAs in both ROOT01 and ROOT02.
3. Configure the provisioning directory and router DSAs with the knowledge of all the DSAs on ROOT01 and ROOT02.

4. Rename and reconfigure provisioning directory DSAs.
5. Reconfigure CA Identity Manager - Provisioning Service dependency on Windows .

This enables the Provisioning Server to communicate with the local provisioning directory router DSA, which routes provisioning directory traffic.

Create and Organize the DSA Knowledge Files on ROOT01 and ROOT02

The samples high-availability root domain folder contains the following DSA knowledge files.

Copy the following DSA knowledge files from the sample high-availability root domain DXHOME\config\knowledge folder to the eTrust Directory deployment knowledge folders on both ROOT01 and ROOT02.

DSA knowledge files for ROOT01

- impd_root01.dxc – primary multi-write DSA on ROOT01
- impd_RO1_root01.dxc – first read-only, load-share DSA on ROOT01
- impd_RO2_root01.dxc – second read-only, load-share DSA on ROOT01
- impd_temp_root01.dxc – operation details DSA on ROOT01
- impd_tempRO1_root01.dxc – (optional) operation details, read-only load-share DSA
- impd_notify_root01.dxc – the notification DSA on ROOT01
- impd_router_root01.dxc – the provisioning directory router on ROOT01

DSA knowledge files for ROOT02

- impd_root02.dxc – primary multi-write DSA on ROOT02
- impd_RO1_root02.dxc – first read-only, load-share DSA on ROOT02
- impd_RO2_root02.dxc – second read-only, load-share DSA on ROOT02
- impd_temp_root02.dxc – operation details DSA on ROOT02
- impd_tempRO1_root02.dxc – (optional) operation details, read-only load-share DSA
- impd_notify_root02.dxc – the notification DSA on ROOT02
- impd_router_root02.dxc – the provisioning directory router on ROOT02

Group knowledge files

- impd.dxc – group knowledge file that contains all the provisioning directory DSA knowledge files
- impd_router.dxc – group knowledge file that contains all the provisioning directory router knowledge files

Create the Provisioning Directory and Router DSAs on ROOT01 and ROOT02

The samples high-availability root domain DXHOME\config\servers folder contains the following DSA DXI files.

Copy the following ROOT01 and ROOT02 DXI files to their respective %DXHOME%\config\servers folders on ROOT01 and ROOT02.

DXI files for the ROOT01 provisioning directory DSAs

impd_root01.dxi – primary multi-write DSA on ROOT01
impd_RO1_root01.dxi – first read-only, load-share DSA on ROOT01
impd_RO2_root01.dxi – second read-only, load-share DSA on ROOT01
impd_temp_root01.dxi – operation details DSA on ROOT01
impd_tempRO1_root01.dxi – (optional) read-only operations DSA on ROOT01
impd_notify_root01.dxi – the notification DSA on ROOT01
impd_router_root01.dxi – provisioning directory router DSA on ROOT01

DXI files for the ROOT02 provisioning directory DSAs

impd_root02.dxi – primary multi-write DSA on ROOT02
impd_RO1_root02.dxi – first read-only, load-share DSA on ROOT02
impd_RO2_root02.dxi – second read-only, load-share DSA on ROOT02
impd_temp_root02.dxi – operation details DSA on ROOT02
impd_tempRO1_root02.dxi – (optional) read-only operations DSA on ROOT02
impd_notify_root02.dxi – the notification DSA on ROOT02
impd_router_root02.dxi – provisioning directory router DSA on ROOT02

Configure the Provisioning Directory and Router DSAs with the Knowledge of All DSAs on ROOT01 and ROOT02

Follow this procedure to configure the provisioning directory and router DSAs to have the knowledge of all the DSAs on both ROOT01 and ROOT02.

To share the DSA knowledge globally across both ROOT01 and ROOT02

1. Make sure that all the DXI files in %DXHOME%\config\servers source the following two lines: impd.dxi includes all the provisioning directory DSA knowledge files; impd_router.dxi includes the two provisioning directory router knowledge files.

```
source impd.dxi;  
source impd_router.dxi;
```

2. Make sure that the `impd_router.dxc` file include the following two source lines:

```
source impd_router_root01.dxc;  
source impd_router_root02.dxc;
```

3. Make sure that the `impd.dxc` file include the following DSA knowledge files. Adjust order if needed:

```
# read-only, load-share, provisioning directory DSAs  
source impd_RO1_root01.dxc;  
source impd_RO2_root01.dxc;  
source impd_RO1_root02.dxc;  
source impd_RO2_root02.dxc;
```

```
# read-only, load-share, operations DSAs (optional)  
source impd_tempRO1_root01.dxc;  
source impd_tempRO1_root02.dxc;
```

```
# multi-write provisioning directory DSAs  
source impd_root01.dxc;  
source impd_root02.dxc;
```

```
# multi-write notification DSAs  
source impd_notify_root01.dxc;  
source impd_notify_root02.dxc;
```

```
# operations DSA  
source impd_temp_root01.dxc;  
source impd_temp_root02.dxc;
```

How to Rename and Reconfigure Provisioning Directory DSAs

By default, a Provisioning Directory installation introduces three provisioning directory DSAs, *etrustadmin*, *etadmintemp*, and *etaops_notify*. The first is the main provisioning directory DSA for a Provisioning domain installation, while the second is a provisioning directory for transient operation details. The third DSA is for notification purpose between the Provisioning Server and Identity Manager Server.

If you adopt a global naming convention, you must modify the DSA names and services. The tasks to rename and reconfigure DSAs include the following steps:

1. Remove default provisioning directory services
2. Install the new services for provisioning directory and router DSAs
3. Restart services and test DSAs

Remove Default Provisioning Directory Services on ROOT01

The following procedure enables you to remove the provisioning directory services on ROOT01.

To remove the provisioning directory services on the ROOT01 Provisioning Server machine

1. Stop the eTrust Directory - etrustadmin service.
2. Stop the eTrust Directory - etadmintemp service.
3. Stop the eTrust Directory - etaops_notify service.
4. Remove these three provisioning directory services using the commands:

```
dxserver remove etrustadmin
dxserver remove etadmintemp
dxserver remove etaops_notify
```

5. Remove the following three provisioning directory DSA initialization files from %DXHOME%\config\servers folder:

```
etrustadmin.dxi
etadmintemp.dxi
etaops_notify.dxi
```

Re-install Services on ROOT01

Execute the following procedure to re-install services on ROOT01.

To reinstall services on ROOT01

1. Provisioning Directory DSA services using the commands:

```
dxserver install impd_root01
dxserver install impd_RO1_root01
dxserver install impd_RO2_root01
dxserver install impd_temp_root01
dxserver install impd_tempRO1_root01 (optional)
dxserver install impd_notify_root01
dxserver install impd_router_root01
```

2. Start the new DSAs to verify the configuration steps using the commands:

```
dxserver start impd_root01
dxserver start impd_RO1_root01
dxserver start impd_RO2_root01
dxserver start impd_temp_root01
dxserver start impd_tempRO1_root01 (optional)
dxserver start impd_notify_root01
dxserver start impd_router_root01
```

Generate Personality Files for All Provisioning Directory DSAs

Use the eTrust Directory `dxcertgen` utility to generate the personality files for all the provisioning Directory DSAs. Refer to "Provisioning Directory Security Settings" for further detail.

The provisioning server will not be able to communicate with provisioning directory router using TLS/SSL, unless the DSAs are configured with proper DSA personality files.

How to Reconfigure CA Identity Manager - Provisioning Server service Dependencies on Windows

When a provisioning directory is installed on the same system as the Provisioning Server for Windows installation, the CA Identity Manager - Provisioning Server service depends on the following Windows services by default:

- eTrust Directory - `etrustadmin`
- eTrust Directory - `etadmintemp`
- eTrust Directory - `etaops_notify`

In a high availability configuration, you must configure the CA Identity Manager - Provisioning Server service to not depend on the following services, `etrustadmin`, `etadmintemp`, and `etaops_notify`, because those services have been renamed.

The following steps are required in order to modify the CA Identity Manager - Provisioning Server service dependencies:

- Removing the unnecessary dependencies.
- Adding the required dependencies.

You can configure the Identity Manager - Provisioning Server service to depend on the local provisioning directory router service, or you can further adjust the Windows service dependency based on your enterprise needs.

Remove the Existing Windows Service Dependencies

To remove the existing dependencies, enter the following command:

```
imps remove im_ps
```

The CA Identity Manager - Provisioning Server service is removed.

Note: When the above command is run, the SLAPD `im_ps` windows service is removed along with all the service dependencies. Running the `im_ps install ...` command will establish only the needed Windows service dependencies.

Add the Provisioning Server Dependency

This procedure adds the CA Identity Manager - Provisioning Server service dependency on the local provisioning directory router service.

Add the local provisioning directory router dependencies using the command:

```
im_ps install im_ps "CA Identity Manager Provisioning Service" auto "DXServer_impd_router_root01"
```

Note: All DSAs and routers need to be configured to use the port number 20392 for their SSL communication. This is the default SSL port on which the Provisioning Directory SSLD daemon listens. You can modify the port number based on your enterprise needs.

Configure the Provisioning Server service to Direct Provisioning Directory Requests to Local Routers

Note: Before you reconfigure the Provisioning Server service to direct provisioning directory traffic to the provisioning directory router, the password that the provisioning server uses to authenticate to the provisioning directory must have been locked down to the provisioning directory router. You need to use Password Manager for this purpose. This needs to be done with an operational provisioning server that can authenticate to the provisioning directory.

On Windows, the CA Identity Manager - Provisioning Server service uses the *PSHOME\data\im_ps.conf* file to configure communications between the provisioning server and the provisioning directory. The entries in this file identify the provisioning directory location by host name and port assignments. The configuration example used in this guide specifies a local provisioning directory. If you are using a remote provisioning directory, you must specify where that provisioning directory resides, in this file.

Update the *im_ps.conf* file so that the last lines resemble the following:

```
DbHost      root01
DbPort      21399
DbTlsPort 21399
```

The CA Identity Manager - Provisioning Server service will now use the local provisioning directory router to route provisioning directory traffic.

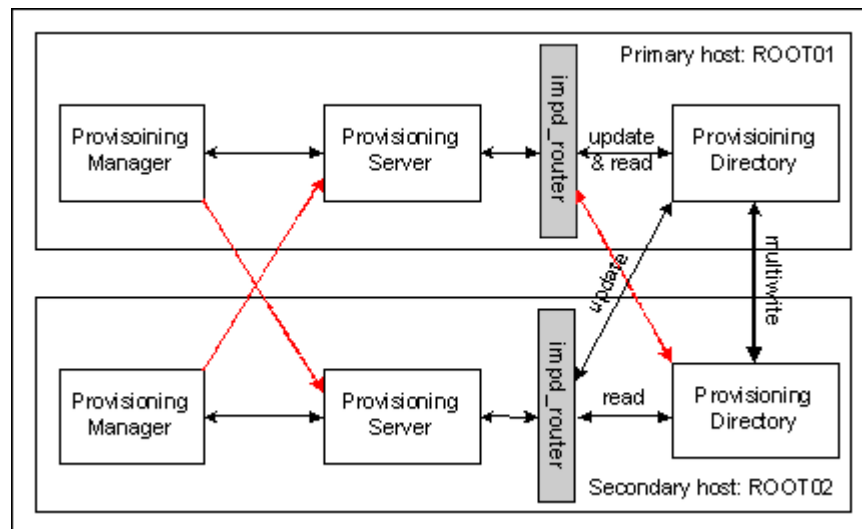
Note: The port number 21399 corresponds to the LDAP port number previously defined in the *impd_router_root01.dxc* file and the *impd_router_root02.dxc* file. If you are using a remote Provisioning Directory, be sure that the value that you use here matches the values in these two configuration files.

Configure the Alternate Domain Server Provisioning Directory on ROOT02

To reach high availability solutions on alternate server, you must configure repository DSAs on the alternate server on ROOT02.

Since a Provisioning Directory installation introduces etrustadmin, etadmintemp, and etaops_notify DSAs, additional configuration work is needed to reconfigure DSAs.

Before you reconfigure alternate server installation, you need to use Password Manager, PwdMgr.exe, to lock down the provisioning directory password to the host/port/tls-port number of the provisioning directory router to which the provisioning server will eventually communicate with. Password Manager needs to be run against an operational provisioning server, preferably before any provisioning server/directory reconfiguration.



Important! For data integrity reasons, it is important that updates to provisioning directories are always performed on the primary Provisioning Directory and then replicated to the secondary Provisioning Directory using the eTrust Directory multiwrite feature.

To do that, knowledge files of the primary Provisioning Directory must be sourced before the secondary one for DSAs defined with the same prefix as follows:

```
source "impd_primaryhostname.dxc"; - Reach the primary first
source "impd_secondaryhostname.dxc"; - Failover to the secondary
source "impd_temp_primaryhostname.dxc"; - Reach the primary first
source "impd_temp_secondaryhostname.dxc"; - Failover to the secondary
```

Stop Directory Services

Use the following procedure to stop the directory services.

To stop the directory services

1. Stop the eTrust Directory - etrustadmin service.
2. Stop the eTrust Directory - etadmintemp service.
3. Stop the eTrust Directory - etaops_notify service.

Remove Existing Windows Service Dependencies

To remove the existing dependencies, enter the following command:

```
imps remove im_ps
```

The CA Identity Manager - Provisioning Server service is removed.

When the above command is run, the SLAPD im_ps windows service is removed along with all the service dependencies.

Running the *im_ps install ...* command establishes only the needed Windows service dependencies.

Remove Unnecessary Services

Execute the following steps to remove unnecessary services:

- Remove the etrustadmin service using the command:

```
dxserver remove etrustadmin
```
- Remove the etadmintemp service using the command:

```
dxserver remove etadmintemp
```
- Remove the etaops_notify service using the command:

```
dxserver remove etaops_notify
```

Import Files from the Primary Server Provisioning Directory Installation

Execute the following procedure to import files from the primary domain installation or from the high-availability samples folder.

To import the primary server files

1. Import the following files from the %DXHOME%\config\knowledge directory on ROOT01 or the samples directory:
 - impd.dxc
 - impd_notify_root01.dxc
 - impd_notify_root02.dxc
 - impd_RO1_root01.dxc
 - impd_RO2_root01.dxc
 - impd_RO1_root02.dxc
 - impd_RO2_root02.dxc
 - impd_root01.dxc
 - impd_root02.dxc
 - impd_router.dxc
 - impd_router_root01.dxc
 - impd_router_root02.dxc
 - impd_temp_root01.dxc
 - impd_temp_root02.dxc
 - impd_tempRO1_root01.dxc
 - impd_tempRO1_root02.dxc
2. Import the following files from samples root\root02\DXHOME\config\servers directory to the root02 %DXHOME%\config\servers directory:
 - impd_notify_root02.dxi
 - impd_RO1_root02.dxi
 - impd_RO2_root02.dxi
 - impd_root02.dxi
 - impd_router_root02.dxi
 - impd_temp_root02.dxi
 - impd_tempRO1_root02.dxi

Remove Unnecessary Files

Execute the following steps to remove unnecessary files. Adjust files to be removed, if needed:

- Change to the %DXHOME%\config\knowledge folder.
 - Remove the etadmintemp.dxc file.
 - Remove the etrustadmin.dxc file
 - Remove the etaops_notify.dxc file
 - Remove the etrustadmin.dxc file
- Change to the %DXHOME%\config\servers folder.
 - Remove the etrustadmin.dxi
 - Remove the etadmintemp.dxi files.
 - Remove the etaops_notify.dxi file

Install and Start Services

Execute the following procedure to install and start the services.

To install and start DSA, SSLD, and Provisioning services

1. Install and start the DSA services using the commands:

```
dxserver install impd_root02
dxserver install impd_temp_root02
dxserver install impd_router_root02
dxserver install impd_tempRO1_root02
dxserver install impd_notify_root02
dxserver install impd_RO1_root02
dxserver install impd_RO2_root02
```

```
dxserver start impd_root02
dxserver start impd_temp_root02
dxserver start impd_router_root02
dxserver start impd_tempRO1_root02
dxserver start impd_notify_root02
dxserver start impd_RO1_root02
dxserver start impd_RO2_root02
```

2. Configure the CA Identity Manager Provisioning Server service so that it uses routers to route Provisioning Directory traffic by updating the file, PSHOME\data\im_ps.conf. Set values as follows:

```
DbHost      root02
DbPort      21399
DbTlsPort   21399
```

Note: You must change the DbHost value to point to the local provisioning directory router using the host name, root02. The port number 21399 corresponds to the LDAP port number defined in the impd_router_root02.dxc.

Reconfigure Dependencies and Restart Services

Reconfigure the dependencies of the CA Identity Manager - Provisioning Server service using the following commands:

```
im_ps install im_ps "IM Provisioning" auto "DXServer_impd_router_root02"
```

Import DSA Personality Files from ROOT01

When the eTrust Directory dxcertgen utility is run, the personality files of the DSAs on ROOT02 were also generated, because the DSA knowledge files of ROOT02 were also present on ROOT01. You can simply import the personality files for DSAs on ROOT02 to the %DXHOME%\config\ssld\personalities folder on ROOT02. In addition, you need to import %DXHOME%\config\ssld\etrustadmin_trusted.pem from ROOT01 to ROOT02.

Optimize Performance for the Provisioning Directory

The Provisioning Server uses inclusion objects to represent association between provisioning directory entries. Each provisioning directory entry is assigned a unique id called eTID; an exclusion object also stores the eTIDs of the parent and child provisioning directory entries respectively as eTPID and eTCID.

For example, the group and user membership is represented using inclusion objects.

Applications that demand frequent searches for inclusion objects can tune the performance as explained below. This approach can be used to tune the group-user membership searches needed by the Identity Manager Server.

To optimize performance for the Provisioning Directory

1. Locate the file that has the default name `etrustadmin.dxi` in this location:
`C:\Program Files\CA\eTrust Directory\dxserver\config\servers`

If you have followed the naming convention in this guide, it is renamed:
`impd_root01.dxi` or `impd_root02.dxi`
2. Add the following indices to the current cache-index settings in this file as follows:
`set cache-index = eTID, eTPID, eTCID, existing attributes`
3. Execute `dxindexdb` to add hash-indices for attributes. For example, if the database is renamed `impd_root01`, you would execute this command :
`dxindexdb impd_root01 +hash eTSuperiorClassEntry eTSubordinateClassEntry`

Provisioning Directory Security Settings

After you configure all the DSAs for both `ROOT01` and `ROOT02`, the next step is to ensure that all DSAs can communicate with each other in a secure manner, without sacrificing performance.

Note: When two DSAs are on the same system, communication can be done via a localhost connection.

The general guidelines for provisioning directory security settings used in Identity Manager high-availability deployments are as follows:

- Each provisioning directory DSA has its own personality file, with the certificate subject field matching the `dsa-name` in the DSA knowledge file.
- Only communication across a network is encrypted. There is no need to encrypt local traffic.
- Trust among various DSAs must be properly established.

Generate Personality Files

eTrust Directory r8.1 provides a utility called `dxcertgen.exe` that you can use to generate personality files, given the DSA knowledge files in the `%DXHOME%\config\knowledge` directory. See the *eTrust Directory Administrator Guide*, for more details on `dxcertgen`.

To generate personality files on the primary root domain server, ROOT01

1. Verify that all of the DSA knowledge files are available in the primary root domain installation.
2. Execute the `dxcertgen` command to generate personality files for the configured Provisioning Server DSAs in `%DXHOME%\config\knowledge`, as shown in the following example:

```
dxcertgen -d 365 -i "dc=etadb" certs
```

-d

Specifies the number of days that certificates are valid, one year in this example.

-i

specifies the issuer of the certificate, `dc=etadb` in this example.

certs

Subcommand of `dxcertgen`, which means certificates

3. Run the following command for reporting after the personality files are generated:

```
dxcertgen report
```
4. Locate the security certificate with the issuer `dc=etadb` in the generated root Certificate Authority (CA) certificate, stored in `%DXHOME%\config\ssld\trusted.pem` file. Move and append this certificate to `etrustadmin_trusted.pem`.

Note: The `etrustadmin_trusted.pem` file is bundled with the `et2_cacert.pem` certificate file.

Import the ROOT Certificate

Execute the following steps to import the root CA certificate from the primary domain, ROOT01, to the alternate domain, ROOT02:

- Copy the personality files for the DSAs of the alternate domain installation to ROOT02.
- Copy the `etrustadmin_trusted.pem` file from ROOT01 to ROOT02.

How to Make Connector Servers Highly Available

With the Connector Server Framework (CSF), you can run multiple Connector Servers and configure the Provisioning Servers to communicate with Connector Server in specific contexts.

As a result, the Provisioning Server can:

- Support Connector Servers on different platforms to manage endpoint types that are unavailable on the platform where the Provisioning Server is installed.
- Communicate with multiple Connector Servers, which each manage a different set of endpoint types or endpoints. Therefore, endpoint types or endpoints can be managed on a parallel basis to achieve load balancing.

Install Connector Servers

Based on the guidelines in this guide, you make connector servers highly available by installing several instances of Java Connector Servers or C++ Connector Servers, or both.

- To install the Java Connector Server, see the *Java Connector Server Implementation Guide*.
- To install the C++ Connector Server, or to upgrade Super Agents to be C++ Connector Servers, use the Identity Manager installation program, which is described in the *Installation Guide*.

Deployment of the Connector Server Framework

With the Connector Server Framework, the Provisioning Server can have multiple Connector Servers in operation. Each Connector Server handles operations on a distinct set of endpoints. Therefore, your organization may choose to deploy Connector Servers on servers that are close in the network to the endpoints. For example, if you have many UNIX /etc endpoints, you might have one Connector Server installed on each of these servers so that each Connector Server controls only the endpoint on the server where it is installed.

The Distributed Connector Server Framework has two important characteristics:

- Scalability - multiple Connector Servers may share the load of working on a set of endpoints.

For example, a lengthy exploration of a endpoint on one Connector Server does not influence the ability to operate on a endpoint that is being controlled by another Connector Server

- Communication channel security - communication between Provisioning Server and Connector Server is encrypted using TLS.

If an endpoint type uses a proprietary protocol to communicate between the Connector Server and endpoints of that protocol, the extent of use of the proprietary protocol may be limited to a local network, or even to just local communication inside one server.

When deciding on a deployment strategy, consider these factors so that you protect the Connector Servers in your organization against unauthorized access:

- The Connector Server may be configured to disclose passwords in clear text.

Any person with access to the server running the Connector Server and with sufficient privileges to modify the configuration of the Connector Server and to restart the Connector Server will be able to make the Connector Server log passwords in the clear.

The Connector Server is based on the open source slapd process. The instructions to make a slapd process log incoming passwords in clear text are in the public domain, for example, by looking at the manual pages at <http://www.openldap.org>

- The Connector Server is only protected by a bind password.

The Connector Server trusts any client who connects to it and is able to provide the proper credentials, such as Bind DN and Bind Password. The Connector Server does not know if the connection comes from a Provisioning Server or not. Any user with internal access may disclose the bind password, then connect to the Connector Server from another server, and so have administrator privileges over the endpoints controlled by the Connector Server.

- The Connector Server is not protected against brute force attacks on the bind password

Unlike the Provisioning Server, the Connector Server is not protected against repeated attempts at binding with different passwords. An attacker may therefore try to guess the password by brute force attack. Should an attacker succeed in guessing the bind password, then the road is open for the attacker to control the endpoints under control of the Connector Server.

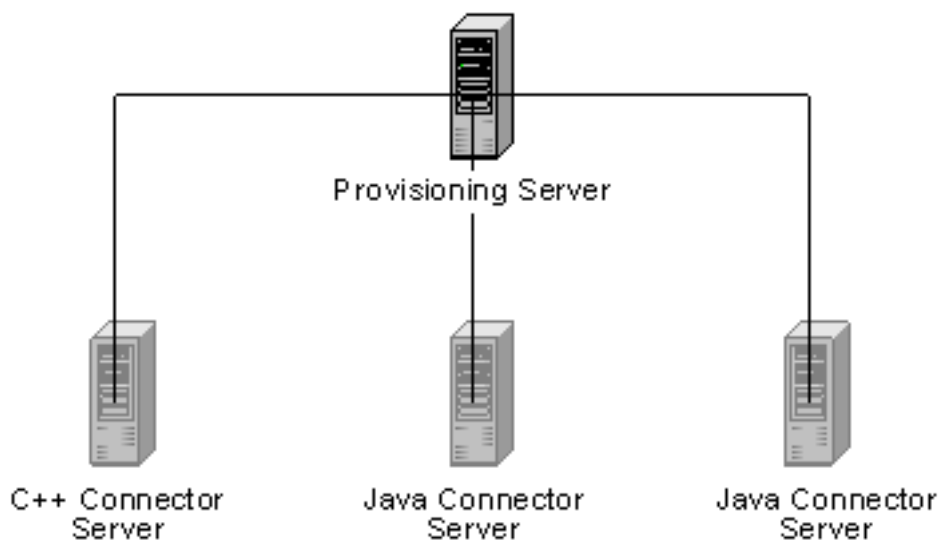
For these reasons you are advised to design your deployment such that

- The same organizational unit is responsible for administrative access to all Provisioning servers and Connector Servers.

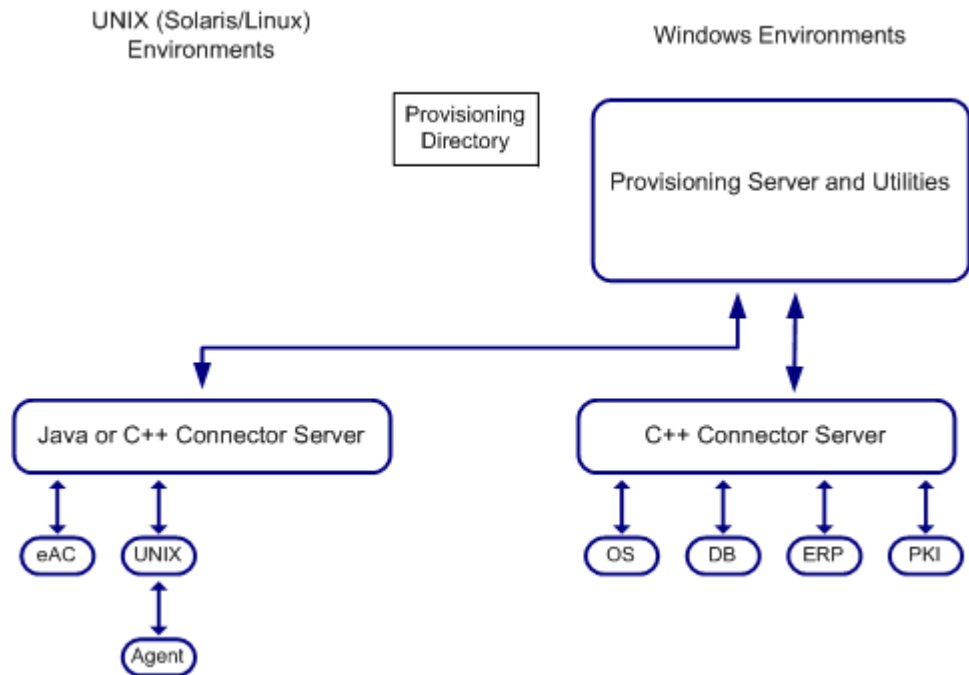
- Your Connector Servers are suitably protected by firewalls or similar such that the ports may not be reached by unauthorized means.
- The ability to connect to Provisioning Servers and Connector Servers on non-TLS ports should be disabled in your production environments.

Deployment Examples

You have various alternatives for deploying connector servers. For example, multiple connector servers can service one Provisioning Server.



You can have centralized directory model with a single domain, which is a Provisioning Server (on Windows) that communicates to one or more connector servers (on Windows or UNIX).



Alternatively, a Provisioning Server (on UNIX) can communicate to one or more connector servers (on Windows or UNIX).

Multi-Platform Installations

The Connector Server Framework is the configuration of Connector Servers that exist on multiple systems, which could be Windows or Solaris systems.

The following use cases are now supported:

- Use Case 1
 - Provisioning Server and connector server installed on a Solaris system.
 - A second Connector Server installed on a Windows system, serving the non-multi-platform connectors.

- Use Case 2
 - Provisioning Server and connector server installed on a Windows system.
 - A second Connector Server installed on Solaris system, serving the multi-platform connectors.
 - A third Connector Server installed on a remote Windows system, serving the other connectors.
- Use Case 3
 - Provisioning Server installed on Windows/Solaris; collocated Connector Server installed by default respectively on the same Windows/Solaris system.
 - Multiple additional Connector Server instances installed on Windows/Solaris systems, serving as end-point agents. This scenario is important for cases where the connector is using a proprietary or un-secured communication channel. Using this topology, the important segment of network traffic is secured by the standard Provisioning Server to Connector Server communication protocol and not by the proprietary protocol.

Configuring the Connector Server Framework

You configure the Connector Server Framework by using the `csfconfig` command. This command uses the data in the Windows Registry (or UNIX counterpart created for Provisioning Server) to connect to a Provisioning Server). The `csfconfig` command must run on the system where the Provisioning Server runs.

Using the command, you can:

- Add or modify a Connector Server connection object with information such as the connector server, host, and port.
- Define for which endpoints or endpoint types the connector server is used; possibly varying this definition for alternate provisioning servers.
- Delete the Connector Server connection information object.
- List all connector server connection objects in a domain.
- Show one or all connector server connection objects.

The `csfconfig` command uses the authorizations provided by a global user credential, so that global user must have the necessary administrative privileges to manipulate the appropriate `ConfigParam` and `ConfigParamContainer` objects.

csfconfig Command

To use the csfconfig command, the command line syntax is:

```
csfconfig [-help[=op]] [operation] [argument]
```

You can use these arguments in any order. The operation argument is required unless you are using the --help argument.

The --help[=op] option provides minimal on-line help. The "=op" argument may be used to list the arguments that are required or optional for the operation. For example, "--help=add" will provide a description of the add operation, while "--help" will provide general information.

If help is requested, other arguments are ignored and no request is sent to the server.

The following operations are available.

add

Add a new CS connection object. A name will be generated by this operation if one is not specified by the user. Required arguments: auth, host. Optional arguments: authpwd, br-add, desc, domain, name, pass, port, usetls, debug.

addspec

Adds a branches specialization for one provisioning server.

When you have installed alternative provisioning servers, sometimes a connector server isn't to be used by all of these provisioning servers. Or sometimes different provisioning servers will want to use the same connector servers for different branches (endpoint types or endpoints). A branches specialization is a list of branches that is specific to one provisioning server. Only provisioning servers without a specialization will use the branches specified in the main CS connection object. Required arguments: auth, name, server. Optional arguments: authpwd, br-add, domain, debug.

list

List all CS connection objects. Required arguments: auth. Optional arguments: authpwd, domain, debug.

modify

Modify a CS connection object. Required arguments: auth, name. Optional arguments: authpwd, br-add, br-rem, desc, domain, host, pass, port, usetls, debug.

modspec

Edits a specialization created by addspec. Required arguments: auth, name, server. Optional arguments: authpwd, br-add, br-rem, domain, debug.

remove

Remove an existing CS connection object. Required arguments: auth, name. Optional argument: authpwd, debug.

remspec

Removes a specialization created by addspec. Required arguments: auth, name, server. Optional arguments: authpwd, domain, debug.

modify

Modify an CS connection object. Required arguments: auth, name. Optional arguments: authpwd, br-add, br-rem, desc, domain, host, pass, port, server, tls, usetls.

show

Show a specific CS connection object or show all CS connection objects. Required arguments: auth. Optional arguments: authpwd, name, domain, debug.

Each operation takes several arguments in the form "name=value". Spaces are not allowed before or after the "=" symbol, and if the value contains any spaces, the argument must be quoted appropriately for the platform (Windows or UNIX). Except as noted, the value must be provided, and must be non-empty.

The following arguments are used for the operations as noted above:

auth=<value>

Identify the global user for authentication.

Value format: "name" where name is the global user's name, and domain is the domain in which the global user is defined.

Note: Optionally, the value can be specified as "name@domain" where domain is the domain in which the global user is defined. If the auth domain is not specified, the default provisioning domain is presumed.

authpwd=<value>

Identify a file containing just the global user's password on the first line. If this argument is not specified, the user will be prompted for a password.

Value format: any appropriate operating system file path.

br-add=<value>

Add a new branch. This argument may be specified multiple times to add multiple branches.

Value format: "[[endpoint,]endpoint type][@[domain]]". Use a branch of "@" by itself to represent all branches in the current domain, or "@domain" for all branches in the specified domain. Add "endpoint type" or "endpoint,endpoint type" to identify a specific endpoint type and/or endpoint in the specified domain. If no domain is specified, the default is the "current" domain, as identified by the auth or domain arguments.

br-rem=<value>

Remove an existing branch. This argument may be specified multiple times to remove multiple branches.

Value format: same format as specified for br-add.

debug=<value>

Turns on trace logging for the command. Tracing messages are written to the file PSHOME\logs\etaclientYYYYMMDD.log file.

Value format: The value "yes" enables logging.

desc=<value>

Provide an arbitrary description for the object. If not specified in an add operation, it will default to the value of the host argument.

Value format: an arbitrary string.

domain=<value>

Define the default domain. If not specified, the domain specified in the auth argument is used as the default.

Value format: any valid domain name.

host=<value>

Define the name of the host on which the ConnectorServer runs.

Value format: any legal host name or IP address.

name=<value>

The name of the ConnectorServer object. If not specified during Add, csfconfig will assign a name and display what name was created.

Value format: A case-insensitive string of one or more characters consisting of upper-case English letters (A-Z), lower-case English letters (a-z), digits (0-9), hyphen(-) or underscore(_).

pass[=<value>]

Define the file containing the password for the ConnectorServer connection object. If the value is not specified, the user will be prompted.

Value format: any appropriate OS file path.

Important! The password you must specify is the password you entered when you installed that ConnectorServer or you changed subsequent to install by running the pwdmgr utility on that ConnectorServer system.

port=<value>

Define the port number for the object. This must be a valid port number.

Value format: an integer.

server[=<value>]

In addspec, modspec and remspec commands, define the name of the Provisioning Server that is served by the ConnectorServer. The branches defined in a specialization override, for a particular provisioning server, the branches defined in the CS configuration object by add and modify commands.

Value format: the name of the host where the Provisioning Server is running as returned by the system's hostname command.

Note: The ConnectorServer configuration objects are stored with the other domain configuration parameters in the provisioning directory. While the ConnectorServer configuration parameters can not be viewed or changed with the provisioning manager directly, one can use the provisioning manager (System task, Domain Configuration button) to get a list of known provisioning servers. To do this, open the "Servers" parameter folder and the known provisioning servers will be listed.

usetls[=<value>]

Indicate if TLS should be used to communicate with the ConnectorServer. The value is optional for the add operation only, in which case it defaults to "yes."

Value format: a string "yes" or "no".

Upon successful completion of the add operation, the name of the newly created Connector Server connection object will be listed. For example:

```
Created CS object with name = SA000
```

For most operations, successful or not, the status and a message (if any) will be shown. For example:

```
The host name, port number, or TLS flag was successfully changed. The branch settings were successfully changed.
```

For some errors, such as invalid command line parameters, no status code or server error message is displayed. In these cases, a simple statement of the error will be shown. For example:

```
$ csfconfig add
No authentication information supplied.
For on-line help, use "--help [=<op>]"
```

The csfconfig utility includes a ping-sa command that checks if the Connector Server is available. It does not check whether the connection information for the Connector Server is correct and whether the SSL connection can be establish.

The operation to ping one specific Connector Server requires no credentials, so any user can use csfconfig to ping a specific Connector Server.

host

The name of the system where the Connector Server is installed.

port

The port used by the Connector Server.

The output will be:

```
ldap://<host>:CA Portal ... alive
```

Or

```
ldap://<host>:CA Portal ... unavailable
```

csfconfig Command Examples

To specify that the UNIX and CA Access Control endpoint types should be served by the ConnectorServer running on host "sunserver01" and the remaining endpoint types served by a ConnectorServer running on host "windows02", you would issue the following commands.

Each command execution prompts you for the etaadmin password.

```
csfconfig add \  
auth="etaadmin" \  
br-add="UNIX – etc" \  
br-add="UNIX – NIS-NIS plus Domains" \  
br-add="Access Control" \  
host="sunserver01" \  
usetls="yes"
```

```
csfconfig add \  
auth="etaadmin" \  
br-add="@ " \  
host="windows02" \  
usetls="yes"
```

Load-Balancing and Failover

Failover and load-balancing of connector requests is achieved by each provisioning server based on the CSF configuration defined using csfconfig or Connector Xpress.

Each provisioning server consults the CSF configuration that applies to it and determines which Connector Servers it should use to access each endpoint or endpoint type. Failover and load-balancing occur when there are multiple connectors servers configured to serve the same endpoint or endpoint type.

Failover and load-balancing are unified and cannot be controlled separately. One cannot indicate that a particular connector server is to remain idle except when needed for failover. Instead, a provisioning server that is configured to use two or more connector servers interchangeably will distribute work between these connector servers (load balancing) during normal operation. Should one or more of the Connector Server become unavailable, the remaining connector servers will provide failover support for the unavailable connector servers.

Reliability and Scalability

With the Connector Server Framework (CSF), the Connector Server high availability features increase reliability and scalability.

Reliability is enhanced by having multiple Connector Servers serve a Provisioning Server, so it can continue to function if one or more Connector Servers become unavailable.

For example, if one Connector Server manages the UNIX endpoint type and another manages the Active Directory endpoint type; and the Active Directory Connector Server becomes unavailable, the Provisioning Server can still manage the UNIX endpoint types.

Scalability is achieved by having a mechanism to add more Connector Servers to manage an increasing amount of endpoint types or endpoints. For example, if the number of endpoint types increases to 100, the Provisioning Server can be configured to have 20 Connector Servers, with each Connector Server managing five endpoint types. Or configure 20 Connector Servers with each Connector Server managing overlapping sets of 10 endpoint types to allow for failover and load balancing behaviors as well.

How Load Balancing Works

Load-balancing mechanism allows the server to be configured to have multiple load-balanced Connector Servers. For each request, a Connector Server is selected to handle that request based on the existing workload. The Connector Server with the least load is selected.

If one load-balanced Connector Server becomes unavailable, it is taken out of consideration. When it comes back up, it can be selected by load-balance scheduler.

Load-balancing implies failover in that if one Connector Server becomes unavailable the requests will be routed (failed over) to the other load-balance Connector Server.

How Failover Works

When alternate Connector Servers are configured, they handle the request when the Connector Servers that normally handle the request become unavailable.

The Provisioning Server initially selects a connector server to handle a request and sends the request to that connector server. If an existing connection to a connector server is available, it is used. Otherwise a new connection to that connector server is established. If a connection cannot be established, the alternative connector servers are tried in turn. If none of the connector servers are available, the request cannot be sent and the current operation fails with this error reason.

Furthermore, a request could be sent on a new or existing connection and LDAP could report that the connection is no longer valid, possibly indicating the connector server has gone down before or during the processing of that request. In such cases, the provisioning server tries the same connector server (but not more than twice in all) before continuing on with the other Connector Servers in its list. This retrying of operations can result in sometimes the endpoint system seeing the same operation twice. In some cases (account creation, password changes) the second attempt could report an error caused by the earlier processing.

Unavailable Connector Servers

A Connector Server is considered unavailable if the Provisioning Server cannot contact that Connector Server. This is different from an operation where the connector returns an unavailable error code.

For example, Connector Server 1 manages UNIX and ACC. The Provisioning Server sends a request to Connector Server 1 to add a UNIX account. Connector Server 1 receives the request and calls the UNIX connector. The UNIX connector cannot reach the target UNIX system and returns an unavailable error code. In this case, the Connector Server is not down. The UNIX endpoint may be down. Thus, the Connector Server is not considered unavailable and the Provisioning Server will not re-route the request to another Connector Server. The Provisioning Server will continue to use Connector Server 1 because Connector Server 1 could still be used to manage the ACC endpoint type. The Provisioning Server will also continue to use Connector Server 1 to manage UNIX account.

Note: If the target UNIX system is down, then the UNIX connector cannot reach that endpoint no matter which Connector Server that UNIX connector uses. Thus, it is useless to failover to another Connector Server.

An unavailable Connector Server is downgraded so that it is only considered by the scheduler when no other Connector Servers are available. For example, Connector Server 1 and Connector Server 2 are configured for load-balancing and failover. Connector Server 1 became unavailable so now the server uses Connector Server2. For subsequent operations, the scheduler only uses Connector Server 2. But if Connector Server 2 becomes unavailable, the Provisioning Server checks whether Connector Server 1 is now available.

Moreover, the Provisioning Server has a dedicated connection monitor thread that periodically checks if unavailable connector servers become available. A Connector Server that can be contacted by the connection monitor thread will no longer be listed as unavailable and hence no longer excluded from use.

State-changing Operations

Certain requests change the state of the connector or Connector Server. In particular, the connectors must be told by the Provisioning Server such information as what the connection parameters are to communicate with an endpoint system or what the attribute mappings (metadata) are for endpoint types defined using Connector XPress. When there are multiple Connector Servers managing the same endpoint types or endpoints, then all of these Connector Servers need to be updated when that information changes.

Thus, while most requests to connector servers are sent to just one connector server, certain requests are state-changing requests and must be sent to all applicable Connector Servers. Example state-changing operations are Modify and Delete operations on endpoint or endpoint types. Note that Add requests do not fall into this category as Connector Servers are designed to learn of endpoints and endpoint types only when they are accessed and will learn about new endpoints or endpoint types later, if necessary.

Chapter 6: Provisioning Directory Monitoring

This section contains the following topics:

[Determine Monitoring Thresholds](#) (see page 135)

[Monitoring Log Files](#) (see page 136)

[Perform Backup and Restore](#) (see page 136)

[Observe Router Traffic Using DXconsole](#) (see page 136)

[Stop Local Read-Only DSAs to Synchronize Local Multiwrite DSA](#) (see page 137)

[Enable SSL Encryption](#) (see page 138)

Determine Monitoring Thresholds

Monitoring is one of the most significant activities as part of the system availability and security. Monitoring is needed to determine the source of performance problems, fault detection, and to take corrective action.

Important! Regular monitoring of the multiwrite queue is highly recommended.

For more information, see Using multiwrite in the section on Replication in *eTrust Directory Administrator Guide*

eTrust Directory can provide SNMP counters to an SNMP-aware Enterprise Management Application. eTrust Directory can also provide SNMP traps under the following conditions:

- Authentication Failure
- Alarms
- Directory Updates

You can also use the eTrust Directory Statistics logs to gauge application load over time. This can be invaluable in providing data to both measure and show the growth of the new service.

Monitoring Log Files

To ensure proper operation of your high-availability configuration, you should monitor the following log files:

- Alarm
- Warn
- Stats
- Diag
- Summary
- Trace

All logs can be flushed through the DXserver console. Only the SUMMARY and TRACE logs can be closed from the console.

Perform Backup and Restore

Backup of your Provisioning Directory is perhaps the single most important operation in managing and monitoring your systems. Proper backup procedures ensure minimal downtime in the event of errors, failures, or the need for disaster recovery. Replication of repositories, as in the high availability configurations discussed in this guide, represent a possible avenue of recovery. However, a regular and carefully followed backup plan is essential to ensuring high-availability.

There are a variety of ways to perform backup and restore operations, ranging from file-system backup processes to specific commands issued by a DXserver administrator.

For more information, see *eTrust Directory Administrator Guide*.

Observe Router Traffic Using DXconsole

eTrust Directory is bundled with a utility called DXconsole. You can use DXconsole to monitor the network traffic going through an eTrust Directory router.

The router knowledge files, e.g., `imps_router.dxc`, define the console port for a router. You can adjust the console port number to satisfy your enterprise needs. To monitor traffic sent through a router, start up DXconsole and connect it to the localhost with the port number defined in the router knowledge file.

Because the Provisioning Server uses LDAP for communication among various components, you can set the trace level to ldap.

For more information, see *eTrust Directory Administrator Guide*.

Stop Local Read-Only DSAs to Synchronize Local Multiwrite DSA

For high availability configurations, an alternate server installation usually has a replicated copy of the same provisioning directory as its primary server installation. Multiple read-only (RO) DSAs are often configured, with one write DSA. The write DSAs of both primary and alternate server installations are configured as multiwrite so that updates to one provisioning directory replicate to the other.

One multiwrite DSA is usually designated as the primary write DSA. All updates are sent to the primary multiwrite DSA, which replicates to all the other multiwrite DSAs. Searches are always done by the local read-only DSAs, for performance considerations.

Configurations based on this recommendation may face a situation where the multiwrite DSAs are not providing reliable real-time replication services. This can occur because of a network outage, an unavailable multiwrite DSA or multiwrite DISP recovery, among other reasons. When this situation occurs, the updates to the primary multiwrite DSA will not be properly replicated to the other multiwrite DSAs in a timely manner. In this case, searches to local read-only DSAs may not produce up-to-date results from the Provisioning Directory.

If for some reason, a multiwrite DSA needs to be synchronized or fails to start, you must stop the corresponding local read-only DSAs. This prevents inconsistent results.

The following set of procedures would produce an inconsistent result on ROOT01:

- Stop the multiwrite DSA service.
- Create a Global User object.

An error message appears because the Global User was created successfully using the multiwrite failover DSA (impd_root02), but reading this object fails because the Provisioning Directory on ROOT01 is not yet updated. The update did not occur because the multiwrite DSA service was stopped.

In the example shown previously, the commands for stopping the read-only DSAs on ROOT01 are as follows:

```
dxserver stop impd_RO1_root01  
dxserver stop impd_RO2_root01
```

Enable SSL Encryption

eTrust Directory r8 SP2 provides a utility called `dxcertgen.exe` that you can use to generate certificates and personality files. Using this utility is the recommended method for enabling encryption for high availability solutions.

For more information, see *eTrust Directory Administrator Guide*.

Appendix A: DSA Port Assignments

You can use the DSA Port Assignment worksheets to preplan your DSA port assignments.

Note: Use the same SSL Daemon on a single port for all of the DSAs. If you want to configure more than one SSL Daemon, each one must listen on a unique port number.

This section contains the following topics:

[Provisioning Server DSA](#) (see page 139)

[Provisioning Directory DSA](#) (see page 139)

[Router DSAs](#) (see page 140)

[Read-Only DSAs for Provisioning Directory](#) (see page 140)

[Provisioning Directory Operations DSAs](#) (see page 140)

Provisioning Server DSA

Enter the host name after the `imps_token`, and list the port number on which the Provisioning Server is listening.

DSA Name	Address	SNMP port	Console port	SSLD port
imps _____		-	-	-

Provisioning Directory DSA

Enter the host name after the provisioning directory token, then list the port numbers that you want to use.

DSA Name	Address	SNMP port	Console port	SSLD port
impd				

Router DSAs

Enter the host name for the `impd_router_` token. Write in the host name for the `impd_router_` token for each router that you are using.

DSA Name	Address	SNMP port	Console port	SSLD port
impd_router				

Read-Only DSAs for Provisioning Directory

Enter the following data:

DSA Name	Address	SNMP port	Console port	SSLD port
impd_RO1				
impd_RO2				

Provisioning Directory Operations DSAs

Enter the following data:

DSA Name	Address	SNMP port	Console port	SSLD port
impd_temp				

DSA Name	Address	SNMP port	Console port	SSLD port
impd_tempRO1				

Index

A

- Add Cluster Nodes • 51
- Add More Policy Servers • 65
- Add Nodes to the Cluster • 56
- Add the Provisioning Server Dependency • 113
- Application Server Information • 29

C

- CA Product References • iii
- Client-Tier • 89
- Cluster Startup • 72
- Configure a Distributed JMS Server • 35
- Configure a Distributed JMS Server for Workflow • 46
- Configure a Policy Server Cluster • 25
- Configure JMS Objects on a Single Server • 33
- Configure Local and Remote Failover Servers • 97
- Configure Managed Nodes • 41
- Configure Server-Tier Router DSA • 100
- Configure Server-Tier Router DSA Group File • 98
- Configure the Alternate Domain Server Provisioning Directory on ROOT02 • 114
- Configure the Cluster with the First Node • 56
- Configure the JK Connector (JBoss) • 70
- Configure the Provisioning Directory and Router DSAs with the Knowledge of All DSAs on ROOT01 and ROOT02 • 109
- Configure the Provisioning Server service to Direct Provisioning Directory Requests to Local Routers • 113
- Configure the Proxy Plug-In (WebLogic) • 69
- Configure the Proxy Plug-In (WebSphere) • 71
- Configuring the Connector Server Framework • 125
- Consistent Port Assignments • 85
- Contact Technical Support • iii
- Create a Distributed JMS Server for WebLogic 9.2 • 45
- Create a Message Store • 62
- Create a WebLogic 8.1 Cluster • 32
- Create a WebLogic 9.2 Cluster • 38
- Create a WebSphere Cluster • 54

- Create and Organize the DSA Knowledge Files on ROOT01 and ROOT02 • 108
- Create Core Group Policies • 59
- Create DSA Indexes • 105
- Create JMS Objects • 33
- Create Local Server DSA Knowledge File • 98
- Create Remote Failover Server Knowledge File • 99
- Create the First Cluster Node • 48
- Create the Provisioning Directory and Router DSAs on ROOT01 and ROOT02 • 109
- csfconfig Command • 126
- csfconfig Command Examples • 131

D

- Deploy the Identity Manager Server to Clustered Nodes • 36
- Deployment Examples • 123
- Deployment of the Connector Server Framework • 121
- Determine Monitoring Thresholds • 135
- Domain Prerequisites • 91
- DSA and Router Names Rule • 81
- DSA Configuration Considerations • 79
- DSA Configuration Example • 86
- DSA Port Assignments • 139
- Dump the Primary Provisioning Directory Contents • 104

E

- Enable Client-Tier Failover for Provisioning Manager • 94
- Enable SSL Encryption • 138
- Export process definitions • 68

G

- Generate Personality Files • 119
- Generate Personality Files for All Provisioning Directory DSAs • 112
- Global Naming Convention • 80

H

- High Availability Overview • 9
- How Failover Works • 133
- How Load Balancing Works • 132

How to Install Identity Manager for High Availability • 17

How to Introduce and Configure Local X.500 Provisioning Directory and Router DSAs • 107

How to Make Connector Servers Highly Available • 121

How to Make Provisioning Clients Highly Available • 93

How to Make Provisioning Directories Highly Available • 101

How to Make Provisioning Servers Highly Available • 95

How to Migrate a Standalone Installation to a Cluster • 22

How to Reconfigure CA Identity Manager - Provisioning Server service Dependencies on Windows • 112

How to Reconfigure DSAs on the Primary Domain Server • 106

How to Rename and Reconfigure Provisioning Directory DSAs • 110

How to Upgrade a High Availability Installation • 18

How to Use DXLink Routers for High Availability • 96

I

Identity Manager Cluster • 29

Identity Manager Server Architecture • 10

Import DSA Personality Files from ROOT01 • 118

Import Files from the Primary Server Provisioning Directory Installation • 116

Import process definitions • 68

Import the ROOT Certificate • 120

Install and Start Services • 117

Install Connector Servers • 121

Install Identity Manager • 40, 57

Install Identity Manager and WebLogic • 32

Installation, Upgrade, and Migration • 17

J

JBoss Clusters • 47

K

Knowledge File Relationships • 97

L

Load the Replicated Provisioning Directory Data • 104

Load-Balancing and Failover • 131

Load-Sharing Limitations • 65, 77

M

Migrate the Policy Store • 25

Migrate Workflow Process Definitions • 67

Modify Policy Server Connection Settings • 64

Modify the Plug-in for an IIS Web Server • 69

Modify the Plug-in for an iPlanet Web Server • 70

Monitoring Log Files • 136

Multi-Platform Installations • 124

O

Observe Router Traffic Using DXconsole • 136

Optimize Performance for the Provisioning Directory • 118

P

Perform Backup and Restore • 136

Planning • 78

Policy Server and Application Server Communication • 66

Policy Server Cluster • 25

Port Assignment Example • 87

Preconfiguration Considerations • 78

Primary and Alternate Provisioning Servers • 88

Provisioning Components • 77

Provisioning Components Architecture • 12

Provisioning Directory DSA • 139

Provisioning Directory Monitoring • 135

Provisioning Directory Operations DSAs • 141

Provisioning Directory Security Settings • 119

Provisioning Directory Tier • 89

Provisioning Domain Specifications • 88

Provisioning Server Disaster Recovery • 101

Provisioning Server DSA • 139

Purpose of High Availability • 9

R

Read-Only DSAs for Provisioning Directory • 140

Reconfigure Dependencies and Restart Services • 118

- Register Node Manager • 39
- Re-install Services on ROOT01 • 111
- Reliability and Scalability • 132
- Remove Default Provisioning Directory Services on ROOT01 • 111
- Remove Existing Windows Service Dependencies • 115
- Remove the Existing Windows Service Dependencies • 112
- Remove Unnecessary Files • 117
- Remove Unnecessary Services • 115
- Replicate the ROOT Domain Provisioning Directory • 103
- ROOT Domain Installation • 90
- Router DSAs • 140

S

- Sample Configuration Files • 81
- Sample DSA Names • 85
- Sample Provisioning Configuration • 86
- Select Load Balancing or Fail Over • 65
- Server Tier • 89
- SiteMinder High Availability for the Application Server Cluster • 63
- Start the Cluster Domain • 45
- Start the Cluster for WebSphere • 74
- Start the Servers for JBoss • 73
- Start the Servers for WebLogic • 72
- State-changing Operations • 134
- Stop Directory Services • 115
- Stop Local Read-Only DSAs to Synchronize Local Multiwrite DSA • 137

T

- Test the Client-Tier Failover Configuration • 95
- Test the Default Multicast Address • 47
- Test the Server Failover • 100

U

- Unavailable Connector Servers • 133
- Update Cluster Members • 60
- Update the Migrated Environments • 75
- Upgrade an Identity Manager Cluster • 18
- Upgrade Highly Available Provisioning Components • 19
- Use SiteMinder for High Availability • 95

V

- Verify the Cluster • 39

- Verify the Clustered Installation • 75

W

- Web Server Clusters • 69
- WebLogic 8.1 Clusters • 30
- WebLogic 9.2 Clusters • 37
- WebLogic Clusters • 30
- WebSphere Cluster Prerequisites • 54
- WebSphere Clusters • 52