# Sun Desktop Manager 1.0 Installation Guide

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

Part No: 819–2725

060518@14986

# Contents

# Preface

This document provides a description of the installation and configuration steps that are required to deploy Sun™ Desktop Manager 1.0.

## Overview

The Sun Desktop Manager is aimed at providing a central configuration for desktop hosts. Settings can be assigned to various elements of an organization or domain structure, enabling the administrator to efficiently manage groups of users or hosts.

## How This Book Is Organized

Chapter 1 provides a brief overview of the Sun Desktop Manager.

Chapter 2 discusses Sun Desktop Manager server-side installation.

Chapter 3 provides information on installing the Java Desktop System Configuration Agent.

Chapter 4 provides installation information about the Java Web Console.

Appendix A contains configuration parameter information.

Appendix B discusses using OpenLDAP and Active Directory with the Desktop Manager.

Appendix C provides information about organizational mapping.

## Related Books

- *Sun Desktop Manager 1.0 Administration Guide*
- *Sun Desktop Manager 1.0 Developer Guide*

# Documentation, Support, and Training

| Sun Function | URL | Description |
|---|---|---|
| Documentation | http://www.sun.com/documentation/ | Download PDF and HTML documents, and order printed documents |
| Support and Training | http://sunsolve.sun.com | Obtain technical support, download patches, and learn about Sun courses |

◆ ◆ ◆ CHAPTER 1

# Overview and Concepts

This document provides a description of the installation and configuration steps that are required to deploy the Sun™ Desktop Manager 1.0. For a more comprehensive overview of the Sun Desktop Manager, see the *Sun Desktop Manager 1.0 Administration Guide*.

## Overview of the Sun Desktop Manager

The Sun Desktop Manager provides a central configuration for desktop hosts. Settings can be assigned to various elements of an organization or domain structure, enabling the administrator to efficiently manage groups of users or hosts.

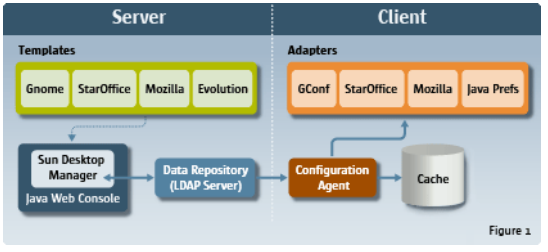


**FIGURE 1–1** Desktop Manager Architecture

The main components of the Desktop Manager are:

- Configuration Repositories
- Management Tools
- Desktop Manager Templates
- Configuration Agent
- Configuration Adapters

Configuration data is centrally stored in configuration repositories. The configuration data is managed (created/deleted/modified/assigned/unassigned) using the management tools, which

consist of a web-based Desktop Manager graphical user interface, and a command line interface. The templates are used by the web-based management tool to render the configuration data in the web browser.

The Configuration Agent retrieves configuration data from the configuration repository on behalf of the user applications. The Agent caches the information retrieved from the central configuration repository.

The management tools are completely de-coupled from the Agent, which means that they only operate on the configuration repository.

The user applications (using the Configuration Adapters) query the configuration data through the Configuration Agent.

The product directly supports the retrieval and application of settings for the following configuration systems:

- GConf. Gnome configuration framework
- StarOffice Registry
- Mozilla Preferences
- Java Preferences

◆ ◆ ◆ CHAPTER 2

# Management Application Installation

This chapter provides instructions about how to install the server-side components of the Sun Desktop Manager.

## Sun Desktop Manager

The Desktop Manager provides a web-based administration tool that runs on the Java Web Console. This 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 Desktop Manager uses several configuration templates to display settings that are specific to different desktop applications such as Gnome, Mozilla, StarOffice, and Evolution.

### ▼ Installation

**Before You Begin**

The Desktop Manager requires an installed Java Web Console version 2.2.5 or higher. Please ensure that a valid version is installed on your system. To determine whether you have a valid version, become the super-user (root) and execute:

```
# smcwebserver status
```

---

**Note –** The Java Web Console 2.2.4 is part of the Solaris™ 10 operating system, however the Desktop Manager requires version 2.2.5 or higher. A copy of version 2.2.5 is provided in the Desktop Manager archive in the server/console directory. It can be installed by running `./setup` in that directory.

---

If the Java Web Console is not installed on your system, or the installed version is not valid for the Desktop Manager, refer to the instructions in Chapter 4 to first install or update the Java Web Console. Afterwards return to this chapter to continue the Desktop Manager installation.

**1 Download the Desktop Manager zip archive and extract the content in a temporary directory**

```
# unzip SunDesktopMgr-1.0.zip
```

2 **Become the super-user (root) and execute the setup script via**

```
# cd SunDesktopMgr-1.0/<platform>/server/manager
# ./setup
```

3 **View the setup script's output for any errors.**

If the installation was performed successfully, the setup script will automatically restart the Java Web Console and you can access the Desktop Manager with a Web browser.

## ▼ Operation

1 **Type the following URL in your browser:**

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

2 **In the login screen, type the user name and password of an existing Unix user.**

The Java Web Console opens.

3 **In the console application launch page, click on the Desktop Manager link.**

- **If you want to skip the console application launch page and go straight to the Desktop Manager, enter the following URL in your browser:**

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

## ▼ Removing the Desktop Manager

◗ **To remove the Desktop Manager from the Java Web Console, change to the temporary directory that you created for the installation, become the super-user (root) and execute**

```
# cd server/manager
# ./setup -u
```

# Migration Issues

The Desktop Manager is compatible with previous versions of the Java Desktop System Configuration Manager (releases 1.0 and 1.1). However, there are a few differences that you should be aware of.

In the previous Configuration Manager versions, all profile data was stored at one specified LDAP server. This LDAP server was configured as part of the overall Configuration Manager installation procedure. This also included the configuration of an LDAP login module that encapsulates the authentication against the LDAP server.

For the Desktop Manager, all the necessary configuration steps are now wizard-based and it is no longer necessary to perform any kind of configuration during the installation. The Desktop Manager now also includes support for multiple configuration repositories. Therefore, you can manage policy data stored on several different LDAP servers, file-based repositories, and so on. The configuration of a specific LDAP login module is no longer necessary.

There have been no changes in the LDAP schemata between the different versions. If you have already configured an LDAP server for a previous Configuration Manager version, there are no changes required when you switch to the Desktop Manager. Therefore, you can take advantage of the Desktop Manager without updating the client (Java Desktop System Configuration Manager1.1. Agent) or the LDAP side.

---

**Note –** Before you install Desktop Manager, you should first remove any previous Configuration Manager or Desktop Manager installations on your system. To remove previous installations, execute (as super-user):

```
# cd server/manager
# ./setup -u
```

---

After you have installed the Desktop Manager, you can then create a configuration repository that points to your existing LDAP server:

## ▼ Creating a configuration repository

1. **Type the following URL in your browser**

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

2. **In the login screen type the user name and password of an existing Unix user.**

   The Java Web Console opens.

3. **In the console application launch page, click on the Sun Desktop Manager 1.0 link.**

4. **Click the New button to start the configuration repository wizard.**

   The wizard guides you through the necessary steps for configuring an LDAP-based configuration repository.

---

**Caution –** The wizard automatically offers to migrate the existing policies data to a new 2.0 format. This migration is optional and can be used mainly to improve the performance of the newer Sun Desktop Manager 1.0 agents. As long as you still need to support the Java Desktop System Configuration Manager 1.1 agents in your environment, do NOT perform this migration.

---

# Desktop Manager Troubleshooting

## Cannot Install

Symptom: at the end of the installation of the Java Web Console a message states that it cannot start because there are no registered applications.

Possible causes: no applications have been installed, including the Desktop Manager. .

Solution: install the Desktop Manager and then start the Java Web Console.

## Connection Refused

Symptom: you try to open an appropriate URL, for example `http://<hostname>.<domainname>:6789`, but you receive a message that the connection is refused.

Possible causes: the Java Web Console is not running on the server.

Solution: to start the Java Web Console, become superuser and execute the following commands:

```
#smcwebserver status
#smcwebserver start
```

## Cannot Log In

Symptom: the user/password combination is rejected on the login page of the Java Web Console.

Possible causes: the corresponding UNIX user account does not exist.

Solution: Check that a corresponding UNIX user name and password is configured on your system. If necessary, create a local UNIX user account for your tests.

## No Desktop Manager Link

Symptom: the Java Web Console application list page does not display the Sun Desktop Manager link.

Possible causes: the Desktop Manager module is not installed.

Solution: To check if the Desktop Manager is installed in the Java Web Console, become superuser and execute the following command:

```
# smreg list -a
```

If the list does not contain the `com.sun.apoc.manager_<version>` application, you need to reinstall the Desktop Manager.

## Null Pointer Exception, Tomcat/Java error or Blank Page

Symptom: you start the Desktop Manager but only a blank page or error messages appear.

Possible causes: if the error mentions `NoClassDefFoundError: sun/tools/javac/Main`, then the Java Web Console is using the wrong Java version.

Solution: the current Java Web Console Java environment can be checked by executing `# smreg list -p` and looking at the `java.home` property. This property must point to a valid Java home, and that home must be a JDK. If this value is incorrectly set, you need to execute the following command:

```
# smreg add -p java.home=<JAVA_HOME>
```

---

**Note –** <*JAVA_HOME*> must point to a valid installation, for example, where `javac` can be found in the `bin` subdirectory.

---

You then need to restart the Java Web Console with the following command:

```
# smcwebserver restart
```

## Cannot Connect to an SSL LDAP Server

Symptom: after providing the LDAP server details in the repository creation wizard, including checking the Use SSL box, pressing Next yields a message box stating the server cannot be contacted.

Possible causes: the improper port number has been provided, the LDAP server is not configured to listen to connections using SSL on that port, or the proper certificates are missing in the Java Web Console key store.

Solution: first check that the LDAP server is configured to listen to SSL connections requests on the port specified in the wizard. If this is correct, ensure that either the Certification Authority or the LDAP server certificate is present in the Java Web Console key store, which is located in `/etc/opt/webconsole/keystore`. The certificate can be added with the command `keytool -import -file <certificate file> -keystore /etc/opt/webconsole/keystore`. The default password for that key store is **`changeit`**. The Java Web Console needs to be restarted for that change to be visible by the Desktop Manager using the command `smcwebserver restart`.

## Cannot Write to Directory

Symptom: while creating a file-based or hybrid backend, a “Cannot write to directory!” error appears.

Possible causes: noaccess user does not have the correct permissions.

Solution: assign write permissions to the noaccess user.

◆ ◆ ◆ CHAPTER 3

# Client Components

To access the configuration data from the Desktop Manager, a desktop client requires the 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, Java Preferences, Mozilla Preferences, and StarOffice Registry.

A version of the Configuration Agent is provided with the Solaris 10 operating system. However, the Desktop Manager requires a newer version of that tool. That newer version is installed as part of the setup of the Desktop Manager client components and associated patches.

To install the Desktop Manager client components:

1. Download the Desktop Manager zip archive and extract the content in a temporary directory.

   ```
   # unzip SunDesktopMgr-1.0.zip
   ```

2. Install the recommended patches.

   These patches are provided in the `SunDesktopMgr-1.0/<platform>/client/Patches` directory. Follow the installation instructions provided with each patch.

3. Become the super-user (root) and execute the setup script through:

   ```
   # cd SunDesktopMgr-1.0/<platform>/client
   # ./setup
   ```

## Configuration Agent

The Configuration Agent is part of a number of different packages, which are listed in the following table:

| Solaris Package Name | Description |
|---|---|
| SUNWapbas | Configuration Shared libraries |
| SUNWapmsc | Configuration Agent miscellaneous files |
| SUNWapoc | Configuration Agent |
| SUNWapdc | Configuration Agent wizard |

When you install these packages, 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.

**Note –** Configuration Agent packages are installed as part of the Solaris with Java Desktop System installation; however, the Desktop Manager patches these files during installation to provide the proper level of functionality.

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 (see Appendix A), 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 start the wizard as super user (root) by means of the `/usr/bin/apoc-config` script.

**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

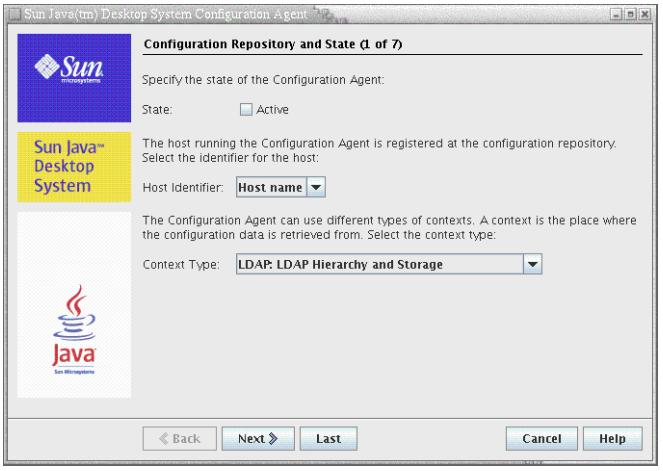


**FIGURE 3–1** Configuration Agent, Configuration Repository

---

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

---

- State: The status of the Configuration Agent. The check box 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 the service management facility ( smf(5) ) on Solaris.
- Host Identifier (HostIdentifierType): can be ”HostName” or “IPAddress”. When searching for host-specific policy data, the Configuration Agent will identify the current host either by host name or IP address. Choose the correct value based on how your host is identified in the selected Context Type.
- Context Type: Use this setting to indicate to the Configuration Agent whether your organizational hierarchy and configuration data are defined in LDAP or File-based storage or a mixture of both.

---

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

---

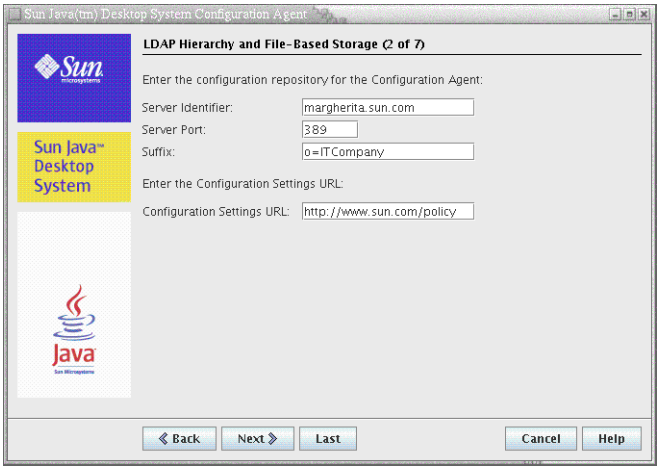


**FIGURE 3–2** Configuration Agent, LDAP Hierarchy and File-Based Storage

---

**Note –** The screen in Figure 3–2 varies, depending on the Context Type chosen on the previous screen. Server Identifier, Server Port and Suffix are required if an LDAP or Hybrid context type is chosen. Configuration Settings URL is required if a File-based or Hybrid context type is chosen.

---

- Server Identifier: host name of the LDAP server.
- Server Port: port number of the LDAP server.
- Suffix: base DN of the LDAP repository.
- Configuration Settings URL: URL specifying the location of the file based repository.

  A list of URLs can be used to specify fallback repositories in case the connection to the first one does not succeed. The list can be composed of one or more white space-separated URLs, each URL being of the form `file://<filepath>`, `http://<host>:<port>/<filepath>`, or `https://<host>:<port>/<filepath>`. See Appendix A for more information.

  ---

  **Note –** The Agent tries to access the LDAP server using a SSL connection first. If this fails the Agent tries a plain SSL connection.

  For the SSL connection to succeed, the proper certificate must be present in the Java runtime environment key store. That key store is located for a standard JRE in `<installation directory>/lib/security/cacerts`, and for a standard JDK in `<installation directory>/jre/lib/security/cacerts`. Either the Certification Authority or the LDAP server certificate must be added to that store, using the command `keytool -import -file <certificate file> -keystore <cacerts file location>`. The default password for that key store is **`changeit`**.

  ---

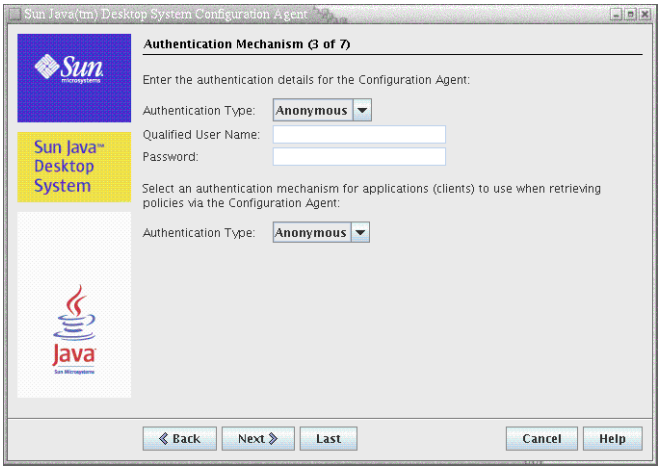


**FIGURE 3–3** Configuration Agent, Authentication Mechanism

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

---

**Note –** If anonymous access is enabled in the directory, the Qualified User Name and the Password settings can be left blank.

---

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

---

**Note –** For more information, see “Data Access/User Authentication” on page 25.

---

# 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, `apocd`, when communicating with the agent (default is **`38901`**).

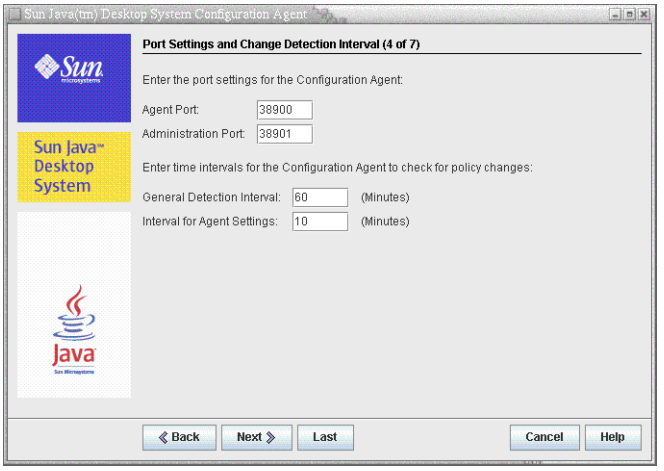


**FIGURE 3–4** Configuration Agent, Port Settings

# 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 test the impact of remote configuration on client applications. After you complete the testing, return this setting to the initial value.

# Operational Settings

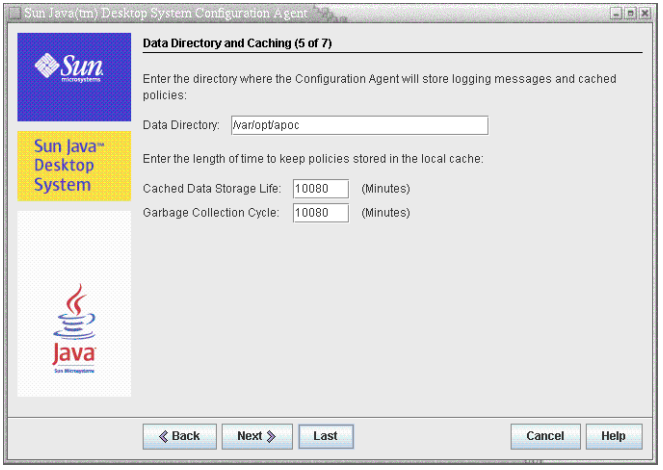


**FIGURE 3–5** Configuration Agent, Data Directory

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.

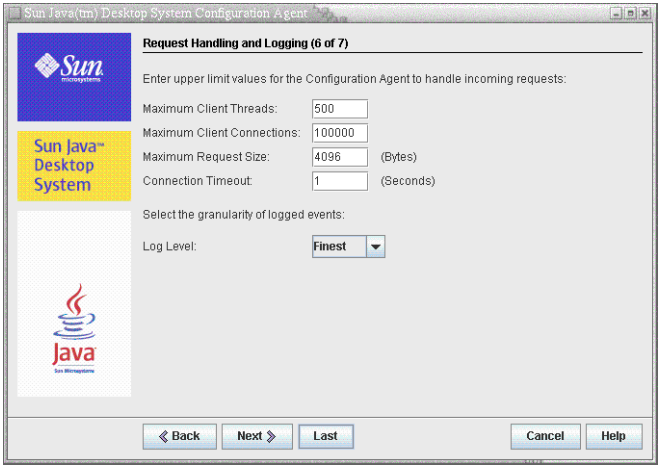


**FIGURE 3–6** Configuration Agent, Request Handling and Logging

- 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 as follows:
  - *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 Desktop 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 Desktop Manager take effect automatically at the next change detection cycle for the agent configuration (see DaemonChangeDetectionInterval).

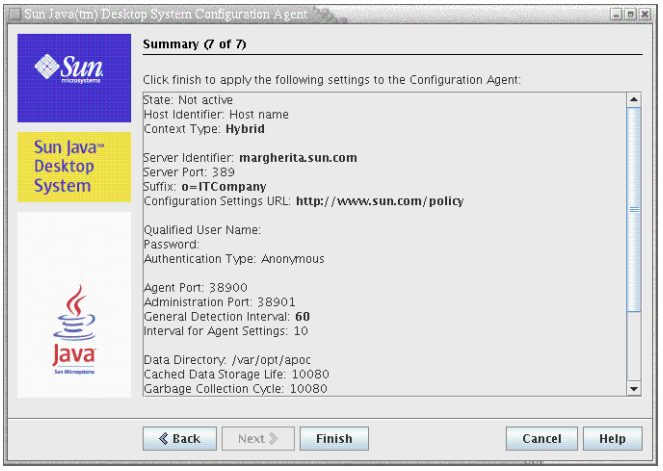


**FIGURE 3–7** Configuration Agent, Summary Page

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`.

---

# Additional Agent Settings

---

**Note –** The following settings are not available in the configuration wizard.

---

- Application of local policy (ApplyLocalPolicy): use this setting to indicate whether or not policy data available on the local host should be made available to client applications. A value of "true" indicates that local policy data should be made available. A value of "false" indicates that local policy data should not be made available. For more details, see "Using Local Policy" on page 23.

# Using Local Policy

You may configure the Configuration Agent to apply configuration settings from locally deployed policy in addition to or as an alternative to any globally available policy. Use the following steps to deploy any such local policy:

### ▼ Deploying a Local Policy

1 **Using the Desktop Manager create a profile with the required policy settings.**

2 **Using the Desktop Manager export the profile to a zip file.**

3 **On your client host, create the directory**
`${DataDir}/Policies/profiles/PROFILE_REPOSITORY_default`**, if it does not already exist.**

`${DataDir}` corresponds to the value of the Configuration Agent's Data Directory which is `/var/opt/apoc` by default.

4 **Copy the previously exported zip file to**
`${DataDir}/Policies/profiles/PROFILE_REPOSITORY_default`.

5 **Ensure that the Configuration Agent is configured to apply available local policies (see "Additional Agent Settings" on page 23 for more details ).**

---

**Note –** If you change the Configuration Agent's "ApplyLocalPolicy" setting, you should reload the Configuration Agent by logging in as `root` and typing the command `/usr/lib/apoc/apocd reload`.

Any local policy deployed in this manner will be made available to clients during the next Configuration Agent change detection cycle.

---

# Automatic Restart of the Configuration Agent

In the event of a failure, the Configuration Agent will be automatically restarted. The service management facility ( smf(5) ) is responsible for making this decision. If the service management facility decides that a restart is inappropriate (for example, if too many failures have already occurred), the Configuration Agent is placed in maintenance mode.

In the event that the Configuration Agent is not restarted, you should disable the Agent temporarily by logging in as root and executing the command `/usr/lib/apoc/apocd disable`, rectify any problems causing the Agent to fail and re-enable the Agent by executing the command `/usr/lib/apoc/apocd enable`.

# 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 element 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 element in the LDAP server through the `Host/UniqueIdAttribute` setting of the organizational mapping file. For more information about organizational mapping, see Appendix C.

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 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
```

If you integrate Kerberos with user login in this way, you should enable the screensaver's Kerberos support. For example, by using the following `/etc/pam.d/xscreensaver` file:

```
auth required pamkrb5.so use_first_pass missing_keytab_ok
ccache=SAFE putenv_direct
```

# Adapters

The application adapters are extensions of the configuration systems supported by the Desktop Manager. The adapters allow the various applications to take the central configuration data into account, depending on configuration systems. The supported configuration systems are:

- GConf: Gnome configuration system, used by the desktop and most Gnome applications, such as Evolution
- StarOfficeRegistry: configuration system used by StarOffice and OpenOffice.org
- Mozilla Preferences: configuration system used by Mozilla
- Java Preferences: configuration API offered to Java applications

A desktop definition adapter is also provided, which adds desktop launchers, menu items, and startup programs to the user desktop.

# GConf Adapter

The GConf adapter is part of the `SUNWapoc-adapter-gconf` package for Solaris. When you install the adapter from the corresponding packageAdapter, the GConf data sources path in `/etc/gconf/2/path` is updated to include the Desktop 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.

## GConf Adapter Configuration

The GConf adapter is configured as part of its installation, however its operation is dependent on the presence in the GConf path file (`/etc/GConf/2/path`) of two data sources representing the mandatory central settings and the default settings. While this path file contains the proper information for GConf to take the central settings into account as expected after the installation of the system, administrators should ensure, that the data sources prefixed with "apoc" are still present in the file, should they need to modify that path to include additional custom data sources. You should also ensure that the data sources are located between local mandatory settings and user settings for the data source representing the mandatory central settings, and between the user settings and local default settings for the data source representing the default central settings.

# Java Preferences Adapter

The Java Preferences adapter is part of the `SUNWapcj` package for Solaris.

## Java Preferences Adapter Configuration

The Java Preferences adapter is provided as an implementation of the Preferences API that must be used as a wrapper to another existing implementation (such as the default file based system provided with the JRE). To enable the use of central configuration in a Java application that makes use of the Preferences API, a startup script for that application must be written, using the `/usr/lib/apoc/apocjlaunch` script as a helper. This script needs to define a few environment variables and then include the `apocjlaunch` script at its end (which starts the Java application with the necessary environment). The environment variables that must be set are:

- `JAVA`: contains the path to the Java runtime executable
- `APPLICATION`: contains the trailing part of the regular Java runtime invocation for that applications. For example,*classname [arguments]* for a single class startup, or `-jar` *jarname [arguments]* for a jar archive startup.

The optional additional environment variables that can be set are:

- `CLASSPATH`: colon-separated list of jars or class files that need to be part of the application class path

- `DEFINES`: string containing the define statements that need to be part of the application startup
- `PREFFACTORY`: class name of the factory in the underlying Preferences API implementation that the application needs to use

# Mozilla Adapter

The Mozilla adapter is part of the `SUNWmozapoc-adapter` package on Solaris.

## Mozilla Adapter Configuration

The Mozilla adapter is setup as part of the installation of this product, and does not need any additional configuration.

# StarOffice Adapter

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

## StarOffice Adapter Configuration

The StarOffice adapter is setup as part of the installation of this product, and does not need any additional configuration.

# Desktop Definition Adapter

The Desktop Definition adapter consists of the following packages:

| Package Name | Description |
|---|---|
| SUNWapleg | configuration access binaries |
| SUNWardsa | desktop definition adapter |
| SUNWardsa-misc | system integration for adapter |

These packages are installed when the Desktop Manager client components are installed, and do not require any additional setup.

## Desktop Definition Adapter Configuration

The Desktop Definition adapter is configured by the setup process to be used whenever a user logs in, and does not require any additional setup.

## Removing Adapters

The Mozilla and StarOffice adapters are removed when these products are removed. The GConf, Java Preferences and Desktop Definition adapters can be removed using the appropriate package management system tools by removing the packages mentioned in the Installation section.

Upon removing the Java Preferences adapter, the startup scripts written for the purpose of launching Java applications using the Preferences API should not be used anymore. The Java invocation in them fails, since some needed classes will no longer be available.

## Adapter Troubleshooting

Most of the problems that can result in not seeing the central configuration data in the corresponding applications are likely to come from the Configuration Agent, since it is the common mechanism used by all adapters to retrieve the data.

If a central configuration change does not seem to take effect for a given setting (or group thereof), a possible explanation is that the user has explicitly set a value for that setting in the application (usually by using the Options or Preferences dialogs of the product). In that case, unless the central settings are defined as protected, which means that value is forced by the administrator and the user is not allowed to modify it, the user preference will take precedence over the values set using the Desktop Manager.

# Configuration Agent Troubleshooting

This section answers some of the questions you may have regarding the nature and workings of the Configuration Agent along with some tips for troubleshooting problems with the Agent.

## Questions and Answers

### What is the Configuration Agent, and how does it work?

The Configuration Agent is a policy caching and delivery application. It was designed and built to ensure that desktop client applications could be centrally configured without significant impact to the performance of these applications and the hosts they run on. This is achieved by:

- Caching any downloaded policy in a locally available cache for future client use
- Sharing any expensive resources (for example, connections to an LDAP server where policy is hosted ) that can and should be shared

The typical scenario where interaction occurs between client applications and the Configuration Agent is extremely simple and can be described as follows:

1. A user launches one of the relevant desktop client applications ( gconfd, Mozilla or StarOffice )
2. The client application connects to the Configuration Agent
3. The client application requests policy data that it needs from the Configuration Agent
4. The Configuration Agent searches its cache for the requested policy data
5. If the policy data is not found in the cache, the Configuration Agent downloads the required data from a pre configured policy repository and stores it in the cache
6. The policy data is sent to the requesting client application
7. The Configuration Agent monitors the policy repository for any modifications to policy data
8. If a modification is detected, the Configuration Agent refreshes its cache so that it is up to date and informs the client application of the modification.

## How do I get and install the Configuration Agent?

The Configuration Agent is available and installed by default with Solaris 10

## I've just installed Solaris 10. What do I do next?

The Configuration Agent is disabled and not configured by default. To use the Configuration Agent, you must at least minimally configure it and enable it. Once you've completed these steps, your desktop client applications will automatically start to use any policy you provide when they are next started.

## I want to configure the Configuration Agent. How do I do that?

To configure the Configuration Agent correctly, use the Configuration Agent Wizard. You can start the wizard by executing (as root) the command `/usr/bin/apoc-config`. The wizard guides you through the steps required to correctly configure the agent. In most cases, the only piece of information you absolutely need to complete the wizard is the location of your policy repository.

It is also possible to configure the Configuration Agent by manually editing its configuration files. This is not recommended as it is far easier to incorrectly configure the Agent this way. Additionally, the Configuration Agent Wizard contains additional logic which determines whether or not a particular configuration change requires a restart or a reload of the Agent.

## I want to enable the Configuration Agent. How do I do that?

There are three possible mechanisms you can use to enable the Agent:

1. Using the Configuration Agent Wizard ( `/usr/bin/apoc-config`), set the Agent State to Active.
2. Using the Configuration Agent controller program ( `/usr/lib/apoc/apocd`), execute the following as root:

   ```
   /usr/lib/apoc/apocd enable
   ```

3. Using smf(5), execute the following as superuser:

```
/usr/sbin/svcadm enable svc:/network/apocd/udp
```

## I've configured and enabled the Configuration Agent. How do I know it's working?

The easiest way to know if the Configuration Agent is correctly configured and working is to create a policy with the Desktop Manager, assign the policy to a user, login to the desktop machine as that user and verify that the policy settings you made are being used. There are many policy settings which are easily detectable in a desktop session, for example, background and theme.

## What is the significance of enabling the Configuration Agent?

The Configuration Agent is an smf(5) compliant service and, as such, the notion of enabling the Agent comes from smf(5). Once the Agent is enabled, the Agent is ready to provide service. The following actions occur when you enable the Agent:

- The Agent starts
- Any desktop client applications started after the Agent is enabled can retrieve policy data
- The Agent automatically restarts during system boot

## How do I know if the Configuration Agent is enabled?

You can use one of the following methods to determine if the Configuration Agent is enabled:

- Use the controller program for the Configuration Agent. Become super user and type the following command:

  ```
  /usr/lib/apoc/apocd is-enabled
  ```

  If the Agent is enabled, the controller program returns the following message:

  ```
  Checking Configuration Agent enabled status ... Enabled
  ```

  Otherwise, the controller program returns the following message:

  ```
  Checking Configuration Agent enabled status ... Not enabled
  ```

- Use smf(5) to execute the following command:

  ```
  /usr/bin/svcs svc:/network/apocd/udp:default
  ```

  If the Agent is enabled, svcs returns the following message:

  ```
  STATE          STIME    FMRI
  online          8:36:04 svc:/network/apocd/udp:default
  ```

  If the Agent is disabled, svcs returns the following message:

```
STATE          STIME    FMRI
disabled       15:58:34 svc:/network/apocd/udp:default
```

If the Agent is in maintenance mode, svcs returns the following message:

```
STATE          STIME    FMRI
maintenance     8:38:42 svc:/network/apocd/udp:default
```

## How do I know if the Configuration Agent is running?

Use one of the following methods to determine if the Configuration Agent is running:

- Become superuser and execute the Configuration Agent controller program:

  ```
  /usr/lib/apoc/apocd status
  ```

  If the Agent is enabled, the controller program returns the following message:

  ```
  Checking Configuration Agent status ... Running
  ```

  Otherwise, the controller program returns the following message:

  ```
  Checking Configuration Agent status ... Not running
  ```

- Execute the following command:

  ```
  /usr/bin/svcs svc:/network/apocd/udp:default
  ```

  If the Agent is running, svcs returns the following message:

  ```
  STATE          STIME    FMRI
  online          8:36:04 svc:/network/apocd/udp:default
  ```

  If the Agent is not running, svcs returns the following message:

  ```
  STATE          STIME    FMRI
  disabled       15:58:34 svc:/network/apocd/udp:default
  ```

  If the Agent is in maintenance mode, svcs returns the following message:

  ```
  STATE          STIME    FMRI
  maintenance     8:38:42 svc:/network/apocd/udp:default
  ```

- Execute the following command:

  ```
  ps -ef | grep apoc
  ```

  If the Configuration Agent is running, you should see the following associated Java process in the output of `ps`:

```
 daemon 29295 29294  0 13:05:22?  0:03 java -Djava.library.path=/usr/lib/apoc
 -cp /usr/share/lib/apoc/apocd.jar:/usr/s
 daemon 29294   1   0 13:05:22?  0:00 sh -c java
-Djava.library.path=/usr/lib/apoc -cp /usr/share/lib/apoc/apocd.jar:
 root 29345 28134  0 13:08:59 pts/1   0:00 grep apoc
```

## Where are the log files?

You can consult the following log files when you need to troubleshoot a Configuration Agent problem:

- smf(5) log files:
  - The `/var/svc/log/network-apocd-udp:default.log` file records events that relate to attempts to start and stop specific instances of the Configuration Agent. The file also contains the messages that Configuration Agent controller program, `/usr/lib/apoc/apocd`, writes to its standard output as well as the output messages for the JVM or the Configuration Agent.
  - The `/var/svc/log/svc.startd.log` log file keeps a record of higher level smf(5) events. For example, if multiple failed attempts to start the Configuration Agent occur in quick succession, smf(5) might decide that the Configuration Agent cannot be started. If this occurs, smf(5) places the Configuration Agent in maintenance mode and writes a log entry to that effect.

  Both of these log files are typically useful if you experience problems with the Configuration Agent starting.
- Configuration Agent logs:

  The Configuration Agent writes log messages to log files in the default log directory, `/var/opt/apoc/Logs`. The "Data Directory" for the Configuration Agent is `/var/opt/apoc`. You can change the location of this directory with the Configuration Agent Wizard (`/usr/bin/apoc-config`) application. You can change the granularity of log messages by changing the "Log Level" with the Configuration Agent Wizard. If you think you did not configure the Configuration Agent correctly, or if you are experiencing some other sort of agent failure, use the Configuration Agent Wizard to set the Log Level to "Finest" before you consult the agent log files. This step ensures that you acquire the maximum amount of available logging information.
- System logs:

  You can also check the `/var/adm/messages` log file, or the `/var/opt/SUNWut/log/messages` log file on a SunRay machine to diagnose problems with the Configuration Agent.

## How do I increase the granularity of the Agent logging mechanism?

See "Where are the log files?" on page 32

## What is maintenance mode?

smf(5) places the Configuration Agent in maintenance mode when it detects problems with starting or restarting the Agent. If smf(5) fails to start the Agent, it attempts multiple restarts until the Agent startup is successful or smf(5) decides that the Agent cannot be started. In the latter case, smf(5) places the Agent in maintenance mode to indicate to you that you need to address the problems it has encountered. Once you have addressed the problems, you can clear the smf(5) state of the Agent to return to normal operations.

## How do I get out of maintenance mode - clearing the smf(5) state?

Become superuser and execute the command `/usr/sbin/svcadm clear svc:/network/apocd/udp`.

## What happens if the Configuration Agent stops running unexpectedly?

smf(5) detects that the Agent has stopped running and tries to restart it. If successive attempts to restart fail for any reason, smf(5) places the Agent in maintenance mode. Running desktop client applications are unaffected if the Agent is restarted successfully. Any such client application reconnects to the Agent automatically when it restarts.

## Do I need to restart desktop client applications if I enable/start the Agent?

The action that you take actually depends on whether the Agent was enabled/running at the time when the specific desktop client application was started. If the Agent was enabled/running, the client application established a connection to the Agent and tries to reestablish the connection when it is lost. That is, every time you start, enable or disable the Agent, client applications always try to reconnect to the Agent once it's running. If the Agent was not enabled/running when the client application started, the client application does not use the Configuration Agent and does not even attempt to connect when the Agent starts. All this effectively means that:

- Desktop client applications that started while the Agent was enabled/running do not need to be restarted.
- Desktop client applications that started while the Agent was not enabled/running must be restarted.

## My desktop client applications appear not to be using configured policies. What do I do?

The most common problem that relates to the Configuration Agent is the inability to see the effects of a configured policy on desktop client applications. The most common reasons for this problem are an incorrectly configured Agent, an incorrectly configured policy repository or the unavailability of the policy repository. The following guidelines can help you to discover and eliminate such problems:

- Ensure that the Agent is configured.
- Ensure that the Agent is enabled/running. If you need to start the Agent, you must also restart the desktop client applications that are currently open.

- If your problems persist, temporarily increase the granularity of the Agent logging mechanism and, if possible, restart the Agent, so that you have complete and fine grained log messages from the time the Agent starts.
- If the Agent fails to start correctly, consult the section "Problems starting the Configuration Agent" on page 34.
- If the Agent starts correctly but your desktop client applications do not use an available policy, consult the section "Problems getting policy from a running Configuration Agent".
- If you are still unable to resolve your problems, please contact Technical Support.

## Problems starting the Configuration Agent

If the Configuration Agent cannot be started and you are satisfied that you have configured and enabled the Configuration Agent, you need to consult the log files. The following sections describe the most common errors for this problem.

### Invalid or inaccessible Agent Data Directory

The Configuration Agent Data Directory is created and used by the Agent to store log files, policy caches, and so on. The default location for this directory is `/var/opt/apoc`.

The Configuration Agent produces the following error message in the smf(5) logs when the Data Directory is set to an inaccessible location, that is, `/dev/null/cant/write/here`. To solve this problem, use the Configuration Agent Wizard (`/usr/bin/apoc-config`) to point the Data Directory to an accessible location.

```
[ Nov 17 14:35:38 Executing start method ("/usr/lib/apoc/apocd svcStart") ]
Starting Configuration Agent ... Warning: Cannot create Log directory
   '/dev/null/cant/write/here/Logs'
Warning:Failed to create log file handler
Nov 17, 2005 2:35:39 PM com.sun.apoc.daemon.misc.APOCLogger config
CONFIG: Daemon configuration:
 MaxRequestSize = 4096
 DaemonAdminPort = 38901
 ThreadTimeToLive = 5
 DaemonChangeDetectionInterval = 10
 IdleThreadDetectionInterval = 15
 PROVIDER_URL =
 DataDir = /dev/null/cant/write/here
 ApplyLocalPolicy = true
 ChangeDetectionInterval = 60
 MaxClientConnections = 50
 GarbageCollectionInterval = 10080
 InitialChangeDetectionDelay = 10
 TimeToLive = 10080
 ConnectionReadTimeout = 5000
 DaemonPort = 38900
```

```
 LogLevel = FINEST
 MaxClientThreads = 5


Nov 17, 2005 2:35:39 PM Daemon main
FINER: THROW
com.sun.apoc.daemon.misc.APOCException
    at com.sun.apoc.daemon.apocd.Daemon.initAuthDir(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.init(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.main(Unknown Source)
[ Nov 17 14:36:08 Method or service exit timed out.  Killing contract 980 ]
[ Nov 17 14:36:08 Method "start" failed due to signal KILL ]
```

## Using a client request port that's already busy

The Configuration Agent uses TCP/IP socket connections to communicate with desktop client applications. By default, these connections are made through port 38900.

The following error message is produced when the Configuration Agent is configured to use port 1234, which is already in use by another service. The error message is recorded in the Configuration Agent Logs. To solve this problem, use the Configuration Agent Wizard (`/usr/bin/apoc-config`) to change the Agent Port setting to a port number which is not in use.

```
Nov 17, 2005 2:50:59 PM com.sun.apoc.daemon.misc.APOCLogger config
CONFIG: Daemon configuration:
 MaxRequestSize = 4096
 DaemonAdminPort = 38901
 ThreadTimeToLive = 5
 DaemonChangeDetectionInterval = 10
 IdleThreadDetectionInterval = 15
 PROVIDER_URL =
 DataDir = /var/opt/apoc
 ApplyLocalPolicy = true
 ChangeDetectionInterval = 60
 MaxClientConnections = 50
 GarbageCollectionInterval = 10080
 InitialChangeDetectionDelay = 10
 TimeToLive = 10080
 ConnectionReadTimeout = 5000
 DaemonPort = 1234
 LogLevel = FINEST
 MaxClientThreads = 5


Nov 17, 2005 2:50:59 PM com.sun.apoc.daemon.misc.APOCLogger info
INFO: Daemon starting
Nov 17, 2005 2:50:59 PM com.sun.apoc.daemon.misc.APOCLogger fine
```

```
FINE: Garbage collection scheduled ( interval = 10080 minutes )
Nov 17, 2005 2:50:59 PM Daemon main
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: java.net.BindException: Address already in use
    at com.sun.apoc.daemon.transport.ChannelManager.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.run(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.main(Unknown Source)
Caused by: java.net.BindException: Address already in use
    at sun.nio.ch.Net.bind(Native Method)
    at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:119)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:59)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:52)
```

## Using an administration port that's already busy

The Configuration Agent uses TCP/IP socket connections to communicate with the Configuration Agent controller program (`/usr/lib/apoc/apocd`). By default, these connections are made through port 38901.

The following error message occurs in the Configuration Agent logs when the Configuration Agent is configured to use port 1234, which is already in use by another service. To solve this problem, use the Configuration Agent Wizard (`/usr/bin/apoc-config`) to change the Administration Port setting to a port number which is not in use.

```
ONFIG: Daemon configuration:
 MaxRequestSize = 4096
 DaemonAdminPort = 1234
 ThreadTimeToLive = 5
 DaemonChangeDetectionInterval = 10
 IdleThreadDetectionInterval = 15
 PROVIDER_URL =
 DataDir = /var/opt/apoc
 ApplyLocalPolicy = true
 ChangeDetectionInterval = 60
 MaxClientConnections = 50
 GarbageCollectionInterval = 10080
 InitialChangeDetectionDelay = 10
 TimeToLive = 10080
 ConnectionReadTimeout = 5000
 DaemonPort = 38900
 LogLevel = FINEST
 MaxClientThreads = 5


Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger info
INFO: Daemon starting
Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger fine
FINE: Garbage collection scheduled ( interval = 10080 minutes )
```

```
Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger fine
FINE: Client manager started
Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger fine
FINE: Channel manager started
Nov 17, 2005 2:55:11 PM Daemon main
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: java.net.BindException: Address already in use
    at com.sun.apoc.daemon.admin.AdminManager.initChannel(Unknown Source)
    at com.sun.apoc.daemon.admin.AdminManager.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.run(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.main(Unknown Source)
Caused by: java.net.BindException: Address already in use
    at sun.nio.ch.Net.bind(Native Method)
    at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:119)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:59)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:52)
    ... 4 more
```

## Problems getting policies from a running Configuration Agent

### Missing or invalid configuration repository specification

The Configuration Agent needs to connect to a valid configuration repository to download and to cache policy information. If you do not correctly identify the configuration repository in the Agent's configuration, for example, by using an invalid format or by not specifying a repository, errors similar to the following one are recorded in the Configuration Agent logs when desktop client applications start up. To solve this problem, use the Configuration Agent Wizard (`/usr/bin/apoc-config`) to identify the configuration repository that you want to use.

```
FINER: New client added
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: CreateSession transaction started
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: Creating new client session
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authenticating user geoffh
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authentication successful
Nov 18, 2005 1:59:23 PM PolicyBackend openPolicyBackend
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.environment.InvalidParameterException: The parameter organisation
PROVIDER_URL#protocol (null) is not valid, the value must be comprised in
{ldaps,ldap,https,http,file}.
    at com.sun.apoc.daemon.apocd.PolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.HostPolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyBackend(Unknown Source)
```

```
    at com.sun.apoc.daemon.apocd.Cache$DataSource.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.open(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.createDataSources(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.createNewCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.openCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.Session.<init>(Unknown Source)
    at com.sun.apoc.daemon.transaction.CreateSessionTransaction.executeTransaction
(Unknown Source)
    at com.sun.apoc.daemon.transaction.Transaction.execute(Unknown Source)
    at com.sun.apoc.daemon.apocd.ClientEventHandler.handleEvent(Unknown Source)
    at com.sun.apoc.daemon.apocd.EventWorkerThread.run(Unknown Source)
Caused by: com.sun.apoc.daemon.misc.APOCException:
 com.sun.apoc.spi.environment.InvalidParameterException:
 The parameter organisation PROVIDER_URL#protocol (null) is not valid,
 the value must be comprised in {ldaps,ldap,https,http,file}.
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyMgr(Unknown Source)
    ... 14 more
Caused by: com.sun.apoc.spi.environment.InvalidParameterException: The parameter
organisation PROVIDER_URL#protocol (null) is not valid, the value must be comprised in
{ldaps,ldap,https,http,file}.
    at com.sun.apoc.spi.PolicyMgrFactoryImpl.createPolicyMgr(Unknown Source)
    ... 15 more
Nov 18, 2005 1:59:23 PM PolicyBackend openPolicyBackend
```

## Unable to connect to Policy Repository

The Configuration Agent needs to connect to a valid configuration repository to download and cache policy information. If a connection cannot be established, errors similar to the following one are recorded to the Configuration Agent logs when desktop client applications are started. In the following case, the host sobuild does not exist, cannot be contacted, or cannot access an LDAP server through port 389. To solve this problem, use the Agent Configuration Wizard (`/usr/bin/apoc-config`) to ensure that you have correctly identified the Policy Repository and, if so, to ensure that access to the Policy Repository is available. For example, for an LDAP repository, you need to ensure that an LDAP server is running, the machine hosting the LDAP server is available on the network, and that the port that you specified is the one being used by the LDAP server.

If you are trying to access an LDAP server using an SSL connection, ensure that the proper certificate is available in the key store associated with the Java runtime environment used to run the Configuration Agent. See the section "Configuration Agent" on page 15 for more details about `apoc-config`.

```
FINER: New client added
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: CreateSession transaction started
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: Creating new client session
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finest
```

```
FINEST: Authenticating user geoffh
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authentication successful
Nov 18, 2005 2:17:43 PM PolicyBackend openPolicyBackend
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.OpenConnectionException: An error occured while connecting to
ldap://sobuild:389.
    at com.sun.apoc.daemon.apocd.PolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.HostPolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.open(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.createDataSources(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.createNewCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.openCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.Session.<init>(Unknown Source)
    at com.sun.apoc.daemon.transaction.CreateSessionTransaction.executeTransaction
(Unknown Source)
    at com.sun.apoc.daemon.transaction.Transaction.execute(Unknown Source)
    at com.sun.apoc.daemon.apocd.ClientEventHandler.handleEvent(Unknown Source)
    at com.sun.apoc.daemon.apocd.EventWorkerThread.run(Unknown Source)
Caused by: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.OpenConnectionException: An error occured while
connecting to ldap://sobuild:389. at
com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyMgr(Unknown Source)
    ... 14 more
Caused by: com.sun.apoc.spi.OpenConnectionException: An error occured while
connecting to ldap://noSuchHost:389.
    at com.sun.apoc.spi.ldap.LdapClientContext.prepareConnection(Unknown Source)
    at com.sun.apoc.spi.ldap.LdapClientContext.connect(Unknown Source)
    at com.sun.apoc.spi.ldap.LdapConnectionHandler.openAuthorizedContext(Unknown Source)
    at com.sun.apoc.spi.ldap.LdapConnectionHandler.connect(Unknown Source)
    at com.sun.apoc.spi.ldap.entities.LdapOrganizationProvider.open(Unknown Source)
    at com.sun.apoc.spi.PolicyMgrFactoryImpl.createPolicyMgr(Unknown Source)
    ... 15 more
Caused by: netscape.ldap.LDAPException: failed to connect to server sobuild:389 (91);
Cannot connect to the LDAP server
    at netscape.ldap.LDAPConnSetupMgr.connectServer(LDAPConnSetupMgr.java:422)
    at netscape.ldap.LDAPConnSetupMgr.openSerial(LDAPConnSetupMgr.java:350)
    at netscape.ldap.LDAPConnSetupMgr.connect(LDAPConnSetupMgr.java:244)
    at netscape.ldap.LDAPConnSetupMgr.access$0(LDAPConnSetupMgr.java:241)
    at netscape.ldap.LDAPConnSetupMgr$1.run(LDAPConnSetupMgr.java:179)
    at java.lang.Thread.run(Thread.java:595)
Nov 18, 2005 2:17:44 PM PolicyBackend openPolicyBackend
```

## Connection to a non-configured Policy Repository

Before the Configuration Agent can locate policy data in a Policy Repository, the Policy Repository must be correctly configured. If you specify a non-configured or incorrectly configured Policy repository, errors similar to the following are recorded in the Configuration Agent logs when desktop client applications are started. To solve this problem, please see the section

```
FINER: New client added
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: CreateSession transaction started
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: Creating new client session
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authenticating user geoffh
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authentication successful
Nov 18, 2005 2:36:55 PM PolicyBackend openPolicyBackend
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.environment.RemoteEnvironmentException: Error on reading the
configuration data on LDAP server ldap://sobuild:389.
    at com.sun.apoc.daemon.apocd.PolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.HostPolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.open(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.createDataSources(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.createNewCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.openCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.Session.<init>(Unknown Source)
    at com.sun.apoc.daemon.transaction.CreateSessionTransaction.executeTransaction
(Unknown Source)
    at com.sun.apoc.daemon.transaction.Transaction.execute(Unknown Source)
    at com.sun.apoc.daemon.apocd.ClientEventHandler.handleEvent(Unknown Source)
    at com.sun.apoc.daemon.apocd.EventWorkerThread.run(Unknown Source)
```

## I see a message in the Configuration Agent Logs regarding a "maximum number of client connections". What does it mean?

Each desktop client application (gconfd, Mozilla, StarOffice) that is enabled by the Configuration Agent opens a connection to the Configuration Agent when it runs. The limit of such connections is specified in the configuration of the Agent. The default limit for connections is 50. On a machine with multiple users, you might have to increase this limit by changing the “Maximum Client Connections” settings with the Configuration Agent Wizard (`/usr/bin/apoc-config`). If the Configuration Agent reaches the maximum number of connections, error messages similar to the following one are recorded in Configuration Agent logs:

```
Nov 18, 2005 3:20:55 PM com.sun.apoc.daemon.misc.APOCLogger warning
WARNING: The maximum number of client connections ( 50 ) has been reached.
No new client connections can be established at this time.
```

## I modified some policies using the Desktop Manager but the modification is not seen on my client machines

One of the assumptions made when designing the Configuration Agent is that policy data created by the Desktop Manager is relatively static, that is, not subject to frequent change. This assumption results in an approach where the Agent intermittently consults the Policy Repository to see if modifications have occurred. By default, the Agent checks the repository once an hour for all running desktop applications. As a result, when you make a change with the Desktop Manager, you need to wait up to one hour before running desktop applications are notified of the change. If you want, you can use the Agent Configuration Wizard (`/usr/bin/apoc-config`) to change the value of the "General Detection Interval" to increase the frequency of repository checks. Alternatively, you can force the Configuration Agent to refresh policy data for all connected applications by becoming superuser and executing the `/usr/lib/apoc/apocd change-detect` command.

◆ ◆ ◆ CHAPTER 4

# Java Web Console

The Java 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 Java 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

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

# Installation

## System Requirements

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

### Client

- Netscape™ 6.2x, and 7.x on Solaris 10
- Netscape 6.2x, and 7.x on Windows 98, 98 SE, ME, 2000, and XP
- Internet Explorer 5.5x and 6.x on Windows 98, 98 SE, ME, 2000, and XP

- Mozilla 1.4x on Solaris
- Firefox 1.0 on Solaris

### Server

- Solaris 10 on x86 and SPARC® platforms
- 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.

## Installing the Java Web Console

The Java Web Console 2.2.4 is part of the Solaris 10 operating system, however the Desktop Manager requires version 2.2.5. A copy of version 2.2.5 is provided in the Desktop Manager archive in the `server/console` directory. It can be installed by running `./setup` in that directory.

If you have the Java Web Console 3.0 installed, then you meed to uninstall the 3.0 version, and then install Java Web Console 2.2.5 from the `server/console` directory, as mentioned above.

# Running the Console

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

**Caution –** Before you start the Java Web Console for the first time, ensure that the Desktop Manager installation is completed. The Java Web Console will *not* run successfully until you have deployed at least one application in the console.

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

Out of the box, the Java 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 Java Web Console, see the smcwebserver and smreg man pages.

# Removing the Java Web Console

**Caution –** If you are on Solaris, you can not remove the Java Web Console, because it is part of the operating system.

# Java Web Console Troubleshooting

## Cannot Install the Java Web Console

Symptom: at the end of the installation, a message states that the Java Web Console cannot start because there are no registered applications.

Possible causes: once Desktop Manager module is installed, it starts the Java Web Console.

## Connection Refused

Symptom: you try to open the appropriate URL, for example `https://`*<your.server>*`:6789`, but the connection is refused.

Possible causes: the Java Web Console is not running on the server.

## Cannot Log In

**Note –** By default, the LDAP login module is not installed. As a result, login credentials are not compared to the ones stored in the LDAP server and only normal system logon credentials are required. This troubleshooting section only applies if you manually installed the LDAP login module.

Symptom: you reach the login page of the Web Console, but the user/password combination is rejected.

Possible causes:

- LDAP server is not running.
- Web Console LDAP authentication module is configured incorrectly
- User does not exist on the LDAP server.
- User has a different password on the LDAP server.

## No Desktop Manager Link

Symptom: you log in to the Web Console, but the application list page does not contain the Desktop Manager.

Possible causes:

- Desktop Manager module is not installed

## Null Pointer Exception, Tomcat/Java Error or Blank

Symptom: you open the Desktop Manager but nothing of value is displayed, just a blank page or some errors.

Possible causes: if the error mentions `NoClassDefFoundError: sun/tools/javac/Main`, the Java Web Console is using the wrong Java installation.

## Other Issues

If the web server is not performing properly, the log files may provide information. They are located at `/var/log/webconsole/`. You can increase the log detail level by using `smreg`:

```
smreg add -p debug.trace.level=3
smreg add -p debug.trace.options=tmp
```

The original settings can be restored by:

```
smreg add -p debug.trace.level=0
smreg add -p debug.trace.options=m
```

A full dump of the configuration database is triggered by:

```
smreg list
```

It is possible that the web server hosting the Desktop Manager may not shut down correctly, leaving its ports in use. This prevents a newly started web server from starting at all. If the `smcwebserver start/restart` command issues an error messages, or if the Desktop Manager is still accessible even after a `smcwebserver stop`, or if the newly started server still behaves like the old instance, check if

either port 6789 is still in use (`netstat -a | grep 6789`) or if the web server is still running (`ps -ef | grep java`). If the one or the other is the case, the according process should be killed and the port 6789 is not in use anymore.

◆ ◆ ◆ APPENDIX A

# Configuration Parameters

These parameters can be defined for the following Desktop Manager components:

- Desktop Manager, in the files defining the Configuration Repositories (located in `/etc/opt/SUNWapcmg/`).
- Configuration Agent, in the `/etc/apoc/policymgr.properties` file.
- Desktop Manager CLI, in the `$HOME/pgtool.properties` file, with the restriction that the CLI only supports pure LDAP repositories.

The parameters can be prefixed to indicate which repository provider they apply to. For each provider, the parameter with the prefix is considered first. If such a parameter is not defined, then the parameter without the prefix is used.

**TABLE A–1** Prefixes

| Prefix Value | Repository Provider |
|---|---|
| ORGANIZATION_ | Organization Tree |
| DOMAIN_ | Domain Tree |
| PROFILE_ | Profiles |
| ASSIGNMENT_ | Assignments |
| LDAP_META_CONF_ | Mapping data in the case of LDAP repositories |

**TABLE A–2** Parameters

| Name | Description | Possible Values | Default Value |
|---|---|---|---|
| PROVIDER_URL | URL specifying the connection to the repository. A list of URLs can be used to specify fallback repositories in case the connection to the first one does not succeed. | List of one or more white-space-separated URLs, each URL being of one of the following forms: `ldap://<host>:<port>/<baseDN>` `ldaps://<host>:<port>/<baseDN>` `file://<filepath>` `http://<host>:<port>/<filepath>` `https://<host>:<port>/<filepath>` | None, mandatory parameter |
| SECURITY_PRINCIPAL | User name for the connection to the repository. | User name of a user that has read and search access rights to the repository or no value for anonymous connections. | None, anonymous connection |
| SECURITY_CREDENTIALS | Password for the user defined in SECURITY_PRINCIPAL. | Scrambled or clear text password. | None |
| SECURITY_CREDENTIALS_ENCODING | Indicates whether the password defined in SECURITY_PRINCIPAL is scrambled. Warning: the password scrambling is just a mask over the password, it constitutes by no means any type of secure encryption. | “scramble” if the password is scrambled (automatically done by wizards when generating the configuration data). “none” if the password appears in clear text; use this value if you wish to edit the password. | “none” |
| MAX_SEARCH_RESULT | Maximum number of results given by a search in any of the repositories. Note: the prefix scheme does not apply to this parameter. | Positive number, 0 means no limit. | 100 |

The following parameters apply to LDAP repositories only.

**TABLE A–3** LDAP Specific Parameters

| Name | Description | Possible Values | Default Value |
|---|---|---|---|
| AuthDn | Fully qualified DN of a user to be used for the first access to the LDAP repository, in order to retrieve the user defined in SECURITY_PRINCIPAL. | User name of a user that has read and search access rights to the repository, or no value for anonymous connections. | None, anonymous access |
| Password | Password for AuthDN. | Scrambled or clear text password. | None |
| Password_ENCODING | Indicates whether the password defined in Password is scrambled. Warning: the password scrambling is just a mask over the password, it constitutes by no means any type of secure encryption. | "scramble" if the password is scrambled (automatically done by wizards when generating the configuration data). "none" if the password appears in clear text; use this value if you wish to edit the password. | "none" |
| Connect Timeout | Connection creation timeout in seconds. | Positive number, 0 for unlimited time. | 1 |

**EXAMPLE A–1** Example for a hybrid backend

Example for a hybrid backend, where the information about the hosts and users is obtained from an existing LDAP repository, while the profiles and their assignments are stored on the file system.

```
#Organization, Domain, MetaConf
PROVIDER_URL = ldap://server1.sun.com:389/o=apoc ldap://server2.sun.com:389/o=apoc
SECURITY_PRINCIPAL = jmonroe
SECURITY_CREDENTIALS = JmonroE
SECURITY_CREDENTIALS_ENCODING = none
AuthDn = cn=reader,ou=special users,o=apoc
Password = lakjflajf
Password_ENCODING = scramble
ConnectTimeout = 5

#Profile
PROFILE_PROVIDER_URL = file:///path/to/repository

#Assignment
ASSIGNMENT_PROVIDER_URL = file:///path/to/repository
```

◆ ◆ ◆ APPENDIX B

# Using OpenLDAP and Active Directory with the Desktop Manager

## Using an OpenLDAP Server with the Desktop Manager

To use an OpenLDAP server as the repository for the Desktop 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 `/usr/share/webconsole/apoc/deploy` directory.

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.

Having extended the OpenLDAP servers schema, the rest of the configuration can be completed using the Add Configuration Repository wizard in the Desktop Manager.

---

**Note –** The Desktop 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 Desktop Manager

To use an Active Directory server as a repository for the Desktop 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 `/usr/share/webconsole/apoc/deploy` directory.

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 for more information on 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.

---

Having extended the Active Directory servers schema, the rest of the configuration can be completed using the Add Configuration Repository wizard in the Desktop Manager.

When prompted for LDAP credentials in the Add Configuration Repository wizard, 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 Desktop 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 desktop host, the domain name of the Active Directory must also be known to that host. Authentication of the desktop 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 user must have authenticated against the Active Directory and the user credentials must be available on the system. This can be achieved by configuring Kerberos authentication on your system, which will generate these credentials when logging in, please refer to the administration guides for your system to find out more information on how to do this.

◆ ◆ ◆ APPENDIX C

# Organizational Mapping

## Organizational Mapping

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

User elements are identified by an object class that all elements 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 elements 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 elements 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 elements are identified in a way that is similar to organization elements. 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 elements are identified in a way that is similar to user elements. 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
```