Solaris WBEM Services Administration Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 806–6827–06
December 2001

011025@2471

Contents

Tables

Figures

Preface

The *Solaris WBEM Services Administration Guide* explains Common Information Model (CIM) concepts and describes how to administer Web-Based Enterprise Management (WBEM) services in the Solaris™ operating environment.

Solaris WBEM Services software makes it easier for software developers to create management applications that run on Solaris and makes the Solaris operating environment easier to manage.

Who Should Use This Book

This book is written for system administrators who manage WBEM-enabled networks and workstations, by running existing WBEM applications or writing new ones.

Before You Read This Book

This book requires knowledge of these topics:

- Object-oriented programming concepts
- Java™ programming
- WBEM Common Information Model (CIM) concepts
- Network management concepts
- Simple Network Management Protocol (SNMP) concepts, if you intend to configure and use SNMP Adapter for WBEM

If you are unfamiliar with these areas, you might find the following references useful:

- *Java™ How to Program*

  H. M. Deitel and P. J. Deitel, Prentice Hall, ISBN 0–13–263401–5
- *The Java Class Libraries, Second Edition, Volume 1*, Patrick Chan, Rosanna Lee, Douglas Kramer, Addison-Wesley, ISBN 0–201–31002–3
- *CIM Tutorial*, provided by the Distributed Management Task Force

The following Web sites are useful resources when working with WBEM technologies.

- Distributed Management Task Force (DMTF)

  See this site at www.dmtf.org for the latest developments on CIM, information about various working groups, and contact information for extending the CIM Schema.
- Rational Software

  See this site at www.rational.com/uml for documentation on the Unified Modeling Language (UML) and the Rose CASE tool.

How This Book Is Organized

Chapter 1 provides an overview of Solaris WBEM Services and Web-Based Enterprise Management (WBEM).

Chapter 2 describes the CIM Object Manager. This chapter covers how to start and how to stop the CIM Object Manager and how to upgrade the CIM Object Manager Repository.

Chapter 3 describes the SNMP Adapter for WBEM, which enables existing SNMP applications to access WBEM data and translates SNMP messages into corresponding CIM properties and instances.

Chapter 4 describes WBEM security mechanisms, security features, and how to set access rights for namespaces and users.

Chapter 5 describes the command syntax for the `mofcomp` command and how to compile a `.mof` file.

Chapter 6 describes the logging features.

Appendix A describes the Solaris Schema files, Managed Object Format (MOF) files that describe managed objects in the Solaris operating environment.

Glossary is a list of words and phrases found in this book and their definitions.

Ordering Sun™ Documents

Fatbrain.com, the Internet's most comprehensive professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com℠ Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*.

TABLE P–1 Typographic Conventions *(Continued)*

Typeface or Symbol	Meaning	Example
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

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

TABLE P–2 Shell Prompts

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

CHAPTER 1

Overview

This chapter provides an overview of Web-Based Enterprise Management (WBEM) and Solaris WBEM Services, software that makes it easier for software developers to create management applications that run on Solaris and make the Solaris operating environment easier to manage.

This chapter covers the following topics:

About WBEM

WBEM is an industry-wide initiative that includes standards for web-based management of systems, networks, and devices on multiple platforms. This standardization enables system administrators to manage desktops, devices, and networks.

WBEM is designed to be compatible with all major existing management protocols, including Simple Network Management Protocol (SNMP), Distributed Management Interface (DMI), and Common Management Information Protocol (CMIP).

WBEM encompasses the following standards:

- Common Information Model (CIM) – Information model for describing managed resources.
- Managed Object Format (MOF) – Language for defining CIM classes and instances.

- eXtensible Markup Language (XML) – Markup language for describing managed resources on the web.

The Distributed Management Task Force (DMTF), a group that represents corporations in the computer and telecommunications industries, is leading the effort to develop management standards. The goal of the DMTF is to develop an integrated approach to managing networks across platforms and protocols, and consequently promote cost-effective products that interoperate as flawlessly as possible.

About the Common Information Model

This section provides a brief introduction to basic CIM terms and concepts as they are used in the Solaris WBEM Services product.

CIM is an object-oriented information model for describing managed resources such as disks, CPUs, and operating systems. A CIM object is a representation, or model, of a managed resource, such as a printer, disk drive, or CPU. CIM objects can be shared by any WBEM-enabled system, device, or application.

Basic CIM Elements

CIM objects with similar properties and purposes are represented as CIM classes. Properties are attributes that describe a unit of data for a class. An instance is a representation of a managed object that belongs to a particular class. Instances contain actual data. For example, `Solaris_ComputerSystem` is a CIM class that represents a computer that runs the Solaris operating environment. The Solaris software that runs on your workstation is an instance of the `Solaris_OperatingSystem` class. `ResetCapability` and `InstallDate` are examples of properties of the `Solaris_ComputerSystem` class.

CIM classes are grouped into meaningful collections called schemas. A schema is a group of classes with a single owner (an organization). A class must belong to only one schema. Schemas are used for administration and class naming. All class names must be unique within a particular schema. The schema name is the determining factor in differentiating classes and properties from others that may have the same name. The naming of schema, class, and property follow this syntax:

Schemaname_classname.propertyname

The CIM Models

The Common Information Model categorizes information from general to specific. Specific information, such as a representation of the Solaris environment, extends the model. CIM consists of the following three layers of information:

- Core Model – A subset of CIM not specific to any platform.
- Common Model – Information model that visually depicts concepts, functionality, and representations of entities related to specific areas of network management, such as systems, devices, and applications.
- Extensions – Information models that support the CIM Schema and represent a very specific platform, protocol, or corporate brand.

Collectively, the Core Model and the Common Model are called CIM Schema.

The Core Model

The Core Model provides the underlying, general assumptions of the managed environment—for example, that specific, requested data must be contained in a location and distributed to requesting applications or users. These assumptions are conveyed as a set of classes and associations that conceptually form the basis of the managed environment. The Core Model is meant to introduce uniformity across schemas intended to represent specific aspects of the managed environment.

For applications developers, the Core Model provides a set of classes, associations, and properties that can be used as a starting point to describe managed systems and determine how to extend the Common Model. The Core Model establishes a conceptual framework for modeling the rest of the managed environment.

The Core Model provides classes and associations to extend specific information about systems, applications, networks, devices, and other network features through the Common Model and extensions.

The Common Model

Areas of network management depicted in the Common Model are independent of a specific technology or implementation but provide the basis for the development of management applications. This model provides a set of base classes for extension into the area of five designated technology-specific schemas: Systems, Devices, Applications, Networks, and Physical.

CIM Extensions

Extension schemas are built upon CIM to connect specific technologies to the model. By extending CIM, a specific operating environment such as Solaris can be made available to a greater number of users and administrators. Extension schemas provide classes for software developers to build applications that manage and administer the extended technology. The Solaris Schema is an extension of the CIM Schema.

Solaris WBEM Services

Solaris WBEM Services software provides WBEM services in the Solaris operating environment. These services make it easier for software developers to create management applications that run in the Solaris operating environment, and makes the Solaris operating environment easier to manage.

Solaris WBEM Services software provides secure access and manipulation of management data. The product includes a built-in Solaris provider that enables management applications to access information about managed resources (devices and software) in the Solaris operating environment.

The CIM Object Manager accepts connections from management applications that use either the RMI or the XML/HTTP protocol, and provides the following services to connected clients:

- Management services, in the form of a CIM Object Manager that checks the semantics and syntax of CIM data and distributes data between applications, the CIM Repository, and managed resources.
- Security services that enable administrators to control user access to CIM information.

  Security services, which you specify for WBEM through the Solaris Management Console User tool, are described in *System Administration Guide: Security Services*.
- Logging services that consist of classes that developers can use to create applications that dynamically record event data in, and retrieve data from, a log record. Administrators use this data to track and determine the cause of events.
- XML services that convert XML data into CIM classes, enabling XML/HTTP-based WBEM clients to communicate with the CIM Object Manager.

Once connected to a WBEM-enabled system, WBEM clients can request WBEM operations, such as, creating, viewing, and deleting CIM classes and instances, querying for properties that have a specified value, enumerating (getting a list of) instances or classes in a specified class hierarchy.

Software Components

Solaris WBEM Services software consists of software components Application, Management, and Provider. These components interact with the operating system and hardware. Figure 1–1 shows the software components and how they interact.

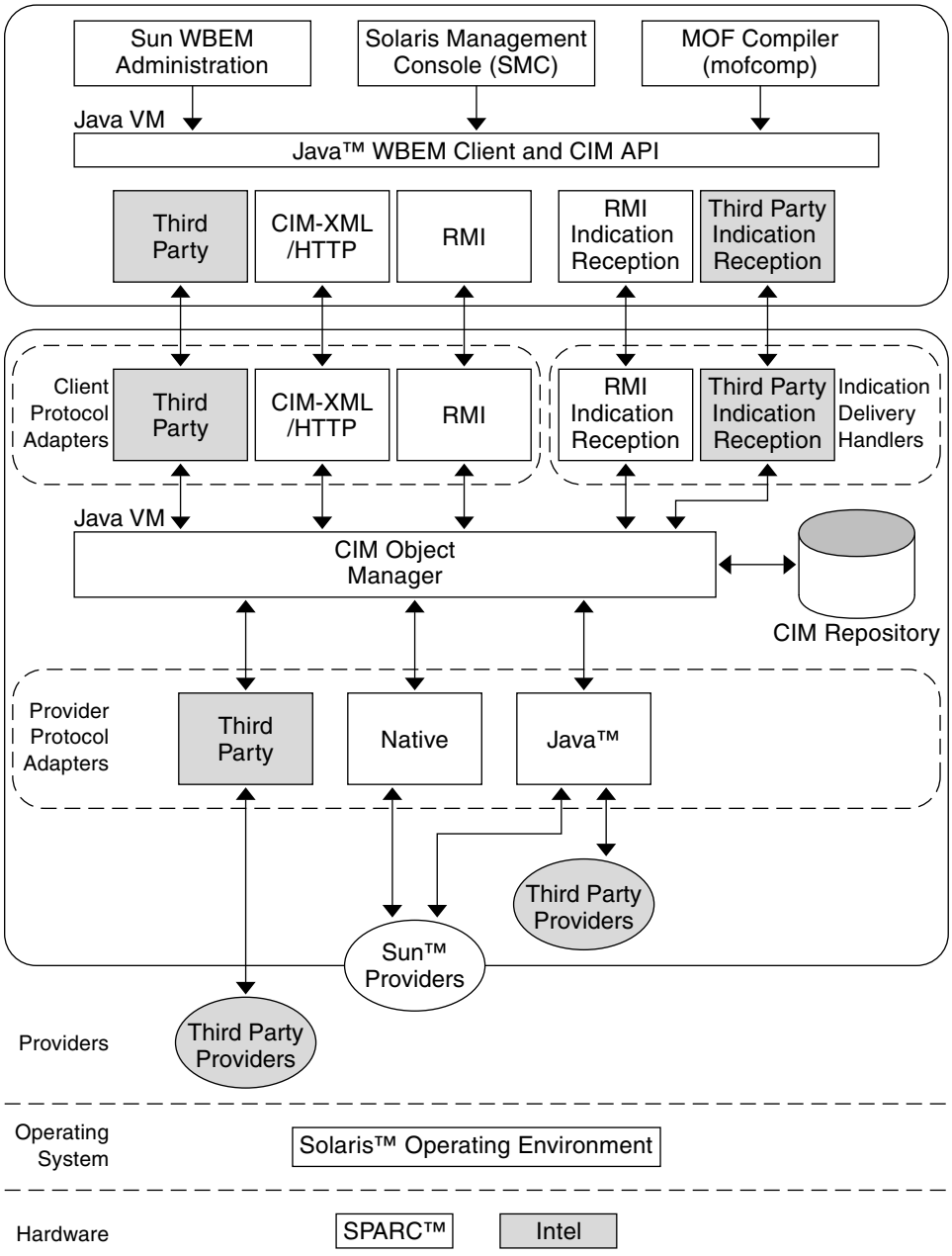

FIGURE 1–1 Solaris WBEM Services Architecture

- Application Layer – WBEM clients process and display data from managed resources. Solaris WBEM Services includes the following applications.

- Sun WBEM User Manager and Solaris Management Console™ Users Tool – Applications that allow system administrators to add and delete authorized users and to set their access privileges to managed resources.
- Solaris Management Console Log Viewer – An application that displays log files. A user can view details of a log record, including the name of the user who issued a logged command and the client computer on which a logged event occurred.
- Managed Object Format (MOF) Compiler – A program that parses a file containing MOF statements, converts the classes and instances defined in the file to Java classes, and then adds the Java classes to the CIM Object Manager Repository, a central storage area for management data.

  MOF is a language for defining CIM classes and instances. MOF files are ASCII text files that use the MOF language to describe CIM objects. A CIM object is a representation, or model, of a managed resource, such as a printer, disk drive, or CPU.

  Many sites store information about managed resources in MOF files. Because MOF can be converted to Java, applications that can run on any system with a Java virtual machine can interpret and exchange this information. You can also use the `mofcomp` command to compile MOF files at any time after installation. For more information about MOF, see the DMTF web page at http://www.dmtf.org.

- Management Layer – Components at this layer provide services to connected WBEM clients.
  - Common Information Model (CIM) Object Manager – Software that manages CIM objects on a WBEM system. CIM objects are stored internally as Java classes. The CIM Object Manager transfers information between WBEM clients, the CIM Object Manager Repository, and managed resources.
  - CIM Object Manager Repository – Central storage area for CIM class and instance definitions.
  - Client and CIM Application Programming Interfaces (APIs) – WBEM client applications use these Java interfaces to request operations, such as creating or viewing classes or instances of managed resources, from the CIM Object Manager.
  - Provider Interface – Providers use these interfaces to transfer information about managed resources to the CIM Object Manager. The CIM Object Manager uses the provider interfaces to transfer information to locally installed providers.
- Provider Layer – Providers act as intermediaries between the CIM Object Manager and one or more managed resources. When the CIM Object Manager receives a request from a WBEM client for data that is not available from the CIM Object Manager Repository, it forwards the request to the appropriate provider.
  - Solaris Provider – Provides the CIM Object Manager with instances of managed resources in the Solaris operating environment. Providers get and set information on managed devices. A native provider is a machine-specific

program written to run on a managed device. For example, a provider that accesses data on a Solaris system will probably include C functions to query the Solaris system. The Java Native Interface (JNI) is the native programming interface for Java that is part of the JDK™. By writing programs using the JNI, you ensure that your code is completely portable across all platforms. The JNI enables Java code that runs within a Java virtual machine to operate with applications and libraries written in other languages, such as C, C++, and assembly.

- Solaris Schema – A collection of classes that describe managed objects in the Solaris operating environment. The CIM and Solaris Schema classes are stored in the CIM Object Manager Repository. The CIM Schema is a collection of class definitions used to represent managed objects that occur in every management environment.

  The Solaris Schema is a collection of class definitions that extend the CIM Schema and represent managed objects in a typical Solaris operating environment. Users can also use the MOF compiler (`mofcomp`) to add CIM Schema, Solaris Schema, or other classes to the CIM Object Manager Repository.

- Operating System Layer – The Solaris provider enables management applications to access information about managed resources (devices and software) in the Solaris operating environment.
- Hardware Layer – A management client can access management data on any supported Solaris platform.

Namespaces

One or more schemas can be stored in directory-like structures called namespaces. A CIM namespace is a directory-like structure that can contain other namespaces, classes, instances, and qualifier types. The names of objects within a namespace must be unique.

In Solaris WBEM Services, when a WBEM client application connects to a particular namespace, all subsequent operations occur within that namespace. When connected to a namespace, the client can access the classes and instances in that namespace (if they exist) and in any namespaces contained in that namespace. For example, if you create a namespace called `child` in the `root\cimv2` namespace, you could connect to `root\cimv2` and access the classes and instances in the `root\cimv2` and `root\cimv2\child` namespaces.

An application can connect to a namespace within a namespace. This is similar to changing to a subdirectory within a directory. Once the application connects to the new namespace, all subsequent operations occur within that namespace. If you open a new connection to `root\cimv2\child`, you can access any classes and instances in that namespace but cannot access the classes and instances in the parent namespace, `root\cimv2`.

The following namespaces are created by default during installation.

- `root` – The top-level namespace that contains other namespaces.
- `root\cimv2` – Contains the default CIM classes and instances that represent objects on your system, such as, `LogicalDisk` and `Netcard`. This is the default namespace.
- `root\security` – Contains the security classes used by the CIM Object Manager to represent access rights for users and namespaces.
- `root\snmp` – Contains the classes for the SNMP Adapter for WBEM.
- `root\system` – Contains CIM Object Manager information and provider paths.

Providers

When a WBEM client application accesses CIM data, the WBEM system validates the user's login information on the current host. By default, a user is granted read access to the CIM and Solaris Schema. The CIM Schema describes managed objects on your system in a standard format that all WBEM-enabled systems and applications can interpret.

Providers are classes that communicate with managed objects to access data. Providers forward this information to the CIM Object Manager for integration and interpretation. When the CIM Object Manager receives a request from a management application for data that is not available from the CIM Object Manager Repository, it forwards the request to a provider.

The CIM Object Manager uses object provider APIs to communicate with providers. When an application requests dynamic data from the CIM Object Manager, the CIM Object Manager uses the provider interfaces to pass the request to the provider.

Providers perform the following functions in response to a request from the CIM Object Manager:

- Map the native information format to CIM classes
  - Get information from a device
  - Pass the information to the CIM Object Manager in the form of CIM classes
- Map the information from CIM classes to native device format
  - Get the required information from the CIM class
  - Pass the information to the device in native device format

Interoperability with Other WBEM Systems

A WBEM client and WBEM system can run on the same system or on different systems. Multiple WBEM clients can establish connections to the same WBEM system. A typical WBEM system can serve four or five WBEM clients.

Solaris WBEM Services supports the Version 1.0 Specification for CIM Operations over HTTP. This specification uses XML to model CIM objects and messages. XML is a standard markup language for describing data on the Web. This standard extends XML markup to define CIM objects and operations. Because XML provides a standard way of describing data that can be sent across the Web, any WBEM client can access CIM data on any WBEM system that can parse XML data.

Sun™ WBEM Software Development Kit

The Sun WBEM Software Development Kit (SDK) contains the components required to write management applications that can communicate with any WBEM-enabled management device. Developers can also use this tool kit to write providers, programs that communicate with managed objects to access data. All management applications developed using the Java WBEM SDK run on the Java platform.

A WBEM client application is a program that uses Java WBEM SDK APIs to manipulate CIM objects. A client application typically uses the CIM API to construct an object (for example, a namespace, class, or instance) and then initialize that object. The application then uses the Client APIs to pass the object to the CIM Object Manager and request a WBEM operation, such as creating a CIM namespace, class, or instance.

The Java WBEM SDK installs and runs in the Java environment. It may be used as a standalone application or with Solaris WBEM Services. You can download the Sun WBEM SDK from http://www.sun.com/solaris/wbem.

CHAPTER 2

CIM Object Manager

The Common Information Model (CIM) Object Manager is software that transfers CIM data between WBEM client applications and managed resources.

This chapter covers the following topics:

About the CIM Object Manager

The CIM Object Manager manages CIM objects on a WBEM-enabled system. A CIM object is a representation, or model, of a managed resource, such as a printer, disk drive, or CPU. CIM objects are stored internally as Java classes.

When a WBEM client application accesses information about a CIM object, the CIM Object Manager contacts either the appropriate provider for that object or the CIM Object Manager Repository. Providers are classes that communicate with managed objects to access data. When a WBEM client application requests data from a managed resource that is not available from the CIM Object Manager Repository, the CIM Object Manager forwards the request to the provider for that managed resource. The provider dynamically retrieves the information.

At startup, the CIM Object Manager performs the following functions:

- Listens for RMI connections on RMI port 5987 and for XML/HTTP connections on HTTP port 5988
- Sets up a connection to the CIM Object Manager Repository

- Waits for incoming requests

During normal operations, the CIM Object Manager performs these functions:

- Performs security checks to authenticate user login and authorization to access namespaces
- Performs syntactical and semantic checking of CIM data operations to ensure that they comply with the latest CIM Specification
- Routes requests to the appropriate provider or to the CIM Object Manager Repository
- Delivers data from providers and from the CIM Object Manager Repository to WBEM client applications

A WBEM client application contacts the CIM Object Manager to establish a connection when it needs to perform WBEM operations, such as creating a CIM class or updating a CIM instance. When a WBEM client application connects to the CIM Object Manager, the WBEM client gets a reference to the CIM Object Manager, which it then uses to request services and operations.

The `init.wbem` Command

Solaris automatically runs `init.wbem`(1M) during installation and every time you reboot a system. The `init.wbem` command starts the CIM Object Manager and Solaris Management Console server, both of which run combined in a single process. You can also use `init.wbem` to stop the CIM Object Manager, to stop the Solaris Management Console server, or to retrieve status from a server.

Generally, you do not need to stop the CIM Object Manager. However, if you change an existing provider, you must stop and restart the CIM Object Manager before using the updated provider.

You can specify three options with `init.wbem`:

- `start` – Starts the CIM Object Manager and Solaris Management Console server on the local host
- `stop` – Stops the CIM Object Manager and Solaris Management Console server on the local host
- `status` – Gets status for the CIM Object Manager and Solaris Management Console server on the local host

Solaris Management Console Server

The Solaris Management Console software provides Solaris management applications such as User Manager, Disk Manager, and Log Viewer. The Solaris Management Console server provides tools for the console to download and performs common services for the console and its tools, such as authentication, authorization, logging, messaging, and persistence.

System Booting

The `init.wbem` command is located in the `/etc/init.d` directory. The file `/etc/rc2.d/S90wbem` runs with the `start` option when initialization state 2 is entered (normally at boot time). The files `/etc/rc0.d/K36wbem`, `/etc/rc1.d/K36wbem`, and `/etc/rcS.d/K36wbem` runs with the `stop` option when initialization states 0, 1, and S are entered (normally when the system halts, or when the system enters either system administrator mode or single-user mode).

Stopping and Restarting the CIM Object Manager

If you change a provider, you must stop and restart the CIM Object Manager before using the updated provider.

▼ To Stop the CIM Object Manager

1. **Become superuser.**
2. **Stop the CIM Object Manager:**

```
# /etc/init.d/init.wbem stop
```

▼ To Restart the CIM Object Manager

1. **Become superuser.**
2. **Restart the CIM Object Manager:**

```
# /etc/init.d/init.wbem start
```

Upgrading the CIM Object Manager Repository

You must update any proprietary custom Managed Object Format (MOF) data to the new Reliable Log repository format that is used with WBEM Services 2.5 in Solaris 9.

Before you upgrade to the Solaris 9 operating environment, you might need to save the JavaSpaces™ datastore. After you upgrade, you must convert or merge data, depending on the version of the Solaris operating environment that you were running on your system before you upgraded to the Solaris 9 operating environment.

Failure to convert or merge the data results in data loss.

Use the following table to determine whether or not to save the JavaSpaces software before you upgrade and whether to convert or merge the WBEM data after you upgrade to the Solaris 9 operating environment.

TABLE 2–1 Convert or Merge WBEM Data

Operating Environment Before Upgrading to Solaris 9	Save JavaSpaces Datastore Before You Upgrade?	Convert or merge?
Solaris 8 (Solaris WBEM Services 2.0) Solaris 8 6/00 (WBEM Services 2.2) Solaris 8 10/00 (WBEM Services 2.2)	Yes	Convert
Solaris 8 1/01 (WBEM Services 2.3) Solaris 8 4/01 (WBEM Services 2.4) Solaris 8 7/01 (WBEM Services 2.4) Solaris 8 10/01 (WBEM Services 2.4) Solaris 9 (Beta) (WBEM Services 2.5)	No	Merge

▼ To Save the JavaSpaces Datastore

1. **Become superuser.**

2. **Do you want to download the files that you will need, or do you want to save your current JavaSpaces datastore?**

Note – The safer method is to save your JavaSpaces datastore rather than to download files.

- If you want to download the files, go to the next step.
- If you want to save your JavaSpaces datastore:

```
# cp /usr/sadm/lib/wbem/outrigger.jar /usr/sadm/lib/wbem/outrigger.jar.tmp
# cp /usr/sadm/lib/wbem/outrigger-dl.jar /usr/sadm/lib/wbem/outrigger-dl.jar.tmp
# cp /usr/sadm/lib/wbem/transient-outrigger.jar \
/usr/sadm/lib/wbem/transient-outrigger.jar.tmp
# cp /usr/sadm/lib/wbem/jini-core.jar /usr/sadm/lib/wbem/jini-core.jar.tmp
# cp /usr/sadm/lib/wbem/jini-ext.jar /usr/sadm/lib/wbem/jini-ext.jar.tmp
# cp /usr/sadm/lib/wbem/tools.jar /usr/sadm/lib/wbem/tools.jar.tmp
# cp /usr/sadm/lib/wbem/pro.zip /usr/sadm/lib/wbem/pro.zip.tmp
```

3. **Determine and record the version of the JDK™ that is currently installed on your system:**

```
# /usr/bin/java -version
java version "1.2.1"
Solaris VM (build Solaris_JDK_1.2.1_04c, native threads, sunwjit)
```

Note – You must be running the same version of the JDK as you used when you created the original JavaSpaces datastore to convert WBEM data.

▼ To Convert WBEM Data

Follow these steps to convert WBEM data.

1. **Upgrade your system to the Solaris 9 operating environment.**
2. **Become superuser.**
3. **Stop the CIM Object Manager:**

```
# /etc/init.d/init.wbem stop
```

Caution – Failure to stop the CIM Object Manager before running `wbemconfig convert` might corrupt your data.

4. **Did you save your current JavaSpaces datastore in "To Save the JavaSpaces Datastore" on page 28?**

   - If yes, restore your JavaSpaces datastore:

```
# cp /usr/sadm/lib/wbem/outrigger.jar.tmp /usr/sadm/lib/wbem/outrigger.jar
# cp /usr/sadm/lib/wbem/outrigger-dl.jar.tmp /usr/sadm/lib/wbem/outrigger-dl.jar
# cp /usr/sadm/lib/wbem/transient-outrigger.jar.tmp \
/usr/sadm/lib/wbem/transient-outrigger.jar
# cp /usr/sadm/lib/wbem/jini-core.jar.tmp /usr/sadm/lib/wbem/jini-core.jar
# cp /usr/sadm/lib/wbem/jini-ext.jar.tmp /usr/sadm/lib/wbem/jini-ext.jar
# cp /usr/sadm/lib/wbem/tools.jar.tmp /usr/sadm/lib/wbem/tools.jar
# cp /usr/sadm/lib/wbem/pro.zip.tmp /usr/sadm/lib/wbem/pro.zip
```

   - If no, download and unzip the file `UpgradeRepository.zip` from http://www.sun.com/solaris/wbem/.

     `UpgradeRepository.zip` contains the `.jar` files that you need to later convert the WBEM data.

5. **In a directory other than the one in which the JDK you are currently using is installed, obtain and install the JDK that you recorded in "To Save the JavaSpaces Datastore" on page 28.**

6. **Change the symbolic link from the currently installed JDK in `/usr/java` to the JDK you recorded in "To Save the JavaSpaces Datastore" on page 28. For example, to change the currently installed JDK to `Solaris_JDK_1.2.1_04c` in `/old_sdk`, type:**

```
# rm /usr/java
# ln -s /old_sdk/Solaris_JDK_1.2.1_04c /usr/java
```

7. **Convert the data in the JavaSpaces datastore to Reliable Log format:**

```
# /usr/sadm/lib/wbem/wbemconfig convert
```

   The `wbemconfig convert` command successfully converts any proprietary custom MOF data, but not any CIM or Solaris MOF data that you have modified. CIM and Solaris MOF data that you have modified is destroyed.

Note – To recompile any modified CIM or Solaris MOF data in the new repository, use the `mofcomp` command to compile the MOF files that contain the class definitions.

8. **Change the symbolic link from `/usr/java` to the location of the JDK software that ships with the Solaris 9 operating environment. For example, to change the symbolic link from `/usr/java1.2`, type:**

```
# rm /usr/java
# ln -s /usr/java1.2 /usr/java
```

9. **Stop the CIM Object Manager:**

```
# /etc/init.d/init.wbem stop
```

10. **Start the CIM Object Manager:**

```
# /etc/init.d/init.wbem start
```

The CIM Object Manager adds repository files that contain the converted data to the directory `/var/sadm/wbem/logr/`, which the Solaris installer created when you upgraded your system to Solaris 9.

▼ To Merge WBEM Data

Follow these steps to merge WBEM data.

1. **Upgrade your system to the Solaris 9 operating environment.**
2. **Become superuser.**
3. **Stop the CIM Object Manager:**

```
# /etc/init.d/init.wbem stop
```

Caution – Failure to stop the CIM Object Manager before you run `wbemconfig convert` might corrupt your data.

4. **Merge the original data in the previous Reliable Log with the data in the Solaris 9 Reliable Log:**

```
# /usr/sadm/lib/wbem/wbemconfig convert
```

Note – The `wbemconfig convert` command successfully converts any proprietary custom MOF data, but not any CIM or Solaris MOF data that you have modified. CIM and Solaris MOF data that you have modified is destroyed. To recompile any modified CIM or Solaris MOF data in the new repository, use the `mofcomp` command to compile the MOF files that contain the class definitions.

Exception Messages

The CIM Object Manager generates exception messages to indicate incorrect MOF syntax and semantics. The *Solaris WBEM SDK Developer's Guide* contains information about exception messages.

CHAPTER 3

SNMP Adapter for WBEM

Intended for use by system administrators, the SNMP Adapter for WBEM enables Simple Network Management Protocol (SNMP) management applications to access system management information that is provided by Solaris WBEM Services.

Used with the Solstice Enterprise Agent (SEA) Master Agent `snmpdx`(1M), the SNMP Adapter for WBEM maps SNMP requests into equivalent WBEM Common Information Model (CIM) properties or instances.

The SNMP Adapter for WBEM also remaps the response from the CIM Object Manager into an SNMP response, which is returned to the management application.

A mapping file contains the corresponding Object Identifier (OID), class name, property name, and Abstract Syntax Notation One (ASN.1) type for each object. You can create your own mapping files.

This chapter covers the following topics:

How the SNMP Adapter for WBEM Works

The Solaris operating environment initializes WBEM Services before starting the Solstice Enterprise Agents Master Agent. By default, the SNMP Adapter for WBEM,

snmpXwbemd(1M), is disabled. However, once you enable it, the Solstice Enterprise Agents Master Agent (snmpdx), starts the SNMP Adapter for WBEM automatically.

An SNMP Manager passes an SNMP Get-request to the Solstice Enterprise Agents Master Agent. The Solstice Enterprise Agents Master Agent then sends the Get-request to the SNMP Adapter for WBEM, which uses the mapping files in /var/sadm/wbem/snmp/map to translate the objects in the Get-request into corresponding CIM objects. The SNMP Adapter for WBEM also translates the CIM objects into SNMP objects in a Get-response.

Note – At present, only Get-request and scalar objects are supported in Solaris 9. Get-next-request, Get-bulk-request, and Set-request as well as other objects are not currently supported.

The SNMP Adapter for WBEM searches this directory alphabetically for the first file to which the extension .map is appended. The adapter then reads all mapping files in the directory and caches their contents. The adapter uses the contents of these files to translate the objects that are specified in the Get-request into corresponding CIM objects. The SNMP Adapter for WBEM subsequently ignores duplicate OIDs in the mapping files in the directory. For example, if this OID appears in 002SUNWlvma.map:

```
1.3.6.1.2.1.1.1.0 My_ComputerSystem Description SnmpString
```

and the same OID appears in 050SUNWwbcou.map, which the SNMP Adapter for WBEM reads after 002SUNWlvma.map:

```
1.3.6.1.2.1.1.1.0 Solaris_ComputerSystem Description SnmpString
```

then, the adapter ignores the OID that is specified in 050SUNWwbcou.map.

The SNMP Adapter for WBEM subsequently generates a Get-response for each Get-request that an SNMP Manager submits. If the SNMP Adapter for WBEM cannot find a corresponding entry in any mapping file, the SNMP Adapter for WBEM returns a Get-response error.

How the Master Agent Routes a Request: SNMP Adapter for WBEM Compared to the Sun SNMP Agent

Until the release of the SNMP Adapter for WBEM, when an SNMP Manager sent a Get-request for an SNMP MIB-2 variable to the Solstice Enterprise Agents Master Agent, the Master Agent routed the request to the Sun SNMP MIB-2 Agent (mibiisa(1M)). Because the SNMP Adapter for WBEM also handles SNMP MIB-2

requests, however, what happens if the Sun SNMP Agent and the SNMP Adapter for WBEM are both running at the same time? How does the Master Agent route a request?

The Master Agent builds a node table based on the subtrees that are defined in each subagent registration file. The `mibiisa` subagent registers the entire MIB-2 subtree and the Sun Microsystems MIB subtree. The SNMP Adapter for WBEM registers the `MIB-2.system` subtree and the `hostRsrc` subtree. The Master Agent does not allow two agents to register the same subtree.

The Master Agent is described in the *Solstice Enterprise Agents 1.0 User Guide*.

At initialization, the Master Agent creates a node table that contains each subtree that is registered. The Master Agent forwards each Get-request to the agent whose subtree best matches the OID that is included in the request. A request for `mib-2.system.5.0`, for example, is forwarded to the SNMP Adapter for WBEM. A request for `mib-2.interfaces.1.0`, on the other hand, is forwarded to the `mibiisa` subagent. If the OID is not defined within any subtree that is registered by the Master Agent, the Master Agent returns an error in the Get-response.

The SNMP Adapter for WBEM supports SNMPv1 requests only.

Configuring the Adapter and Mapping SNMP to CIM Objects

Configuration Files

The files you use to configure the SNMP Adapter for WBEM, which are located in `/etc/snmp/conf/`, are described in the following sections.

In `snmpXwbem.acl`, you define the Access Control Language (ACL) policies that are associated with the SNMP Adapter for WBEM, in this format:

```
#pragma ident  "@(#)snmpXwbem.acl   1.2   01/04/18 SMI"
#Copyright (c) 2001 by Sun Microsystems, Inc.
#All rights reserved.

#        Configuration file of the SNMP subagent for WBEM

###################
# access control #
###################
```

```
# The list of community names needed for read/write access
# to the entire MIB.

# If the list is empty, the only valid community name is "public"
# and its access type is read-only
#
# A * in the managers list indicates requests can be received from
# any host.

acl = {
        {
                communities = public, private
                access = read-only
                managers = *
        }
}

####################
# trap parameters #
####################
trap = {
}
```

A comma-separated list of communities and a comma-separated list of managers are allowed. The access policies are read-only. An empty `trap` clause is required. Traps are not supported by the SNMP Adapter for WBEM in Solaris 9.

In `snmpXwbem.reg`, you define the Object Identifier (OID) of the subtree for which the SNMP Adapter for WBEM is responsible, in this format:

```
#pragma ident   "@(#)snmpXwbem.reg      1.3     01/10/04 SMI"
#
#Copyright (c) 2001 by Sun Microsystems, Inc.
#All rights reserved.

#       Configuration file of the SNMP subagent for WBEM

##########
# macros #
##########

# The following 3 macros are predefined:
#
#       mib-2 =         1.3.6.1.2.1
#       enterprise =    1.3.6.1.4.1
#       sun =           1.3.6.1.4.1.42
#
# Your can define your own macros, so that you can
# manipulate strings instead of OIDs in defining the agent.
# See the "agent" section below.

macros = {
        system = mib-2.1
```

```
        hostRsrc = mib-2.25
}


##########
# agent #
##########

# You must fill in at least the following fields:
#
# - name:               the name of your agent (for example, the executable
#                       file name of your agent)
#
# - subtrees:           the list of OIDs / subtrees of OIDs your agent
#                       supports. The listed items must be separated by
#                       a comma.
#
# You can also change or add the following fields:
#
# - timeout:            the number of micro-seconds the SNMP Relay will
#                       wait for a response from your agent
#
# - watch-dog-time:     the number of seconds the SNMP Relay will wait to
#                       test whether the subagent is active, if there has
#                       been no activity for the watch-dog-time interval
#
# - port:               the UDP port number on which you will start
#                       your agent

agents =
{
        {
                 name = "WBEMsubagent"
                 subtrees = { system, hostRsrc }
                 timeout = 20000000
                 watch-dog-time = 240
        }
}
```

The unit of measure for `timeout` is microseconds. The unit of measure for `watch-dog-time` is seconds. By default, the Master Agent tries to start the SNMP Adapter for WBEM every four minutes (or number of seconds to which `watch-dog-time` is set).

Note – The Master Agent automatically determines the port to be used by the SNMP Adapter for WBEM.

In `snmpXwbem.rsrc-`, you define a pointer to the registration file and you define how the SNMP Master Agent is to start the SNMP Adapter for WBEM, in this format:

```
#pragma ident   "@(#)snmpXwbem.rsrc-   1.2   01/04/18 SMI"
#Copyright (c) 2001 by Sun Microsystems, Inc.
#All rights reserved.

#        Configuration file of the SNMP subagent for WBEM

##########
# agents #
##########
resource =
{
         {
                  registration_file = "/etc/snmp/conf/snmpXwbem.reg"
                  security = "/etc/snmp/conf/snmpXwbem.acl"
                  policy = "spawn"
                  type = "legacy"
                  command = "/usr/sadm/lib/wbem/snmpXwbemd -p $PORT"
         }
}
```

Note – The Master Agent automatically determines the port to be used by the SNMP Adapter for WBEM.

Mapping Files

When the Master Agent passes a Get-request to the SNMP Adapter for WBEM, the adapter uses the mapping files in `/var/sadm/wbem/snmp/map` to translate the Get-request into a CIM object request. The Solaris operating environment includes a mapping file for you in this directory, but you can also define your own mapping file for the CIM instrumentation you want to view through an SNMP Manager.

This section describes what you need to know to create an SNMP Adapter for WBEM mapping file.

Contents of the Mapping File That Is Included in Solaris

This example shows the contents of the mapping file that the Solaris operating environment includes for you:

```
#
#pragma ident    "@(#)050SUNWwbcou.map    1.0    01/04/03 SMI"
#
# Copyright (c) 2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# *** Description of contents ***
```

```
#
# First non-commented non-blank line contains required Version label.
# Remaining non-commented non-blank lines are considered map entries
# used as described below:
#
# Column 1 - SNMP OID - Uniquely describes an SNMP variable
# Column 2 - CIM Class Name - CIM class associated with this variable
# Column 3 - CIM Property Name - CIM property that maps to SNMP OID variable
# Column 4 - ASN.1 type - SNMP datatype that dictates how data is mapped
#            to/from SNMP requests.  Supported types are: SnmpString, SnmpOid,
#            SnmpTimeticks, SnmpCounter, SnmpInt, SnmpGauge, SnmpIpAddress,
#            SnmpOpaque)
# Column 5 and greater are ignored
#
Version 1.0

1.3.6.1.2.1.1.1.0 Solaris_ComputerSystem Description SnmpString
1.3.6.1.2.1.1.3.0 Solaris_OperatingSystem LastBootUpTime SnmpTimeticks
1.3.6.1.2.1.1.4.0 Solaris_ComputerSystem PrimaryOwnerContact SnmpString
1.3.6.1.2.1.1.5.0 Solaris_ComputerSystem Name SnmpString

1.3.6.1.2.1.25.1.5.0 Solaris_OperatingSystem NumberOfUsers SnmpGauge
1.3.6.1.2.1.25.1.6.0 Solaris_OperatingSystem NumberOfProcesses SnmpGauge
1.3.6.1.2.1.25.1.7.0 Solaris_OperatingSystem MaxNumberOfProcesses SnmpGauge
1.3.6.1.2.1.25.1.2.0 Solaris_OperatingSystem LocalDateTime SnmpString
```

The contents of this mapping file associate the SNMP MIB-2 System Group scalar objects with their corresponding CIM objects:

MIB-2 System Group Scalar Object	CIM Object
sysDescr	Solaris_ComputerSystem.Description
sysUpTime	Solaris_OperatingSystem.LastBootUpTime
sysContact	Solaris_ComputerSystem.PrimaryOwnerContact
sysName	Solaris_ComputerSystem.Name

The contents of this mapping file also associate the SNMP Host Resources MIB objects with their corresponding CIM objects:

SNMP Host Resources MIB Object	CIM Object
hrSystemNumUsers	Solaris_OperatingSystem.NumberOfUsers
hrSystemProcesses	Solaris_OperatingSystem.NumberOfProcesses
hrSystemMaxProcesses	Solaris_OperatingSystem.MaxNumberOfProcesses

SNMP Host Resources MIB Object	CIM Object
hrSystemDate	Solaris_OperatingSystem.LocalDateTime

The syntax of the contents of a mapping file is described in "Syntax of the Contents of a Mapping File" on page 40.

Note – At present, the only way to retrieve host resource data is through the CIM Object Manager, as Solaris does not currently provide an SNMP Host Resource agent.

Syntax of a Mapping File Name

To ensure that the SNMP Adapter for WBEM reads your mapping file, name the file according to this syntax:

alphanumeric-string `.map`

alphanumeric-string represents an alphanumeric string. For example, here is the name of the mapping file that Solaris includes: `050SUNWwbcou.map`.

You include the three digits to ensure that the SNMP Adapter for WBEM reads the files in a more precise order (`002SUNWlvma.map` is read before `050SUNWwbcou.map`, for example).

Note – You must allow root to at least read the mapping files that you create.

```
% chmod 400 002SUNWlvma.map
```

Syntax of the Contents of a Mapping File

The following table describes the elements and the syntax of the contents of a mapping file.

TABLE 3–1 Contents of a Mapping File

Element	Description	Required?
#	A comment, which is always ignored.	No

TABLE 3–1 Contents of a Mapping File *(Continued)*

Element	Description	Required?
`Version 1.0`	The version of the mapping file. The text string that specifies the version must be the first uncommented line. If you do not specify a version as shown, the mapping file is ignored.	Yes
`1.3.6.1.2.1.1.1.0`	The SNMP Object Identifier, or OID, which is the key you want to extract from the SNMP request. The SNMP OID describes an SNMP variable. Because the SNMP Adapter for WBEM currently supports scalars only, the OID must end with the text string `.0`.	Yes
`Solaris_ComputerSystem`	The CIM class name that is associated with the variable.	Yes
`Description`	The CIM property name that defines a characteristic of the specified class and that maps to the SNMP OID variable.	Yes

TABLE 3–1 Contents of a Mapping File *(Continued)*

Element	Description	Required?
`SnmpString`	The ASN.1 data type. Values you can specify (including how they are mapped), are: ■ `SnmpString` – move string, number, or CIM `LocalDateTime` into `SnmpString` ■ `SnmpInt` – move CIM number data types (including a string in number format) into `SnmpInt` (signed, 32-bit integer) ■ `SnmpCounter` – move CIM number data types (including string in number format) into `SnmpCounter` (unsigned, 32-bit integer) ■ `SnmpGauge` – move CIM number data types (including string in number format) into `SnmpGauge` (unsigned, 32-bit integer) ■ `SnmpTimeticks` – move the time difference, represented in hundredths of a second, into `SnmpTimeticks` (this value is calculated by subtracting the CIM value from the current time: `sysUpTime`, for example, is calculated by subtracting `bootTime` from `currentTime`) ■ `SnmpIpAddress` – move string into `SnmpIpAddress` (you must specify the string in IP address format) ■ `SnmpOid` – move string into `SnmpOid` (you must specify the string in OID format) ■ `SnmpOpaque` – move vector of bytes into `SnmpOpaque`	Yes

Installing and Using the SNMP Adapter for WBEM

This section describes how to install, start, stop, and use the SNMP Adapter for WBEM.

▼ To Install the SNMP Adapter for WBEM

- **Install the Solaris operating environment on your system.**

  The SNMP Adapter for WBEM software is installed on your system along with the Solaris software.

▼ To Start the SNMP Adapter for WBEM

When you're ready to retrieve data from the CIM Object Manager through your SNMP application, follow these steps to start the SNMP Adapter for WBEM.

1. **Become superuser.**
2. **Stop the Master Agent.**

   ```
   # /etc/init.d/init.snmpdx stop
   ```

3. **Change directory to** `/etc/snmp/conf`**.**
4. **Move** `snmpXwbem.rsrc-` **into the SNMP Adapter for WBEM resource file.**

   ```
   # mv snmpXwbem.rsrc- snmpXwbem.rsrc
   ```

5. **Restart the Master Agent.**

   ```
   # /etc/init.d/init.snmpdx start
   ```

▼ To Disable the SNMP Adapter for WBEM

You can disable the SNMP Adapter for WBEM to modify a file in `/etc/snmp/conf` or when you are finished retrieving data from the CIM Object Manager. Follow these steps to stop the SNMP Adapter for WBEM.

1. **Become superuser.**
2. **Stop the Master Agent.**

   ```
   # /etc/init.d/init.snmpdx stop
   ```

3. **Stop the SNMP Adapter for WBEM.**

   ```
   # /usr/bin/pkill -9 -x -u 0 snmpXwbemd
   ```

4. **Change directory to** `/etc/snmp/conf`**.**
5. **Temporarily rename** `snmpXwbem.rsrc`**.**

   ```
   # mv snmpXwbem.rsrc snmpXwbem.rsrc-
   ```

6. **Restart the Master Agent.**

```
# /etc/init.d/init.snmpdx start
```

▼ To Force the SNMP Adapter for WBEM to Reread the Mapping File Directory

After you place a new mapping file or update an existing mapping file in `/var/sadm/wbem/snmp/map`, you must signal the SNMP Adapter for WBEM to reread all mapping files in the directory. You specify the signal SIGHUP to signal the adapter.

Follow these steps to force the SNMP Adapter for WBEM to reread all mapping files (without having to restart the CIM Object Manager).

1. **Become superuser.**
2. **Does a new mapping file or a new entry in a mapping file reference a subtree that is not registered by the SNMP Adapter for WBEM?**
   - If yes, go to the next step.
   - If no, go to step 5.
3. **Update** `/etc/snmp/conf/snmpXwbem.reg` **so that it includes the new subtree.**
4. **Stop and restart the Master Agent.**

```
# /etc/init.d/init.snmpdx stop
# /etc/init.d/init.snmpdx start
```

5. **Signal the SNMP Adapter for WBEM that you have updated a mapping file.**

```
# /usr/bin/pkill -1 -x -u 0 snmpXwbemd
```

Troubleshooting Problems With the SNMP Adapter for WBEM

The following sections provide a list of specific console error messages that you might encounter when using the SNMP Adapter for WBEM.

If you encounter errors, problems, or unexpected results that are not described in this section or, if you want to troubleshoot problems more precisely, use the Solaris Management Console Log Viewer to view log data.

Instructions that describe how to start the Solaris Management Console Log Viewer are presented in “Viewing Log Data Through Log Viewer” on page 78.

Sending and Receiving Requests

Error Message

```
ERROR: sending request to Adapter Service.

ERROR: receiving request from Adapter Service.
```

Cause

Either `snmpXwbemd` believes WBEM is enabled but cannot communicate with the Adapter Service, or the request timed out.

▼ Solution

Send another request. If sending another request fails, verify that the request and response FIFOs do not contain pending messages (that is, contain 0 bytes):

1. **Type:**

   # **cd /var/sadm/wbem/snmp**

2. **Type:**

   # **ls -l**

   The request and response FIFOs are listed.

3. **Does either FIFO contain pending messages (contain more than 0 bytes)?**

   If yes:

   a. **Stop the Master Agent.**

      # **/etc/init.d/init.snmpdx stop**

   b. **To ensure that you need to stop WBEM, use the Solaris Management Console Log Viewer to view log data and determine the problem.**

      Instructions that describe how to start the Solaris Management Console Log Viewer are presented in “Viewing Log Data Through Log Viewer” on page 78.

c. **If necessary, stop WBEM.**

```
# /etc/init.d/init.wbem stop
```

d. **Change to the directory in which the FIFOs are located.**

```
# cd /var/sadm/wbem/snmp
```

e. **Remove both FIFOs.**

```
# rm _adapter_rcv.fifo
# rm _adapter_snd.fifo
```

f. **Restart the Master Agent.**

```
# /etc/init.d/init.snmpdx start
```

g. **If you stopped WBEM in step c, restart it.**

```
# /etc/init.d/init.wbem start
```

If no:

h. **To ensure that you need to stop WBEM, use the Solaris Management Console Log Viewer to view log data and determine the problem.**

Instructions that describe how to start the Solaris Management Console Log Viewer are presented in "Viewing Log Data Through Log Viewer" on page 78.

i. **If necessary, stop WBEM.**

```
# /etc/init.d/init.wbem stop
```

j. **If you stopped WBEM in step i, restart it.**

```
# /etc/init.d/init.wbem start
```

FIFO Cannot Be Opened

Error Message

```
ERROR: request FIFO cannot be opened.

ERROR: response FIFO cannot be opened.
```

Cause

A protocol problem occurred when the SNMP Adapter for WBEM received a request or when the SNMP Adapter for WBEM processed a response.

▼ Solution

Send another request. If sending another request fails, verify that the request and response FIFOs do not contain pending messages (that is, contain 0 bytes):

1. **Type:**

   # **cd /var/sadm/wbem/snmp**

2. **Type:**

   # **ls -l**

   The request and response FIFOs are listed.

3. **Does either FIFO contain pending messages (contain more than 0 bytes)?**

   If yes:

k. **Stop the Master Agent.**

   # **/etc/init.d/init.snmpdx stop**

l. **To ensure that you need to stop WBEM, use the Solaris Management Console Log Viewer to view log data and determine the problem.**

   Instructions that describe how to start the Solaris Management Console Log Viewer are presented in "Viewing Log Data Through Log Viewer" on page 78.

m. **If necessary, stop WBEM.**

   # **/etc/init.d/init.wbem stop**

n. **Change to the directory in which the FIFOs are located.**

   # **cd /var/sadm/wbem/snmp**

o. **Remove both FIFOs.**

   # **rm _adapter_rcv.fifo**

   # **rm _adapter_snd.fifo**

p. **Restart the Master Agent.**

   # **/etc/init.d/init.snmpdx start**

q. **If you stopped WBEM in step m, restart it.**

   # **/etc/init.d/init.wbem start**

   If no:

r. **To ensure that you need to stop WBEM, use the Solaris Management Console Log Viewer to view log data and determine the problem.**

   Instructions that describe how to start the Solaris Management Console Log Viewer are presented in "Viewing Log Data Through Log Viewer" on page 78.

s. **If necessary, stop WBEM.**

```
# /etc/init.d/init.wbem stop
```

t. **If you stopped WBEM in step s, restart it.**

```
# /etc/init.d/init.wbem start
```

FIFO Cannot Be Created

Error Message

```
ERROR: FIFO cannot be created.
```

Cause

A system error occurred when the SNMP Adapter for WBEM attempted to create the request or the response FIFO.

Solution

Verify that `/var/sadm/wbem/snmp` exists and has write access.

WBEM Services Are Not Started

Error Message

```
ERROR: WBEM Services are not started.
```

Cause

The Master Agent cannot detect if WBEM Services have started and are running.

Solution

Restart WBEM and wait the number of seconds to which `watch-dog-time` in `snmpXwbem.reg` is set.

```
# /etc/init.d/init.wbem start
```

After a period of time, the Master Agent starts the SNMP Adapter for WBEM automatically. By default, the Master Agent tries to start the SNMP Adapter for WBEM every four minutes (or number of seconds to which `watch-dog-time` is set).

Note – If you don't want to wait for the Master Agent to start the SNMP Adapter for WBEM automatically, stop and then restart the Master Agent.

```
# /etc/init.d/init.snmpdx stop
```

```
# /etc/init.d/init.snmpdx start
```

The Master Agent immediately starts the SNMP Adapter for WBEM.

CHAPTER 4

Administering Security

This chapter describes WBEM security mechanisms and the features that the CIM Object Manager enforces. This chapter covers these topics:

WBEM Security Mechanisms

WBEM employs several mechanisms to ensure secure access to its data, including:

- Authentication, which is the process of specifying a client's user identity to the WBEM server, and then demonstrating that that client really is that particular user by specifying the user's credentials.
- Role Assumption, which is the process of assuming that a Solaris Role-Based Access Control (RBAC) role identity is to be used by the WBEM server when it checks authorization.
- Secure Messaging, which is the process of adding a secure message authenticator to each client request message. This authenticator enables the WBEM server to check the origin of the message and to determine if that message was modified during its delivery to the WBEM server.
- Authorization, which is the process of verifying that an authenticated user or a role identity has been granted access to the WBEM data that is managed by each WBEM method call. You use the Solaris Management Console User tool and Sun WBEM User Manager for authorization management.

- Auditing, which is the process of writing an audit record of a specific operation that was performed by the WBEM server. These records track the changes that an authenticated user makes to the management data on the WBEM server system.
- Logging, which is the writing of particular security related events in the WBEM log. You can view the WBEM log with the Solaris Management Console Log Viewer.

Each mechanism is described in more detail in the sections that follow.

Authentication

When a client application obtains a `CIMClient` handle to a CIM Object Manager server, the client's user identity must be authenticated by the CIM Object Manager on the WBEM server. The user's WBEM client must provide a Solaris user identity and its accompanied login password. The identity and credential is used in a security authentication exchange between the client and WBEM server to verify that the client is a valid Solaris user who is allowed to log in to the WBEM server system.

If the WBEM server cannot verify the user identity and credential, and the user's identity is invalid, the WBEM server returns a CIM security exception with the NO_SUCH_PRINCIPAL error. If the WBEM server cannot verify the user's identity and credential, and the user's password is invalid for that user's identity, the WBEM server returns a CIM security exception with the INVALID_CREDENTIAL error.

Role Assumption

The Solaris implementation of WBEM supports the ability for a client to assume the identity of a Solaris role when that client is authenticated by the CIM Object Manager on the WBEM server. When the WBEM server uses RBAC authorizations to check authorization permission, the WBEM server checks the permission that is granted to the assumed role rather than the permission that is granted to the underlying user identity. (RBAC roles are described in more detail in "Role-Based Access Control (Overview)" in *System Administration Guide: Security Services*.)

The client must provide the Solaris role identity and password in addition to a Solaris user identity and password when the client attempts to obtain a handle to the CIM client. If the WBEM server cannot verify the Solaris role identity, the WBEM server returns a CIM security exception with the NO_SUCH_ROLE error. If the role password is invalid for the specified role identity, the WBEM server returns the INVALID_CREDENTIAL error in the CIM security exception.

If both the role identity and role password are valid, but the user is not allowed to assume the role, the WBEM server returns the CANNOT_ASSUME_ROLE error in the CIM security exception.

A role identity can be assumed only when the user selects the CIM_RMI protocol. Role assumption is not supported by the CIM_XML protocol.

Secure Messaging

In the CIM_RMI protocol, each request from the client to the WBEM server contains a message authenticator that is:

1. Constructed from the data parameters in the message.
2. Encrypted with a session key established during the authentication exchange.

The WBEM server verifies this message authenticator, which guarantees that the request came from the same client that was authenticated and that the message was not modified or replayed during its communications to the server.

If the message was modified, replayed, or created by a source which was not the original client, the WBEM server returns a CIM security exception with the CHECKSUM_ERROR error. The WBEM server also writes a log message to the WBEM log.

Authorization

The WBEM server uses the authenticated user or role identity for all authorization checks on subsequent operations to the CIM client after the WBEM server obtains the handle to the CIM client.

WBEM supports two types of authorization checking, based on:

- Access Control Lists (ACLs) that are maintained by the WBEM server for specific namespaces.
- RBAC authorizations that are configured as part of the Solaris operating environment.

The particular authorization checking mechanism that WBEM uses depends on how the MOF class provider is implemented. The particular authorization checking mechanism that WBEM uses for a specific MOF class operation depends on:

- The particular operation that WBEM executes.
- How the MOF class provider is implemented.

The classes defined in `Solaris_Acl1.0.mof` implement WBEM ACL-based security. WBEM ACL-based security provides a default authorization scheme for Solaris WBEM Services, and, under specific circumstances, applies to a particular set of CIM operations. ACL-based security is uniquely provided by Solaris WBEM Services.

You use Sun WBEM User Manager (`wbemadmin`(1M)) to establish an ACL for a specific namespace on the WBEM server. Sun WBEM User Manager enables you to add user names, or role names, to the ACL for the namespace, and also enables you to assign each user "read" or "write" permission. Sun WBEM User Manager is described in "Using Sun WBEM User Manager to Set Access Control" on page 55.

Write permission allows a user to modify the class metadata and instances of MOF classes in that namespace. The local WBEM server root user identity is always granted write permission to all namespaces on the server. All authenticated users without an explicit ACL entry are granted read permission by default.

Operations that include the accessing of MOF class metadata, such as getClass, use the WBEM ACLs. These operations include the checking of permissions that are granted to the authenticated user by the ACL for the namespace that contains the MOF class. You can set an RBAC role in an ACL entry, but the ACL entry is always checked against the user identity rather than the role identity.

Operations that involve MOF class instances might include the checking of either WBEM ACLs or RBAC authorizations.

You can also grant permissions to a user, or role identity, that allow that user to access and modify the instances of MOF classes whose providers use the RBAC authorizations. You grant these permissions by using the Rights tool in the Solaris Management Console User tool. Granting permissions to a user is described in "Creating or Changing a Rights Profile" in *System Administration Guide: Security Services*.

If the instances for a MOF class are stored in the WBEM persistent data store, the CIM Object Manager checks WBEM ACL for the namespace that contains the MOF class. If the MOF class provider implementation accesses its own data store, or accesses system data in the Solaris operating environment, the MOF class provider implementation almost always uses RBAC authorization checking.

In general, if a MOF class definition contains a Provider qualifier, the provider implementation usually makes RBAC authorization checks. If the MOF class definition does not contain a Provider qualifier, the CIM Object Manager checks the ACL that controls access to the namespace for the class to ensure that access is granted.

the instances of that class are stored in the WBEM persistent data store and the ACL controlling access to the namespace for the class is checked for granted access.

Auditing

The WBEM server writes audit records for certain events during processing. For example, the WBEM server writes audit records whenever the authentication of a client succeeds or fails, and whenever an operation that modifies user information is executed. The WBEM server uses the underlying Solaris Basic Security Module (BSM)

to write its audit records. You must enable the BSM auditing mechanism (`bsmconv`(1M)) in the Solaris operating environment on the WBEM server to ensure that audit information is recorded.

Note – If you are using Trusted Solaris™, you do *not* need to enable the BSM auditing mechanism.

Logging

The WBEM server writes log records to the WBEM log for particular security events: when an authenticated session for a client is established or when authorization checking fails, for example. You can review the WBEM log in the Solaris Management Console Log Viewer, which is described in Chapter 6.

You can identify security-related log events by the category Security log, which is listed in the Category column. You can view only security log messages by selecting the category Security on the Log Viewer filter dialog box. Most security log messages include the user identity of the client and the name of the client host.

Using Sun WBEM User Manager to Set Access Control

Sun WBEM User Manager enables you and other privileged users to:

- Add and delete authorized users.
- Set access privileges for authorized users.
- Manage user authentication and access to CIM objects on a WBEM-enabled system.

Note – The user for whom you specify access control must have a Solaris user account.

What You Can and Cannot Do With Sun WBEM User Manager

You can set access privileges for individual namespaces or for a combination of a user and a namespace. When you add a user and select a namespace, the user is granted read access to CIM objects in the selected namespace by default.

Note – An effective way to combine user and namespace access rights is to first restrict access to a namespace, and then grant individual users read, read and write, or write access to that namespace.

You cannot set access rights on individual managed objects. However you can set access rights for all managed objects in a namespace as well as on a per-user basis.

If you log in as root, you can set the following types of access to CIM objects:

- Read Only – Allows read-only access to CIM Schema objects. Users with this privilege can retrieve instances and classes, but cannot create, delete, or modify CIM objects.
- Read/Write – Allows full read, write, and delete access to all CIM classes and instances.
- Write – Allows write and delete, but not read access to all CIM classes and instances.
- None – Allows no access to CIM classes and instances.

Using Sun WBEM User Manager

This section describes how to start and use Sun WBEM User Manager.

▼ How to Start Sun WBEM User Manager

1. **In a command window, type:**

   ```
   # /usr/sadm/bin/wbemadmin
   ```

   Sun WBEM User Manager starts, and a Login dialog box opens.

Note – Context-help information is available in the Context Help panel when you click on the fields in the Login dialog box.

2. **Fill in the fields on the Login dialog box:**
   - In the User Name field, type the user name.

     Note – You must have read access to the `root\security` namespace to log in. By default, Solaris users have guest privileges, which grant them read access to the default namespaces. Users with read access can view , but cannot change, user privileges.

     You must log in as root or a user with write access to the `root\security` namespace to grant access rights to users.

   - In the Password field, type the password for the user account.
3. **Click OK.**

   The User Manager dialog box opens. The dialog box contains a list of users and their access rights to WBEM objects within the namespaces on the current host.

▼ How to Grant Default Access Rights to a User

1. **Start Sun WBEM User Manager.**
2. **In the Users Access portion of the dialog box, click Add.**

   A dialog box opens that lists the available namespaces.
3. **Type the name of a Solaris user account in the User Name field.**
4. **Select a namespace from the listed namespaces.**
5. **Click OK.**

   The user name is added to the User Manager dialog box.
6. **To save changes and close the User Manager dialog box, click OK. To save changes and keep the dialog box open, click Apply.**

   The user that you specified is granted read access to CIM objects in the namespace that you selected.

▼ How to Change Access Rights for a User

1. **Start Sun WBEM User Manager.**
2. **Select the user whose access rights you want to change.**
3. **To grant the user read-only access, click the Read check box. To grant the user write access, click the Write check box.**
4. **To save changes and close the User Manager dialog box, click OK. To save changes and keep the dialog box open, click Apply.**

▼ How to Remove Access Rights for a User

1. **Start Sun WBEM User Manager.**
2. **In the Users Access portion of the dialog box, select the user name for which you want to remove access rights.**
3. **Click Delete to delete the user's access rights to the namespace.**

   A confirmation dialog box that prompts you to confirm your decision to delete the user's access rights opens.
4. **To confirm, click OK.**
5. **To save changes and close the User Manager dialog box, click OK. To save changes and keep the dialog box open, click Apply.**

▼ How to Set Access Rights for a Namespace

1. **Start Sun WBEM User Manager.**
2. **In the Namespace Access portion of the dialog box, click Add.**

   A dialog box opens. The dialog box lists the available namespaces.
3. **Select the namespace for which you want to set access rights:**

   Note – By default, users have read-only access to a namespace.

   - To allow no access to the namespace, make sure the Read and Write check boxes are not selected.
   - To allow write access, click the Write check box.
   - To allow read access, click the Read check box.

4. **To save changes and close the User Manager dialog box, click OK. To save changes and keep the dialog box open, click Apply.**

▼ How to Remove Access Rights for a Namespace

1. **Start Sun WBEM User Manager.**
2. **In the Namespace Access portion of the dialog box, select the namespace for which you want to remove access control, and then click Delete.**

   Access control is removed from the namespace, and the namespace is removed from the list of namespaces on the Sun WBEM User Manager dialog box.
3. **To save changes and close the User Manager dialog box, click OK. To save changes and keep the dialog box open, click Apply.**

Using the Sun WBEM SDK APIs to Set Access Control

You can use Sun WBEM Software Development Kit Application Programming Interfaces (SDK APIs) to set access control on a namespace or on a per-user basis. These security classes are stored in the `root\security` namespace:

- `Solaris_Acl` – Base class for Solaris Access Control Lists (ACL). This class defines the string property *capability* and sets its default value to `r` (read only).
- `Solaris_UserAcl` – Represents the access control that a user has to the CIM objects within the specified namespace.
- `Solaris_NamespaceAcl` – Represents the access control on a namespace.

You can set access control for individual users to the CIM objects within a namespace by creating an instance of the `Solaris_UserACL` class and then using the APIs to change the access rights for that instance. Similarly, you can set access control for namespaces by creating an instance of the `Solaris_NameSpaceACL` class and then using APIs, such as the `setInstance` method, to set the access rights for that instance.

An effective way to combine the use of these two classes is to use the `Solaris_NameSpaceACL` class first to restrict access to all users to the objects in a namespace. Then, you can use the `Solaris_UserACL` class to grant selected users access to the namespace.

The `Solaris_UserAcl` Class

The `Solaris_UserAcl` class extends the `Solaris_Acl` base class, from which it inherits the string property *capability* with a default value `r` (read only).

You can set the *capability* property to any one of these values for access privileges.

Access Right	Description
`r`	Read
`rw`	Read and Write
`w`	Write
none	No access

The `Solaris_UserAcl` class defines the following two key properties. Only one instance of the namespace and user name ACL pair can exist in a namespace.

Property	Data Type	Purpose
`nspace`	`string`	Identifies the namespace to which this ACL applies.
`username`	`string`	Identifies the user to which this ACL applies.

▼ To Set Access Control for a User

1. **Create an instance of the `Solaris_UserAcl` class. For example:**

```
...
/* Create a namespace object initialized with root\security
(name of namespace) on the local host. */

CIMNameSpace cns = new CIMNameSpace("", "root\security");

// Connect to the root\security namespace as root.
cc = new CIMClient(cns, user, user_passwd);

// Get the Solaris_UserAcl class
cimclass = cc.getClass(new CIMObjectPath("Solaris_UserAcl");

// Create a new instance of the Solaris_UserAcl
class ci = cimclass.newInstance();
...
```

2. **Set the *capability* property to the desired access rights. For example:**

```
...
/* Change the access rights (capability) to read/write for user Guest
on objects in the root\molly namespace.*/
ci.setProperty("capability", new CIMValue(new String("rw"));
ci.setProperty("nspace", new CIMValue(new String("root\molly"));
ci.setProperty("username", new CIMValue(new String("guest"));
...
```

3. **Update the instance. For example:**

```
...
// Pass the updated instance to the CIM Object Manager
cc.setInstance(new CIMObjectPath(), ci);
...
```

The `Solaris_NamespaceAcl` Class

The `Solaris_NamespaceAcl` extends the `Solaris_Acl` base class, from which it inherits the string property *capability* with a default value `r` (read-only for all users). The `Solaris_NamespaceAcl` class defines this key property.

Property	Data Type	Purpose
`nspace`	`string`	Identifies the namespace to which this access control list applies. Only one instance of the namespace ACL can exist in a namespace.

▼ To Set Access Control for a Namespace

1. **Create an instance of the `Solaris_namespaceAcl` class. For example:**

```
...
/* Create a namespace object initialized with root\security
(name of namespace) on the local host. */
CIMNameSpace cns = new CIMNameSpace("", "root\security");

// Connect to the root\security namespace as root.
cc = new CIMClient(cns, user, user_passwd);

// Get the Solaris_namespaceAcl class
cimclass = cc.getClass(new CIMObjectPath("Solaris_namespaceAcl");

// Create a new instance of the Solaris_namespaceAcl
```

```
class ci = cimclass.newInstance();
...
```

2. **Set the *capability* property to the desired access rights. For example:**

```
...
/* Change the access rights (capability) to read/write
to the root\molly namespace. */
ci.setProperty("capability", new CIMValue(new String("rw"));
ci.setProperty("nspace", new CIMValue(new String("root\molly"));
...
```

3. **Update the instance. For example:**

```
// Pass the updated instance to the CIM Object Manager
cc.setInstance(new CIMObjectPath(), ci);
```

Troubleshooting

This section describes what to do when:

- A client (user) cannot be authenticated by the CIM Object Manager on the WBEM server.
- A role cannot be assumed.
- An ACCESS_DENIED error occurs.

If a Client (User) Cannot Be Authenticated by the CIM Object Manager on the WBEM Server

If a client cannot be successfully authenticated by the CIM Object Manager on the WBEM server, the WBEM server returns a CIM security exception when it attempts to establish the CIM client handle in the client application. The exception contains an error code that indicates why the authentication attempt failed.

If the WBEM server cannot verify the user identity and credential, and the user's identity is invalid, the WBEM server returns a CIM security exception with the NO_SUCH_PRINCIPAL error. If the WBEM server cannot verify the user's identity and credential, and the user's password is invalid for that user's identity, the WBEM server returns a CIM security exception with the INVALID_CREDENTIAL error.

If the WBEM server cannot verify the Solaris role identity, the WBEM server returns a CIM security exception with the NO_SUCH_ROLE error. If the role password is invalid for the specified role identity, the WBEM server returns the

INVALID_CREDENTIAL error in the CIM security exception. If both the role identity and role password are valid, but the user is not allowed to assume the role, the WBEM server returns the CANNOT_ASSUME_ROLE error in the CIM security exception.

These CIM security exceptions are described in more detail in the following table.

Error	Probable Cause	Solution
NO_SUCH_PRINCIPAL	Specified user identity was not a valid user identity in the Solaris operating environment on the WBEM server, or that the user account for that user identity either has no password or is locked.	Check that the user is a valid user identity, that is, can log in to the Solaris operating environment on the WBEM server machine. The WBEM server Solaris system might be using user identities from a name service configured on the server, so you might need to check the name service tables.
INVALID_CREDENTIAL	Password for the specified user (or role, if assuming a role identity) is not valid for that user in the Solaris operating environment on the WBEM server.	Check that the user's password is correct.

Error	Probable Cause	Solution
NO_SUCH_ROLE	Role identity that is assumed in the authentication to the WBEM server is not a valid RBAC role in the Solaris operating environment on the WBEM server.	The role identity might be a valid entry in the `passwd` table on the server, but you will *not* be able to log in to the server under that identity (Solaris does not allow you to login directly to role identities). So, you must check the `passwd` table for the role identity, and check the `user_attr` table to ensure that the role is defined as a role type of user. Role identities in the `user_attr` table each contain an attribute in the syntax *type=role*. You can also check for a valid user or role identity with the Solaris Management Console User tool. You can use User Management to check for a user and you can use Role Management to check for a role. However, when using the User tool, you *must* know the correct source of the tables on the CIM Object Manager server. In other words, if the CIM Object Manager server is using a name service such as NIS, you must access the master server for that name service.
CANNOT_ASSUME_ROLE	Role identity is valid, but the specified user identity in the authentication exchange is not configured to assume that role.	Explicitly assign users to roles with the Administrative Role tool in the Solaris Management Console User tool collection, which is described in "Changing Role Properties" in *System Administration Guide: Security Services*.

If Other CIM Security Exception Errors Appear

The WBEM server can return other error indications in the CIM security exception. However, these indications typically indicate a system failure in the authentication exchange. The WBEM client configuration might not be compatible with the WBEM server configuration for the security options for the authentication exchange.

If these error indications appear, check that the WBEM installation on the client machine contains the appropriate configuration property values for security in `WbemClient.properties`. This file is usually located in the vendor extension subdirectory in the WBEM installation directory: `/usr/sadm/lib/wbem/extension`.

Also, check the client application CLASSPATH setting to ensure that `sunwbem.jar` and the extension directory path name are on the class path.

If an Authorization Check Fails

If a client is not authorized to access or modify the data associated with a request to the WBEM server, the WBEM server returns a CIM security exception for that request with the ACCESS_DENIED error.

The ACCESS_DENIED error indicates that a WBEM request could not be completed because the authenticated user or role has not been granted the appropriate access to the data being managed by that request.

Check the security messages in the WBEM log for the failed request. Authorization failure log messages specify `Access denied` in the Summary column. The User column lists the name of the authenticated user or role name that was used in the check, and the Source column lists the name of the provider that is making the check. Note that the name of the provider that is listed in this column is a user-friendly provider name, and not the provider implementation class name.

The detailed message contains the name of the permission that was being checked (and which has not been granted to the user or role).

If the permission appears as *namespace*:*right*, the authorization check was using a name space ACL. The authenticated user has not been granted that permission (read or write) for that namespace.

Use Sun WBEM User Manager (`wbemadmin`) to grant the user the appropriate permission. Sun WBEM User Manager is described in "Using Sun WBEM User Manager to Set Access Control" on page 55.

If the permission appears as `solaris.`*application*`.`*right*, the authorization check was using an RBAC authorization.

Use the Administrative Role tool in the Solaris Management Console User tool collection, which is described in "Changing Role Properties" in *System Administration Guide: Security Services*, to grant the rights that you want to the user or role.

CHAPTER 5

MOF Compiler

This chapter describes the Managed Object Format (MOF) Compiler, and covers the following topics.

The MOF Compiler

Definitions

Managed Object Format (MOF) is a language for defining CIM classes and instances. MOF files are ASCII text files that use MOF to describe CIM objects. A CIM object is a computer representation or model of a managed resource, such as a printer, disk drive, or CPU.

How the MOF Compiler Works

Generally, the MOF Compiler:

- Parses a file containing MOF statements
- Converts the classes and instances defined in the file to Java classes
- Adds the Java classes to the CIM Object Manager Repository, which is a central storage area that contains management data

The compiler loads the Java classes into the default namespace, `root\cimv2`, unless a `#pragma namespace("`*namespace_path*`")` statement appears in the MOF file.

The Solaris installer runs the `mofreg`(1M) command, which starts the MOF compiler, before installation to compile MOF files that describe the CIM and Solaris Schema. The CIM Schema is a collection of class definitions used to represent managed objects that occur in every management environment. The Solaris Schema is a collection of class definitions that extend the CIM Schema and represent managed objects in a typical Solaris operating environment. You can use the `mofcomp`(1M) command to compile MOF files at any time after installation.

Many sites store information about managed resources in MOF files. Because MOF can be converted to Java, Java applications can interpret and exchange this information on any system on which a Java Virtual Machine is available.

Compiling a MOF File

You can compile a MOF file with or without a `.mof` extension. The MOF files that describe the CIM and Solaris Schema are located in `/usr/sadm/mof`.

Note – You must become superuser before you recompile `CIM_Schema25.mof` or `Solaris_Schema1.0.mof`.

The `mofcomp` Command

The `mofcomp`(1M) command compiles the specified MOF file into CIM classes and instances. These CIM classes and instances are stored in the CIM Object Manager Repository as Java classes, which are passed to the CIM Object Manager.

You must run the `mofcomp` command as root or as a user with write access to the namespace in which you are compiling.

▼ To Compile a MOF File

- **To compile a MOF file (without options), type:**

  `#` **`mofcomp`** *filename*

  Example:

  `#` **`mofcomp /usr/sadm/mof/Solaris_Application1.0.mof`**

The MOF file is compiled into the CIM Object Manager Repository.

Security Advisory

If you run the `mofcomp` command with the `-p` option or the `-u` and `-p` options, and you include a password on the command line, another user can run the `ps` command or the `history` command and see your password.

Note – If you run a command that requires you to provide your password, immediately change your password after you run the command.

The following examples show the unsecure use of the `mofcomp` command.

```
% mofcomp -p Log8Rif /usr/sadm/mof/Solaris_Acl1.0.mof
```

```
% mofcomp -up molly Log8Rif /usr/sadm/mof/Solaris_Acl1.0.mof
```

Change your password immediately after you run the `mofcomp` command with either the `-p` or `-up` options.

Generating a MOF File From an SNMP MIB File

When you want to access Simple Network Management Protocol (SNMP) information through the SNMP Provider, you use a Management Information Base (MIB) file to generate a MOF file. The `mib2mof`(1M) command generates qualifiers that enable the SNMP Provider to map CIM operations that are performed on the CIM classes in the MOF file to SNMP operations.

Note – The SNMP Provider for WBEM supports SNMP traps. Traps are reported in the CIM process indication event `CIM_SNMPTrapIndication`. When a client subscribes to the provider for this event, the provider listens on port 162 for SNMP V1 and SNMP V2 traps. The information is copied from the trap to the indication and then the indication is delivered to the client.

The MOF files that describe the CIM and Solaris Schema are located in `/usr/sadm/mof`.

▼ To Generate a MOF File From an SNMP MIB File

1. **Become superuser.**
2. **Type the command:**

   # **mib2mof** *SNMP_MIB_filename*

   Example:

   # **mib2mof sysctl.mib**

CHAPTER 6

System Logging

WBEM log files enable system administrators to track errors, warnings, and informational messages that the management subsystem generates. The WBEM logging service enables application developers and writers of providers to write log messages to the log files. For example, you might want to write out log messages when a system is not able to access a serial port, when a system successfully mounts a file system, or when the number of processes that are running on a system exceeds the allowed number.

This chapter covers the following topics:

About Logging

When a provider encounters a particular condition in the underlying managed system that it wants to record, the provider requests that the logging service, through an application programming interface, store information in the WBEM log file. Providers can not only log errors or warnings, but can also log informational messages as well. Log files provide the administrator with a tool to trace the cause of failures by providing a way to trace the history of the actions, errors, and warnings that led to the failure.

Providers can write log messages using the application programming interfaces that are described in "Using a Client's Application Programming Interface to Read and to Write Log Messages" on page 73. The application programming interface also

provides a way to forward log messages to `syslogd`(1M), the default logging system in the Solaris operating environment. You can view log messages in the Solaris Management Console Log Viewer.

Log Files

By default, all log messages are stored in files in `/var/sadm/wbem/log`. The size to which each log file can grow is limited. When a log file reaches this limit, WBEM automatically backs up the log file and creates a new log file in its place. The number of log files and the number of backed up log files that can exist at one time is limited.

You can manually back up the active log file by invoking `ClearLog()` in the `Solaris_MessageLog` class.

The names of the log files, the directory in which to store the log files, the file size limit, the number of files to store, and whether to forward messages to `syslogd`(1M) are properties that you can configure. You manipulate these properties by modifying the properties in the singleton instance of the `Solaris_LogServiceProperties` class.

Log Message Format

The format of each log entry is defined by the `Solaris_LogEntry` class, which is a subclass of `CIM_LogRecord`. You can find `CIM_LogRecord` in `Solaris_Device1.0.mof`.

A log message comprises the following elements:

- `Category` – Type of message: application, system, or security
- `Severity` – Severity of the condition that caused the message to be written: informational, warning, or error
- `Application` – Name of the application (or the provider) that is writing the log message
- `User` – Name of the user who was using the application when the log message was generated
- `Client Machine` – Name and Internetworking Protocal (IP) address of the system that `User` was using when the log message was generated
- `Server Machine` – The name of the system on which the incident that generated the log message occurred
- `Summary Message` – A descriptive summary of the incident

- `Detailed Message` – A detailed description of the incident
- `Data` – Any contextual information that might provide a better understanding of the incident
- `SyslogFlag` – A boolean flag that specifies whether or not to send the message to `syslogd`(1M)

Using a Client's Application Programming Interface to Read and to Write Log Messages

To Use a Client's Application Programming Interface to Read and to Write Log Messages

You can use the Solaris Management Console Log Viewer to view the contents of the log file, as described in "Viewing Log Data Through Log Viewer" on page 78. You can also set up a client to use the client's application programming interface to read and to write log messages.

▼ To Read From the Log File

- **Use this code sample as a guide to set up a client to read from the log file.**

```
import javax.wbem.client.*;
import javax.wbem.cim.*;
import javax.wbem.security.*;

import java.rmi.*;
import java.util.*;
import java.io.*;
import java.lang.*;
import java.util.Enumeration;

/**
 * This example displays a list of log records.
 */
public class ReadLog {
   public static void main(String args[]) throws CIMException {
```

```
String protocol = CIMClient.CIM_RMI;
if (args.length < 3) {
 System.out.println("Usage: ReadLog host username password [rmi|http]");
 System.exit(1);
}
CIMClient cc = null;
CIMObjectPath cop = null;
CIMObjectPath serviceObjPath = null;
Vector inVec = new Vector();
Vector outVec = new Vector();
try {
    CIMNameSpace cns = new CIMNameSpace(args[0]);
    UserPrincipal up = new UserPrincipal(args[1]);
    PasswordCredential pc = new PasswordCredential(args[2]);
    if (args.length == 4 && args[3].equalsIgnoreCase("http")) {
        protocol = CIMClient.CIM_XML;
    }
    cc = new CIMClient(cns, up, pc, protocol);

    cop = new CIMObjectPath("Solaris_LogEntry");
    Enumeration e = cc.enumerateInstances(cop, true, false,
                    false, false, null);
    for (; e.hasMoreElements(); ) {
        System.out.println("---------------------------------");
    CIMInstance ci = (CIMInstance)e.nextElement();
    System.out.println("Log filename : " +
((String)ci.getProperty("LogName").getValue().getValue()));
    int categ =
(((Integer)ci.getProperty("Category").getValue().getValue()).intValue());
    if (categ == 0)
        System.out.println("Category : Application Log");
    else if (categ == 1)
        System.out.println("Category : Security Log");
    else if (categ == 2)
        System.out.println("Category : System Log");
    int severity =
(((Integer)ci.getProperty("Severity").getValue().getValue()).intValue());
    if (severity == 0)
        System.out.println("Severity : Informational");
    else if (severity == 1)
        System.out.println("Severity : Warning Log!");
    else if (severity == 2)
        System.out.println("Severity : Error!!");
    System.out.println("Log Record written by :" +
((String)ci.getProperty("Source").getValue().getValue()));
    System.out.println("User : " +
((String)ci.getProperty("UserName").getValue().getValue()));
    System.out.println("Client Machine : " +
((String)ci.getProperty("ClientMachineName").getValue().getValue()));
    System.out.println("Server Machine : " +
((String)ci.getProperty("ServerMachineName").getValue().getValue()));
    System.out.println("Summary Message : " +
((String)ci.getProperty("SummaryMessage").getValue().getValue()));
    System.out.println("Detailed Message : " +
```

```
    ((String)ci.getProperty("DetailedMessage").getValue().getValue()));
        System.out.println("Additional data : " +
    ((String)ci.getProperty("RecordData").getValue().getValue()));
        boolean syslogflag = ((Boolean)ci.getProperty
                        ("SyslogFlag").getValue().getValue()).booleanValue();
        if (syslogflag == true) {
            System.out.println("Record was written to syslog");
        } else {
            System.out.println("Record was not written to syslog");
        }
        System.out.println("---------------------------------");
        }
    }
    catch (Exception e) {
        System.out.println("Exception: "+e);
        e.printStackTrace();
    }

    // close session.
    if (cc != null) {
        cc.close();
    }
    }
}
```

To Use a Client's Application Programming Interface to Create Log Messages

You can also set up a client to use the client's application programming interface to create log messages.

▼ To Create Log Messages

- **Use this code sample as a guide to set up a client to create log messages.**

```
import javax.wbem.client.*;
import javax.wbem.cim.*;
import javax.wbem.security.*;

import java.rmi.*;
import java.util.*;
import java.io.*;
import java.lang.*;
import java.util.Enumeration;

/**
 * This example writes out a new log message.
 */
```

```
public class CreateLog {
    public static void main(String args[]) throws CIMException {

    if (args.length < 3) {
        System.out.println("Usage: CreateLog host username password " +
                    "[rmi|http]");
        System.exit(1);
    }

    String protocol = CIMClient.CIM_RMI;
    CIMClient cc = null;
    CIMObjectPath cop = null;
    BufferedReader d = new BufferedReader(new InputStreamReader(System.in));
    String input_line = "";
    System.out.print("How many log records do you want to write? ");
    int num_recs = 0;

    try {
        num_recs = Integer.parseInt(d.readLine());
    } catch (Exception ex) {
        ex.printStackTrace();
        System.exit(1);
    }
    try {
        CIMNameSpace cns = new CIMNameSpace(args[0]);
        UserPrincipal up = new UserPrincipal(args[1]);
        PasswordCredential pc = new PasswordCredential(args[2]);
        if (args.length == 4 && args[3].equalsIgnoreCase("http")) {
            protocol = CIMClient.CIM_XML;
        }
        cc = new CIMClient(cns, up, pc, protocol);

        Vector keys = new Vector();
        CIMProperty logsvcKey = null;


        System.out.println("Please enter the record Category: ");
        System.out.println("\t(0)application, (1)security, (2)system");
        logsvcKey = new CIMProperty("category");
        input_line = d.readLine();
        logsvcKey.setValue(new CIMValue(Integer.valueOf(input_line)));
        keys.addElement(logsvcKey);
        System.out.println("Please enter the record Severity: ");
        System.out.println("\t(0)Informational, (1)Warning, (2)Error");
        logsvcKey = new CIMProperty("severity");
        input_line = d.readLine();
        logsvcKey.setValue(new CIMValue(Integer.valueOf(input_line)));
        keys.addElement(logsvcKey);
        logsvcKey = new CIMProperty("Source");
        System.out.println("Please enter Application Name: ");
        logsvcKey.setValue(new CIMValue(d.readLine()));
        keys.addElement(logsvcKey);
        logsvcKey = new CIMProperty("SummaryMessage");
        System.out.println("Please enter a summary message: ");
```

```
        logsvcKey.setValue(new CIMValue(d.readLine()));
        keys.addElement(logsvcKey);
        logsvcKey = new CIMProperty("DetailedMessage");
        System.out.println("Please enter a detailed message: ");
        logsvcKey.setValue(new CIMValue(d.readLine()));
        keys.addElement(logsvcKey);
        logsvcKey = new CIMProperty("RecordData");
        logsvcKey.setValue(
            new CIMValue("0xfe 0x45 0xae 0xda random data"));
        keys.addElement(logsvcKey);
        logsvcKey = new CIMProperty("SyslogFlag");
        logsvcKey.setValue(new CIMValue(new Boolean(true)));
        keys.addElement(logsvcKey);
         CIMObjectPath logreccop =
                new CIMObjectPath("Solaris_LogEntry", keys);
        CIMClass logClass = cc.getClass(logreccop);
        CIMInstance ci = logClass.newInstance();
        ci.setClassName("Solaris_LogEntry");
        ci.setProperties(keys);
        // System.out.println(ci.toString());
        for (int i = 0; i < num_recs; i++) {
            cc.createInstance(logreccop, ci);
        }
    }
    catch (Exception e) {
        System.out.println("Exception: "+e);
        e.printStackTrace();
    }

    // close session.
    if (cc != null) {
        cc.close();
    }
    }
}
```

To Use Provider Application Programming Interfaces to Write Log Messages

The code that you use to generate log messages from a provider are similar to the code that is shown in "To Use a Client's Application Programming Interface to Create Log Messages" on page 75. Instead of using a `CIMClient` object to access the CIM object manager, you use a `cimomhandle` object. The provider must create an instance of the `Solaris_LogEntry` class and use the `cimomhandle` object to send the instance to the provider for the `Solaris_LogEntry` class.

Viewing Log Data Through Log Viewer

You can view all details of a log record in the Solaris Management Console Log Viewer.

Starting Log Viewer

After you have created a log record, you can start the Solaris Management Console application and Log Viewer. A log record is automatically created when you start the Solaris Management Console software.

▼ To Start the Solaris Management Console Application and Log Viewer

1. **Type the command:**

   ```
   % smc
   ```

2. **In the Navigation panel, double-click This Computer (or single-click the expand/compress icon next to This Computer).**

   A tree of commands is displayed below This Computer.

3. **Double-click System Status.**

   The Log Viewer icon is displayed.

4. **Click the Log Viewer icon.**

   Log Viewer starts.

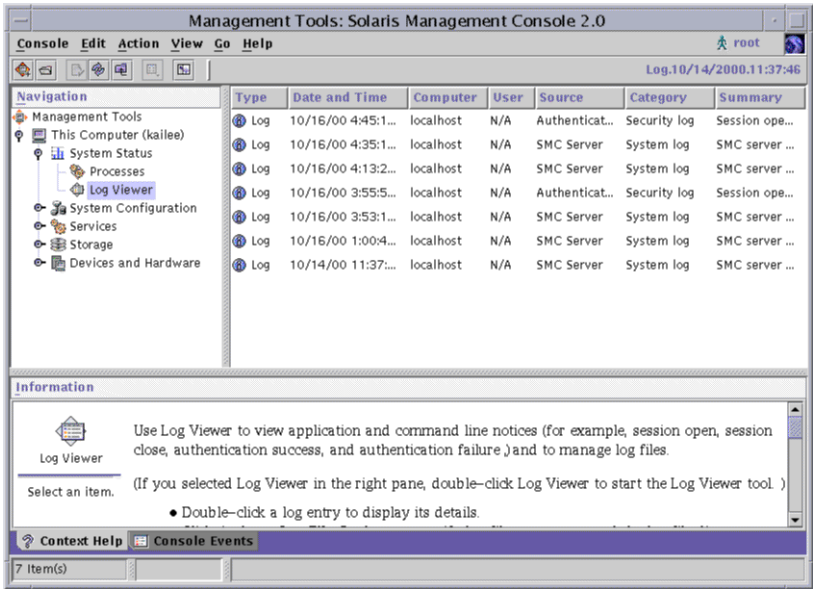

FIGURE 6–1 The Solaris Management Console Application, Log Viewer Selected

APPENDIX **A**

The Solaris Schema

During installation, the CIM Object Manager compiles MOF files that describe the CIM Schema and the Solaris Schema in the directory `/usr/sadm/mof/`. CIM Schema files, which implement the Core and Common Models of the Common Information Model, are denoted by the use of CIM in their associated file names. The Solaris Schema files, denoted by the use of Solaris in their file names, provide the implementation of the Solaris extension into the Common Information Model. This appendix describes the Solaris Schema files.

Documentation of the Solaris providers listed in this chapter is included in the MOF file in which the provider is specified.

Solaris Schema Files

This table provides a brief overview of the Solaris Schema files in `/usr/sadm/mof`.

TABLE A–1 Solaris Schema Files

Solaris schema file	What this schema file provides
`Solaris_Schema1.0.mof`	Lists all of the MOF files of the Solaris Schema, in `#pragma Include` statements. Specifies the order in which the MOF files are read and compiled.
`Solaris_CIMOM1.0.mof`	Contains all the system properties that the CIM Object Manager uses.
`Solaris_Core1.0.mof`	Enables WBEM core features to be implemented. Enables you to set locales, qualifiers, and providers.
`Solaris_Application1.0.mof`	Models Solaris packages and patches in CIM.
`Solaris_System1.0.mof`	Models the Solaris Schema components for a system, including the operating system and processes of the system. Extends CIM Schema definitions through the definition of the `Solaris_Process` and `Solaris_OperatingSystem` classes.
`Solaris_Device1.0.mof`	Enables a description of your system's processor, serial ports, printing devices, and time settings to make your computer work with the CIM Object Manager.
`Solaris_Acl1.0.mof`	Sets the base class and qualifiers for user ACLs.
`Solaris_Network1.0.mof`	Defines classes pertaining to network domains, IP subnets, and naming services (including NIS, NIS+, LDAP, DNS, and server `/etc` files).
`Solaris_Users1.0.mof`	Defines classes for working with user accounts.
`Solaris_Event1.0.mof`	Defines a single class, `Solaris_RMIDelivery`, which is a subclass of `CIM_IndicationHandler`. This class is defined to facilitate the delivery of indications to management clients over Sun Microsystems' implementation of the CIM Remote Method Invocation (RMI) protocol.
`Solaris_SNMP1.0.mof`	Defines classes that pertain to configuration information for an SNMP device.
`Solaris_LVM1.0.mof`	Defines classes that pertain to storage devices
`Solaris_Project.0.mof`	Defines classes that model the Solaris project database.

The following sections describe the contents of each schema file in more detail.

The Solaris_Schema1.0.mof File

The Solaris_Schema1.0.mof file is the high-level container of all other MOF files comprised by the Solaris Schema. This file lists the MOF files in the order in which you must compile them.

The Java classes that you generate from each compilation are then sent to the CIM Object Manager, where they are either enacted as events or sent to the CIM Repository for storage as objects. The following listing of the Solaris_Schema1.0.mof file shows the Include statements in the order that is required for compilation.

```
/*
Title        Solaris Master MOF 1.0
Description include pragmas for all other mofs
Date        03/10/01
Version     1.1
Copyright   (c) 2000 Sun Microsystems, Inc. All Rights Reserved.
*/

#pragma Include ("Solaris_Core1.0.mof")
#pragma Include ("Solaris_Application1.0.mof")
#pragma Include ("Solaris_System1.0.mof")
#pragma Include ("Solaris_Device1.0.mof")
#pragma Include ("Solaris_Network1.0.mof")
#pragma Include ("Solaris_Users1.0.mof")
#pragma Include ("Solaris_Project1.0.mof")
#pragma Include ("Solaris_Event1.0.mof")
#pragma Include ("Solaris_CIMOM1.0.mof")
#pragma Include ("Solaris_SNMP1.0.mof")

// This must be the last include since it changes the CIM namespace
#pragma Include ("Solaris_Acl1.0.mof")
```

The compiler parses a line of the Solaris_Schema1.0.mof file, compiles the file specified in the Include statement, and then parses the next line of the Solaris_Schema1.0.mof file, until all included files are compiled.

The Solaris_CIMOM1.0.mof File

The Solaris_CIMOM1.0.mof file contains all the system properties used by the CIM Object Manager.

```
/*
Title    :            Solaris CIMOM mof specification
```

```
Description:    Models the system properties used by the CIMOM
Date:                05/09/00
Version:            1.0
Copyright (c) 2000 Sun Microsystems, Inc. All Rights Reserved.
*/
#pragma namespace ("root/system")

Qualifier Abstract : boolean = false,
            Scope(class, association, indication),
            Flavor(DisableOverride, Restricted);
Qualifier Association : boolean = false,
            Scope(class, association),
            Flavor(DisableOverride);
Qualifier Key : boolean = false,
            Scope(property, reference),
            Flavor(DisableOverride);
Qualifier Override : string = null,
            Scope(property, method, reference);
Qualifier Description : string = null,
             Scope(any),
             Flavor(Translatable);
Qualifier Expensive : boolean = false,
            Scope(property, reference, method, class, association);
Qualifier In : boolean = true,
            Scope(parameter);
Qualifier Provider : string = null,
            Scope(any);

[Provider("internal"),
 Description("Each instance becomes part of the classpath")
]
class Solaris_ProviderPath {
    [key]
    string pathurl;
};

[Provider("internal"),
 Description("This class represents the CIMOM")
]
class Solaris_CIMOM {
    [key]
    string name;

    [Description("Shuts down the CIMOM")]
    sint8 shutdown();
};
```

The Solaris_Core1.0.mof File

The Solaris_Core1.0.mof file is the first of the Solaris Schema files to be compiled after the Solaris_Schema1.0.mof file. This file provides the definition of the Solaris_ComputerSystem class of the Solaris Provider, and also the Solaris_LogRecord, Solaris_LogService, Solaris_LogServiceProperties, and Solaris_LogServiceSetting classes.

The Solaris_Application1.0.mof File

The Solaris_Application1.0.mof file enables you to set up packages and patches for your applications that extend the Solaris Schema.

The Solaris_Application1.0.mof file defines the following classes:

- Solaris_Package
- Solaris_Patch
- Solaris_SoftwareFeature
- Solaris_SoftwareElement
- Solaris_InstalledSoftwareElement

In addition, the Solaris_Application1.0.mof file defines the following association classes:

- Solaris_ProductSoftwareFeatures
- Solaris_SoftwareFeatureSoftwareElements
- Solaris_SoftwareFeatureDependency
- Solaris_SoftwareElementDependency
- Solaris_ProductSoftwareFeatureDependency
- Solaris_ProductSoftwareElementDependency
- Solaris_SoftwareFeatureSoftwareElementDependency
- Solaris_SoftwareElementProductDependency
- Solaris_SoftwareElementSoftwareFeatureDependency
- Solaris_ProductSoftwareElements
- Solaris_SoftwareFeatureParentChild
- Solaris_SoftwareFeatureProductDependency

The Solaris_System1.0.mof File

The Solaris_System1.0.mof file defines the following classes:

- Solaris_Process
- Solaris_OperatingSystem
- Solaris_InstalledOS
- Solaris_RunningOS
- Solaris_OSProcess
- Solaris_DataFile
- Solaris_LocalFileSystem
- Solaris_UFS
- Solaris_HSFS
- Solaris_NFS
- Solaris_Directory
- Solaris_ScheduledJob
- Solaris_ScheduledJob_Cron
- Solaris_JobScheduler
- Solaris_JobScheduler_Cron
- Solaris_CpuUtilizationInformation
- Solaris_CpuVminfo
- Solaris_CpuSysinfo
- Solaris_DiskIOInformation
- Solaris_DisklessClient
- Solaris_OsService
- Solaris_Eeprom
- Solaris_EepromSetting

In addition, the Solaris_System1.0.mof file defines the following association classes:

- Solaris_NFSExport
- Solaris_Mount
- Solaris_NFSMount
- Solaris_UFSMount
- Solaris_HSFSMount
- Solaris_OwningJobScheduler
- Solaris_HostedJobScheduler
- Solaris_SystemDevice
- Solaris_EepromElementSetting

The Solaris_Device1.0.mof File

The Solaris_Device1.0.mof file defines the following classes:

- Solaris_Processor
- Solaris_DiskDrive
- Solaris_DiskPartition
- Solaris_CpuUtilizationPerformanceMonitor
- Solaris_CpuVminfoPerformanceMonitor
- Solaris_CpuSysinfoPerformanceMonitor
- Solaris_DiskIOPerformanceMonitor
- Solaris_SerialPort
- Solaris_SerialPortConfiguration
- Solaris_SerialPortSetting
- Solaris_Environment
- Solaris_Printer
- Solaris_PrintJob
- Solaris_PrintService
- Solaris_PrintQueue
- Solaris_TimeZone
- Solaris_PrintSAP
- Solaris_EthernetAdapter
- Solaris_Keyboard
- Solaris_SoundDevice
- Solaris_MessageLog
- Solaris_MessageLogRecord
- Solaris_LogEntry
- Solaris_SyslogRecord
- Solaris_RecordInLog
- Solaris_LogInDataFile
- Solaris_LogServiceProperties
- Solaris_LogServiceSetting
- Solaris_MessageLogSetting

In addition, the Solaris_Device1.0.mof file defines these association classes:

- Solaris_MediaPresent
- Solaris_QueueForPrintService
- Solaris_OwningPrintQueue
- Solaris_PrinterServicingQueue
- Solaris_SystemTimeZone

The Solaris_Acl1.0.mof File

The Solaris_Acl1.0.mof file specifies the Solaris WBEM Services security classes. This file defines these base classes for access control lists, users, and namespaces:

- Solaris_Acl
- Solaris_UserAcl
- Solaris_NamespaceAcl

The Solaris_Network1.0.mof File

The Solaris_Network1.0.mof file defines classes that pertain to network domains, IP subnets, and naming services (including NIS, NIS+, LDAP, DNS, and server /etc files). This file defines these classes:

- Solaris_AdminDomain
- Solaris_SystemAdminDomain
- Solaris_NisAdminDomain
- Solaris_NisplusAdminDomain
- Solaris_LdapAdminDomain
- Solaris_DnsAdminDomain
- Solaris_IPProtocolEndpoint
- Solaris_IPSubnet

The Solaris_Users1.0.mof File

The Solaris_Users1.0.mof file defines these classes:

- Solaris_UserAccount
- Solaris_UserGroup
- Solaris_UserTemplate
- Solaris_UserHomeDirectory
- Solaris_AuthorizationAttribute
- Solaris_ExecutionProfile
- Solaris_ProfileAttribute
- Solaris_MailBox
- Solaris_EmailAlias

The Solaris_Event1.0.mof File

The Solaris_Event1.0.mof file defines one class Solaris_RMIDelivery, which is a subclass of CIM_IndicationHandler. This class is defined to facilitate the delivery of indications to management clients over Sun's implementation of the CIM RMI protocol. Applications that use this protocol create delivery end points by creating an instance of the Solaris_RMIDelivery class. At present, clients can subscribe for events as defined in the Distributed Management Task Force (DMTF) CIM Specification v2.5 Events Model (you must be a member of DMTF to access this specification). CIM_Events25.mof defines the event classes.

At present, this feature is available only to clients that connect over RMI, and not to clients that connect over Hypertext Transfer Protocol (HTTP).

The Solaris_SNMP1.0.mof File

The Solaris_SNMP1.0.mof file defines classes that pertain to configuration information for an SNMP device. The Solaris_SNMP1.0.mof file defines these classes:

- Solaris_SNMPSystem
- Solaris_SNMPSystemConf
- Solaris_SNMPGroupConf

The Solaris_LVM1.0.mof File

The Solaris_LVM1.0.mof file defines classes that pertain to storage devices, such as:

- State database replicas within a slice
- Range of extents within a storage extent that can be used for data
- Stripes
- Concatenated stripes
- Mirrors
- RAID Level 5 devices
- UFS logging file systems

- Spare pools
- Disk sets
- Storage volumes

The `Solaris_LVM1.0.mof` file defines these classes:

- `Solaris_VMStateDatabase`
- `Solaris_VMExtent`
- `Solaris_VMStripe`
- `Solaris_VMConcat`
- `Solaris_VMMirror`
- `Solaris_VMRaid5`
- `Solaris_VMTrans`
- `Solaris_VMHotSparePool`
- `Solaris_VMDiskSet`
- `Solaris_VMStorageVolume`

In addition, the `Solaris_LVM1.0.mof` file defines the following association classes:

- `Solaris_VMConcatComponent`
- `Solaris_VMDriveInDiskSet`
- `Solaris_VMExtentBasedOn`
- `Solaris_VMExtentInDiskSet`
- `Solaris_VMHostInDiskSet`
- `Solaris_VMHotSpareInUse`
- `Solaris_VMHotSpares`
- `Solaris_VMMirrorSubmirrors`
- `Solaris_VMRaid5Component`
- `Solaris_VMStatistics`
- `Solaris_VMStripeComponent`
- `Solaris_VMTransLog`
- `Solaris_VMTransMaster`
- `Solaris_VMUsesHotSparePool`
- `Solaris_VMVolumeBasedOn`

The `Solaris_Project1.0.mof` File

The `Solaris_Project1.0.mof` file defines classes that models the Solaris project database.

The `Solaris_Project1.0.mof` file defines the class `Solaris_Project`. In addition, the `Solaris_Project1.0.mof` file defines the following association classes:

- `Solaris_ProjectUser`

- Solaris_ProjectGroup

Glossary

This Glossary defines terms used in the Solaris WBEM Services documentation. Many of these terms are familiar to developers, but have new or altered meaning in the WBEM environment.

alias A symbolic reference in either a class or instance declaration to an object located elsewhere in a MOF file. Alias names follow the same rules as instance and class names. Aliases are typically used as shortcuts to lengthy paths.

aggregation relationship A relationship in which one entity is made up of the aggregation of some number of other entities.

association class A class that describes a relationship between two classes or between instances of two classes. The properties of an association class include pointers, or references, to the two classes or instances. All WBEM classes can be included in one or more associations.

Backus-Naur Form (BNF) A metalanguage that specifies the syntax of programming languages.

cardinality The number of values that may apply to an attribute for a given entity.

class A collection or set of objects that have similar properties and fulfill similar purposes.

CIM Object Manager Repository A central storage area managed by the Common Information Model Object Manager (CIM Object Manager). This repository contains the definitions of classes and instances that represent managed objects and the relationships among them.

CIM Schema A collection of class definitions used to represent managed objects that occur in every management environment.

See also core model, common model, and extension schema.

The CIM is divided into the metamodel and the standard schema. The metamodel describes what types of entities make up the schema. It

	also defines how these entities can be combined into objects that represent managed objects.
common model	The second layer of the CIM schema, which includes a series of domain-specific but platform-independent classes. The domains are systems, networks, applications, and other management-related data. The common model is derived from the core model. *See also* extension schema.
core model	The first layer of the CIM schema, which includes the top-level classes and their properties and associations. The core model is both domain- and platform-independent. *See also* common model and extension schema.
Distributed Management Task Force (DMTF)	An industry-wide consortium committed to making personal computers easier to use, understand, configure, and manage.
domain	The class to which a property or method belongs. For example, if status is a property of `LogicalDevice`, it is said to belong to the `LogicalDevice` domain.
dynamic class	A class whose definition is supplied by a provider at runtime as needed. Dynamic classes are used to represent provider-specific managed objects and are not stored permanently in the CIM Object Manager Repository. Instead, the provider responsible for a dynamic class stores information about its location. When an application requests a dynamic class, the CIM Object Manager locates the provider and forwards the request. Dynamic classes support only dynamic instances.
dynamic instances	An instance that is supplied by a provider when the need arises and is not stored in the CIM Object Manager Repository. Dynamic instances can be provided for either static or dynamic classes. Supporting instances of a class dynamically allows a provider to always supply up-to-the-minute property values.
enumeration	Java term for getting a list of objects. Java provides an `Enumeration` interface that has methods for enumerating a list of objects. An individual object on this list to be enumerated is called an element.
extension schema	The third layer of the CIM Schema, which includes platform-specific extensions of the CIM Schema such as Solaris and UNIX. *See also* common model and core model.
flavor	*See* qualifier flavor.
indication	An operation executed as a result of some action such as the creation, modification, or deletion of an instance, access to an instance, or

	modification or access to a property. Indications can also result from the passage of a specified period of time. An indication typically results in an event.
inheritance	The relationship that describes how classes and instances are derived from parent classes or superclasses. A class can spawn a new subclass, also called a child class. A subclass contains all the methods and properties of its parent class. Inheritance is one of the features that allows WBEM classes to function as templates for actual managed objects in the WBEM environment.
instance	A representation of a managed object that belongs to a particular class, or a particular occurrence of an event. Instances contain actual data.
instance provider	A type of provider that supports instances of system- and property-specific classes. Instance providers can support data retrieval, modification, deletion, and enumeration. Instance providers can also invoke methods. *See also* property provider.
interface class	The class used to access a set of objects. The interface class can be an abstract class representing the scope of an enumeration. *See also* enumeration and scope.
Interface Definition Language (IDL)	A generic term for a language that lets a program or object written in one language communicate with another program written in an unknown language.
key	A property that is used to provide a unique identifier for an instance of a class. Key properties are marked with the Key qualifier.
key qualifier	A qualifier that must be attached to every property in a class that serves as part of the key for that class.
managed object	A hardware or software component that is represented as a WBEM class. Information about managed objects is supplied by data and event providers as well as the CIM Object Manager Repository.
Managed Object Format (MOF)	A compiled language for defining classes and instances. The MOF compiler (mofcomp) compiles `.mof` text files into Java classes and adds the data to the CIM Object Manager Repository. MOF eliminates the need to write code, thus providing a simple and fast technique for modifying the CIM Object Manager Repository.
management application	An application or service that uses information originating from one or more managed objects in a managed environment. Management applications retrieve this information through calls to the CIM Object Manager API from the CIM Object Manager and from providers.
management information base	A database of managed objects.

metamodel A CIM component that describes the entities and relationships representing managed objects. For example, classes, instances, and associations are included in the metamodel.

metaschema A formal definition of the Common Information Model, which defines the terms used to express the model, its usage, and its semantics.

method A function describing the behavior of a class. Including a method in a class does not guarantee an implementation of the method.

MOF file A text file that contains definitions of classes and instances using the Managed Object Format (MOF) language.

Named Element An entity that can be expressed as an object in the metaschema.

namespace A directory-like structure that can contain classes, instances, and other namespaces.

object path A formatted string used to access namespaces, classes, and instances. Each object on the system has a unique path which identifies it locally or over the network. Object paths are conceptually similar to Universal Resource Locators (URLs).

override Indicates that the property, method, or reference in the derived class overrides the similar construct in the parent class in the inheritance tree or in the specified parent class.

polymorphism The ability to alter methods and properties in a derived class without changing their names or altering interfaces. For example, a subclass can redefine the implementation of a method or property inherited from its superclass. The property or method is thereby redefined even if the superclass is used as the interface class.

Thus, the `LogicalDevice` class can define the variable status as a string, and can return the values "on" or "off." The Modem subclass of `LogicalDevice` can redefine (override) status by returning "on," "off," and "connected." If all `LogicalDevice` classes are enumerated, any `LogicalDevice` that represents a modem can return the value "connected" for the status property.

property A value used to characterize the instances of a class. Property names cannot begin with a digit and cannot contain white space. Property values must have a valid Managed Object Format (MOF) data type.

property provider A program that communicates with managed objects to access data and event notifications from a variety of sources, such as the Solaris operating environment or a Simple Network Management Protocol (SNMP) device. Providers forward this information to the CIM Object Manager for integration and interpretation.

qualifier A modifier containing information that describes a class, an instance, a property, a method, or a parameter. The three categories of qualifiers

	are: those defined by the Common Information Model (CIM), those defined by WBEM (standard qualifiers), and those defined by developers. Standard qualifiers are attached automatically by the CIM Object Manager.
qualifier flavor	An attribute of a CIM qualifier that governs the use of a qualifier. WBEM flavors describe rules that specify whether a qualifier can be propagated to derived classes and instances and whether or not a derived class or instance can override the qualifier's original value.
range	A class that is referenced by a reference property.
reference	A special string property type that is marked with the reference qualifier, indicating that it is a pointer to other instances.
required property	A property that must have a value.
schema	A collection of class definitions that describe managed objects in a particular environment.
scope	An attribute of a CIM qualifier that indicates which CIM elements can use the qualifier. Scope can only be defined in the Qualifier Type declaration; it cannot be changed in a qualifier.
selective inheritance	The ability of a descendant class to drop or override the properties of an ancestral class.
Simple Network Management Protocol (SNMP)	A protocol of the Internet reference model used for network management.
singleton class	A WBEM class that supports only a single instance.
Solaris Schema	A Sun extension to the CIM Schema that contains definitions of classes and instances to represent managed objects that exist in a typical Solaris operating environment.
standard schema	A common conceptual framework for organizing and relating the various classes representing the current operational state of a system, network, or application. The standard schema is defined by the Distributed Management Task Force (DMTF) in the Common Information Model (CIM).
static class	A WBEM class whose definition is persistent. The definition is stored in the CIM Object Manager Repository until it is explicitly deleted. The CIM Object Manager can provide definitions of static classes without the help of a provider. Static classes can support either static or dynamic instances.
static instance	An instance that is persistently stored in the CIM Object Manager Repository.

subclass	A class that is derived from a superclass. The subclass inherits all features of its superclass, but can add new features or redefine existing ones.
subschema	A part of a schema owned by a particular organization. The Win32 and Solaris Schema are examples of subschema.
superclass	The class from which a subclass inherits.
transitive dependency	In a relation having at least three attributes R (A, B, C), the situation in which A determines B, B determines C, but B does not determine A.
trigger	A recognition of a state change (such as create, delete, update, or access) of a class instance, and update or access of a property. The WBEM implementation does not have an explicit object representing a trigger. Triggers are implied either by the operations on basic objects of the system (create, delete, and modify on classes, instances and namespaces) or by events in the managed environment.
Unified Modeling Language (UML)	A notation language used to express a software system using boxes and lines to represent objects and relationships.
Unicode	A 16-bit character set capable of encoding all known characters and used as a worldwide character-encoding standard.
UTF-8	An 8-bit transformation format that may also serve as a transformation format for Unicode character data.
virtual function table (VTBL)	A table of function pointers, such as an implementation of a class. The pointers in the VTBL point to the members of the interfaces that an object supports.
Win32 Schema	A Microsoft extension to the CIM Schema that contains definitions of classes and instances to represent managed objects that exist in a typical Win32 environment.

Index