Sun Identity Manager 8.1 System Administrator's Guide

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

Part No: 820–5823
February 2009

090312@21990

Contents

Preface

This *Identity Manager 8.1 System Administrator's Guide* contains information that system administrators must understand to administer the systems running Sun™ Identity Manager software (Identity Manager).

Before You Read This Book

Identity Manager is a component of Sun Java Enterprise System, a software infrastructure that supports enterprise applications distributed across a network or Internet environment. You should be familiar with the documentation provided with Sun Java Enterprise System, which can be accessed online at `http://docs.sun.com/coll/entsys_04q4`.

Because Identity Manager Directory Server is used as the data store in an Identity Manager deployment, you should be familiar with the documentation provided with that product. Directory Server documentation can be accessed online at `http://docs.sun.com/coll/DirectoryServer_04q2`.

Who Should Use This Book

This book is for application server and database administrators, front-line support engineers, and partners who administer systems running Identity Manager and are interested in optimizing Identity Manager performance and responsible for maintaining Identity Manager in a deployment environment.

Identity Manager administrators must understand the following technologies:

- Lightweight Directory Access Protocol (LDAP)
- Java™ technology
- JavaServer Pages™ (JSP™) technology
- Hypertext Transfer Protocol (HTTP)
- Hypertext Markup Language (HTML)
- Extensible Markup Language (XML)

How This Book is Organized

This guide is organized into the following chapters:

Chapter 1, "Configuring Identity Manager Server Settings," describes how to set Identity Manager properties and configure Identity Manager servers to run specific tasks.

Chapter 2, "Working with Firewalls, Load Balancers, or Proxy Servers," describes Identity Manager uses Uniform Resource Locators (URLs) and contains information about using Identity Manager when firewalls, load balancers, or proxy servers are in place.

Chapter 3, "Working with Logs," describes tasks you might perform when you are working with audit or system logs.

Chapter 4, "Performance Tuning," describes tools, methodologies, and references you can use to improve Identity Manager performance and to debug performance-related issues.

Chapter 5, "Tracing and Troubleshooting," explains how to use Identity Manager tracing to fix problems, solve performance issues, and understand how things flow in the product components.

Chapter 6, "Errors and Exceptions," contains information about Identity Manager error and exception messages.

Related Documentation and Help

Other publications in the Sun Identity Manager library include:

Primary Audience	Title	Description
All Audiences	*Sun Identity Manager Overview*	Provides an overview of Identity Manager features and functionality. Provides product architecture information and describes how Identity Manager integrates with other Sun products, such as Sun Open SSO Enterprise and Role Manager.
	Sun Identity Manager 8.1 Release Notes	Describes known issues, fixed issues, and late-breaking information not already provided in the Identity Manager documentation set.

Primary Audience	Title	Description
System Administrators	*Sun Identity Manager 8.1 Installation*	Describes how to install Identity Manager and optional components such as the Sun Identity Manager Gateway and PasswordSync.
	Sun Identity Manager 8.1 Upgrade	Provides instructions on how to upgrade from an older version of Identity Manager to a newer version.
	Sun Identity Manager 8.1 System Administrator's Guide	Contains information and instructions to help system administrators manage, tune, and troubleshoot their Identity Manager installation.
Business Administrators	*Sun Identity Manager 8.1 Business Administrator's Guide*	Describes how to use Identity Manager's provisioning and auditing features. Covers the user interfaces, user and account management, reporting, and more.
System Integrators	*Sun Identity Manager Deployment Guide*	Describes how to deploy Identity Manager in complex IT environments. Topics covered include working with identity attributes, data loading and synchronization, configuring user actions, applying custom branding, and so on.
	Sun Identity Manager Deployment Reference	Covers workflows, forms, views, and rules, and includes a chapter on the XPRESS language.
	Sun Identity Manager 8.1 Resources Reference	Provides information about installing, configuring, and using resource adapters.
	Sun Identity Manager Service Provider 8.1 Deployment	Describes how to deploy Sun Identity Manager Service Provider, and how views, forms, and resources differ from the standard Identity Manager product.
	Sun Identity Manager 8.1 Web Services	Describes how to configure SPML support, which SPML features are supported (and why), and how to extend support in the field.

Documentation Updates

Corrections and updates to this and other Identity Manager publications are posted to the Identity Manager Documentation Updates website:

`http://blogs.sun.com/idmdocupdates/`

An RSS feed reader can be used to periodically check the website and notify you when updates are available. To subscribe, download a feed reader and click a link under Feeds on the right side of the page. Starting with version 8.0, separate feeds are available for each major release.

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to `http://docs.sun.com` and click the Feedback link. In the online form, enter the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document.

For example, the title of this book is *Identity Manager 8.1 System Administrator's Guide*, and the part number is *820-5823-10*.

Typographic Conventions

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123` (Monospace)	API and language elements, HTML tags, web site URLs, command names, file names, directory path names, onscreen computer output, sample code.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123` (Monospace bold)	What you type, when contrasted with onscreen computer output.	`% su Password:`
AaBbCc123 (Italic)	Book titles, new terms, words to be emphasized. A placeholder in a command or path name to be replaced with a real name or value.	Read Chapter 6 in the *Identity Manager 8.1 System Administrator's Guide*. These options are called *class* options. Do *not* save the file. The file is located in the *install-dir*`/bin` directory.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Note – The Windows command-line prompt is `c:\`.

◆ ◆ ◆ CHAPTER 1

Configuring Identity Manager Server Settings

This chapter describes how to set Identity Manager properties and configure your Identity Manager servers to run specific tasks.

The information in this chapter is organized into the following topics:

Using `Waveset.properties` to Configure Identity Manager

Some Identity Manager settings can only be updated by editing `Waveset.properties` in a text editor. The `Waveset.properties` file is located in the `config` directory in your base Identity Manager installation directory (`$WSHOME/config` or `%WSHOME%/config`).

The `Waveset.properties` file is loosely organized into the following sections:

- Misc Options
- Authentication Options
- Object Cache Options
- Provisioning Options
- Task Scheduler Options
- Resource Adapter Options
- List Cache Properties
- Rule Driven Members Cache Properties
- Session Settings
- Server Listeners

- UI Options
- Global Options for the JNDI Context Pooling
- Task Executor Options
- XML Parsing Options

▼ To Edit `Waveset.properties`

1. **In a text editor, open `Waveset.properties`, which is located in the `config` directory in your base Identity Manager installation directory (`$WSHOME/config` or `%WSHOME%/config`).**
2. **Modify the setting (or settings) that you want to change and save the file.**
3. **Repeat as needed on other Identity Manager instances.**

Using the Administrator Interface to Configure Identity Manager

This section provides basic instructions for using the Administrator tool to configure server-specific settings for reconciler, scheduler, JMX and other tasks.

▼ To Configure Identity Manager Settings from the Administrator Interface

To edit server-specific settings that enable Identity Manager servers to run only specific tasks

1. **In the Administrator interface, click Configure → Servers.**
   The Configure Servers page opens.
2. **When the Configure Servers page displays, click a server name in the list.**
   Identity Manager displays the Edit Server Settings page.
3. **Modify the settings on the Edit Server Settings page that you want to change, and then click Save.**

Reconciler Settings

The reconciler is the Identity Manager component that performs reconciliation. To learn about reconciliation, see "Account Reconciliation" in *Sun Identity Manager 8.1 Business Administrator's Guide*.

This section provides instructions to help you perform the following tasks:

- "To Configure Reconciler Settings" on page 15
- "To View Reconciler Status" on page 16

Note – For information about tuning and troubleshooting the reconciler, see Chapter 5, "Tracing and Troubleshooting."

▼ To Configure Reconciler Settings

Use the following steps to configure the Reconciler:

1. **Follow the steps described in "Using the Administrator Interface to Configure Identity Manager" on page 14.**

2. **Select the Reconciler tab.**

   Reconciler settings display on the Edit Server Settings page by default.

3. **Accept the default values or deselect the Use the default option to specify custom values.**

   Note – To change the default reconciler settings used by Identity Manager servers, see "Editing Default Server Settings" on page 19.

   ---

4. **Configure the following settings.**

   - **Parallel Resource Limit**. Specify the maximum number of resource threads that the reconciler can process in parallel. Resource threads allocate work items to worker threads, so if you add additional resource threads, you may also need to increase the maximum number of worker threads. For new installations, the default value is 3.
   - **Minimum Worker Threads**. Specify the number of processing threads that the reconciler will always keep alive. For new installations, the default value is 2.
   - **Maximum Worker Threads**. Specify the maximum number of processing threads that the reconciler can use. The reconciler will only start as many threads as the workload requires. This places a limit on that number. Worker threads automatically close if they are idle for a short duration. For new installations, the default value is 6.

▼ To View Reconciler Status

Use the following steps to view reconciler status information:

1 **Open the Reconciler Status debug page by typing the following URL into your browser:**

`http://<AppServerHost>:<Port>/idm/debug/Show_Reconciler.jsp`

Where `AppServerHost` is a host that has the reconciler enabled.

Note – You must have the Debug capability to view `/idm/debug/` pages. For information about capabilities, see "Assigning Capabilities to Users" in *Sun Identity Manager 8.1 Business Administrator's Guide*.

---

2 **Refresh the Reconciler Status page to view updated reconciler status information. For additional information about this page, click Help.**

Scheduler Settings

The scheduler component controls task scheduling in Identity Manager.

To configure scheduler settings on a particular server,

1. Follow the steps under "Using the Administrator Interface to Configure Identity Manager" on page 14.
2. Select the Scheduler tab.

   You can accept the default values or deselect the Use default option to specify the following custom values.

   - **Scheduler Startup**. Select a startup mode for the scheduler on this server:
     - **Automatic**. Starts when the server is started. This is the default startup mode.
     - **Manual**. Starts when the server is started, but remains suspended until manually started.
     - **Disabled**. Does not start when the server is started.
   - **Tracing Enabled**. Select this option to activate scheduler debug tracing to standard output on this server.
   - **Maximum Concurrent Tasks**. Select this option to specify the maximum number of tasks, other than the default, that the Scheduler will run at any one time. Requests for additional tasks above this limit will either be deferred until later or run on another server.

- **Task Restrictions**. Specify the set of tasks that can execute on the server. To do this, select one or more tasks from the list of available tasks. The list of selected tasks can be an inclusion or exclusion list depending on the option you select. You can choose to allow all tasks except those selected in the list (the default behavior), or allow only the selected tasks.

3. Click Save to save your changes to the server settings.

To change the default scheduler settings for Identity Manager servers, see "Editing Default Server Settings" on page 19. For information about tuning and troubleshooting the scheduler, see "Tuning the Scheduler" on page 64 and "Tracing the Task Scheduler" on page 116.

Email Template Server Settings

To configure SMTP server settings, follow the steps under "Using the Administrator Interface to Configure Identity Manager" on page 14. Select the Email Template tab.

Specify the default email server by clearing the Use Default selection and entering the mail server to use, if other than the default. The text you enter is used to replace the *smtpHost* variable in Email Templates.

Simple Mail Transfer Protocol (SMTP) is the standard for email transmissions across the Internet.

To change the default SMTP settings for Identity Manager servers, see "Editing Default Server Settings" on page 19.

Configuring JMX Monitoring

Java Management Extensions (JMX) is a Java technology that allows for managing and/or monitoring applications, system objects, devices, and service oriented networks. The managed/monitored entity is represented by objects called MBeans (for Managed Bean).

This section describes how to configure JMX on an Identity Manager server so that a JMX client can monitor the system for changes.

Note – You can also configure Identity Manager to make audit events available using JMX. For information, see "The JMX Publisher Type" in *Sun Identity Manager 8.1 Business Administrator's Guide*.

▼ To Configure JMX Polling Settings

Use the following process to configure JMX polling settings on an individual server:

1. **Follow the steps described in "Using the Administrator Interface to Configure Identity Manager" on page 14. Select the JMX tab.**

2. **Enable JMX cluster polling and configure the interval for the polling threads.**

   Use the following options:

   - **Enable JMX**. Enables or disables the polling thread for the JMX Cluster MBean. To enable JMX, clear the default selection (Use the `false` default setting). Because of the use of system resources for polling cycles, enable this option only if you plan to use JMX.
   - **Polling Interval (ms)**. Changes the default interval at which the server polls the repository for changes, when JMX is enabled. Specify the interval in milliseconds.

     The default polling interval is set to 60000 milliseconds. To change it, clear the check box for this option and enter the new value in the entry field provided.

3. **Click Save to save changes to the server settings.**

Note – To change the default JMX polling settings for Identity Manager servers, see "Editing Default Server Settings" on page 19.

Viewing JMX Data

Use a JMX client to view data gathered by JMX. JConsole, which is included in the JDK 1.5, is one such client.

Using JConsole Locally

To use JConsole on the same machine your server is running on, set the `JAVA_OPTS` property as follows:

```
-Dcom.sun.management.jmxremote
```

JConsole will connect using the correct PID.

Using JConsole Remotely

To use JConsole remotely, set the `JAVA_OPTS` property as follows:

- `-Dcom.sun.management.jmxremote.port=9004`
- `-Dcom.sun.management.jmxremote.authenticate=false`
- `-Dcom.sun.management.jmxremote.ssl=false`
- In the `jre/lib/management` directory, edit `jmxremote.access` and make sure the following two lines appear uncommented in the file:
  - `monitorRole readonly`

- `controlRole readwrite`

  To see the Identity Manager MBeans, connect to the server with an URL similar to the following:

  ```
  service:jmx:rmi:///jndi/rmi://localhost:9004/jmxrmi
  ```

Other settings may also be necessary depending on your environment. Refer to the JConsole documentation for more information.

Note – You can also view JMX data by going to the Identity Manager debug page (http://*host*:*port*/idm/debug) and clicking the Show MBean Info button.

For more information on JMX, visit this website:

`http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/docs.jsp`

Editing Default Server Settings

The Default Server Settings feature lets you set the default settings for all Identity Manager servers. The servers inherit these settings unless you select differently in the individual server settings pages.

▼ To Edit the Default Server Settings

1. **In the Administrator interface, click Configure → Servers.**

   The Configure Servers page opens.

2. **Click Edit Default Server Settings.**

   The Edit Default Server Settings page opens.

   The Edit Default Server Settings page displays the same options as the individual server settings pages. For help, refer to the documentation for the individual server settings pages.

   Changes you make to each default server setting is propagated to the corresponding individual server setting, unless you have deselected the Use default option for that setting.

   Click Save to save changes to the server settings.

◆ ◆ ◆ CHAPTER 2

Working with Firewalls, Load Balancers, or Proxy Servers

This chapter describes how Identity Manager uses Uniform Resource Locators (URLs) and how to configure Identity Manager to obtain accurate URL data when firewalls, load balancers, or proxy servers are in place.

Servlet APIs

The Web-based Identity Manager user interface is highly dependent on Uniform Resource Locators (URLs) to specify the location of pages to be retrieved by the Web client.

Identity Manager depends on the Servlet APIs provided by an application server (such as Glassfish, Apache Tomcat, IBM WebSphere, or BEA WebLogic) to determine the fully qualified URL in the current HTTP request so that a valid URL can be placed in the generated HTML and HTTP response.

Some configurations prevent the application server from determining the URL the Web client uses for an HTTP request. Examples include:

- A port-forwarding or Network Address Translation (NAT) firewall placed between the Web client and Web server, or between the Web server and application server
- A proxy server (such as Tivoli Policy Director WebSEAL) placed between the Web client and Web server, or between the Web server and application server

For instances in which the Servlet APIs do not provide accurate URL data from an HTTP request, the correct data can be configured in the `Waveset.properties` file (located in your Identity Manager installation `config` directory).

The following attributes control Identity Manager's Web-based documentation root and whether Identity Manager uses the HTML BASE HREF tag.

- `ui.web.useBaseHref` (Default value: `true`)— Set this attribute to one of the following values:
  - `true`— Identity Manager uses the HTML BASE HREF tag to indicate the root of all relative URL paths.
  - `false`— All URLs placed into HTML contain fully qualified paths; including scheme, host, and port.
- `ui.web.baseHrefURL`— Set this attribute to a non-empty value to define the BASE HREF used in generated HTML, which overrides the value that is calculated using servlet APIs.

Overriding this calculated value can be useful when those APIs do not return the whole truth, which occurs when:

- The application server is behind a firewall using port forwarding or NAT
- The connector between the application server and Web server does not provide accurate information
- The application server is front-ended by a proxy server

Configuring Logs to Work with Load Balancers

You can configure Identity Manager to automatically log the client IP address contained in the `x-Forwarded-For` HTTP request header, which is the standard header for identifying the originating IP address of a client connecting to a web server through a load balancer or HTTP proxy. If necessary, you can also configure Identity Manager to use a custom HTTP header.

To force Identity Manager to log IP addresses contained in a custom HTTP request header, use the following steps:

1. Open `Waveset.properties` in a text editor.
2. Search for `client.headerIPVariable=` and uncomment the line.
3. Type the name of the HTTP request header that you want Identity Manager to use instead.

   Alternatively, you can disable this feature by setting the property equal to zero.
4. Save `Waveset.properties`.
5. Restart Identity Manager.

Note – You can apply this configuration to both audit logs and system logs.

If you do not want Identity Manager to automatically log the client IP address, you can disable this feature by uncommenting the `client.headerIPVariable` in the `Waveset.properties` file.

◆ ◆ ◆ CHAPTER 3

Working with Logs

This chapter describes tasks that you might have to perform when working with Audit logs or System logs.

These topics include:

Preventing Audit Log Tampering

You can configure Identity Manager to prevent the following forms of audit log tampering:

- Adding or inserting audit log records
- Modifying existing audit logs records
- Deleting audit log records or the entire audit log
- Truncating audit logs

All Identity Manager audit log records have unique, per-server sequence numbers and encrypted hash of records and sequence numbers.

When you create a Tamper Detection Report, it scans the audit logs per server for:

- Gaps in the sequence number (indicating a deleted record)
- Hash mismatches (indicating a modified record)
- Duplicate sequence numbers (indicating a copied record)
- Last sequence number that is less than expected (indicating a truncated log)

▼ To Configure Tamper-Resistant Logging

1 **Create a tampering report by selecting Reports > New > Audit Log Tampering Report.**

2 **When the Define a Tampering Report page displays, as shown in Figure 3–1, enter a title for the report and then Save it.**

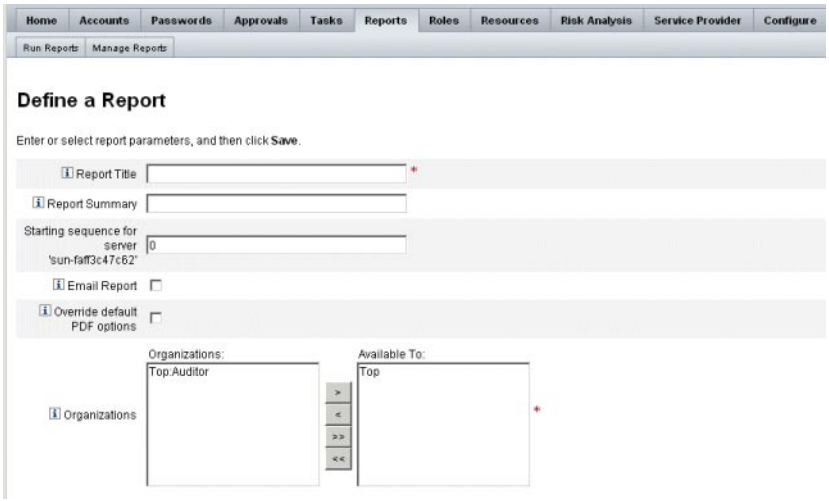


FIGURE 3–1 Configuring an Audit Log Tampering Report

You can also specify the following optional parameters:

- **Report Summary**. Enter a descriptive summary of the report.
- **Starting sequence for server '<*server_name*>'**. Enter the starting sequence number for the server.
- This option enables you to delete old log entries without having them flagged as tampering and limits the report's scope for performance reasons.
- **Email Report**. Enable to email report results to a specified email address.
- When you select this option, the page refreshes and prompts you for email addresses. However, keep in mind that email is not safe for text content-sensitive information (such as account IDs or account history) may be exposed.
- **Override default PDF options**. Select to override the default PDF options for this report.
- **Organizations**. Select organizations that should have access to this report.

3 **Next, select Configure → Audit to open the Audit Configuration page, as shown in Figure 3–2.**

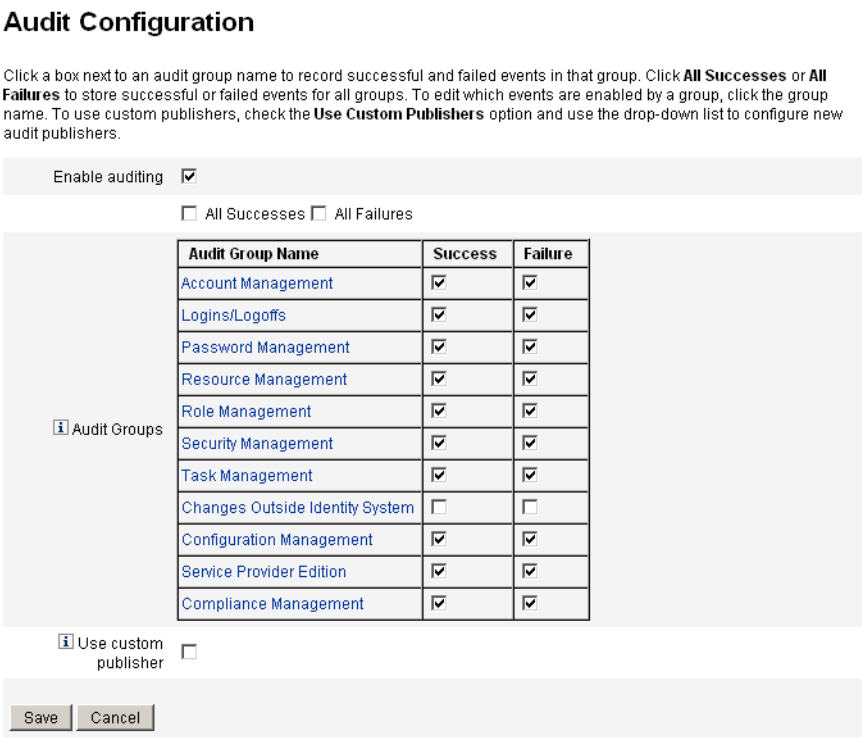


FIGURE 3–2 Tamper-Resistant Audit Logging Configuration

4 **Select Use Custom Publisher, and then click on the Repository publisher link.**

5 **Select Enable tamper-resistant audit logs, and then click OK.**

6 **Click Save to save the settings.**

You can turn this option off again, but unsigned entries will be flagged as such in the Audit Log Tampering Report, and you must reconfigure the report to ignore these entries.

Removing Records from the System Log

The system log captures errors generated by Identity Manager. Periodically, the system log should be truncated to keep it from growing too large. Use the System Log Maintenance Task to remove old records from the system log.

▼ To Schedule a Task to Remove Old Records From the System Log

1. **In the Administrator interface, click Server Tasks → Manage Schedule.**
2. **In the Tasks Available for Scheduling section, click the System Log Maintenance Task.**
   The Create New System Log Maintenance Task Task Schedule page opens.
3. **Complete the form and click Save.**

◆ ◆ ◆ CHAPTER 4

Performance Tuning

You can significantly improve the performance of Sun Identity Manager (Identity Manager) software across nearly all activities with the proper tuning. In addition to changing settings within the software, you can make performance improvements by tuning the application server, the Java™ Virtual Machine (JVM™ machine), hardware, operating system, and network topology.

Additionally, you can use several tools to diagnose and monitor performance. Several of these tools exist within Identity Manager, such as trace and method timers. You can also use other Sun Microsystems and third-party tools to debug performance issues with Identity Manager.

This chapter describes tools, methodologies, and references you can use to improve performance and to debug performance-related issues.

Note – The tuning process spans many entities and is specific to your deployment environment. This chapter describes different tuning methods for optimizing Identity Manager performance, but these methods are *only intended as guidelines*. You might have to adapt these methods for your deployment environment.

This chapter covers the following topics:

Before You Begin Tuning

Review all of the information in this section before you start tuning Identity Manager.

Important Notes

Note – The tuning methods described in this chapter are only provided as *guidelines*. You might have to modify some of these tunings for your deployment. In addition, Be sure to validate tunings *before* applying changes in a production environment.

Before you can tune Identity Manager, you must:

- Be familiar with tuning application servers
- Be familiar with Java 5.0 (required for Sun Identity Manager 8.1)
- Understand performance limitations within your deployment environment
- Be able to identify areas needing performance improvements
- Understand the checklists provided in this chapter

Related Documentation and Web Sites

In addition to the information provided in this chapter, consult the documents and web sites listed in this section for information related to tuning Identity Manager.

Recommended Reading

See the following documents for information related to performance tuning.

TABLE 4–1 Related Documentation

document Title	Description
IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 5.0 Diagnostics Guide	Explains how to use AIX JVM to diagnose performance problems.
Java Tuning White Paper	Contains information, techniques, and pointers related to Java performance tuning.
Oracle MetaLink: Note:114671.1: Gathering Statistics for the Cost Based Optimizer	Explains how to use system statistics and Oracle®'s Cost-Based Optimizer (CBO). **Note**: This document is available to Oracle Metalink subscribers. Registration is required.

TABLE 4–1 Related Documentation *(Continued)*

document Title	Description
Solaris Dynamic Tracing Guide	Explains how to use DTrace to observe, debug, and tune your system's behavior.
Sun Java™ System Application Server Performance Tuning Guide	Describes how to obtain optimal performance from your Sun Java System Application Server. Download the necessary version of this book from the Sun Microsystems documentation web site. (`http://docs.sun.com/app/docs`)
Tuning Garbage Collection with the 5.0 Java Virtual Machine	Describes how to tune your garbage collection application by using JVM.
Turbo-charging Java HotSpot Virtual Machine, v1.4.x to Improve the Performance and Scalability of Application Servers	Explains how to download and use the `PrintGCStats` script and how to collect statistics to derive optimal JVM tunings.
Understanding System Statistics	Describes how Oracle's Cost-Based Optimizer uses system statistics.
Using JConsole to Monitor Applications	Describes how to use JConsole to monitor applications that run on the Java platform.

Useful Web Sites

The following table describes some web sites that you might find useful when trying to tune Identity Manager performance.

TABLE 4–2 Useful Web Sites

Web Site URL	Description
`http://sunsolve.sun.com`	Sun web site containing diagnostic tools, forums, features and articles, security information, and patch contents. **Note**: The information on this site is divided into three areas: ■ **Internal**. Available only to Sun employees ■ **Contract**. Available only to customers with contract access ■ **Public**. Available to everyone
`http://forum.java.sun.com/`	Sun Developer Network (SDN) web site where you can browse forums and post questions.
`http://jrat.sourceforge.net/`	JRat web site that describes how to use the Java Runtime Analysis Toolkit, an open source performance profiler for the Java platform.

TABLE 4–2 Useful Web Sites *(Continued)*

Web Site URL	Description
https://metalink.oracle.com/	Oracle's internal forum site that contains information about tuning Oracle databases. **Note**: You must be an Oracle Metalink subscriber to access the information provided on this site.
http://performance.netbeans.org/howto/jvmswitches/index.html	NetBeans™ web site containing information about tuning JVM switches for performance.
https://sharespace.sun.com/gm/folder-1.11.60181?	Identity Manager link on Sun's Share Space. **Note**: You must sign up for a Share Space ID to access information provided on this site.
https://sharespace.sun.com/gm/document-1.26.2296	Identity Manager FAQ on Sun's Share Space. **Note**: You must sign up for a Share Space ID to access this FAQ.
http://www.slamd.com/	SLAMD Distributed Load Generation Engine web site.
http://www.opensolaris.org/os/community/dtrace/	OpenSolaris Community: DTrace web page.
http://www.solarisinternals.com/ http://docs.sun.com/app/docs/prod/solaris.10	Web sites containing information related to tuning the Solaris OS.

Tuning Roadmap

How well your Identity Manager solution performs can depend on the following deployment-specific settings:

- Resource configuration
- How many resources are connecting
- What type of resources are connecting
- How attributes are mapped on the resources
- Exact resource version
- Network topology
- Number (and distribution) of domain controllers
- Number of installed Identity Manager Gateways
- Number of concurrent settings
- Number of concurrent processes (running workflows)
- Number of concurrent users
- Number of concurrent Identity Manager administrators
- Total number of users under management

When you are trying to debug performance problems, start by analyzing and describing the problem. Ask yourself the following questions:

- Where do you see the performance issue? In reconciliation, workflows, custom workflows, GUI page loading, provisioning, Access Review?
- Are you CPU bound, memory bound, resource bound, or network bound?
- Have you examined your configuration (hardware, network, parameters, and so forth) for problems?
- Have you recently changed anything in your deployment environment?
- Have you tried profiling troublesome resources natively to see if the problem is on the resource side and not with Identity Manager?
- What size are the views?
- Are you provisioning to several resources?
- Are resources on slow networks connecting to Identity Manager?
- Are you running additional applications on the server with Identity Manager?
- Do your organizations have a Rule-Driven Members rule?
- Have you run a series of thread dumps to see if there is a consistent pattern?

Note – Looking at just a single thread dump can be misleading.

- Have you recently turned on tracing?
- Have you checked your JVM garbage collection?
- Have you added organizations that are adding load to memory management?

Tuning Your Deployment Environment

This section provides information about tuning your deployment environment, including:

Tuning Your Java EE Environment

This section describes some tuning suggestions you can use to optimize your Java Platform, Enterprise Edition (Java EE platform) environment.

These tuning suggestions were derived from a series of experiments in which a considerable increase in throughputs was observed for the use cases tested. The increases were attributed to JVM sizing and to switches that affected garbage collector behavior.

Note – For more information about tuning Java, JConsole, or JVM, visit the web sites noted in Table 4–1 and Table 4–2.

The following sections provide information about tuning Java and the JVM in your Java EE environment.

Tuning Java

For information, best practices, and examples related to Java performance tuning, see the *Java Tuning White Paper* at:

`http://java.sun.com/performance/reference/whitepapers/tuning.html`

Tuning the JVM

The following tuning scripts were used to derive the tuning suggestions noted in this section. These scripts were added to the `domain.xml` file (located in the domain configuration directory, which is typically `domain-dir/config`) on a Sun Java System Application Server.

- `PrintGCStats` – A data mining shell script that collects data from `verbose:gc` logs and displays information such as garbage collection pause times, parameter calculations, and timeline analyses over the application's runtime by sampling the data at user-specified intervals.

  Note – For more information about how to use this script and garbage collection statistics to derive optimal JVM tunings, see the following web site:

  `http://java.sun.com/developer/technicalArticles/Programming/turbo/#PrintGCStats`

- `PrintGCDetails` – A shell script that can provide more verbose garbage collection statistics.
- `PrintGCTimeStamps` – A shell script that adds time-stamp information to the garbage collection statistics collected by using the `PrintGCDetails` script.

To help ensure the best JVM performance, verify the following:

- Be sure that you are using the required Java version noted in the "Supported Software and Environments" section of the *Sun Identity Manager 8.1 Release Notes* to ensure you are using the most current features, bug fixes, and performance enhancements.
- Be sure that you are using a newer version of garbage collection.

  Frequently, customers do not remove the older, default garbage collection scheme when installing an application server. Running Identity Manager with an older garbage collector creates many objects, which forces the JVM to constantly collect garbage.
- If you deployed Identity Manager on Sun Java System Application Server, you can increase throughput by adding garbage collection elements to the deployed Identity Manager instance `server.xml` file.

  If you expect a peak load of more than 300 users, try modifying the following settings to increase performance:

  – For HTTP listeners configured for the deployed Identity Manager instance, edit the listener definition element in the `server.xml` file and set the number of acceptor threads to the *Number of active CPUs on the host* divided by the *Number of active HTTP listeners on the host*.

    For example:

    ```
    <http-listener id="http-listener-1" \address="0.0.0.0" port="80"
    \acceptor threads="Calculated Acceptor Threads" ...>
    ```

  – Because the static content of most Identity Manager deployments is not projected to change frequently, you can edit the File Cache settings (on the File Cache Configuration page) for static content. Specify a high number (such as the number of seconds in 24 hours) for the maximum age of content within the file cache before the content is reloaded.

    To access the File Cache Configuration page, click the File Caching tab on the web-based Administrative Console for the HTTP server node. (See the latest *Sun Java System Web Server Administrator's Guide* for detailed instructions.)

    Note – Sun Java System Application Server exposes tunables that affect the size of various thread pools and connection queues that are maintained by the HTTP container.

    By default, most of these tunables are set for a concurrent user load of 300 users or less.

Tuning Your Application Server

The following guidelines are provided to help you tune your application server:

Note – Other than heap size, you can use the default parameter settings for most application servers. You might want to modify the server's heap size, depending on the release being used.

Tuning a Sun Java System Application Server

The "Tuning the Application Server" chapter, in the latest *Sun Java System Application Server Performance Tuning Guide*, contains information about tuning a Sun Java System Application Server. This document is available from the following URL at `http://docs.sun.com/app/docs`.

In addition, if you are using Sun Java System Application Server 8.2 Enterprise Edition, the following changes solve "concurrent mode failures," and should give you better and more predictable performance:

- If you are constantly running old generation collections, review your application's heap footprint and consider increasing the size. For example:
  - 500 Mbytes is considered a modest size, so increasing this value to 3 Gbytes might improve performance.
  - With a 2–Gbyte young generation collection, each scavenge promotes about 70 Mbytes. Consider giving this 70 Mbytes at least one more scavenge before promoting. For example, you might need the following `SurvivorRatio`:

    ```
    2 GB/70 M X 2 = 4096/70 = 55
    ```

    where:

    ```
    -XX:SurvivorRatio=32 -XX:MaxTenuringThreshold=1
    ```

    This ratio prevents premature promotion and the added problem of "nepotism," which can degrade scavenge performance.
- If you specified `-XX:CMSInitiatingOccupancyFraction=60`, and the CMS collections are still starting before they reach that threshold. For example:

  ```
  56402.647: [GC [1 CMS-initial-mark: 265707K(1048576K)] 1729560K(3129600K),
  3.4141523 secs]
  ```

  Try removing `-XX:CMSInitiatingOccupancy=60` (using the default value of 69percent), and add the following line:

  ```
  -XX:UseCMSInitiatingOccupancyOnly
  ```

- If your young generation collection is 2 Gbytes and the old generation collection is 1 Gbyte, this situation might also be causing premature CMS collections. Consider reversing this ratio. Use a 1–Gbyte young generation collection and a 2–Gbyte old generation collection, as follows

  ```
  -Xms3G -Xmx3G -Xmn1G
  ```

  Also, remove `-XX:NewRatio`. This ratio is redundant when you have explicitly specified young generation and overall heap sizes.
- If you are using a `5uXX` version of the Java Development Kit (JDK™ software), and have excessively long "abortable preclean" cycles, you can use `-XX:CMSMaxAbortablePrecleanLoops=5` as a temporary workaround.

You might have to adjust this value further.

- Add the following line to view more information about garbage collector performance.

  ```
  -XX:+PrintHeapAtGC
  ```

  Note – Use this command with caution because it increases how much verbose garbage collection data is produced. Be sure that you have enough disk space on which to save the garbage collector output.

- If you are using the Sun Fire™ T2000 server, large-heap data Translation Look-aside Buffers (DTLBs) can become a scarce resource. Using large pages for the Java heap often helps performance. For example:

  ```
  -XX:+UseLargePages
  ```

Tuning a WebSphere Application Server

If you are tuning Identity Manager on an IBM WebSphere® application server, consider limiting how much memory is allocated for the heap because heap memory can affect the memory used by threads.

If many threads are created simultaneously and the heap size increases, the application's space limit can be quickly impacted and the following error results:

```
JVMCI015:OutOfMemoryError
```

Tuning Your Repository Database

Identity Manager relies on the repository database to store and manage its identity and configuration data. For this reason, database performance can greatly influence Identity Manager's performance.

Note – Detailed information about performance tuning and managing databases is beyond the scope of this document because this information is dataset-specific and vendor-specific. In addition, customer database administrators (DBAs) should already be experts on their own databases.

This section characterizes the Identity Manager application and provides general information about the nature of Identity Manager data and its typical usage patterns to help you plan, tune, and manage your databases.

This information is organized into the following sections:

Repository Table Types

The Identity Manager repository contains three types of tables, and each table has slightly different usage characteristics. Information about these tables is organized into the following sections:

Attribute Tables

Attribute tables enable you to query for predefined single-valued or multi-valued object attributes.

For most object types, stored attributes are hard-coded.

Note – The `User` and `Role` object types are exceptions to this rule. The inline attributes that are stored in the object table for `User` and `Role` are configurable, so you can configure additional custom attributes as queryable.

When you search for objects based on attribute conditions, Identity Manager accesses attribute tables in joins with the corresponding object tables. Some form of join (such as a `JOIN`, an `EXISTS` predicate, or a `SUB-SELECT`) occurs for each attribute condition.

The number of rows in the attribute table are proportional to the number of rows in the corresponding object table. The values distribution might exhibit *skew*, where multi-valued attributes have a row per value and some objects might have more attributes or more attribute values than others. Typically, there is a many-to-one relation between rows in the attribute table and rows in the object table.

Attribute tables have `ATTR` in the table name.

Change Tables

Identity Manager uses a change table to track changes made to a corresponding object table. These table sizes are proportional to the rate of object change, but the tables are not expected to grow without bound. Identity Manager automatically truncates change tables.

Change tables can be highly volatile because the lifetime of a row is relatively short and new rows can be created frequently.

Access to a change table is typically performed by a range scan on the time-stamp field.

Change tables have `CHANGE` in the table name.

Object Tables

The Identity Manager repository uses object tables to hold serialized data objects, such as Large Objects (LOBs). Object tables can also hold commonly queried, single-valued object attributes.

For most object types, stored attributes are hard-coded.

Note – The `User` and `Role` object types are exceptions to this rule. The inline attributes that are stored in the object table are configurable, and you can configure additional custom attributes as queryable for `User` and `Role`.

The number of rows in an object table equals the number of objects being stored. The number of objects stored in each object table depends on which object types are being stored in the table. Some object types are numerous, while other types are few.

Generally, Identity Manager accesses an object table by object ID or name, though Identity Manager can also access the table by using one of the attributes stored in the table. Object IDs and names are unique across a single object type, but attribute values are not unique or evenly distributed. Some attributes have many values, while other attributes have relatively few values. In addition, several object types can expose the same attribute. An attribute may have many values for one object type and few values for another object type. The uneven distribution of values might cause an uneven distribution of index pages, which is a condition known as *skew*.

Object tables are tables that *do not* have `ATTR` or `CHANGE` suffixes in the table name.

XML Columns

Every object table contains an XML column, which is used to store each serialized object except the `LOG` table-set. Certain `LOG` table-set optional attributes are stored in the XML column if these attributes are present. For example, if digital signing is enabled.

Data Classes

You can roughly divide Identity Manager data into a number of classes that exhibit similar properties with respect to access patterns, cardinality, lifetime, volatility, and so forth. Each of the following classes corresponds to a set of tables in the repository:

User Data

User data consists of user objects.

You can expect this data to grow quite large because there is an object for each managed identity. After an initial population phase, you can expect a proportionally small number of creates because the majority of operations will be updates to existing objects.

User objects are generally long-lived and they are removed at a relatively low rate.

User data is stored in `USEROBJ`, `USERATTR`, and `USERCHANGE` tables.

Role Data

Role data consists of `Role` objects, including Roles subtypes such as Business Roles, IT Roles, Applications, and Assets.

Role data is similar to organization data, and these objects are relatively static after a customer deploys Identity Manager.

Note – An exception to the preceding statement is a deployment that is integrated with an external source containing an authoritative set of roles. One integration style might be to feed role changes into Identity Manager, which causes Identity Manager Role data to be more volatile.

Generally, the number of role objects is small when compared to the number of identity objects such as users (assuming that multiple users share each role), but this depends on how each enterprise defines its roles.

Role data is stored in `ROLEOBJ`, `ROLEATTR`, and `ROLECHANGE` tables.

Account Data

Account data solely consists of account objects in the Account Index.

As with user data, account data can become rather large, with an object for each known resource account. Account objects are generally long-lived, removed at a relatively low rate, and after initial population, are created infrequently. Unless you frequently add or remove native accounts, or enable native change detection, account object modifications occur infrequently.

Identity Manager stores account data in `ACCOUNT`, `ACCTATTR`, and `ACCTCHANGE` tables.

Compliance Violation Data

Compliance Violation data contains violation records that indicate when the evaluation of an Audit Policy failed. These violation records exist until the same Audit Policy is evaluated against the same User and the policy passes. Violation records are created, modified, or deleted as part of an Audit Policy Scan or as part of an Access Review.

The number of violation records is proportional to the number of Audit Policies that are used in scans and the number of Users. An installation with 5000 users and 10 Audit Policies might have 500 violation records (5000 x 10 x 0.01), where the 0.01 multiplier depends on how strict the policies are and how user accounts are changed.

Identity Manager stores Compliance Violation records in `OBJECT`, `ATTRIBUTE`, and `OBJCHANGE` tables.

Entitlement Data

Entitlement data predominately consists of user entitlement objects, which are only created if you are doing compliance access reviews.

Entitlement records are created in large batches, modified slowly (days) after initial creation, and are then untouched. These records are deleted after an Access Review is deleted.

Identity Manager stores entitlement data in ENTITLE, ENTATTR, and ENTCHANGE tables.

Organization Data

Organization data consists of object group or organization objects.

Object group data is similar to configuration data, and this data is relatively static after being deployed. Generally, the number of objects is small (one for each defined organization) when compared to task objects or to identity objects such as users or accounts, however, the number can become large compared to other configuration objects.

Organization data is stored in ORG, ORGATTR, and ORGCHANGE tables.

Task Data

Task data consists of objects that are related to tasks and workflows, including state and result data.

The data contained in these tables is short-lived compared to other classes because objects are created, modified, and deleted at a high rate. The volume of data in this table is proportional to the amount of activity on the system.

Task data is stored in TASK, TASKATTR, and TASKCHANGE tables.

Configuration Data

Configuration data consists of objects related to Identity Manager system configuration, such as forms, roles, and rules.

Generally, configuration data is:

- Relatively small compared to other classes
- Only expected to change during deployment and upgrade, and changes occur in large batches
- Not expected to change much after being deployed

Identity Manager stores configuration data in ATTRIBUTE, OBJCHANGE, and OBJECT tables.

Export Queue Data

If you enable Data Exporting, some records are queued inside Identity Manager until the export task writes those records to the Data Warehouse. The number of records that are queued is a function of Data Exporting configuration and the export interval for all queued types.

The following data types are queued by default, and all other data types are not:

- `ResourceAccount`
- `WorkflowActivity`
- `TaskInstace`
- `WorkItem`

The number of records in these tables grows until the export task drains the queue. The current table size is visible through a JMX™ Bean.

Records added to this table are never modified. These records are written during other Identity Manager activities, such as reconciliation, provisioning, and workflow execution. When the Data Exporter `export` task runs, the task drains the table.

Identity Manager stores Export Queue data records in `QUEUE`, `QATTR`, and `QCHANGE` tables.

Log Data

Log data consists of audit and error log objects. Log data is write-once only, so you can create new audit and error log objects, but you cannot modify these objects.

Log data is long-lived and can potentially become very large because you can only purge log data by explicit request. Access to log data frequently relies on attributes that are stored in the object table instead of in the attribute table. Both the distribution of attribute values and queries against the log specifically depend on how you are using Identity Manager.

For example, the distribution of attribute values in the log tables depends on the following:

- What kind of changes are made
- Which Identity Manager interface was used to make the changes
- Which types of objects were changed

The pattern of queries against the log table also depends on which Identity Manager reports, which custom reports, or which external data mining queries a customer runs against the log table.

Identity Manager stores audit log records in `LOG` and `LOGATTR` tables, and error log records in `SYSLOG` and `SLOGATTR` tables. This data does not have corresponding change tables.

Object IDs

Identity Manager generates globally unique identifiers (GUIDs) for objects by using the `VMID` class provided in the JDK software.

These GUID values exhibit a property that gets sorted by its string representations, based on the order in which the objects are created. For example, when you create new objects with Identity Manager, the newer objects have object IDs that are greater than the older objects. Consequently, when Identity Manager inserts new objects into the database, the index based on object IDs can encounter contention for the same block or blocks.

Prepared Statements

Generally, Identity Manager uses prepared statements for activities (such as inserting and updating database rows), but does not use prepared statements for queries.

If you are using Oracle, this behavior can create issues with the library cache. In particular, the large number of statements versions can cause contention on the library cache latch.

To address this contention, change the Oracle `CURSOR_SHARING` parameter value from `EXACT` to `SIMILAR`. Changing this value causes Oracle to replace literals in SQL statements with bind variables, thereby reducing the number of versions.

Character Sets and Encodings

Because Identity Manager is a Java application that generally reads and writes character data rather than bytes, it does not restrict which encoding the database uses.

Identity Manager only requires that the data is sent and returned correctly. For example, the data does not become corrupted when written or reread. Use an encoding that supports multi-byte characters and is appropriate for the customer's data. Generally, UTF-8 encoding is sufficient, but enterprises with a large number of true multi-byte characters, such as Asian or Arabic, might prefer UTF-16.

Most database administrators prefer to use an encoding that supports multi-byte characters because of the following:

- Their deployments often grow to support international characters.
- Their end users cut-and-paste from a Microsoft application's text containing characters that look like ASCII but are actually multi-byte, such as em dashes (–).

General Guidelines for Tuning a Repository Database

This section describes some general guidelines for tuning a repository database:

- DBAs must frequently run statistics to monitor what is happening with the repository database.
- If you are using a data source, set the `connectionPoolDisable` attribute to `true` in the `RepositoryConfiguration` object to disable automatic internal connection pooling in the Identity Manager repository.

  For example, setting `<RepositoryConfiguration connectionPoolDisable='true'>` allows you to avoid having two connection pools (one for Identity Manager and one for your application server).
- You can edit the `RepostioryConfiguration` object to enhance the performance of searches against specific, single-valued attributes. For example, you might edit this object to add an extended attribute, such as `employeeID`, that is used to search for Users or as a correlation key.

The default `RepositoryConfiguration` object looks like the following example:

```
<RepositoryConfiguration ... >
   <TypeDataStore Type="User" ... attr1="MemberObjectGroups",
attr2="lastname" attr3="firstname" attr4="" attr5="">
   </TypeDataStore>
</RepositoryConfiguration>
```

Note – The ellipses represent XML attributes that are not relevant here.

Each of the `attr1`, `attr2`, `attr3`, `attr4`, and `attr5` XML attributes specifies a single-valued attribute to be copied into the `waveset.userobj` table. The `waveset.userobj` table can contain up to five inline attributes. The attribute value named by `attr1` in `RepositoryConfiguration` will be copied into the `"attr1"` database column in this table.

Inline attributes are stored in the base `object` table for a `Type` (rather than as rows in the child `attribute` table).

Using inline attributes improves the performance of repository queries against those attributes. (Because inline attributes reside in the main "object" table, queries against inline attributes are faster than those against non-inline attributes, which are stored in the child "attribute" table. A query condition against a non-inline attribute requires a "join" to the attribute table.)

By default, Identity Manager uses the `MemberObjectGroups`, `lastname`, and `firstname` inline attributes.

- You can add two more attributes to enable faster searching, as long as those attributes are queryable.

  For example, if your deployment contains an `employeeID` extended attribute, adding that attribute inline will improve the performance of repository searches against that attribute.
- If you do not need `lastname` or `firstname`, you can remove them or replace them with other attributes.
- *Do not* remove `MemberObjectGroups`. Identity Manager uses this attribute internally to speed up authorization checks.

For more information about which object types are stored in each set of tables, see "Data Classes" on page 38.

Vendor-Specific Database Tuning Guidelines

This section describes some vendor-specific guidelines for tuning Oracle and SQL Server repository databases.

Note – Currently, MySQL™ databases are only supported in development and for demonstrations.

Oracle Databases

This section describes guidelines for tuning Oracle repository databases:

- The Identity Manager application does not require Oracle database features or options.
- If you are using an Oracle repository database and Identity ManagerService Provider or Identity Manager, you might encounter problems with object table fragmentation because Identity Manager uses `LONG`, rather than `LOB`, data types by default. Using `LONG` data types can result in large amounts of "unallocated" extent space, which cannot be made into usable space.

  To mitigate this problem, do the following:

  – Take `EXPORT` dumps of the Object table and re-import them to free up unallocated extent space. After importing, you must stop and restart the database.

  – Use `LOB` data types and DataDirect Technologies' Merant drivers, which provide a standard `LOB` implementation for Oracle.

  – Use Locally Managed Tablespaces (LMTs), which offer automatic free space management. LMTs are available in Oracle 8.1.5.

Identity Manager does not require Oracle `init.ora` parameter settings for SGA sizing, buffer sizing, open cursors, processes, and so forth.

- While the Identity Manager repository is a general-purpose database, it is best described as an object database.

  Of the Identity Manager tables, the `TASK` table-set comes closest to having transaction-processing characteristics. The `LOG` and `SYSLOG` table-sets are also exceptional because these tables do not store serialized objects.

  See "Repository Table Types" on page 36 and "Data Classes" on page 38 for descriptions of the tables, the object types stored in each table, and the general access pattern for each table.

- If you have performance issues with the Oracle database, check for issues related to poor query plans being chosen for what Identity Manager expects to be relatively efficient queries.

  For example, Identity Manager is configured to perform a full table-scan when an index is available for use. These issues are often visible in Automated Workload Repository (AWR) reports provided in the SQL by the `buffer gets` table. You can also view issues in the Enterprise Manager tool.

  Performance problems typically appear to be the result of bad or missing database table statistics. Addressing this problem improves performance for both the database and Identity Manager.

  The following articles (available from Oracle) are a good source of information about the cost-based optimizer (CBO) in Oracle:

  - *Understanding System Statistics*
  - *Oracle MetaLink: Note:114671.1: Gathering Statistics for the Cost Based Optimizer*
  - *Cost Control: Inside the Oracle Optimizer*

  - You might also investigate using SQL Profiles, which are another method for choosing the best query plans. You can use the SQL Advisor within Enterprise Manager to create these profiles when you identify poorly performing SQL.

  If you detect unexpected growth in the Oracle redo log, you might have workflows that are caught in an infinite loop with a manual action. The loop causes constant updates to the repository, which in turn causes the size of each `TaskInstance` to grow substantially. The workflow errors are caused by improper handling of `WF_ACTION_TIMEOUT` and by users closing their browser in the middle of a workflow.

  To prevent problematic workflows, preview each manual action before a production launch and verify the following:

  - Have you set a timeout?
  - Have you created appropriate transition logic to handle a timeout for the activity with the manual action?
  - Is the manual action using the exposed variables tag when there is a large amount of data in the `TaskInstance`?

Frequently, you can significantly improve Identity Manager performance if you change the CURSOR_SHARING parameter value from EXACT to SIMILAR.

Identity Manager uses prepared statements for some activities (such as inserting and updating database rows), but does not use these statements for most queries.

When you use Oracle, this behavior can cause issues with the library cache. In particular, the large number of statement versions can create contention on the library cache latch. Changing CURSOR_SHARING to SIMILAR causes Oracle to replace literals in SQL statements with bind variables, which greatly reduces the number of versions.

See "Prepared Statements" on page 42 for more information.

SQL Server Databases

Some customers who used an SQL Server 2000 database as a repository reported that as concurrency increased, SQL Server 2000 reported *deadlocking* problems that were related to SQL Server's internal use of pessimistic locking (primarily lock escalation).

These deadlock errors display in the following format:

```
com.waveset.util.IOException:
  ==> com.microsoft.sqlserver.jdbc.SQLServerException: Transaction (Process ID 51)
was deadlocked on lock | communication buffer resources with another
process and has been chosen as the deadlock victim. Rerun the transaction.
```

To prevent or address deadlocking problems, do the following:

1. Use the SQL Server 2005 database.
2. Configure the READ_COMMITTED_SNAPSHOT parameter by formatting the command as follows:

   ```
   ALTER DATABASE waveset SET READ_COMMITTED_SNAPSHOT ON
   ```

   Enabling the READ_COMMITTED_SNAPSHOT parameter does the following:

   - Removes contention during the execution of SELECT statements that can cause blocks, which greatly reduces the potential for deadlocks internal to SQL Server.
   - Prevents uncommitted data from being read and guarantees that SELECT statements receive a consistent view of committed data.

   For more information about the READ_COMMITTED_SNAPSHOT parameter, see: http://msdn2.microsoft.com/en-us/library/ms188277.aspx.

Tuning Identity Manager Performance

Suggestions for optimizing Identity Manager's performance are organized into the following areas:

General Performance Tunings

In general, you can optimize Identity Manager performance if you do the following:

- Turn off tracing (such as Java class, userform, and workflow tracing). Tracing can add substantial overhead.
- Run the Identity Manager built-in Audit Log Maintenance Task and System Log Maintenance Task to configure log record expirations. Log records can grow without bound, so use these tasks to prevent the repository database from running out of space. For information, see the *Sun Identity Manager 8.1 Business Administrator's Guide*.
- Check the `README` file in Identity Manager updates (formerly called *service packs* or *installation packs*) to see if any performance improvements have been made to the product. If so, schedule an upgrade.
- Consider the performance impact when fetching data from one or more remote systems, including the Identity Manager repository.
- Increase the number of application server instances running Identity Manager, either on the same server or by adding servers, and use a load-balancing tool to distribute the requests between instances.

- Keep the size of files referenced in a binary attribute as small as possible. Loading extremely large graphics files, for example, can decrease Identity Manager performance.
- Write robust and readable XML code that minimizes duplication (for example, refactored), that uses memory efficiently, and that mitigates the impact to overall system performance.
- Configure Identity Manager system monitoring to track events in real time.

  You can view these events in dashboard graphs to quickly assess system resources, spot abnormalities, understand historical performance trends (based on the time of day, the day of week, and so forth), and interactively isolate problems before looking at audit logs. Dashboards do not provide as much detail as audit logs, but can provide hints about where to look for problems in the logs.

  For more information about dashboards, see Chapter 8, "Reporting," in *Sun Identity Manager 8.1 Business Administrator's Guide*.

Tuning Active Sync Adapter Performance

Because synchronization is a background task, how you configure an Active Sync adapter can affect server performance.

Use the Resources list to manage Active Sync adapters. Choose an Active Sync adapter and access start, stop, and status refresh control actions from the *Synchronization* section of the Resource Actions list.

To improve Active Sync adapter performance, do the following:

- Evaluate and adjust polling intervals based on the type of activity being performed.

  The polling interval determines when the Active Sync adapter will start processing new information. For example, if the adapter reads in a large list of users from a database and updates these users in Identity Manager each time, you could run this process in the early morning every day. Some adapters have a quick search for new items to process and can be set to run every minute.
- Edit the synchronization file for the resource to specify the host where the adapters will run.

  You can configure Active Sync adapters that require more memory and CPU cycles to run on dedicated servers to help load balance the systems.
- If you have the appropriate administrator capability, you can change Active Sync resources to disable, manually start, or automatically start Active Sync adapters.

  When you set an adapter to automatic, the adapter restarts when the application server starts. When you start an adapter, it runs immediately and executes at the specified polling interval. When you stop an adapter, it stops the next time the adapter checks for the stop flag.
- Adjust the level of detail captured by the synchronization logs.

Synchronization logs capture information about the resource that is currently processing. Each resource has its own log file, path, and log level. The amount of detail captured by the adapter log depends on the specified logging level. You specify these values in the Logging section of the Synchronization Policy for the appropriate user type (Identity Manager or Service Provider).

Tuning Bulk Loading

To improve performance during bulk load operations, do the following:

- Simplify default workflows to improve processing time (especially for bulk processing actions such as Active Sync, bulk actions, and reconciliation) by removing the callout to the Approval subprocess.
- Keep user forms that are assigned to administrators as simple as possible. For example:
  - When creating a form for data loading, remove any code that is designed to display data.
  - When using bulk add actions, be sure that your CSV file defines basic attributes such as `firstname` and `lastname`. You can then remove these attributes from the administrator user form.

Note – Do not modify the default forms provided with Identity Manager. Instead, make a copy of the form, give the copy a unique name, and modify the renamed copy. This approach prevents your customized forms from being overwritten during upgrades and product updates.

See Chapter 3, "Identity Manager Forms," in *Sun Identity Manager Deployment Reference* for more information about creating and editing forms.

- Implement the following features in deployment environments where you have NIS (Network Information Service) implemented:
  - Add an account attribute named `user_make_nis` to the schema map and use this attribute in your reconciliation or other bulk provisioning workflow. Specifying this attribute causes the system to bypass the step of connecting to the NIS database after each user update on the resource.
  - To write the changes to the NIS database after provisioning has completed, create a ResourceAction named `NIS_password_make` in the workflow.

Tuning Configurable XML Objects

Configurable XML objects offer a broad spectrum of user interface specifications that enable you to define how data is presented to users for different tasks and to automate complex business processes. However, this same flexibility can affect efficiency, performance, and reliability.

This section describes some guidelines for tuning Identity Manager's configurable XML objects, which consist of forms, rules, and workflows. The information is organized into the following sections:

Tuning Forms

You can use Identity Manager forms to define interfaces to interact with views or variable contexts in an executing task. Forms also provide an execution context for business and transformation logic on a set of data elements. Although you can create very powerful, dynamic forms that perform a variety of tasks, reducing the complexity of forms increases efficiency.

The following sections describe some methods for improving the performance of your customized forms:

Optimizing New Forms

When designing new Identity Manager forms, system integrators can optimize a form's performance by doing the following:

- Performing "expensive" queries only one time, wherever possible. To minimize these queries,
  - Use `<Field>` `<Default>` elements to execute and store query results.
  - Use field names to reference values in later fields.
  - For custom tasks
    - Calculate the value in the task before a `ManualAction`, then store that value in a task variable.
    - Use `variables.tmpVar` to reference variables in the form.
    - Use `<setvar name='tempVar'/>` to clear the variable after a `ManualAction`.

- Using <defvar> for calculations that are performed for the initial display and with each refresh.

Optimizing Administrator Forms

To improve the performance of administrator forms, do the following:

- Specify TargetResources that only fetch specific resources for editing. (See "Tuning Workflows" on page 52 for more information.)
- Use cacheList and cacheTimeout caching parameters for objects that change infrequently if you are working with FormUtil.getResourceObjects or FormUtil.listResourceObjects.
- Store the results of time-consuming calculations and fetches in <Field> elements and evaluate in the <Default> expression to help ensure that an operation occurs only one time.
- Use update.constraints to limit which resources are fetched at runtime (see "Dynamic Tabbed User Form" in *Sun Identity Manager Deployment Reference*).
- Use background approval (ManualAction with different owners and one-second timeouts) for faster page submissions.
- Be aware that Identity Manager refreshes *all fields* defined on *all panels* of a Tab Panel Form when the page reloads, regardless of which panel is selected.

Optimizing End-User Forms

To improve the performance of end-user forms, do the following:

- Use TargetResources to limit view checkouts to just those resource accounts of interest, which reduces fetch time for view and the memory consumed by TaskInstance and WorkItems.
- Consider using Session.getObject(Type, name) to return a WSUser if just the view properties and attributes of the Identity Manager user object are of interest (useful for managing multiple deferred task triggers).
- Be aware that end-user tasks typically have more WorkItems than Provisioning tasks, so end user tasks are especially susceptible to WorkItem size.
- Consider using temporary generic objects for "view" editing that is constructed on view check-out then merged back into a full view for check-in.
- Consider using scalable forms instead of the default Create and Edit User interfaces.

  When you use the default User forms to edit a user, Identity Manager fetches the resources owned by that user the moment you start editing the user's account. In deployment environments where users have accounts on many resources, this potentially time-intensive operation can result in performance degradation.

Optimizing Expressions in Form Fields

Some activities performed in forms call resources that are external to Identity Manager. Accessing these resources can affect Identity Manager performance, especially if the results contain long lists of values, such as compiling a list of groups or email distribution lists.

To improve performance during these calls, follow the guidelines in "Using a Java Class to Obtain Field Data" in *Sun Identity Manager Deployment Reference*.

Also, avoid using JavaScript™ in performance-critical expressions such as `<Disable>` expressions. Short XPRESS expressions are easier to debug if you use the built-in tracing facilities. Use JavaScript for complex logic in workflow actions.

If a form is slow to display, you can use the `debug/Show_Timings.jsp` page to determine the problem. Look for calls to `Formconvert.convertField()`. This method shows how long each field took to compute its value.

Tuning Rules

You use Identity Manager rules to encapsulate constants and XPRESS logic that can be reused in forms, workflows, and other configurable components in the product.

When writing rules, use the following guidelines (as applicable) to obtain optimal performance:

- Use static declarations to return a constant value.
- Use `defvar` methods to implement algorithms with temporary values for incremented values or for values that are referenced only one time.
- Use `putmap`, `setlist`, or `setvar` methods for complex or expensive calculations whose value must be returned multiple times. Be sure to eventually set the value to `<null>`.

Tuning Workflows

You customize Identity Manager workflows to facilitate and automate complex business processes with various human and electronic touchpoints.

You can use the following methods to improve custom workflow performance:

- Simplify default workflows to improve processing time (especially for bulk processing actions such as Active Sync, bulk actions, and reconciliation) by removing the callout to the Approval subprocess.
- Ensure that no infinite loops exist in your workflows. In particular, be sure that break flags are updated and properly checked in the loops that exist in approval subprocesses.
- Put fetched objects into a variable for use later if you must contact the repository for the same object multiple times.

  Using a variable is necessary because Identity Manager does not cache all objects.
- Specify `TargetResources` options in WorkflowServers `checkoutView` to restrict the number of resources that are queried for account information.

The following example shows how to restrict the number of resources being queried for account information.

```
<Argument name='TargetResources'>
  <list>
    <string>resource name[| #]</string>
  </list>
</Argument>
```

Note – In the preceding example, `[| #]` is an optional parameter that you can use when more than one account exists on a particular resource. In most cases, the resource name is sufficient.

- Clear unnecessary view variables left by forms, especially large maps and lists. For example:

  `<setvar name='myLargeList'></null></setvar>`

  The view is copied multiple times in a `TaskInstance` object, so large views greatly increase the size of each `TaskInstance` and corresponding `TaskResult`.

- Use `resultLimit` (in seconds) in the `TaskDefinition`, or set this option during task execution to quickly dispose of completed tasks. Large numbers of `TaskInstances` impact the following:
  - How `taskResults.jsp` in footers and some JSP™ tasks in the Administrator interface are displayed
  - How JSP tasks are displayed
  - Querying each `TaskInstance` for task renaming
  - Database size

  Set the following options as needed:
  - (*Preferred selection*) `delete` — Causes an older `TaskInstance` of the same name to be deleted before the new task begins execution.
  - `wait` — Suspends the current `TaskInstance` until the older `TaskInstance` is deleted or expired due to reaching its `resultLimit`.
  - `rename` — Inserts a time stamp into the `TaskInstance` name to avoid naming collisions.
  - `terminate` — Deletes an older `TaskInstance` of the same name. Each currently executing `TaskInstance` of the same name is terminated.

Tuning `WorkItems` (`ManualActions`)

The number and size of `WorkItems` (indicated by `ManualActions` in a workflow) can affect memory and system performance *significantly*. By default, Identity Manager copies an entire workflow context into a `WorkItem`, then writes the workflow context back out after submission.

To improve performance for `WorkItems` and `ManualActions` do the following:

- Reduce the size of `WorkItems`.

  By default, `ManualAction` creates a `WorkItem`, then copies each variable in the task context into `WorkItem.variables`. Limiting task context variables prevents overwrites from parallel approvals.

  - Use `ExposedVariables` to limit which variables are copied back into `WorkItem`. For example:

    ```
    <ExposedVariables><List><String>user ...
    ```

  - Use `EditableVariables` to limit the variables assimilated back into the executing task from `WorkItem`. For example:

    ```
    <EditableVariables><List><String>user ...
    ```

    Remember to include an approval flag, a form button value, and the actual approver's name.

- Change the confirmation page and background processing to improve user interface response time.
  - Create a confirmation `ManualAction` or background `ManualAction`, owned by another user such as Configurator.
  - Set `timeout='-5'` (5 seconds) and `ignoreTimeout='true'` to prevent an error message if a user submits an action after the task is executed and the `WorkItems` are deleted.
- Optimize memory use by setting large attribute values, such as value maps and lists, to `null` on submission or instantiate them as Activity-scoped variables that quickly pass out of scope.
- Shorten the lifetime of finished tasks.
  - Prevent dead-end tasks by ensuring that each `WorkItem` specifies a `Timeout` and that the workflow anticipates a `Timeout` for each `WorkItem`.
  - Consider using the `resultLimit` and `resultOption` options in the `TaskDefinition` to determine how the Scheduler handles a task after the task completes.
    - ❒ Use `resultLimit` to control how many seconds a task is allowed to live after the task has completed. The default is zero (`0`), which means that the task instance will be deleted immediately after task completion.
    - ❒ Use `resultOption` to specify what action to take when repeated instances of a task are started (such as `wait`, `delete`, `rename`, or `terminate`). The default is `delete`.

      If you want to immediately delete tasks that complete successfully, but you also want to keep tasks containing errors long enough to debug, you can conditionally delete finished tasks.

      Set a `resultLimit` in the `TaskDefinition` to a sufficient time period to debug issues. You can set `resultLimit` to zero (or a small value) if no errors are reported at runtime (such as `WF_ACTION_ERROR is <null/>`) after a `WorkflowServices` call.

- Evaluate and fix poorly scoped variables. Scope variables according to where they are declared, as follows:
  - **Global variables** are values that must be used across many activities (such as the case owner, view) and as approval flags in subprocesses.

    If a variable is declared as an element of `<TaskDefinition>`, scope the variable globally. If a variable is declared `external`, its value is resolved up the call stack.
  - **Activity variables** of expensive values (such as those variables that require a resource fetch or that store a large list or map of values) can be referenced in a `WorkItem`.

    If a variable is declared as an element of `<Activity>`, ensure that the variable is visible to actions and transition elements in `Activity`.

    Note – Beginning with Identity Manager Version 2005Q3M1 SP1, use `<Activity>` variable values in Forms, rather than in workflows, to avoid copying values on `WorkItem` creates.

  - **Activity variables** are values used in transition logic.

    If a variable is declared as an element of `<Action>`, the variable should pass out of scope on completion of the action. Action variables are typically used in `WorkflowApplication` invocations to "change" the names of variables set by the application (such as View → User).
- Do not specify synchronous execution (`syncExec='true'`) for the last page in a wizard workflow.

  If set to `true`, the task will run to completion when the user executes the task. The interface will hang until the task completes or encounters another stopping point (such as another `ManualAction`).
- Remove unnecessary approval checks.

  Note – For Active Sync, use a streamlined provisioning task in place of the system-specified provisioning task specified by `viewOptions.Process`.

- Do not modify the provisioning tasks provided with Identity Manager.

  You must create a new task, then identify that task in the form and in the process mappings configuration (unless the task is not listed).

Tuning Database Statistics

As a database administrator, you should frequently run statistics to monitor your repository database.

Performance problems are often caused by bad or missing database table statistics. Fixing this problem improves performance for both the database and Identity Manager performance.

See the following Oracle articles for more information:

- *Understanding System Statistics*
- *Oracle MetaLink: Note:114671.1: Gathering Statistics for the Cost Based Optimizer*
- *Cost Control: Inside the Oracle Optimizer*

Also consider using SQL Profiles, which is another method for choosing the best query plans. You can use the SQL Advisor within Enterprise Manager to create these profiles when you identify poorly performing SQL.

Tuning Data Exporter

Data Exporter enables you to export new, changed, or deleted Identity Manager data to an external repository that is suitable for reporting or analytic work. The actual exporting of data is done in batches, where each type of data to be exported is able to specify its own export cycle. The data to be exported comes from the Identity Manager repository and, depending on the length of the export cycle and the amount of changed data, the volume of exported data can be large.

Some Identity Manager data types are queued into a special table for later export. Specifically, `WorkflowActivity` and `ResourceAccount` data is queued because this data is not persisted otherwise. Any persisted data type can also be queued if the warehouse needs to see all changes to the type, or if the type has a lifecycle that does not correspond to the export cycle, such as `TaskInstance` and `WorkItem` data.

To maximize performance, only queue and export the types of data that you require in the warehouse. Data exporting is disabled by default, but if you enable data exporting, it exports all data types. Turn off any data types that you do not need.

When the export task exports data, the task attempts to complete the export as quickly as possible, using multiple threads to achieve as much throughput as possible. Depending on the I/O speed of the Identity Manager repository and the warehouse, the `export` task can fully utilize the processors on the Identity Manager server, which causes any interactive performance to degrade. Ideally, the export should occur on a machine dedicated to that task or at least occur during periods when there is no interactive activity on the machine.

The export task supports the following tuning parameters:

- Queue read block size
- Queue write block size
- Queue drain thread count

The drain thread count is the most important throughput. If a large number of records are in the queue table, increasing the number of threads (up to 24) tends to increase throughput. However, if the queue is dominated by one type of record, fewer drain threads might actually be faster. The export task attempts to divide the queue table contents into as many sets as there are threads allocated, and to give each thread a set to drain. Note that these threads are in addition to the drain threads that are draining the other repository tables.

Tuning the General XML

You can usually optimize the general XML by using static `XMLObject` declarations wherever possible. For example, use:

- `<List>` instead of `<list>`
- `<String>` instead of `<s>`
- `<Map><MapEntry ...></Map>` instead of `<map>`

Also, depending on the context, you might have to wrap objects instead of using the `<o></o>` element.

Tuning Sun Identity Manager Service Provider

You can use Identity Manager dashboard graphs to quickly assess the current system, spot abnormalities, and understand historical trends (such as concurrent users or resource operations over a time period) for Sun™ Identity Manager Service Provider(Service Provider).

Note – Service Provider does not have an Administrator interface. You use the Identity Manager Administrator interface to perform almost all administrative tasks (such as viewing dashboard graphs).

For more information about tuning Service Provider see *Sun Identity Manager Service Provider 8.1 Deployment*.

Tuning the Identity Manager Web Interface

When you are working with the Identity Manager Web Interface, you can optimize performance by using the OpenSPML toolkit that is co-packaged with Identity Manager.

Note – Using the `openspml.jar` file from the `http://openspml.org/` web site might cause memory leaks.

Tuning Initial Loads

To improve performance during a large, initial user load, follow this procedure:

1. Disable all Audit Events from the Identity Manager Administrator interface.

   Note – Audit Logging can add several records per operation, making future audit reports perform more slowly.

   a. Choose Configure → Audit.

   b. On the Audit Configuration page, deselect the Enable auditing box and click Save.

2. Disable the list cache by shutting down the web server or by changing the `ChangeNotifier.updateSearchIntervalCount` property (on the `debug/Show_WSProp.jsp` debug page) to **`0`**..

   The list cache keeps lists of users in frequently accessed organizations in memory. To maintain these lists, the list cache searches for and checks all newly created users.

3. Clear the current list cache on the `debug/Clear_List_Cache.jsp` page.
4. Ensure that the workflow being used to process the users does not contain approvals.
5. Use alternative load methods, which include:
   - Splitting the load and running the data in zones
   - Using bulk loads, which are much faster
   - Loading from a file
6. Disable Data Exporter for the `WorkflowActivity` type.

Tuning Memory Requirements

You must determine your memory needs and set values in your application server's JVM by adding maximum and minimum heap size to the Java command line. For example:

```
java -Xmx512M -Xms512M
```

To improve performance do the following:

- Set the maximum and minimum heap size values to the same size.
- Depending on your specific implementation, you might want to increase these values if you run reconciliation.
- For performance tuning purposes, you may also set the following in the `waveset.property` file:

  ```
  max.post.memory.size value
  ```

Note – The `max.post.memory.size` specifies the maximum number of bytes that a posted file (for example by using an `HTML FileSelect` control) might contain without being spooled to the disk. For cases where you do not have permission to write to temporary files, increase the `max.post.memory.size` to avoid having to spool to the disk. The default value is 8 Kbytes.

For additional information about system requirements, see the *Sun Identity Manager 8.1 Release Notes*.

Tuning Operating System Kernels

For information about tuning Solaris™ and Linux operating system kernels, see the "Tuning the Operating System" chapter in the *Sun Java System Application Server Enterprise Edition Performance Tuning Guide.*

For information about tuning Oracle operating system kernels, see the product documentation provided with your Oracle system.

Tuning Provisioner

Network latency tends to be a common cause for performance issues when dealing with view provisioning. Tracing individual resource adapters can help you determine what is causing performance problems.

To improve provisioner performance, do the following:

- Set `provisioner.maxThreads` in the `Waveset.properties` file to control the number of simultaneous account provisioning threads where a thread is started for each resource operation.

  Generally, you can achieve optimal performance by setting this value to **`10`**. Specifying a value greater than 20 significantly degrades the provisioner's performance.

- Configure quota settings in the `Waveset.properties` file to control the number of concurrent operations (such as reprovisioning) a user can execute for a specific task. Increasing the number of concurrent actions can help more operations complete faster, but trying to process too many actions at once might cause bottlenecks.

  You can create configuration sets on a per-pool basis. For example, if you create configuration A, configuration B, and configuration C, when you create a `TaskDefinition` (workflow), you can assign a specific pool configuration to the workflow from the configurations that you defined.

  The following example shows the quota settings that limit user `bob` to running one reprovisioning task at a time:

  ```
  Quota.poolNames=ReProvision,Provision
  Quota.pool.ReProvision.defaultLimit=1
  Quota.pool.ReProvision.unlimitedItems=Configurator
  Quota.pool.ReProvision.items=bob,jan,ted
  Quota.pool.ReProvision.item.bob.limit=1
  ```

  To enforce the task quota, reference `poolName` in a `TaskDefinition`. The format is as follows:

  ```
  <TaskDefinition ... quotaName='{poolName}'..>
  ```

  Most users start only one task at a time. For proxy administrators who perform reconciliation or Active Sync tasks, set the task quota higher.

Note – Avoid using the Configurator user for reconciliation and Active Sync tasks. The `Configurator` has access to unlimited tasks and can monopolize available resources, which adversely affects concurrent processes.

Tuning Reconciliation

The Reconciler is the Identity Manager component that performs reconciliation. This section suggests methods for improving Reconciler performance, including:

- "General Suggestions for Tuning Reconciliation" on page 61
- "Tuning the Reconciler Server Settings" on page 62

■ "Tuning Reconciliation for Multiple Resources" on page 63

General Suggestions for Tuning Reconciliation

In general, you can improve Reconciler performance if you do the following:

- Avoid using the Configurator user for reconciliation tasks. The Configurator has access to unlimited tasks and can monopolize available resources, which adversely affects concurrent processes.

  Instead, use a streamlined, minimal user for reconciliation and Active Sync tasks. Because the subject executing the task is serialized as part of the task, a minimal user takes less space, or overhead, for each task and update in the repository.
- Use information on the Reconciler status page (`debug/Show_Reconciler.jsp`) to decide which settings to adjust based on queue sizes, available system resources, and performance benchmarks. Be aware that these settings are dependent on the environment.
- Use the System Memory Summary page (`debug/Show_Memory.jsp`) to see how much total and free memory is available. Reconciliation is a memory-intensive function, and you can use this information to determine whether there is sufficient memory allocated to the JVM. You can also use this page to launch garbage collection or to clear unused memory in the JVM for investigating heap usage.
- When you assign user forms to proxy administrators who are performing reconciliations, keep the user forms as simple as possible and only use essential fields. Depending on the schema map, including a field that calculates the `waveset.organization` attribute is generally sufficient.

  Note – Administrators who need to view or edit the Identity Manager schema for Users or Roles must be in the `IDM Schema Configuration AdminGroup` and must have the `IDM Schema Configuration` capability.

- Use per-account workflows judiciously. The reconciliation process does not start provisioning tasks for performance reasons by default.

  If you must use a per-account workflow task, edit the reconciliation policy to limit the Reconciler's automatic responses to events of interest only. (See the Situation area of the Edit Reconciliation Policy page.)

Tuning the Reconciler Server Settings

Although the default settings are usually adequate, you can sometimes improve Reconciler performance if you adjust the following settings on the Edit Server Settings page:

- **Parallel Resource Limit**. Specifies the maximum number of resource threads that the Reconciler can process in parallel.

  Resource threads allocate work items to worker threads, so if you add additional resource threads, you might also have to increase the maximum number of worker threads.
- **Minimum Worker Threads**. Specifies the number of processing threads that the Reconciler always keeps open.
- **Maximum Worker Threads**. Specifies the maximum number of processing threads that the Reconciler can use. The Reconciler starts only as many threads as the workload requires, which places a limit on that number. Worker threads automatically close if they are idle for a short duration.

During idle times, the threads stop if they have no work to do, but only down to the minimum number of threads specified. As the load increases, the Reconciler adds more threads until the maximum number of threads is reached. The Reconciler never has less than the minimum number of threads or more than the maximum.

Generally, more threads allow more concurrency. However, at some point, too many threads can put too much load on the machine or just do not provide additional benefit.

Note – Recommending generic, optimal settings is not possible because deployments are so different. Reconciler settings must be adjusted differently for each deployment environment.

▼ To Change the Reconciler Server Settings

Perform the following steps to change the Reconciler server settings:

1. **Log into the Administrator interface.**
2. **Click the Configure → Servers → Reconciler tabs.**
3. **When the Edit Server Settings page is displayed, adjust the settings as necessary.**

   See "Editing Default Server Settings" on page 19 for more information.

Tuning Reconciliation for Multiple Resources

If you are configuring reconciliation for multiple resources in Identity Manager, you have several options:

- All of the resources on the same server, all at the same time.

  This option is the most efficient from the Identity Manager perspective, but if you have many resources (for example more than 20), you are likely to experience Java resource issues.
- All of the resources on the same server, each at a different time.

  This option is easier on Java resource loading, but puts a significant burden on your schedule configuration.
- Each resource on a different server, all at the same time.

  This option minimizes elapsed time, but increases the number of servers.

An ideal solution does not exist for this configuration because deployments are so different. You might have to mix and match these options to find an acceptable solution for your deployment.

Preparing a usage survey, based on the business reasons behind this functionality, might help you decide how to proceed.

Address these questions:

- Why are you reconciling these resources?
- Do you have the same the goal for each of these resources?
- Are each of these resources equally important or critical?
- Must all resources be reconciled on the same schedule, or can you spread out the reconciliations?
- How often must each resource be reconciled?

Also, remember that the reconciliation server does not have to be one of the pools that handles web traffic. You can add a server that you never interact with directly because this server exists solely for transaction processing. Having a server dedicated to transaction processing might make the first option more attractive for very large systems.

Tuning Resource Queries

Note – Network latency tends to be a common cause of performance issues during view provisioning. Tracing individual resource adapters can help you determine what is causing performance problems.

You can improve resource query performance if you use `FormUtil.getResourceObjects` to implement the query.

Use one of the following methods to cache query results:

- `getResourceObjects(Session session, String objectType, String resID, Map options, String cacheList, String cacheTimeout, String cacheIfExists)`
- `getResourceObjects(String subjectString, String objectType, String resId, Map options, String cacheList, String cacheTimeout, String clearCacheIfExists)`

Note –

- Set `cacheTimeout` in milliseconds.
- Restrict searches to specific `searchContext`, if applicable.
- Return the minimum number of attributes in `options.searchAttrsToGet`.

Tuning the Scheduler

The Scheduler component controls task scheduling in Identity Manager.

This section suggests methods for improving Scheduler performance, including:

- "General Suggestions for Tuning the Scheduler" on page 64
- "Tuning the Scheduler Server Settings" on page 65

General Suggestions for Tuning the Scheduler

The following `TaskDefinition` options determine how the Scheduler handles tasks after they are completed:

- `resultLimit` — Controls how many seconds a task is allowed to run after the task has completed. The default setting varies for different tasks. A setting of zero immediately removes tasks after completion.
- `resultOption` — Controls what action is taken when repeated instances of a task are started. The default setting is `delete`, which removes extra task instances.

These default settings are designed to optimize memory by shortening the lifetime of finished Scheduler tasks. Unless there is a compelling reason to change these settings, use the defaults.

If you want to immediately delete tasks that completed successfully, but you also want to keep tasks containing errors long enough to debug, you can do the following:

- Conditionally delete finished tasks by setting the `resultLimit` value to a time period that is sufficient for debugging issues.
- Set the `resultLimit` to zero (or a small value) if no errors (such as `WF_ACTION_ERROR is <null/>`) are reported at runtime after a `WorkflowServices` call.

Tuning the Scheduler Server Settings

You can sometimes improve Scheduler performance by adjusting the following settings on the Edit Server Settings page:

- **Maximum Concurrent Tasks**. Specifies the maximum number of tasks that the Scheduler can run at one time.

  When more tasks are ready to run than the Maximum Concurrent Tasks setting allows, the extra tasks must wait until there is room available or until they are run on another server.

  If too many tasks are being swapped out of memory and sharing CPU time, the overhead slows down performance. Alternatively, setting the maximum too low results in idle time. The Scheduler checks for available tasks every minute, so a waiting task waits at least a minute before being run.

  The default Maximum Concurrent Tasks setting (`100`) is usually adequate. You can decide whether to adjust this setting up or down based on which tasks are being run in the deployment and by profiling the runtime behavior after the deployment is otherwise complete.

  In some cases, you might want to suspend or disable the Scheduler. For example, if you want a server dedicated to handling the End User interface, disabling the Scheduler will prevent tasks from running on that server. The server would only serve the End User interface pages and store launched tasks for other servers to execute.
- **Task Restrictions**. Specifies the set of tasks that can execute on the server.

  The Task Restrictions setting can provide a finer granularity of control over what tasks are allowed to run on a server. You can restrict tasks individually or through the server settings.

Note – Recommending generic, optimal settings is not possible because deployments are so different. Scheduler settings must be adjusted differently for each deployment environment.

▼ To Change the Scheduler Server Settings

1. **Log in to the Administrator interface.**
2. **Click the Configure → Servers → Scheduler tabs.**
3. **When the Edit Server Settings page is displayed, adjust the settings as necessary.**
   See "Editing Default Server Settings" on page 19 for more information.

Tuning Sessions

Identity Manager maintains a least recently used (LRU) cache of authenticated sessions for use by authenticated users. By using existing authenticated sessions, you can speed up repository access for objects and actions that require a session.

To optimize the authentication pool size, change the `session.userPoolSize` value in the `Waveset.properties` file to the maximum number of expected, concurrent user sessions on the server.

Tuning the Sun Identity Manager Gateway

The Sun Identity Manager Gateway generates a thread for each connection, and uses a different pool for each unique combination of resource type, Gateway host, and Gateway port. The Gateway checks for idle connections every five minutes. When a connection has been idle for 60 seconds, the Gateway closes and removes that connection from the pool.

When the Gateway receives a request, it does the following:

- If there are no idle connections in the corresponding pool, the Gateway creates a new connection.
- If an idle connection exists in the pool, the Gateway retrieves and reuses that connection.

You must configure the maximum number of connections on the resource, and you must configure these connections the same way for all resources of the same type, that are using the same Gateway. For that resource type, the first connection made to the Gateway on a given host and port uses that resource's maximum connections value.

Note – When you change the maximum number of connections on a resource, you must start and stop the server for the change to take effect.

The following example shows how connections, requests, and Gateway threads are related.

If you set the maximum number of connections to 10 on an Active Directory resource, and you are using two Identity Manager servers, then you can have up to 20 simultaneous connections (10 from each Identity Manager server) to the Gateway for that Active Directory resource. The Gateway can have 10 simultaneous requests outstanding from each server, and the Gateway processes each request on a different thread. When the number of simultaneous requests exceeds the maximum number of Gateway connections, additional requests are queued until the Gateway completes a request and returns the connection to the pool.

Note – Although the Gateway code is multi-threaded, this characteristic does not address the APIs or services being used by the Gateway. For Active Directory, the Gateway uses the ADSI interface provided by Microsoft. No investigation has been done to determine whether this interface handles Gateway requests in parallel.

Other methods for improving Gateway performance, include:

- Locating the Gateway near (from a network connectivity perspective) the domain controllers of the managed domain
- Increasing the block size on a Gateway resource can increase throughput during reconciliation or load operations

  Increased throughput results have been noted for basic reconciliations with no custom workflows and in which no attribute reconciliations are being performed. Initially, the Gateway does consume more system memory, but this memory is eventually released.

  Be aware that there is a diminishing return. At some point, larger block sizes do not result in proportionately increased performance. For example, the following data shows the speed observed for a Load from Resource of 10,000 users from an Active Directory resource. Also, the peak memory usage for the Gateway process during the load is included.

Block Setting	Users Created Per Hour	Peak Gateway Memory Usage
100	500	20 MB
200	250	25 MB
500	9690	60 MB
1000	10044	92 MB

- For Exchange Server 2007, the `PowerShellExecutor` performs actions for Exchange Server 2007. You can modify the following registry settings to change the behavior of the `PowerShellExecutor` inside the Gateway.

Note – Both settings can have a large impact on the behavior and memory usage of the Gateway. Changes to these parameters should only be considered after careful testing.

- `powerShellTimeout`
  - **Content**. Timeout for PowerShell actions (registry type `REG_DWORD`)
  - **Default**. 60000 ms (1 minute)

    When the `powerShellTimeout` setting times out, any RunSpace actions are interrupted and canceled to prevent runaway actions in the PowerShell environment that cause the Gateway to become unresponsive.

    Decreasing the `powerShellTimeout` value to a small value can prematurely cancel actions, and can prevent the RunSpace initialization from finishing correctly. Observed startup times for the first RunSpace in the pool range from 2—5 seconds.

    The `powerShellTimeout` value is read-only on startup, and you cannot change it without restarting the gateway.
- `runSpacePoolSize`
  - **Content**. Number of RunSpaces in the pool (registry type `REG_DWORD`)
  - **Default**. 5
  - **Minimum**. 5
  - **Maximum**. 25

    The number of RunSpaces in the pool allow for parallel execution of PowerShell actions by the gateway. One provisioning action or update of a user in Exchange 2007 can result in multiple PowerShell actions being executed.

    A started RunSpace can consume a large amount of memory. For the first RunSpace, the typical size is approximately 40 MB. Subsequent RunSpaces normally use between 10—20 MB.

    Note – The preceding figures can differ in specific environments and are only given as *guidelines*, so be careful when changing this value.

    The `runSpacePoolSize` value is read-only on startup, and you cannot change the pool size value without restarting the Gateway.

Tuning the Task Bar

The Administrator interface task bar displays links to previously performed provisioning tasks, which causes the interface to render more slowly when there are a large number of tasks.

To improve interface performance, remove the `taskResults.jsp` link from each JSP by deleting the `<List>...</List>` element from the `UserUIConfig` object.

The following example shows `<List>...</List>` entries within `<TaskBarPages>`.

EXAMPLE 4–1 Modifying the `UserUIConfig` Object

```
<TaskBarPages>
  <List>
    <String>account/list.jsp</String>
    <String>account/find.jsp</String>
    <String>account/dofindexisting.jsp</String>
    <String>account/resourceReprovision.jsp></String>
    <String>task/newresults.jsp</String>
    <String>home/index.jsp</String>
  </List>
 </TaskBarPages>
```

Debugging Performance Issues

This section describes the different Identity Manager and third-party debugging tools you can use to debug performance issues.

The information is organized into the following sections:

- "Working With Identity Manager Debug Pages" on page 69
- "Working With Other Debugging Tools" on page 75

Working With Identity Manager Debug Pages

Note – Tracing affects system performance. To help ensure optimal performance, specify the minimum tracing level or turn tracing off after debugging the system.

This section provides instructions for accessing the Identity Manager Debug pages and describes how to use these pages to identify and debug Identity Manager performance issues.

See the following sections for information:

- "Accessing the Debug Pages" on page 70
- "Control Timings (`callTimer.jsp`)" on page 71
- "Edit Trace Configuration (`Show_Trace.jsp`)" on page 71
- "Host Connection Pool (`Show_ConnectionPools.jsp`)" on page 72
- "List Cache Cleared (`Clear_XMLParser_Cache.jsp`)" on page 72

Accessing the Debug Pages

Note – You must have the *Debug*, *Security Administrator*, or *Waveset Administrator* capabilities to access and execute operations from the Identity Manager Debug pages. Administrators and Configurator are assigned this capability by default.

If you do not have the Debug capability, an error message results.

▼ To Access the Identity Manager Debug Pages

1. **Open a browser and log in to the Administrator interface.**

2. **Type the following URL:**

   `http://` *host*`:`*port* `/idm/debug`

   where:

   - *host* is the application server on which you are running Identity Manager.
   - *port* is the number of the TCP port on which the server is listening.

3. **When the System Settings page displays, type the `.jsp` file name for the debug page you want to open.**

   For example:

   `http://` *host*`:`*port* `/idm/debug/`*pageName.jsp*

Note – Some debugging utilities are not linked from the System Settings page, but you can use them to enhance your ability to gather data for product performance and usability. For a complete list of debug pages, open a command window and list the contents of the `idm/debug` directory.

Control Timings (`callTimer.jsp`)

Use the Control Timings page to collect and view call timer statistics for different methods. You can use this information to track bottlenecks to specific methods and invoked APIs. You can also use options on the Call Timings page to import or export call timer metrics.

Note – Call timing statistics are only collected while trace is enabled.

▼ To View Call Timer Statistics

1. **Open the Control Timings page, and click Start Timing & Tracing to enable trace and timing.**
2. **To stop the timing, click Stop Timing & Tracing or click Stop Timing.**

   The page re-displays and populates the Show Timings table with a list of methods for which statistics are available and the methods' aggregate call timer statistics (*not broken down by caller*).

   This table contains the following information:

   - Method name (Click a method name to see which methods it calls)
   - Total time
   - Average time
   - Minimum time
   - Maximum time
   - Total calls
   - Total errors

3. **To clear the list, click Clear Timing.**

Note – You can also use the `callTimer` command to collect call timer data from the Console. This command is useful when you are debugging performance issues during an upgrade or in other situations where Identity Manager is not running on the application server.

Edit Trace Configuration (`Show_Trace.jsp`)

Use the Edit Trace Configuration page to enable and configure tracing for the Java classes provided with your Identity Manager installation.

Specifically, you can use this page to configure the following trace settings:

- Choose methods, classes, or packages to trace and specify the level of trace you want to capture.
- Send trace information to a file or to standard output.
- Specify the maximum number of trace files to be stored and the maximum size for each file.
- Specify how dates and times are formatted in the trace output file.
- Specify the maximum number of methods to be cached.
- Indicate how to write data to the trace file.

  Write data to the trace file as the data is generated, or queue the data and then write it to the file.

Host Connection Pool (`Show_ConnectionPools.jsp`)

If you are not using a data source, you can use the Host Connection Pool page to view connection pool statistics. These statistics include the pool version, how many connections were created, how many are active, how many connections are in the pool, how many requests were serviced from the pool, and how many connections were destroyed.

You can also use the Host Connection Pool page to view a summary of the connection pools used to manage connections to the Gateway. You can use this information to investigate low-memory conditions.

List Cache Cleared (`Clear_XMLParser_Cache.jsp`)

Use the List Cache Cleared page to clear recently used XML parsers from the cache and to investigate low memory conditions.

Method Timings (`Show_Timings.jsp`)

Use the Method Timings page to quickly detect and assess hotspots at a method level.

The following information is gathered from Identity Manager methods and displayed on the Method Timings page:

- Method names
- How many times the methods were called
- How many times the methods exited with an error status
- Average time consumed by the methods
- Minimum and maximum times consumed by invocations of each method

The Method Timings page also contains a table with the following links. You can click these links to view additional information.

- **Details**. Shows call stack information.
- **History**. Shows a graph of call duration compared with the time of the most recent calls.
- **History data**. Shows a list of the most recent calls, showing what time the call was made and the duration of the call.

  Identity Manager does not keep stack history by default. To keep stack history and to control its depth, edit `Waveset.properites` and look at the `MethodTimer` keys.

Note – The Clear ALL option on the Method Timings page clears all results. This option is enabled by default.

Object Size Summary (`Show_Sizes.jsp`)

Use the Object Size Summary page to detect problematically large objects that can affect your system.

The Object Size Summary page shows information about the size of objects (in characters) stored in the repository. These objects are listed by type, along with the total number of objects of each type, and the objects' total combined size, average size, maximum size, and minimum size.

Click an entry in the Type column to view additional size information about that object type. For example, click Configuration to view the ID, name, and size of the largest configuration objects in the repository.

You can also access this size information from the Console command line.

▼ To Access Object Size Information from the Command Line

1. **Open the console.**
2. **At the command prompt, type:**

   `showSizes [ `*`type`*`[`*`limit`*` ]]`

Note – For upgrades, existing objects will report a size of 0 until they have been updated or otherwise refreshed.

Provisioning Threads for Administrator Configurator (`Show_Provisioning.jsp`)

Use the Provisioning Threads for Administrator Configurator to view a summary of the provisioning threads in use by the system. This summary is a subset of the information available in `Show_Threads.jsp`.

Note – Looking at just a single thread dump can be misleading.

System Cache Summary (`Show_CacheSummary.jsp`)

Use the System Cache Summary page to view information about the following items to help you investigate low-memory conditions:

- Administrator-associated object caches
- System object cache
- User login sessions
- XML parser cache

System Memory Summary (`Show_Memory.jsp`)

Use the System Memory Summary page to view how much total and free memory you have available in Mbytes. When you are using memory-intensive functionality such as Reconciliation, this information can help you determine whether there is sufficient memory allocated to the JVM.

You can also use this page to launch garbage collection or to clear unused memory in the JVM for investigating heap usage.

System Properties (`SysInfo.jsp`)

The System Properties page provides information about your environment, including software versions, paths and environmental variables.

System Threads (`Show_Threads.jsp`)

Use the System Threads page to see which processes are running so you can verify that automated processes (such as reconciliation or Active Sync) are running.

This page includes information about the process type, process name, its priority, if the process is a daemon, and if the process is still running.

Note – Looking at just a single thread dump can be misleading.

User Session Pool Cleared (`Clear_User_Cache.jsp`)

Use the Session Pool Cleared page to clear all of the cached sessions for users who have recently logged in and to investigate low memory conditions.

Waveset Properties (`Show_WSProp.jsp`)

Use the Waveset Properties page to view and *temporarily* edit properties in the `Waveset.properties` file. You can test different property settings for a particular server on which the `Waveset.properties` file resides without having to restart the server to pick up the changes. The edited property settings only remain in effect until the next time you restart the server.

XML Resource Adapter Caches Flushed and Cleared (`Clear_XMLResourceAdapter_Cache.jsp`)

Use the XML Resource Adapter Caches Flushed and Cleared page to clear test XML resource adapters from the cache and to investigate low memory conditions.

Working With Other Debugging Tools

You can use the following Sun Microsystems' and third-party tools to identify potential performance bottlenecks:

- "Identity Manager Profiler" on page 75
- "Using DTrace" on page 76
- "Using JMX" on page 77
- "Using JConsole" on page 79
- "Using JRat" on page 80

These tools can be particularly useful if your deployment uses custom Java classes.

Identity Manager Profiler

Identity Manager provides a Profiler utility to help you troubleshoot performance problems in your deployment.

Customized forms, Java, rules, workflows, and XPRESS can cause performance and scale problems. The Profiler profiles how much time is spent in these different areas, enabling you to determine whether these forms, Java, rules, workflows, or XPRESS objects are contributing to performance and scale problems and, if so, which parts of these objects are causing the problems.

Note – For more information about Profiler, see "Working with the Identity Manager Profiler" in *Sun Identity Manager 8.1 Release Notes*.

Using DTrace

The DTrace facility is a dynamic tracing framework for the Solaris 10 operating system that enables you to monitor JVM activity.

DTrace contains more than 30,000 probes and uses integrated user-level and kernel-level tracing to give you a view into your production system. You can also trace arbitrary data and expressions by using the D language, which is similar to C or `awk`. The DTrace facility also includes special support for monitoring the JVM, and enables you to watch your whole system and span outside the JVM.

DTrace is easiest to use with Java 6 because probes are built into the JVM. The facility also works with Java 1.4 and Java 5, but you must download JVM PI or JVM TI agents from the following URL:

`https://solaris10-dtrace-vm-agents.dev.java.net/`

The following example shows how to write a DTrace script.

EXAMPLE 4–2 Example DTrace Script

```
#!/usr/sbin/dtrace -Zs
#pragma D option quiet
hotspot$1:::
{
  printf("%s\n", probename);
}
```

In this example, you would replace *$1* with the first argument to the script, which is the PID of the Java process you want to monitor. For example:

```
# ./all-jvm-probes.d 1234
```

The following table describes the commands you can use to enable different DTrace probes.

TABLE 4–3 DTrace Commands

Command	Description
`-XX:+DTraceMonitorProbes`	Enables JVM support in Java 6 (patches for Java 1.4 and Java 5)

TABLE 4–3 DTrace Commands *(Continued)*

Command	Description
`-XX:+ExtendedDTraceProbes`	Provides the following information: ■ JVM startup (begin and end) and shutdown ■ Thread starting and stopping ■ Class loading and unloading ■ Garbage collection (several options available) ■ JIT compilation begin and end ■ Compiled method loading and unloading ■ Monitor contention, wait, and notify ■ Method entry, method return, and object allocation
`/usr/sbin/dtrace -n 'hotspot*:::'`	Enables all JVM probes for all Java processes on the system
`/usr/sbin/dtrace -n 'hotspot1234:::'`	Enables all JVM probes for only the Java process with PID *1234*
`/usr/sbin/dtrace -n 'hotspot1234:::gc-begin'`	Enables only the probe that starts when garbage collection for process *1234* begins

Note – Because DTrace causes additional work in the system, enabling this facility affects system performance. The effect is often negligible, but can become substantial if you enable many probes with costly enablings.

Instructions for minimizing the performance effect of DTrace are provided in the "Performance Considerations" chapter of the *Solaris Dynamic Tracing Guide*.

For more information about DTrace, see `/usr/demo/dtrace` and `man dtrace`.

Using JMX

Identity Manager enables you to use Java Management Extensions (JMX) to capture and expose operational statistics for certain resource adapter operations. You can use this data for diagnostic and predictive purposes, such as to monitor system health and reports.

This statistical data includes the following:

- The number of times the action was performed
- The minimum, maximum, and average duration of the operations

Objects	Actions Monitored
For Accounts	■ Create ■ Update ■ Delete ■ Get ■ Authenticate
For Actions	Run
For Other Objects	■ Create ■ Update ■ Delete ■ Get ■ List

JMX creates MBeans for each resource adapter, by server, and registers these beans with names that match the following pattern:

`serverName=`*server name*`, resourceAdapterType=`*Resource Adapter Type*`,`
`resourceAdapterName=`*Resource Adapter Name*

Identity Manager records statistics for all completed operations, whether they completed successfully or with errors. However, Identity Manager does not record statistics for incomplete operations, such as any operations that throw exceptions.

You can configure `excludes` as follows:

1. From the Administrator interface, select Configure → Servers.
2. On the Configure Servers page, perform one of the following tasks:
   - Click the Edit Default Server Settings button to edit the default server settings.
   - Click a server link to edit the policy for that server.
3. Click the JMX tab and enable the JMX Enable Resource Adapter Monitor box to turn on resource monitoring.
   - To exclude specific resources, add regular expressions to the JMX Resource Adapter Monitor Excludes list.
   - To exclude monitoring specific actions, add regular expressions to the JMX Resource Adapter Monitor Operation Excludes list.

All `excludes` use regular expressions. For excluding certain resources, JMX just matches on the resource name. For example, if you have adapters named

```
resource1
resource2
```

```
resource3
resource10
resource11
```

and you specify the following pattern

```
.*1$
```

which means, match any 0 or more of any character (`.*`) until something that ends with a `1` (`1$`). JMX will exclude `resource1` and `resource11`.

For operations, the process is similar. If your operations have the following names, the patterns must match against those names.

```
ACCOUNT_CREATE
ACCOUNT_UPDATE
ACCOUNT_DELETE
ACCOUNT_GET
ACCOUNT_AUTHENTICATE
OBJECT_CREATE
OBJECT_UPDATE
OBJECT_DELETE
OBJECT_GET
OBJECT_LIST
ACTION_RUN
```

For example, the `^ACCOUNT.*` pattern excludes all operations that start with `ACCOUNT`. Or, using this pattern excludes updates and deletes:

```
.*UPDATE$
.*DELETE$
```

Note – For more information about configuring and using JMX, see "Configuring JMX Monitoring" on page 17 and "The JMX Publisher Type" in *Sun Identity Manager 8.1 Business Administrator's Guide*.

Using JConsole

The Java Monitoring and Management Console (*JConsole*) is a Java Management Extension (JMX) technology-compliant graphical management tool that is co-packaged with at least JDK 5. JConsole connects to a running JVM and gathers information from the JVM MBeans in the connected JMX agent.

Specifically, you can use JConsole to perform the following tasks:

- Detect low memory and deadlocks

  JConsole accesses the memory system, memory pools, and MBeans garbage collector to provide information about memory use, including memory consumption, memory pools, and garbage collection statistics.
- Enable or disable garbage collection
- Enable or disable verbose tracing
- Monitor local and remote applications
- Monitor and manage MBeans including current heap memory use, non-heap memory use, and how many objects are pending for finalization
- View information about performance, resource consumption, and server statistics
- View summary information about the JVM and monitored values, threads running on the application, and loaded classes
- View information about operating system resources (Sun's platform extension), such as:
  - CPU process time
  - How much total and free physical memory is available
  - The amount of committed virtual memory (how much virtual memory is guaranteed to be available to the running process)
  - How much total and free swap space is available
  - The number of open file descriptions (UNIX®only)

Note – For more information about using JConsole to monitor applications on the Java platform, see the Sun Developer Network (SDN) article titled *Using JConsole to Monitor Applications*, which is available from the following URL:

`http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

Identity Manager supplies some JMX MBeans that provide information about the following:

- Identity Manager Server Cluster
- Data Exporter
- Scheduler

Using JRat

You can use the Java Runtime Analysis Toolkit (JRat), an open-source performance profiler for the Java platform, to identify potential performance bottlenecks, especially if your deployment uses custom Java classes. JRat monitors your application's execution and persists the application's performance measurements.

For example, if you have a custom workflow for provisioning, you can use JRat to see which classes are being invoked and how much time is required to run your workflow compared to the default Identity Manager provisioning workflow.

For more information about JRat, see `http://jrat.sourceforge.net`.

◆ ◆ ◆ CHAPTER 5

Tracing and Troubleshooting

This chapter explains how you can use tracing to help fix errors, solve performance issues, and understand the flow within Identity Manager. This chapter also describes methods for troubleshooting problems that might occur with different Identity Manager components.

This chapter covers the following topics:

Before You Begin Tracing or Troubleshooting

Review the information in the following sections before tracing or troubleshooting Identity Manager:

Intended Audience

This chapter is intended for application server and database administrators, front-line support engineers, and partners who are responsible for maintaining Identity Manager in a deployment environment.

Before troubleshooting problems with Identity Manager, you must

- Be familiar with Java 5.0 (required for Sun Identity Manager 8.1)
- Understand the components that you are trying to troubleshoot

Important Notes About Tracing and Troubleshooting Identity Manager

Before you start tracing or troubleshooting Identity Manager, note the following:

- Tracing affects system performance. To ensure optimal performance, use the minimum tracing level needed or turn off tracing after debugging the systems.
- Do not enable tracing for the `com.waveset` class. The `com.waveset` class is verbose and has many classes, so tracing this class might cause your server to hang.
- Do not configure Identity Manager to trace at the advanced level unless instructed to do so by Sun Support.
- If you set `exception.trace` in the `Waveset.properties` file to debug a specific problem, do not use it for an extended period of time and be sure you disable it after debugging. Setting `exception.trace` in the `Waveset.properties` file can significantly affect system performance.

Before Calling Support

Before you call Sun Technical Support for assistance with a problem, consider the following suggestions for narrowing the problem area:

- Try using web browser search tools to research the problem.
- Use resource-specific support options when available. Your problem might be related to a resource and fixed using a resource patch.
- Remove Identity Manager from the equation by using an appropriate client tool when possible. For example, by using a Java LDAP browser.

Also, you must collect and have the following information available before calling Sun Support for assistance.

Information to Collect	How to Get this Information
Product version, including ■ A list of all installed patches, hotfixes, and e-fixes ■ A list of customizations	Use the following commands from the Identity ManagerConsole: ■ `installed` ■ `inventory`
Identity Manager topography, including: ■ How your Identity Manager cluster is configured ■ Environment localization of the issue and any extra information about your environment ■ A list of servers	The information ■ Must be manually determined ■ Must be manually determined ■ Can be obtained by selecting Configure → Servers from the Identity Manager Administrator interface on a running Identity Manager server
Recent changes to your environment	Must be manually determined
Java version and type	Use the following Java command: `java -version`
Application server version and type	Depends on the Application Server being used, but must be manually determined
Operating system level and information	Must be manually determined
Default XML output, which is a list of all repository objects	Use the `export default.xml default` command from the Console
Task instance data, including ■ A list of all running tasks ■ The size of all current task instances	Use the following commands from the Console: ■ `listTasks` ■ `showSizes TaskInstance`
System Logs	**Note –** You might be asked to provide additional system logs based on the type of issue you are reporting.

Related Documentation and Web Sites

In addition to the information provided in this chapter, consult the documents and web sites listed in this section for information related to tracing and troubleshooting Identity Manager.

Recommended Reading

See the following documents for information related to tracing and troubleshooting Identity Manager:

- "Testing Your Customized Form" in *Sun Identity Manager Deployment Reference* for the recommended method of tracing XPRESS functions.
- The article titled *Using JConsole to Monitor Applications* for information about using JConsole to monitor applications that run on the Java platform. This article is available from the following URL:

  `http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

Useful Web Sites

The following table describes web sites related to troubleshooting Identity Manager.

TABLE 5–1 Useful Web Sites

Web Site URL	Description
`http://sunsolve.sun.com/`	Sun web site containing diagnostic tools, forums, features and articles, security information, and patch contents. **Note**: The information on this site is divided into three areas: ■ **Internal**. Available only to Sun employees ■ **Contract**. Available only to customers with contract access ■ **Public**. Available to everyone
`http://www.sun.com/service/online/us`	Sun Technical Support web site.
`https://sharespace.sun.com/gm/folder-1.11.60181?`	Identity Manager folder on Sun Microsystems' Share Space, which contains Identity Manager's FAQ, links to forums, features and articles, and more. **Note**: You must sign up for a Share Space ID to access information provided on this site.
`https://identitymanageride.dev.java.net`	Open source Sun Identity Manager Integrated Development Environment (Identity Manager IDE) project. Includes instructions for installing and configuring Identity Manager IDE.

Process Overview

Generally, you can identify and resolve problems in your deployment if you follow the steps shown in the following illustration.

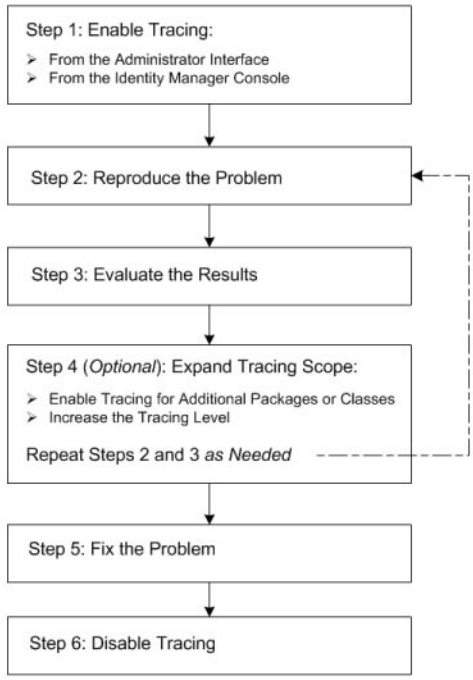


FIGURE 5–1 Tracing and Troubleshooting a Problem

See "Tracing Identity Manager Objects and Activities" on page 88 for information about how to enable tracing for different Identity Manager objects and activities.

See "Troubleshooting and Fixing Common Problems" on page 125 for information about how to troubleshoot problems that commonly occur for those objects and activities.

Tracing Identity Manager Objects and Activities

Trace output can be very helpful when you are trying to identify and resolve problems, or when you are developing custom resource adapters.

This section describes how to enable tracing for a variety of Identity Manager objects and activities. The information is organized as follows:

Note – Tracing affects system performance. To ensure optimal performance, specify the minimum tracing level or turn tracing off after debugging the system.

How To Configure Tracing

You can enable and configure tracing from several locations within Identity Manager. Instructions are provided in the following sections:

▼ To Enable Trace from the System Settings Page

The System Settings page is the primary Identity Manager Debug page.

From this page, you can

- View and edit objects in the repository
- Clear caches

- Set specialized traces
- Reload the `Waveset.properties` file

Use the following steps to enable tracing from the Identity Manager System Settings page:

1 **Open a browser and log in to the Identity Manager Administrative interface.**

2 **Type the following URL:**

`http://`*hostport*`/idm/debug`**

where:

- *host* is the local server on which you are running Identity Manager.
- *port* is the number of the TCP port on which the server is listening.

3 **When the System Settings page displays, click Show Trace (located near the bottom of the page) to manage a single trace configuration, where you can create, modify, or delete up to ten trace settings.**

Note – The remaining instructions assume you have only one trace configuration.

To manage multiple trace configurations, click the Show Trace List button instead. When the Trace Configuration page displays, click the configuration name to edit the current settings.

Identity Manager supplies one configuration, called *Global*, by default. However, if you have multiple servers in an Identity Manager instance, defining different configurations for particular servers might be useful. If the name of a trace configuration matches the name of the current host, the host configuration overrides the Global configuration.

Next Steps After you configure tracing, you can view and edit the default global trace configuration object or create a new configuration object as described in the following sections:

- "To Edit the Default Configuration Object" on page 89
- "To Create a New Trace Configuration Object" on page 91

▼ To Edit the Default Configuration Object

1 **From the Edit Trace Configuration page, click the Trace Enabled box to enable tracing.**

Note – Deselecting this box stops tracing, but keeps your configuration. You can turn tracing on and off without having to remember and retype the classes you were tracing.

---

2 **Specify which classes, packages, or methods to trace by typing their names into the table.**

For example:

- To trace all classes in the `waveset.repository` package, enter `com.waveset.repository`.
- To trace the `AbstractDataStore` class in the `waveset.repository` package, enter **`com.waveset.repository.AbstractDataStore`**.
- To trace the list method in the `AbstractDataStore` class in the `waveset` package, enter **`com.waveset.repository.AbstractDataStore#list`**.

Do not enable trace for the `com.waveset` class. The `com.waveset` class is verbose and has many classes, so tracing this class might cause your server to hang.

3 In the same table, choose a Method/Class tracing level from the Level menu.

Each level captures different types of information, as described in the following table:

Trace Level	Description
0	Minimum debugging output, traces exceptions and error information only
1	Trace Level 0 events, plus entries and exits for public methods
2	Trace Level 1 events, plus entries and exits for non-public methods
3	Trace Level 2 events, plus decision points and significant variables
4	Maximum debugging output

Note – Method/Class tracing produces a predictable, but possibly very large volume of trace output. Try to be as specific as possible when specifying methods and classes to trace.

4 (*Optional*) To enable subcall tracing, choose a level from the Subcall Trace menu. This menu uses the same trace numbering scale described in the previous table.

Note –

- The default Subcall Tracing level is *none*, which disables subcall tracing on a per-method or per-class basis.
- Subcall Tracing levels are independent of the Method/Class tracing levels you specified in the previous step.

When you enable Subcall Tracing for a particular method that supports subcall tracing, you are automatically setting the tracing level for methods that are *called by* this method. Subcall Tracing enables you to produce a short, but detailed burst of trace output that is triggered by the entry into a specified method and terminated by the method's exit.

For example, assume you created a trace configuration setting for the `com.waveset.adapter.NewRes#init` method, set Method/Class tracing to level one and set Subcall tracing to level three.

Also, assume that the `init` method calls two other methods:

- `NewRes#subcallA`
- `NewRes#subcallB`

  When the `init` method runs, the `com.waveset.adapter.NewRes#init` method produces trace output at level one until reaching `subcallA`. When `subcallA` begins executing, the trace level changes to three and continues at that level until `subcallA` exits. The `com.waveset.adapter.NewRes#init` method returns to the `init` method and restores the trace level to one. Later, when `init` calls `subcallB`, there is another burst of level three trace detail that lasts until `subcallB` exits. Finally, when `init` exits, level one tracing stops.

5 **Send the trace results to a specified file location or to `stdout`.**

If you choose output to a file, the Trace File field displays. Use this field to specify an alternate location and file name for trace output files. The default location and file name is *path_to_idm_install*`\export\pipeline\config\WSTrace.log`

6 **Specify the maximum number of trace files to store (default is 2).**

7 **Specify the maximum size for each file (default is 512K).**

8 **Specify whether to write the trace output file as generated (synchronously) or to queue the data and then write it to the trace file (asynchronously).**

9 **Save your changes.**

▼ To Create a New Trace Configuration Object

Use the following steps to create a new trace configuration object:

1 **Decide which package or method you want to trace.**

Generally, you specify a resource adapter name or use information that was revealed in an error message.

2 **Log in to the Identity Manager Administrator interface and open the System Settings page as described in To Enable Trace from the System Settings Page.**

3 **On the Systems Setting page, click Show Trace List.**

4 **When the Identity Manager Trace Configuration page displays, click New.**

5 **Enable tracing from the Edit Trace Configuration page.**

Choose one of the following options from the Trace Configuration menu:

- **Global**. Select to enable tracing for all servers.
- **Server_name**. Select a server name to enable tracing for a particular server.

6 **Select the Trace Enabled box to enable tracing for this object and configure the remaining parameters on this page as described in To Edit the Default Configuration Object.**

7 **Save your changes.**

To Configure Tracing From Individual Debug Pages

This section describes how to enable tracing from an Identity Manager Debug page.

1 **Open a browser and log in to the Identity Manager Administrative interface.**

2 **Type the following URL:**

`http://` *host*`:`*port* `/idm/debug/`*pageName.jsp*

where:

- *host* is the local server on which you are running Identity Manager.
- *port* is the number of the TCP port on which the server is listening.
- *pageName.jsp* is the particular Debug page you want to open.

  For example, to trace adapter classes and methods for a custom adapter, you can open the Edit Trace Configuration page by typing the following URL:

  `http://`*host* `:`*port*`/idm/debug/Show_Trace.jsp`

To Enable Tracing From the Identity Manager Console

This section explains how to enable tracing from the Identity Manager console.

1 **Set `$WSHOME`.**

For example, to set the variable to the default installation directory:

```
set WSHOME=C:\Program Files\tomcat\webapps\idm
```

2 **Type `lh console` to open the Identity Manager console from the `bin` directory.**

3 **From the console, type `trace` to see a detailed summary of available trace options, including `enable` and `disable`.**

The syntax for this command is:

```
trace [ -s server ] $subcommand
```

How to View Trace Files

By default, Identity Manager sends tracing information to a file named `WSTrace#.log` that is stored in the *path_to_idm_install*`\export\pipeline\config\` directory. If necessary, you can specify alternate file names and locations when you configure tracing for an object.

Each log file is numbered sequentially, up to the maximum number of files specified on the Edit Trace Configuration page. For example, if you specify a maximum of three files, the files are named `WSTrace1.log`, `WSTrace2.log`, and `WSTrace3.log`.

Use one of the following methods to view these log files:

- If you are sending trace information to a file, open the trace file from the specified location. For example:

  path_to_idm_install `\export\pipeline\config\WSTrace2.log`

- If you are sending trace information to `stdout`, open your application server's `stdout` file in a text editor to view the trace output logs.

Tracing the Identity Manager Server

Identity Manager is a Java-based product whose executables consist of Java classes grouped as packages. When the code is implemented, many classes can supply trace output.

Tracing the Identity Manager server can provide helpful information such as where a server is failing, having problems, or not running. You can use the Identity Manager Debug pages to enable package-level and method-level tracing in a running Identity Manager server.

Note – Configure Identity Manager to trace at this advanced level *only* if instructed by Sun Support.

Tracing Adapters

You can use trace information to verify that a resource adapter has started, to verify that all setting changes were saved, and to diagnose problems with the adapter.

This section contains information to help you configure adapter tracing, and is organized into the following topics:

General Notes About Tracing Adapters

When you enable tracing for an adapter, you must identify the methods that you want to trace by typing:

com.waveset.adapter.*sample***.***MyResourceAdapter*

You must also provide calls to create log entries for any new methods in your custom resource adapter so the adapter can write its resource settings to a log file.

In some cases, you can specify additional logging parameters for the resource instance, such as:

- Maximum Log Archives
- Maximum Active Log Age
- Log File Path
- Maximum Log File Size
- Log Level

Note –

- To further debug the synchronization process, you can configure synchronization logging for an ActiveSync adapter. Instructions are provided in the Identity Manager online help.
- Tracing custom Java code can be useful when you are writing your own resource adapter. See "Tracing Custom Code" on page 102 for more information.
- You can instrument code with trace points to help assess and resolve problems more effectively. See "Instrumenting Code with Trace Points" on page 94 for more information.
- For more information about tracing specific adapter methods, see the *Sun Identity Manager 8.1 Resources Reference*.

Instrumenting Code with Trace Points

The Identity Manager trace facility enables you to instrument code with trace points in the adapters component (resource adapters). This section contains some guidelines to help you to consistently use trace points to assess and resolve adapter problems.

Some guiding principles for instrumenting code with trace points are as follows:

- Make trace points as useful as possible for troubleshooting in both production and development environments by displaying relevant information.
- Create one entry point and one exit or exception trace point for every significant method.
- Use trace points as efficiently as possible during normal operations and when tracing is enabled by
  - Minimizing the number of trace level checks
  - Minimizing object creation when tracing is enabled

You can specify a trace level for a trace point to control how much information displays. The following table describes each of the trace levels that are defined in the `com.sun.idm.logging.TraceManager` interface.

TABLE 5–2 Defined Trace Levels

Trace Level	Trace Variable	Usage
1	`Trace.LEVEL1`	Entry and exit points of public methods of the resource adapter interface
2	`Trace.LEVEL2`	Entry and exit points of non-public methods or methods not included in the Identity Manager component external interface
3	`Trace.LEVEL3`	Decision points or significant variables
4	`Trace.LEVEL4`	Very detailed information, such as significant variables within a loop
n/a	`Trace.ALWAYS`	Do not perform a trace-level check **Note**: Use this option if you have already specified the trace level in a condition.

Note – Use the `com.sun.idm.logging.Trace` class to prevent circular dependencies on the `com.sun.idm.logging` package that implements the `com.sun.idm.logging.TraceManager` interface.

When adding trace points to code, be aware of the following information:

- Trace arguments to the method and return values from the method.

  If the arguments "print large," where the text representation is not brief, specify whether the Object is null or not. If the argument or return value is an array or a list, and the value could print large, showing the array size or list size is sufficient. You can use the `com.waveset.util.Util.length()` utility method to guard against null values.
- To format trace information or to create objects that present trace information in a meaningful way, put an `if` statement around the trace statements, using the `Trace.getLevel` method and the `isLogging(level)` method, so that Identity Manager does not marshal the trace point data unless the specified trace level is enabled.

  When using this construct, specify the `Trace.ALWAYS` trace level for trace points inside the condition. This trace level prevents an unnecessary check for the trace level when that check has already been performed in the condition.
- If you are tracing an object value that requires a method call to get a printable representation of the object, such as `obj.getName()`, guard against a null object to prevent a `NullPointerException` occurring if trace becomes enabled.

Note – In general, do not use trace points for simple `getter` and `setter` methods or for any other trivial methods.

- When tracing a `WSUser` object, use the `toString()` method instead of the `getName()` method to present the value of the object. The `toString())` method is more robust and prints more useful information than the `getName()` method. Instead of tracing `user.getName()`, trace `user`, which calls `toString()` implicitly.
- If a method or constructor is just a wrapper, where that method or constructor does nothing significant except call another method or constructor with the same name but a different signature, just put trace points into the destination method or constructor.
- If performance is critical and the method is called many times, use information trace points to indicate errors. Do not add entry or exit trace points to the method. Use this option carefully.
- Before adding a trace point, ask yourself if that trace point will provide useful information when determining problems in the field.

The following sections describe specific trace levels in greater detail and provide examples showing how to use trace points in code.

- "Using Entry and Exit Trace Points" on page 96
- "Using Information Trace Points" on page 98
- "Using Exception Trace Points" on page 99
- "Using Trace Point Templates" on page 100

Using Entry and Exit Trace Points

Consider these guidelines before adding entry and exit trace points to your code:

- Use `Trace.Level1` entry or exit trace points for all Identity Manager component external interface methods, which are the public methods declared in the resource adapter interface.
- Use `Trace.Level2` entry or exit trace points for significant non-public methods including constructors and static methods or methods not included in the Identity Manager component external interface.
- Specify arguments in the trace points.

  If the argument is not a primitive type or `java.lang.String`, and the argument requires any kind of object creation or method calls, conditionalize the entry trace point so that the formatting work is only done when tracing is enabled.
- When specifying an exit trace point, show the return value if the method is a primitive type or `java.lang.String`. If the return value requires some sort of formatting, conditionalize the object formatting as specified in these guidelines. Use the following:

```
int getLevel()
int getLevel (Method)
boolean isLogging(level,method)
boolean level1 (method)
boolean level2 (method)
boolean level31 (method)
boolean level4 (method)
```

- If you are tracing information that requires appending several `java.lang.Strings` together, use `java.lang.StringBuffer` instead of `java.lang.String`. Appends are faster using `java.lang.StringBuffer`. Also, conditionalize as in the preceding bullet item.
- When adding entry or exit trace points for significant constructors, remember that you cannot place a trace point before a `this()` or `super()` method call within a constructor. Put the entry trace point immediately after the `this()` or `super()` method call.
- Do not use an exit trace point to trace an exception condition. If the method throws an exception, use an exception trace point at the same trace level as the entry or exit trace point just before the exception is thrown. See "Using Exception Trace Points" on page 99for more information.

  If a method contains an entry trace point, that method also must contain a code path that executes either a `throwing` method, a `caught` method, or an exit.

Following are some simple entry and exit trace statements. For these examples, assume the following `CLASS` variables are declared for each statement.

```
private static final String CLASS =
"com.waveset.adapter.MyResourceAdapter";
protected static Trace _trace = Trace.getTrace();
```

EXAMPLE 5–1 Entry Trace Point Example

```
final String METHOD = methodName;
_trace.entry(_trace.LEVEL1, CLASS, METHOD);
_trace.entry(_trace.LEVEL1, CLASS, METHOD, user);
if (_trace.level1(CLASS, METHOD)) {
_trace.entry(_trace.ALWAYS, CLASS, METHOD,
user= + user);
}

// Show the size of an array argument
// ASSUME: password is an argument to the method
// Note the use of the Util.length() method. It is
// a convenience method that guards against null.
if (_trace.level1(CLASS, METHOD)) {
    StringBuffer sb = new StringBuffer(32);
    sb.append(password length=);
    ab.append(Util.length(password));
    _trace.entry(_trace.ALWAYS, CLASS, METHOD, sb.toString());
```

EXAMPLE 5–1 Entry Trace Point Example *(Continued)*

```
}
```

EXAMPLE 5–2 Exit Trace Point Example

```
_trace.exit(_trace.LEVEL1, CLASS, METHOD);

_trace.exit(_trace.LEVEL1, CLASS, METHOD, returnValue);
if (_trace.level1(CLASS, METHOD)) {
    _trace.exit(_trace.ALWAYS, CLASS, METHOD,
      returnValue != null ? returnValue.getName() : returnValue);
}

// Show the size of an array
String[] accounts = ...
if (_trace.level1(CLASS, METHOD)) {
    StringBuffer sb = new StringBuffer(32)
    sb.append(accounts length=);
    ab.append(accounts.length);
    _trace.exit(_trace.ALWAYS, CLASS, METHOD, sb.toString());
}
```

Using Information Trace Points

Consider these guidelines before adding information trace points to your code:

- Generally, use `Trace.Level3` or `Trace.Level4` for information, variable and data trace points. The trace level depends upon the significance, expense, and the verbosity of the information presented.
- Use `Trace.Level3` and `Trace.Level4` information and variable trace points to trace significant conditions, branches, and other information as they occur in the code path.
- A variable trace point is a convenience method that is the same as an information trace point, except that a variable trace point prints out `variable=<variable>` instead of just printing the argument value. In addition, this method does not format the information unless the trace level specified in the argument is matched.
- Do not use an information trace point if the method contains an exception condition but does not rethrow the exception. Use the `caught` method. For an example, see "Using Exception Trace Points" on page 99.
- Use `Trace.Level3` and `Trace.Level4` data trace points to show information in byte arrays. You can use a lower trace level if the information is appropriate for that trace level and the information presented is brief.
- You can use `Trace.Level1` and `Trace.Level2` information, variable, and data trace points to trace significant, but brief, information. However, you should rarely use these trace points at these levels.

Following is an example of a simple information trace statement. For this example, assume the following `CLASS` variables are declared.

```
private static final String CLASS =
"com.waveset.adapter.MyResourceAdapter";
protected static Trace _trace = Trace.getTrace();
```

EXAMPLE 5–3 Information Trace Point Example

```
_trace.info(_trace.LEVEL3, CLASS, METHOD, Some Message);
WavesetResult result = new WavesetResult();
try {
   someMethod();
}  catch(Exception e) {
   String msg = Some Error Message;
   WavesetException we = new Waveset(msg, e);
   _trace.caught(_trace.LEVEL3, CLASS, METHOD, e);
   result.addException(we);
}
```

Using Exception Trace Points

Consider these guidelines before adding exception trace points to your code:

- Use exception trace points in all methods where entry or exit trace points are specified, and in methods that catch and rethrow exceptions. Identity Manager sometimes throws exceptions in which a new exception wraps the original exception. Use the same trace level being used by the entry and exit trace points.
- Use an exception trace point if an exception is created and thrown from the current method.
- Use an exception trace point `caught` method if an exception is caught and handled. For example, when the exception is not thrown from the current method. This situation typically occurs when Identity Manager catches and handles the exception by placing it in a container, such as a `WavesetResult`, to be examined later.
- If the exception thrown from the current method is not a *checked* exception, which is an exception that extends `java.lang.RuntimeException` or `java.lang.Error`, the method exit point will not be traced unless other measures are taken. One way to set an exception trace point in this situation is to use a `try/catch` block around the critical section of the method and rethrow the exceptions that occur. In general, this means that you should use `Trace.Level3` and higher when it is critical to know whether a method has succeeded or failed.

Following is an example of a simple exception trace statement. For this example, assume the following `CLASS` variables are declared:

```
private static final String CLASS =
"com.waveset.adapter.MyResourceAdapter";
```

```
protected static Trace _trace = Trace.getTrace();
```

EXAMPLE 5–4 Exception Trace Point Example

```
try {
   someMethod();
}  catch(Exception e) {
   _trace.throwing(_trace.ALWAYS, CLASS, METHOD, e);
   throw e;
}


try {
   someMethod();
}  catch(Exception e) {
   _trace.throwing(_trace.ALWAYS, CLASS, METHOD, e);
   WavesetException we = new WavesetException(Some Message, e);
   throw we;
}


if (error) {
  WavesetException e = new WavesetException(Some Error Message.;
  _trace.throwing(_trace.LEVEL3, CLASS, METHOD, e);
  throw e;
}

try {
   someMethod();
}  catch(Exception e) {
   _trace.caught(_trace.LEVEL1, CLASS, METHOD, e);
}
// execution continues.
someOtherMethod();
```

Using Trace Point Templates

Using trace point templates can help you provide consistent and useful trace points in Identity Manager resource adapters. Identity Manager provides Eclipse templates in the following location:

```
src/wps/doc/eclipse-trace-templates.xml
```

To import these templates into Eclipse, select Window → Preferences → Java → Editor → Templates.

Note – You can create similar templates if you are using Emacs or IDEA.

Other Tracing Guidelines

This section describes some additional tracing guidelines, including the following:

- "Tracing Inner Classes" on page 101
- "Tracing Static Initializers" on page 101

Tracing Inner Classes

Remember to trace significant methods in inner classes. Be sure to declare a final static `CLASS` variable if there are any trace methods in the inner class.

EXAMPLE 5–5 Example of Using a `CLASS` Variable in Inner Classes

```
private final static String CLASS =
com.waveset.adapter.SomeResourceAdapter$AdapterInnerClass;
```

Tracing Static Initializers

In general, do not use the trace facility in static initializers. Identity Manager executes these initializers when the class is loaded, which can occur when the repository is initialized. Use the `Debug` class to trace anything significant in static initializers.

Tracing Auditor

You can trace the following methods to troubleshoot issues with Identity Auditor:

- `com.sun.idm.auditor.policy` to trace issues with Audit Scans.
- `com.sun.idm.auditor.accessreview` to trace issues with Access Reviews.
- `com.sun.idm.auditor.report` to trace issues with Audit Reports.
- `com.sun.idm.auditor.view` to trace issues with Auditor Views.

▼ To Enable Tracing:

1. **Open a browser and log in to the Administrator interface.**
2. **Select Configure → Servers.**
3. **When the Configure Servers page displays, click the server name in the Server column to edit the settings.**
4. **On the Edit Server Settings page, click the Scheduler tab.**

5 **Select the Tracing Enabled box to activate Scheduler debug tracing and write the results to `stdout`.**

6 **Save your changes.**

Tracing Custom Code

You can use the Identity Manager tracing facility in custom-written code to provide seamless integration when troubleshooting your deployment.

Use the following classes to provide this tracing facility:

- Use `com.sun.idm.logging.trace.Trace` objects to interface with the tracing subsystem.
- Use `com.sun.idm.logging.trace.TraceManager` as a factory for these objects.

You can also use these `com.sun.idm.logging.trace` facilities to record trace output. See the Javadoc™ for more information about these facilities and classes.

Note – Be aware that the `com.sun.idm.logging.trace` facilities were *not* available prior to the Identity Manager Version 6.0 SP1 release. For earlier versions of Identity Manager, you must use the `com.waveset.util.Trace` class instead.

Tracing Exceptions

Exception logs are stack traces that you can view from the Identity Manager Debug pages or from the `config/Waveset.properties` file. Trace data does not include all exceptions by default, but exception logging can be an important and informative troubleshooting tool.

Use one of the following methods to enable exception logging:

- Open the Waveset Properties page (`debug/Show_WSProp.jsp`) in the Identity Manager Administrator interface. Locate the `exception.trace` key and change the Value to **`true`**.
- Open the `config/Waveset.properties` file in a text editor, and change the `exception.trace` key value to **`true`**. The following figure shows an example exception trace.

Identity Manager sends exception logs to `stdout` on the web application instance, which is often the application server console.

Note – When you are finished, disable exception logging to stop unnecessary output to the application server logs. To disable exception logging, set the `exception.trace` key value back to **`false`**.

Tracing Forms

You can enable tracing to troubleshoot edited forms and to check for `expression` statement errors within your form fields.

Use either of the following methods to enable tracing:

- Open the Waveset Properties page (`debug/Show_WSProp.jsp`) in the Identity Manager Administrator interface. Locate the `form.trace` key and change the Value to **`true`**.
- Open the `config/Waveset.properties` file in a text editor, and change the `form.trace` key value to **`true`**.

Identity Manager reports any problems with form expression syntax to standard output.

Note – The `form.trace` key is disabled by default because it produces trace information for every field on every page, including the Accounts List page, which affects system performance. Consider using a more targeted form and field tracing method.

When you are finished troubleshooting your forms, remember to disable tracing by changing the `form.trace` key value back to **`false`**.

Global XPRESS tracing might also be helpful while you are developing and updating forms and form processes. Although Global XPRESS tracing produces a large amount of output that affects system performance, this tracing method shows XPRESS output and might expose where problems are happening in your form.

For more information, see

- "Tracing Global XPRESS" on page 104
- "Testing Your Customized Form" in *Sun Identity Manager Deployment Reference*

Tracing Global XPRESS

While not generally recommended, you can use global XPRESS tracing to trace any and all XPRESS code, wherever the code is located. For example, you can trace XPRESS code in forms, views, and workflows. The resulting trace shows XPRESS output that can expose potential problems.

Note – XPRESS tracing is disabled by default because it produces a large amount of output, which affects system performance.

See "Testing Your Customized Form" in *Sun Identity Manager Deployment Reference* for more information about tracing XPRESS functions.

▼ To Enable Global XPRESS Tracing:

1. **Open a command window.**
2. **Change directories to `config/Waveset.properties` in the default Identity Manager installation directory.**
3. **Open the `config/Waveset.properties` file and edit the `xpress.trace` line to read:**

   ```
   xpress.trace=true
   ```

4. **Save the `Waveset.properties` file.**

5 **Restart your application server or reload the `Waveset.properties` file from the Identity Manager debug pages.**

6 **Replicate the XPRESS trace output to a file by adding this line to the `Waveset.properties` file:**

```
xpress.traceFile=FileName.txt

xpress.traceFileOnly=true
```

When you set `xpress.traceFileOnly=true` in `Waveset.properties`, all XPRESS statement evaluations will generate trace messages to a file specified by `xpress.traceFile`. Otherwise, when `xpress.traceFile` has a value, trace messages are redirected to both the console and a file.

Tracing PasswordSync

This section describes how to enable tracing for PasswordSync and how to configure tracing in Direct access or JMS modes.

To Enable Trace for PasswordSync

You can use the following methods to configure tracing for Identity Manager's PasswordSync feature:

- "Using the PasswordSync Configuration Tool" on page 105
- "Editing the Registry Keys" on page 106

Using the PasswordSync Configuration Tool

This section describes how to configure tracing from the PasswordSync Configuration Trace tab.

Note – For more information about installing and configuring PasswordSync, see Chapter 11, "PasswordSync," in *Sun Identity Manager 8.1 Business Administrator's Guide*.

The first time you run the configuration tool, the wizard does not allow you to configure tracing. Subsequently, when you run the configuration tool, the wizard displays a Trace tab where you can configure tracing.

The following figure shows the PasswordSync Configuration tool Trace tab.

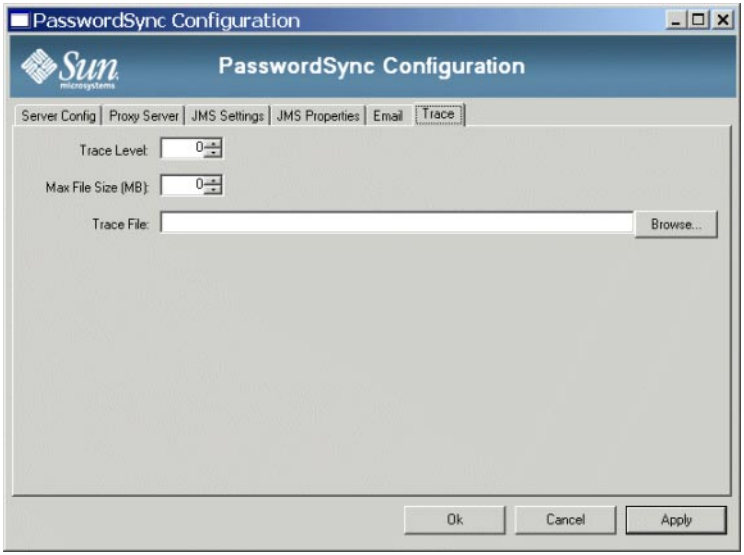


FIGURE 5–2 Trace Tab in the PasswordSync Configuration Tool

From this tab, you can specify the following:

- Use the Trace Level field to specify the level of detail you want PasswordSync to provide when writing to the trace log. A value of `0` turns tracing off, while a value of `4` shows full detail.
- Use the Max File Size field to specify a maximum size for the log file.

  When the trace file exceeds the size specified in the Max File Size (MB) field, PasswordSync starts a new trace file and appends `.bk` to the old trace file name. For example, if your trace level is set to 100 Mbytes, and your trace file writes to `C:\logs\pwicsvc.log`, when the trace file exceeds 100 Mbytes PasswordSync renames the file to `C:\logs\pwicsvc.log.bk`. PasswordSync then creates a new `C:\logs\pwicsvc.log file` where new trace file messages are written.
- Use the Trace File field to specify a location for the PasswordSync trace file.

Editing the Registry Keys

To enable additional PasswordSync configuration settings, edit the following PasswordSync registry keys using the PasswordSync configuration tool.

Note – Using PasswordSync configuration tool is the safest method for editing PasswordSync registry keys. Editing these keys directly in the Windows Registry is discouraged.

TABLE 5–3 Registry Keys

Key Name	Type	Description

`dumpFilebase`	`REG_SZ`	Set this registry key to enable Windows to generate a dump file if the PasswordSync DLL displays an exception.
		
		You must set this registry key to the fully qualified directory path where you want to write the memory dump. For example: `c:\temp`
		
		Set the registry value to write the memory dump each time Identity Manager catches an exception during password processing.
		
		Note: On Windows 2000 server (any service pack), you also must install in the configured directory `DbgHelp.dll`, which is available from Microsoft. The minimum release version for the `DbgHelp.dll` file must be Version 5.1. Download the `DbgHelp.dll` file here:
		
		`http://www.microsoft.com/whdc/DevTools/Debugging/default.mspx`
		
		If `DbgHelp.dll` is not installed, no dump files will be generated on Windows 2000.
		
		The format for dump file names is
		
		`lhpwic-YYYYMMDD-HHmm-xxxxx.dmp`
		
		In this name, `YYYYMMDD` will be the date of the dump, `HHmm` is the time of the dump (24-hour clock), and xxxxx is the thread number of the application.
		
		You must manually remove dump files! Dump files range in size from 20 MB to more than 100 MB, depending on the size of the Windows Local Security Authority Subsystem (LSASS) process. Over time, systems with limited disk space could fill up if these dump files are not removed.

`installdir`	`REG_SZ`	Directory where the PasswordSync application is installed.

The PasswordSync registry keys are located in the following location:

`HKEY_LOCAL_MACHINE\SOFTWARE\Waveset\Lighthouse\PasswordSync`

Other keys are present in this location.

To Collect Logs for the Different Modes

PasswordSync trace logs are the same, whether you are using a direct access mode or JMS mode configuration. However, these trace logs might only provide partial information. You must configure different classes for each configuration to collect logs on the server side, as described in the following sections.

Tracing in Direct Mode

When using PasswordSync with a direct access mode configuration, the trace logs show failures, but not all logged failures are real failures. For example, in some circumstances the view check-in takes a long time, which shows as a failure in the log. You must trace on the server side to see this information.

In Direct mode, PasswordSync talks to the servlet that generates the view to be checked into the repository. You can trace the `com.waveset.rpc.GenericMessageHandler` class at level 4 to view all phases of password synchronization, from receiving the password change to the response generated and returned to the servlet. Level 4 is the only level that supplies enough detail for troubleshooting.

Tracing in JMS Mode

When using PasswordSync with a JMS mode configuration, the logs only show successful or failed deliveries to the JMS server. From this point on, you must rely on server side logs. JMS tracing is a little more complex.

You can trace the `com.waveset.rpc.PasswordSyncHandler` class at level 4 to convert the messages generated by the PasswordSync `dll` into a JMS message and add those messages to the JMS queue. Limited tracing is available in this class, and only level 4 provides enough information to help with troubleshooting.

If PasswordSync successfully delivers the JMS message to the JMS queue, the tracing will not help you find the cause of a problem. The next, and final step is to trace the JMS adapter. See the *Sun Identity Manager 8.1 Resources Reference* for instructions.

Tracing Rule-Driven Members Caches

You can trace rule-driven members caches and use the results to tune cache properties in `Waveset.properties`.

If you trace `com.waveset.server.RuleDrivenMembersCache` at Level 1, the resulting information includes the number of adds, removes, hits to the cache, and so forth. Use this information to evaluate cache sizes and decide whether tuning the cache properties in `Waveset.properties` is necessary.

You can use the following properties in `Waveset.properties` to control the rule-driven members cache:

- Use `ruledrivenmemberslistcache.size` = *value* to specify the maximum number of object lists to be cached per subject. (Default is 20.)
- Use `ruledrivenmemberslistcache.rowlimit` = *value* to specify the maximum total number of objects to be cached per subject. (Default is 100000.)

By default, Identity Manager evaluates the User Members Rule associated with a specified organization and creates a user members list cache containing a dynamic list of users. However, you can also create a user members list cache containing a specified user's set of dynamic member organizations for a given subject. The key to this cache is the object type concatenated with the object ID. For example, the `User` object type concatenated with the `User#ID#Configurator` object ID. The value for each key is a list of object groups for which this object is dynamically a member.

To determine whether the object being evaluated is a dynamic member, the cache evaluates the same User Members Rule per organization as that used by the list cache. If the object is a dynamic member, Identity Manager adds that object to the list and then caches the list. Identity Manager caches both empty and non-empty lists to ensure the highest cache hit rate.

For this cache, you can use these properties in `Waveset.properties` to control memory requirements affecting performance:

- Use `ruledrivenmembersobjectcache.size` = *value* to specify the number of member object group lists to be cached per subject. The default value is 100.
- Use `ruledrivenmembersobjectcache.rowlimit` = *value* to specify the maximum total number of member object groups to cache per subject. The default value is 100000.

Tracing Sun Identity Manager Service Provider Delegated Administration

Enabling Identity Manager tracing at Method/Class Level 2 for the following classes allows you to trace authorization flows when listing or accessing Sun Identity Manager Service Provider (Service Provider) users and when AdminRoles are dynamically assigned when Service Provider users login.

- Trace `com.sun.idm.idmx.view.IDMXBrowseViewer` when searching for Service Provider users.
- Trace `com.sun.idm.idmx.view.IDMXUserViewer` when creating, editing, or deleting Service Provider users.
- Trace `com.sun.idm.idmx.api.IDMXServerUtils` when using both of the preceding classes.
- Trace `com.waveset.security.authn.WSSPELoginModule` when you are logging in as a Service Provider user.

Note – You configure Service Provider tracing from the Identity Manager debug pages. If necessary, review "Tracing the Identity Manager Server" on page 93 for instructions.

Tracing Reconciliation

If you are having problems with a reconciliation task, you can use the standard tracing facility on `com.waveset.task.Reconciler` to trace the Reconciler.

Use either of the following methods to enable tracing:

- Open the Waveset Properties page (`debug/Show_WSProp.jsp`) in the Identity Manager Administrator interface. Locate the `exception.trace` key and change the Value to **`true`**.
- Open the `config/Waveset.properties` file in a text editor, and change the `exception.trace` key value to **`true`**.

You can also enable tracing from the System Settings page and trace the following reconciliation methods, at the Method/Class trace Levels noted, to view useful debugging information.

TABLE 5–4 Reconciliation Methods/Classes to Trace

Trace this `com.waveset.recon.`*Method/Class*	**At this Trace Level**	**To See**
`ReconTask$WorkerThread#reconcileAccount`	2	The individual account being reconciled
`ReconTask$WorkerThread#performResponse`	2	The individual account and user during response
`ReconUtil#deleteAccountIndex`	2	User information to be deleted from the account index
`UserContext#aquireRepoLock`	2	The user who is being locked for update
`UserContext#releaseRepoLock`	2	The user who is being unlocked in the repository
`ReconUtil#deleteAccountIndex`	2	User information to be deleted from the account index
`UserContext#aquireRepoLock`	2	The user who is being locked for update
`ReconTask$WorkerThread#failUserExamination`	2	All user examination requests that failed with the error
`ReconTask$WorkerThread#failUserResponses`	2	All user response requests that failed with the error
`UserContext#releaseRepoLock`	2	The user who is being unlocked in the repository
`ReconTask$ResourceThread#examineResource`	3	How many accounts read from the resource
`ReconTask$ResourceThread#queueAccountReconciles`	3	Information about each account read from the resource such as `accountId`, `accountGUID`, `accountDisabled`
`ReconTask$ResourceThread#examineLighthouse`	3	How many Identity Manager users claim to own an account on the reconciled resource queued
`ReconTask$WorkerThread#findClaimants`	3	All Identity Manager users who claim to have an account on the resource

TABLE 5–4 Reconciliation Methods/Classes to Trace *(Continued)*

Trace this `com.waveset.recon.`*Method/Class*	**At this Trace Level**	**To See**
`ReconTask$WorkerThread#confirmPossibleOwners`	3	A list of all confirmed owners of resource accounts
`ReconTask$WorkerThread#applyResponse`	3	The response list that is being applied
`AccountContext#processAttributeWorkflow`	3	The attribute changes and the formatted changes during the launch of the attribute change workflow
`ReconUtil#getRuleState`	3	Full User view with the user's attribute during rule processing
`ReconUtil#evaluateCorrelationRule`	3	The value of the correlation rule state and result for examination
`ReconUtil#confirmPotentialOwners`	3	A list of users who have been confirmed using the confirmation rule
`ReconUtil#getIfExistsAccountIndexEntry`	3	To output info related to examination of the account index for a specified entry
`ReconUtil#launchWorkflow`	3	The task instance and task definition information when launched
`ReconUtil#indexFoundAccount`	3	The account and situation recorded during a create or update of the index for an account known to exist
`ReconUtil#indexMissingAccount`	3	The account and situation recorded during a create or update of the index for an account NOT known to exist
`ReconUtil#listAccountsIndexSaysExist`	3	Account information that the index says exists
`ReconTask$ResourceThreadwaitForLighthouseWorkItems#`	4	How many users queued and processed during the Identity Manager user examination process
`ReconTask$ResourceThread#waitForReconcileWorkItems`	4	How many reconciles and responses queued and processed for the specified resource
`ReconTask$WorkerThread#correlateUsers`	4	A list of correlated/confirmed users
`ReconTask$WorkerThread#respondOrRequeue`	4	The error message if there is a problem performing the previous response method for an account
`Response#perform`	4	The user, response, and the user's resource info XML before and after the modifications took place
`Response#perform`	4	A summary of the response action on the user
`Response#createNewUserFromAccount`	4	Full details of the response action on the user
`ReconUtil#evaluateConfirmationRule`	4	The value of the confirmation rule state and result for examination
`ReconUtil#getCorrelatedUsers`	4	A list of correlated users matching the specified rule result

Note – Remember, the higher the tracing level, the larger the trace file. Also tracing all of these methods at the same time will create a *very* large trace file.

When you are finished troubleshooting, remember to disable tracing by setting the `exception.trace` key value back to **`false`**.

Tracing the `setRepo` Command

If you see errors while you are using the `setRepo` command to configure the Identity Manager repository, use the following flags to isolate and debug the problem:

```
-Dtrace.enabled=true
-Dtrace.level.com.waveset.repository.AbstractDataStore=2
-Dtrace.level.com.waveset.repository.DefaultTypeHandler=4
// Use one of the following based on your repository type
-Dtrace.level.com.waveset.repository.OracleDataStore=4
-Dtrace.level.com.waveset.repository.SqlServerDataStore=4
-Dtrace.level.com.waveset.repository.MysqlDataStore=4
-Dtrace.level.com.waveset.repository.DB2DataStore=4
```

Identity Manager sends output from the `setRepo` command to the default `$WSHOME/config/WSTrace.log` file.

Tracing SPML

This section describes methods for enabling trace for SPML Version 1.0 and SPML Version 2.0.

To Enable Tracing for SPML 1.0

SPML 1.0 provides the following options for turning on trace output so you can log Identity Manager's SPML traffic and diagnose problems.

Method 1: Enable the `setTrace` Method

You can use the `setTrace` method, provided by the `SpmlClient` and `LighthouseClient` classes, to enable tracing for SPML 1.0.

When you enable this `setTrace` method, the XML for the request sent by the client and the XML for the response received from the server are printed to the *client* console as they are sent and received.

The `setTrace` method takes a Boolean argument. For example:

```
SpmlClient client = new SpmlClient();
 client.setURL("http://localhost:8080/idm/spml");
 client.setTrace(true);
```

Method 2: Initializing the `org.openspml.server.SOAPRouter` Servlet

You can enable tracing when initializing the `org.openspml.server.SOAPRouter` servlet, which is a third-party, open source class from the OpenSPML organization. This servlet controls the output of RPC traffic information for the servlet handling SPML requests.

To enable this tracing method, add the following to the `WEB-INF/web.xml` file:

```
<servlet>
    <servlet-name>rpcrouter2</servlet-name>
    <display-name>OpenSPML SOAP Router</display-name>
    <description>no description</description>
    <servlet-class>
        org.openspml.server.SOAPRouter
    </servlet-class>
    <init-param>
        <param-name>trace</param-name>
        <param-value>true</param-value>
    </init-param>
    ...
  </servlet>
```

The following is sample output for an SPML 1.0 trace:

```
SpmlClient: sending to http://example.com:8080/idm/servlet/rpcrouter2
<spml:addRequest xmlns:spml='urn:oasis:names:tc:SPML:1:0'
xmlns:dsml='urn:oasis:names:tc:DSML:2:0:core'>
  <spml:operationalAttributes>
    <dsml:attr name='session'>


<dsml:value>session token</dsml:value>

    </dsml:attr>
  </spml:operationalAttributes>
  <spml:identifier type='urn:oasis:names:tc:SPML:1:0#GUID'>
    <spml:id>suetonius</spml:id>
  </spml:identifier>
  <spml:attributes>
    <dsml:attr name='objectclass'>
      <dsml:value>person</dsml:value>
    </dsml:attr
    <dsml:attr name='password'
      <dsml:value>password</dsml:value>
```

```
    </dsml:attr>
    <dsml:attr name='gn'>
      <dsml:value>Suetonius</dsml:value>
    </dsml:attr>
    <dsml:attr name='sn'>
      <dsml:value>Tranquillus</dsml:value>
    </dsml:attr>
    <dsml:attr name='email'>
      <dsml:value>twelve@example.com</dsml:value>
    </dsml:attr>
  </spml:attributes>
</spml:addRequest>


SpmlClient: received
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'>
<SOAP-ENV:Body>
<spml:addResponse xmlns:spml='urn:oasis:names:tc:SPML:1:0'
xmlns:dsml='urn:oasis:names:tc:DSML:2:0:core' result='urn:oasis:names:tc:
SPML:1:0#success'>
  <spml:operationalAttributes>
    <dsml:attr name='session'>

<dsml:value>session token</dsml:value>
    </dsml:attr>
  </spml:operationalAttributes>
  <spml:identifier type='urn:oasis:names:tc:SPML:1:0#GUID'>
    <spml:id>suetonius</spml:id>
  </spml:identifier>
</spml:addResponse>
/SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note – For more information about the SOAP `rpcrouter` servlet, refer to your OpenSPML Toolkit documentation.

Method 3: Pass the `trace` Operational Attribute

You can enable tracing for an individual SPML RPC request by passing a `trace` operational attribute to the RPC request on the server side.

Tracing occurs during servlet initialization, and it controls how information is output for the RPC traffic of a servlet handling SPML Version 1.0 requests. For example, the trace prints the raw XML that is sent back and forth on whatever the `System.out` is for that servlet (which is a function of the Application container). For example:

```
AddRequest ar = new AddRequest();
 ar.setOperationalAttribute("trace", "true");
```

When you use the `trace` attribute, how the attribute affects server operation is vendor-specific. Currently, Identity Manager prints the raw request and response data to the *server* console, which is useful if the client application is not associated with a console window.

For more information consult your OpenSPML Toolkit product documentation.

To Enable Tracing for SPML 2.0

SPML 2.0 provides the following options for turning on trace output so you can log Identity Manager's SPML traffic and diagnose problems.

Method 1: Using the `org.openspml.v2.transport.RPCRouterServlet` Servlet

As with SPML 1.0, you can enable tracing for SPML 2.0 when initializing the `org.openspml.v2.transport.RPCRouterServlet` class, which controls the output of RPC traffic information for the servlet handling SPML 2.0 requests.

To enable this tracing method, add the following to the `WEB-INF/web.xml` file:

```
<servlet>
    <servlet-name>openspmlRouter</servlet-name>
    <display-name>OpenSPML SOAP Router</display-name>
    <description>A router of RPC traffic - nominally SPML 2.0 over SOAP</description>
    <servlet-class>
        org.openspml.v2.transport.RPCRouterServlet
    </servlet-class>
    <init-param>
      <param-name>trace</param-name>
      <param-value>true</param-value>
    </init-param>
    ...
  </servlet>
```

The following example illustrates output from an `rpcrouter` servlet trace:

```
RPCRouterServlet:
<?xml version='1.0' encoding='UTF-8'?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header/>
<SOAP-ENV:Body><lookupRequest
xmlns='urn:oasis:names:tc:SPML:2:0' requestID='random name' executionMode='synchronous'
returnData='everything'>
<openspml:operationalNameValuePair xmlns:openspml='urn:org:openspml:v2:util:xml' name='
session'value=session token'/>
<psoID ID='random name' targetID='spml2-DSML-Target'/>
</lookupRequest>
```

```
</SOAP-ENV:Body></SOAP-ENV:Envelope>

RPCRouterServlet:  response:
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope

  xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'>
<SOAP-ENV:Body>
<lookupResponse xmlns='urn:oasis:names:tc:SPML:2:0' status='failure' requestID='random
name'error='noSuchIdentifier'>
<openspml:operationalNameValuePair xmlns:openspml='urn:org:openspml:v2:util:xml'
name='session'

value=session token/>
<errorMessage>Item User:random name was not found in the repository, it may have been
deleted in another session.</errorMessage>
</lookupResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note – For more information see "Method 2: Initializing the `org.openspml.server.SOAPRouter` Servlet" on page 113, in the "To Enable Tracing for SPML 1.0" on page 112 section.

Method 2: Using the SPML Access Log

SPML 2.0 has a simple text access log that can be a useful troubleshooting tool. This log is always available and it enables you to view information, such as what kind of requests have been received, how long it took to process those requests, and whether the requests were successful without having to enable tracing.

Instructions for configuring this SPML text access log are provided in "Configuring SPML Tracing" in *Sun Identity Manager 8.1 Web Services*

Tracing the Task Scheduler

If a Scheduler task is having problems, you can use the standard tracing facility on `com.waveset.task.Scheduler` to trace the task scheduler. The output shows detailed information about task scheduling.

▼ To Enable Tracing:

1. **Open a browser and log in to the Administrator interface.**
2. **Select Configure → Servers.**

3. **When the Configure Servers page displays, click the server name in the Server column to edit the settings.**
4. **On the Edit Server Settings page, click the Scheduler tab.**
5. **Select the Tracing Enabled box to activate Scheduler debug tracing and write the results to `stdout`.**
6. **Save your changes.**

Note –

- In a clustered environment, tracing occurs on each instance.
- See "Tracing the Identity Manager Server" on page 93 for more information about defining and editing trace configuration objects, and about viewing trace files.

Tracing Workflows

Enabling workflow tracing can help you resolve problems with workflow activities and understand the workflow process.

Note –

- In a clustered environment, tracing occurs on each instance.
- To debug workflows in a multiple server deployment environment, consider shutting down all but one server. If all servers are running, you cannot determine which server in the environment is executing the workflow, which causes troubleshooting problems.

To Enable Tracing for Workflows

Use the following steps to enable workflow tracing:

▼ To Enable Workflow Tracing:

1. **Open a browser and log in to the Identity Manager Administrator interface.**
2. **From the System Setting page, choose Configuration from the List Objects Type menu.**
3. **Click the List Objects button.**
4. **When the List Objects of type: Configuration page displays, scroll down the list of objects to locate the System Configuration object and click the edit link.**

5 **When the Checkout Object: Configuration, `#ID#CONFIGURATION:SYSTEMCONFIGURATION` page displays, you can edit any of the following `workflow` options in the `SystemConfiguration` object:**

Note – Typically, you enable only one option, but it is possible to enable more than one option at a time.

These attributes are not dependent on each other. You can turn on one type of trace while the other types are turned off.

- Specify `workflow.consoleTrace=`**`true`** to redirect workflow trace messages to the application server console, which can be useful when workflows are terminating due to a fatal exception because this attribute prints more trace output than `workflow.fileTrace`. (Default value is `false`.)
- Specify `workflow.fileTrace=`*PathtoFile*to redirect workflow trace messages to a file that is easy to read. This attribute does not have a value by default.

  ```
  <Attribute name='fileTrace'/>
  ```

  Add a value tag to the `workflow.fileTrace` attribute and, using Unix-style forward slashes, enter the path to a log file. Identity Manager stores relative pathnames relative to the application server's installation directory. For example:

  - **On Windows**. `<Attribute name='fileTrace' value='C:\mydir\workflow.txt'/>`
  - **On Solaris/UNIX**. `<Attribute name='fileTrace' value='/mydir/workflow.txt'/>`
- Specify `workflow.traceLevel=`*tracingLevel* to specify the level of workflow tracing you want to see.

  Specify `workflow.Trace=`**`true`** to trace workflow processing. You must restart the application server to start tracing with this option. Identity Manager stores the trace results in the task's `WavesetResult` object. Use this tracing option when file system access is unavailable. (Default value is `false`.)
- Trace messages in the task's `WavesetResult`. (Default value is `1`.)

Specifying **`workflow.Trace=true`** appends trace messages into one long, unformatted string that is difficult to read. Use this option only when you do not have access to the file system.

Note – With the first two options, you might lose some of the workflow trace if a fatal exception occurs.

---

6 **Save the `SystemConfiguration` object.**

7 **Restart your application server, or reload the `SystemConfiguration` object from the Identity Manager debug area.**

To Enable Tracing for a Designated Workflow

Use one of the following methods to enable tracing for a designated workflow:

- **Editing the WFProcess definition**. To enable trace unconditionally for a particular process, edit the XML of the `TaskDefinition` object by adding the `trace='console'` attribute to the `<WFProcess>` element.
- **Editing the workflow variable**. Gives you more control over the timing of tracing by using a workflow variable. The workflow engine will look for a top-level workflow variable named `trace`.

The following example shows how to trace a workflow variable.

EXAMPLE 5–6 Tracing a Workflow Variable

```
<Variable name='trace'>
   <cond><eq><ref>accountId</ref><s>jfaux</s></eq>
     <s>workflowTrace.txt</s>
   </cond>
 </Variable>
```

The `trace` variable turns tracing on only if the workflow is operating on a user named `jfaux`. You could also specify `trace` in a form field to control tracing interactively.

In this example, the trace output is written to the `workflowTrace.txt` file.

Locating Version Information

You can use one of the following methods to get information about the Identity Manager version you are currently using:

- Hover your cursor above the Help button in the upper right corner of the Identity Manager application window. A pop-up is displayed that contains the version number.
- Open the `Waveset.properties` file, and check the information provided at the top of the file.

Use the Identity Manager System Properties page (`/debug/SysInfo.jsp`) to get information about which JVM versions, and other system software versions, you are currently using.

Tracing the Identity Manager Gateway Objects and Activities

This section describes how to trace objects and activities in Sun Identity Manager Gateway, the information is organized as follows:

Note –

- When viewing or editing a Gateway trace file, use Notepad to avoid file restrictions.
- When you start the Gateway, the program appends new trace entries to the trace file instead of deleting entries. To locate the point at which the Gateway trace entries begin, look for a `Gateway version` string.
- The Gateway version is output in the trace automatically when you start the Gateway. You can also type `gateway -v` from the command line to get the version.

How to Configure Tracing from the Gateway Debug Page

You can enable tracing from the Gateway Debug page (`Gateway.jsp`) or from the command line to debug problems with Windows accounts on Identity Manager.

Instructions are provided in the following sections:

From the Gateway Debug Page

Enable tracing from the Gateway Debug page (`Gateway.jsp`) if you cannot access the Gateway. You can specify and retrieve Gateway trace files from this debug page.

▼ To Enable Tracing:

1. **Log in to the Identity Manager Administrator interface.**
2. **Type the following URL in to your browser to open the Gateway Debug page:**

   `http://`*host*`:`*port*`/idm/debug/Gateway.jsp`
3. **Choose a resource to trace from the Gateway Resource list.**

4 **If necessary, modify existing settings.**

Click the following buttons to modify the settings:

- **Get Version**. Returns the Gateway version and the operating system of the machine on which you are running the Gateway.
- **Get Trace File**. Returns the contents of the trace file.
- **Get Trace Parameters**. Returns the path of the trace file, the trace level, and the maximum size of the trace file.
- **Set Trace Parameters**. See "To Create a New Trace Configuration Object" on page 91 for information about these options.
- **Get Loaded Modules**. Returns the load addresses of modules (DLLs) being used by the Gateway.

The Get Loaded Modules list consists of load addresses, followed by module names and only includes loaded modules. The list does not include delay-loaded modules that have not been called.

The Get Loaded Modules option only supports Active Directory and Domino.

From the Command Line

Enabling trace from the command line is useful if you want a wider range of options.

▼ To Enable Tracing:

1 **Open a command window.**

2 **Start the Gateway, specifying the necessary trace command arguments.**

The following table describes the Gateway tracing command line arguments.

Argument	Description
-f	Specify the path to the trace file

Argument	Description
-l	Specify the level of tracing: ■ **Level 0**. Disables tracing. (Default) ■ **Level 1**. Traces the flow of control between components and generally defines a *low-detail* trace point that includes entry and exit from high-level functional methods. ■ **Level 2**. Generally defines a *medium-detail* trace point that includes entry and exit from every method, and information and data trace points for high-level functional methods. Level 2 adds the flow of control within each component, major decision points, and items of information. ■ **Level 3**. Generally defines a *high-detail* trace point that includes entry and exit from every method, information and data trace points for high-level functional methods, and significant subroutines. Level 3 adds lower-level decision points and items of information. ■ **Level 4**. Generally defines a *hyper-detail* trace point that includes everything traced in the other trace levels. Level 4 traces at a very low level and provides a level of detail that is seldom needed but might be useful in characterizing complex behaviors of some components. **NOTE**: Not all components support level 4. Trivial methods, such as getters and setters, generally do not have entry or exit trace points because they add overhead.

-m	Specify the maximum trace file size in kilobytes
	When the trace file reaches `-m` Kbytes, Identity Manager closes the current trace file, deletes any existing back-up files, renames the current trace file to the name specified by the `-f` argument with `.bk` appended, and opens a new trace file with the `-f` argument name.
	For example, if you specified `-f beeble.trc` on the command line, the following two files result after `-m` Kbytes are recorded:
	`beeble.trc.bk`
	`beeble.trc`
	Where `beeble.trc` contains the most recent traces.

Usage: `gateway -f name -l -m`

For example:

```
cd %WSHOME%\bin\winnt
gateway -d -p 11319 -f %CD%\gateway.trc -l 2 -m 500
```

The preceding invocation starts the Gateway with the following characteristics:

- `-d` – Use regular application (not a service)
- `-p 11319` – Use port 11319

  You must configure this port for Gateway resources from the Identity Manager resource configuration. For example, for an Active Directory resource

- `-f %CD%\gateway.trc` – Directory to which the trace output is written. Identity Manager writes the trace output to a text file in this directory.
- `-l 2` – Output level 2 of Gateway tracing.
- `-m` – Maximum size in Kilobytes of trace log file.

Note – If specified, Identity Manager saves `-f`, `-l`, and `-m` values in the registry, so the next time you run Gateway from the command line or as a service, the same values are used.

Identity Manager sends the Gateway trace output to the console *and* to a trace file.

How to Configure Tracing for the `PowerShellExecutor.dll` Add-On

The `PowerShellExecutor.dll` is an add-on that implements communication between the gateway and Microsoft PowerShell. The PowerShell is used to manage Exchange Server 2007 accounts. This add-on cannot share tracing facilities with the rest of the gateway and provides a similar stand-alone tracing facility as the rest of the gateway.

The trace configuration for the `PowerShellExecutor` is stored in the same registry key as the other gateway registry keys:

```
HKEY_LOCAL_MACHINE\Software\Waveset\Lighthouse\Gateway
```

You create this base key when you configure tracing through the Identity Manager debug pages or when you start the gateway with trace command arguments.

On shut down, the gateway writes the current `PowerShellExecutor` settings for the tracing to the registry. These settings include:

- `traceFileName`

  Content. File name for the trace output (registry type `REG_SZ`)

  Default. `" "`

  Name of the trace file to generate for the `PowerShellExecutor` tracing. Where the name:

  – Can be a fully qualified path, including the filename

  – Cannot end in a slash (`\`)

  The full path, except the file, provided in the `traceFileName` must exist.

  If configured, log rotation adds a timestamp to the configured filename after rotation, when the file is no longer active. This timestamp displays in the following format:

  `yyyyMMddHHmmss`

- `traceLevel`

Content. Trace level (registry type `REG_DWORD`)

Default. 0 (no tracing)

Allowed. 0–4

This key is shared with the rest of the gateway. The whole gateway always provides tracing at the same level.

- `traceMaxSize`

  Content. Maximum file size in bytes (registry type `REG_DWORD` or `REG_QWORD`)

  Default. 100000 bytes

  Minimum. 100000 bytes

  Tracing text is written as UTF–8 encoded text with a byte order mark to make it portable to other systems.

- `traceMaxFiles`

  Content. Number of trace files (registry type `REG_DWORD`)

  Default. 2

  Minimum. 1

  This setting controls the number of trace files to keep on the system. Setting the maximum number of files to keep to **1**, causes the file to be overwritten when the maximum size is reached. The oldest file, based on last write time, is removed when the maximum number of files is reached.

- `traceConfigInterval`

  Content. Time out in milliseconds (registry type `REG_DWORD`)

  Default. 300000 ms (5 minutes)

  Minimum. 60000 ms (1 minute)

All trace settings are reread from the registry based on this timeout value. In a production environment, consider setting this value to a large value, such as 24 hours, to minimize overhead.

How to Capture Dr. Watson Logs

If the Gateway encounters a serious problem and exits abnormally, you can send the resulting Dr. Watson logs to Sun Support for analysis.

Note – You must have administrator privileges on the system to view these logs.

▼ To Capture a Dr. Watson Log:

1. **Open the Windows Event Viewer.**
2. **Open the application log.**
3. **Look for an event with `DrWatson` source.**
4. **Open the event to view detailed information.**
5. **Ensure that the Bytes option is selected for Data.**
6. **Right-click in the display dialog and choose Select all from the menu.**
7. **Type `Ctrl-C` to copy the information.**
8. **Paste the information into Notepad and save the file.**
9. **Send the file in an email to Sun Support with a detailed description of the problem. Be sure to indicate which version of the Identity Manager Gateway you are running.**

Troubleshooting and Fixing Common Problems

Use the information provided in the following sections to help diagnose and fix problems you might encounter as you work with Identity Manager:

Note – For additional troubleshooting information, including the Identity Manager FAQ, go to the following URL:

`https://sharespace.sun.com/gm/document-1.26.2296/IdM_FAQ.html?`

You must sign up for a Share Space ID to access information provided on this site.

Working with Debugging Tools

You can use several different debugging tools to help identify and fix problems in your Identity Manager deployment. These tools include:

Using the Identity Manager Debug Pages

You can use Identity Manager Debug pages to help identity and fix problems in your deployment. For example, you can enable or disable tracing for various activities and objects, collect statistical information, verify that processes are running, or investigate bottlenecks and memory problems.

The following table describes the most commonly used Debug pages and their actual `.jsp` file names.

Note – For a comprehensive list of all Identity Manager Debug pages, open a command window and list the contents of the `idm/debug`directory.

TABLE 5–5 Identity Manager Debug Pages

Page Name	Use This Page to
Control Timings (`callTimer.jsp`)	Collect and view call timer statistics for different methods. You can use this information to track bottlenecks to specific methods and invoked APIs. You can also use options on the Call Timings page to import or export call timer metrics, such as. ■ How long to fetch initial User view (with no resources) during a scan ■ How long to refresh initial User view (including resources) during a scan ■ How long to evaluate policy on the User view ■ How long each user scan takes (including User view fetch, policy evaluation, and so forth) ■ How long to fetch a list of users for an access scan ■ How long to evaluate the attestation rule in access review **Note –** Call timing statistics are only collected while trace is enabled.
Edit Trace Configuration (`Show_Trace.jsp`)	Enable and configure trace settings for the Java classes provided with your Identity Manager installation. You can specify ■ Which methods, classes, or packages to trace and the level of trace. ■ Whether to send trace information to a file or to standard output and how dates and times are formatted in the trace output file. ■ Maximum number of trace files to store and the maximum size of each file. ■ Specify the maximum number of methods to be cached. ■ Indicate how to write data to the trace file and whether to write data to the trace file as data is generated or to queue and then write the data to a file.
Host Connection Pool (`Show_ConnectionPools.jsp`)	View connection pool statistics (if you are not using a data source), including pool version, how many connections were created, how many are active, how many connections are in the pool, how many requests were serviced from the pool, and how many connections were corrupted. You can also use the Host Connection Pool page to view a summary of the connection pools used to manage connections to the Gateway. You can use this information to investigate low-memory conditions.

TABLE 5–5 Identity Manager Debug Pages *(Continued)*

Page Name	Use This Page to
List Cache Cleared (`Clear_XMLParser_Cache.jsp`)	Clear recently used XML parsers from the cache and investigate low memory conditions.
Method Timings (`Show_Timings.jsp`)	Detect and assess hotspots at a method level. Use this page to gather information from Identity Manager methods, including: ■ Method names ■ How many times the methods were called ■ How many times the methods exited with an error status ■ Average time consumed by the methods ■ Minimum and maximum times consumed by invocations of each method
Object Size Summary (`Show_Sizes.jsp`)	Detect problematically large objects that can affect your system. This page shows the size of objects (in characters) stored in the repository, including the objects' total combined size, average size, maximum size, and minimum size. Click entries in the Type column to view the ID, name, and size of the largest configuration objects in the repository.
Provisioning Threads for Administrator Configurator (`Show_Provisioning.jsp`)	View a summary of provisioning threads in use by the system (a subset of the information available in `Show_Threads.jsp`).
System Cache Summary (`Show_CacheSummary.jsp`)	View the following information to investigate low-memory conditions: ■ Administrator-associated object caches ■ System object cache ■ User login sessions ■ XML parser cache
System Memory Summary (`Show_Memory.jsp`)	View how much total and free memory is available (in MB) when you are using memory-intensive functionality, such as reconciliation, to help determine whether there is sufficient memory allocated to the JVM. You can also use this page to launch garbage collection or to clear unused memory in the JVM for investigating heap usage.
System Properties (`SysInfo.jsp`)	View information about your environment.
System Threads (`Show_Threads.jsp`)	View which processes are running to verify that automated processes are running. Includes information about the process type, process name, priority, if the process is a daemon, and if the process is still alive (running).
User Session Pool Cleared (`Clear_User_Cache.jsp`)	Use the Session Pool Clearer page to investigate low memory conditions.

TABLE 5–5 Identity Manager Debug Pages *(Continued)*

Page Name	Use This Page to
Waveset Properties (`Show_WSProp.jsp`)	View and *temporarily* edit properties in the `Waveset.properties` file. Edited property settings remain in effect only until the next server restart.
XML Resource Adapter Caches Flushed and Cleared (`Clear_XMLResourceAdapter_Cache.jsp`)	Clear test XML resource adapters from the cache and use to investigate low memory conditions.

Note – See "Working With Identity Manager Debug Pages" on page 69 for more information about these Debug pages.

▼ To Access Individual Identity Manager Debug Pages

Before You Begin You must have the *Debug* capability to access and execute operations from the Identity Manager Debug pages. If you do not have the Debug capability, an error message results. Administrators and the Configurator are assigned the `Debug` capability by default.

1. **Open a browser and log into the Administrator interface.**

2. **Type the following URL to open the System Settings page:**

   `http://`*host***`:`***port***`/idm/debug`**

   where:

   - *host* is the local server on which you are running Identity Manager.
   - *port* is the number of the TCP port on which the server is listening.

     From this page, you can enable or disable tracing for various Identity Manager activities and objects and use the information displayed on these pages to troubleshoot problems in your deployment.

     Some Debug pages are not linked to the System Settings page, and you must type the page's `.jsp` file name to open the page. For example:

     `http://` *host***`:`***port***`/idm/debug/`***pageName.jsp***

     Where *pageName.jsp* is the particular Debug page you want to open.

Using Identity Manager IDE

The Sun™ Sun Identity Manager Integrated Development Environment (Identity Manager IDE) is Java application that enables you to view, customize, and debug Sun Identity Manager (Identity Manager) objects in your deployment.

Specifically, the Identity Manager IDE provides a graphical Debugger that you can use to debug Identity Manager forms, rules, and workflows. You can use this Debugger to set breakpoints and watches, step through code, examine and modify variables, examine classes and the callstack, follow threads, and run multiple sessions.

Instructions for installing and configuring the Sun Identity Manager Integrated Development Environment (Identity Manager IDE) are now available from the following URL:`https://identitymanageride.dev.java.net`.

Using Identity Manager System Monitoring

You can configure Identity Manager system monitoring to track system events. System monitoring collects and aggregates statistics at various levels to present a real-time view of system events, based on your specifications.

Viewing this information in dashboard graphs enables you to quickly assess system resources, view abnormalities, understand historical performance trends, and interactively isolate problems before looking at audit logs. Although dashboards do not provide as much detail as audit logs, dashboards can indicate where to look for problems in the logs.

For more information about dashboards and system monitoring, see Chapter 8, "Reporting," in *Sun Identity Manager 8.1 Business Administrator's Guide*.

Working With Adapter Logs

Adapter logs capture information about the adapter that is currently processing. You can use this information to monitor the adapter's progress and to diagnose and debug adapter problems.

Note – You must enable tracing and identify the methods for which tracing is requested before any logging can occur. Also, your customized adapter must include calls that create log entries for new methods.

Nearly every adapter has its own log file, path, and log level. You can specify the level of detail captured by the adapter log, along with these other values in the Logging section of the Synchronization Policy for the appropriate Identity Manager or Service Provider user type.

For more information about using adapter log files as a debugging tool, see "Troubleshooting Adapters" on page 132.

Debugging with DTrace

DTrace is a comprehensive, dynamic tracing framework for the Solaris operating environment. DTrace provides more than 30,000 probes into your production system and integrates user- and kernel-level tracing. You can use DTrace to monitor JVM activity. This facility also allows you to use the D language (similar to C or `awk`) to trace arbitrary data and expressions.

Debugging with JConsole

The Java Monitoring and Management Console (*JConsole*) is a Java Management Extension (*JMX*) technology-compliant graphical management tool bundled with JDK 5 (and later). JConsole connects to a running JVM and gathers information from the JVM MBeans in the connected JMX agent.

For example, you can use JConsole to

- Detect low memory
- Enable or disable garbage collection
- Enable or disable verbose tracing
- Detect deadlocks
- Control Identity Manager log levels
- Access information about operating systems resources (Sun's platform extension)
- Monitor and manage MBeans
- View information about the JVM and monitored values, threads running on the application, and class loading

---

Note – For more information about JConsole, see the article titled, *Using JConsole to Monitor Applications*. You can view this article from the following URL:

`http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

Debugging Errors Displayed in the Browser

If a red error message displays in the Identity Manager interface after you have performed an action, you might be able to view more complete information and further analyze the error by viewing and saving the page source.

To view the page source

- If you are using Internet Explorer, select View → Source from the menu bar.
- If you are using Netscape™, select View → Page Source from the menu bar.

If you still need help resolving the problem,

1. View the page source, and then select File → Save to save the file to your system.
2. Locate the error in your saved file.
3. Send the error information, the URL from the page where the problem occurred, and a description of the problem in an email to Sun Support for resolution assistance.

Troubleshooting Adapters

To troubleshoot an adapter, review the adapter's log file. Almost all adapters write their resource settings to a log file, and you can use this information to confirm that the adapter started and that all setting changes have been saved.

Note – You must enable tracing and identify the methods for which tracing is requested before any logging can occur. Also, your custom adapter must include calls that create log entries for new methods.

If necessary, review "Tracing the Identity Manager Server" on page 93 for instructions about how to enable tracing.

Most adapter log files are located in the `$WSHOME/config` directory and they are named `WSTrace1.log`.

Active Sync-enabled adapters that make log calls to the `ActiveSyncUtil` instance create a log file or set of log files in a directory specified by the `Log File Path` resource attribute. Be sure to check these log files for additional Active Sync-related log entries.

The information in this section is organized as follows:

To Debug an Adapter

Follow these general steps to debug your custom adapter.

1. Create a test program for your adapter, and be sure this Java file performs the following basic functions:
   a. Create a new resource
   b. Create a user
   c. Get a user
   d. Update a user
   e. Delete a user
   f. Perform create, get, update and delete operations on multiple users

   ---

   Note – A sample test file (`SkeletonResourceTests.java`) is provided in the `/REF` directory on your installation CD.

   ---

2. Set an appropriate logging level for debugging.

For example, for the first debugging pass, increase the logging level to 4 (maximum debugging output), set the log file path, and specify a maximum file size.

When you start the adapter, all of the resource settings are written to the log file. You can use this information to validate that the adapter started and that all setting changes were saved.

3. Compile and test your adapter.
   - To compile the test program, open a command window and enter the `javac -d . test/filename.java` command. This command creates the class file in the appropriate `com/waveset/adapter/test` directory.
   - To test your new adapter using this class file, be sure that your compiled adapter is in the `com/waveset/adapter` directory and use the following command to run the adapter:

     `java– D waveset.home=`*path* `com.waveset.adapter.test.`*MyResourceAdapter*
4. Create an HTML help file for your resource.

Note –

- Example help files are supplied in the `idm.jar` file located in the `com/waveset/msgcat/help/resources` directory.
- See *Sun Identity Manager Deployment Reference* for information about how to include online help with the application.

---

5. (*For Active Sync-enabled adapters only*) To reset synchronization on the last resource, delete the `XmlData SYNC_`*resourceName* object.
6. Read the error log and modify the adapter.
7. Reset the logging level.

   For example, specifying Level 2 debugging yields information about the adapter settings and any errors, but limits the amount of log detail to a manageable level.
8. Before starting Identity Manager, you must identify the new adapter in the `$WSHOME/config/Waveset.properties` file by placing the adapter name under the `resource.adapters` entry or Identity Manager cannot recognize the adapter.
9. Install your adapter and its associated help file into Identity Manager.

Note – Before Identity Manager can recognize an instance of a new adapter in the display, you must create a new resource of that type from the List Resource page.

From this page, select New → *new adapter* and use the Resource Wizard to create the new adapter.

---

10. Use Identity Manager to create a resource and a user on that resource.

Tip – When troubleshooting an Active Sync-enabled adapter, if you edit the `XmlData SYNC_`*resourceName* object to remove the `MapEntry` for the Active Sync synchronization process from the Debug page, the adapter starts over from the first detected change.

If you used the IAPI event, you must set the `Property()` method to store synchronization state for the resource, such as a `last change processed` value. Setting this method is very useful for troubleshooting adapters. You can set the adapter to run and ignore past changes. Subsequently, you can modify the adapter and see your change results in the adapter log file.

If your resource is an Active Sync resource, you might see additional information if you enable logging on the resource edit page. Set the logging level (0-4) and the file path where the log file will be written (as *resource_name*`.log`).

11. (*For Active Sync-enabled adapters only*) Restart synchronization for the last resource.

To Debug `LoginConfig` Changes

To debug `LoginConfig`-related changes to your adapter, you must

1. Enable trace for the selected files and trace the following classes at Method/Class Level 1 trace:
   - `com.waveset.security.authn.WSResourceLoginModule`
   - `com.waveset.session.LocalSession`
   - `com.waveset.session.SessionFactory`
   - `com.waveset.ui.LoginHelper`
   - `com.waveset.ui.web.common.ContinueLoginForm`
   - `com.waveset.ui.web.common.LoginForm`
2. Test Single Sign-On (SSO) pass-through authentication login through Telnet as follows:
   a. After correctly configuring the SSO login module, `telnet` directly to the HTTP port and send an HTTP request to `login.jsp`.
   b. Paste the following request, which contains an SSO login module that looks for the `sm_user` HTTP header, into your `telnet` session:

```
HEAD /idm/login.jsp HTTP/1.0
Accept: text/plain,text/html,*/*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: LOCALHOST
sm_user: Configurator
```

   A trace displays to indicate that your user has logged in correctly. For example:

```
2003.07.08 14:14:16.837 Thread-7
WSResourceLoginModule#checkForAuthenticatedResourceInfo()
Found authenticated resource accountId, 'Configurator@Netegrity
SiteMinder'on Identity Manager user 'Configurator'.
null null 2003.07.08 14:14:16.837
Thread-7 WSResourceLoginModule#checkForAuthenticatedResourceInfo()
Exit null null 2003.07.08 14:14:16.837 Thread-7 WSResourceLoginModule#login()
Exit, return code = true null null 2003.07.08 14:14:16.847
Thread-7 LocalSession#login() Login succeeded via Netegrity SiteMinder
null null 2003.07.08 14:14:16.847 Thread-7 LocalSession#login()
Overall authentication succeeded null null 2003.07.08 14:14:16.897 Thread-7
LocalSession#checkIfUserDisabled()
Entry null null 2003.07.08 14:14:16.897 Thread-7
LocalSession#checkIfUserDisabled() Exit
null null 2003.07.08 14:14:16.927 Thread-7 LocalSession#login()
Exit null null
```

To Debug Adapter Connection Problems

This section describes methods for debugging some common adapter connection problems.

The topics in this section are organized as follows:

Note – Generally, you can identify adapter connection issues by tracing the adapter class `com.waveset.adapter.`*adapter_classname*. For example:

```
com.waveset.adapter.ADSIResourceAdapter
```

If necessary, review instructions for enabling trace in "Tracing the Identity Manager Server" on page 93.

Adapter Authentication Problems

Some common authentication problems include

- Missing authentication properties

  You must include a property name for the specified DataSource type in the set of names output in the trace.

- An Identity Manager user is missing a matching resource account

  If authentication succeeds for the resource adapter, but an exception occurs indicating that no Identity Manager user could be found with a matching resource account ID, be sure that the resource `accountId` associated with the user is the same as the `accountId` returned by your resource adapter's authenticate method.

  To verify the Identity Manager user's resource `accountId`, review Identity Manager's Edit Trace Configuration page (`debug/Show_Trace.jsp`). If a mismatch exists, change the content of the name being returned by your `authenticate` method or change your resource's ID template. The template must generate a resource `accountId` that matches the `accountId` being returned by the `authenticate` method.

Active Sync Adapter Problems

The most common problems with custom Active Sync adapters are form-related. These errors generally occur because you have not provided necessary information, such as password or email information, in a required field.

Identity Manager prints form validation errors to the adapter log after the final XML of the view. For example:

```
20030414 17:23:57.469: result from submit (blank means no errors):
 20030414 17:23:57.509: Validation error: missing required field password
```

Identity Manager also prints all messages to the adapter log. These messages include account creation and update times, adapter errors, and a summary of the schema map data.

Active Sync resource adapters store information about the last change processed in the `SYNC.resourceName XMLData` object.

Domino Gateway Adapter Problems

Following are some common Domino gateway and adapter configuration errors and instructions for fixing these problems:

- If an error message states the 'New Domino Gateway' resource is not accessible and the connection is refused, try stopping and restarting the Identity Manager Gateway.
- If an error message states no ID file name is specified and the path to the userID file is set incorrectly, specify a target location for the userID file and edit the resource adapter to set this attribute to a correct path. Typically, the target location for the userID file is the directory into which you installed the Gateway.
- If an error message states you are not authorized to use the server, you have not set the correct access permissions for the ID file. Specify the correct permissions for this file and retry.

Mainframe Host Adapter Problems

When RACF, ACF2, or TopSecret host adapters fail to reuse or cache connections, users are forced to log in frequently, which negatively impacts performance. Generally, the cache timeout setting causes this problem.

To check the cache timeout setting, trace Identity Manager's adapter connection pool as follows:

1. From Identity Manager's Edit Configuration Object page, trace the `com.waveset.adapter.HostConnPool#reapConnections` method at level 4.

   If necessary, review instructions for enabling trace in "Tracing the Identity Manager Server" on page 93.
2. Capture trace for a sufficiently long period of time (*at least* 30-60 minutes), while the adapter performs operations.
3. Review the trace output in the application server `stdout` or trace file and look for `Info reaping connection` entries.

   If this entry occurs more than once every 30 minutes, you have a good indication that connections are being timed out unnecessarily.

   To resolve this problem, increase the `Idle Timeout` resource attribute value to prevent connections from being reaped too frequently. The `Idle Timeout` attribute controls how long a connection remains idle before the connection is logged out. The default value is 90 seconds, which causes new logins to occur frequently.

   Ideally, specify a value that is greater than the average idle time for your deployment environment. For example, adjust the `Idle Timeout` attribute to 30 minutes (1800000 milliseconds) or more.

PeopleSoft Adapter Problems

This section describes methods for troubleshooting the following PeopleSoft adapter problems:

- Executing a "Test Connection" from the PeopleSoft resource adapter causes a failure with a generic exception.
  1. Open the `Waveset.properties` file and set `exception.trace=true`.
  2. Retry the test connection, and if you see the following results:

```
FormState: expansion complete
java.lang.NullPointerException: PSProperties not loaded from file
WavesetException:
WavesetException:
==> com.waveset.util.WavesetException:
FormState: derivation

Log in to the PeopleSoft web interface to verify that you
are using the correct UID and password.
```

- The PeopleSoft application server logs are not showing login attempts.

  This problem generally occurs because you did not use the `psjoa.jar` file supplied with the PeopleSoft installation to which you are connecting. (See the *Sun Identity Manager 8.1 Resources Reference* for more information about the PeopleSoft resource adapter).

SAP Adapter Problems

If an error results when you try to test the connection from an SAP or SAP HR Active Sync adapter to the SAP system, open a command window and run this command from your installation directory `WEB-INF/lib`:

```
java -jar sapjco.jar
```

The `sapjco.jar` command shows which version of the SAP Java Connector (JCO) is installed and whether the adapter is installed correctly. The command also returns the Java™ Native Interface (JNI™) platform-dependent and the RFC libraries that communicate with the SAP system.

If these platform-dependent libraries are not found, consult the SAP documentation to find out how to correctly install the SAP Java Connector.

UNIX Adapter Problems

This section contains information about debugging some common problems with UNIX adapters.

- If you see timeout errors when provisioning to a UNIX resource adapter, you can determine where the provisioning process is failing by tracing the `com.waveset.adapter.ScriptedConnection` method at Method/Class level 4, which gives you the maximum logging output.
- When becoming `root` to perform administrative commands, if the resource adapter executes a `su root` command instead of a `su - root` command, the environment does not inherit any of the custom environment variables defined for root; including any custom prompts (environment variable of PS1).

  When configuring a UNIX adapter, you can determine which prompt to enter into the Root Shell Prompt field as follows:

  1. Telnet or `ssh` to the system as the user you specified in the Login User field.
  2. After typing the password and logging in, type **`su root`** without a dash and press return.
  3. Type the root password.
  4. The next prompt displayed is the prompt you must enter into the Root Shell Prompt field.

Troubleshooting Auditor

You can trace the following methods to troubleshoot issues with Identity Auditor:

- `com.sun.idm.auditor.policy` to trace issues with Audit Scans.
- `com.sun.idm.auditor.accessreview` to trace issues with Access Reviews.
- `com.sun.idm.auditor.report` to trace issues with Audit Reports.
- `com.sun.idm.auditor.view` to trace issues with Auditor Views.

In addition, you can set the following hidden flags by modifying Forms or `TaskDefinitions`:

- **Audit Policy Scan Task - maxThreads (int)**. Number of concurrent threads used. (Defaults to 5).
- **Audit Policy Scan Task - userLock (int)**. Time (in milliseconds) to wait to acquire a lock on the User object. (Defaults to 5000)
- **Audit Policy Scan Task - scanDelay (int)**. Time (in milliseconds) to delay between launching a new scan thread. (Defaults to 0, no delay)
- **Audit Policy Scan Task - clearuserLocks (boolean)**. If `true`, the scanner attempts to clear the user lock before scanning.

In addition, the Show Timings page (`/debug/callTimer.jsp`) provides the following information:

- How long it takes to fetch the initial User view, with no resources, during the scan
- How long it takes to refresh the User view, including resources, during the scan
- How long it takes to evaluate the policy on the User view
- How long each user scan takes, including User view fetch, policy evaluation, and so forth
- How long it takes to fetch a list of users for access review
- How long it takes to evaluate the attestation rule in access review

Troubleshooting `ClassNotFound` Exceptions

In the event of a `ClassNotFound` exception error, verify that the missing class is included in the server's classpath. If the classpath is configured properly, try configuring your application server such that the application class loader loads before the parent class loader. Sometimes loading the application classpath before the server classpath can resolve this issue. Consult your application server documentation for instructions.

Troubleshooting Form Problems

This section describes some common form problems and how to fix these problems.

- You added a <script> tag to your user form, but see the following exception when trying to access the user form:

  ```
  java.lang.NoClassDefFoundError: org/mozilla/javascript/NativeScript
  ```

  You must put the WEB-INF/lib/javascript.jar file into the application server's classpath. For example:

  – If you installed Tomcat as a service (Windows only), edit the following registry key:

    HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tomcat\Parameters\JVM Option Number 0

    Append the path to your jar file to the end of the JVM Option Number 0 key string value, wherever it is located. For example:

    ;C:\tomcat\lib\javascript.jar

  – If you just started Tomcat from the command line, move the jar file into the tomcat/lib directory.

- You customized a Tabbed User Form, but when you try accessing the form through User Creation, you see the following exception:

  ```
  com.waveset.util.WavesetException: Maximum form stack depth exceeded
  ```

  You are limited to pushing 1000 hard-coded elements on the stack and you exceeded this limit.

  If you added a check to detect Field containers that include a FieldRef or Field that matches the name of the Field container, you might have inadvertently created a circular reference on the form. To fix this problem, change the FieldRef or Field name.

- You used a MultiSelect field in a form for LDAP users, assigned groups to a user in this field, edited the user again, and then see the same group on both sides of the MultiSelect. For example:

  – **Left side value**. cn=Group1,dc=test,dc=Com

  – **Right side value**. cn=Group1,dc=test,dc=com

    In this example, the LDAP baseContext resource field is set to dc=test,dc=**com** and the LDAP groups are listed as dc=test,dc=**Com**, which causes a problem because LDAP is not case sensitive, but the MultiSelect widget is case-sensitive.

    To alleviate this problem, change the baseContext on the LDAP resource to match the case in your LDAP resource dc=test,dc=Com or use the <upcase> XPRESS function to uppercase both the left and right side displays.

Troubleshooting the Gateway

When troubleshooting the Sun Identity Manager Gateway, it is often useful to run the Gateway from the command line. Using command line options allows you to input a wider range of start-up options, which includes starting the Gateway as a normal application instead of a service and running the Gateway on a different port.

Note – You must `kill` the Identity Manager Gateway as a service before running it from the command line. For example, type

```
gateway.exe -k
```

The following table describes the Gateway command line arguments.

Argument	Description
-i	Install this program as an NT service, with specified startup
-r	Remove this program from the Service Manager
-s	Start the service
-k	Kill the service
-t	Set start-up for an existing service
-d	Debug, and run as a regular application
-p	Specify a TCP/IP port number (Default is 9278)
-f	Specify the path to the trace file
-l	Specify the level of tracing (Default is 0, no information)
-m	Specify the maximum trace file size in kilobytes
-v	Display the version

Usage: `gateway -i n -r -s -k -t n -d -p n -f name -l n -m n -v`.

Note –

- The `-d` and `-s` options are mutually exclusive.
- See "From the Command Line" on page 121 for more information about the Gateway tracing levels.

You can also use the Identity Manager Gateway Debug page (`debug/Gateway.jsp`) to troubleshoot the Gateway. See "How to Configure Tracing from the Gateway Debug Page" on page 120 for more information.

Troubleshooting Java Code Problems

If you have the basic Java programming skills required to work with Identity Manager, you should be able to diagnose and resolve most Java code problems.

However, a fairly common problem occurs where someone opens a connection to the database but does not close the connection properly. If you do not close the connection properly, performance issues result.

Troubleshooting Low Memory Conditions

This section describes tools that you can use to investigate low memory conditions, including:

- "From the Identity Manager Debug Pages" on page 142
- "From JConsole" on page 143

From the Identity Manager Debug Pages

Note – You must have the *Debug* capability to access and execute operations from the Identity Manager Debug pages. Administrators and the Configurator are assigned this capability by default.

If you do not have the Debug capability, an error message results.

You can open the following Identity Manager Debug pages from the Administrator interface to monitor how much memory is being used by your system:

- **Host Connection Pool** page (`debug/Show_ConnectionPools.jsp`). View a summary of connection pool statistics (if you are not using a data source), including the pool version, how many connections were created, how many are active, how many connections are in the pool, how many requests were serviced from the pool, and how many connections were destroyed.

  You can also use the Host Connection Pool page to view a summary of the connection pools used to manage connections to the Gateway. You can use this information to investigate low-memory conditions.
- **List Cache Cleared** (`debug/Clear_XMLParser_Cache.jsp`). Clear the cache of recently used XML parsers.
- **Private collection pool** (`debug/Show_JDBC.jsp`). View a summary of the cache of Java DataBase Connectivity (JDBC™) connections being used by the repository and some resource adapters.
- **System Memory Summary page** (`debug/Show_Memory.jsp`). View the used and total memory in the system. You must click the Garbage Collect button to get the most current used memory value.
- **System Memory Summary page** (`debug/Show_Memory2.jsp`). View an updated `Show_Memory.jsp` page that allow you to clear all unused memory in the JVM so you can investigate heap usage.
- **User Session Pool Cleared** (`Clear_User_Cache.jsp`). Clear the cached sessions for recently logged in users.
- **XML Resource Adapter Caches Flushed and Cleared** (`Clear_XMLResourceAdapter_Cache.jsp`). Clear the cache of the test XML resource adapter.

From JConsole

Use the Java Monitoring and Management Console (*JConsole*) to detect low memory and deadlocks. JConsole is a Java Management Extension (JMX™) technology-compliant graphical management tool that is co-packaged with JDK 5 (and later).

JConsole accesses the memory system, memory pools, and MBeans garbage collector to provide information about memory use such as memory consumption, memory pools, and garbage collection statistics. In addition, You can use JConsole to monitor MBeans for information about current heap memory use and non-heap memory use.

Note – For information about using JConsole to monitor applications that run on the Java platform, see *Using JConsole to Monitor Applications*. This document is available from the following URL:

http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html

Troubleshooting PasswordSync Problems

When you are trying to troubleshoot problems with PasswordSync, review the following logs for information:

- **PasswordSync Error Logs**. PasswordSync writes all failures to the Windows Event Viewer. (For more information about Event Viewer, see Windows' Help.) The source name for error log entries is *PasswordSync*.
- **PasswordSync Trace Logs**. PasswordSync writes all trace logs to the file location specified when you configured tracing. See "Using the PasswordSync Configuration Tool" on page 105.

Some common PasswordSync problems and solutions include

- PasswordSync is not propagating password changes from the Windows server to Identity Manager.

  PasswordSync relies on the registry settings when creating a connection from Active Directory to the Identity Manager Server. PasswordSync reads the registry and processes the settings, but PasswordSync does not perform any checks to see if it can create a connection.

  The following example shows a registry entry for a PasswordSync server. This example includes the default registry setting values, but does not show all of the settings used by PasswordSync.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Waveset\Lighthouse\PasswordSync]
"reinitIntervalMilli"=dword:0001d4c0
"securityIgnoreCertRevoke"=dword:00000000
"securityAllowInvalidCert"=dword:00000000
"directMode"=dword:00000001
"lhuser"="config"
"lhcred"="rsVtQZpa5Ys="
"endpointURL"="http://10.10.10.10:8080/idm/servlet/PasswordSync"
"installdir"="C:\\Program Files\\Sun Microsystems\\Sun Identity Manager
PasswordSync"
"tracelevel"=dword:00000000
"tracemaxKB"=dword:00002710
"tracefile"="C:\\Program Files\\Sun Microsystems\\Sun Identity Manager
PasswordSync\\trace.log"
```

  If you have not enabled tracing at an appropriate level, PasswordSync does not log connection failures in much detail. To see more detailed trace information, edit the PasswordSync registry settings as described on "Editing the Registry Keys" on page 106. Specify `tracelevel` 4 to output the maximum trace information, and change the `tracefile` value to point to a writable file. For example:

```
"tracelevel"=dword:00000004
"tracefile"="C:\\Program Files\\Sun\\IdentityManager\\PasswordSync\\pwicsvc.log"
```

  The registry settings will be reread based on the `<i>reinitIntervalMilli</i>` setting in the registry. After rereading the registry settings, PasswordSync automatically starts or stops tracing, depending on the trace parameters set in the registry. For each intercepted password change, PasswordSync logs the actions taken to push the password to Identity Manager.

- If a connection fails during creation, you might encounter the following situations:

Note – Each of these situations has its own error code and set of log entries. Identity Manager removes the date, time stamp, and process number from these entries to keep them short.

- An incorrect or unreachable URL error that occurs when the server cannot be reached, is not running, or does not reply with a correct response.

  Check that PasswordSync can access the server and page.

  - ❒ Be sure the server is running and that you have configured your firewalls and routers correctly.
  - ❒ Check the application server to be sure it is running, and that PasswordSync can connect to the endpointURL without the application path. If PasswordSync can does not return a page or an error, the application server is not running.
  - ❒ Check the servlet response by opening the endpointURL in a standard browser. If you do not see an error that starts with: `com.waveset.util.WavesetException` see if the servlet is compiling and available..
  - ❒ JMS usage for PasswordSync relies on the `jms.jar` being available in the classpath. The following exception message displays if you access the endpointURL without the correct file in place:

    ```
    com.waveset.util.WavesetException: A JMS request arrived, but
    JMS PasswordSync is unavailable. Is JMS jar file available?
    ```

- An incorrect user name error generally occurs when the userID stored in the `lhuser` entry is incorrect. Use the `Configure.exe` utility to replace the user or replace the `lhuser` registry key value with a valid userID.
- An incorrect password error generally occurs when the password stored in the `lhcred` entry is not correct when used in combination with the userID stored in `lhuser`. Use the `Configure.exe` utility to replace the password, but do not manually edit the `lhcred` registry key.
- A garbage in the password entry error generally occurs when the registry key is corrupted and or when the registry key is manually edited, which causes garbage in the password entry.
- This situation causes the process to hang in `RAEncryptor::Decrypt3DES` and PasswordSync cannot decrypt the entry. Use the `Configure.exe` utility to replace the password.

Troubleshooting Reconciliation Problems

When you are trying to troubleshoot problems with a reconciliation task, review the Reconciliation Status Debug page (`debug/Show_Reconciler.jsp`) to see what the resource threads are working on.

Some common reconciliation problems include

- Reconciliation does not start.
  1. Open the System Threads Debug page (`debug/Show_Threads.jsp`) to see if the Reconciler*server_name* task is running.
  2. If the task is not running, open the System Settings page and click Trace Scheduler.

     Running the Scheduler restarts the Reconciler *server_name* task.
  3. Check the System Threads page again, and if the Reconciler *server_name* task has not restarted, the Scheduler might be hung.
  4. Restart your application server.
- Reconciliation fails for a certain user object.
- If the object exists in Identity Manager, try editing the object directly. If the Identity Manager account does not exist, load the one account from the resource.
- Verify the reconciliation user has the proper access rights.
- Try extracting the object to a file using the same code path as reconciliation.
- Open the System Memory Summary Debug page (`debug/Show_Memory.jsp` or `debug/Show_Memory2.jsp`) and verify you have enough free memory.

Troubleshooting Repository Connection Problems

Identity Manager's `lh` commands are very useful when you are troubleshooting connection problems. These commands use Identity Manager's web application installation, but remove the application server from the equation.

This section describes the following

Using `lh` Commands to Debug Problems

This section describes how to use the `lh` commands; starting with using the more basic commands and progressing to using commands that exercise most of Identity Manager.

After becoming familiar with these debugging tools, you can develop your own variations for using these `lh` commands.

Using `lh setRepo -c -n`

Use the `lh setRepo -c -n` command to perform the most basic connection test, which allows you to examine the current repository location *without connecting*. You can use this command to verify that parameters, such as URL and JDBC driver, are correct.

- If the connection is successful, you can read the `ServerRepository.xml` bootstrap file.
- If the connection fails, try to solve this failure first. A decryption error is the most common cause of this failure. For example, you might have a J2EE mismatch or classpath conflict.

Using `lh setRepo -c -v`

Use the `lh setRepo -c -v` command to *connect to* and examine the current repository location. (The `-v` provides verbose output.) You can use this command to exercise almost all of the Repository code without requiring the Identity Manager server.

- If the connection is successful, then you are successfully connected to the current repository location.
- If the connection fails, try to solve this problem first. Solving your connection problem can be very helpful in resolving DNS, firewall, or remote connection privilege problems.

Note – For more information, see "Testing DataSource Connections" on page 149.

Using `setRepo`

Use the `setRepo` command throughout the debugging process, to specify a new repository location or to set the repository to the same location.

You can use this command to confirm that all of the necessary components, such as the JAR files, are in place. The `setRepo` command also lets you vary connection information, such as `userid` and `password`, to debug table ownership or privilege problems.

Using `lh console`

Use this command to actually start an Identity Manager Server using the JAR files in the `WEB-INF/lib` and the classes in `WEB-INF/classes` under `WSHOME`. The `lh console` command uses your Identity Manager installation environment and actually starts the Identity Manager application, but *removes* the application server from the equation.

- If the connection is successful, the problem is specific to the application server environment or configuration.
- If the connection fails, review the failure messages.
- If the connection failure is the same as the application server failure, you have reproduced this failure with significantly fewer variables.

- If the failure appears to be different from the application server failure, try fixing the Identity Manager connection problem first because there are fewer variables and more of these variables are under Identity Manager control.

Testing DataSource Connections

If you are testing a DataSource connection, the `lh setRepo -c` command might fail.

This failure is especially likely if you configured Identity Manager to use the application server's database connectivity service or the application server's directory service. These services often work only in the environment that a running application server provides to a web application.

Initially, approach the DataSource configuration you want in a step-by-step manner. Once you are comfortable with these steps, you can adapt your approach to suit your needs.

1. Try using a direct JDBC DriverManager connection, such as a non-DataSource connection, that bypasses the application server's database connectivity service.
2. Use a DataSource, but store the DataSource object in a directory service *other than* application server's directory service.

   Note – If you have no other directory service available, you can download a free directory service, including the reference implementation of JNDI that uses only the local file system.

   If these steps work, you have localized the problem to the application server.

   Then, if useful, you can add the application server's database connectivity service or the application server's directory service, whichever service works outside of the environment that the application server provides to web applications.

Troubleshooting Server-Related Problems

You can analyze your application server logs for fatal errors and other server-related problems.

To troubleshoot server problems, use the application server's Administrative Console to increase the logging level for each module. For more information, see the product documentation supplied with your server.

Most application servers have a standard location for standard out files (`stdout`) and standard error files (`stderr`) for the JVM running the application server. To analyze your application server logs, locate the logs directory or the log files specified for your Identity Manager application server.

- Open the `stdout`file to view minor messages and other tracings.
- Open the `stderr` file to view fatal and critical exceptions.

Note – You will see Identity Manager start up and shut down the messages in this trace output.

Beginning with Identity Manager Version 7.1, a sealing violation exception occurs in the application server log when you use Identity Manager with Oracle 10g on Sun Application Server Enterprise Edition 8.2.

This exception generally occurs if you are using more than one Java Archive file (JAR file) containing Oracle JDBC drivers.

To prevent this problem, be sure the `CLASSPATH` contains only one JDBC driver JAR file, and that you use the `ojdbc14_g.jar` provided with Oracle 10g. In addition, you must use the `ojdbc14_g.jar` provided by the Oracle 10g installation to ensure correct operation.

Troubleshooting Service Provider Problems

If you are using the Sun Identity Manager Service Provider End User Login page in WebSphere, and a `javax.servlet.UnavailableException` occurs with a 404 error displayed in your browser, you must reset some properties in the IBM 1.5 JDK.

Use the following steps:

1. In the `was-install/java/jre/lib` directory, rename the `jaxb.properties.sample` to `jax.properties` and uncomment these two lines:

   ```
   javax.xml.parsers.SAXParserFactory=
           org.apache.xerces.jaxp.SAXParserFactoryImpl
   javax.xml.parsers.DocumentBuilderFactory=
           org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
   ```

2. Save the file and restart the application server.

Troubleshooting an SPML Configuration

To test an SPML configuration:

1. Open the Connect page and click Test.

   A dialog indicating that the connection was successful pops up.

2. Open the Schema page and click Submit.

   The system displays a hierarchical view of the schemas supported by the Identity Manager server.

   If you cannot establish a successful connection

   - Double-check the URL you entered.

- If the error you receive contains phrases such as `no response` or `connection refused`, then the problem is most likely the host or port used in the connection URL.
- If the error suggests that a connection was made, but the web application or servlet could not be located, the problem is most likely in the `WEB-INF/web.xml` file.

Troubleshooting Upgrades

If you encounter problems during the upgrade, check the upgrade log files located in the `$WSHOME/patches/logs` directory. The file names for the logs are based on a timestamp and the stage of the upgrade.

If, following an upgrade, Identity Manager fails to start with the following exception, your JDK/JRE may be the problem:

```
java.lang.IllegalStateException: Error attempting to decrypt:
Given final block not properly padded
```

Verify that you are using a JDK/JRE supplied by the same vendor that you were using previously. For example, you cannot upgrade to a Sun JDK if previously you were using a JDK from IBM. To fix this problem, uninstall the JDK/JRE and install the JDK or JRE from your previous vendor.

◆ ◆ ◆ CHAPTER 6

Errors and Exceptions

This chapter describes the error and exception messages generated by Identity Manager.

The topics in this chapter include:

Before Working with Identity Manager Messages

Review the following information before you start working with Identity Manager error and exception messages:

Important Notes About Identity Manager Messages

Be sure to read the following information before you start working with Identity Manager error and exception messages:

- Examples in this chapter use a locale (*xx_XX locale*) that was devised for example purposes only.
- When you interpret error messages, be aware of the following:
  - Some messages have different keys but display the same error message text.
  - Some messages are used by multiple components.
  - Exceptions are generally listed by exception type, component, or both.
  - Some exceptions are caused by internal programming errors that cannot be viewed by Identity Manager users.

  - Some exceptions are simple wrappers and their parameters are the entire exception. For example, Identity Manager exceptions wrap resource messages, adapter code, and multipart exceptions, such as password policy violations.
- If you are using parameterized messages with single or double quotes in the message, you must use an additional single or double quote to escape the message. (You will see only one quote symbol when the message is output to the system.)

  For example, the following message begins and ends with two *single* quote marks:

  ```
  "'0}"
  ```
- If you need help diagnosing problems, contact Sun Technical Support:

  `http://www.sun.com/service/online/us`

Related Documentation and Web Sites

In addition to the information provided in this chapter, consult the documents and web sites listed in this section for information related to tuning Identity Manager for your deployment.

Recommended Reading

See the following documents for information related to Identity Manager error and exception messages:

- For information about creating a customized message catalog, see Chapter 9, "Customizing Message Catalogs," in *Sun Identity Manager Deployment Guide*.
- For information about creating or editing System Log reports, see "SystemLog Reports" in *Sun Identity Manager 8.1 Business Administrator's Guide*.

Useful Web Sites

The following table describes some web sites you might find useful.

TABLE 6–1 Useful Web Sites

Web Site URL	Description
`http://sharespace.sun.com/gm/document-1.26.2296`	Identity Manager FAQ on Sun's Share Space **Note**: You must sign up for a Share Space ID to access this FAQ.

TABLE 6–1 Useful Web Sites *(Continued)*

Web Site URL	Description
http://sunsolve.sun.com/	Sun web site containing diagnostic tools, forums, features and articles, security information, and patch contents **Note**: The information on this site is partitioned into three areas: ■ **Internal**. Sun employees only ■ **Contract**. Available only to customers with contract access ■ **Public**. Available to everyone
http://www.sun.com/service/online/us	Sun Technical Support web site

About Identity Manager Errors and Exceptions

This section describes the Identity Manager error and exception message system and the components that can generate errors and exceptions.

The information is organized into the following sections:

- "Where Messages Are Stored" on page 155
- "How Messages Are Displayed" on page 156
- "Error Severity Levels" on page 157

Where Messages Are Stored

Error messages are stored as follows:

- Identity Manager messages are stored in a `WPMessages.properties` file in the `idmcommon.jar` and in the `RAMessages.properties` file in the `idmadapter.jar`.
- The `WPMessages.properties` file is located in *project_directory*`/waveset/idm/common/src/com/waveset/msgcat`.
- The `RAMessages.properties file` is located in *project_directory*`/waveset/idm/adapter/src/com/waveset/adapter`.

  Sun Identity Manager Service Provider uses a standard Java message resource bundle for displaying strings in the user interface. This resource bundle is normally included in the `idmspe.jar` file and must be extracted if you plan to customize message strings.

  ```
  $ cd $WSHOME/WEB-INF/classes
  $ jar xvf ../lib/idmspe.jar com/sun/idm/idmx/msgcat/I
  DMXMessages.properties
  ```

- Identity Auditor messages are stored in the `AUMessages.properties` file, which is located in

project_directory/waveset/idm/auditor/src/com/sun/idm/auditor/msgcat/AUMessages.properties

- The Data Exporter's default implementation includes its own message bundle, called `WICMessages.properties`. The `WICMessages.properties` file is located in the `exporter.jar` file in

  `com/sun/idm/warehouse/msgcat/WICMessages.properties`

Note – You can create a customized message catalog to add messages or to modify messages provided with the system.

See Chapter 9, "Customizing Message Catalogs," in *Sun Identity Manager Deployment Guide* for instructions.

How Messages Are Displayed

For easy identification, Identity Manager displays page-level error and exception messages along the top of the page as boxed text with a unique error icon.

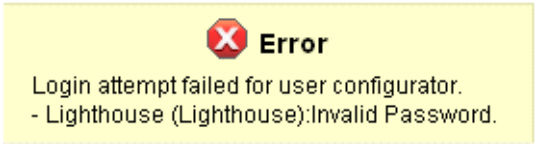


FIGURE 6–1 Example Login Authentication Error

Note – In the Identity Manager messaging system, items displayed as `{0}` or `{1}` represent parameters that are supplied by code. For example:

```
The file is larger than the maximum supported length of
 {0} bytes.
```

In this exception, the code replaces the `{0}` with the parameter value representing the maximum number of bytes.

Error Severity Levels

Within Identity Manager, error severities are defined as follows:

- **Fatal**. A severe error that causes your system to crash, resulting in the loss or corruption of unsaved data.
- **Error**. A severe error that might cause the loss or corruption of unsaved data. Immediate action must be taken to prevent losing data.
- **Warning**. Action must be taken at some stage to prevent a severe error from occurring in the future.
- **Info**. An informative message, usually describing server activity. No action is necessary.

Note – You can check the Identity Manager System Log for more information about an error or exception message. (See "Viewing Errors in the System Log Report" on page 157.)

Viewing Errors in the System Log Report

System Log reports can provide information about errors generated by Identity Manager. The System Log report consists of the error's timestamp, severity, server name, component name, error code or ID, stack trace (structure of the execution stack at that point in the program's life), and error text information.

You can use Identity Manager's Administrator interface or command-line interface to run and view System Log reports.

Note – Instructions for creating and editing System Log reports are provided in *Sun Identity Manager 8.1 Business Administrator's Guide*.

▼ To Run a System Log Report From the Administrator Interface

Perform the following steps to run a System Log report from the Administrator interface:

1 **Log in to the Identity Manager Administrator interface.**

2 **Select Reports → Run Reports to open the Run Reports page.**

3 **Locate the appropriate System Log Report entry in the Report Type column, and then click the Run button in that same row.**

The Report Results page displays, listing the system messages that were reported during the specified interval. For example, Figure 6–2 depicts information about two system messages.

Report Results

System Messages from the previous week

Monday, February 25, 2008 3:00:20 PM CST

Lists all system messages reported during the past 7 days

Number of records reported: 2

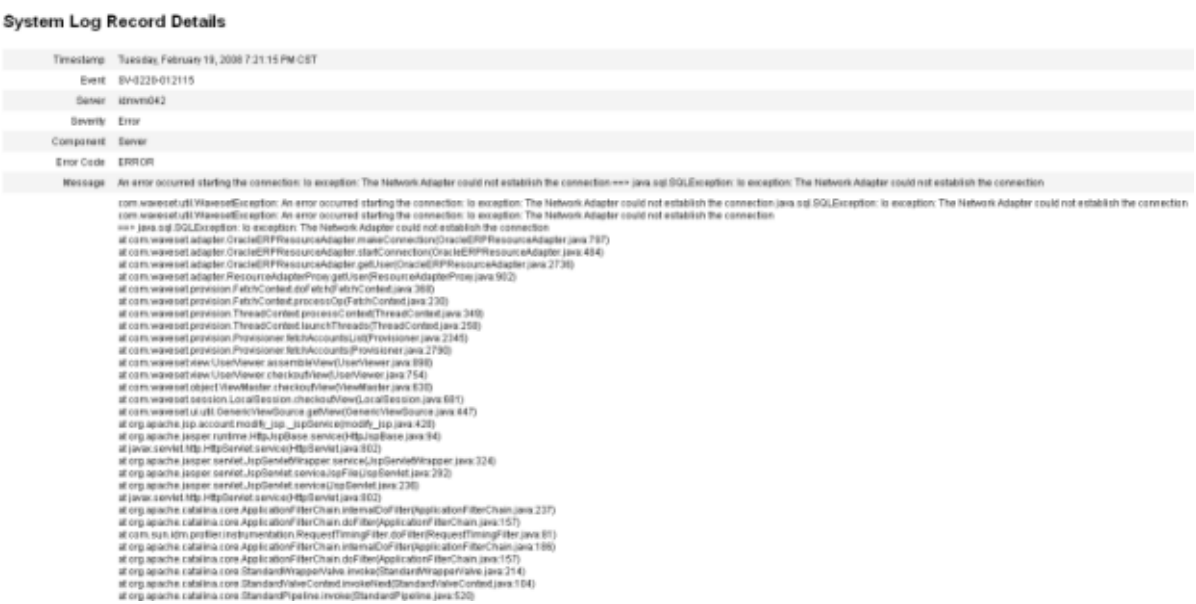
▼TimeStamp	Event	Severity	Server	Component	Error Code	Message
Thursday, February 21, 2008 3:09:06 PM CST	N/A	Fatal	idmvm042	Server	SECURITY	Security Violation: Incoming HttpServletRequest considered invalid by CSRFGUARD from address: 129.147.62.21
Tuesday, February 19, 2008 7:21:15 PM CST	SV-0220-012115	Error	idmvm042	Server	ERROR	An error occurred starting the connection: Io exception: The Network Adapter could not establish the connection ==> java.sql.SQLException: Io exception: The Network Adapter could not establish the connection

FIGURE 6–2 Example Report Results Page

The Report Results table shows the following information:

– **Timestamp**. Shows the day, date, and time the error occurred.

  Click the Timestamp links to view detailed information about that System Log record. For example, if you clicked the first Timestamp link shown in Figure 6–2, the following information displays.



- **Event**. Identifies the syslog ID of the target entry (when applicable).
- **Severity**. Shows the severity level of the error.

  Severity levels include

  - **Fatal**. A severe error that causes the system to crash, resulting in the loss or corruption of unsaved data.
  - **Error**. A severe error that might cause the loss or corruption of unsaved data. Immediate action must be taken to prevent losing data.
  - **Warning**. Action must be taken at some stage to prevent a severe error from occurring in the future.
  - **Info**. An informative message, usually describing server activity. No action is necessary.
- **Server**. Identifies the server on which the error occurred.
- **Component**. Identifies the system component that generated the error.
- **Error Code**. Shows the error code associated with that error.
- **Message**. Shows the actual error message text.

▼ To Run a System Log Report From the Command-Line Interface

Note – These instructions assume you are familiar with the Identity Manager command-line interface and `lh` commands. For more information, read Appendix A, "lh Reference," in *Sun Identity Manager 8.1 Business Administrator's Guide*.

Perform the following steps to run and view a System Log report from the command line:

1. **Open a command window.**
2. **Change directories to the default Identity Manager installation directory.**
3. **At the prompt, type the `lh syslog [`*options*`]` command.**

   Use these *options* to include or exclude information:

   - `-d`: *Number* – Show records for the previous number of days (Default is 1.)
   - `-F` – Show only records with fatal severity level
   - `-E` – Show only records with an error severity level or higher
   - `-i` *logid* – Show only records with a specified Syslog ID

     Syslog IDs are displayed on some error messages and reference a specific System Log entry.
   - `-W` – Show only records with a warning severity level or higher (default)
   - `-X` – Include reported cause of error, if available

Customizing a Default Error Message

Note – To add message catalog entries or to modify entries provided with the system, you must create a customized message catalog. See Chapter 9, "Customizing Message Catalogs," in *Sun Identity Manager Deployment Guide* for instructions.

Identity Manager's default error messages are stored in a `WPMessages.properties` file in the `idmcommon.jar` file and in an `RAMessages.properties file` in the `idmadapter.jar`.

- The `WPMessages.properties` file is located in *project_directory*`/waveset/idm/common/src/com/waveset/msgcat`
- The `RAMessages.properties file` is located in *project_directory*`/waveset/idm/adapter/src/com/waveset/adapter`

The Identity Manager Service Provider's default error messages are located in the `IDMXMessages.properties` file.

You can customize these default error messages by modifying attributes in the `ErrorUIConfig` object provided with that message.

▼ To Modify the `ErrorUIConfig` Object:

1. **Log in to Identity Manager Administrator interface.**
2. **Open the System Settings page by typing `http://` *host:port*`/idm/debug` in to your browser.**
3. **Locate the Type menu, located next to the List Objects button. Choose Configuration from the menu.**

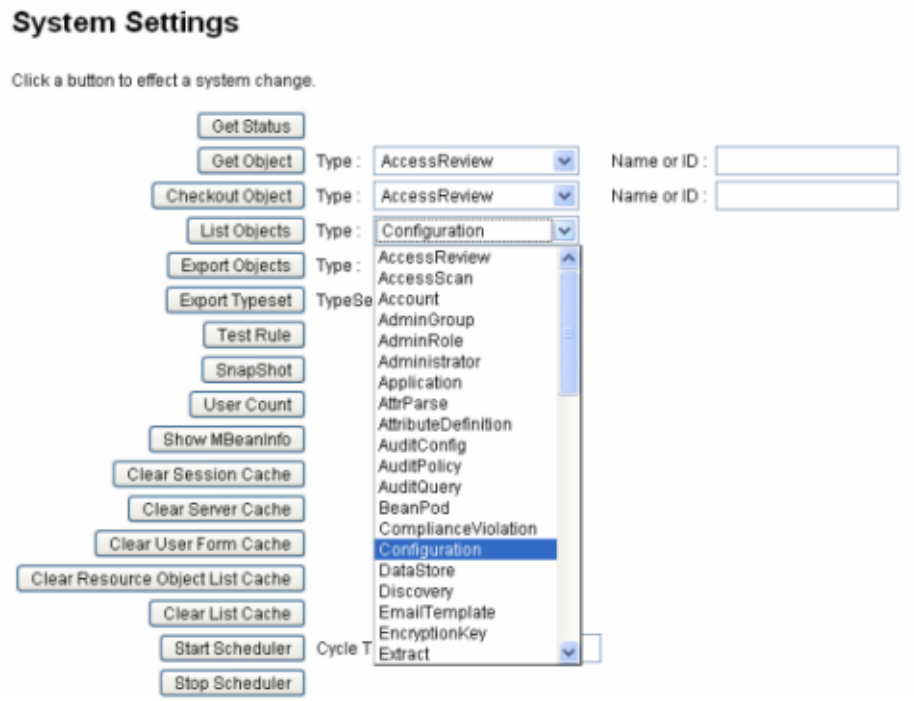


FIGURE 6–3 List Objects Type Menu

4. **Click the List Objects button.**

5 On the List Objects of type: Configuration page, click the `ErrorUIConfig` edit link.

The following example shows the XML for an `ErrorUIConfig` object on Checkout Object: Configuration page:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<!--  MemberObjectGroups="#ID#Top" extensionClass="GenericObject"
id="#ID#9787BA467E01441B:178655:1156195C008:-7FFE"
lastModifier="com.waveset.object.ErrorUIConfig" name="ErrorUIConfig"-->
<Configuration id='#ID#9787BA467E01441B:178655:1156195C008:-7FFE' name='ErrorUIConfig'
lock='Configurator#1200600790328' creator='com.waveset.object.ErrorUIConfig'
createDate='1191343145328' lastModifier='com.waveset.object.ErrorUIConfig'
lastModDate='1191343145328'>
  <Extension>
    <Object>
      <Attribute name='Enabled'>
        <Boolean>true</Boolean>
      </Attribute>
      <Attribute name='ErrorMsgID' value='UI_ERROR_DEFAULT_FATAL_MESSAGE'/>

    </Object>
  </Extension>
  <MemberObjectGroups>
    <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
  </MemberObjectGroups>
</Configuration>
```

6 You can modify the following `ErrorUIConfig` object attributes:

- **Enabled**. Controls whether the message is enabled (`true`) or disabled (`false`).

  Note – The ability to disable this attribute is provided for backward-compatibility; however, disabling this attribute will result in cryptic messages and you will not see the typical extended messages in the System Log.

- **ErrorMsgID**. Identifies the error message to be displayed.

  Change the `ErrorMsgID` attribute value to provide the message text you want displayed.

  The current setting for this attribute is as follows, and it references the `UI_ERROR_DEFAULT_FATAL_MESSAGE` message in the message catalog:

  ```
  <Attribute name='ErrorMsgID' value='UI_ERROR_DEFAULT_FATAL_MESSAGE'/>
  ```

Note – If you are using parameterized messages with single or double quotes in the message, you must use an additional single or double quote to escape the message. (You will see only one quote symbol when the message is output to the system.)

For example, the following message begins and ends with two *single* quote marks:

```
"'{0}"
```

7 When you are finished, click Save to save your changes.

Note – If you have difficulty creating a default error message, contact your Administrator or Administrator hotline.

Index

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

W

X