Sun Java Desktop System Configuration Manager, Release 1 - Installation Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–0635
September, 2004

040902@9495

Contents

Tables

Figures

Examples

CHAPTER 1

Introduction

Provides a brief introduction about the Sun Java™ Desktop System Configuration Manager, Release 1.

Introduction

The Sun Java™ Desktop System Configuration Manager, Release 1 is aimed at providing a central configuration for desktop hosts running the Sun Java™ Desktop System. Settings can be assigned to various elements of an organization structure, enabling the administrator to manage groups of users or hosts in a simple way. Its main components are:

- An LDAP server containing the organizational structure of users and hosts to be managed, which will hold the configuration data,
- A web-based management tool allowing administrators to define and assign configuration data to elements of that organizational structure,
- Desktop components installed on the client host, which retrieve the configuration data on behalf of the currently logged in user and expose it to the various applications making up the Sun Java Desktop System.

The management tool is a web-based application run within the Sun Web Console. It allows the administrator to browse the organization structure of the LDAP server and assign policies to its elements. The policies are displayed and edited according to policy templates, which define the settings that the management tool will be manipulating.

The desktop components are organized around the Sun Java™ Desktop System Configuration Agent, which retrieves configuration data from the LDAP server on behalf of users and makes it available to a number of configuration system adapters, which complement the local configuration (default settings provided by applications

and user settings) with the policy settings. Currently supported configuration systems are GConf (which handles the configuration of Gnome applications such as the Gnome desktop or Evolution), Mozilla™ Preferences and StarRegistry (StarOffice's configuration system).

CHAPTER 2

LDAP Server

This chapter provides information about setting up and deploying an LDAP server for use with the Configuration Manager.

Concepts

Within the Java Desktop System Configuration Manager framework, configuration data are associated with entities, which are entries in the LDAP repository and correspond to elements of the organizational structure of the company.

The recognized entities are:

- Organization: typically represents either an organizational (divisions, group, teams) or geographical (continent, country, site) unit of the overall hierarchy.
- User: represents a leaf node of the overall hierarchy and, as the name implies, usually a user.
- Domain: represents a logical structuring unit for the network organization.
- Host: also represents a leaf node of the overall hierarchy but points to a machine on the network.
- Role: represents properties, usually a distinction in terms of function (administrator, site management), applied to a set of users.

The organization and user entities are used to define a user tree, while the domain and host entities define a host tree. These two trees are independent but are manipulated in a similar way in the framework.

The relationship of organization and domain entities with other entries is defined by the physical location of the entries within the repository. That is, the organization and domain entities can include any entry that is located below these two entities in the tree. The relationship of roles to users or hosts is defined by the attributes of the user and host entries.

The configuration data that are associated with an entity are stored in special entries that are managed by the framework. These entries are identified by the service name and the service container that are associated with the entries.

Setup

To use an existing LDAP server with the Configuration Manager, you need to:

- Extend the server schema to support the custom object classes and the attributes that are used by the Configuration Manager to store configuration data.
- Customize and store in the server the mapping information for the entries in the repository as well as the entities that are supported by the Configuration Manager.

Deployment Tools

The following deployment tools from the installation CD are required to use an existing LDAP server with the Configuration Manager:

- `88apoc-registry.ldif`: A schema file that introduces the object classes and attributes that are required for the storage of configuration data.
- `OrganizationMapping`: A default properties file that describes the mapping between LDAP entries and Configuration Manager entities.
- `UserProfileMapping`: A default properties file that describes the mapping between LDAP user entry attributes and Configuration Manager user profile attributes.
- `createServiceTree`: A script that stores the mapping files in the LDAP repository.
- `deployApoc`: A script that extends the schema of the LDAP server and that stores the mapping files in the LDAP repository.

Schema Extension

The configuration data are stored in entry trees that are attached to the entries that the data are associated with. Before you can store the object classes and attributes that are used by these trees on an LDAP server, you must add the objects and classes to the LDAP server schema. For example, the schema extension file that is provided uses the LDIF format to add these objects and classes to the Sun Java™ System Directory Server. To add these objects and classes to other LDAP servers, you need to use a format that is recognized by the servers.

Organizational Mapping

To define the mapping between the LDAP entries and Configuration Manager entities, the `Organization` mapping file must be edited. Values that match the layout of the LDAP repository must be provided for the various keys.

User entities are identified by an object class that all entities use, as well as an attribute whose value must be unique within the whole repository. A display name format can be provided which will affect how users are displayed in the management application and optionally a container entry can be defined if the user entries within an organization use such an entry. The key names and their default values are:

```
# Object class that all user entries use
User/ObjectClass=inetorgperson
# Attribute whose value in user entries is unique within the repository
User/UniqueIdAttribute=uid
# Optional container in organization entries of the user entries,
# remove line if not used
User/Container=ou=People
# Display name format within the management application
User/DisplayNameFormat=sn, givenname
```

Role entities are identified by a list of possible object classes that they use, along with the corresponding naming attributes. These lists use the format `<item1>,<item2>,...,<itemN>` and must be aligned. That is, the lists must have the same number of items and the nth object class must be used with the nth naming attribute. Two keys define the relationship between roles and users as well as between roles and hosts. The *VirtualMemberAttribute* key must specify an attribute whose values can be queried from a user or host entry. The key must also contain the full DNs of the roles that the entry belongs to. The *MemberAttribute* key must specify an attribute from a user or host entry for the search filter. The key must also contains the full DNs of the roles that the user or host belongs to. The *VirtualMemberAttribute* key can be a Class Of Service virtual attribute, whereas the *MemberAttribute* key must be a physical attribute that can be used in a filter. The key names and their default values are:

```
# List of object classes for roles
Role/ObjectClass=nsRoleDefinition
# Aligned list of corresponding naming attributes
Role/NamingAttribute=cn
# Physical attribute (usable in a filter) containing the DNs
# of the roles of a user/host
Role/MemberAttribute=nsRoleDN
# Attribute whose query on a user or host return the DNs of the
# roles it belongs to
Role/VirtualMemberAttribute=nsRole
```

Organization entities are identified in a way similar to roles, with two aligned lists of object classes and corresponding naming attributes. The key names and their default values are:

```
# List of object classes for organizations
Organization/ObjectClass=organization
```

```
# Aligned list of corresponding naming attributes
Organization/NamingAttribute=o
```

Domain entities are identified in a way that is similar to organization entities. The key names and their default values are:

```
# List of object classes for domains
Domain/ObjectClass=ipNetwork
# Aligned list of corresponding naming attributes
Domain/NamingAttribute=cn
```

Host entities are identified in a way that is similar to user entities. The key names and their default values are:

```
# Object class that all host entries use
Host/ObjectClass=ipHost
# Attribute whose value in host entries is unique within the repository
Host/UniqueIdAttribute=cn
# Optional container in domain entries of the host entries,
# remove line if not used
Host/Container=ou=Hosts
```

User Profile Mapping

To define the mapping between the LDAP user entries attributes and Configuration Manager user entities attributes, the User Profile mapping file must be edited. Each key corresponds to a Configuration Manager user attribute. A key can be assigned as a value to the name of an attribute in a user entry, as identified by the organizational mapping. Attributes which are used in the `User/DisplayNameFormat` setting must be assigned in the *User Profile* mapping. The key names and their default values are:

```
# inetOrgPerson.givenName
org.openoffice.UserProfile/Data/givenname = givenname
# person.sn
org.openoffice.UserProfile/Data/sn = sn
# inetOrgPerson.initials
org.openoffice.UserProfile/Data/initials = initials
# organizationalPerson.street
org.openoffice.UserProfile/Data/street = street,postalAddress,streetAddress
# organizationalPerson.l (city)
org.openoffice.UserProfile/Data/l = l
# organizationalPerson.st (state)
org.openoffice.UserProfile/Data/st = st
# organizationalPerson.postalCode
org.openoffice.UserProfile/Data/postalcode = postalcode
# country.c (country)
org.openoffice.UserProfile/Data/c =
# organizationalPerson.o (company)
org.openoffice.UserProfile/Data/o = o,organizationName
# deprecated -- no LDAP corollary
org.openoffice.UserProfile/Data/position =
# organizationalPerson.title
```

```
org.openoffice.UserProfile/Data/title = title
# inetOrgPerson.homePhone
org.openoffice.UserProfile/Data/homephone = homephone
# organizationalPerson.telephoneNumber
org.openoffice.UserProfile/Data/telephonenumber = telephonenumber
# organizationalPerson.facsimileTelephoneNumber
org.openoffice.UserProfile/Data/facsimiletelephonenumber =
facsimiletelephonenumber,officeFax
# inetOrgPerson.mail
org.openoffice.UserProfile/Data/mail = mail
```

Deployment

Once the mapping files have been customized to reflect the state of the LDAP repository, they can be deployed. If the schema of the LDAP server already contains the required object classes and attributes, the script `createServiceTree` can be run directly, otherwise the script `deployApoc` must be run.

The `deployApoc` script is targeted for use with Sun Java™ System Directory Servers. It will copy the provided schema extension file to the proper directory and cycle the LDAP server, then invoke the `createServiceTree` script. It must be run as a user with permissions to copy files in the schema repository and restart the server and be invoked by:

`./deployApoc` <*Directoy Server Directory*>

The <*Directoy Server Directory*> parameter must be the path to the `slapd-`<*server name*> subdirectory of a Directory Server installation. Assuming the installation used the default directories and the server is named *myserver.mydomain*, that directory would be `/var/Sun/mps/slapd-myserver.mydomain`.

The `createServiceTree` script, whether invoked directly or from the `deployApoc` script, will prompt the user for the location of the LDAP server (host name, port number and base DN) and for the definition of a user with administrative rights (full DN and password). The script then creates a bootstrap service tree in the LDAP server and stores the mapping files in it. It can be run as any user and is invoked by:

```
./createServiceTree
```

The user is then prompted for:

- **Host name** (default: **`localhost`**): host name of the LDAP server,
- **Port number** (default: **`389`**): port number of the LDAP server,
- **Base DN**: base DN of the LDAP repository,
- **User DN** (default: **`cn=Directory Manager`**): full DN of a user with enough permissions to create new entries under the base DN,
- **Password**: password of that user,

An entry whose DN is:

```
ou=ApocRegistry,ou=default,ou=OrganizationConfig,ou=1.0,
ou=ApocService,ou=services, <baseDN>
```

is created and filled with the contents of the two mapping files.

As mentioned previously, the operations performed by the `deployApoc` script assume an LDAP server whose installation directories, layout, and schema extension procedure closely match Sun Java System Directory Server's one. Other directories will require a manual extension of the schema before being able to run the `createServiceTree` script. For further information concerning the use of OpenLDAP and ActiveDirectory, refer to Appendix C.

The created tree, which matches the one which will hold configuration data associated to entities, is aligned with the structure of the trees used for service management in Sun Java System Identity Server.

Additional Considerations

The Configuration Manager framework requires that a connection to the LDAP server, with read and search permissions, can be created in order to identify which full DN is associated with a given user or host identifier coming from the desktop. As such, the repository must be configured so as to either allow anonymous connection, or a special user with read and search access must be created for that purpose.

The management application creates service trees under entries mapped into entities to hold the configuration data for these entities. As such, user entries used for management purposes need to have the right to create subentries under the entries they are managing.

Authentication of the users of the framework from the desktop clients can be done with two methods named `Anonymous()` and `GSSAPI()`. The `Anonymous()` method requires that anonymous access for read and search is enabled throughout the repository as the desktop clients will not provide any credentials when attempting to retrieve data from the LDAP server. To use the `GSSAPI()` method (using Kerberos for authentication), the LDAP server must be configured as described in the "Managing Authentication and Encryption" chapter of the *Sun Java™ System Directory Server 5 2004Q2 Administration Guide*.

CHAPTER 3

Sun™ Web Console

The Sun Web Console is designed to produce a common, web-based system management solution for Sun Microsystems. It serves as one location where users can access system management applications, all of which provide a consistent user interface.

The console is based on a web model for many reasons. However, the primary reason is to enable system administrators to use a web browser to access their system management applications.

The Sun Web Console provides the following:

- Common authentication and authorization
- Common logging
- A single entry point for all system management applications though the same HTTPS-based port
- A common look and feel

A major advantage of the console is that the administrator can log in once and use any application inside the console.

System Requirements

The Sun Web Console supports multiple client and server operating systems as well as several browsers.

Client

- Netscape™ 4.7x, 6.2x, and 7.x on Solaris™ 8 or higher
- Netscape 4.7x, 6.2x, and 7.x on Windows 98, 98 SE, ME, 2000, and XP
- Internet Explorer 5.x and 6.x on Windows 98, 98 SE, ME, 2000, and XP
- Mozilla on Linux and on Solaris

Server

- Solaris 8 or higher
- Red Hat 8 or higher, Red Hat Enterprise Linux 2.1
- SuSE Linux 2.1 or higher
- J2SE™ Version 1.4.1_03 or higher

  If J2SE 1.4.1 or earlier is detected on your server, the setup program prompts you to upgrade the installation using the J2SE version from the Java Desktop System Management Tools CD.
- Tomcat: 4.0.3 or higher

  Tomcat is included on the Java Desktop System Management Tools CD

Installing the Sun Web Console

Before you install the Sun Web Console, read the package summary and known issues sections in Appendix A of this guide.

The installation binaries for the Sun Web Console for Solaris SPARC (version 8 or higher) and Linux operating systems are available on the Java Desktop System Management Tools CD.

▼ To install the Sun Web Console

Steps

1. **On the Java Desktop System Management Tools CD, change to the Sun Web Console directory that corresponds to the operating system where you want to install the console.**

   On Linux systems, change to `/linux/swc` and for Solaris SPARC, change to `/solsparc/swc`.

2. **Type `./setup`**

By default, the Sun Web Console does not create an installation log file. To create an installation log with the name "logfile," type **`./setup | tee logfile`**.

Note – Most of the installation and the configuration of the web console is automatically performed when you run setup. For more detailed information on the setup application for the Sun Web Console, see Appendix A.

---

3. **If you want to localize the Sun Web Console, you need to install two additional packages for each language. Use the table below to determine the package names for the language, and do one of the following:**
   - On Solaris, type **`pkgadd -d path/pkgname.pkg pkgname`**, where *pkgname* is the name of the language package that you want to add.
   - On Linux, type **`rpm -i path/pkgname<...>.rpm`**, where *pkgname* is the name of the package that you want to add.

Package Name	Description
SUNWcmcon, SUNWcmctg	Simplified Chinese Sun™ Web Console 2.0
SUNWdmcon, SUNWdmctg	German Sun™ Web Console 2.0
SUNWemcon, SUNWemctg	Spanish Sun™ Web Console 2.0
SUNWfmcon, SUNWfmctg	French Sun™ Web Console 2.0
SUNWhmcon, SUNWhmctg	Traditional Chinese Sun™ Web Console 2.0
SUNWimcon, SUNWimctg	Italian Sun™ Web Console 2.0
SUNWjmcon, SUNWjmctg	Japanese Sun™ Web Console 2.0
SUNWkmcon, SUNWkmctg	Korean Sun™ Web Console 2.0
SUNWsmcon, SUNWsmctg	Swedish Sun™ Web Console 2.0

Running the Console

You typically only need to stop and to restart the Sun Web Console server when you want to register a new application.

Before you start the Sun Web Console for the first time, ensure that the Configuration Manager installation is completed.

- To start the Sun Web Console, type **`smcwebserver start`**.

- To stop the Sun Web Console, type **`smcwebserver stop`**.
- To access the Sun Web Console, enter the following URL in your browser: `https://`*<hostname>*`.`*<domainname>*`:6789`

Out of the box, the Sun Web Console supports Unix-based authentication and Role-Based Access Control (RBAC). However, you can also configure other authentication mechanisms, such as LDAP authentication.

Note – The default session time-out is 15 minutes. You can configure the time-out length with the `smreg` command. For example, to set the time-out length to 5 minutes, type **`smreg add -p -c session.timeout.value=5`**.

For more information on commands for the Sun Web Console, see the smcwebserver and smreg man pages.

Uninstalling the Sun Web Console

To uninstall the Sun Web Console, run **`/usr/lib/webconsole/setup -u`**.

Note – Do not run this command if you are in the `/usr/lib/webconsole` directory or any of the related subdirectories, otherwise `pkgrm` fails.

Sun Web Console Port Information

The Configuration Manager uses the ports of the Sun Web Console:

- 8005 to shut down the service, and
- 6789 for https access.

The two ports can be changed in `/etc/opt/webconsole/server.xml`. After changing the ports, restart the Sun Web Console with **`/usr/sbin/smcwebserver restart`**.

CHAPTER 4

Sun Java™ Desktop System Configuration Manager, Release 1

The Configuration Manager provides a administration tool that runs on the Sun Web Console. This web-based user interface allows an administrator to traverse the hierarchy of an organization to define policies for desktop applications. These policies can be defined for each item in the hierarchy, for example, for organizations, roles, users, domains and hosts. The Configuration Manager uses several configuration templates to display settings that are specific to different desktop applications such as Gnome, Mozilla, StarOffice, and Evolution.

Installing the Configuration Manager

Before you can install the Configuration Manager you need a working installation of the Sun Web Console.

▼ To Install the Configuration Manager:

Steps

1. **Change to the corresponding Configuration Manager directory on the Java Desktop System Management Tools CD.**

   On Linux systems, change to `/linux/apoc`. On Solaris SPARC, change to `/solsparc/apoc`.

2. **Type `./setup`.**

3. **Enter the host name of the LDAP server.**

   The default name is **`localhost`**.

4. **Enter the port number of the LDAP server (default: `389`).**

5. **Enter the base DN of the LDAP repository.**

6. **Enter the name of the object class that is used to identify user entities. The default object class is `inetorgperson`.**

   For more details, see "Organizational Mapping" on page 17 in the LDAP Server chapter.

7. **Enter an attribute name that is unique within the whole LDAP repository. The default attribute is `uid`.**

   For more details, see "Organizational Mapping" on page 17 in the LDAP Server chapter.

8. **Enter the full DN of a user who has the necessary access rights to perform queries on the LDAP server.**

   Use any full DN that has read and search access. For anonymous access, leave this field empty.

9. **Enter a password for the user who you assigned the LDAP access rights to.**

   If you set up anonymous access to the LDAP server, then ignore this step.

   During the installation, an additional login module is added to the Sun Web Console that allows you to authenticate users through LDAP.

   At the end of the installation, the Sun Web Console automatically restarts so that you can access the Configuration Manager.

Note – You can modify the previous Configuration Manager settings at any time by using the `/usr/share/webconsole/apoc/configure` script. For example, you can use the script to change to a different LDAP server without reinstalling the Configuration Manager.

Running the Configuration Manager

▼ To start the Configuration Manager:

Steps

1. **Type the following URL in your browser:**
   `https://<`*hostname*`>.<`*domainname*`>:6789`

2. **At the prompt, type the user name (uid) and the password of an existing LDAP user.**

The Sun Web Console opens.

3. **In the console window, click Sun Java Desktop System Configuration Manager, Release 1.**

More Information

Direct access to the Configuration Manager

If want to skip the launch page of the Sun Web Console and go straight to the Configuration Manager, enter the following URL in your browser:

`https://`<*hostname*>**`.`**<*domainname*>**`:6789/apoc`**

Uninstalling the Configuration Manager

To uninstall the Configuration Manager from the Sun Web Console, change to the corresponding Configuration Manager directory on the Java Desktop System Management Tools CD, and then run `./setup -u`.

Note – When you uninstall the Configuration Manager, the LDAP login module is removed from the Sun Web Console.

CHAPTER 5

Desktop Components - Linux

To access the configuration data from the Configuration Manager, a desktop client requires the Sun Java™ Desktop System Configuration Agent. The Configuration Agent communicates with the remote configuration data repository and the adapters as well as integrates data into specific configuration systems. The configuration systems that are currently supported are GConf, Mozilla Preferences, and StarOffice Registry.

All of these components are shipped and installed as part of the Java Desktop System.

Data Access/User Authentication

The Configuration Agent retrieves information from the LDAP server based on the login ID of a desktop user. The `User/UniqueIdAttribute` setting of the organizational mapping file maps the login ID to a user entity in the LDAP server. The Configuration Agent also retrieves information about the host, such as the name or the IP address of the host. This information is mapped to a host entity in the LDAP server through the `Host/UniqueIdAttribute` setting of the organizational mapping file.

There are two methods to access the LDAP server, namely anonymously or with GSSAPI. For anonymous access, no action is required on the desktop. For the GSSAPI method, Kerberos credentials must be acquired on the desktop. To integrate Kerberos credential acquisition with the user login, the `pam_krb5()` module must be installed and configured on the Java Desktop System host. You can find example configurations for the `pam()` module in the `/usr/share/doc/packages/pam_krb5/README.SuSE` directory on the Java Desktop System CD. You can also use `gdm` to integrate Kerberos with the user login, for example, by using the following `/etc/pam.d/gdm` file:

```
#%PAM-1.0
auth   required    pam_unix2.so  nullok #set_secrpc
auth   optional  pam_krb5.so use_first_pass missing_keytab_ok ccache=SAFE putenv_direct
account required    pam_unix2.so
password required    pam_unix2.so  #strict=false
session required    pam_unix2.so  # trace or none
session required    pam_devperm.so
session optional    pam_console.so
```

Configuration Agent

The Configuration Agent is part of the `apoc` package. When you install the corresponding RPM, the files that are required for this API are installed and registered with `inetd`. You can install the RPM manually or through the Java Desktop System installation.

Bootstrap Information

To access the remote configuration data, the Configuration Agent must be provided with the location of the LDAP server. You can add this location through the YaST2 configuration tool, autoYaST, or by manually editing the `policymgr.properties` properties file in the `/opt/apoc/lib` directory. In YaST2, you can add this data in the Network/Advanced section.

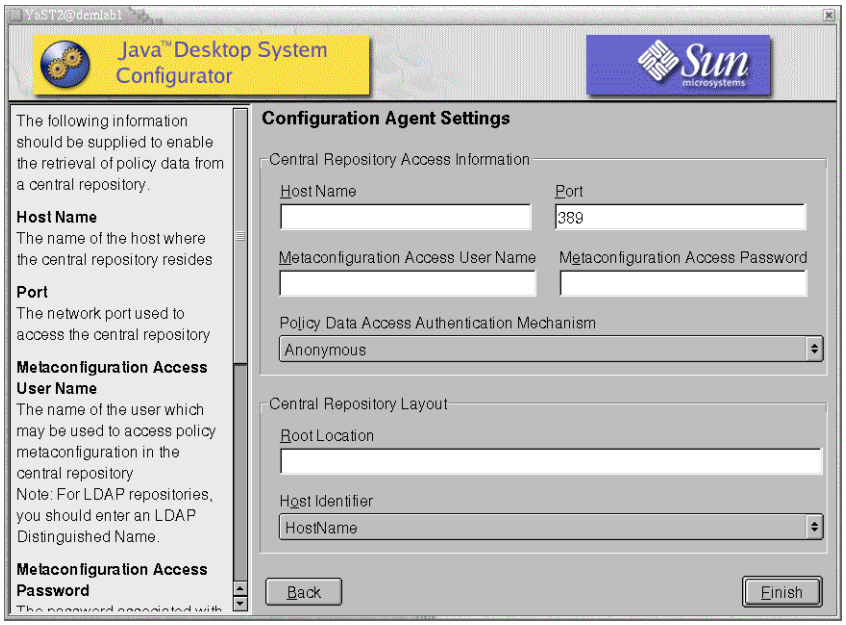

FIGURE 5–1 Java Desktop System Configuration Agent in YaST

The following information is required to run the Configuration Agent:

Note – Associated property file keys are indicated in parentheses, where appropriate.

- **Host Name** (Server): host name of the LDAP server.
- **Port** (Port): port number of the LDAP server.
- **Metaconfiguration Access User Name** (AuthDn): full DN of a user with read and search access rights on the repository.

  Note – If anonymous access is enabled in the directory, this setting can be left blank.

- **Metaconfigurator Access Password** (Password): password of a registered LDAP user.

  Note – If anonymous access is enabled in the directory, this setting can be left blank.

- **Policy Data Access Authentication Mechanism** (AuthType): can be anonymous or GSSAPI, depending on how the LDAP server authenticates users.

- **Root Location** (BaseDn): base DN of the LDAP repository.
- **Host Identifier** (HostIdentifier): can be HostName or IPAddress and must be set to match the contents of the LDAP attribute that is used to identify hosts. This attribute is defined in the mapping files as Host/UniqueIdAttribute.
- **Connect Timeout** (Connect Timeout): this indicates the number of seconds, after which attempts to connect to the LDAP server will time out. The default value is 1 second.

Note – Whenever you change the bootstrap and operational settings, the Configuration Agent must be restarted.

To restart the Configuration Agent on the Desktop, ensure that none of the related client applications are running, log in as root, and type the command **`/opt/apoc/bin/apocd restart`**.

Operational Settings

You can configure the operational settings of the Configuration Agent locally or remotely. To configure the settings locally, edit the `apocd.properties` file in the `/opt/apoc/lib` directory. To configure the settings remotely, use the Configuration Agent policy in the Configuration Manager. The following settings can be configured in the properties file:

- **DaemonPort**: port where the Configuration Agent listens for communication from its clients on the desktop
- **MaxClientThreads**: maximum number of client requests that can be simultaneously processed
- **MaxClientConnections**: maximum number of client connections
- **MaxRequestSize**: maximum size of client requests
- **DaemonChangeDetectionInterval**: interval in minutes between the change detection cycles for this list of configuration settings
- **ChangeDetectionInterval**: interval in minutes between the change detection cycles for the client configuration data
- **GarbageCollectionInterval**: interval in minutes between the garbage collection cycles in the local configuration database
- **TimeToLive**: interval in minutes that non-offline configuration data remains in the local database
- **LogLevel**: level of detail in the agent log files

The `DaemonPort` setting can only be modified locally and requires a restart of the agent for the changes to take effect. All other settings take effect at the next change detection cycle for the agent configuration. The logging level that is specified in `LogLevel` must be a value that is consistent with the Java Logger levels. In order of decreasing severity, these levels are: *SEVERE*, *WARNING*, *INFO*, *CONFIG*, *FINE*, *FINER* and *FINEST*.

Propagating Configuration Data Changes

You can use the `ChangeDetectionInterval` setting that is described in "Operational Settings" on page 32 to tune the propagation of remote configuration data changes to client side applications. The value that you provide for this setting is the maximum length of time in minutes that elapses before changes that are made remotely are reflected in the client applications. Smaller values for the `ChangeDetectionInterval` result in increased Configuration Agent and LDAP server activity. As a result, use caution when you adjust the value of the setting. For example, in an initial deployment phase, you can set this value to one minute so that you can easily test the impact of remote configuration on client applications. After you complete the testing, return this setting to the initial value.

Configuration Agent Port Information

The Configuration Agent uses two ports:

- The daemon port (default is 38900), which is used by the daemon to communicate with client applications.
- The daemon admin port (default is 38901), which is used by the daemon controller program, `apocdct1`, when communicating with the daemon.

Changing the daemon port:

To change the daemon port, you must modify the `DaemonPort` property in the daemon's `apocd.properties` file and the `apocd` entries in `/etc/services` and `/etc/inetd.conf`. Afterward, restart the daemon and reload `inetd`.

Changing the daemon admin port:

To change the daemon admin port, you must modify the `DaemonAdminPort` property in the daemon's `apocd.properties` file. Afterward, restart the daemon.

GConf Adapter

The GConf adapter is part of the `apoc-adapter-gconf` package. When you install the adapter from the corresponding RPM, the GConf data sources path in `/etc/gconf/2/path` is updated to include the Configuration Manager sources. A backup of the old path is stored in `/etc/gconf/2/path.apocBackup`. If the old path refers to custom data sources, you will need to update the path by merging the changes from the default path to the newly installed Manager path. The two data sources that are provided by the adapter are:

- "apoc:readonly:": provides access to non-protected settings from the policies. Insert this data source after the user settings and before the local defaults.
- "apoc:readonly:mandatory@": provides access to protected settings from the policies. Insert this data source after the local mandatory settings and before the user settings.

Mozilla Adapter

The Mozilla adapter is part of the `mozilla-apoc-integration` package. When you install the adapter from the corresponding RPM, the required files are added to an existing installation of Mozilla and are automatically registered.

StarOffice Adapter

The StarOffice adapter is included in a standard StarOffice installation and allows you to access the policy configuration data without any special modifications.

CHAPTER 6

Desktop Components - Solaris™

This chapter provides information specific to the Solaris operating environment.

To access the configuration data from the Configuration Manager, a desktop client requires the Sun Java™ Desktop System Configuration Agent. The Configuration Agent communicates with the remote configuration data repository and the adapters as well as integrates data into specific configuration systems. The configuration systems that are currently supported are GConf, Mozilla Preferences, and StarOffice Registry.

Configuration Agent

The Configuration Agent is part of the `SUNWapbas`, `SUNWapmsc`, and `SUNWapoc` packages. When you install the Solaris SVR4 package, the files that are required for this API are installed. You can install the packages manually or through the Java Desktop System installation. After installation, you must configure and enable the Configuration Agent on your system.

To access the remote configuration data, the Configuration Agent requires some minimal bootstrap information, such as the host name and port of the LDAP server. This information is maintained in a set of properties files, such as `policymgr.properties`, `apocd.properties`, `os.properties`. These files are stored locally in the `/etc/apoc` directory. You can manually edit these properties files, or you can use the configuration wizard for the Configuration Agent.

The configuration wizard offers a graphical user interface that guides you through the necessary settings of the Configuration Agent. For each page of the wizard, a corresponding help screen is available. You can startup the wizard as super user (root) by means of the `/usr/bin/apoc-config` script. A corresponding desktop menu entry is also available under Preferences/System Tools/Network Settings, or under system-settings:///Network Settings in the Nautilus file manager.

Note – The wizard can also be started without launching the graphical interface. For example, execute `/usr/bin/apoc-config -nodisplay` to start the wizard in console mode.

Bootstrap Information

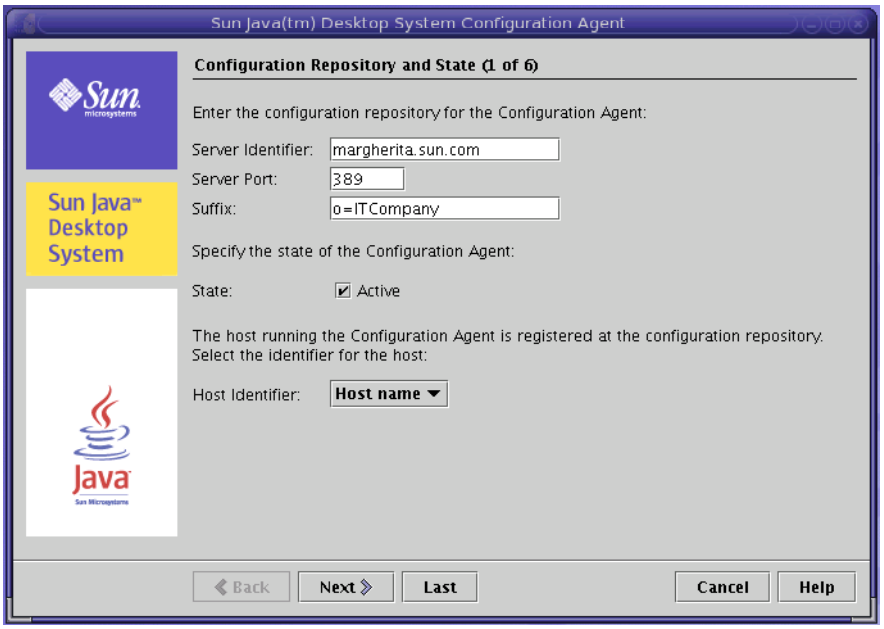

FIGURE 6–1 Configuration Agent, Configuration Repository

Note – Associated property file keys are indicated in parentheses, where appropriate.

- **Server Identifier** (Server): host name of the LDAP server.
- **Server Port** (Port): port number of the LDAP server.
- **Suffix** (BaseDn): base DN of the LDAP repository.
- **State**: The status of the Configuration Agent. The checkbox can be used to either activate or deactivate the Configuration Agent. To make use of the configuration repository, the Configuration Agent must be active. The activation automatically includes the necessary registration with `inetd`.

Note – To manually enable or disable the Configuration Agent, log in as **root** and type the command `/usr/lib/apocd enable` or `/usr/lib/apocd disable`, respectively.

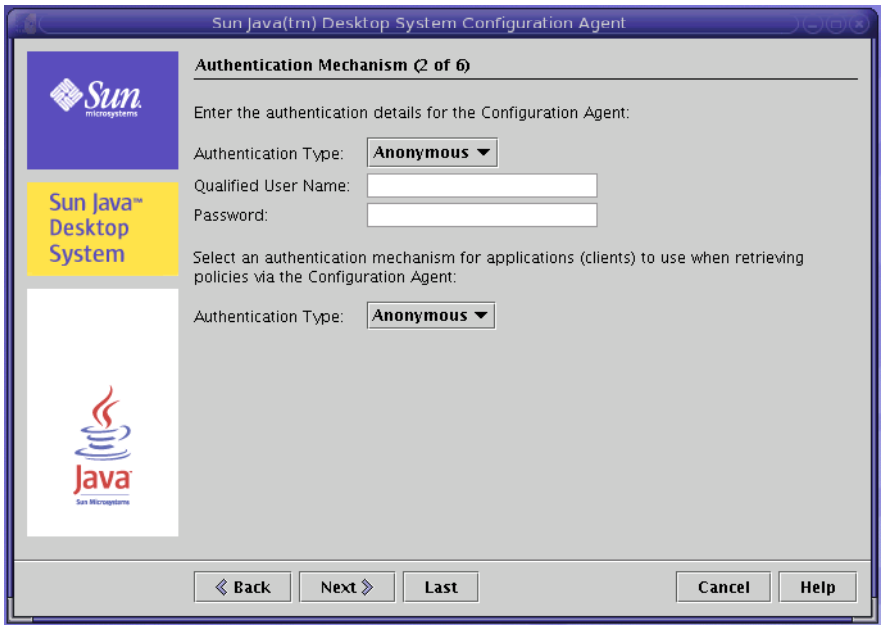

FIGURE 6–2 Configuration Agent, Authentication Mechanism

- **Host Identifier** (HostIdentifier): can be "HostName" or "IPAddress". The identifier must be set to match the contents of the LDAP attribute that is used to identify hosts. This attribute is defined in the mapping files as Host/UniqueIdAttribute.
- **Authentication Type** for the Configuration Agent: can be "Anonymous" or "Simple". If "Anonymous" is selected, the **Qualified User Name** and **Password** fields are automatically disabled.
- **Qualified User Name** (AuthDn): full DN of a user with read and search access rights on the repository.

Note – If anonymous access is enabled in the directory, this setting can be left blank.

- Password (Password): password of a registered LDAP user

Note – If anonymous access is enabled in the directory, this setting can be left blank.

- **Authentication Type** for applications (AuthType): can be “Anonymous” or “GSSAPI”, depending on how the LDAP server authenticates users.

Port Settings

The Configuration Agent uses two ports:

- **Agent Port** (DaemonPort): used by the agent to communicate with client applications (default is `38900`).
- **Administration Port** (DaemonAdminPort): used by the agent controller program, `apocdct1`, when communicating with the agent (default is `38901`).

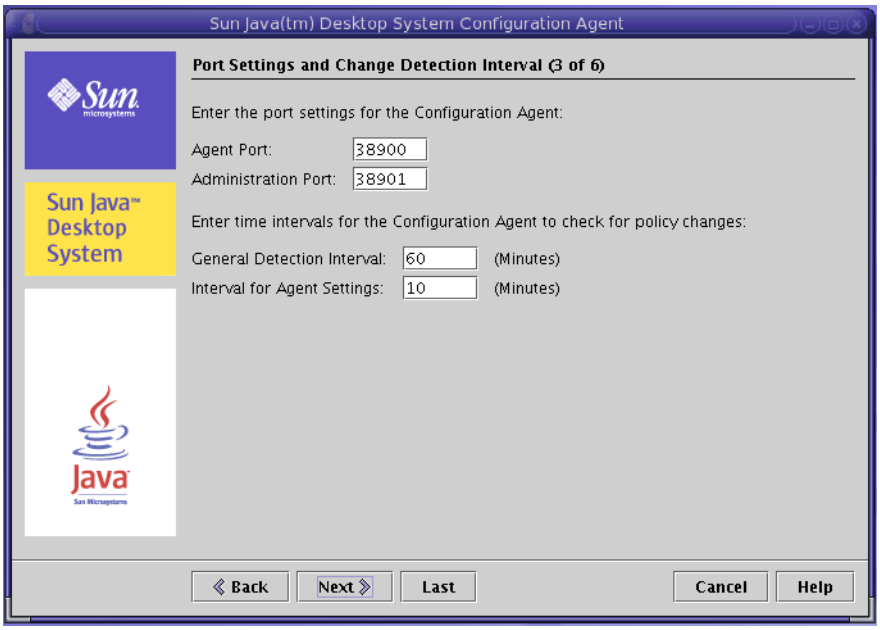

FIGURE 6–3 Configuration Agent, Port Settings

Change Detection Interval

The Configuration Agent periodically checks for any changes in the configuration data using the following two intervals:

- **General Detection Interval** (ChangeDetectionInterval): interval in minutes between the change detection cycles for the desktop application's (client's) configuration data.

  Note – Specifying `-1` turns off change detection.

- **Interval for Agent Settings** (DaemonChangeDetectionInterval): interval in minutes between the change detection cycles for the agent-specific configuration settings.

  Note – Specifying `-1` turns off change detection.

You can use the general detection interval to tune the propagation of remote configuration data changes to client side applications. The value provided for this setting is the maximum length of time in minutes that elapses before remotely made changes are reflected in the client applications.

Smaller values result in increased Configuration Agent and LDAP server activity. As a result, use caution when you adjust the value of the settings. For example, in an initial deployment phase, you can set the value to one minute so that you can easily test the impact of remote configuration on client applications. After you complete the testing, return this setting to the initial value.

Operational Settings

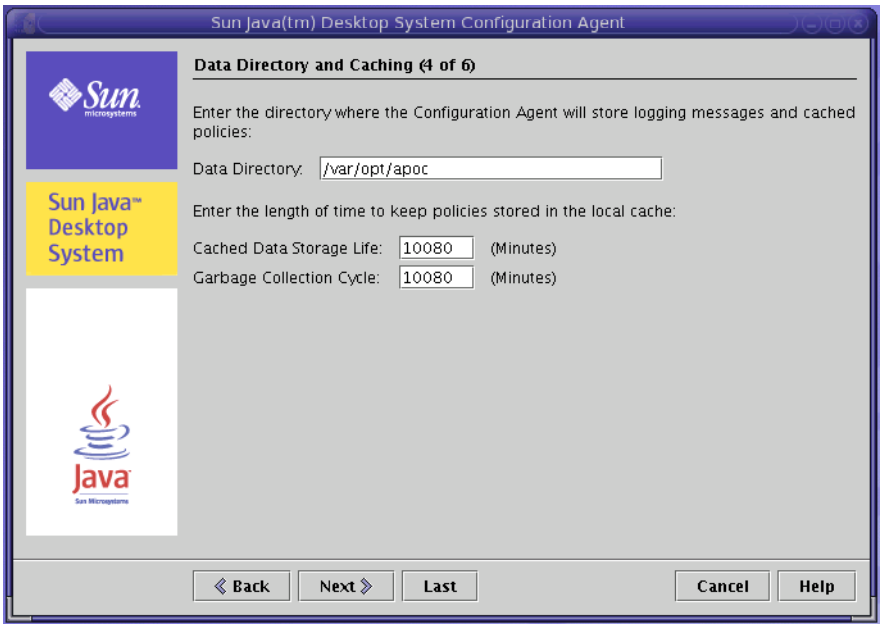

FIGURE 6–4 Configuration Agent, Data Directory

The following settings can be configured:

- **Data Directory** (DataDir): the directory used to store runtime data. The default is `/var/opt/apoc`.
- **Cached Data Storage Life** (TimeToLive): interval in minutes that non-offline configuration data remains in the local database.

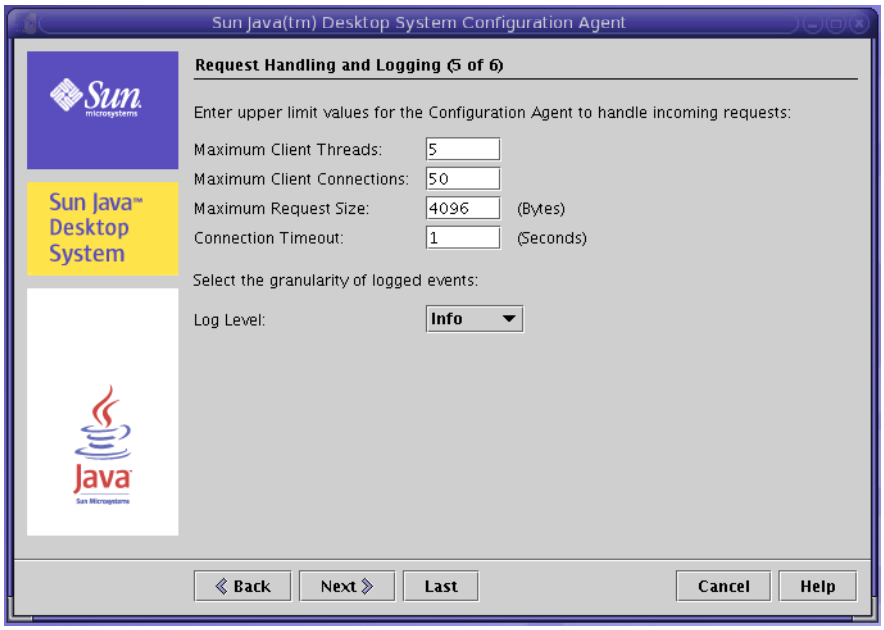

FIGURE 6–5 Configuration Agent, Request Handling and Logging

- **Garbage Collection Cycle** (GarbageCollectionInterval): interval in minutes between the garbage collection cycles in the local configuration database.
- **Maximum Client Threads** (MaxClientThreads): maximum number of client requests that can be processed simultaneously.
- **Maximum Client Connections** (MaxClientConnections): maximum number of client connections.
- **Maximum Request Size** (MaxRequestSize): maximum size of client requests.
- **Connection Timeout** (ConnectTimeout): denotes the allowed interval of the LDAP server to answer a connection request. The default is one second.
- **Log Level** (LogLevel): level of detail in the agent log files. The logging level is consistent with the Java Logger levels. In order of decreasing severity, these levels are:
  - *SEVERE*
  - *WARNING*
  - *INFO*
  - *CONFIG*
  - *FINE*
  - *FINER*
  - *FINEST*

Note – Most of the operational settings, with the exception of the **Data Directory** and **Connection Timeout** settings, can also be maintained centrally through corresponding policies stored in the LDAP server. If you want to use this feature, do not adapt the corresponding settings by means of the wizard. Instead, use the Configuration Agent policies within the Configuration Manager to centrally specify operational settings.

Applying Agent Settings

With the exception of "Data Directory" and "Connection Timeout", operational settings that have been stored on the LDAP server by means of the Configuration Manager take effect automatically at the next change detection cycle for the agent configuration (see DaemonChangeDetectionInterval).

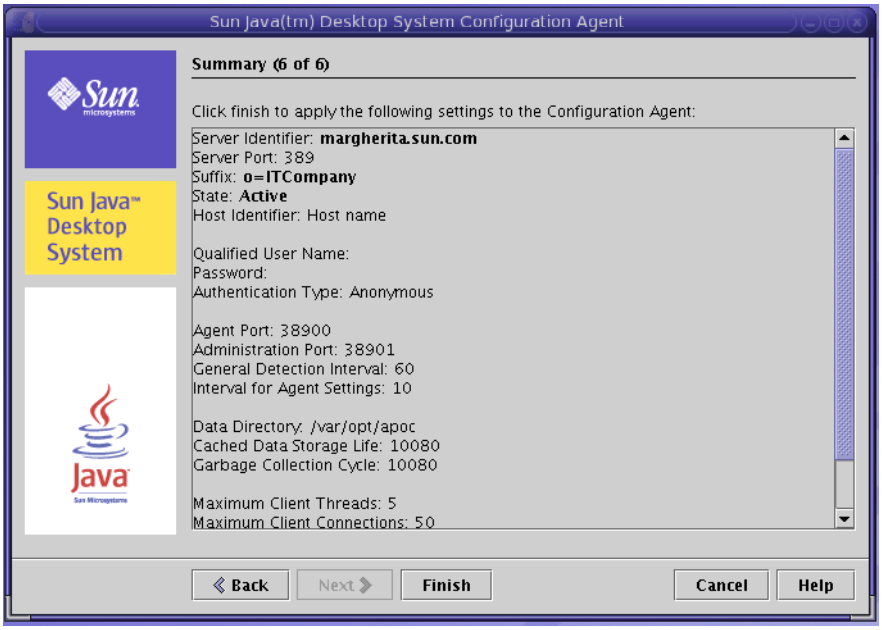

FIGURE 6–6 Configuration Agent, Summary Page

All other settings changed locally require a reload or restart of the Configuration Agent. The reload or restart is performed automatically if you use the configuration wizard.

Note – To manually restart the Configuration Agent, ensure that no related client applications are running, log in as root, and type the command `/usr/lib/apoc/apocd restart`.

GConf Adapter

The GConf adapter is part of the `SUNWapoc-adapter-gconf` package. When you install the adapter from the corresponding package, the GConf data sources path in `/etc/gconf/2/path` is updated to include the Configuration Manager sources. The two data sources that are provided by the adapter are:

- "apoc:readonly:": provides access to non-protected settings from the policies. Insert this data source after the user settings and before the local defaults.
- "apoc:readonly:mandatory@": provides access to protected settings from the policies. Insert this data source after the local mandatory settings and before the user settings.

Mozilla Adapter

The Mozilla adapter is part of the `SUNWmozapoc-adapter` package. When you install the adapter from the corresponding RPM, the required files are added to an existing installation of Mozilla and are automatically registered.

StarOffice Adapter

The StarOffice adapter is included in a standard StarOffice installation and allows you to access the policy configuration data without any special modifications.

APPENDIX **A**

Sun Web Console Packages

Known Issues

Security

Without a user's knowledge, a session can be left in an active state by certain user actions. For example, when a user closes a browser window, the user is not automatically logged out of the Sun Web Console. Instead, a user must explicitly log out of a session in the Sun Web Console before closing an application windows.

Setup Script Usage

Synopsis: `setup [-h] | [-n] | [-d <var>,<arch>[,client1,client2,...]] [-u [-f]]`

- `-h` = Prints usage statement
- `-n` = Does not start the server at the end of installation
- `-u` = Uninstalls the Sun Web Console
- `-f` = Uninstalls the Tomcat and Java 1.4 if theses packages were installed with the setup application. You can only use this parameter in conjunction with the `-u` parameter.

For a complete description of the available setup parameters, run **`setup -h`**.

To uninstall the Sun Web Console, run **`/usr/lib/webconsole/setup -u`**.

Note – Do not run this command if you are in the `/usr/lib/webconsole` directory or any of the related subdirectories, otherwise `pkgrm` fails.

Sun Web Console Packages

Solaris Packages

Package Name	Description
SUNWmctag	Sun Web Console UI Tag library
SUNWmcon	Sun Web Console
SUNWmcos	Common Solaris services for Sun Web Console
SUNWmcosx	Solaris release-specific services for Sun Web Console
SUNWmconr	Sun Web Console Root
SUNWjato	Sun One Application Framework runtime
SUNWtcatu	Tomcat

Linux RPMs

Package Name	Description
SUNWmctag	Sun Web Console UI Tag library
SUNWmcon	Sun Web Console
SUNWmcos	Common Linux services for Sun Web Console
SUNWmcosx	Linux release-specific services for Sun Web Console
SUNWmconr	Sun Web Console Root

Package Name	Description
SUNWjato	Sun One Application Framework runtime
tomcat4	Tomcat

APPENDIX **B**

Configuration Manager Packages

Configuration Manager Packages

Solaris Packages

Package Name	Description
SUNWapm	Configuration Manager
SUNWapmca	Configuration Agent templates
SUNWapmev	Evolution templates
SUNWapmgo	Gnome templates
SUNWapmmo	Mozilla templates
SUNWapmso	StarOffice templates

Linux RPMs

Package Name	Description
apoc-manager	Configuration Manager
apoc-agent-templates	Configuration Agent templates

Package Name	Description
apoc-evolution-templates	Evolution templates
apoc-gnome-templates	Gnome templates
apoc-mozilla-templates	Mozilla templates
apoc-staroffice-templates	StarOffice templates

APPENDIX **C**

Using OpenLDAP and Active Directory with the Configuration Manager

Using an OpenLDAP Server with the Configuration Manager

To use an OpenLDAP server as the repository for the Configuration Manager data, the schema of the server must be extended to feature the object classes and attributes used to store configuration data. A custom schema file named `apoc.schema` can be found in the `openldap` subdirectory of the Configuration Manager deployment tool provided in the Java Desktop System Management Tools CD.

This file must be copied in the `schema` subdirectory of the OpenLDAP configuration directory (`/etc/openldap`) and added to the OpenLDAP schema by including it in the `slapd.conf` file located in that directory. This is done by inserting a line that reads `include /etc/openldap/schema/apoc.schema` at the end of the sequence of schema includes that are present in that file. For more information on extending the schema of an OpenLDAP server, refer to the server's manual.

In order to prepare the OpenLDAP database to store configuration data, the deployment tool provided with the Configuration Manager must be used. The schema having already been extended by the previous step of the installation, only the `createServiceTree()` script needs to be run. The script must be started from the deployment tool directory as any user by the following command: `./createServiceTree`. The script prompts the user for the information about the OpenLDAP database as indicated in the deployment tool section of this document. A default mapping file using typical object classes and attributes featured in OpenLDAP is provided in the `openldap` subdirectory of the deployment tool. The file is called `OrganisationalMapping` and can be deployed by copying it over the file with the same name in the main deployment tool directory prior to launching `createServiceTree()`.

Note – The Configuration Manager Agent will try and connect to the OpenLDAP server anonymously by providing the DN of the user it requires data for, but no password. This mode of anonymous authentication can be disabled by default in some releases of OpenLDAP servers, in which case it must be enabled by adding a line reading `allow bind_anon_cred` in the common server parameters defined in the file `slapd.conf` located in the OpenLDAP configuration directory (`/etc/openldap`). For more information on that parameter, refer to the server's manual.

Using an Active Directory Server with the Configuration Manager

To use an Active Directory server as a repository for the Configuration Manager data, the schema of the server must be extended to feature the object classes and attributes used to store configuration data. A schema extension file named `apoc-ad.ldf` can be found in the `ad` subdirectory of the Configuration Manager deployment tool provided on the Management Tools CD. Refer to the deployment tool section for more information.

The `apoc-ad.ldf` file must be imported in the Active Directory schema using the following steps:

1. Enable schema extensions. Refer to Active Directory documentation or more information how to perform that operation.
2. Execute the following from the command prompt: **`ldifde -i -c "DC=Sun,DC=COM" <`***`BaseDN`***`> -f apoc-ad-registry.ldf`**.

Note – Replace <*BaseDN*> with the Active Directory base DN.

In order to prepare the Active Directory server to store configuration data, the deployment tool must be used. The schema having already been extended by the previous step of the installation, only the `createServiceTree` script needs to be run. It must be started from the deployment tool directory as any user by the following: `./createServiceTree`. The script prompts the user for the information about the Active Directory database. A default mapping file using typical object classes and attributes featured in Active Directory is provided in the ad subdirectory of the deployment tool directory. This file is called `OrganisationalMapping` and can be deployed by copying it over the file with the same name in the main deployment tool directory prior to launching `createServiceTree`.

From that point, the Active Directory server can be used with the Configuration Manager. When installing the Configuration Manager, provide the full DN and password of a user with read rights to the tree. This can be a user that is not able to use Active Directory for any other purpose. Refer to Active Directory documentation for more information on how to setup such a user. In addition, the domain name for the Active Directory must be known to the machine that is running the Configuration Manager. You can do this by adding a line mapping the IP address of the Active Directory server with its domain name to the `/etc/hosts` file of that machine.

In order to retrieve the configuration data from a Java Desktop System host, the domain name of the Active Directory must also be known to that host. Authentication of the Java Desktop System user can be done in two ways: anonymously and using GSSAPI.

- To authenticate using anonymous connections, the Active Directory server must be configured to grant read rights to everyone. Refer to Active Directory documentation for more information on how to perform that operation.
- To authenticate using GSSAPI, the file `/etc/krb5.conf`, which specifies the Kerberos parameters, must be modified to define the Active Directory realm and point to the Active Directory server as its Key Distribution Center (KDC). It must also specify, as the default encryption types, the DES types supported by Active Directory, namely `des-cbc-crc` and `des-cbc-md5`. Refer to the Kerberos documentation for more information on how to perform that operation. Before accessing the configuration data, valid credentials for the user who is logged in the Java Desktop System must be obtained. This can be performed manually by running the `kinit` command and by providing the user password defined in Active Directory. Other schemes may generate these credentials automatically at login. Refer to the Java Desktop System documentation for further information.