HP OpenView Service Information Portal 2.0

Administrator Guide

Windows NT®, Windows® 2000, HP-UX, and Solaris

Manufacturing Part Number: J4797-90001

April 2001

© Copyright 2001 Hewlett-Packard Company.

Legal Notices

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend. All rights are reserved. No part of this document may be copied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this material is subject to change without notice.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY

3404 E. Harmony Road

Fort Collins, CO 80528 U.S.A.

Use of this manual and flexible disk(s), tape cartridge(s), or CD-ROM(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Copyright Notices. ©Copyright 1983-2001 Hewlett-Packard Company, all rights reserved.

Reproduction, adaptation, or translation of this material without prior written permission is prohibited, except as allowed under the copyright laws.

Trademark Notices.

Adobe® is a trademark of Adobe Systems Incorporated.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Intel486 is a U.S. trademark of Intel Corporation.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Netscape™ and Netscape Navigator™ are U.S. trademarks of Netscape Communications Corporation.

OpenView® is a registered U.S. trademark of Hewlett-Packard Company.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

Oracle Reports™, Oracle7™, and Oracle7 Server™ are trademarks of Oracle Corporation, Redwood City, California.

OSF/Motif® and Open Software Foundation® are trademarks of Open Software Foundation in the U.S. and other countries.

Pentium® is a U.S. registered trademark of Intel Corporation.

SQL*Net® and SQL*Plus® are registered U.S. trademarks of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of the Open Group.

Windows NT® is a U.S. registered trademark of Microsoft Corporation.

Windows® 2000 is a U.S. registered trademark of Microsoft Corporation.

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Contents

Conventions

The following typographical conventions are used in this manual.

Font	What the Font Represents	Example
Italic	For book or manual titles, and for manpage names.	Refer to the *OVW Developer's Guide*.
	To provide emphasis.	You *must* follow these steps.
	To specify a variable that you must supply when entering a command.	At the prompt type: **rlogin** *your_name* where you supply your login name.
Bold	For glossary terms.	The **distinguishing attribute** of this class...
Computer	Text and items on the computer screen.	The Root map window ... The system replies: Press Enter
	Command names	Use the grep command ...
	File and directory names.	/usr/bin/X11
	Process names.	Check to see if pmd is running.
	Window/dialog box names	In the IP Internet map window...
Computer Bold	Text that you must enter.	At the prompt, type: **ovstatus**.
Keycap	Keyboard keys.	Press **Return**.
[Button]	Buttons on the user interface.	Click [NET]. Click on the [Apply] button.
Menu Items	A menu name followed by a colon (:) means that you select the menu, then the item. When the item is followed by an arrow (->), a cascading menu follows.	Select Edit:Find->Objects by Comment

Contact Information

Technical Support and Training

Technical support and training information can be found on the HP OpenView World Wide Web site at:

`http://openview.hp.com/`

Documentation Feedback

Your comments on and suggestions for the documentation help us understand your needs and better meet them.

You can provide feedback about documentation:

- via e-mail to: `ovdoc@fc.hp.com`, or
- via the HP documentation site at: `http://www.docs.hp.com`

If you encounter *serious errors* in the documentation that impair your ability to use the product, please contact the HP Response Center or your support representative so that your feedback can be entered into CHARTS (the HP Change Request Tracking System).

1 Introduction

What This Manual Covers

This manual covers several areas of SIP deployment. It will guide you through the following tasks:

1. Learning about SIP's Configuration Files.
2. Implementing a supplied or custom authentication provider.
3. Implementing a customer model—a mapping of customers to their resources.
4. Learning about SIP's customer data filtering approach.
5. Deciding on an approach to portal development and deployment that suits your needs.
6. Designing a custom look and feel to your portals.
7. Integrating your own applications and data.
8. Deploying your customer portals.

Getting Additional Documentation

Both printable and online documentation is available. The table below lists printable documents available to you after SIP 2.0 is installed. All document files are stored in product directories under the following directory:

Windows NT/2000: `<install_dir>\htdocs\C\manuals\`
UNIX: `/opt/OV/SIP/htdocs/C/manuals/`

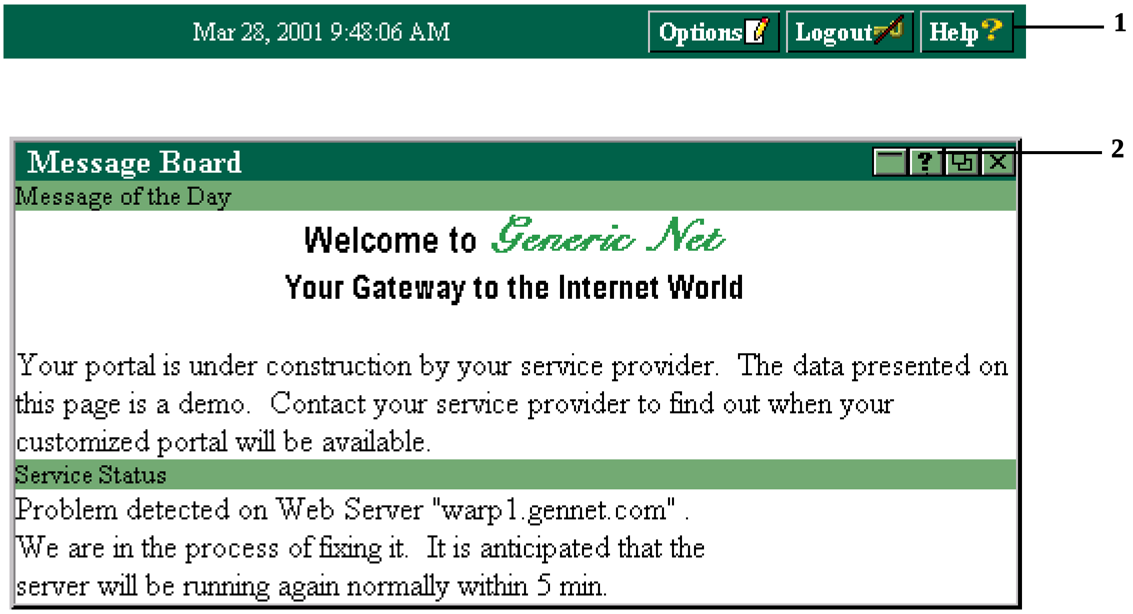
Table 1-1 **HP OpenView Service Information Portal 2.0 Documentation**

Document Title	**Filename**
Getting Started	`Getting_Started.pdf`
Administrator Guide	`Administrator_Guide.pdf`
Configuring NNM	`Configuring_NNM.pdf`
Presenting NNM Data	`Presenting_NNM_Data.pdf`
Configuring VantagePoint Navigator	`Configuring_VPN.pdf`
Presenting VantagePoint Navigator Data	`Presenting_VPN_Data.pdf`
Configuring VP-IS and Presenting Data	`Configuring_and_Presenting_VPIS.pdf`

Online help is available when you need instruction on the SIP user interface and how to use it. The list of help topics available from the [`Help`] button on the main portal page differs depending upon the editing permissions—ViewAdmin, UserPreferences, or ReadOnly—assigned to the current role.

Below are sample screens that point out the various online help options.

Figure 1-1 Online Help Available from the Main Portal Page



Message Board

Message of the Day

Welcome to *Generic Net*

Your Gateway to the Internet World

Your portal is under construction by your service provider. The data presented on this page is a demo. Contact your service provider to find out when your customized portal will be available.

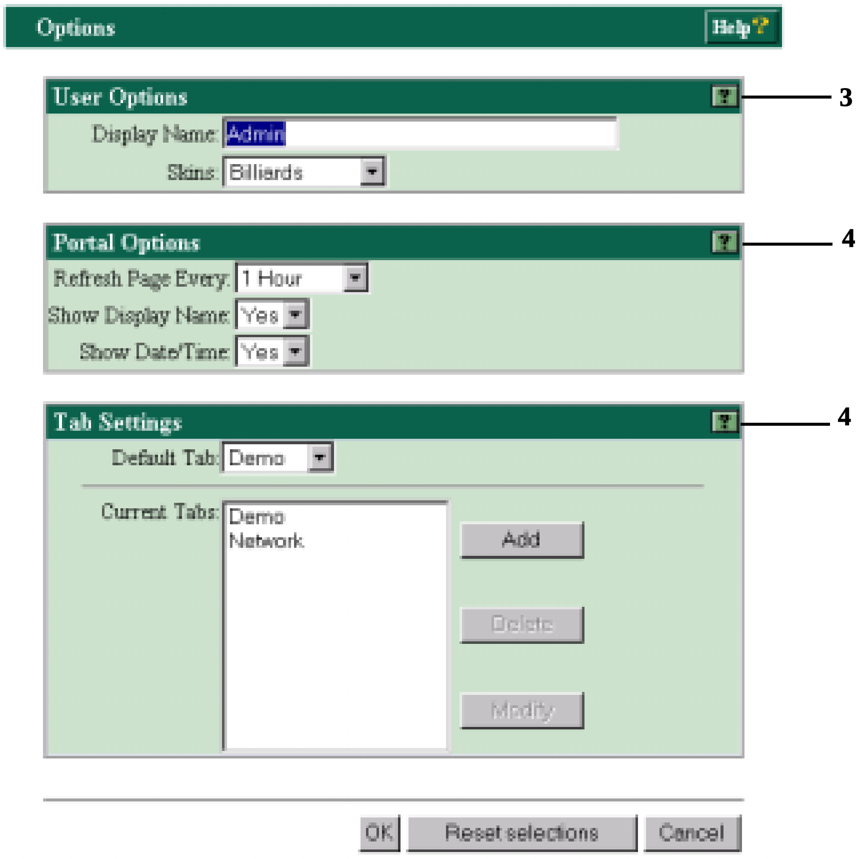
Service Status

Problem detected on Web Server "warp1.gennet.com" .
We are in the process of fixing it. It is anticipated that the server will be running again normally within 5 min.

1 - The `[Help]` button displays help on performing tasks with the user interface. The list of displayed help topics differs depending upon the editing permissions—ViewAdmin, UserPreferences, or ReadOnly—assigned to the current role. The tasks described do not include module-specific information.

2 - The `[?]` button on a module titlebar displays module help. These topics describe the module and data being displayed and are targeted to your end customers. A help topic for each module is provided by default with the SIP, but you can write your own help topic for a module and override the default topic.

Figure 1-2 Online Help Available from the Options Page

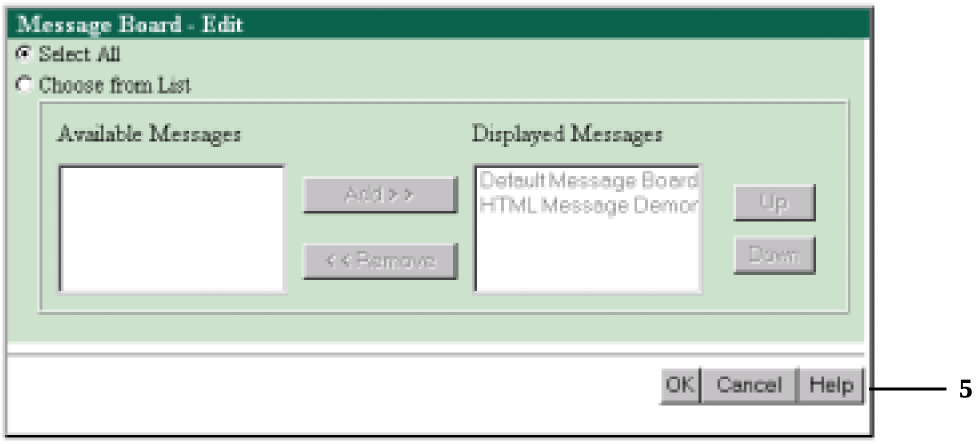


Display Name: Admin
Skins: Billiards
Refresh Page Every: 1 Hour
Show Display Name: Yes
Show Date/Time: Yes
Default Tab: Demo
Current Tabs: Demo
Network

3 - The `User Options` section and [?] button are visible when the editing permissions level is set to UserPreferences or ViewAdmin for the current role. If a user has ReadOnly permissions, the `[Options]` button and page are not viewable.

4 - The `Portal Options` and `Tab Settings` sections and associated [?] buttons are only visible when the editing permissions level is set to ViewAdmin for the current role.

Figure 1-3 **Online Help Available from Module Edit Pages**



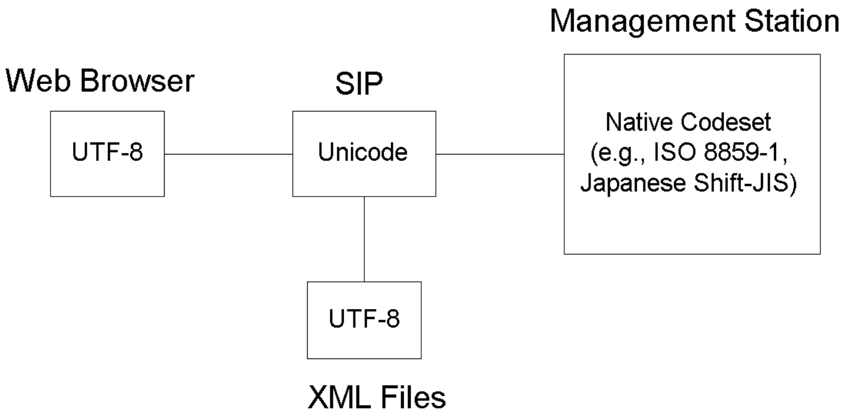
5 - The `[Help]` button on module Edit pages displays help on performing module editing tasks through the user interface.

2 Internationalization (I18N) and SIP

I18N Model for Service Information Portal

Service Information Portal uses UTF-8 when communicating with the web browser and interpreting data in XML files. When getting data from management stations, SIP converts from the native codeset (such as ShiftJIS) used by the management station to Unicode.

Figure 2-1 **I18N Model**



What You Can Expect From SIP

Service Information Portal:

- Installs under any language on any language system (such as Japanese NT).
- Uses locale-specific formats for date and time.
- Does locale-specific string comparison.
- Supports one Web browser codeset—UTF-8.
- Displays localized data from NNM (map data), Customer Views (customer names), VPIS, and VPN.
- Supports non-ASCII string data specific to SIP as long as it is in the UTF-8 codeset. This includes:
    - — Module titles specified in the module registration XML files and the portal view files
    - — Tab names
    - — User and role display names
    - — Network Device Health gauge titles
    - — Alarm category names
    - — Generic module output data
    - — Message Board message text
    - — Bookmark link names
    - — Customer names in the “simple customer model” XML file
    - — Help files
- Supports ASCII string data only for:
    - — File and directory names
    - — Login names
    - — URLs

Limitations

- SIP is not localized to a specific language (such as Japanese), hence, different strings are not displayed based on the locale; however, SIP can display non-ASCII data.
- SIP does not ship the Java converters necessary to convert non-UTF-8 codeset data into UTF-8 codeset. SIP assumes that these will be part of the JRE on each platform.
- SIP has only been tested in a “mono-lingual” environment where all management servers were in the same locale.

Running SIP in Non-English Language Mode

Following are required configuration tasks that help you prepare to run SIP on a system and browser that have been configured to operate in non-English language mode.

Configuring the Browser Settings

SIP supports display of characters in UTF-8 code set to the Web browser. Make sure your web browser is either in the "Auto-Select" or "UTF-8" modes to ensure that the data is properly displayed.

If the Web browser has "Auto-Select" enabled for Encodings, the UTF-8 data should be displayed correctly through the portal. If not, you will need to set the browser's Encoding to "Unicode (UTF-8)".

Configuring XML Files Used By SIP

If non-ASCII characters are added to XML, XSL, or JSP files, you must preserve the UTF-8 codeset for these characters.

NOTE The Microsoft Windows 2000 Notepad editor allows files to be saved in UTF-8 code set. Microsoft Windows NT does not supply an editor that can edit UTF-8 code set. On many UNIX platforms, the `iconv` command can be used to translate from ShiftJIS (or any other code set) into UTF-8 to make editing in a non-UTF-8 code set simpler.

Configuring SIP to Access NNM Data

Completion of the following configuration tasks allows SIP to access non-ASCII data from NNM. These instructions assume that you have completed the steps in the `Configuring_NNM.pdf` file and the Presenting_NNM_Data.pdf.

Configuring the Topology Module to Access Non-English NNM Data

If an NNM management station (from which the NNM Topology Map module gathers information) is running in a language that uses non-ASCII characters, the `encoding` attribute in the `nmConfig.xml` file (on the SIP server) must specify the code set for the JDK Converter to use. If no code set is configured, no conversion is done.

NNM map data is translated into UTF-8 characters by the portal.

The Java JDK Converter 1.1 can be configured on a per-NNM-management-station basis to determine how to interpret the incoming code set.

1. On the NNM management station, open the following file:

   Windows NT/2000: `<`*`NNM_install_dir`*`>\www\conf\locales.jconv`

   UNIX: /etc/opt/OV/share/www/conf/locales.jconv

   This file contains a table of JDK Converter 1.1 values for all languages.

   Locate the appropriate value for the NNM management station. Find the OS locale in which NNM is running (for example on Windows NT, "Japanese_Japan.932") and the corresponding JDK 1.1 Converter (for example, "SJIS"). In this case, in the next step, you would enter:

   `encoding=SJIS`

2. On the SIP server, open the following file and enter the appropriate `encoding` attribute for the `NNMStation` element. See the `nmConfig.dtd` file if you need more information:

   Windows NT/2000:
   `<`*`install_dir`*`>\conf\share\modules\NM\nmConfig.xml`

   UNIX: `/etc/opt/OV/SIP/conf/share/modules/NM/nmConfig.xml`

3. After changing the `nmConfig.xml` file, you must stop and restart the servlet engine before changes take effect:

   WindowsNT/2000:
   From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**.

   UNIX:

   As `root`, stop and restart the web server and servlet engine by running the following. (The `DISPLAY` variable must be configured prior to restarting the webserver and servlet engine.)

Stop on HP-UX: **`/sbin/init.d/ovsip stop`**
Start on HP-UX: **`/sbin/init.d/ovsip start`**

Stop on Solaris: **`/etc/init.d/ovsip stop`**
Start on Solaris: **`/etc/init.d/ovsip start`**

4. To make the localized submap information visible in your portal views:

   a. In the SIP portal, open each `*PortalView*.xml` file that you have created.

   b. Access one instance of the NNM Topology Map module.

   c. Click on the edit button to display the submap selection dialog.

   d. Follow the directions for selecting submaps (click `[Help]` for more information or see the `Presenting_NNM_Data.pdf file`).

      You will now see the localized submap names in the list. Select the ones that you want.

   e. Repeat this procedure until you have modified all NNM Topology Map module instances in all `PortalView.xml` files.

      If desired, update the information in the default NNM Topology Map module as well (`OVDefaultTopology.xml`).

Configuring the Alarms Module to Access Non-English NNM Data

Within each SIP alarm category definition file (`*NmAlarmCat*.xml`), the `NNMBaseCategory` attribute must *exactly* match NNM's alarm category strings in the `trapd.conf` file. Therefore, if an NNM management station (from which the NNM Alarms module gathers information) is running in a language that uses non-ASCII characters, you must provide a set of `*NmAlarmCat*.xml` files for the localized alarm categories.

SIP requires that the non-ASCII characters be in the UTF-8 code set. NNM, itself, uses the traditional OS code set (for example, Shift-JIS, EUC, or ISO 88591) and does not support UTF-8 nor Unicode code sets.

You can copy the NNM alarm category strings from NNM's `trapd.conf` file and paste them into the desired `*NmAlarmCat*.xml` files, provided you follow these directions to ensure that the `*NmAlarmCat*.xml` files are saved in UTF-8 code set.

NOTE If you are displaying alarms in SIP from NNM management stations running in multiple languages, you need to provide multiple sets of the `NmAlarmCat.xml` files. If all the NNM management stations providing information to SIP run under the same language, you only need one set of `NmAlarmCat.xml` files.

1. Open the following two files in an editor that is running under the code set that is in use on the NNM management station and is capable of converting or saving a file in UTF-8 code set (if you are trying to access alarms from an NNM management station running in Japanese Shift-JIS, see "Contrib version of the standard NNM alarm categories in Shift-JIS" on page 34):
   - On the NNM management station:
     - *Windows NT/2000:* `<NNM_install_dir>\conf\$LANG\trapd.conf`
     - *UNIX:* `/etc/opt/OV/share/conf/$LANG/trapd.conf`
   - On the SIP server, open the `NmAlarmCat.xml` file you wish to modify:
     - *Windows NT/2000:* `<SIP_install_dir>\conf\share\modules\alarms\*`
     - *UNIX:* `/etc/opt/OV/SIP/conf/share/modules/alarms/*`

TIP *UNIX:* `vi` is capable of opening a file in any code set. After editing a file using `vi` (in a code set other than UTF-8), UNIX has a command called `iconv` that converts most code sets into UTF-8.

Windows 2000: Notepad runs under various language settings by changing Windows 2000's Regional Options, Locale setting. Notepad provides a `save-as` to the UTF-8 code set.

Windows NT: No known tools to accomplish this task.

2. In the `trapd.conf` file, locate the CATEGORY settings, for example:

```
CATEGORY 2 "Error Alarms" "LOCALIZED-STRING-FOR-Error Alarms"
CATEGORY 3 "Threshold Alarms" "LOCALIZED-STRING-FOR-Threshold Alarms"
CATEGORY 4 "Status Alarms" "LOCALIZED-STRING-FOR-Status Alarms"
CATEGORY 5 "Configuration Alarms" "LOCALIZED-STRING-FOR-Configuration Alarms"
CATEGORY 6 "Application Alert Alarms" "LOCALIZED-STRING-FOR-Appl Alert Alarms"
```

3. Copy the string from the second set of quotes, and paste it into the `NNMBaseCategory` attribute in the *NmAlarmCat*`.xml` file. For example:

```
<?xml version='1.0' encoding='utf-8' standalone='no' ?>
<!DOCTYPE AlarmCategoryDef SYSTEM "NmAlarmCat.dtd">

<AlarmCategoryDef DisplayTitle="Error Alarms"
                  NNMBaseCategory="LOCALIZED-STRING-FOR-Error Alarms">
    <Severities critical="1"
                major="1"
                minor="1"
                warning="1"
                normal="1"/>
    <Acknowledgement acknowledged="1"
                     unacknowledged="1"/>
</AlarmCategoryDef>
```

4. If you want the SIP alarm category title to be localized as well, modify the `DisplayTitle` string in the same manner.
5. Save or convert the *NmAlarmCat*`.xml` file in the UTF-8 code set. Place it in the same directory as the other *NmAlarmCat*`.xml` files.
6. Open the `NmAlarmCatsIndex.xml` file and add your new *NmAlarmCat*`.xml` file name to the list.
7. After making changes in the `../conf/share/modules/alarms` directory, you must stop and restart the SIP servlet engine before changes take effect:

   WindowsNT/2000:
   From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**.

   UNIX:

   As `root`, stop and restart the web server and servlet engine by running the following. (The `DISPLAY` variable must be configured prior to restarting the webserver and servlet engine.)

Stop on HP-UX: **`/sbin/init.d/ovsip stop`**
Start on HP-UX: **`/sbin/init.d/ovsip start`**

Stop on Solaris: **`/etc/init.d/ovsip stop`**
Start on Solaris: **`/etc/init.d/ovsip start`**

8. You must now open any `PortalView.xml` files and the default Topology module (`OVDefaultTopology.xml`) file and edit the Alarms module instances to point to the new localized alarm category names.

Contrib version of the standard NNM alarm categories in Shift-JIS A set of *`NmAlarmCat`*`.xml` files already set up with the standard NNM alarm categories is provided in the `SIP/contrib` directory. You will also find a file named `NmAlarmCatsIndex-ja.xml` from which you can copy and paste the new alarm categories into your `NmAlarmCatsIndex.xml` file. Follow the directions in the previous section to use these files.

Configuring SIP to Access Customer Views Data

The supported method for setting the locale used by `getcvdata.exe`, which returns Customer Views data, is to pass it through CGI parameters: `AcceptLang=<web_locale>&Developer`

Both “`AcceptLang`” and “`Developer`” need to be set. If “`Developer`” is not set, the “`AcceptLang`” value will not be used. The value of `AcceptLang` is a Web locale. This is translated into an OS locale on the NNM server using the NNM locale mapping table in:
`.../www/conf/locales.mapping`.

Here is an example invocation of getcvdata.exe in `.../conf/framework/OVPortalConfig.xml`. On HP-UX 11, the `AcceptLang` value of, for example, “`ja`” would translate into the locale “`ja_JP.SJIS`” using the default `locales.mapping` file:

```
<CustomerModelSources>
 <CustomerModel href="http://host/OvCgi/getcvdata.exe?AcceptLang=ja&Developer"/>
</CustomerModelSources>
```

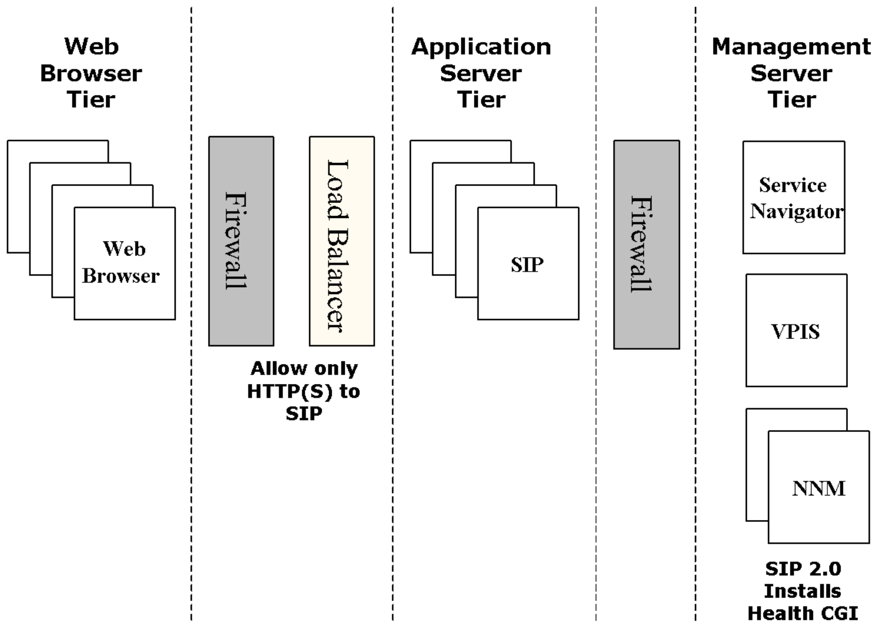
3 Deploying SIP in a Distributed Environment

SIP Distribution Model

The SIP distribution model is a three-tiered model:

- Web Browser Tier
- Application Server Tier
- Management Server Tier

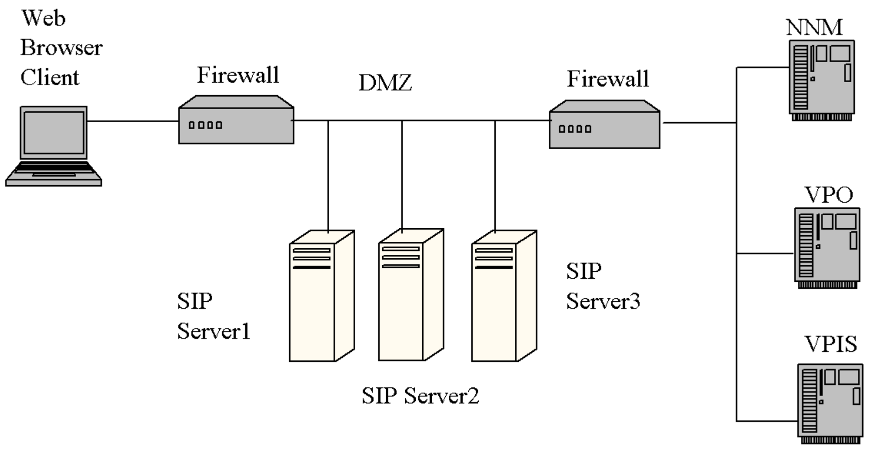
Figure 3-1 SIP Three-Tiered Distribution Model



The web browser to SIP server communication can go through a firewall and only requires HTTP or HTTPS.

The SIP server to management server communication can also go through a firewall. The ports that need to be opened through the firewall depend on the specific management products used. Figure 3-2 on page 37 shows a distribution model for firewalls.

Figure 3-2 Distribution Model (Firewalls)



Distributed and Shared Configuration Files

Multiple SIP servers can share portal configuration, using HTTP or HTTPS to distribute and share SIP configuration. This makes it easier to maintain multiple SIP servers, because configuration changes only need to be made in one place instead of on each SIP server. The SIP configuration server can be in the DMZ or behind another firewall.

Service Information Portal provides a mechanism for configuring remote access to these configuration files through the `SIPPath.properties` file. User roles, user preferences, portal views, and module configuration files can be shared or distributed across the network.

The only entries in `SIPPath.properties` that can be configured for remote access via HTTP or HTTPS are `SIP_SHARE_CONF_DIR` and `SIP_PASSWD_FILE`. The other directories must be local directories on the SIP server.

For more information on the `SIPPath.properties` file, see “Distributed and Shared Configuration Files” on page 52.

The `SIPPath.properties` file is located in the following directory:

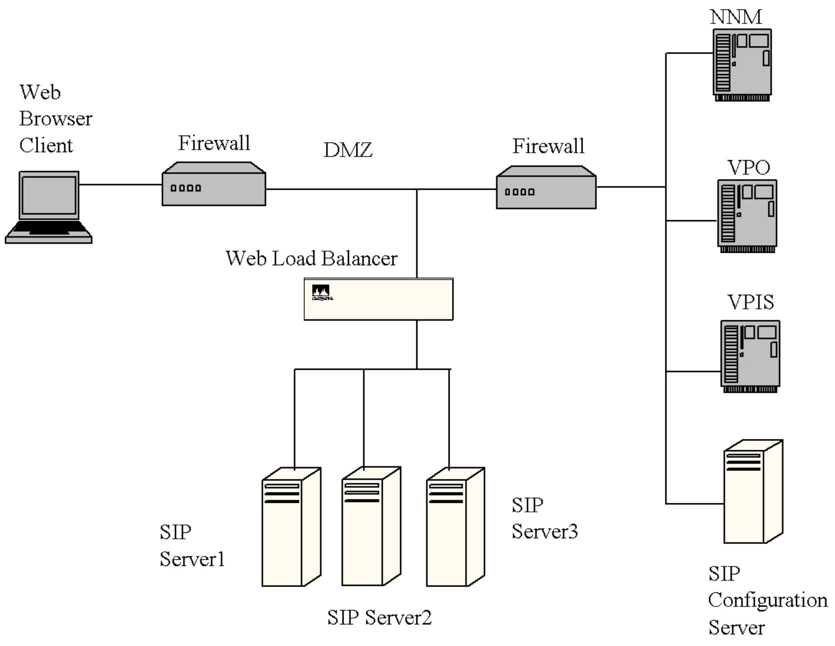
Windows NT/2000:
`<`*`install_dir`*`>\webapps\ovportal\SIPPath.properties`

UNIX: `/opt/OV/SIP/webapps/ovportal/SIPPath.properties`

Scalability Through Multiple Shared Servers

Scalability can be achieved through the use of multiple SIP servers. This can be done using a Web server load balancer that supports web session affinity. In this type of configuration, you will likely want to use shared SIP configuration to make it easier to manage the environment.

Figure 3-3 Distribution Model (Shared Configuration)



Whitepaper on Setting Up SIP in a Distributed Environment

For a whitepaper on "Configuring Service Information Portal (SIP) to Work Within a Distributed Secure Environment," refer to `http://www.openview.hp.com`.

4 Understanding SIP Configuration Files and XML

What You Need to Know About SIP Configuration Files and XML

SIP uses XML as the means of storing and retrieving all configuration information. XML is used to represent configuration data on the Service Information Portal server. The key thing to know is that XML is not being sent to the web browser over the wire. It is being used to store configuration files on the file system.

This chapter describes XML in general terms, and explains how HP OpenView Service Information Portal uses it for various types of configuration information.

The following sections give you a general understanding of XML. If you are already familiar with XML and do not need an overview, go to “Use of XML for SIP Configuration Files” on page 45.

Overview of XML

XML refers to eXtensible Markup Language, which is a language for creating vocabularies (or markup languages) to describe any kind of data that you want to structure.

XML is similar to HTML in that it uses tagged elements. However, HTML allows you to describe presentation—the way that text appears when displayed in a web browser—while XML allows you to describe the semantic meaning of data.

Therefore, XML is an enabling technology for creating vocabularies that represent data that you can easily share with others. A significant number of XML vocabularies are being written and used today, such as domain-specific markup languages for Math, Music, and Chemistry. New technologies are being developed that understand the structure of XML vocabularies in order to operate on the data.

XML is human readable yet structured enough that programs can read, parse, and use it. XML is an open standard, meaning that it is not owned by any particular company. There are many implementations of XML parsers and supporting tools.

- XML is a language for creating your own markup languages that share the same basic structure. It is a “meta-markup language.”
- XML allows you to create vocabularies (or markup languages) that are syntactically very similar to HTML.
- XML vocabularies are self-describing in that they allow others to easily understand the structure of your data.
- XML is an open standard of the World Wide Web Consortium. Visit the W3C site for more information: `http://www.w3.org/`

Why Use XML?

- XML allows you to separate the data from how it is presented.
  - — XML markup describes a document’s structure and meaning, not how it is formatted or displayed.
- You can use a variety of techniques to process a specific XML vocabulary and yield HTML to be displayed.

- You can transmit XML in the same way as HTML (i.e. via HTTP) to share data (although SIP does not use XML in this way).
- XML defines the data, its syntax and structure, not the presentation of the data. Implicit in the tags is the meaning of the documents.
- XML is a web technology; like HTML, communication occurs over HTTP (although, again, SIP does not use XML in this way).

Sample XML Vocabulary to Describe Book Inventory

XML is a technology for creating languages or vocabularies to easily represent specific kinds of data. Below is an example of an XML document that describes book inventory data.

```
<?xml version="1.0"?>
<BookCatalog>
        <Book category="reference">
                <Author> Nigel Rees</Author>
                <Title>Sayings of the Century</Title>
                <Price>8.95</Price>
        </Book>
        <Book category="fiction">
                <Author>Evelyn Waugh</Author>
                <Title>Sword of Honour</Title>
                <Price>12.99</Price>
        </Book>
</BookCatalog>
```

Document Type Definitions (DTDs)

DTDs describe the syntax for a language. An XML parser looks at the DTD and applies it to the XML that is associated with the DTD to make sure the XML conforms to it. Unlike HTML, XML is very strict about enforcing valid syntax.

- Formal and precise definition of an XML vocabulary (or markup language)
- Can be used by an XML parser to validate that an XML document is

not only well-formed XML but also follows the syntactic rules of the vocabulary.

- DTD is not XML. Instead it is extended BNF (Backus-Naur Form, or EBNF).

DTD for Book Inventory XML Vocabulary

```
<!ELEMENT BookCatalog (Book*)>
<!ELEMENT Book (Author,Title,Price)>
<!ATTLIST Book
   category CDATA #REQUIRED >
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
```

Book Inventory XML Document That References DTD

Referring to the second line of the XML below, note the way that DTDs are referenced from the XML document.

```
<?xml version="1.0"?>
<!DOCTYPE BookCatalog SYSTEM "BookCatalog.dtd">
<BookCatalog>
        <Book category="reference">
                <Author> Nigel Rees</Author>
                <Title>Sayings of the Century</Title>
                <Price>8.95</Price>
        </Book>
        <Book category="fiction">
                <Author>Evelyn Waugh</Author>
                <Title>Sword of Honour</Title>
                <Price>12.99</Price>
        </Book>
</BookCatalog>
```

How Is XML Data Processed?

Two standard mechanisms can be used to process XML data: DOM and SAX.

- DOM (Document Object Model): a tree-based representation of an XML document created by an XML parser.
- SAX (Simple API for XML): Event-driven API provided by XML parser that notifies you as it processes the XML.

Use of XML for SIP Configuration Files

SIP configuration files can be categorized according to nine configuration types. Table 4-1 lists the types of configurations and the associated file or directory.

You can also understand the configuration files from a visual standpoint by looking at the SIP directory structure diagrams. To view the diagrams, you have two options:

- Log into SIP as "admin" and switch to the "Welcome" role. There you will find tabs for `Windows Directories` and `UNIX Directories`. Each tab contains several diagrams.
- Alternatively, view the following GIF files:

  Windows NT/2000:
  `<`*`install_dir`*`>\htdocs\C\help\SIP\:`
  `NTstructureSIP2a.gif`
  `NTstructureSIP2b.gif`
  `NTstructureSIP2c.gif`

  UNIX: `/opt/OV/SIP/htdocs/C/help/SIP/:`
  `UXstructureSIP2a.gif`
  `UXstructureSIP2b.gif`
  `UXstructureSIP2c.gif`
  `UXstructureSIP2d.gif`
  `UXstructureSIP2e.gif`

Table 4-1 XML Configuration Files in SIP 2.0

Configuration Type	**File or Directory**
Portal configuration	*Windows NT/2000:* `<`*`install_dir`*`>\conf\framework\OVPortalConfig.xml` *UNIX:* `/etc/opt/OV/SIP/conf/framework/OVPortalConfig.xml`
Module registration	*Windows NT/2000:* `<`*`install_dir`*`>\SIP\registration\` *UNIX:* `/etc/opt/OV/SIP/registration/`
Module configuration	*Windows NT/2000:* `<`*`install_dir`*`>\conf\share\modules\` *UNIX:* `/etc/opt/OV/SIP/conf/share/modules/`

Table 4-1 XML Configuration Files in SIP 2.0

Configuration Type	File or Directory
LDAP authentication configuration	*Windows NT/2000:* `<`*`install_dir`*`>\conf\share\authentication\LDAP\LDAP.xml` *UNIX:* `/etc/opt/OV/SIP/conf/share/authentication/LDAP/LDAP.xml`
Customer model configuration	*Windows NT/2000:* `<`*`install_dir`*`>\conf\share\organizations\` *UNIX:* `/etc/opt/OV/SIP/conf/share/organizations/`
User/Role configuration	*Windows NT/2000:* `<`*`install_dir`*`>\conf\share\roles\` *UNIX:* `/etc/opt/OV/SIP/conf/share/roles/`
Portal view configuration	*Windows NT/2000:* `<`*`install_dir`*`>\conf\share\views\` *UNIX:* `/etc/opt/OV/SIP/conf/share/views/`
User preferences configuration	*Windows NT/2000:* `<`*`install_dir`*`>\conf\share\users\` *UNIX:* `/etc/opt/OV/SIP/conf/share/users/`

Portal Configuration

The Portal Configuration file—`OVPortalConfig.xml`—holds global settings that apply to all customer portals. (Note, however, that the portal header, portal footer, and help topic can be overridden at the portal view level.) Here are the configurations stored in the `OVPortalConfig.xml` file:

- Tracing level
- Names of the portal header and portal footer JSPs
- Authentication provider configuration
- List of available ‘skins’
- List of available refresh rates
- Customer model data sources

Module Registration

Module registration files indicate to the SIP foundation what capabilities the module provides. Essentially, the registration file defines such information as a unique ID for the module, a reference to the servlet, the module's associated default help file, and the module's capabilities.

Module Configuration

Module configuration files are general and global configurations that are associated with a module and stored in an XML file as a convenient way to specify the configuration. They are like any other configuration files, but they happen to be XML.

LDAP Authentication Configuration

If you decide to use the LDAP authentication provider, you need to configure the LDAP configuration file supplied with SIP. In this configuration file, you specify the LDAP server and port, as well as other LDAP-specific information.

Customer Model Configuration

The Customer Model configuration is an XML file formatted according to the `SimpleCustomerModel.dtd`. The file is a mapping of customers to resources, where resources are the customers' associated hosts, interfaces, and services.

User/Role Configuration

The user/role configuration defines all valid portal users and associates each user with at least one role. Roles are also defined in the user/role configuration. A role defines what a user can see and do through the portal at a particular point in time.

Portal View Configuration

A portal view configuration is a configured set of modules and how they appear on tabs. It is also a configured set of portal view attributes, such as name in the welcome banner, portal color scheme, refresh rate, default tab, portal header, and portal footer, and others.

Portal view configurations can be based upon fourteen DTDs; however, you can think of them as one extended language that describes the portal view configuration file and is split into multiple DTDs for modularity purposes:

- The overall DTD—`PortalView.dtd`—defines the overall structure of the portal page.
- The `filter.dtd` describes the filtering elements.
- The remaining twelve DTDs are specific to each module that is configured in the portal view file: For example, `OVAlarms.dtd`, `OVBookmarks.dtd`, `OVGeneric.dtd`, etc.

  Sample View Files — “Canned” view files provide a starting point for the creation of new portal view files. Seven sample portal view files are provided with SIP.

User Preferences Configuration

Users who are associated with a role that has the editing permissions level set to “UserPreferences” or “ViewAdmin” can customize the name that appears in the portal welcome banner and the color scheme in which the portal is displayed. Because these two attributes are set in the portal view file which can be used by multiple users, when a user makes a change to one or both of these attributes, a user preferences configuration is created for the user and named for the user’s login.

Changes made to `User Options` are stored as `login.xml` in the User Preferences directory.

Refresh Model for SIP Configuration and Data Files

Depending upon the type of configuration or data file, changes to those files take effect in different ways:

- Some changes take effect when you **access** the information by displaying or refreshing the portal page.
- Some changes take effect on a scheduled, **periodic** basis, based on a configurable timer that determines the refresh rate for the information.
- Some changes take effect when you perform an administrative action, such as a command, that **forces** information to be refreshed.
- Some changes take effect when you **restart** the servlet engine. Certain data is loaded when the Portal servlet is first initialized and is not refreshed unless the servlet engine is stopped and restarted.

Table 4-2 Refresh Model for SIP Configuration and Data Files

Refresh Method	**Configuration or Data**	**How to Refresh the Data or File**
Access	Password file LDAP authentication data Portal views User preferences Alarms data Topology data VantagePoint Navigator data XSL style sheets Java Server Pages (except for `common.jsp`)	Display or refresh the portal page.

Table 4-2 Refresh Model for SIP Configuration and Data Files

Refresh Method	**Configuration or Data**	**How to Refresh the Data or File**
Periodic	Customer Model	Changes to the Customer Model take effect on a periodic basis, depending upon the value of the `timeout` attribute in the `OVPortalConfig.xml` file.
	Data Collector data	The data used by the NNM health gauges is gathered by NNM according to the schedule (in minutes) specified in the `rawDataRefresh` attribute in the `netHealthConfig.xml` file. The NNM data collection configuration, itself, is updated to match current SIP requirements by manual command line entry (`ovcolautoconf`) or according to a schedule set up by the administrator
	Symbol registration information	If changes are made within NNM to the “Symbol Type” assigned to particular devices, SIP receives the changes according to the schedule established by the `symbolFetchRateInMin` attribute in the `topologyConfig.xml` file.
Force	User/Role Model	Changes to the User-Role Model take effect after you rebuild the roles database using the following command: *Windows NT/2000:* `install_dir\bin\create_role_db` *UNIX:* `opt/OV/SIP/bin/create_role_db`

Table 4-2 Refresh Model for SIP Configuration and Data Files

Refresh Method	**Configuration or Data**	**How to Refresh the Data or File**
Restart	Portal configuration Module registration Module configuration Authentication provider configuration LDAP authentication configuration SIP Path Properties common.jsp	Stop and restart the servlet engine. *WindowsNT/2000*: From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**. *UNIX*: As `root`, stop and restart the web server and servlet engine by running the following. (The `DISPLAY` variable must be configured prior to restarting the webserver and servlet engine.) Stop on HP-UX: **`/sbin/init.d/ovsip stop`** Start on HP-UX: **`/sbin/init.d/ovsip start`** Stop on Solaris: **`/etc/init.d/ovsip stop`** Start on Solaris: **`/etc/init.d/ovsip start`**

Distributed and Shared Configuration Files

Service Information Portal provides support for distributed and shared configuration files. User roles, user preferences, portal views, and module configuration files can be shared or distributed across a network.

1. Using an ASCII editor, open the following file:

   Windows NT/2000:
   `<install_dir>\webapps\ovportal\SIPPath.properties`

   UNIX: `/opt/OV/SIP/webapps/ovportal/SIPPath.properties`

2. Set `SIP_SHARE_CONF_DIR` to a URL or shared file system path where you have installed your shared SIP configuration. The path should contain the equivalent of `/etc/opt/OV/SIP/conf/share/` in a local SIP configuration.

Figure 4-1 URL Example

```
SIP_SHARE_CONF_DIR=http://<YourServer.YourCompany.com>/OvShareDocs/
```

You must configure the web server with an appropriate alias for `/OvShareDocs/`. For example, you would configure an apache web server by adding a line like the following to `httpd.conf`:

```
Alias /OvShareDocs/ /etc/opt/OV/SIP/conf/share
```

Figure 4-2 HTTPS URL Example

```
SIP_SHARE_CONF_DIR=https://<YourServer.YourCompany.com>/OvShareDocs/
```

You can also serve shared configuration from a secure web server. In order for this to work, you must configure a Java SSL implementation like JSSE on the SIP client systems. For documentation of how to install JSSE, see “Installing JSSE or Equivalent” on page 77.

Figure 4-3 UNIX Shared File Path Example

```
SIP_SHARE_CONF_DIR=/net/YourServer/OvShareDocs
```

Clients must be UNIX systems. You can configure this using the NFS automounter on the client system and exportfs(1M) on a HP-UX server or share(1M) on a Solaris server.

Figure 4-4 Windows Shared File Path Example

```
SIP_SHARE_CONF_DIR=\\YourServer\OvShareDocs
```

In this case, you would share the folder on the server system, and the SIP client, which must be Windows, would automatically access it over the network. The server can also be a UNIX system, if you have NFS installed on the Windows system.

NOTE The `OVShareDocs` directory should contain the files that would otherwise appear in the `/share` directory:

Windows NT/2000: `<install_dir>\conf\share\`
UNIX: `/etc/opt/OV/SIP/conf/share/`

Rules for Direct Editing of XML Files

- Make a backup before modifying XML files.
- Understand editing permissions on XML files.
- Validate the XML after you modify it.
- Be careful not to lose changes made through the GUI. This can happen when you edit through the XML file and edit through the GUI at the same time.

Backing Up XML Files

Make a backup of XML configuration files before you customize them. If you edit the file and get incorrect XML syntax, you may want the ability to revert to the previous version of the file.

Understanding Editing Permission on XML Files

When using the editing windows within the SIP portal, the web server needs to have read/write permissions to the underlying files in order to save your changes. The apache web server and SIP run as:

Solaris: user `"nobody"`

HP-UX: user `"www"`

At runtime, `umask` is set by the `tomcat.sh` script to 022, so files are created mode 0644 and directories created mode 0755.

Therefore, at install time, SIP sets permissions and ownership for files to `mode 0644` and directories to `mode 0755`. If you add or change anything, make sure directories are owned by the appropriate user specified above, files set to `mode 0644`, and directories set to `mode 0755`.

For `tomcat` to operate properly, the following directories and all files underneath them need to have the correct permissions set (user as specified above, files set to mode 0644, and directories are set to mode 0755):

- `/opt/OV/SIP/tomcat`
  (directory only, so tomcat can create the work directory when needed)

- `/opt/OV/SIP/tomcat/conf`
  (directory only)
- `/opt/OV/SIP/tomcat/logs`
  (directory, all subdirectories, and all files)
- `/opt/OV/SIP/tomcat/webapps`
  (directory, all subdirectories, and all files)
- `/opt/OV/SIP/tomcat/work`
  (directory, all subdirectories, and all files)

For SIP to operate properly, these directories and all `.xml` files (not `.dtd` files) underneath them need to have the correct permissions set (`user` set to anyone with editing permissions, files set to mode 0644, and directories are set to mode 0755):

- `/etc/opt/OV/SIP/conf/share/organizations`
  (directory, all subdirectories, and all .xml files)
- `/etc/opt/OV/SIP/conf/share/users`
  (directory, all subdirectories, and all .xml files)
- `/etc/opt/OV/SIP/conf/share/modules`
  (directory, all subdirectories, and all .xml files)
- `/etc/opt/OV/SIP/conf/share/roles`
  (directory, all subdirectories, and all .xml files)
- `/etc/opt/OV/SIP/conf/share/views`
  (directory, all subdirectories, and all .xml files)

Validating XML Files

The Service Information Portal will detect and report an invalid XML configuration file. However, after you make modifications to XML files, you may want to validate your XML syntax.

Provided with SIP is the command `xmlvalidate`, which checks whether the XML file is both well-formed and valid. This command uses the same XML parser as SIP, so if the file passes `xmlvalidate`, it will work with SIP.

For the command to work from outside the `bin` directory, add the following to your PATH variable:

Windows NT/2000: `%SIP_HOME\bin`
UNIX: `/opt/OV/SIP/bin`

The correct usage of the `xmlvalidate` command is:

`xmlvalidate <`*`xml filename`*`>`.

An XML file is “well-formed” if it conforms to a minimal set of rules defined for all XML documents. It is “valid” if it conforms to the DTD listed at the beginning of the XML file.

Sometimes an error reported by `xmlvalidate` may not clearly indicate how to fix the problem. For example, a message like “Attribute ‘name’ must be declared for element type ‘XYZ’, is an indication that the attribute ‘name’ may have been misspelled.

As an alternative to `xmlvalidate`, you can find an XML validation tool for Windows NT at `www.xmlspy.com`.

Avoiding Loss of Changes

If you are using the portal interface to change a configuration and directly editing the XML configuration file at the same time, be careful not to lose the changes made through the interface by writing out the file over the interface changes.

5 Understanding SIP Security

The SIP Security Model

The SIP security model is made up of authentication and authorization.

Authentication is the process by which a user identifies and validates him/herself to the system. Authorization is the granting of access privileges to an authenticated user that determines what the user can see and do while logged into the system.

Authentication

SIP supplies several authentication providers from which you can choose. Additionally, SIP lets you integrate a custom authentication provider.

Information in the SIP "Administrator Guide," (Administrator_Guide.pdf) Chapter 6, "Configuring an Authentication Provider," on page 63, will help you determine the type of authentication provider that best suits your needs, and explain exactly how to configure an authentication provider.

Authorization

After a user is authenticated and considered a valid SIP user, authorization ensures that the user sees only the data you want him to see, and changes only the things you want her to change.

SIP uses an authorization model called the User-Role Model. Access rights are associated with roles. A **role** defines what a user can see and do through the portal at a particular point in time. By associating users with roles and assigning to each role what you want users in that role to be able to see and do, you achieve portal security.

All valid portal users are associated with at least one role. Users that are associated with multiple roles can easily switch among them through a drop-down list box in the interface.

A role consists of four parts:

- **A portal view**
  A portal view is what is displayed by the portal to a user. It is a configured set of modules and how they appear on tabs. It is also a configured set of portal view attributes, such as name in the welcome

banner, portal color scheme, refresh rate, default tab, portal header, portal footer, and others. (Two of these attributes—the name in the welcome banner and the color scheme—can be overridden by the user if they have at least "UserPreferences" editing permissions.)

- **An editing permissions level**
  Editing permission levels define the interactive editing operations that a user can perform through the portal interface. Each level of editing permissions includes all the operations defined by the previous level and adds some additional operations. The three levels are: ReadOnly, UserPreferences, and ViewAdmin. For an explanations of each, see the SIP "Administrator Guide," (Administrator_Guide.pdf) "Choosing the Level of Edit Permissions for a Role" on page 284.
- **A set of management data**
  Management data refers to the information about services, nodes, and interfaces that is displayed by management modules through the portal. In the User-Role Model, you specify the customers (organizations) whose data will be displayed to the users who are associated with a given role. Only the management data associated with the specified organizations is displayed.
- **An extensible list of role properties**
  Role properties can be used in several different ways: (1) To provide additional authorization information for the role, particularly for modules based on the Generic module; (2) As a way of filtering customer data (it can be an easy way to configure data that is displayed to different customers); and (3) To implement a single sign-on solution for logging in to back-end management applications.

Understanding the Three Authorization Models

SIP supports three authorization models:

- Explicit user entry
- User view file (Default Role)
- Default user

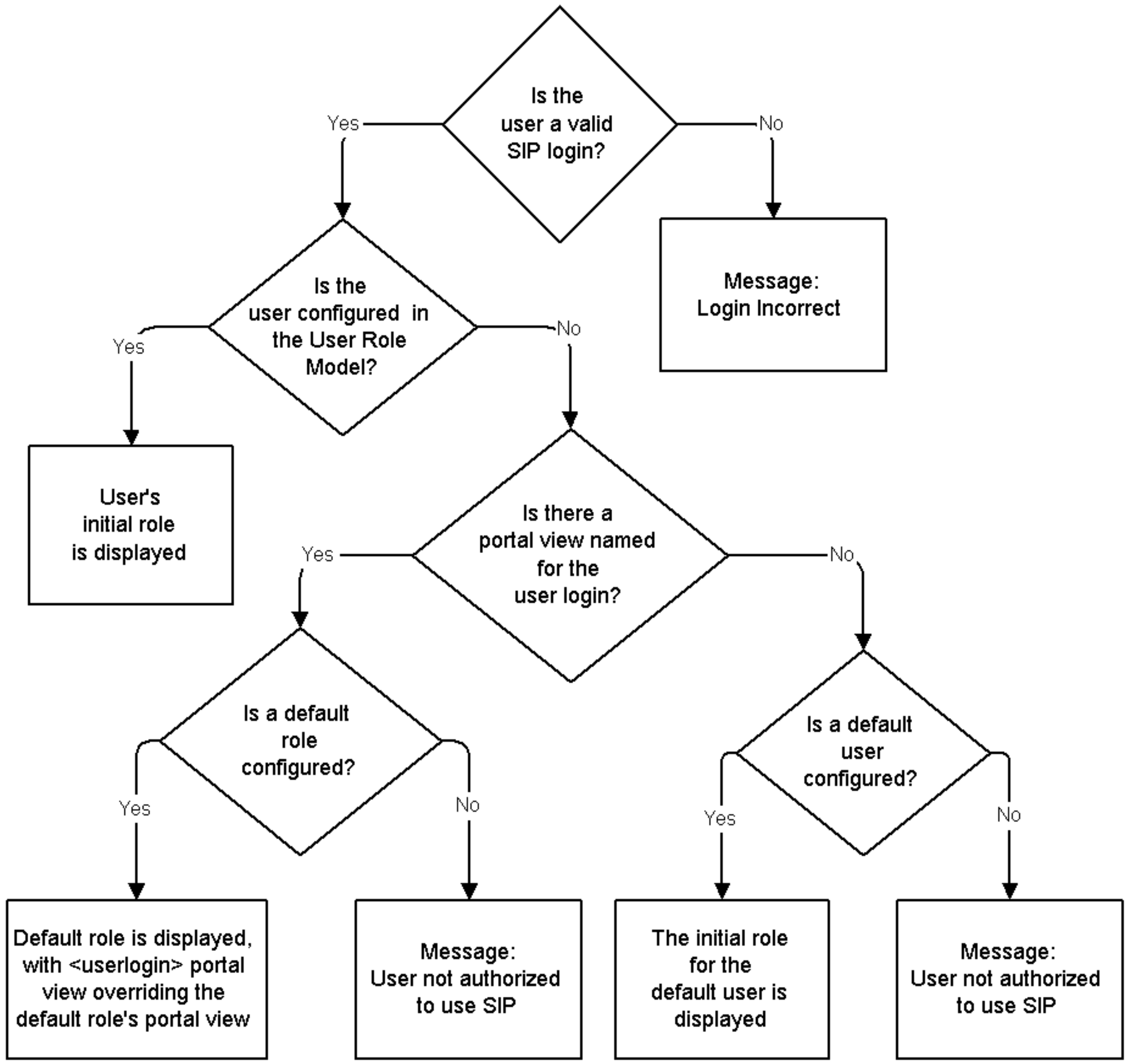
Following is an explanation of each model and a decision tree diagram that describes what validity checks occur when a user logs into SIP. Note that the authorization is checked in the above order.

Explicit User Entry In this case, the user login corresponds to an explicit user entry in the User-Role model. The user gets the initial role defined in the User entry. (You can check if there is an explicit entry for a user in the created roles database by running `test_role_db -u <user>`.)

User View File (Default Role) If there is no explicit user entry for the user login but there is a view file named <login>.xml and there is a default role (indicated by a value of “yes” for the `defaultRole` attribute of a configured role), the user gets the default role with the view file used as the view. This means of authorization is provided as a mechanism similar to SIP 1.0 which allows users to be configured by simply having a configuration file named for the user. This mechanism can be easily disabled by not specifying any default role. (You can see if any default role is configured by running `test_role_db` without any arguments.)

Default User If neither of the two preceding mechanisms allow the user to be authorized, the user gets the rights of the default user (if there is one configured). (The default user is specified by a value of “yes” for the defaultUser attribute of a User entry.) The user is logged in with the initial role of the default user and can switch to any of the roles assigned to the default user. (You can see if the default user is configured by running `test_role_db` without any arguments.) You can disable this mechanism by not having a default user.

Figure 5-1 User Authorization


Is the user a valid SIP login?
Yes
No
Message: Login Incorrect
Is the user configured in the User Role Model?
Yes
No
User's initial role is displayed
Is there a portal view named for the user login?
Yes
No
Is a default role configured?
Yes
No
Is a default user configured?
Yes
No
Default role is displayed, with <userlogin> portal view overriding the default role's portal view
Message: User not authorized to use SIP
The initial role for the default user is displayed
Message: User not authorized to use SIP

The User-Role Model is explained in more detail in Chapter 14, “Deploying Customer Portals,” on page 263. The filtering of customer data is explained in Chapter 8, “Filtering Data by Customers,” on page 119.

6 Configuring an Authentication Provider

Choosing An Authentication Provider

SIP 2.0 comes with six ready-to-use authentication providers. By default, the No Password Authentication Provider is registered with SIP. If you want to use one of the other five, you will need to do the following:

- Choose one of the supplied authentication providers or a custom authentication provider developed by you or someone else.
- If you choose the LDAP Authentication Provider, configure it.
- Register the authentication provider with SIP.
- Configure user logins, if applicable.

Table 6-1 on page 65 describes each of the six supplied authentication providers. (If none of these meets your needs, you may need to integrate a custom authentication provider.)

Background Information on Authentication

Of the supplied authentication providers, there are two types: those that do portal authentication and those that do external authentication. The three questions below help differentiate the two:

- Do you want SIP to perform the authentication or do you want an external mechanism to authenticate the user and then simply communicate to SIP whether the user has been authenticated?
- Will your portal users log in to SIP and log out of it as a standalone program, or will they run SIP from some other program?
- Do you want code that is integrated with the portal to do the authentication, or do you want to redirect the authentication responsibilities to some other program, system, or server?

Portal Authentication

The portal itself can display a login page and perform authentication, using services provided by the configured authentication provider. In this model, the user logs in to the portal, and can log out from the portal independently of login to any other authentication system.

In this type of authentication model, SIP gathers the user name and password from the user and then uses the authentication services or data source defined for the authentication provider to determine if the user is authenticated.

External Authentication

In this type of authentication provider, SIP never sees the password for the user. Some authentication service determines that the user has been authenticated and merely provides SIP with the user name. Based on the configured external authentication provider, SIP trusts that the user has been appropriately authenticated. The scope and duration of the user's authentication are defined externally, and are not a concern of SIP. SIP need only recognize whether or not the user is authenticated.

An external authentication provider can be used to make SIP a part of a larger single sign-on solution.

The Web Server Authentication Provider provided with SIP is an example of an external authentication provider. If the web server supports one of the various forms of authentication performed by the Web Server (HTTP Basic, HTTP Digest, HTTPS Client, or Windows NT Challenge/Response), and the portal administrator has configured it, the authentication provider class must merely recognize whether a user has already been authenticated. In this model, the scope and duration of the user's authentication are defined by the Web Server.

Table 6-1 Supplied Authentication provider

Supplied Authentication provider	Description	Type
Null Authentication Provider	No authentication is performed and no user name is required. Anyone who requests the portal becomes the user "anyuser." Note: What the user sees depends upon whether "anyuser" is defined in the User-Role Model, and whether there is a default user configured in the User-Role Model if "anyuser" is not explicitly defined.	external

Table 6-1 Supplied Authentication provider

Supplied Authentication provider	Description	Type
No-Password Authentication Provider	This is the default authentication provider. Requires only a user name. The user name must match the name of a user in the User-Role Model (unless the defaulting mechanisms, such as a default user, are used. No authentication is performed. This provider is useful when you want to set up and try out portal views for different users.	portal
Password File Authentication Provider	Requires a user name and password. The password is stored encrypted, using UNIX crypt(3), and the supplied password is encrypted and compared to the stored password. This is reasonably secure if the network communication channel is protected (via SSL, for example) and if the password file is not accessible outside the system.	portal
LDAP Authentication Provider	Requires a user name and password. SIP connects to the LDAP server, discovers whether the user name and password can be authenticated, then disconnects from the LDAP server.	portal
Web-Server Authentication Provider	No user name and password are required if the user has already been authenticated by the web server and browser. The user is considered authenticated if `HTTPServletRequest.getRemoteUser()` returns non-null. This assumes that you have set up one of the various web server authentication mechanisms and that the portal user has satisfied that mechanism’s Web Server Authentication.	external

Table 6-1 Supplied Authentication provider

Supplied Authentication provider	**Description**	**Type**
NNM SSO Authentication Provider	No user name and password are required if the user has already been authenticated by the NNM Session Manager web login mechanism. Since this was the login mechanism used for SIP 1.0, use of this authentication provider provides login functionality identical to SIP 1.0. This authentication provider requires that NNM is running on the same host as SIP.	external

Registering an Authentication Provider

After you choose an authentication provider, you need to indicate to SIP the one to use. The authentication provider is a portal-wide setting; therefore, it is configured in the `OVPortalConfig.xml` file.

1. Open the `OVPortalConfig.xml` file located in the following directory:

   Windows NT/2000: `<`*`install_dir`*`>\conf\framework\`
   UNIX: `/etc/opt/OV/SIP/conf/framework/`

2. Find the `Authentication` element in this file, and enter the following:

   - `LoginPage` (required)
   - `AuthenticationProviderClass` (required)
   - `ShowLogoutButton`
   - `LogoutPage`
   - `SessionTimeout`

   For sample configurations of each of the supplied authentication providers, see Figure 6-1 through Figure 6-6. For an explanation of each of the `Authentication` attributes, see Table 6-2 on page 71.

3. When finished, save the file.

4. After changing the portal configuration file, you must stop and restart the servlet engine before the changes will take effect in SIP. To do so:

   WindowsNT/2000:
   From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**.

   UNIX:

   As `root`, stop and restart the web server and servlet engine by typing the following (The `DISPLAY` variable must be configured prior to restarting the web server and servlet engine.):

Stop on HP-UX: **`/sbin/init.d/ovsip stop`**
Start on HP-UX: **`/sbin/init.d/ovsip start`**

Stop on Solaris: **`/etc/init.d/ovsip stop`**
Start on Solaris: **`/etc/init.d/ovsip start`**

Table 6-2 Configurable Attributes of the Authentication Element

Attribute	Description
`LoginPage`	The required `LoginPage` attribute is the URL of the login page used by the Authentication Provider. Portal and External Authentication Providers use this URL differently. A Portal Authentication Provider requires a login page containing a form with an action that reinvokes ovportal with the parameters `J_USERNAME` (user name) and `J_PASSWORD` (password). For an example, see the login page supplied with SIP, located in: *Windows NT/2000:* `<`*`install_dir`*`>\webapps\ovportal\jsp\security\login.jsp` *UNIX:* `/opt/OV/SIP/webapps/ovportal/jsp/security/login.jsp` If the login page is on the same host as SIP, this is referred to with the relative URL: `/ovportal/jsp/security/login.jsp` An External Authentication Provider can use several different types of login pages, depending on the security policy it implements. For example, the Null Authentication Provider uses `/ovportal` as its login page, since all logins automatically succeed. The NNM SSO Authentication Provider uses the login page of the NNM Web UI: `http://localhost/OvCgi/ovlaunch.exe?URL=/ovportal` Note that in order for this to work, the NNM Web UI must share the same host and web server port as SIP, because otherwise the cookies created by ovsessionmgr will not be valid for the SIP session. The WebServerAuthenticationProvider uses the "No Authentication Available" error page as its login page, since either the web server has already taken care of authentication, or some misconfiguration has occurred.

Table 6-2 Configurable Attributes of the Authentication Element

Attribute	Description
`AuthenticationProviderClass`	The fully-qualified name of a Java class that implements either the `PortalAuthenticationProvider` interface or the `ExternalAuthenticationProvider` interface. For example: `AuthenticationProviderClass="com.hp.ov.portal.security.FileAuthenticationProvider"`
`ShowLogoutButton`	If the Authentication Provider permits logout, configure `ShowLogoutButton="yes"`. For example, logout is meaningful in the case of the File Authentication Provider, but not in the case of the Web Server Authentication Provider (since you would have to exit the web browser to log out).
`LogoutPage`	If the authentication provider permits logout, and if it has information or controls that it wishes to present to the user after logout, configure the `LogoutPage` attribute with the URL to the logout page. For example: `/ovportal/jsp/security/logout.jsp`
`SessionTimeout`	The time, in seconds, that a SIP session can remain inactive before SIP will automatically terminate the session (automatic portal refreshes do not count as activity). Default session timeout is 32400 seconds, or 9 hours. "0" means use the application server's default session timeout, normally 30 minutes. "-1" means never time out.

Figure 6-1 Configuration for Null Authentication

```
<Authentication
    LoginPage="/ovportal"
    AuthenticationProviderClass=
       "com.hp.ov.portal.security.NullAuthenticationProvider"
    ShowLogoutButton="no"/>
```

Figure 6-2 Configuration for No Password Authentication

```
<Authentication
  LoginPage="/ovportal/jsp/security/userName.jsp"
  AuthenticationProviderClass=
  "com.hp.ov.portal.security.NoPasswordAuthenticationProvider"
  ShowLogoutButton="yes"
  LogoutPage="/ovportal/jsp/security/logout.jsp"/>
```

Figure 6-3 Configuration for Password File Authentication

```
<Authentication
    LoginPage="/ovportal/jsp/security/login.jsp"
    AuthenticationProviderClass=
       "com.hp.ov.portal.security.FileAuthenticationProvider"
    ShowLogoutButton="yes"
    LogoutPage="/ovportal/jsp/security/logout.jsp"/>
```

Figure 6-4 Configuration for LDAP Authentication

```
<Authentication
    LoginPage="/ovportal/jsp/security/login.jsp"
    AuthenticationProviderClass=
        "com.hp.ov.portal.security.LDAPAuthenticationProvider"
    ShowLogoutButton="yes"
    LogoutPage="/ovportal/jsp/security/logout.jsp"/>
```

Figure 6-5 Configuration for Web Server Authentication

```
 <Authentication
   LoginPage="/ovportal/jsp/security/noAuthAvail.jsp"
   AuthenticationProviderClass=
   "com.hp.ov.portal.security.WebServerAuthenticationProvider"
   ShowLogoutButton="no"/>
```

Figure 6-6 Configuration for NNM SSO Authentication (See Note below)

```
<Authentication
  LoginPage=
   "http://<NNM_SIP_host>/OvCgi/ovlaunch.exe?URL=/ovportal/"
  AuthenticationProviderClass=
   "com.hp.ov.portal.security.NNMSSOAuthenticationProvider"
  ShowLogoutButton="yes"
   LogoutPage="/ovportal/jsp/security/logout.jsp"/>
```

NOTE The NNM SSO Authentication Provider requires that NNM is running on the same host as SIP. The *<NNM_SIP_host>* referred to in Figure 6-6 is the fully-qualified hostname of the machine running SIP and NNM.

On Windows, the NNM SSO Authentication example in Figure 6-6 on page 74 will work without further configuration.

On UNIX, note that the login page URL is using the default port, rather than port 8880, which the NNM Web GUI normally uses on UNIX. If you want the NNM Web GUI to authenticate for SIP, it must run on the same web server (and same port) as SIP, or else the cookies the NNM Web UI creates will not be visible to SIP.

To make this work on UNIX, add the following to SIP's apache web server configuration file `/opt/OV/SIP/apache/conf/httpd.conf`:

```
Alias /OvDocs/ /opt/OV/www/htdocs/
ScriptAlias /OpenView /opt/OV/www/cgi-bin/OpenView
ScriptAlias /OvCgi/ /opt/OV/www/cgi-bin/
Alias /OvBackgrounds/ /etc/opt/OV/share/backgrounds/
```

Configuring User Logins

If you want to use an authentication provider that requires user logins and passwords, you need to configure the portal users to be permitted to log in. You do this by creating users and passwords, or by configuring the web server, or by some other method required by the registered authentication provider.

Password File Authentication Provider

For the Password File Authentication Provider, SIP provides `htpasswd`, a program for configuring user logins. The resulting `passwd` file stores user names and encrypted passwords. For example,

```
operator:2TD4P8x16h38o
larry:C0uxYt.Ububrg
david:Z3EVMN/1mtcSc
```

As shipped with SIP, the `passwd` file will contain one user: `ovuser` with the password `ovuser`.

Creating User Logins and Passwords

- On UNIX:

  As root, type:

`/opt/OV/SIP/bin/htpasswd  /etc/opt/OV/SIP/etc/passwd <username>`

- On Windows NT/2000:

  In a command window, type:

`"<install_dir>\bin\htpasswd" "<install_dir>\etc\passwd" <username>`

where <*username*> is the name of the user you want to add. You will be prompted for the user's password.

NOTE

A word about `htpasswd` parameters:

You must use the default encryption mechanism. Do not use the `-m` and `-s` parameters which specify an encryption mechanism other than the default.

If you use `-c` to create a passwd file and the file already exists, you will overwrite the existing file and lose any entries in that file.

If your circumstances are such that you have internal users (without root permissions) who need to create their own passwords for security purposes, those users can use the `-n` parameter to generate an entry and give it to you to edit into the `passwd` file.

For your information, the default password file location is configured in the `SIPPath.properties` file located in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\webapps\ovportal\`
UNIX: `/etc/opt/OV/SIP/webapps/ovportal/`

Deleting User Logins

1. Using a text editor, open the `passwd` file located in the following directory:

   Windows NT/2000: `<`*`install_dir`*`>\SIP\etc\passwd`
   UNIX: `/etc/opt/OV/SIP/etc/passwd`

2. Delete the login entry, and save the file.

Web-Server Authentication Provider

The Web Server Authentication Provider provided with SIP is an example of an external authentication provider.

If the web server supports one of the various forms of authentication performed by the Web Server (HTTP Basic, HTTP Digest, HTTPS Client, or Windows NT Challenge/Response), and you have configured it, the Web Server Authenticatin Provider must merely recognize whether a user has already been authenticated. In this model, the scope and duration of the user’s authentication are defined by the Web Server.

Configuring user logins is a web-server-specific task; refer to Apache and IIS documentation.

Configuring the LDAP Authentication Provider

The LDAP Authentication Provider works by getting the username and password from the login page, and binding with the LDAP server using that username and password. It does not look up the user's password in the LDAP database.

Installing JSSE or Equivalent

If the LDAP server performs authenticated binds using SSL connections, you need to install a secure socket implementation, such as JSSE (Java Secure Socket Extension), available from `java.sun.com`.

1. Go to `http://java.sun.com/products/jsse/`
2. Download "JSSE 1.0.2 global software and documentation," saving the file as `jsse-1_0_2-gl.zip`.
3. Unpack this file with the command: `jar xvf jsse-1_0_2-gl.zip`.
4. Refer to `jsse1.0.2/INSTALL.txt` for installation instructions.

Registering the LDAP Authentication Provider

Register the LDAP Authentication Provider in `OVPortalConfig.xml`, as described on page 68. For an example registration, see Figure 6-4 on page 73.

Configuring the LDAP Authentication Provider

1. Configure the LDAP Authentication Provider. It is configured in the following file:

   Windows NT/2000:
   `<`*`install_dir`*`>\conf\share\authentication\LDAP\LDAP.xml`

   UNIX:
   `/etc/opt/OV/SIP/conf/share/authentication/LDAP/LDAP.xml`

NOTE The `LDAP.dtd` file is found in the same directory. The parameters defined in these files are described in Table 6-3 below.

2. Test the LDAP Authentication Provider configuration to confirm that you configured the parameters correctly.

   Run the command:

   Windows NT/2000:
   <***install_dir***>**\bin\LDAP** ***username password***

   UNIX: **/opt/OV/SIP/bin/LDAP** ***username password***

   This will output to the command window verbose tracing of each step in the LDAP authentication. Any failures should indicate what parameter is configured incorrectly. Be sure to test each of the following cases:

   - No such user (should fail)
   - Valid user, wrong password (should fail)
   - Valid user, correct password (should succeed)

3. To make the configuration changes take effect, stop and restart the servlet engine:

   WindowsNT/2000:
   From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **net stop tomcat** and **net start tomcat**.

   UNIX:

   As `root`, stop and restart the web server and servlet engine by running the following. (The `DISPLAY` variable must be configured prior to restarting the webserver and servlet engine.)

   Stop on HP-UX: **/sbin/init.d/ovsip stop**
   Start on HP-UX: **/sbin/init.d/ovsip start**

   Stop on Solaris: **/etc/init.d/ovsip stop**
   Start on Solaris: **/etc/init.d/ovsip start**

Table 6-3 Attributes of the LDAPServerConfig Element

Attributes	**Description**
`baseDN`	A required attribute that defines the context in which the LDAP Authentication Provider will search the LDAP database for the user to authenticate. `baseDN` is a string. For example, `baseDN="ou=Employees, o=acme.com"`

`uidAttr`	The user ID attribute to search for. For example, if the LDAP.xml file contains:
	`uidAttr="uid"`
	and the user submits to the login page the name:
	"`Wile_E_Coyote@acme.com`"
	and `baseDN` is:
	`baseDN="ou=Employees, o=acme.com"`
	Then, the LDAP Authentication Provider will look up the user with the search:
	(`ou=Employees, o=acme.com, uid=Wile_E_Coyote@acme.com`)

`LDAPServer`	A required attribute that defines the LDAP server to which to authenticate. For example: `LDAPServer="ldap.acme.com"`
`LDAPPort`	A required attribute that defines the port at which the LDAP server is listening. If `useSSL` is "yes", this should be the port at which the LDAP server is accepting SSL connections. The standard port for LDAP is 389, and for LDAP over SSL is 636. For example: `LDAPPort="636"`

Table 6-3 Attributes of the LDAPServerConfig Element

Attributes	**Description**
`useSSL`	Should be "yes" if the LDAP server performs authenticated binds using an SSL connection, and "no" if it authenticates passwords submitted in plain text and not over an SSL connection. For example: `useSSL="yes"`
`authType`	The type of authentication used by the LDAP server. Valid values of `authType` are: `none`: Use no authentication (anonymous) `simple`: Use weak authentication (clear-text password) `sas1_mech`: A space-separated list of SASL mechanism names. SASL is specified in RFC 2222. The LDAP authentication provider has only been verified to support "simple" authentication, but over an SSL connection (useSSL="yes"), this is reasonably secure. For example: `authType="simple"`
`searchFirst`	If this is "yes", the LDAP Authentication Provider will search for username in the database before attempting to bind as that user. This is appropriate if the uidAttr is an alias, and the LDAP server requires the user to bind as the real Distinguished Name returned by the search, if successful. Performance will be better if this can be "no", since the database need only be accessed once. For example: `searchFirst="yes"`

7 Implementing a Customer Model for Mapping Customers to Resources

Understanding the SIP 2.0 Customer Model

A customer model is mapping of customers to resources, where resources are the associated hosts, interfaces, and services. The SIP customer model makes it possible to present data filtered by customer.

SIP 2.0 uses the so-called “simple customer model” which is defined via an XML DTD. Your customer model can be defined in several XML files, or a mix of programs and files. The resulting "sub-models" are merged into one model. The source of the customer model data is a URL or file. The URL or filename can reference one of the following:

- An XML file that you create from scratch.
- A CGI program that dynamically returns XML content from a remote NNM Customer Views server.
- A CGI program that performs a one-time migration of NNM Customer Views data into an XML file, which then becomes a source of customer model data.
- A servlet that converts NNM Object Database information to XML lists of nodes and interfaces that can be referenced in other parts of the customer model.
- A program that provides a mapping from an arbitrary data store or provisioning system to the required "simple customer model" format. For information on developing a custom customer model data source, see “Developing a Custom Customer Model Data Source” on page 116.

The use of an XML file to define the customer model overcomes some of the shortcomings of the Customer Views customer model used in SIP 1.0. With an XML file:

- A customer resource mapping can be defined in the SIP customer model prior to discovery of resources by NNM.
- Support for non-IP interfaces, for example, switch ports, not supported by Customer Views.
- A customer resource mapping can be defined in terms of services.

As previously mentioned, the simple customer model maps customers to hosts, interfaces, and services. The list of named services is new in SIP 2.0. It serves as a simple extensibility mechanism in that it does not contain the definition of the services, but merely provides a way to refer to particular services that are defined elsewhere.

The Three Parts of Customer Data Filtering

The mapping of customers to resources is the first step in the overall process of filtering customer data in SIP. Ultimately, the mapping is contained in one or more XML files that serve as a master list of all customers (organizations) whose data will be displayed through SIP.

This chapter explains how to create the customer-to-resource mapping. The User Role model described in Chapter 14, “Deploying Customer Portals,” on page 263, explains the filtering of customer data after the data is in a form that SIP can use.

- **The mapping of customers to resources**. This involves getting customer data into a format that conforms to the simple customer model DTD. In SIP, the mapping files are referred to as the **customer model**.
- **The filtering of customer data for security purposes**. Security filtering is the first (and most important) of two levels of filtering. It defines what data the customer can potentially see. It ensures that a customer can see only the data that is appropriate to them. In SIP 2.0 this is accomplished through an authorization model referred to as the User-Role Model. By associating users with roles and assigning to each role what you want the user to be able to see and do, you achieve portal security. Security filtering through the User-Role Model is explained in Chapter 8, “Filtering Data by Customers,” on page 119.
- **The filtering of customer data for display purposes**. Display filtering is applied on a module-by-module or submodule-by-submodule basis and further restricts what the customer sees. Because display filtering is applied on a module-by-module basis, it is explained in the documentation for each management product. See “Getting Additional Documentation” on page 21.

NOTE **SIP 1.0 users**: For a comparison of how customer data filtering was done in SIP 1.0, see Table 7-2 on page 85.

Table 7-1 SIP 2.0 Customer Data Filtering

	Mapping Customers to Resources	Filtering Customer Data for Security	Filter Customer Data for Module Display
SIP 2.0	Create the customer model, defined in several XML files, or a mix of CGIs, servlets, and XML files: • Create an XML file from scratch. • Run a CGI program to dynamically return organization data from NNM Customer Views in the form of XML content. • Run a CGI program to perform a one-time migration of Customer Views organization data to an XML file. • Run a servlet to retrieve and filter NNM object database information and convert it to XML content that is a partial customer model. Register the sources of the customer model with SIP in `OVPortalConfig.xml`.	On a role basis, define the customers whose data can potentially be displayed. `MgmtData` element in *`UserRole`*`.xml`. `MgmtData` element may have an `OrganizationFilter` child element. `MgmtData` filter is documented in “Security Filtering Element Definitions and Examples” on page 130.	On a module-by-module** basis, further restrict what is displayed: `NodeSelection` filter in *`PortalView`*`.xml` (and child elements: `OrganizationFilter`, `IPHostFilter`, `CapabilityFilter`) `InterfaceSelection` filter in *`PortalView`*`.xml` (and child elements: `OrganizationFilter`, `IPInterfaceFilter`) `NodeSelection` and `InterfaceSelection` filters are documented in “Display Filtering Element Definitions and Examples” on page 136.

** In the case of the NNM Alarms module, the `NodeSelection` element is defined in the alarms category files (*`NmAlarmCat`*`.xml`), not in *`PortalView`*`.xml`.

Table 7-2 SIP 1.0 Customer Data Filtering

	Mapping Customers to Resources	**Filtering Customer Data for Security**	**Filtering Customer Data for Module Display**
SIP 1.0	Configure customer mappings to nodes and interfaces in NNM Customer Views.	On a customer portal basis, define the customers whose data is potentially displayed. `SecurityFilter` in *`OVUserPortal`*`.xml`	On a module-by-module* basis, further restrict what is displayed: `NodeSelection` filter in *`OVUserPortal`*`.xml` (and child elements: `CustomerFilter`, `IPHostFilter`, `CapabilityFilter`) `InterfaceSelection` filter in *`OVUserPortal`*`.xml` (and child elements: `CustomerFilter` `IPInterfaceFilter`)

* In the case of the NNM Alarms module, the `NodeSelection` element is defined in the alarms category files (*`NmAlarmCat`*`.xml`), not *`OVUserPortal`*`.xml`.

Process of Implementing a Customer Model

Two tasks (explained in the following sections) are required when setting up the customer-to-resources mapping and making it available to SIP:

1. Create the customer model, which can be defined in several XML files, or a mix of programs and files.
2. Register the sources of the customer model with SIP.

NOTE The configuring of customer data filtering is done as part of the process of creating roles and associating them with users. This is done through direct editing of *`UserRole`*`.xml` files. For more information on customer data filtering, see “Understanding the User-Role Model” on page 266.

Overview of Creating a Customer Model

The customer model can be defined in several XML files, or a mix of CGIs, servlets, and XML files. The resulting "sub-models" are merged into one customer model.

There are four ways to create customer model data. These four approaches are not mutually exclusive and can be used in combination:

- **Create a customer model from scratch.**

  Use this approach if you want to create customer model data based upon the `SimpleCustomerModel.dtd` but not leverage from other customer models.

- **Configure NNM Customer Views to dynamically return its customer mappings to nodes and interfaces.**

  Use this approach if you want to use the customer model from a remote Customer Views server. The returned organization data conforms to the `SimpleCustomerModel.dtd` and is a complete customer model. (This is the customer model used in SIP 1.0.)

- **Perform a one-time migration of NNM Customer Views organization mappings to an XML file.**

  Use this approach if you want to leverage the NNM Customer Views customer model but also want the flexibility to manually add to it. Essentially, this gives you a one-time snapshot of the data to be used as a starting point for additional customer model data.

- **Create a partial customer model from NNM object database (ovwdb) information.**

  Use this approach if you want to retrieve filtered information from the NNM object database (ovwdb) and automatically return XML content. The generated data is a partial customer model formatted according to the `SimpleCustomerModel.dtd`.

- **Integrate data using a custom customer model data source.**

  Use this approach if you have a program that converts from a database to the simple customer model.

The following sections explain all four approaches.

Creating a Customer Model From Scratch

You can create multiple XML files and have different parts of the customer model stored in different files.

1. Create an XML file based on the `SimpleCustomerModel.dtd`. The DTD is located in the following directory:

   Windows NT/2000: `<`*`install_dir`*`>\conf\share\organizations\`
   UNIX: `/etc/opt/OV/SIP/conf/share/organizations/`

2. Save the XML file. Although it can be located anywhere on the file system (as long as the path to the document is registered in `OVPortalConfig.xml`) the standard location is:

   Windows NT/2000: `<`*`install_dir`*`>\conf\share\organizations\`
   UNIX: `/etc/opt/OV/SIP/conf/share/organizations/`

NOTE Later, when you register the customer model source in `OVPortalConfig.xml`, relative paths are interpreted relative to the `organizations` directory.

3. Register the source of the customer model data with SIP. For detailed instructions, see “Registering the Customer Model Data Source with SIP” on page 105.

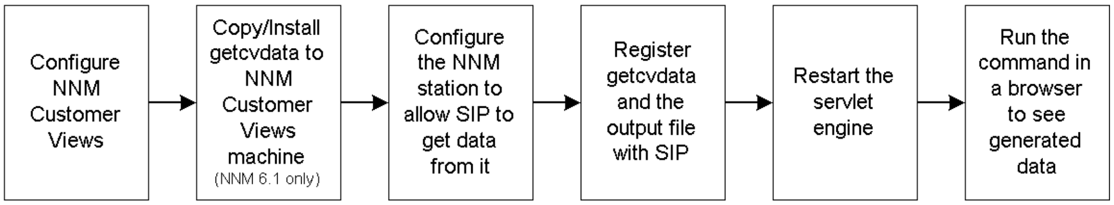
4. Restart the servlet engine, as described in “Restarting the Servlet Engine” on page 94

Configuring NNM Customer Views to Dynamically Return Organization Data

Through use of a CGI program, you can dynamically return customer model data from one or more remote Customer Views servers. This program—`getcvdata`—returns a complete customer model. That is, the mappings are complete and they are consistent with the `SimpleCustomerModel.dtd.`

If you configure SIP to call getcvdata on multiple Customer Views servers, it will combine the information from each of these servers into a consolidated customer model.

Figure 7-1 Process of Dynamically Generating Customer Views Organization Data


Configure NNM Customer Views
Copy/Install getcvdata to NNM Customer Views machine (NNM 6.1 only)
Configure the NNM station to allow SIP to get data from it
Register getcvdata and the output file with SIP
Restart the servlet engine
Run the command in a browser to see generated data

Configuring NNM Customer Views

The `ovcustomer` command creates organizations (customers) and associates specific nodes and interfaces with the organizations.

In the context of Customer Views, the term “organization” refers to the organizations provided by NNM Customer Views and includes both “customers” and “providers.”

To configure Customer Views for use with SIP, perform the four tasks listed below. The commands for doing so are described in Table 7-3 on page 89. For detailed information, see the Customer Views user documentation.

1. Start Customer Views.
2. Create Organizations.

3. Associate Nodes with Organizations.
4. Associate Interfaces with Organizations.

The `ovcustomer` command can be run interactively or in batch mode. To run the `ovcustomer` command in interactive mode, run the `ovcustomer` command and then enter specific commands at the “`ovcustomer>`” prompt. Shown below are the relevant `ovcustomer` commands:

Table 7-3 **`ovcustomer` Commands**

Action	Command
Create a new organization.	`ovcustomer>`**`create_org`** ***`<organizationType>`*** ***`<organizationName>`*** “customer” and “provider” are supported values for *organizationType*. An organization name with spaces should be placed in quotes (for example, `"My Customer"`).
Print the list of organizations.	`ovcustomer>`**`print_org`**
Associate a node with an organization.	`ovcustomer>`**`add_associations_to_org`** ***`<organizationName>`*** ***`<Hostname>`***
Print the nodes associated with a specific organization.	`ovcustomer>`**`print_associated_node`** ***`<organizationName>`***
Associate an interface with an organization.	`ovcustomer>`**`add_associations_to_org <organizationName> <IPaddress>`**
Print the interfaces associated with a specific organization.	`ovcustomer>`**`print_associated_interface <organizationName>`**

Copying getcvdata to NNM Stations Running Customer Views

You can skip this step if the NNM management stations running Customer Views are running NNM version 6.2.

SIP installs the CGI program `getcvdata` (along with other programs) in the SIP directory structure as `installCGIs.zip` and `installCGIs.tar.Z`. You need to copy the compressed file to each NNM management station that is running Customer Views on NNM version 6.1.

IMPORTANT The other programs installed with `installCGIs.zip` and `installCGIs.tar.Z` are essential to establishing full communication between NNM and SIP. To correctly install these programs, please reference *Configuring NNM* (`Configuring_NNM.pdf`), “Establishing Communication Between NNM and SIP: On the SIP Server” step 5. For location of PDF files, see “Getting Additional Documentation” on page 21.

- If SIP is running on Windows NT/2000, copy the following zip or tar file to a newly created temporary directory on the NNM 6.1 management station that is running Customer Views:
  - Windows NT/2000:
    `\<`*`SIP_inst_dir`*`>\cgi-bin\WindowsNT\installCGIs.zip`
  - HP-UX:
    `\<`*`SIP_inst_dir`*`>\cgi-bin\HP-UX11\installCGIs.tar.Z`
  - Solaris:
    `\<`*`SIP_inst_dir`*`>\cgi-bin\Solaris2.X\installCGIs.tar.Z`
- If SIP is running on UNIX, copy the following zip or tar file to a newly created temporary directory on the NNM 6.1 management station that is running Customer Views:
  - Windows NT/2000:
    `/opt/OV/SIP/cgi-bin/WindowsNT/installCGIs.zip`
  - HP-UX:
    `/opt/OV/SIP/cgi-bin/HP-UX11/installCGIs.tar.Z`

— Solaris:
`/opt/OV/SIP/cgi-bin/Solaris2.X/installCGIs.tar.Z`

Installing getcvdata

You can skip this step if the NNM management station running Customer Views is running NNM version 6.2.

1. At the command prompt on the NNM station, navigate to the `installCGIs.zip` or `installCGIs.tar.Z` file that you placed on this NNM 6.1 management station in the previous section.
2. Unzip or uncompress and untar the file.
3. At the command prompt, type:

   - *Windows NT\2000*:
     `\NNM_inst_dir\bin\Perl\bin\perl.exe installCGIs.pl`
   - *UNIX*:
     `/opt/OV/bin/Perl/bin/perl installCGIs.pl`

4. You can now remove the `installCGIs` file and the directory structure around it.

Configuring the NNM Management Station to Allow SIP to Obtain Information

You can skip this step only if SIP is running on the same computer as NNM.

In this step, you modify the files that configure which computers are allowed to request information from the NNM management station.

1. On the NNM management station, open the following two authorization configuration files,

   - *`NNM_install_dir`*`/conf/ovw.auth`
     This file controls which hosts and users are authorized to connect to NNM sessions running on the management station.
   - *`NNM_install_dir`*`/conf/ovwdb.auth`
     This file controls which hosts and users are authorized to connect to the NNM object database processes.

TIP If you see a line that simply has two + symbols (+ +), you can skip this step because NNM is configured to allow any computer to request information (security not implemented).

2. Add a ***`SIPserverHostName`*** **`+`** line to the list for each SIP server that needs to obtain information from this NNM management station.
3. After completing this configuration, refer to “Registering the Customer Model Data Source with SIP” on page 105.

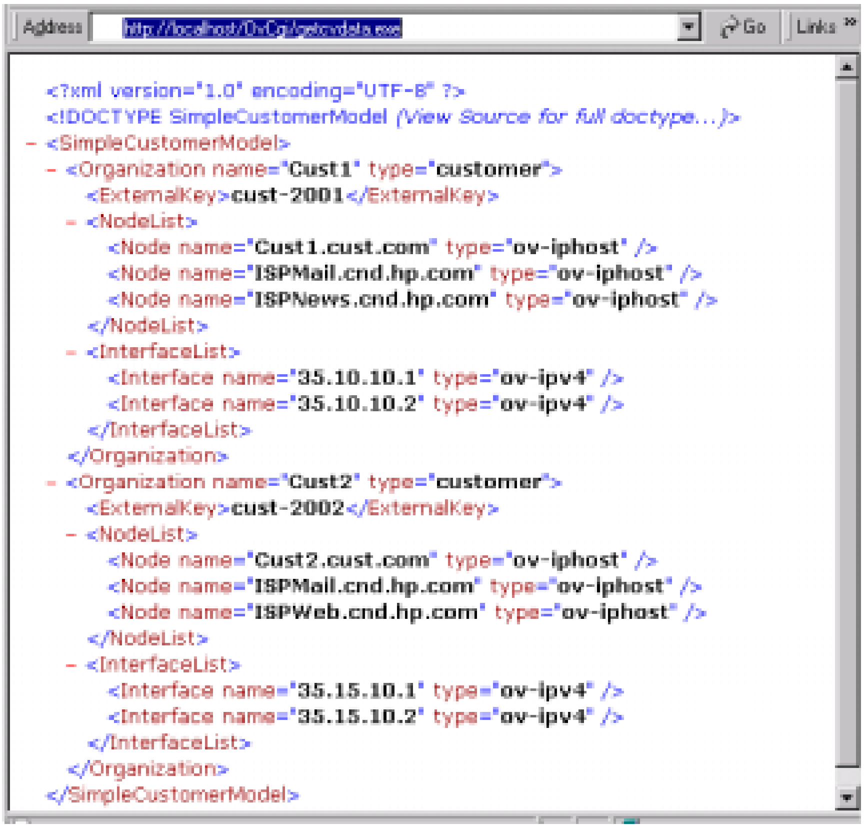
Viewing the Generated Data in a Browser

In this step you generate and display the lists and verify that the results meet your expectations and requirements. When you run `getcvdata`, the XML content is stored in memory on the SIP server. It is refreshed based on a scheduled time, or when the servlet engine is restarted.

1. Open a browser window on the SIP server and type:

   Windows NT\2000: **`http://`*`NNMhostname`*`/OvCgi/getcvdata.exe`**
   UNIX: **`http://`*`NNMhostname`*`:8880/OvCgi/getcvdata.exe`**

2. Examine the XML output and verify that the results meet your expectations and requirements. Here is an example:


Address http://localhost/OvCgi/getcvdata.exe Go Links

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SimpleCustomerModel (View Source for full doctype...)>
- <SimpleCustomerModel>
  - <Organization name="Cust1" type="customer">
      <ExternalKey>cust-2001</ExternalKey>
    - <NodeList>
        <Node name="Cust1.cust.com" type="ov-iphost" />
        <Node name="ISPMail.cnd.hp.com" type="ov-iphost" />
        <Node name="ISPNews.cnd.hp.com" type="ov-iphost" />
      </NodeList>
    - <InterfaceList>
        <Interface name="35.10.10.1" type="ov-ipv4" />
        <Interface name="35.10.10.2" type="ov-ipv4" />
      </InterfaceList>
    </Organization>
  - <Organization name="Cust2" type="customer">
      <ExternalKey>cust-2002</ExternalKey>
    - <NodeList>
        <Node name="Cust2.cust.com" type="ov-iphost" />
        <Node name="ISPMail.cnd.hp.com" type="ov-iphost" />
        <Node name="ISPWeb.cnd.hp.com" type="ov-iphost" />
      </NodeList>
    - <InterfaceList>
        <Interface name="35.15.10.1" type="ov-ipv4" />
        <Interface name="35.15.10.2" type="ov-ipv4" />
      </InterfaceList>
    </Organization>
  </SimpleCustomerModel>
```

Registering getcvdata with SIP

1. Open `OVPortalConfig.xml` located in the following directory:

   Windows NT/2000: `<`*`install_dir`*`>\conf\framework\`
   UNIX: `/etc/opt/OV/SIP/conf/framework/`

2. In the `CustomerModelSources` element, enter the URL to launch `getcvdata`. See the example below for an example of defining `getcvdata`. For more examples of defining the customer model file, see Figure 7-9, Figure 7-10, and Figure 7-11 on page 106.

Figure 7-2 Example Registration for getcvdata

```
<OVPortalConfig>
  <CustomerModelSources timeout="60">
    <CustomerModel
       href="http://NNMHostname/OvCgi/getcvdata.exe"/>
  </CustomerModelSources>
</OVPortalConfig>
```

NOTE For UNIX systems, the above example `href` is replaced by:

```
href="http://NNMHostname:8880/OvCgi/getcvdata.exe"/>
```

3. To periodically update the customer model according to a schedule (in addition to each time that SIP's `tomcat` service is restarted), enter a `timeout="`*`minutes`*`"` attribute, as shown in the example above.

Restarting the Servlet Engine

After changing `OVPortalConfig.xml`, you must stop and restart the servlet engine before the changes will take effect in SIP.

WindowsNT/2000:
From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**.

UNIX:

As `root`, stop and restart the web server and servlet engine by running the following. (The `DISPLAY` variable must be configured prior to restarting the webserver and servlet engine.)

Stop on HP-UX: **`/sbin/init.d/ovsip stop`**
Start on HP-UX: **`/sbin/init.d/ovsip start`**

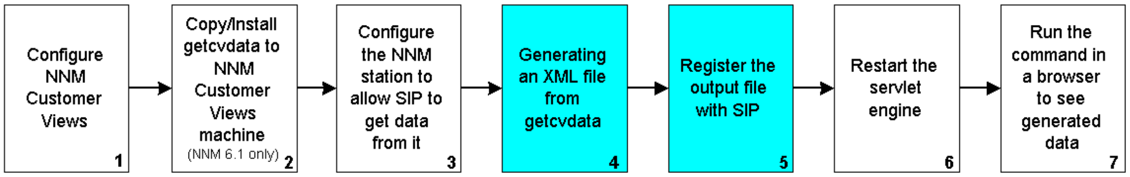
Stop on Solaris: **`/etc/init.d/ovsip stop`**
Start on Solaris: **`/etc/init.d/ovsip start`**

Running a CGI Program that Performs a One-Time Migration of Customer Views Organization Data

The `getcvdata.exe` command exports Customer Views organization data to XML content that conforms to the `SimpleCustomerModel.dtd`.

As shown below, the process is the same as “Configuring NNM Customer Views to Dynamically Return Organization Data” on page 88, with the exception of steps 4 and 5: generating the XML file and registering the XML file with SIP.

Figure 7-3 Process of Getting a One-Time Migration of Customer Views Organization Data


Configure NNM Customer Views
1
Copy/Install getcvdata to NNM Customer Views machine
(NNM 6.1 only)
2
Configure the NNM station to allow SIP to get data from it
3
Generating an XML file from getcvdata
4
Register the output file with SIP
5
Restart the servlet engine
6
Run the command in a browser to see generated data
7

Generating an XML File from getcvdata

1. From a command prompt, run this command:

   Windows NT/2000:
   <*NNM_install_dir*>\www\cgi-bin\getcvdata.exe > *CustomerModel*.xml

   UNIX: **/opt/OV/www/cgi-bin/getcvdata.exe > *CustomerModel*.xml**

   Using this command without parameters returns all information for all the customers known by the Customer Views server.

2. The generated XML file references the DTD path. If you place the XML file in the `/conf/share/organizations` directory on the SIP server or a remote SIP configuration server, the DTD path is defined appropriately (Figure 7-4 on page 96). If you place the XML file in some other location, you need to modify the path of the DTD in the

DOCTYPE statement of the XML file so it correctly references the location of the DTD file.

NOTE If the XML file is local, run `xmlvalidate` to validate that it can find the DTD file. (See “Validating XML Files” on page 55.)

If the XML file is referenced through a remote URL, the DTD will need to be installed in a web-accessible location, which must then be referenced in the XML file.

Figure 7-4 Example of DTD Reference in File

```
<!DOCTYPE SimpleCustomerModel SYSTEM "SimpleCustomerModel.dtd">
```

NOTE Although the XML file can be located anywhere on the file system (as long as the path to the document is registered in `OVPortalConfig.xml`) the standard location is:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\organizations\`
UNIX: `/etc/opt/OV/SIP/conf/share/organizations/`

Registering the Output File with SIP

This process is the same as “Registering getcvdata with SIP” on page 93, except that you only need to register the customer model data file, not the `getcvdata` program.

For registration examples, see Figure 7-9 and Figure 7-10 on page 105, and Figure 7-11 on page 106.

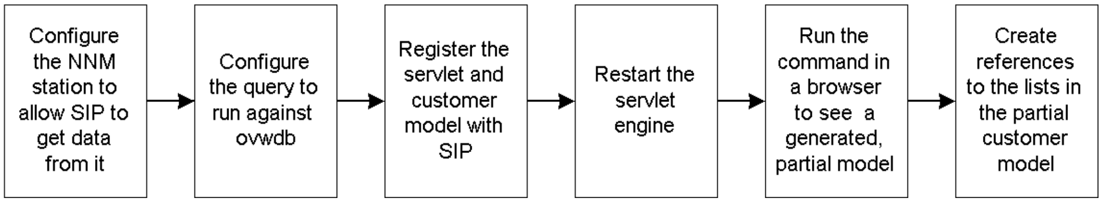
Converting NNM Object Database Information to XML

SIP provides a servlet that retrieves filtered information from NNM's object database (ovwdb) and automatically returns XML content that is stored in memory. The generated data is a partial customer model formatted according to the `SimpleCustomerModel.dtd`. Essentially, the XML content contains lists of nodes and interfaces that can be mapped to organizations (customers) in the *`CustomerModel`*`.xml` file.

You can use this tool in two ways:

- To dynamically create lists of nodes and interfaces that are generated according to a schedule that you control and that are stored in memory; simply refer to these lists by name in your `Organization` definitions.
- To create correctly formatted lists of nodes and interfaces that can be copied and pasted into your `Organization` definitions.

Figure 7-5 Process of Dynamically Converting NNM Object Database Data to XML


Configure the NNM station to allow SIP to get data from it
Configure the query to run against ovwdb
Register the servlet and customer model with SIP
Restart the servlet engine
Run the command in a browser to see a generated, partial model
Create references to the lists in the partial customer model

The first step is to configure the NNM management station to allow SIP to get data from it. See "Configuring the NNM Management Station to Allow SIP to Obtain Information" on page 91.

Configuring the Query to Run Against ovwdb

The query is configured in an XML file and is used to filter the ovwdb data. It defines the nodes and interfaces that should be returned and grouped into specific lists.

You can do two kinds of filtering:

- **Perl regular expression-based filtering**. Use Perl5 regular expressions to define subsets of the customer model objects. For example, the Managed Resources module can be configured to list all nodes that match `.*\.customer1\.com`.

NOTE Because regular expression wildcarding is supported, "dots" in hostnames should be escaped with '\'. For example, to match all nodes in the domain `eagle.wingnuts.com`, you would specify:

```
.*\.eagle\.wingnuts\.com
```

- **Capability filtering**. The `CapabilityFilter` defines any valid NNM object database field.

  The `CapabilityFilter` is a child element of `IPHostFilter`. It commonly refers to the capability filters such as isRouter, isNode, etc. However, `CapabilityFilter` may refer to any NNM object database field within HP OpenView Network Node Manager (NNM). Below is an example `CapabilityFilter`:

```
<NodeSelection>
   <CapabilityFilter op="OR">
       <Capability field="IPStatus" value="Critical"/>
       <Capability value="isServer"/>
   </CapabilityFilter>
</NodeSelection>
```

  The above example will return any node that has the `isServer` capability, or is critical.

  Be aware that an empty `CapabilityFilter` yields the empty set which allows nothing to pass.

The following steps guide the configuring of the query:

1. Open the `NNMData.xml` file located in the following directory:

   Windows NT/2000: `<`*`install_dir`*`>\conf\share\modules\NM\`
   UNIX: `/etc/opt/OV/SIP/conf/share/modules/NM/`

2. Enter the configuration information, referring to the example below, in which all IP hosts are returned from the local NNM station. For more information, see the `NNMData.dtd`. If you need information about NNM management station port settings, see *Configuring NNM* (`Configuring_NNM.pdf`):

```
<NNMSimpleCustomerModel>
  <NNMNodeList name="List1">
    <NNMStation hostname="localhost" ovwdbPort="9999" />
      <NodeSelection>
        <IPHostFilter>
          <IPHost hostname=".*" />
        </IPHostFilter>
      </NodeSelection>
  </NNMNodeList>
  <NNMInterfaceList name="List2">
    <NNMStation hostname="localhost" ovwdbPort="9999" />
      <InterfaceSelection>
         <IPInterfaceFilter>
           <IPInterface ipAddr=".*" />
         </IPInterfaceFilter>
      </InterfaceSelection>
    </NNMInterfaceList>
</NNMSimpleCustomerModel>
```

NOTE The only way to add non-IP addresses to a customer model is to add them manually to the XML file. Then, the only way to get the non-IP addresses into the portal view is to NOT set IP-specific display filtering. For more information on display filtering, see “Display Filtering: How the NodeSelection Filter Produces a List of Nodes” on page 125 and “Display Filtering Element Definitions and Examples” on page 136.

Registering the Servlet and Customer Model Data with SIP

1. Open the `OVPortalConfig.xml` file located in the following directory:

   Windows NT/2000: `<install_dir>\conf\framework\`
   UNIX: `/etc/opt/OV/SIP/conf/framework/`

2. In the `CustomerModelSources` element, enter the name of the customer model file, and enter the URL to launch the NNM SimpleCustomerModel tool. See the example below:

Figure 7-6 Example Registration for NNMSimpleCustomerModel

```
<OVPortalConfig>
```

```
  <CustomerModelSources timeout="60">
    <CustomerModel href="CustomerModel.xml)/>
    <CustomerModel
href="http://SIPhostname/ovportal/NNMSimpleCustomerModel"/>
  </CustomerModelSources>
</OVPortalConfig>
```

3. To automatically update the lists according to a schedule (in addition to each time that the servlet engine is restarted), enter a `timeout="`*`minutes`*`"` attribute, as shown in the example above.

Restarting the Servlet Engine

After changing `OVPortalConfig.xml`, you must stop and restart the servlet engine before the changes will take effect in SIP.

WindowsNT/2000:
From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**.

UNIX:

As `root`, stop and restart the web server and servlet engine by running the following. (The `DISPLAY` variable must be configured prior to restarting the webserver and servlet engine.)

Stop on HP-UX: **`/sbin/init.d/ovsip stop`**
Start on HP-UX: **`/sbin/init.d/ovsip start`**

Stop on Solaris: **`/etc/init.d/ovsip stop`**
Start on Solaris: **`/etc/init.d/ovsip start`**

Running the Program in a Browser to See Generated Data

The time required to generate the lists depends upon the size of the object database on the NNM management station. Once the list is generated, it is stored in memory on the SIP server until the next scheduled time, or until the servlet engine is restarted.

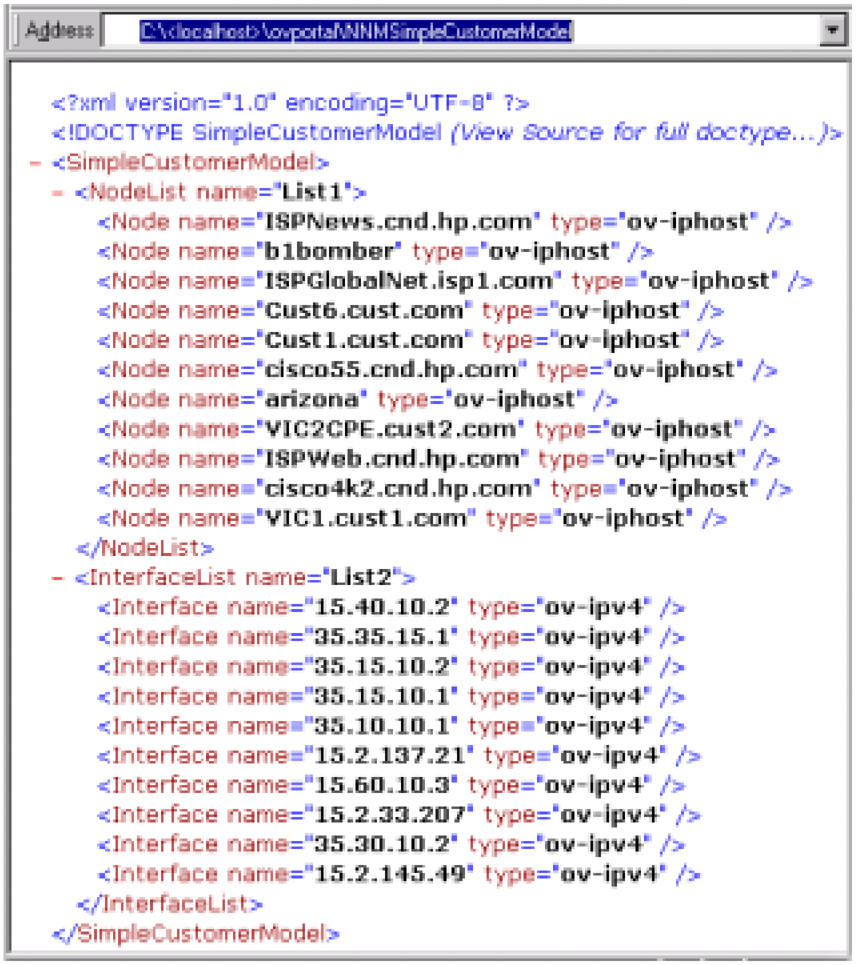
1. Open a browser window on the SIP server and type:

   `http://`*`localhost`*`/ovportal`

2. Type the following to generate and display the lists and verify that the results meet your expectations and requirements. For this tool to work, you must use the localhost:

   http://*localhost*/ovportal/NNMSimpleCustomerModel

3. Examine the XML output and verify that the results meet your expectations and requirements. The output should look something like the following example:



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SimpleCustomerModel (View Source for full doctype...)>
- <SimpleCustomerModel>
 - <NodeList name="List1">
     <Node name="ISPNews.cnd.hp.com" type="ov-iphost" />
     <Node name="b1bomber" type="ov-iphost" />
     <Node name="ISPGlobalNet.isp1.com" type="ov-iphost" />
     <Node name="Cust6.cust.com" type="ov-iphost" />
     <Node name="Cust1.cust.com" type="ov-iphost" />
     <Node name="cisco55.cnd.hp.com" type="ov-iphost" />
     <Node name="arizona" type="ov-iphost" />
     <Node name="VIC2CPE.cust2.com" type="ov-iphost" />
     <Node name="ISPWeb.cnd.hp.com" type="ov-iphost" />
     <Node name="cisco4k2.cnd.hp.com" type="ov-iphost" />
     <Node name="VIC1.cust1.com" type="ov-iphost" />
   </NodeList>
 - <InterfaceList name="List2">
     <Interface name="15.40.10.2" type="ov-ipv4" />
     <Interface name="35.35.15.1" type="ov-ipv4" />
     <Interface name="35.15.10.2" type="ov-ipv4" />
     <Interface name="35.15.10.1" type="ov-ipv4" />
     <Interface name="35.10.10.1" type="ov-ipv4" />
     <Interface name="15.2.137.21" type="ov-ipv4" />
     <Interface name="15.60.10.3" type="ov-ipv4" />
     <Interface name="15.2.33.207" type="ov-ipv4" />
     <Interface name="35.30.10.2" type="ov-ipv4" />
     <Interface name="15.2.145.49" type="ov-ipv4" />
   </InterfaceList>
</SimpleCustomerModel>
```

NOTE

If you save the output of the `NNMSimpleCustomerModel` program as an XML file, the applicable DTD is inserted into the file, unlike other customer models where you must ensure that the XML file correctly references the DTD path. For an example, see Figure 7-7 on page 102.

Figure 7-7 Sample SimpleCustomerModel.dtd Inserted Into XML Files

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SimpleCustomerModel [
<!ELEMENT SimpleCustomerModel (NodeList|InterfaceList|Error|
Warning)*>
<!ELEMENT NodeList (Node*)>
<!ATTLIST NodeList name CDATA #IMPLIED>
<!ELEMENT Node EMPTY>
<!ATTLIST Node type CDATA "ov-iphost" name CDATA #REQUIRED>
<!ELEMENT InterfaceList (Interface*)>
<!ATTLIST InterfaceList name CDATA #IMPLIED>
<!ELEMENT Interface EMPTY>
<!ATTLIST Interface type CDATA "ov-ipv4" name CDATA #REQUIRED>
<!ELEMENT Error EMPTY>
<!ATTLIST Error msg CDATA #REQUIRED>
<!ELEMENT Warning EMPTY>
<!ATTLIST Warning msg CDATA #REQUIRED>
]>
```

Creating References to the Resource Lists

1. Open*`YourCustomerModel`*`.xml`. Although the XML file can be located anywhere on the file system (as long as the path to the document is registered in `OVPortalConfig.xml`) the standard location is:

   Windows NT/2000: `<`*`install_dir`*`>\conf\organizations\`
   UNIX: `/etc/opt/OV/SIP/conf/organizations/`

2. In your `Organization` definitions, simply point to the dynamically generated lists. For example:

```
<SimpleCustomerModel>
   <Organization name="CustomerNOC">
       <NodeListRef href="List1" />
       <InterfaceListRef href="List2" />
   </Organization>
```

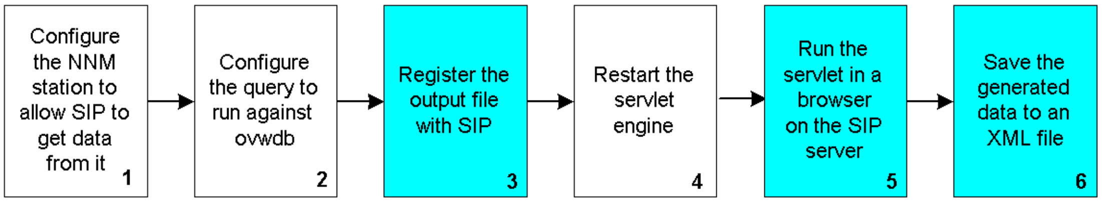
```
</SimpleCustomerModel>
```

Copying and Pasting Lists of Nodes and Interfaces into the Customer Model

If you do not want to create references to the lists, you can copy and paste the node and interface lists into the `Organization` definitions in the *`CustomerModel`*`.xml` file.

As shown below, the process is the same as "Converting NNM Object Database Information to XML" on page 97, with the exception of steps 3, 5 and 6 shown in shading below. Once you have an XML file containing the lists of nodes and interfaces, you can associate them with the correct customer (organization) by copying and pasting them into the appropriate `Organization` definitions in the *`CustomerModel`*`.xml` file.

Figure 7-8 Process of Creating a One-Time Generation of NNM Resource Lists


Configure the NNM station to allow SIP to get data from it 1
Configure the query to run against ovwdb 2
Register the output file with SIP 3
Restart the servlet engine 4
Run the servlet in a browser on the SIP server 5
Save the generated data to an XML file 6

Registering the Output File with SIP

This process is the same as "Registering the Servlet and Customer Model Data with SIP" on page 99, except that you only need to register the file in which the customer model data is saved, not the `NNMSimpleCustomerModel` servlet.

Saving the Generated Data to an XML File

After running the servlet in a browser to see the generated XML, instead of creating references to the lists, save the file and copy and paste

Copying and Pasting the Lists of Nodes and Interfaces into the Customer Model

1. Open the *`YourCustomerModel`*`.xml` file. Although the XML file can be

located anywhere on the file system (as long as the path to the document is registered in `OVPortalConfig.xml`) the standard location is:

Windows NT/2000: `<install_dir>\conf\organizations\`
UNIX: `/etc/opt/OV/SIP/conf/organizations/`

2. In your `Organization` definitions, simply paste the generated nodes and interface lists, and save the file. For example:

```
<SimpleCustomerModel>
  <Organization name="Acme" type="customer">
      <NodeList>
         <Node name="host.acme.com"/>
         <Node name="server.acme.com"/>
      </NodeList>
      <InterfaceList>
         <Interface name="15.40.10.2" type="ov-ipv4"/>
         <Interface name="35.10.10.2" type="ov-ipv4"/>
      </InterfaceList>
  </Organization>
</SimpleCustomerModel>
```

NOTE Another variation of this approach is copying and pasting the entire lists and referring to the lists using `NodeListRef` and `InterfaceListRef`, as shown in the example in step 2 on page 102.

Registering the Customer Model Data Source with SIP

You need to indicate to SIP the name and location of the sources of the customer model data.

1. Open the `OVPortalConfig.xml` file located in the following directory:

   Windows NT/2000: `<`*`install_dir`*`>\SIP\conf\framework\`
   UNIX: `/etc/opt/OV/SIP/conf/framework/`

2. Find the `CustomerModelSources` element. The four attributes of this element are described in Table 7-4.

3. Configure the `href` attribute. See the examples below.

NOTE If you specify a relative file path (e.g., "`CustomerModel.xml`"), it is interpreted relative to the `conf/share/organizations` directory.

Figure 7-9 Example of File Placed in Organizations Directory

```
<CustomerModel href="CustomerModel.xml"/>
```

NOTE The above example is correct whether the directory is configured to be local or remote. If the `/organizations` directory is actually remote, that is, the `SIP_CONF_SHARE_DIR` in `SIPPath.properties` is remote, the same syntax is used to specify it relative to the `organizations` directory.

Figure 7-10 Example of Local File in a Location Other Than the Organizations Directory

```
<CustomerModel href="c:/temp/CustomerModel.xml"/>
```

Figure 7-11 Example of Remote file

```
<CustomerModel
href="http://othermachine/CustomerModel.xml"/>
```

Figure 7-12 Example of getcvdata

```
<OVPortalConfig>
  <CustomerModelSources timeout="60">
    <CustomerModel
       href="http://NNMHostname/OvCgi/getcvdata.exe"/>
  </CustomerModelSources>
</OVPortalConfig>
```

Figure 7-13 Example of NNMSimpleCustomerModel

```
<OVPortalConfig>
  <CustomerModelSources timeout="60">
    <CustomerModel href="CustomerModel.xml)/>
    <CustomerModel
href="http://NNMhostname/ovportal/NNMSimpleCustomerModel"/>
  </CustomerModelSources>
</OVPortalConfig>
```

4. If you have placed the file in a location other than the `/organizations` directory, make sure the DTD path in the XML file is specified as described in Figure 7-4 on page 96.

Table 7-4 Attributes of the CustomerModelSources Element

Attribute	**Description**
`type`	Specifies the document type of the customer model. For SIP 2.0, "SimpleCustomerModel" is the only supported `type`.

Table 7-4 Attributes of the CustomerModelSources Element

Attribute	**Description**

`href`	A required attribute that specifies the name and location of the customer model data source file. Supported protocol values are `file`, `http`, and `https`. For syntax examples of both local and remote `href`s, see Figure 7-9 through Figure 7-11. The href is evaluated relative to the following directory:
	
	Windows 2000/NT: `*install_dir*\conf\share\organizations`
	
	UNIX: `etc/opt/OV/SIP/conf/share/organizations`
	
	Example href:
	
	`<CustomerModelSources>`
	`<CustomerModel href="CustomerModel.xml"/>`
	`</CustomerModelSources>`

`parameters`	Specifies whether the customer model provider (e.g., `getcvdata.exe`) supports queries for getting parts of the customer model. If this value is "yes", the data source must support the parameters described in Table 7-5 on page 107.
`timeout`	Specifies the timeout value in minutes for using the provided information. The customer model information is refreshed every "timeout" minutes. If the value is "0", the data source will only be loaded when the servlet engine is restarted.

Table 7-5 Parameters for getcvdata.exe

Parameter	**Description**
No parameter	If you use `getcvdata.exe` without parameters, it returns all information for all the customers known by the Customer Views server.

Table 7-5 Parameters for getcvdata.exe

Parameter	Description
`OrgList`	If you invoke `getcvdata.exe` with the `OrgList` CGI parameter, you will get a list of all the organizations and their attributes: `name`, `type`, and `ExternalKey`. However, you will not get the child information (nodes, interfaces, and services).
`Organization`	If you invoke `getcvdata.exe` with the `Organization` CGI parameter, you will get all the information for only the specified organization. `Organization="<orgname>"`

SimpleCustomerModel.dtd

The SimpleCustomerModel.dtd is located in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\conf\organizations\`
UNIX: `/etc/opt/OV/SIP/conf/organizations/`

Table 7-6 SimpleCustomerModel Child Elements

Elements	Descriptions
`Organization`	
`ServiceList`	Can be a child element of `Organization` or specified outside of any `Organization`. If a child element, `ServiceList` is a listing of services that are associated with an organization. If specified outside an `Organization`, it is a listing of services that can be associated with multiple organizations.

Table 7-6 SimpleCustomerModel Child Elements

Elements	**Descriptions**
`NodeList`	Can be a child element of `Organization` or specified outside of any `Organization`. If a child element, `NodeList` is a listing of nodes that are associated with an organization. If specified outside an `Organization`, it is a listing of nodes that can be associated with multiple organizations.
`InterfaceList`	Can be a child element of `Organization` or specified outside of any `Organization`. If a child element, `InterfaceList` is a listing of interfaces that are associated with an organization. If specified outside an `Organization`, it is a listing of interfaces that can be associated with multiple organizations.

Table 7-7 Organization Attributes and Child Elements

Elements and Attributes	**Descriptions**
`type`	An optional attribute that specifies the type of customer or organization. For Customer Views data, this would have values of "customer" and "provider."
`name`	A required attribute that specifies a unique name for identifying the organization.
`DisplayString`	An optional displayable string. If this element is not present, the `name` attribute is used for display purposes.

Table 7-7 Organization Attributes and Child Elements

Elements and Attributes	**Descriptions**
`ServiceLevel`	An optional element that specifies the level of service (e.g., "platinum," "gold," "silver," "bronze). In Customer Views, this idea is implicit in groups of customers or organizations.
`ExternalKey`	An optional element that represents an external database key for the organization. This is provided to migrate the `organizationExternalKey` attribute stored by Customer Views.

Table 7-8 ServiceList Attributes and Child Elements

Elements and Attributes	**Descriptions**
`name`	A optional attribute that specifies a unique name for identifying the `ServiceList`. The `ServiceListRef` uses the `href` attribute to reference a particular `ServiceList` by name. If multiple `ServiceList` elements have the same name value, they are viewed as the same list of services and the definitions are merged.
`Service`	
`ServiceListRef` `href`	An optional element that can be used to have multiple organizations share the same `ServiceList` definition. A `ServiceList` referenced by a `ServiceListRef` element must be identified by an `href` attribute.

Table 7-9 Service Attributes and Child Elements

Elements and Attributes	Descriptions
`type`	An optional attribute that specifies the type of service (e.g., "webHosting" and "e-mail." The `type` value is an arbitrary string defined by the application or module dealing with the service. It is used to filter on the list of services, so that a module only needs to deal with the services it is concerned with.
`name`	A required attribute that uniquely identifies the service.
`DisplayString`	An optional displayable string. If this element is not present, the `name` attribute is used for display purposes.
`Depth`	An optional element that specifies the number that represents a depth limit for displaying information about the service. This value is used by the VP Navigator module. A depth value of 0 suggests that a service is recursive.

Table 7-10 NodeList Attributes and Child Elements

Elements and Attributes	Descriptions
`name`	A optional attribute that specifies a unique name for identifying the `NodeList`. The `NodeListRef` uses the `href` attribute to reference a particular `NodeList` by name. If multiple `NodeList` elements have the same name value, they are viewed as the same list of nodes and the definitions are merged.
`Node`	

Table 7-10 NodeList Attributes and Child Elements

Elements and Attributes	**Descriptions**
`NodeListRef` `href`	An optional element that can be used to have multiple organizations share the same `NodeList` definition. A `NodeList` referenced by a `NodeListRef` element must be identified by an `href` attribute.

Table 7-11 Node Attributes and Child Elements

Elements and Attributes	**Descriptions**
`type`	An optional attribute that specifies the type of node name. This is specified for future extensibility. For SIP 2.0, only one value is understood by the modules: "ov-iphost". An "ov-iphost" type of name is a hostname, DNS name, or IP address.
`name`	A required attribute that uniquely identifies the node. If the type is "ov-iphost", the `name` value can be an IP hostname, a DNS name, or an IP address.
`DisplayString`	An optional displayable string. If this element is not present, the `name` attribute is used for display purposes.

Table 7-12 InterfaceList Attributes and Child Elements

Elements and Attributes	Descriptions
name	A optional attribute that specifies a unique name for identifying the InterfaceList. The InterfaceListRef uses the href attribute to reference a particular InterfaceList by name. If multiple InterfaceList elements have the same name value, they are viewed as the same list of interfaces and the definitions are merged.
Interface	
InterfaceListRef href	An optional element that can be used to have multiple organizations share the same InterfaceList definition. An InterfaceList referenced by an InterfaceListRef element must be identified by an href attribute.

Table 7-13 Interface Attributes and Child Elements

Elements and Attributes	Descriptions
`type`	An optional attribute that specifies the type or "address family" of the interface. Two values used are "ov-ipv4" and "ov-ifv4". An "ov-ipv4" interface has the form of an IP address in dotted notation (e.g., "15.2.5.23"). An "ov-ifv4" interface has the form "<IPHostname>/ifAlias::ifDescr", where IPhostname can be an IP hostname, DNS name, or IP address, and ifAlias, and ifDescr refer to the SNMP MIB variables of those names for the specified interface. Either ifAlias or ifDescr can be empty, but the combination of them should uniquely identify an interface on a given device.
`name`	A required attribute that uniquely identifies the interface. The format of this attribute's value is dictated by the value of the "type" attribute.
`DisplayString`	An optional displayable string. If this element is not present, the `name` attribute is used for display purposes.

Figure 7-14 Simple Customer Model Sample XML

```
<!DOCTYPE SimpleCustomerModel SYSTEM "SimpleCustomerModel.dtd">

<SimpleCustomerModel>

   <Organization type="customer" name="Cust1">
      <NodeList>
         <Node name="cisco4k2.cnd.hp.com"/>
         <Node name="15.2.3.23"/>
         <Node type="ov-iphost" name="cisco2522"/>
      </NodeList>
      <InterfaceList>
         <Interface name="15.0.2.33"/>
         <Interface type="ov-ipv4" name="15.2.3.23"/>
         <Interface type="ov-vpn4" name="10.2.1.1/3434"/>
      </InterfaceList>
   </Organization>

   <Organization type="customer" name="Cust2">
      <ServiceList>
         <Service type="webHosting" name="Cust2WebHosting"/>
         <Service type="mail" name="Cust2Mail"/>
         <Service type="news" name="Cust2News">
             <Depth>2</Depth></Service>
      </ServiceList>
      <NodeListRef href="WebServers"/>
   </Organization>

   <Organization type="customer" name="VIC1">
      <ServiceLevel>Gold</ServiceLevel>
      <ExternalKey>vic-1001</ExternalKey>
   </Organization>

   <Organization type="provider" name="PeerISP1"/>

   <NodeList name="WebServers">
      <Node name="www.cnd.hp.com"/>
      <Node name="web.fc.hp.com"/>
   </NodeList>

</SimpleCustomerModel>
```

Developing a Custom Customer Model Data Source

You can develop a program that provides a mapping from an arbitrary data store or provisioning system to the required "simple customer model" format.

SIP 2.0 has an extensible mechanism for poulating the customer model. SIP is capable of receiving content for the Customer Model from any source that can produce XML content in accordance with the `SimpleCustomerModel.dtd` file. SIP can be configured to read data from a local file, or any arbitrary URL that is accessible from the SIP station. The URL can be a servlet, CGI or other technology that returns XML content via HTTP.

For example, the Generic Net Service Provider maintains a database of their customers and their relationship to managed resources. A perl CGI script, `getMyCustomerData.pl` could be written and installed on a web server, `customerdb.generic.net`. Suppose the URL to that script is "`http://customerdb.generic.net/cgi-bin/getMyCustomerData.pl`", you could simply add the URL as a Customer Model Source in the `OVPortalConfig.xml` file as follows:

```
 <CustomerModelSources>
    <CustomerModel
       href="http://customerdb.generic.net/cgi-bin/getMyCustomerData.pl"/>
</CustomerModelSources>
```

The perl cgi script could contain some code similar to this:

```
#!/usr/local/bin/perl
...

print "content-type: text/xml\n\n";
print "<SimpleCustomerModel>";
@orgs = getOrganizationsFromDb();
foreach org (@orgs) {
      print "<Organization name=\"$org\">"
      outputOrganizationData($org);
      print "</Organization>"
}
print "</SimpleCustomerModel>";
>
```

Of course this could be written in any language using a technology that is capable of returning XML via HTTP. Parameters may be specified in the URL as well. For example, you could invoke this perl script differently from multiple SIP servers as follows:

Figure 7-15 OVPortalConfig.xml on SIP server1

```
<CustomerModelSources>
  <CustomerModel
  href="http://customerdb.generic.net/cgi-bin/getMyCustomerData.pl?server=one"/>
</CustomerModelSources>
```

Figure 7-16 OVPortalConfig.xml on SIP server1

```
<CustomerModelSources>
  <CustomerModel
  href="http://customerdb.generic.net/cgi-bin/getMyCustomerData.pl?server=two"/>
</CustomerModelSources>
```

In this case, the CGI can interpret the parameters and return different content to each server. As stated previously, this content must conform to the `SimpleCustomerModel.dtd`.

`SimpleCustomerModel.dtd` provides a mechanism for remote reporting of errors. If the CGI encounters an error or warning, it can report those along with any Customer Model Data, and they will be added to the appropriate SIP log file. For example, the CGI could return:

```
<SimpleCustomerModel>
   <Error msg="The database is currently unavailable"/>
</SimpleCustomermodel>
```

8 Filtering Data by Customers

Introduction to Customer Data Filtering

Customer data filtering is the action of applying filters to the roles that are associated with portal users in order to display only the data that is appropriate to them.

In HP OpenView Service Information Portal, the management data that is presented in the user interface for a given user login is filtered by the specific customer or customers that were associated with the user's role. (In this context, the term "customer" refers to the organizations.)

Modules That Do Customer Data Filtering

Customer data filtering applies only to the modules that display management data. It does not apply to the framework-supplied Message Board module or Bookmarks module.

You can do some filtering with Generic-based modules that you create; for more information, see Chapter 12, "Integrating Your Own Applications and Data," on page 205.

The Process of Filtering Customer Data

Customer data filtering begins at the implementation of the customer model. Recall that the customer model is the mapping of all relevant organizations to their resources. It defines the association between customers (organizations) and their nodes, interfaces, and services.

Once the customer model is implemented and you want to deploy secure customer portals, you need to associate one or more organizations with each role, and assign roles to your SIP users. This step is referred to as **security filtering**. Security filtering applies to all management data modules in the portal view that are associated with a role, and it is applied through use of the **Management Data Filter**. To restrict what a user sees, each role can be customized to have one or more associated organizations (customers).

Essentially, the customer model is a listing of your customer organizations and their associated resources. The act of selecting one or more organizations from that listing (whose data you want to display through a portal) occurs when you define the `MgmtData` element in a User-Role package file.

After security filtering is applied, you can apply a finer level of filtering on a module-by-module and submodule-by-submodule basis. This is called **display filtering**. It allows an additional level of filtering to determine what actually gets presented in a particular instance or sub-instance of a module.

For example, whereas the MgmtData filter affects all Network Device Health gauges displayed in a portal view, a display filter can be implemented on a gauge-by-gauge basis.

As a second example, you may have a portal user who is an operator in your service provider environment. This person might be responsible for monitoring the data of five customers. Their Management Data Filter is set up to define the five customers. But when they view information, they may want multiple instances of the alarms module, one for each customer. They could set up a display filter to look at information for the individual customers.

Display filtering is applied in the portal view file through three filters: **Node Selection Filter**, **Interface Selection Filter**, and **Organization Filter**.

The Node Selection and Interface Selection filters offer some support for wildcarding through the use of Perl5 regular expressions (see your Perl documentation for information about valid expressions). For example, *periods* in entries must be escaped with a backslash (\) character and the asterisks (*) must be escaped with a period (.): `.*\.eagle\.wingnuts\.com`

All three of these filters are defined in the following DTD:

Windows NT/2000: `<install_dir>\conf\share\views\filter.dtd`
UNIX: `/etc/opt/OV/SIP/conf/share/views/filter.dtd`

Types of Filtering

Service Information Portal provides three types of customer data filtering: Node filtering, interface filtering, and service filtering. All customer filtering tasks are performed through direct editing of XML files. The types of filtering are described in coming sections.

- Node Filtering
- Interface Filtering
- Service Filtering

Key Points About Customer Data Filtering

Following are key points that are essential to understanding the filtering of data that is displayed in a customer's portal. Following the key points is an explanation of the two types of filtering.

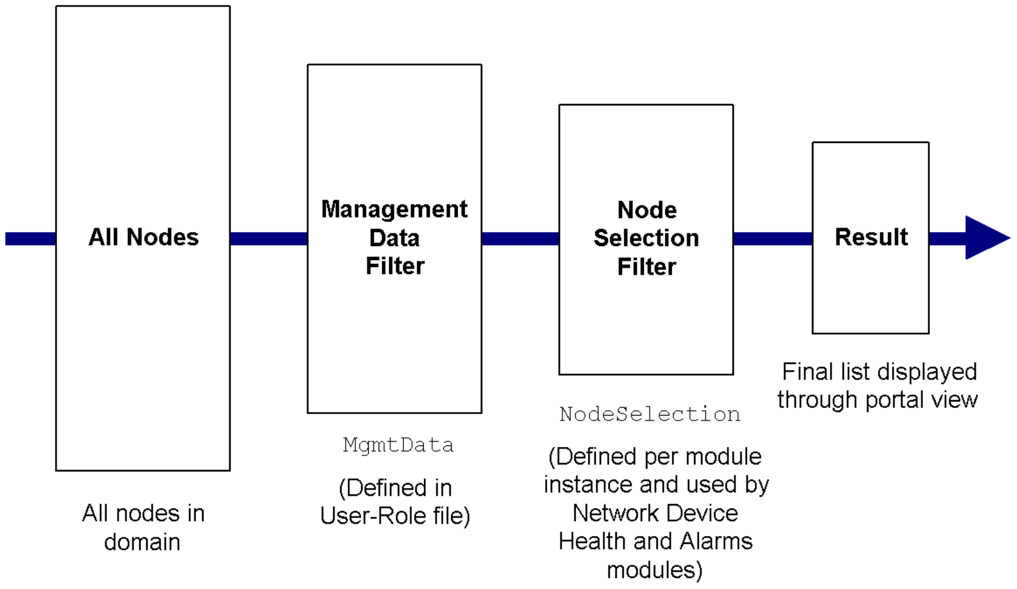
- **Two levels of filtering help you restrict the data being displayed.** Security filtering is the first (and most important) of two levels of filtering provided by the Service Information Portal. Security filtering defines what data the customer can potentially see and is applied to all management data modules. Display filtering is applied on a module-by-module or submodule-by-submodule basis and further restricts what the customer actually sees.
- **It is essential that you apply first-level filtering—security filtering.** The SIP User-Role Model requires that you apply the essential first level of filtering to each role that you create.
- **Optionally, you can apply second-level filtering—display filtering.** You can apply a finer, second level of filtering at the module or submodule level.
- **Display filtering does not apply to all three NNM modules.** The Network Health Device module requires display filtering. The Alarms module allows optional display filtering. The Topology module does not allow display filtering.

Node Filtering

With node filtering, you apply the defined filter and a list of nodes is returned. These nodes define the allowed set of information. All three NNM modules acknowledge node filtering at the security filter level (MgmtData filter). The Network Health Device module requires either or both node filtering or interface filtering at the display filter level (NodeSelection). The Alarms module allows optional display filtering. The Topology module does not allow display filtering.

Figure 8-1 below depicts what is happening in the system when node filtering is applied. First, you start with all nodes in the customer model data. When you apply the Management Data security filter, you define a set of nodes. Optionally, you can then apply the Node Selection display filter to further restrict the set of nodes and produce a result list of nodes that is used for display.

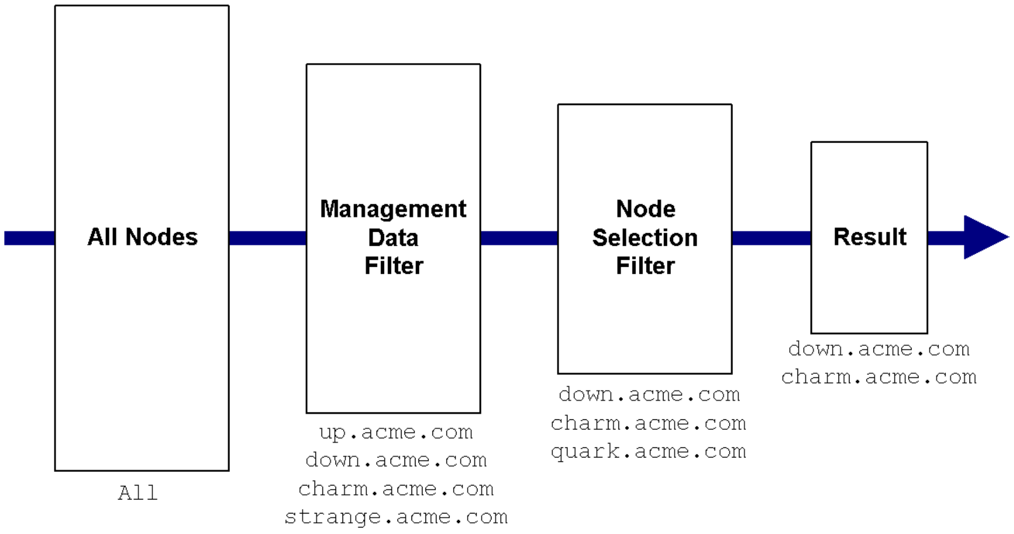
Figure 8-1 How Node Filtering Works


All Nodes
All nodes in domain
Management Data Filter
MgmtData
(Defined in User-Role file)
Node Selection Filter
NodeSelection
(Defined per module instance and used by Network Device Health and Alarms modules)
Result
Final list displayed through portal view

Domain is defined differently for different modules. For the Network Device Health module, it is all the nodes in ovwdb in the NNM management stations from which it is getting data. For the Alarms module, it is the nodes associated with generated alarms. For the Topology Map module, it is the nodes on a submap.

Figure 8-2 applies an example to the above diagram.

Figure 8-2 Node Filtering Example



Security Filtering: How the MgmtData Filter Produces a List of Nodes

For each configured Role, there is a `MgmtData` element that defines a subset of all the data that the user can see. Figure 8-3 on page 125 shows sample XML code for the Management Data Filter. For more information, see the following DTD:

Windows NT/2000: `<install_dir>\conf\share\roles\UserRole.dtd`
UNIX: `/etc/opt/OV/SIP/conf/share/roles/UserRole.dtd`

TIP **For SIP 1.0 Users**: The Management Data Filter is similar to the SIP 1.0 `SecurityFilter` except that the `IPHostFilter` and `IPInterfaceFilter` child elements are no longer supported. This means that IP host and IP interface filters cannot be directly specified in the security filter. They must instead be specified in the customer model.

Figure 8-3 **MgmtData Element Example Defining Data for Multiple Organizations**

```
<MgmtData>
   <OrganizationFilter>
      <OrganizationRef href="Cust1"/>
      <OrganizationRef href="VIC1"/>
      <OrganizationRef href="PeerISP1"/>
   </OrganizationFilter>
</MgmtData>
```

The `OrganizationRef` elements in the `OrganizationFilter` element are references to the `Organization` elements in the Simple Customer Model. Therefore, when you specify a customer in the `MgmtData` element in the User-Role file, you are saying: Present to the portal user the information filtered by the services, nodes, and interfaces associated with the Organization as defined in the Simple Customer Model.

Below is a list of rules that describe the behavior of the `MgmtData` element and its optional child element—`OrganizationFilter`

- If no `OrganizationFilter` is present, the `MgmtData` element has a value of ANY or ALL nodes.
- If `OrganizationFilter` is present but empty, the `MgmtData` element has a value of NO nodes.

Display Filtering: How the NodeSelection Filter Produces a List of Nodes

To produce a list of nodes from the `NodeSelection` Filter, three child filters can be used: `IPHostFilter`, `CapabilityFilter`, and `OrganizationFilter`. For detailed information, see “NodeSelection Filter” on page 136.

Also, see the following DTD:

Windows NT/2000: `<install_dir>\conf\share\views\filter.dtd`
UNIX: `/etc/opt/OV/SIP/conf/share/views/filter.dtd`

Interface Filtering

Interface filtering works the same way except that a list of interfaces is returned. Interface filtering provides more granularity, or a lower level of filtering. Network Device Health does interface filtering. The Topology Map does not use InterfaceSelection filters, but it does filter interfaces according to the `MgmtData` filter (security filtering). The Alarms module does not use interface filtering: If there is an interface-down event on a particular router, you cannot specify that you only want those interface events on that router. You can only specify that you want events on that router or not.

In Figure 8-4 on page 126, interface filtering is being applied using the `InterfaceSelection` filter.

Figure 8-4 How Interface Filtering Works

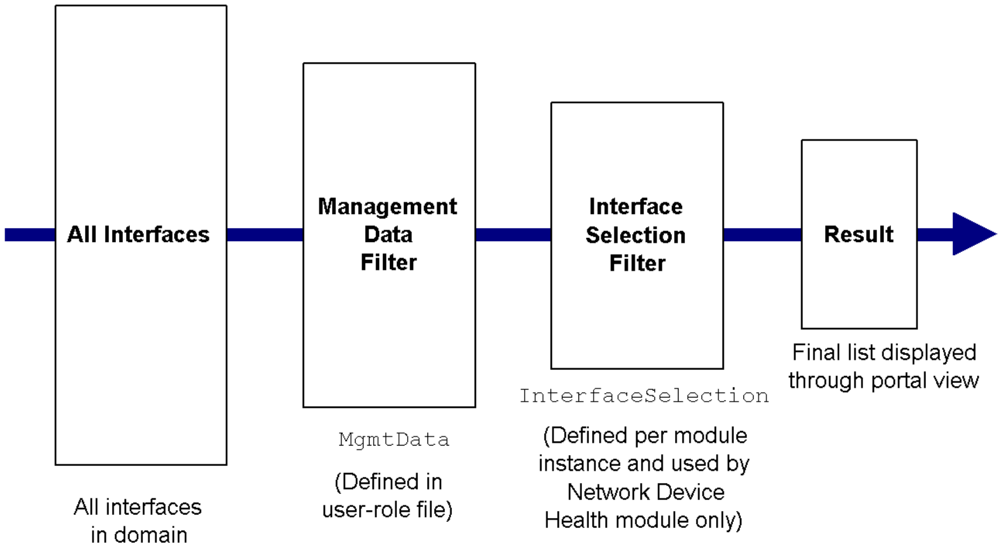
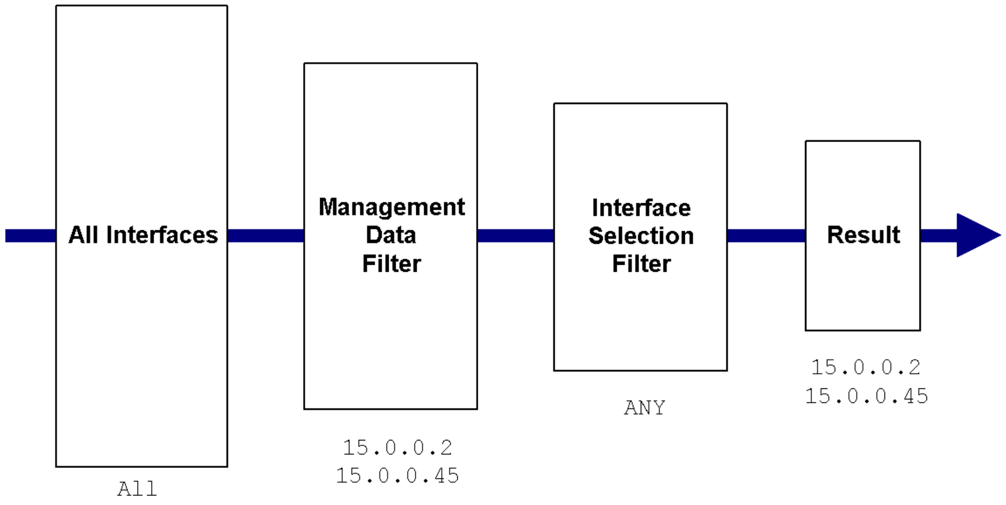

All Interfaces
Management Data Filter
Interface Selection Filter
Result
Final list displayed through portal view
InterfaceSelection
(Defined per module instance and used by Network Device Health module only)
MgmtData
(Defined in user-role file)
All interfaces in domain

Figure 8-5 on page 127 applies an example to the above diagram.

Figure 8-5 Example of Interface Filtering



Filtering of Customer Data in SIP Modules

S = Security Filter

D = Display Filter

Table 8-1 Filtering of NNM Module Data

Type of Filtering	Topology Map	Alarms	Network Device Health
Node Filtering	S -	S D	S D
Interface Filtering	S -	- -	S D
Service Filtering	- -	- -	- -

Table 8-2 Filtering of VP Navigator Module Data

Type of Filtering	Service Browser	Service Graph	Service Card	Service Health	Service Reports	Custom Service View
Node Filtering	- -	- -	- -	- -	- -	- -
Interface Filtering	- -	- -	- -	- -	- -	- -
Service Filtering	S -	S -	S -	S -	S -	S -

Filtering of VP-IS Module Data

For the VP-IS Internet Services module, a VP-IS customer name is associated with a SIP role. The VP-IS module uses the customer name to filter the data based on the VP-IS customer model.

Filters

Table 8-3

Type of Filtering	**Level of Filtering**	**Filters**	**Child Filters**
Node Filtering	Security	`MgmtData`	`OrganizationFilter` (NodeList)
	Display	`NodeSelection`	`OrganizationFilter` `IPHostFilter` `CapabilityFilter`
Interface Filtering	Security	`MgmtData`	`OrganizationFilter` (InterfaceList)
	Display	`InterfaceSelection`	`OrganizationFilter` `IPInterfaceFilter`
Service Filtering	Security	`MgmtData`	`OrganizationFilter` (ServiceList)
	Display	`ServiceRef` `GraphRef`	_

Security Filtering Element Definitions and Examples

Security filtering is defined through the `MgmtData` element in the User-Role package file. It applies to all modules that display customer resource data from the customer model.

The `MgmtData` element is stored in a User-Role package file and has one optional child element: `OrganizationFilter` (defined in the `filter.dtd` file). The Management Data Filter applies to all modules that display customer resource data from the customer model.

The `OrganizationFilter` is defined by specifying one or more organizations (customers) in the User-Role package file.

For more information, see the following DTD:

Windows NT/2000: `<install_dir>\conf\share\roles\UserRole.dtd`
UNIX: `/etc/opt/OV/SIP/conf/share/roles/UserRole.dtd`

NOTE The Role element must specify one of the following: `MgmtData`, `MgmtDataRef`, or `DefaultMgmtData`.

Table 8-4 MgmtData Element and Optional Child Element

Elements and Attributes	Description
`MgmtData`	An optional child element of the `Role` element. Used to specify the management data to be presented through the portal. Used for node, interface, and service filtering and defines what data the customer can potentially see. Can appear directly within a `Role` element, or it can be defined independently and referenced by multiple roles. `MgmtData` element is essentially a list of organizations (or customers) as defined in the customer model. The list of organizations is translated at runtime into a list of resources (services, nodes, interfaces) that determine what information is presented to the user.

Table 8-4 MgmtData Element and Optional Child Element

Elements and Attributes	Description
`name`	An optional attribute of `MgmtData`, `name` specifies a unique identifier for the `MgmtData`. This is the value used in the `href` attribute of the `MgmtDataRef` element to refer to this management data specification. Only one "management data" specification with a given name is allowed in the User-Role Model. If a subsequent one is found with the same name, it is ignored.
`orgName`	An optional attribute of `MgmtData`, `orgName` specifies the name of an organization (or customer) to use to determine the management data to present. This is provided as a shortcut so that an `OrganizationFilter` element does not need to be defined if there is only a single organization for the "management data" definition. If both the `orgName` attribute and a child `OrganizationFilter` are specified, the organization specified by `orgName` is added to the list of organizations listed in the `OrganizationFilter` element.
`OrganizationFilter`	An optional child element of `MgmtData`, `OrganizationFilter` allows multiple organizations (or customers) to be specified for the "management data." `OrganizationFilter` is used for node, interface, and service filtering. Nodes associated with an organization are those both explicitly associated with a customer and implicitly associated with the customer (through the explicit association of an interface).
`OrganizationRef`	An optional child element of `OrganizationFilter`, `OrganizationRef` refers to an organization defined in the customer model. The `href` attribute of `OrganizationRef` is a reference to a `Organization` element. It must be the same as the name attribute of a `Organization` element.

Table 8-4 MgmtData Element and Optional Child Element

Elements and Attributes	Description
`MgmtDataRef`	An optional child element of `Role`, `MgmtDataRef` allows multiple roles to refer to the same `MgmtData` definition. The referenced `MgmtData` element can be defined in any User-Role package file. The `href` attribute of `MgmtDataRef` is a reference to a `MgmtData` element. It must be the same as the `name` attribute of a `MgmtData` element.
`DefaultMgmtData`	An optional attribute of the `Role` element, `DefaultMgmtData` is a reference to the `MgmtData` element.

Figure 8-6 Example of "Acme" Organization in Customer Model: The MgmtData Element in the User-Role File References the Organization

```
<SimpleCustomerModel>
    <Organization name="Acme">
        <NodeList>
            <Node name="host.acme.com"/>
            <Node name="server.acme.com"/>
        </NodeList>
        <InterfaceList>
            <Interface name="15.40.10.2" type="ov-ipv4"/>
            <Interface name="35.10.10.2" type="ov-ipv4"/>
        </InterfaceList>
        <ServiceList>
            <Service name="email" type="email">
                <DisplayString>Email</DisplayString>
                <Depth>0</Depth>
            </Service>
            <Service name="geo_orga" type="business"/>
            <Service name="cluster" type="server"/>
        </ServiceList>
    </Organization>
</SimpleCustomerModel>
```

Figure 8-7 Sample Management Data Filters

```
<!-- This will show all management data -->

<MgmtData name="AllData"/>


<!-- This is another way to show all management data -->
<!-- It is a MgmtData element that has no "orgName" attribute and -->
<!-- It has no OrganizationFilter child element. -->

<MgmtData/>


<!-- This will show no management data -->

<MgmtData name="NoData">
     <OrganizationFilter/>
</MgmtData>


<!-- This is another way to show no management data -->
<!-- It is a MgmtData element that has no "orgName" attribute and -->
<!-- It has an empty OrganizationFilter child element. -->

<MgmtData>
     <OrganizationFilter/>
</MgmtData>


<!-- This will show data only for the "Acme" organization -->

<MgmtData name="AcmeOrg">
     <OrganizationFilter>
        <OrganizationRef href="Acme"/>
     </OrganizationFilter>
</MgmtData>


<!-- This is another way to show data only for the "Acme" organization -->
<!-- It is a MgmtData element that has an "orgName" attribute and -->
<!-- It has no OrganizationFilter child element. -->

<MgmtData orgName="AcmeOrg"/>
```

```
<!-- This will show data only for multiple organizations. -->
<!-- It is a MgmtData element that has a -->
<!-- non-empty OrganizationFilter child element. -->

<MgmtData name="GoldCustomers">
    <OrganizationFilter>
        <OrganizationRef href="Acme"/>
        <OrganizationRef href="Nabob"/>
        <OrganizationRef href="Aureum"/>
    </OrganizationFilter>
</MgmtData>
```

Figure 8-8 Sample Corresponding Role Definitions

```
<Role name="AcmeBusiness" title="Business">
   <PortalViewRef href="samples/business.xml" copy="Acme/business.xml"/>
   <EditPermission level="UserPreferences"/>
   <MgmtDataRef href="AcmeOrg"/>
</Role>


<Role name="IntegrationExamples" title="Integration Examples">
   <PortalViewRef href="samples/integration.xml"/>
   <EditPermission level="ReadOnly"/>
   <MgmtDataRef href="NoData"/>
</Role>

<Role name="Blank" title="Create A View">
   <PortalViewRef href="samples/blank.xml" copy="samples/scratch.xml"/>
   <EditPermission level="ViewAdmin"/>
   <MgmtDataRef href="AllData"/>
</Role>

<Role name="GoldNetOperator" title="Gold Customers">
   <PortalViewRef href="NetOperator.xml"/>
   <EditPermission level="UserPreferences"/>
   <MgmtDataRef href="GoldCustomers"/>
</Role>
```

Figure 8-9 Example of Role with MgmtData Element Instead of MgmtDataRef

```
<Role name="GoldNetOperator" title="Gold Customers">
   <PortalViewRef href="NetOperator.xml"/>
   <EditPermission level="UserPreferences"/>
   <MgmtData name="GoldCustomers">
      <OrganizationFilter>
        <OrganizationRef href="Acme"/>
        <OrganizationRef href="Nabob"/>
        <OrganizationRef href="Aureum"/>
      </OrganizationFilter>
   </MgmtData>
```

Figure 8-10 Example of Default Mgmt Data Specified for User-Role Package

```
<UserRolePackage title="Acme Users and Roles" defaultMgmtData="AllData">

<Role name="AcmeTechnical" title="Technical">
   <PortalViewRef href="samples/technical.xml"
   <EditPermission level="UserPreferences"/>
   <DefaultMgmtData/>
</Role>

<Role name="AcmeBusiness" title="Business">
   <PortalViewRef href="samples/business.xml"
   <EditPermission level="UserPreferences"/>
   <DefaultMgmtData/>
</Role>
```

Display Filtering Element Definitions and Examples

Whereas `MgmtData` defines what the portal customer can potentially see, the two display filters—`NodeSelection` and `InterfaceSelection`—provide a finer level of control over what the portal customer actually sees. Display filters are used individually by the network management modules and set in the portal view file or the default XML snippet for a module. Display filtering does not apply to the service management modules.

The Topology Map module does not use display filters because the node and interface filter is defined when the submap is specified. The `MgmtData` element is, essentially, applied to the content of the submap.

For more information on the `NodeSelection` and `InterfaceSelection` filters, see the following DTD:

Windows NT/2000: `<install_dir>\conf\share\views\filter.dtd`
UNIX: `/etc/opt/OV/SIP/conf/share/views/filter.dtd`

NodeSelection Filter

The `NodeSelection` element lets you further restrict the set of nodes that are displayed. Of the NNM modules, the `NodeSelection` element is used only by the Alarms module and the Network Device Health module.

`NodeSelection` filtering is useful for such things as restricting by nodes the alarms you present and restricting by network device health category (such as router health, server health, CPE health, or key device health) the gauges you present.

`NodeSelection` has three child elements:

- `OrganizationFilter` is used to restrict for display purposes the set of organizations defined by the `MgmtData` element and displayed in a portal view.

  For example, you may have defined ten customers with the `MgmtData`, but in this particular portal view, you only want to display data for two of them. For information on `OrganizationFilter`, see Table 8-4 on page 130.

- `IPHostFilter` defines a hostname or IP address.

  The `IPHostFilter` element is another child element of the `NodeSelection` element. In the `IPHostFilter`, you can list hostnames or IP addresses, but an IP address just references the node; the interface is not identified.

  When writing filters, use Perl5 regular expressions (see your Perl documentation for information about valid expressions). For example, *periods* in entries must be escaped with a backslash (\) character and the asterisks (*) must be escaped with a period (.):
  `.*\.eagle\.wingnuts\.com`

  Furthermore, specification of a host by IP address will work only if the NNM station knows the node by that same IP address (i.e., the NNM name for the node is the specified IP address). Below is an example `IPHostFilter`:

```
<NodeSelection>
   <IPHostFilter>
      <IPHost hostname="15\.2\.5\.125"/>
      <IPHost hostname="15\.2\.6\.254"/>
      <IPHost hostname="eagle\.wingnuts\.com"/>
      <IPHost hostname="hawk\.wingnuts\.com"/>
   </IPHostFilter>
</NodeSelection>
```

  An empty `IPHostFilter` yields an empty set which allows nothing to pass.

- `CapabilityFilter` defines any valid NNM object database field.

  Another child element of `IPHostFilter` is the `CapabilityFilter`. This commonly refers to the capability filters such as isRouter, isNode, etc. However, `CapabilityFilter` may refer to any NNM object database field within HP OpenView Network Node Manager (NNM). Below is an example `CapabilityFilter`:

```
<NodeSelection>
   <CapabilityFilter op="OR">
       <Capability field="IPStatus" value="Critical"/>
       <Capability value="isServer"/>
   </CapabilityFilter>
</NodeSelection>
```

  The above example will return any node that has the `isServer` capability, or is critical.

Be aware that an empty CapabilityFilter yields the empty set which allows nothing to pass.

TIP

For SIP 1.0 users: In SIP 1.0, capability filters contained two attributes: title and value. The value attribute was interpreted to be an ovwdb field of type Boolean with the value equal to "true." In SIP 2.0, the CapabilityFilter has been improved. Now it can contain a title, field, and value. title should be interpreted exactly as used in SIP 1.0. field is now the ovwdb field name. In SIP 2.0, value should not be confused with value in SIP 1.0. Now value is the value of the ovwdb field to use. If value is not set, it defaults to "true."

NOTE

When you specify all three of these filters, they are AND'd together. The result of the NodeSelection element, when all three child elements are specified, is the intersection of them.

If any of these are included in the portal view file yet left empty, they are considered an empty set which allows nothing to pass.

Table 8-5 NodeSelection Element and Three Child Elements

Elements and Attributes	Description
NodeSelection	Used only for node filtering. If none of the three child elements is present, NodeSelection has a value of any or all nodes. If any one of the three child elements is present, it defines the allowed set of nodes for NodeSelection. If more than one of the three is present, the intersection of these (only what is in common to all) defines the allowed set of nodes for NodeSelection.
IPHostFilter	Used only for node filtering. This can be a hostname or IP address. An IPHostFilter with no children is an empty set of nodes.

Table 8-5 NodeSelection Element and Three Child Elements

Elements and Attributes	Description
`CapabilityFilter`	This is any valid NNM ovwdb field. Can be used by the Alarms and Network Device Health modules. A `CapabilityFilter` with no children is an empty set of nodes. “All Capabilities” is indicated by the absence of a `CapabilityFilter` element in the `NodeSelection` element. Not supplying a capability filter in the `NodeSelection` filter results in not using node capabilities when determining the intersection of candidate nodes.
`OrganizationFilter`	Used to restrict for display purposes the set of customers defined by the `MgmtData` filter. Used only for node filtering. An `OrganizationFilter` with no children is an empty set of nodes. For more information on the `OrganizationFilter`, see Table 8-4 on page 130.

Table 8-6 NodeSelection Attributes

Elements and Attributes	Description
`title`	Descriptive name for a `NodeSelection`. Not visible to the user.
`id`	Unique identifier for a `NodeSelection`.
`op`	Logical operation to be applied if multiple children are specified. Value is "AND" (not modifiable).

Table 8-7 IPHostFilter Attributes and Child Element

Elements and Attributes	Description
`title`	Descriptive name for an `IPHostFilter`. Not visible to the user.

Table 8-7 IPHostFilter Attributes and Child Element

Elements and Attributes	Description
`id`	Unique identifier for an `IPHostFilter`.
`op`	Logical operation to be applied if multiple children are specified. Value is "OR" (not modifiable).
`IPHost`	Used only for node filtering. An `IPHostFilter` with no children is an empty set of nodes. `hostname` is an attribute of `IPHost`. It can be a hostname or IP address. Specification of a host by IP address will work only if the NNM station knows the node by that same IP address (i.e., the NNM name for the node is the specified IP address). Only ".*" is supported by the Alarms module. All other modules can use the full regular expression set. See also “NodeSelection Filter” on page 136.

Table 8-8 CapabilityFilter Attributes and Child Element

Elements and Attributes	Description
`id`	Descriptive name for a `CapabilityFilter`. Not visible to the user.
`op`	Logical operation to be applied if multiple children are specified. Value is "OR" (not modifiable).
`Capability`	Any valid NNM ovwdb field. `title` attribute is a descriptive name for the Capability and is not visible to the user. `field` attribute is the ovwdb field name. `value` attribute is the value of the ovwdb field name.

Figure 8-11 Sample NodeSelection Using a CapabilityFilter to Display All Routers that Pass the MgmtData Filter

```
<NodeSelection title="Routers" id="Routers" op="AND">
```

```
   <CapabilityFilter>
      <Capability field="isRouter" value="true"/>
   </CapabilityFilter>
</NodeSelection>
```

InterfaceSelection Filter

The `InterfaceSelection` element lets you further restrict the set of interfaces that are displayed.

The `InterfaceSelection` element is used for interface filtering by the Network Device Health module only.

The `InterfaceSelection` element has two child elements:

- `IPInterfaceFilter` is used to filter according to a list of IP Interfaces. The filter list must contain IP addresses.

  When writing filters, use Perl5 regular expressions (see your Perl documentation for information about valid expressions). For example, *periods* in entries must be escaped with a backslash (\) character and the asterisks (*) must be escaped with a period (.): `.*\.eagle\.wingnuts\.com`

  Below is an example of `IPInterfaceFilter`:

  ```
  <IPInterfaceFilter>
     <IPInterface ipAddr="15\.2\.5\.130"/>
     <IPInterface ipAddr="15\.2\.6\.54"/>
  </IPInterfaceFilter>
  ```

- `OrganizationFilter` is used to restrict for display purposes the set of organizations defined by the `MgmtData` element.

  The first of its child elements is `OrganizationFilter`, which can be used to further restrict the set of organizations whose data will be displayed in a portal view. For example, you may have defined ten customers with the `MgmtData`, but in this particular portal view, you only want to display data for two of them. For information on `OrganizationFilter`, see Table 8-4 on page 130.

Table 8-9 InterfaceSelection Element and Two Child Elements

Elements and Attributes	Description
`InterfaceSelection`	Used only for interface filtering. If neither of the two child elements is present, `InterfaceSelection` has a value of any or all interfaces. If one of the two child elements is present, it defines the allowed set of interfaces for `InterfaceSelection`. If both child elements are present, the intersection of these (only what is common to both) defines the allowed set of interfaces for `InterfaceSelection`.
`IPInterfaceFilter`	Used only for interface filtering. An `IPInterfaceFilter` with no children is an empty set of interfaces.
`OrganizationFilter`	Used to restrict, for display purposes, the set of customers defined by `MgmtData`. An `OrganizationFilter` with no children is an empty set of interfaces.

Table 8-10 InterfaceSelection Attributes

Elements and Attributes	Description
`title`	Descriptive name for `InterfaceSelection`. Not visible to the user.
`id`	Unique identifier for `InterfaceSelection`.
`op`	Logical operation to be applied if multiple children are specified. Value is "AND" (not modifiable).

Table 8-11 IPInterfaceFilter Attributes and Child Element

Elements and Attributes	Description
`title`	Descriptive name for an `IPInterfaceFilter`. Not visible to the user.
`id`	Unique identifier for an `IPInterfaceFilter`.
`op`	Logical operation to be applied if multiple children are specified. Value is "OR" (not modifiable).
`IPInterface`	`ipAddr` is an attribute for the IP address in dotted notation, either explicit or wildcarded with a Perl5 regular expression.

Figure 8-12 InterfaceSelection Example of All Interfaces That Pass the MgmtData Fitler

```
<InterfaceSelection
   <IPInterfaceFilter>
      <IPInterface ipAddr=".*"/>
   </IPInterfaceFilter>
</InterfaceSelection>
```

9 Designing Portal Views

Introduction to Portal Views

A **portal view** is a set of tabs and the configured modules that appear to a user. It also includes a set of general attributes associated with the way the portal looks to the user, such as the header, banner, "skin" (color scheme and fonts), and so forth.

When users log into the portal, they are presented with a portal view determined by their "initial role." A role defines what a user can see, as well as what the user can do through the portal at a particular point in time. Each role has associated with it:

- a portal view
- an editing permissions level
- a set of management data
- an extensible list of role properties. ("Role properties" is an extensibility mechanism used to provide to modules addition authorization information associated with a role. For more information, see “Giving Your Module Access to SIP Data Through Variable Substitution” on page 221.)

Before you can deploy customer portals, you can copy and create a portal view file and then use the portal interface to edit it. Alternatively, you can create portal view files through the direct editing of XML configuration files. When you are ready to deploy customer portals, you will create various roles, and to each role, you will associate one of the portal views (Chapter 14, “Deploying Customer Portals,” on page 263).

TIP

For SIP 1.0 users: Portal view files in SIP 2.0 (`*PortalView*.xml`) correspond in large part to the SIP 1.0 user configuration files (`*OVUserPortal*.xml`). Essentially, the file name has changed and two elements have been renamed and moved: The `SecurityFilter` element has been renamed `MgmtData` and the editing permissions attribute (`userEditable`) has changed to `EditPermission` element. Both have been moved to the User-Role package file.

Because a portal view can be shared by multiple users in SIP 2.0, a user preferences file is created when a user changes the display name in the welcome banner or the color scheme. This file is named `*login*.xml` and is stored in the `../conf/share/users/` directory.

Location of Portal View Files

Portal view files are stored in the following directory:

Windows NT/2000: `<*install_dir*>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

Sample Portal Views

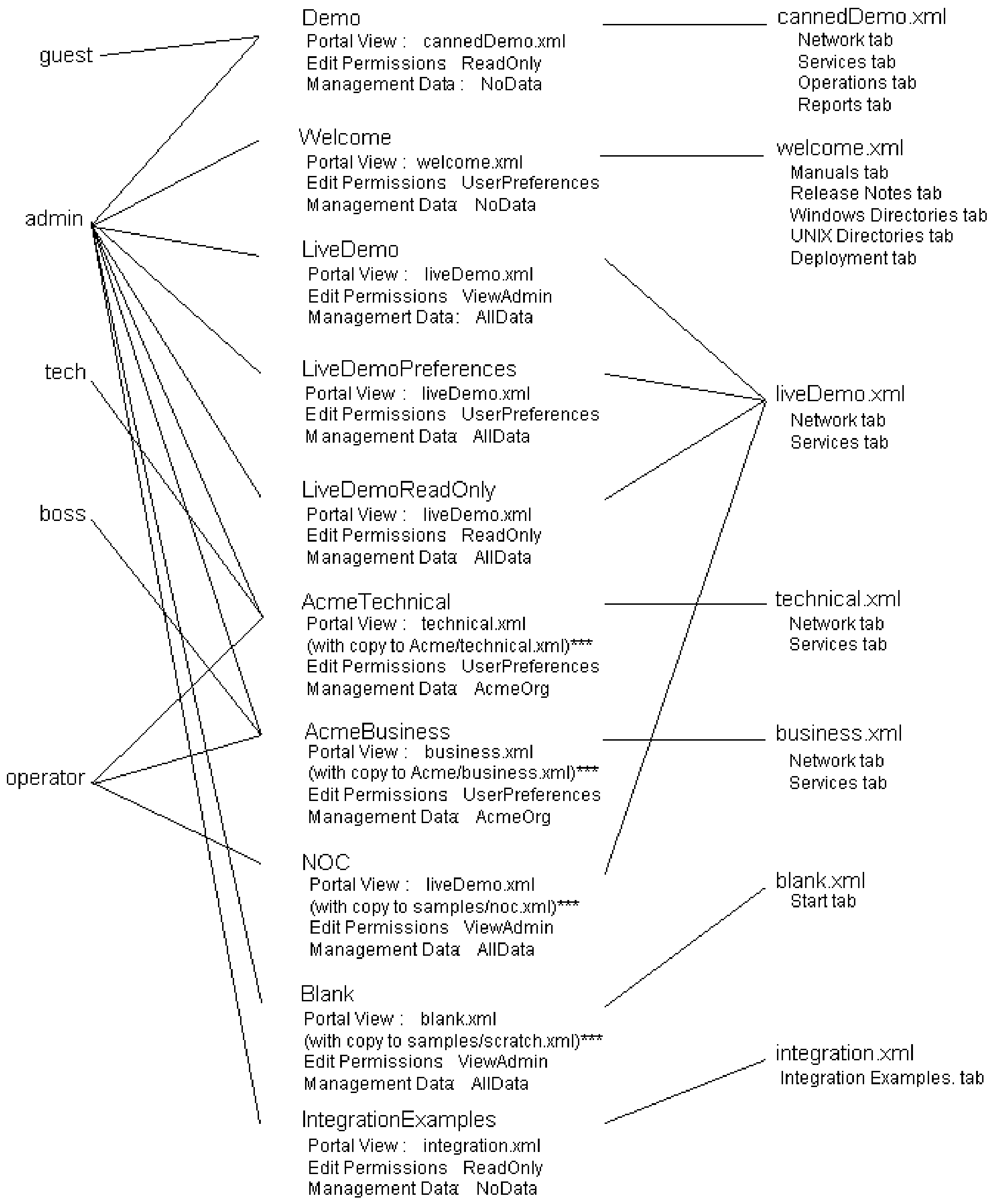
Seven sample portal views are shipped with SIP 2.0:

- `welcome.xml`
- `blank.xml`
- `business.xml`
- `cannedDemo.xml`
- `integration.xml`
- `liveDemo.xml`
- `technical.xml`

You can use these as a starting point for creating your own custom portal views. Figure 9-1 shows the user/role/view configuration samples shipped with SIP 2.0.

Notice in the diagram that three parts comprise a portal view: (1) configured modules displayed to a user, (2) tabs on which the modules are organized, and (3) a set of general attributes associated with the portal view.

Figure 9-1 Out-Of-the-Box User/Role/View Configuration Samples


guest
admin
tech
boss
operator
Demo
Portal View : cannedDemo.xml
Edit Permissions ReadOnly
Management Data : NoData
Welcome
Portal View : welcome.xml
Edit Permissions UserPreferences
Management Data NoData
LiveDemo
Portal View : liveDemo.xml
Edit Permissions ViewAdmin
Management Data : AllData
LiveDemoPreferences
Portal View : liveDemo.xml
Edit Permissions UserPreferences
Management Data AllData
LiveDemoReadOnly
Portal View : liveDemo.xml
Edit Permissions ReadOnly
Management Data AllData
AcmeTechnical
Portal View : technical.xml
(with copy to Acme/technical.xml)***
Edit Permissions UserPreferences
Management Data AcmeOrg
AcmeBusiness
Portal View : business.xml
(with copy to Acme/business.xml)***
Edit Permissions UserPreferences
Management Data AcmeOrg
NOC
Portal View : liveDemo.xml
(with copy to samples/noc.xml)***
Edit Permissions ViewAdmin
Management Data AllData
Blank
Portal View : blank.xml
(with copy to samples/scratch.xml)***
Edit Permissions ViewAdmin
Management Data AllData
IntegrationExamples
Portal View : integration.xml
Edit Permissions ReadOnly
Management Data NoData
cannedDemo.xml
Network tab
Services tab
Operations tab
Reports tab
welcome.xml
Manuals tab
Release Notes tab
Windows Directories tab
UNIX Directories tab
Deployment tab
liveDemo.xml
Network tab
Services tab
technical.xml
Network tab
Services tab
business.xml
Network tab
Services tab
blank.xml
Start tab
integration.xml
Integration Examples. tab

* All sample users and roles are configured in the `../roles/samples.xml` file with two exceptions: `guest` user and `cannedDemo` role are stored in the `default.xml` file.

** All sample portal views are stored in the `../views/samples/` directory with one exception: `welcome.xml` is stored in the `views` directory.

***Four portal views have the *copy* feature associated with them. This means that changes made to these views through the SIP interface will be saved to a separate, specified file.

Creating Portal Views

When creating portal views, you are encouraged to leverage existing ones and modify them through the portal interface, as much as possible. You can make final changes through direct editing of the XML files, if necessary.

1. Copy an existing view file.
2. Edit it using the SIP graphical interface.
3. Make final changes in the XML file.

Creating a Portal View File By Copying an Existing One

In each portal view file is a reference to the `PortalView.dtd`. This DTD is specified relative to the `conf/share/views` directory. If your portal view file is located in the `conf/share/views` directory, it looks like this:

```
<!DOCTYPE PortalView SYSTEM "PortalView.dtd">
```

If you copy portal view files between the `conf/share/views` directory and any of its subdirectories, make sure you change the reference to the `PortalView.dtd`. For example, if you place a portal view file in the `conf/share/views/samples` directory, the reference to the DTD would be:

```
<!DOCTYPE PortalView SYSTEM "../PortalView.dtd">
```

Sharing an Initial Portal View Among Multiple Roles

If you want to allow multiple roles to share an initial portal view and then have views diverge for selected roles if they are modified, you can use the "`copy`" attribute. This attribute is specified as part of the `PortalViewRef` element in the User-Role package file. For example:

```
<Role name="AcmeTechnical" title="Technical">
   <PortalViewRef href="samples/technical.xml"
                  copy="Acme/technical.xml"/>
   <EditPermission level="UserPreferences"/>
   <MgmtDataRef href="AcmeOrg"/>
</Role>
```

PortalView.dtd

The tables below list the elements and attributes you will encounter in the portal view configuration files. The `PortalView.dtd` is located in the same directory as the portal view files.

The portal view file contains the following elements:

- `PortalView`
- `Sheet`
- `Column`
- `ModuleInstance`

Table 9-1 PortalView Element

Attribute	**Description**
`userName`	The user's name as you want it displayed in the salutation on the main portal page. <string> The `userName` can be overridden by user preference and `displayName` for the user in the User-Role Model.
`refreshRate`	The number of seconds to wait before refreshing the user's portal. <positive integer, [0..(2^32)-1>
`colorScheme`	Specifies the color scheme to use when displaying the user's portal.
`defaultSheetID`	Specifies which Tab is active when the portal view is displayed. The value for this attribute should match the id of one of the Sheets defined in the portal view file. Each sheet is assigned a unique id by the management portal when it is created. <string>
`showDateTime`	Specifies whether or not to display the current time and date on the main portal page. <"yes" | "no">
`showUserName`	Specifies whether or not to display the user's name on the main portal page. <"yes" | "no"> Can be overridden by user preferences.
`portalHeader`	This attribute is not present by default. If you want to override the default header, you must add this attribute.
`portalFooter`	This attribute is not present by default. If you want to override the default footer, you must add this attribute.

Table 9-2 Sheet Element

Attribute	**Description**
`title`	Specifies the name of the tab. <string>
`id`	A unique identifier for a sheet in the portal view file. The `defaultSheetID` attribute for the `PortalView` element will be set to the id for one of the sheets defined in the portal view file. The id string MUST begin with an alpha character, not a numeric character. <string>

Table 9-3 Column Element

Attribute	**Description**
`width`	A required attribute that specifies the width of the column to be placed on the sheet. Acceptable values are (narrow | NARROW | wide | WIDE).

Table 9-4 ModuleInstance Element

Attribute	**Description**
`title`	Specifies the name of the module. This value is used as the module name in the portal view. <string>
`id`	A unique identifier for a `ModuleInstance`. This identifier is used by SIP to differentiate between instances of the same class within the portal view XML file. `id` is defined as an ID-tokenized attribute, meaning that an XML file using this attribute for this `ModuleInstance` element must specify unique text for each element. The id string MUST begin with an alpha character, not a numeric character. <string>
`classid`	The unique identifier for a module class. This string is in the form: `classid="com.hp.ov.portal.modules.alarms"` This string is used by SIP to help identify which module to load, and to generate instances of a module class. The classid for a module is defined in the module registration file *`OVModuleRegistration`*`.xml` *Windows NT/2000:* `<`*`install_dir`*`>\registration\` *UNIX:* `/etc/opt/OV/SIP/registration/`

Table 9-4 ModuleInstance Element

Attribute	Description
`help`	Specifies the URL to the help content for this module instance. This attribute allows you to override the default help URL defined in the module registration file. If you place your help topic somewhere under the `/OvSipDocs` directory, your topic will be displayed in the same decorative window in which SIP help topics are displayed. The recommended format is: `help="/OvSipDocs/C/help/<mod_directory>/<`*`topic`*`>.html"`
`rollupState`	Specifies whether the module is currently rolled "up" or rolled "down." The only thing visible on a rolled up module is its title bar and, if applicable, its instance headers. A module that is rolled down displays everything. <"up" | "down">

Customizing Tabs

With HP OpenView Service Information Portal you can create any number of tabs and customize them in a variety of ways, all depending upon how you want to present information to your customers. Following are the customization options available for tabs:

- Choose the layout of a new tab: a narrow-column format or a wide-column format. This is done from the `[Options]` button on the main portal page or by directly editing the portal view XML file.
- Choose the modules to be displayed on a tab. This is done by using the module selection list on the bottom of each tab or by directly editing the portal view XML file.
- Choose the submodules to be displayed on a tab. This is done from the `[Edit]` button on module title bars or by directly editing the portal view XML file.
- Choose the order in which tabs appear on the tab bar. This can only be done through direct editing of the XML file.

Tabs are defined in a portal view file. For a description of the XML elements, see “PortalView.dtd” on page 151.

Portal view files are located in the following directory:

Windows NT/2000:`<`*`install_dir`*`>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

Adding a Tab

There are two ways to add a tab: through the `[Options]` button on the user interface, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

1. Make a backup of the XML file before you customize or modify it.
2. Using a text editor, open the portal view file you want to modify.
3. In the XML file, the `Sheet` element is equivalent to a tab. Copy and paste a `Sheet` element from elsewhere in the file or from an another portal view file.
4. Enter a `Sheet id` that is unique among all ids in the XML file. The

`Sheet id` must start with an alpha character. For example, `"Sheet 1"`.

5. Name the new tab by typing the name in the `title` element. For example, “Network” or “Performance Data.”
6. Enter a `Column` element (required). For example, to create a narrow column, type

   ```
   <Column width="narrow">
   </Column>
   ```

   For a wide column, type

   ```
   <Column width="wide">
   </Column>
   ```
7. The `Sheet` is now syntactically complete. If this portal view is associated with a role and the role is associated with a user, you can log in and verify that the tab was added.
8. If you want to add modules, see “Adding Modules to a Portal View” on page 159.
9. After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Deleting a Tab

There are two ways to delete a tab: through the `[Options]` button on the user interface, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

1. Make a backup of the XML file before you customize or modify it.
2. Using a text editor, open the portal view file you want to modify.
3. In the XML file, the `Sheet` element is equivalent to a tab. Find the sheet that you want to delete, and delete the lines between and including the `<Sheet></Sheet>` tags.
4. Save the file. If this portal view is associated with a role and the role is associated with a user login, you can log in as the authorized user and verify that the tab was removed.
5. After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Renaming a Tab

There are two ways to rename a tab: through the `[Options]` button on the user interface, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

1. Make a backup of the XML file before you customize or modify it.
2. Using a text editor, open the portal view file that you want to modify.
3. In the XML, the `Sheet` element is equivalent to a tab. Find the sheet you want to rename, and change the value of the `title` element.
4. Save the file. If this portal view is associated with a role and the role is associated with a user login, you can log in as the authorized user and verify that the tab was removed.
5. After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Changing the Default Tab

There are two ways to change the default tab: through the `[Options]` button on the user interface, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

1. Make a backup of the XML file before you customize or modify it.
2. Using a text editor, open the portal view file you want to modify.
3. In the XML file, the `Sheet` element is equivalent to a tab. Find the sheet that you want to designate as the default tab, and copy the value of the `id` attribute of the `Sheet` element.
4. Find the `defaultSheetID` attribute of the `PortalView` element (usually near the beginning of the file), and replace the `defaultSheetID` value with the `id` attribute of the `Sheet` element that you copied.
5. Save the file. If this portal view is associated with a role and the role is associated with a user login, you can log in as the authorized user and verify that the tab was removed.
6. After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Changing the Column Format of a Tab

There are two ways to change the default tab: through the `[Options]` button on the user interface, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

1. Make a backup of the XML file before you customize or modify it.
2. Using a text editor, open the portal view file you want to modify.
3. In the XML file, the `Sheet` element is equivalent to a tab. Find the sheet that you want to modify, and then find the `Column` elements within it.
4. To change the width of a column, find the `width` attribute and change it from `narrow` to `wide` or visa versa.
5. To remove a column, delete all of the XML for the Column beginning with <`Column`> and ending with </`Column`>.
6. To add a column, enter a `Column` element. For example, to create a narrow column, type

   ```
   <Column width="narrow">
   </Column>
   ```

   For a wide column, type

   ```
   <Column width="wide">
   </Column>
   ```

7. Save the file. If this portal view is associated with a role and the role is associated with a user login, you can log in as the authorized user and verify that the tab was removed.
8. After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Changing the Display Order of Tabs

The only way to choose the order in which tabs appear on the tab bar is through direct editing of the XML file.

1. Make a backup of the XML file before you customize or modify it.
2. Using a text editor, open the portal view file you want to modify.
3. In the XML file, the `Sheet` element is equivalent to a tab. Find the

sheet that you want to reorder, and cut the lines between and including the `<Sheet></Sheet>` tags.

4. Paste the XML into another location in the file.
5. Save the file. If this portal view is associated with a role and the role is associated with a user login, you can log in as the authorized user and verify that the tab was removed.
6. After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Displaying Modules

Customizing a portal tab includes choosing the modules that will appear on the tab.

You can add modules to a tab in one of two ways: by using the drop-down list boxes located at the bottom of each portal tab or through direct editing of the portal view XML files.

Modules are defined in a portal view file. For a description of the XML elements, see “PortalView.dtd” on page 151.

Portal view files are located in the following directory:

Windows NT/2000:`<`*`install_dir`*`>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

Adding Modules to a Portal View

There are two ways to add modules to a portal view: through the selection list at the bottom of a tab, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

NOTE The module configuration that is added when you use the GUI to add modules is the default module configuration registered in the *`OVModuleRegistration`*`.xml` file, for example,

`defaultConfigXML="defaults/OVDefaultAlarms.xml"`

Registration files are stored in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\registration\`
UNIX: `/etc/opt/OV/SIP/registration/`

1. Make a backup of the XML file before you customize or modify it.
2. To add a new module to a tab, start by copying a `ModuleInstance` element from an existing portal view file.

   By copying an existing one, you help ensure that the `classid` is the correct one. Here is an example `ModuleInstance`:

```
<ModuleInstance classid="com.hp.ov.portal.modules.alarms"
 id="Alarm" rollupState="down">
     <AlarmDisplay>
     <CategoryDefName href="ErrorAlarms.xml"/>
     <CategoryDefName href="ThresholdAlarms.xml"/>
     <CategoryDefName href="StatusAlarms.xml"/>
     <CategoryDefName href="ConfigurationAlarms.xml"/>
     <CategoryDefName href="ApplicationAlertAlarms.xml"/>
     <CategoryDefName href="AllAlarms.xml"/>
     </AlarmDisplay>
</ModuleInstance>
```

NOTE As an alternative to reusing an existing `ModuleInstance`, you can copy the XML from the default module configuration file. For every module that is registered with SIP, there is a default module configuration file. The XML in this file is added to a portal view when you add a module through the user interface. The `defaultConfigXML` and its `classid` are referenced in the *`OVModuleRegistration`*`.xml` file stored in the following directory:

Windows NT/2000: <*`install_dir`*>`\registration\`
UNIX: `/etc/opt/OV/SIP/registration/`

If you use the default module configuration file XML instead of copying an existing `ModuleInstance`, you will need to wrap the XML in a `ModuleInstance` element in the portal view file and use the correct `classid` from the module registration file.

3. Paste the copied module instance into the portal view file to which you want to add the module.
4. Change the XML elements as needed.
5. Make sure that the `id` attribute value is unique among `id` values in the file. The `id` value can contain letters and numbers, but cannot start with a number.
6. Save the file. If this portal view is associated with a role and the role is associated with a user login, you can log in as the authorized user

and verify that the module was added.

7. After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Removing Modules From a Portal View

There are two ways to remove modules from a portal view: through the Close[X] button on the module titlebar, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

When you remove a module from a tab by clicking the [X] on a module title bar, the module is removed from the portal view file altogether, causing you to lose any configurations you made to the module.

A handy way to temporarily turn modules off is to set the `rollup` attribute value to "`up`".

After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Restoring Modules That Have Been Removed

If you removed a module by clicking the close button [X] on the module title bar, the module was completely removed and cannot be restored.

If you temporarily disabled a module instance by setting the `rollup` attribute value to "`up`", restore it by simply setting `rollup` to "`no`".

Changing the Display Order of Modules

There are two ways to change the display order of modules: through the selection list at the bottom of a tab, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

Modules are displayed in a portal in the order in which they appear in the portal view file. Rearranging them directly through the portal view file allows you to retain any customizations you have made to the modules. When you remove and add modules through the user interface to reorder them, you risk the chance of losing module customizations.

After you make modifications to the XML file, validate the syntax. For more information, see “Validating XML Files” on page 55.

Presenting Module Data

You can customize a module by configuring the submodules within the module. You can do this in one of two ways: from the `[Edit]` button on the title bar of a module or through direct editing of the portal view XML files. Below is a partial list of submodules provided with SIP 2.0 modules.

Table 9-5 Submodules Provided with SIP 2.0 Modules

Module	**Submodule Name in User Interface**	**Submodule Element in Portal View File**
NNM Network Device Health	Health Categories: Router Health Interface Health Key Device Health CPE Health Server Health	`<Summary>`
NNM Alarms	Alarm Categories: Error Alarms Threshold Alarms Status Alarms Configuration Alarms Application Alert Alarms	`<CategoryDefName>`
NNM Topology	Submap	`<Submap>`
Message Board	Message	`<Message>`
Bookmarks	Entry, Group, or Shared Group	`<Entry>, <Group>, <SharedGroup>`
Generic Module	*`Custom Submodule Name`*	`<Submodule>`

Adding Submodules to a Portal View

There are two ways to add submodules: through the `[Edit]` button on the module titlebar, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

Submodules are defined in a portal view file and are a child element of the corresponding module element, which is a child element of a `ModuleInstance`. For a partial list of submodules provided with SIP, see Table 9-5 on page 162. For information on a specific submodule, see the module's DTD file in the following directory:

Windows NT/2000:`<install_dir>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

Modifying Submodules

Some submodules (such as Topology Map and VPIS) can be modified through the portal GUI, while others can only be modified through direct editing of the XML files.

Changing the Display Order of Submodules

There are two ways to change the display order of submodules: through the `[Edit]` button on the module titlebar, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

Removing Submodules

There are two ways to remove submodules: through the `[Edit]` button on the module titlebar, and through direct editing of the portal view file. If you go through the user interface and need instructions, see the online help.

Specifying Other Attributes of the Portal View

In addition to tabs and modules, you can customize several other attributes of a portal view:

- The name displayed in the portal welcome banner
- The “Skins,” or color scheme
- The rate at which the portal is refreshed
- Whether a name is displayed in the portal welcome banner
- Whether the date and time are displayed in the welcome banner.

Setting User Preferences

Users who are associated with a role that has the editing permissions level set to “UserPreferences” or “ViewAdmin” will see an `[Options]` button on the portal button bar. When you access the `Options` page, you are able to set the `Display Name` and the `Skins`.

Changes made to `User Options` are stored as *`login`*`.xml` in the User Preferences directory.

Changing the User Name in the Button Bar

1. From the main portal page, click the `[Options]` button.
2. In the `User Options` box, type the name that you want displayed in the portal button bar.
3. To save and return to the main portal page, click `[OK]`.

Changing the Portal Skin

1. From the main portal page, click the `[Options]` button.
2. From the `Skins` drop-down list box, select a skin that provides the look and feel you want your portal to have.
3. To save and return to the main portal page, click `[OK]`.

Setting Options

Only the users who are associated with a role that has the editing permissions level set to “ViewAdmin” can see the `Portal Options` and `Tab Settings` sections on the `Options` page.

Changing the Refresh Rate

Refresh rate refers to the frequency (in seconds) at which the data on the portal page is regenerated to reflect the most current state of the data. Be aware that frequent refreshes can decrease portal performance.

Showing/Hiding the User Name in the Button Bar

1. From the main portal page, click the `[Options]` button.
2. In the `User Options` box, change the setting in the `Show Display Name` field.
3. To save and return to the main portal page, click `[OK]`.

Showing/Hiding the Date and Time in the Button Bar

1. From the main portal page, click the `[Options]` button.
2. In the `User Options` box, change the setting in the `Show Date/Time` field.
3. To save and return to the main portal page, click `[OK]`.

10 Using SIP-Supplied Modules

Overview of Supplied Modules

Service Information Portal provides several out-of-the-box modules. Some of the modules present information from HP OpenView management products: Network Node Manager, VantagePoint Navigator, and VantagePoint Internet Services. Some modules are third-party integrations that have been developed using SIP's Generic module. Other modules, called "general" modules, deliver content that may not be related to OpenView management products: Message Board module, Bookmarks module, and Generic module. Documentation on using general modules is covered in this manual.

HP OpenView Management Modules

Documentation on using NNM, VP Navigator, and VPIS modules are covered in separate books located under the following directory:

Windows NT/2000: `<install_dir>\htdocs\C\manuals\`
UNIX: `/opt/OV/SIP/htdocs/C/manuals/`

Table 10-1 **HP OpenView Service Information Portal 2.0 Module Documentation**

Document Title	**Filename**
Presenting NNM Data	`Presenting_NNM_Data.pdf`
Presenting VantagePoint Navigator Data	`Presenting_VPN_Data.pdf`
Presenting VPIS Data	`Presenting_VPIS_Data.pdf`

Other Integrations

Integrations of several other HP and third-party products have been developed and provided with SIP. They are located under the following directory:

Windows NT/2000: `<install_dir>\integrations\`
UNIX: `/opt/OV/SIP/integrations/`

Supplied integrations are:

- Concord
- Genesie
- Keynote
- NNM 6.2 CDP View
- NNM Commands
- NNM 6.2 Node Details
- NNM 6.2 Node Views
- NNM Reports
- NNM 6.2 Show Path
- NNM Web
- Opticom
- Problem Diagnosis
- Remedy
- Service Desk
- VP Reporter
- Webtrends
- Yahoo Headlines

See the Readme files in these directories in the integrations directory on how to register and configure these modules. Until they are explicitly registered, these modules are not available through the portal.

SIP Framework Modules

- **Message Board Module**: A way of getting important information to your customers through the portal interface. Refer to “Sending Messages to Your Customers” on page 172.
- **Bookmarks Module**: A way of integrating website links into the portal interface. Refer to “Providing Links to Other Websites” on page 179.
- **Generic Module**: A non-programmatic way to provide additional information to your customers beyond what the supplied modules offer. You can create your own full-featured modules that integrate

your own applications and data into SIP. These modules can be added and customized through the portal interface, extending the functionality of SIP without writing portal-specific Java code. For information on developing modules based on the Generic module, see Chapter 12, “Integrating Your Own Applications and Data,” on page 205.

Customizing the Help Topics for Supplied Modules

HP OpenView Service Portal provides the capability to add help for a Generic-based module. You can provide help only for the module as a whole, not for individual submodules.

When adding a Generic module to a portal view file, you can specify a URL to an HTML file that contains help for the module using the `help` attribute.

The help attribute Specifies the URL to the help content for this module instance. This attribute allows you to override the default help URL defined in the module registration file. If you place your help topic somewhere under the `/OvSipDocs` directory, your topic will be displayed in the same decorative window in which SIP help topics are displayed. The recommended location is:
`help="/OvSipDocs/C/help/<mod_directory>/<topic.html>"`

For example:

```
<ModuleInstance id="myModule
        title="MyModule"
        help="/OvSipDocs/C/help/MyDirectory/MyHelp.html">
```

Each Generic module will navigate the customer to the specified HTML help file whenever the module's "?" button is activated.

The Generic module can have multiple `Submodule` elements, hence, all the `Submodule` child elements of a single `Generic` element will reference the same HTML help file.

The `help` attribute must be a URL and can point at documentation from the local system. You are free to present a help page using the most appropriate style for the module.

If you choose not to specify a URL for help, the system will start a new browser window and display a blank page for help.

Sending Messages to Your Customers

The Message Board provides a way of getting important information to your customers. Messages are customizable through direct editing of XML files. You can create Messages that include information such as the status of problems or new services that are available to your customers.

Creating a New Message

When directly editing XML files, see “Rules for Direct Editing of XML Files” on page 54.

TIP The Message Board module does not display empty messages that are assigned to a portal view. In other words, if in a portal view you reference a message file that contains no content (the message file length (size) is zero), the message will not be displayed. You can use this feature to prepare your portal views for a future message you may want to communicate with some urgency. For example, if you assign to your portal views a message called "urgent" but leave the content blank, you can quickly disseminate information through this message in an emergency situation. Just be sure to remove the content of the message file after the emergency and not the file itself. If you remove a file that is assigned to be presented in a portal view, the portal will display the message "Data currently unavailable."

1. Create an HTML file with the contents of the message.
2. Put the file in the `messageboard` directory or in a sub-directory of the `messageboard` directory:

   Windows NT/2000:
   `<`*`install_dir`*`>\conf\share\modules\messageboard\`

   UNIX: `/etc/opt/OV/SIP/conf/share/modules/messageboard/`
3. Open the `messageDisplayNames.xml` file also located in the `messageboard` directory, and add a `Message` element that defines a title (`title` attribute) for the newly defined message file (`file` attribute), as shown in the following example:

```
<MessageDisplayNames>
<Message file="default" title="Default Message Board"/>
<Message file="newMessage.html" title="What’s New at Acme"/>
</MessageDisplayNames>
```

If your message is in a sub-directory of the `messageboard` directory, specify the file’s path relative to the `messageboard` directory, e.g., file="`subdir/mymessage`".

If you need information on the XML elements, see “messageDisplayNames.dtd” on page 177.

NOTE Notice that the file name can include the file extension or not.

4. Once the message files are in the `messageboard` directory, you can use the web interface to add, remove, rearrange them in the portal view file. Do so by clicking the `[Edit]` button on the Message Board module title bar.

Changing the Content of a Message

Be aware that the changes you make to the content of a message will affect all portal views that display the message.

Directly edit the HTML file that contains the contents of the message. Message files are located in the following directory:

Windows NT/2000:
`<`*`install_dir`*`>\conf\share\modules\messageboard\`
UNIX: `/etc/opt/OV/SIP/conf/share/modules/messageboard/`

Deleting Messages

Be aware that when you delete a message file, you need to delete it from all portal views that presently display it. If you remove a file that is assigned to be presented in a portal view, the portal will display the error message "Data currently unavailable."

When directly editing XML files, see “Rules for Direct Editing of XML Files” on page 35. If you need information on the XML elements, see “messageDisplayNames.dtd” on page 177.

1. Edit the `messageDisplayNames.xml` file to delete the message

element that you no longer want to use. The file is located in the following location:

Windows NT/2000:
`<install_dir>\conf\share\modules\messageboard\`

UNIX: `/etc/opt/OV/SIP/conf/share/modules/messageboard/`

2. Save and close `messageDisplayNames.xml`.
3. From the `messageboard` directory, delete the HTML file that contains the contents of the message.

Sending Messages to Your Customers

The first step in creating and assigning messages is to define the messages such as “Message of the Day” or “Current Problem Status Board.” This is done through direct editing of the HTML and XML files. Afterward, you can add the messages to individual portal views, as described in the following sections:

- Choosing Messages to Be Displayed In a Portal View
- Removing Messages from a Portal View
- Changing the Display Order of Messages

Choosing Messages to Be Displayed In a Portal View

There are two ways to select the messages that you want to display in a portal view: Through the Message Board Edit GUI and through direct editing of the XML files. The following instructions explain the latter. Information on the former is available from the online help.

When directly editing XML files, see “Rules for Direct Editing of XML Files” on page 54. If you need information on the XML elements, see “OVMessageBoard.dtd” on page 176 and “PortalView.dtd” on page 151.

1. Using an ASCII editor, open the portal view file that you want to display a particular message through. Portal view files are location in the following directory:

   Windows NT/2000: `<install_dir>\SIP\conf\share\views\`
   UNIX: `/etc/opt/OV/SIP/conf/share/views/`

2. Find the `MessageBoard` element, and edit it to add or delete the messages you want to display. Following is an example of a Message

Board module instance with two messages defined for the `MessageBoard` element:

```
<ModuleInstance
     classid="com.hp.ov.portal.modules.messageboard"
     help="/OvSipDocs/C/help/SIP/messageView.html"
     id="module9" rollupState="down" title="Message Board">
  <MessageBoard>
     <Message filename="default"/>
     <Message filename="Welcome"/>
  </MessageBoard>
</ModuleInstance>
```

Messages are displayed in the order in which they appear in this file.

3. Save and close the file.
4. Select a role that displays this portal view, and verify that the message was added.

Removing Messages from a Portal View

There are two ways to select the messages that you want to display in a portal view: Through the Message Board Edit GUI and through direct editing of the XML files. The following instructions explain the latter. Information on the former is available from the online help.

When directly editing XML files, see “Rules for Direct Editing of XML Files” on page 54. If you need information on the XML elements, see “OVMessageBoard.dtd” on page 176 and “PortalView.dtd” on page 151.

1. Using an ASCII editor, open the portal view file from which you want to remove a particular message. Portal view files are location in the following directory:

   Windows NT/2000: `<install_dir>\conf\share\views\`
   UNIX: `/etc/opt/OV/SIP/conf/share/views/`

2. Find the `MessageBoard` element, and delete the message filename of the message you no longer want to display through this portal view.
3. Save and close the portal view file.
4. Select a role that displays this portal view, and verify that the message was removed.

Changing the Display Order of Messages

There are two ways to select the messages that you want to display in a portal view: Through the Message Board Edit GUI and through direct editing of the XML files. The following instructions explain the latter. Information on the former is available from the online help.

When directly editing XML files, see “Rules for Direct Editing of XML Files” on page 54. If you need information on the XML elements, see “OVMessageBoard.dtd” on page 176 and “PortalView.dtd” on page 151.

1. Using an ASCII editor, open the portal view file that displays the messages you want to reorder. Portal view files are location in the following directory:

   Windows NT/2000: `<install_dir>\conf\share\views\`
   UNIX: `/etc/opt/OV/SIP/conf/share/views/`

2. Find the `MessageBoard` element, and edit it to reorder the Message elements. Messages are displayed in the order in which they appear in this file.
3. Save and close the portal view file.
4. Select a role that displays this portal view and verify that the messages appear in the correct order.

OVMessageBoard.dtd

A complete example of the `MessageBoard` module in a portal view file follows:

```
<MessageBoard>
   <Message filename="default"/>
   <Message filename="newMessage.html"/>
</MessageBoard>
```

Figure 10-1 OVMessageBoard.dtd

```
<!-- OVMessageBoard.dtd -->
<!-- Copyright (c) 2000 Hewlett-Packard Company -->
<!-- $Revision: /main/BACCHUS/1 $ -->
<!-- $Date: 2000/11/07 00:56 UTC $ -->

<!ELEMENT MessageBoard (Message)+>

<!ATTLIST MessageBoard
```

```
    id CDATA #IMPLIED>
<!ELEMENT Message EMPTY>
<!ATTLIST Message
    filename CDATA #REQUIRED>
```

Table 10-2 Message Attribute

Attribute	Description
filename	An attribute of Message, specifies the name of the file that contains the message content and that is located in the messageboard directory.

messageDisplayNames.dtd

The messageDisplayNames.xml file contains mappings between message file names and the title that will be displayed in the portal. messageDisplayNames.xml is also used as an index of available messages. The entries in this file become the list of Available Messages in the Message Board Edit GUI that is accessible when the user has ViewAdmin permissions.

A complete example of a MessageDisplayNames element follows:

```
<MessageDisplayNames>
  <Message file="default" title="Default Message Board"/>
  <Message file="newMessage.html" title="What’s New at Acme"/>
</MessageDisplayNames>
```

Figure 10-2 messageDisplayNames.dtd

```
<!-- messageDisplayNames.dtd -->
<!-- Copyright (c) 2001 Hewlett-Packard Company -->
<!-- $Revision: /main/BACCHUS/1 $ -->
<!-- $Date: 2001/01/16 20:25 UTC $ -->
<!ELEMENT MessageDisplayNames (Message*)>
<!ELEMENT Message EMPTY>
<!ATTLIST Message
    file CDATA #REQUIRED
    title CDATA #REQUIRED>
```

Table 10-3 **Message Attributes**

Attribute	**Description**
`file`	A required attribute of `Message`, specifies the name of the file that contains the message content and that is located in the `messageboard` directory. The name of `file` must match "`filename`" in the portal view XML `Message` element.
`title`	A required attribute of `Message`, specifies the message title that should be displayed in the portal interface.

Providing Links to Other Websites

In your customer portals, you can use the **Bookmarks Module** to provide a list of links to useful websites.

The Bookmarks module is customizable only through direct editing of the portal view file.

Choose Bookmarks to Be Displayed in a Customer's Portal

To modify a customer's bookmark settings, you must edit the associated portal view file. Portal view files are located under the following directory:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

When directly editing XML files, see "Rules for Direct Editing of XML Files" on page 54.

Here are three terms you need to know:

- `Entry`: A bookmark `Entry` is a single hyperlink that will be displayed in a customer's portal.
- `Group`: A bookmark `Group` is a logical grouping of entries. It consists of a `Group` element, which in turn contains a list of `Entry` elements described above. When displayed in a customer's portal, this set of entries are preceded by the name of the group.
- `SharedGroup`: A bookmark `SharedGroup` functions the same as the `Group` described above, with one distinction: instead of including the Entry list within the portal view file, a `SharedGroup` is a link to a bookmark group contained in a central location. This facilitates easy updates to many customer portals simultaneously. One may add the `SharedGroup` to a set of customers and then update the single copy to update the group for all of them.

Bookmarks are not required to be in a group. However, it may be confusing if some are contained in groups and some are not, because ungrouped entries will appear to be included within the previous group when displayed to the customer.

Adding a Bookmark Entry to a Portal View

1. Using an ASCII editor, open the portal view file to which you want to add a bookmark entry. Portal views are located in the following directory:

   Windows NT/2000: `<install_dir>\conf\share\views\`
   UNIX: `/etc/opt/OV/SIP/conf/share/views/`

2. Find the `Bookmarks` module and add the `Entry` element as shown in the following example:

   ```
   <ModuleInstance
   classid="com.hp.ov.portal.modules.bookmarks"
           display="yes" id="BookmarksModule-Demo1"
           rollupState="down">
      <Bookmarks>
         <Entry href="http://www.hp.com/e-services"
                target="bookmarkwin"
                title="Generic Net Primary Site"/>
         <Entry href="http://www.hp.com" target="bookmarkwin"
                title="HP Primary Site"/>
         <Entry href="http://www.openview.hp.com"
                target="bookmarkwin"
                title="HP OpenView Site"/>
      </Bookmarks>
   ```

   The `Entry` element has three attributes:

   - `href` - The URL link for the bookmark.
   - `title` - The name of the link to be presented in the portal view.
   - `target` - The name of a new window to create (or reuse) when opening the URL so that the current portal window is not replaced. Optional. The value of `target` should not contain spaces.

3. When you are finished, save and close the portal view file.

Grouping Entries

You can organize entries into a group under a group title. The `Group` element has one attribute, the `title`:

- `title` - Displayed before any contained `Entry` elements in an alternate style as a heading for the group.

See the example below:

```
<ModuleInstance classid="com.hp.ov.portal.modules.bookmarks"
        display="yes" id="BookmarksModule-Demo1"
        rollupState="down">
  <Bookmarks>
    <Group title="My Favorite OpenView Products">
      <Entry title="Service Information Portal"
       href="http://openview.hp.com/products/servinfoportal/index.asp"
           target="bookmarkwin"/>
      <Entry title="PolicyXpert"
        href="http://openview.hp.com/products/policyexpert/index.asp"
         target="bookmarkwin"/>
     <Entry title="Network Node Manager"
       href="http://openview.hp.com/products/nnm/index.asp"
       target="bookmarkwin"/>
    </Group>
  </Bookmarks>
```

Adding a Shared Group

There are two parts to setting up a shared group: creating the shared group entry in the sharedBookmarks.xml file, and adding a reference to the shared group into portal view files. Both are explained in the following procedure.

1. Using an ASCII editor, open the sharedBookmarks.xml file located in the following directory:

   Windows NT/2000:
   <*install_dir*>\conf\share\modules\bookmarks\
   UNIX: /etc/opt/OV/SIP/conf/share/modules/bookmarks/

2. Add a SharedGroup element with a Group child element, as shown in the following example, which shows two groups being shared:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE SharedBookmarks SYSTEM "sharedBookmarks.dtd">
<SharedBookmarks>
   <Group name="hplist" title="HP Links (Shared Group)">
      <Entry title="HP OpenView Home"
            href="http://www.openview.hp.com"
            target="hpwin"/>
      <Entry title="HP Home" href="http://www.hp.com"
             target="hpwin"/>
   </Group>
   <Group name="group1" title="Group 1 Title">
      <Entry title="entry11" href="href1"
             target="group1win"/>
```

```
        <Entry title="entry12" href="href2"
               target="group1win"/>
    </Group>
</SharedBookmarks>
```

NOTE The value of Group name is what is referenced from a portal view file as a way of designating this group as able to be shared.

3. Save and close the sharedBookmarks.xml file.
4. Open the portal view file to which you want to add a shared bookmark entry. Portal views are located in the following directory:

   Windows NT/2000: <*install_dir*>\conf\share\views\
   UNIX: /etc/opt/OV/SIP/conf/share/views/

5. Find the Bookmarks module and add the SharedGroup entry. See the example below:

```
<Bookmarks>
  <Group title="Generic Net Bookmarks">
    <Entry href="/OvSipDocs/C/demo/operations/Op_Trouble_Shooting.htm"
           target="bookmarkwin" title="Operations Troubleshooting"/>
    <Entry href="http://www.hp.com/e-services" target="bookmarkwin"
           title="Generic Net Primary Site"/>
  </Group>
  <SharedGroup name="hplist"/>
</Bookmarks>
```

There is only one attribute for a SharedGroup that is added to a portal view file:

- name - This string is used to look up the group in the sharedBookmarks.xml file.

Change the Order of Bookmarks

The order in which bookmarks are displayed is determined by the order in which they appear in the portal view file. To change the order, edit the portal view file (or, as appropriate, the sharedBookmarks.xml) to rearrange the order of bookmark Entry, Group, and SharedGroup elements.

When directly editing XML files, see “Rules for Direct Editing of XML

Files" on page 54.

Using an ASCII editor, open the portal view file to which you want to add a bookmark entry. Portal views are located in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

The `sharedBookmarks.xml` file is located in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\modules\bookmarks\`
UNIX: `/etc/opt/OV/SIP/conf/share/modules/bookmarks/`

Remove Bookmarks From a Customer's Portal

To remove bookmarks from a customer's portal, edit the portal view file and delete any `Entry`, `Group`, or `SharedGroup` that you wish to remove. You may also comment out the element by preceding it with "<!--" and ending it with "-->".

Note that if you want to change the content of a `SharedGroup`, you will need to modify it in the `sharedBookmarks.xml` file located in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\modules\bookmarks\`
UNIX: `/etc/opt/OV/SIP/conf/share/modules/bookmarks/`

When directly editing XML files, see "Rules for Direct Editing of XML Files" on page 54.

NOTE Any changes to this central file will affect all customers who are sharing this group. If you only want to modify the group for this one customer, you'll need to either copy the `Group` into the portal view file associated with the customer or you will need to created a new `SharedGroup` for this customer.

Present the Same Bookmark to All or a Set of Customers

There is no automatic way to include a bookmark for all portal views, but the `SharedGroup` feature allows the actual list of bookmarks to reside in a single shared file.

Although you can enter a bookmark or a group of bookmarks into each portal view file, the simplest way to present the same bookmark to a group of (or all) customers is to use the `SharedGroup` feature of the bookmarks module.

For instructions on creating a central `SharedGroup`, see “Choose Bookmarks to Be Displayed in a Customer’s Portal” on page 179.

Once the central `SharedGroup` has been set up, in each portal view file, add a Bookmarks module that contains a pointer to the `SharedGroup`. Use the following syntax:

```
<Bookmarks>
   <SharedGroup name="groupname"/>
</Bookmarks>
```

Modify Shared Bookmarks After Portals Have Been Created

Since shared bookmarks (i.e., `SharedGroups`) are located in a central file rather than in each portal view file, you can modify them in that one location and they are automatically changed for all users associated with that portal view (upon the next browser refresh).

To modify the shared bookmarks, edit `sharedBookmarks.xml` in the directory listed below:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\modules\bookmarks\`
UNIX: `/etc/opt/OV/SIP/conf/share/modules/bookmarks/`

For details on the syntax for this file, see “Adding a Shared Group” on page 181.

OVBookmarks.dtd

The `OVBookmarks.dtd` is stored in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

There are three valid elements for bookmarks as defined in the `OVBookmarks.dtd`:

- `Entry`: Individual bookmarks entries.
- `Group`: Contains Entry elements that are grouped under a title.
- `SharedGroup`: Points to a shared bookmark group.

Table 10-4 Entry Attributes

Parameter	**Description**
`title`	The name of the link to be presented in the customer's portal.
`href`	The URL link for the bookmark.
`target`	Optional. Provides the name of a new window to create (or reuse) when opening the URL so that the current portal window is not replaced. The value of `target` should not contain spaces.

Table 10-5 Group Attributes

Parameter	**Description**
`title`	Displayed before any contained `Entry` elements in an alternate style as a heading for the group.

Table 10-6 SharedGroup Attributes

Parameter	**Description**
`name`	The name of a `SharedGroup` listed in the portal view file. When the portal finds the matching group in the `sharedBookmarks.xml` file, it interprets the matching group as a local bookmark group. `SharedGroup` can consist of zero or more `Group` definitions.

Following is a complete module instance that can be inserted directly into a portal view file. This code is automatically inserted into a portal view when a new bookmarks module is added from the portal interface.

Figure 10-3 Default Bookmarks Module Instance

```
<ModuleInstance classid="com.hp.ov.portal.modules.bookmarks"
                help="/OvSipDocs/C/help/SIP/bookmkView.html"
                id="module10" rollupState="down" title="Bookmarks">
   <Bookmarks>
         <Entry href="http://www.hp.com/e-services"
                target="bookmarkwin"
                title="Generic Net Primary Site"/>
         <Entry href="http://www.hp.com" target="bookmarkwin"
                title="HP Primary Site"/>
         <Entry href="http://www.openview.hp.com" target="bookmarkwin"
                title="HP OpenView Site"/>
      <Group title="My Favorite OpenView Products">
         <Entry href="http://openview.hp.com/products/servinfoportal/index.asp"
                target="bookmarkwin" title="Service Information Portal"/>
         <Entry href="http://openview.hp.com/products/policyexpert/index.asp"
                target="bookmarkwin" title="PolicyXpert"/>
         <Entry href="http://openview.hp.com/products/nnm/index.asp"
                target="bookmarkwin" title="Network Node Manager"/>
      </Group>
      <SharedGroup name="hplist"/>
   </Bookmarks>
```

sharedBookmarks.dtd

The `sharedBookmarks.dtd` file is located in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\modules\bookmarks\`
UNIX: `/etc/opt/OV/SIP/conf/share/modules/bookmarks/`

There are two valid child elements of the `SharedBookmarks` element: `Group` and `Entry`. The `Entry` element is defined in the same as it is defined in the `OVBookmarks.dtd` file (Table 10-4 on page 185). The `Group` element is defined in the same as it is defined in the `OVBookmarks.dtd` file (Table 10-5 on page 185), with the exception of this additional attribute:

`Group name`: The name of a shared group as referenced in the portal view file.

Figure 10-4 sharedBookmarks.dtd

```
<!-- sharedBookmarks.dtd -->
<!-- Copyright (c) 2001 Hewlett-Packard Company -->
<!-- $Revision: /main/BACCHUS/2 $ -->
<!-- $Date: 2001/01/30 22:23 UTC $ -->
<!ELEMENT SharedBookmarks (Group*)>
<!ELEMENT Group (Entry*)>
<!ATTLIST Group name CDATA #REQUIRED
      title CDATA #REQUIRED>
<!ELEMENT Entry EMPTY>
<!ATTLIST Entry title CDATA #REQUIRED
      target CDATA "_self"
      href CDATA #REQUIRED>
```

11 Designing a Custom Look and Feel to Your Portals

Introduction

HP OpenView Service Information Portal provides several ways for you to customize the portal's overall look and feel to suit your company or that of your customers. Here are the types of customizations you can make:

- Your own banner and footer that can contain your own graphic images.
- Your own portal 'skins' (color schemes).

Customizing the Portal Header and Footer

In the header at the top of the portal page, you can display your own custom logo or that of your customer, as well as a custom background image and text.

Customizing the Header

You will use the supplied default header as the basis of the custom header. After you have defined the custom header, you will override the default header by referencing yours in the customer's portal view file.

1. Go to the directory where the default header is stored:

   Windows NT/2000: `install_dir\webapps\ovportal\jsp\core\`
   UNIX:`/opt/OV/SIP/webapps/ovportal/jsp/core/`

2. Copy and rename the default header file `header.jsp` to a custom header file name (for example, `customheader.jsp`).

   With an ASCII editor, open the new header file and modify it to change the image files or add text. (The default code for header.jsp is shown in Figure 11-1 on page 192.) The following rules apply when modifying the customheader.jsp file:

   - The *header* comprises the background image (banner), logo images, and text.
   - The `customheader.jsp` file is an HTML fragment that should NOT have `<HTML><\HTML>` or `<BODY><\BODY>` tags inserted. When editing the HTML, you need to preserve the beginning and end table tags.
   - When you make reference to images, use the following path: `<img src="/OvSipDocs/C/images/">`

     Make sure you place the image files in the above location.
   - If you want the banner to show through a logo image, make the logo image transparent.
   - The background image (color strip) in the default header is referred to as the *banner*. The default code for the banner is:

```
background="/OvSipDocs/C/images/framework/default/bg-fade.gif"
```

This default is only used if the skin is specified to be "[None]."

- Note that `class` equals "`header`" in the *`customheader`*`.jsp` file. The `header` class is defined in the cascading style sheet (CSS) files. If you want the header background image (banner) and position, as well as the text color to change based on the selected skin, `class` must equal `header`. But if you want to remove this dependency, you can remove the `class` attribute and value from your *`customheader`*`.jsp` file.

Figure 11-1 Default Code for header.jsp

```
<!-- Banner Title bar -->
<table class="header" cellpadding="10" cellspacing="0"
       border="0" width="100%" height="60"
       background="/OvSipDocs/C/images/framework/default/bg-fade.gif">
   <tr class="header">
     <td class="header" align="left" valign="middle">
       <img src="/OvSipDocs/C/images/custom/generic_net.gif"
            alt="You know it when its Generic!"
            width="141" height="43"/></td>
     <td class="header" align="right" valign="middle">
       <img src="/OvSipDocs/C/images/framework/hplogo.gif"
       alt="Hewlett-Packard" width="140" height="52"/></td></tr>
</table>
```

3. After modifying the text and image references, save and close the *`customheader`*`.jsp` file.

4. Next, insert a reference to the new header file into the portal view file of the customer to whom you want to display the customized header. With an ASCII editor, open the portal view file, which is stored in the following location:

   Windows NT/2000: `<`*`install_dir`*`>\conf\share\views\`
   UNIX:`/etc/opt/OV/SIP/conf/share/views/`

5. The `PortalView` element can contain the `portalHeader` attribute. (It is not there by default, so you may need to add it.) Type the attribute as shown in the last line of the following example:

Figure 11-2 Example of portalHeader Element Added to a Portal View File

```
<PortalView colorScheme="/OvSipDocs/styles/default.css"
            defaultSheetID="Sheet2" refreshRate="3600"
            showDateTime="yes" showUserName="yes"
            userName="Any User"
           portalHeader="customheader.jsp">
```

NOTE If you specify the wrong path and the portal cannot find the file, the portal will display the default header defined in the `OVPortalConfig.xml` file.

6. When you are finished, save the portal view file. You can expect to see the change the next time the portal view for this customer is displayed.

7. If you want to change the banner in the header for each of the skins, edit the header class definition in each CSS file that is registered in the `OVPortalConfig.xml` file. CSS files that are registered in the `OVPortalConfig.xml` appear as options in the `Skins` selection list on the `Options` page. (Figure 11-3 on page 194 shows the `header` class definition in the `default.css` file.) Different banner images can be specified for each CSS file. Place the new background images somewhere under the following directory:

   Windows NT/2000: *`install_dir`*`\SIP\htdocs\C\images\`
   UNIX: `/opt/OV/SIP/htdocs/C/images/`

   If you change the background image and change the text color, make sure the text color is well coordinated with the colors defined for the `BODY` class in the CSS file. For example, the monochrome CSS file defines a black background for the portal body. If header text is not defined as white, you will not see it.

   To perform this type of customization, you need to be proficient in Cascading Style Sheets (CSS) v.1. Visit the W3C site for more information: http://www.w3.org/Style/CSS/.

Figure 11-3 header Class in default.css

```
/*
* These are resources for the ISP-supplied header HTML.
* These styles may not work depending upon the supplied HTML.
*/
  .header {
    background-image:
                    url("/OvSipDocs/C/images/framework/default/bg-fade.gif");
    background-position: left top;
    color: black;
          }
```

8. If you want to remove the background image (banner) in the header, you must edit both the *customheader*.jsp and the CSS files. In the *customheader*.jsp file remove the code that specifies the background image. In each CSS file, remove the background-image definition and the background-position definition from the header definition.

Customizing the Footer

If you change the banner color in the header, you should consider doing the same for the footer, so that the colors are coordinated. You can change the color of the footer or place an image in it if you prefer.

The custom footer is defined the same way as the header. In a portal view file, a `portalFooter` attribute can be added to the `PortalView` element. If you want to add one, use the same procedure described for portal headers. The class value in the *customfooter*.jsp file would be "`footer`", and you would just reference the *customfooter*.jsp from the portal view file.

Customizing the Skins

To perform this task, you need to be proficient in Cascading Style Sheets (CSS) v.1. Visit the W3C site for more information:
`http://www.w3.org/Style/CSS/`

Mapping the ColorScheme Element to a CSS File

Before adding or extending existing style sheets it is important to understand the association between the `ColorScheme` name found in the `OVPortalConfig.xml` file and the cascading style sheet file.

Here is an example that illustrates the mapping:

On the `Options` page the `Skins` drop-down selection list includes `Seascape`. It appears on the selection list because it was added as a `ColorScheme` element in the `OVPortalConfig.xml` file, as shown below:

```
<ColorSchemes>
   <ColorScheme title="Seascape"
       styleSheet="/OvSipDocs/styles/blue.css">
</ColorSchemes>
```

The filename for the Seascape color scheme is `blue.css` and is stored in:

Windows NT/2000: *`install_dir`*`\SIP\htdocs\styles\`
UNIX: `/opt/OV/SIP/htdocs/styles/`

If you want to add additional items to the "Skins" selection list in the `Options` page, simply add to the existing entries in the `OVPortalConfig.xml` file and add the corresponding cascading style sheet as described below.

Creating a Cascading Style Sheet

Create a new *`mycssfile`*`.css` document and save it in the following directory:

Windows NT/2000: *`install_dir`*`\SIP\htdocs\styles\`
UNIX: `/opt/OV/SIP/htdocs/styles/`

You can add new classes to the cascading style sheets, but do not eliminate any of the existing ones. If you do you may inadvertently impact modules that rely on those classes.

Each supplied module defines the default appearance for the module. The cascading style sheets can be changed as a way of overriding the default appearance.

It may be easier to start from an existing file and make changes to it.

When creating a new skin, you may want to create or reference tabs that match your new color scheme. SIP provides a set of layered image files as a starting point for creating tabs. The format of these files is .psp (Jasc Paint Shop Pro). They are located in the following directory:

Windows NT/2000:
`install_dir\SIP\htdocs\C\images\framework\psp\`

UNIX: `/opt/OV/SIP/htdocs/C/images/framework/psp/`

NOTE If you want to change other graphical images on the portal page (other than those in the banner), you need to use a graphics editor. The interface images are stored in the following directories:

Windows NT/2000:
```
install_dir\SIP\htdocs\C\images\framework
install_dir\SIP\htdocs\C\images\health
install_dir\SIP\htdocs\C\images\service
```

UNIX:
```
/opt/OV/SIP/htdocs/C/images/framework
/opt/OV/SIP/htdocs/C/images/health
/opt/OV/SIP/htdocs/C/images/service
```

SIP relies heavily upon CSS class names to identify visual regions of the page. In Table 11-1 on page 197 is a description of each region. Refer to the CSS files for additional comments and uses.

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	**Description title**
`BODY`	Attributes for the entire page
`color`	Text color for page
`background-color`	Background color for page (can be replaced with background-image)
`font-style`	Style for text on page (normal, italic, oblique)
`font-size`	Size of text on page (point size, percentage)
`font-weight`	Thickness of text on page (normal, bold, bolder, lighter)
Below are resources for service provider header HTML. These styles may not work depending upon the supplied HTML:	
`.header`	Attributes for the header portion of the page
`background-image`	Image used for banner in header
`background-position`	Placement for banner in header (top, center, bottom, left, right)
`color`	Text color for text contained in header
Below are resources that control all title and heading styles:	
`.title`	Primary title for a page or area.
`color`	Text color in the title
`background-color`	Background color for the title
`font-style`	Style for text in the title (normal, italic, oblique)
`font-weight`	Thickness of text in the title (normal, bold, bolder, lighter)
`.subtitle`	Area beneath the primary title that looks like extension of the title (should have same settings as title)
`color`	Text color in the subtitle
`background-color`	Background color for the subtitle
`font-style`	Style for text in the subtitle (normal, italic, oblique)
`font-weight`	Thickness of text in the subtitle (normal, bold, bolder, lighter)
`.heading`	Highlighted area beneath the primary title that has a different but complementary color scheme
`color`	Text color in the heading
`background-color`	Background color for the heading
`font-style`	Style for text in the heading (normal, italic, oblique)
`font-weight`	Thickness of text in the heading (normal, bold, bolder, lighter)

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	Description title

`.subheading`
 	
`  color`
`  background-color`
`  font-style`
`  font-weight`
	Area beneath the primary heading that looks like an extension of the heading (should have same settings as heading)
	Text color for the subheading
	Background color for the subheading
	Style for text in the subheading (normal, italic, oblique)
	Thickness of text in the subheading (normal, bold, bolder, lighter)

`.heading A`
`  color`
	Anchor tag within heading
	Text color for anchor tag in heading

`.content`
`  color`
`  background-color`
	Primary data presentation area
	Text color for data presentation area
	Background color for data presentation area

`.altContent`
`  color`
	Altenative for data presentation area
	Alternative text color for data presentation

`.content A`
`  color`
	Anchor tag within primary data presentation area
	Text color for anchor tag in primary data presentation area

`.altContent A`
`  color`
	Altenative for anchor tag in data presentation area
	Alternative text color for anchor tag in data presentation

Below are resources for the rows of tabs near the top:	
`.tabBar`	The region occupied by the tabs.

`.tabBar.edgeunsel`
`  background-image`
`  background-position`
	Left edge of unselected tab
	Image for left edge of unselected tab
	Position for image of left edge of unselected tab

`.tabBar.edgesel`
`  background-image`
`  background-position`
	Left edge of selected tab
	Image for left edge of selected tab
	Position for image of left edge of selected tab

`.tabBar.unselsel`
 	
`  background-image`
`  background-position`
	Left edge of unselected tab and right edge of selected tab
	Image for edge between an unselected and selected tab
	Position for image of edge between an unselected and selected tab

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	**Description title**

`.tabBar.selunsel`
`background-image`
`background-position`
	Left edge of selected tab and right edge of selected tab
	Image for edge between a selected and unselected tab
	Position for edge between an selected and unselected tab

`.tabBar.unselunsel`
`background-image`
`background-position`
	Edges between two unselected tabs
	Image for edges between two unselected tabs
	Position for image of edge between two unselected tabs

`.tabBar.unseledge`
`background-image`
`background-position`
	Right edge of unselected tab
	Image for right edge of unselected tab
	Position for image of right edge of unselected tab

`.tabBar.seledge`
`background-image`
`background-position`
	Right edge of selected tab
	Image for right edge of selected tab
	Position for image of right edge of selected tab

`.tabBar.active`
`color`
`background-image`
`background-position`
`background-repeat`
`font-style`
`font-weight`
	Center of selected tab
	Text color for text on the selected tab
	Image for the center of the selected tab
	Position for the image of the center of the selected tab
	Handling of repetitions of the image for the center of the selected tab (repeat-x for tab appearance)
	Style for text in the heading (normal, italic, oblique)
	Thickness of text on selected tab (normal, bold, bolder, lighter)

`.tabBar.inactive`
`color`
`background-image`
`background-position`
`background-repeat`
`font-style`
`font-weight`
	Center of unselected tabs
	Text color for text on the unselected tabs
	Image for the center of the unselected tabs
	Position for the image of the center of the unselected tabs
	Handling of repetitions of the image for the center of the unselected tabs (repeat-x for tab appearance)
	Style for text in the heading (normal, italic, oblique)
	Thickness of text on unselected tabs (normal, bold, bolder, lighter)

`.tabBar A`
`text-decoration`
	Anchor tag in tabbar
	Decorations added to the anchor text (underline, overline, line-through, blink, none)

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	Description title

`.tabBar A:hover`
`text-decoration`
	Hovering mouse over anchor tag in tabbar
	Decorations added to the anchor text (underline, overline, line-through, blink, none)

Below are resources for the controls on the module titlebars:	

`.control`
`color`
`background-color`
	Primary controls used on the page
	Text color for primary controls used on page
	Background-color for primary controls used on page

Below are resources that control the toolbar that is usually immediately below the tabBar:	

`.toolbar`
`color`
`background-color`
	General toolbar area
	Text color for the general toolbar area
	Background-color for general toolbar area

`.toolbar .context`
 	
`width`
`font-weight`
 	
`padding-left`
	Display box on the left-hand side of the toolbar that contains the user name and roles drop-down
	Width of display box used for user name and roles
	Thickness of font in the user name area of toolbar (normal, bold, bolder, lighter)
	Amount of spacing on the left hand side of the display box

`.toolbar .status`
 	
`width`
`font-size`
`font-style`
`font-weight`
 	
`padding-left`
	Display box in the center of the tool bar that contains the time and date information
	Width of display box used for time and date
	Size of font for time and date (point size, percentage)
	Style for text in the heading (normal, italic, oblique)
	Thickness of font in the user name area of toolbar (normal, bold, bolder, lighter)
	Amount of spacing on the left hand side of the display box

`.toolbar .buttons`
 	
`color`
`width`
	Display box on the right-hand side of the toolbar that contains the buttons for options, logout, help, etc.
	Color for text in the buttons in the toolbar
	Width of display box used for buttons

`.toolbar .buttons .button`
 	
`color`
`width`
	Display area for the internals of an individual button in the tool bar
	Text color for label in individual button in toolbar
	Width of display area for individual button in toolbar

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	Description title
Below are resources for the entire content area occupied by the modules	

`.modules`
`border-style`
`border-color`
`border-width`
	General settings for modules
	Style for the borders surrounding the modules (solid, double)
	Width of the border surrounding modules

Below are resources for narrow columns. Typically these will be a fixed width and the trailing columns will be as big as they need:	

`.narrow`
`width`
	Specifications for narrow columns of modules
	Space to be used by the modules in narrow columns

`.wide`	Specifications for wide columns of modules
Below are resources that control the display area for an individual module including the titlebar, heading and content regions. This is used to create a border around the module or to adjust padding between modules:	

`.moduleBox`
`border-style`
`border-color`
`border-width`
	General settings for the outside of each module
	Style for the borders surrounding each module (solid, double)
	Color used for the borders surrounding each module
	Width of the border surrounding each module

Below are resources that control the heading and content areas along (without the title bar):	

`.module`
`border-style`
`border-color`
`border-width`
	General settings for the outside of each module
	Style for the borders surrounding each module (solid, double)
	Color used for the borders surrounding each module
	Width of the border surrounding each module

`.module .title`
`color`
`background-color`
`padding-left`
	Settings for the title area within each module
	Color for text in the title area of each module
	Background color for the title area of each module
	Padding on the left hand side of the title area before the text in the title begins

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	Description title
`.module .title .caption`	Settings for the caption or label on the left-hand side of the module title
`color`	Color for text in the caption within the module title
`background-color`	Background color for area containing the caption within the module title
`color`	Padding on the left hand side before caption text begins within the module title
`.module .title .controls`	Settings for the controls on the right on the right hand side of the module title
`color`	Color for text in the control area of the module title
`background-color`	Background color for the control area of the module title
`.module .subtitle`	Area beneath the module title that looks like an extension of the module title (should have same settings as module title)
`color`	Color for text in the caption within the module subtitle
`background-color`	Background color for area containing the caption within the module subtitle
`padding-left`	Padding on the left hand side before caption text begins within the module subtitle
`.module  .heading`	Highlighted area beneath the module title that contains a label for submodules
`color`	Text color in the module heading
`background-color`	Background color for the heading
`padding-left`	Padding on the left hand side before the caption text in the module heading
`.module.subheading`	Area beneath the module heading that looks like an extension of the module heading (should have same settings as module heading)
`color`	Text color in the module heading
`background-color`	Background color for the heading
`padding-left`	Padding on the left hand side before the caption text in the module heading
`.module .content`	Data presentation area within the module
`color`	Text color in the module data presentation area
`background-color`	Background color for the module data presentation area
`padding-left`	Padding on the left hand side before the module content begins

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	Description title
Below are resources for the edit pages:	

`.editarea`
`color`
`background-color`
 	
`font-size`
	Basic area on which edit controls are presented
	Text color for text presented in the edit area
	Background color for area on which edit controls are presented
	Size of fonts presented on the area where edit controls are presented (point size, percentage)

`.editarea .label`
 	
`color`
`background-color`
 	
`font-size`
 	
`font-weight`
	Highlighted area within edit area that allows for a label to separate different sections of edit controls
	Text color for text presented in the edit area
	Background color for area on which edit controls are presented
	Size of fonts presented on the area where edit controls are presented (point size, percentage)
	Thickness of fonts used in highlighted areas within edit area.

Below are resources for the modules with tables:	

`.oddrow`
`color`
`background-color`
	Settings for odd numbered rows in tables (e.g., 1, 3, 5)
	Color for text in odd numbered table rows
	Background color for odd numbered table rows

`.evenrow`
`color`
`background-color`
	Settings for even numbered rows in tables (e.g., 1, 3, 5)
	Color for text in even numbered table rows
	Background color for even numbered table rows

Below are the footer resources that control the add drop-down list and anything else at the bottom of the screen:	

`.footer`
 	
`padding`
`color`
`background-color`
	Area below the modules that contains controls for adding modules. Also Service provided footers
	Spacing surrounding the footer area
	Color for text presented in the footer area
	Background color for footer area

Below are the resources that define the Network Health background images for the gauges and detail indicators:	

Table 11-1 Cascading Style Sheet Classes and Descriptions

Class name	**Description title**
`.NHGauge .gauge`	The high-level gauge in the Network Device Health module.
`background-image`	Image used to present 3D image of gauge on which values are presented
Below are the NHdetail resources that correspond directly with the health ratings defined in the <Rating> tag of the netHealthConfig.xml file:	
`.NHdetail .normal`	Presentation controls for objects in Network Device Health detail tables that have normal health
`background-image`	Image used in detail tables to represent objects with normal health
`background-position`	Position for image within the detail table cell
`.NHdetail .minor`	Presentation controls for objects in Network Device Health detail tables that have minor severity health
`background-image`	Image used in detail tables to represent objects with minor severity health
`background-position`	Position for image within the detail table cell
`.NHdetail .critical`	Presentation controls for objects in Network Device Health detail tables that have critical severity health
`background-image`	Image used in detail tables to represent objects with critical severity health
`background-position`	Position for image within the detail table cell
`.NHdetail .unknown`	Presentation controls for objects in Network Device Health detail tables that have unknown health
`background-image`	Image used in detail tables to represent objects with unknown health
`background-position`	Position for image within the detail table cell

12 Integrating Your Own Applications and Data

Chapter Contents

Introduction and Tutorial

In addition to presenting NNM, VP Navigator, and VP-IS information to your customers, you can integrate other applications and data through use of the Generic module.

The **Generic module** is a non-programmatic way to easily and quickly incorporate into SIP your existing web applications, reports, and other data. You can create your own full-featured modules that can be added and customized through the portal interface, extending the functionality of SIP without writing portal-specific Java code.

Using the Generic module based on a simple specification in an XML file, you can create a module that is configurable through a user interface.

The Generic module also provides limited proxying capabilities for the URLs. This way, you can integrate into portal web sites that are behind a firewall and are not accessible to your end customer.

The Generic module offers you the following capabilities:

- **Display content from a specified URL**. For example, if you have an HTML file on a different server or a CGI program or other web application, you can display the contents of the URL by referencing the URL in a portal view file.
- **Display output from executable commands**. For example, if you want to run a command, such as `ping` or a command that generates results from a database, and display the output of the command in the portal, you can enter the command in a portal view file.
- **Display HTML from an external file**. For example, if you have a report in an HTML file on the local machine or a tool that generates reports in the form of text, you can display those reports through the portal by referencing the HTML file in a portal view file.
- **Display embedded HTML**. For example, if you have a tool that generates reports in the form of a GIF image, you can display the report through SIP by embedding the HTML in a portal view file.

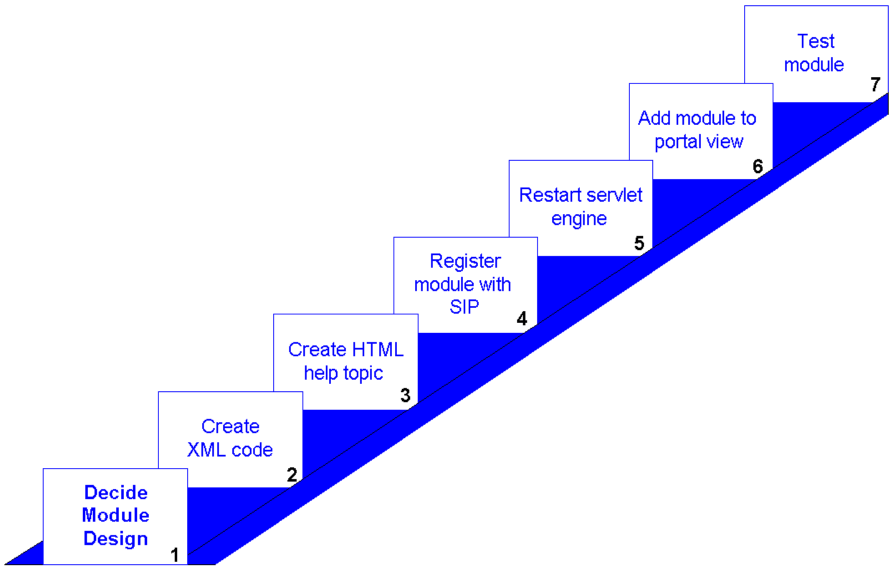
For examples of the Generic module, log into SIP as `admin` user and switch to the `Integration Examples` role or `Demo` role. All modules are based on the Generic module.

TIP

For SIP 1.0 users: Generic modules can now be added through the portal interface and customized through an Edit GUI. In SIP 2.0, you can use the Generic module to create modules that have all of the features of a supplied, full-features module.

Process of Creating Integrated Modules

The diagram below summarizes the process of creating integrated SIP modules based on the Generic module.



Going through the Process: A Tutorial

You can easily understand the basic process of creating integrated modules by following this short tutorial. You are instructed to take a supplied, sample module that is based on the Generic module (in lieu of creating XML code, Step 2), register it with SIP (Step 4), restart the servlet engine (Step 5), and add it to a portal view (Step 6). The sections following the tutorial describe the steps in detail.

Create the XML Code

For this tour, you will use an existing default module file. Provided with SIP are several sample integrations that are based on the Generic module. Instead of developing code for this exercise, you will use one of the sample integrations. (The thing to note is that default module code must be placed in a specific location.)

The default module file contains a snippet of XML code that makes up a complete Generic-based module. This code is added to a portal view file when you add a module through the portal interface.

Register the Module with SIP

1. Go to the following directory where the sample integration code and sample registration file are stored:

   Windows NT/2000:
   `<`*`install_dir`*`>\integrations\Yahoo_Headlines`

   UNIX: `/opt/OV/SIP/integrations/Yahoo_Headlines`

2. Copy the files `OVDefaultYahooHeadlines.xml` and `OVRegYahooHeadlines.xml` into the following directories, which are required locations:

   - `OVDefaultYahooHeadlines.xml`

     Windows NT/2000: `<`*`install_dir`*`>\registration\defaults`
     UNIX: `/etc/opt/OV/SIP/registration/defaults`

   - `OVRegYahooHeadlines.xml`

     Windows NT/2000: `<`*`install_dir`*`>\registration\`
     UNIX: `/etc/opt/OV/SIP/registration/`

NOTE The registration file defines such information as a unique ID for the module, a reference to the servlet, the module's associated help file, and module's capabilities.

Restart the Servlet Engine

After adding or changing a module registration file, you must stop and restart the servlet engine before the changes will take effect in SIP.

WindowsNT/2000:
From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**.

UNIX:

As `root`, stop and restart the web server and servlet engine by running the following. (The `DISPLAY` variable must be configured prior to restarting the webserver and servlet engine.)

Stop on HP-UX: **`/sbin/init.d/ovsip stop`**
Start on HP-UX: **`/sbin/init.d/ovsip start`**

Stop on Solaris: **`/etc/init.d/ovsip stop`**
Start on Solaris: **`/etc/init.d/ovsip start`**

Add the Module to a Portal View

1. Start SIP by opening a browser and entering the following: **`http://`*`yourhostname.com`*`/`**`ovportal`
2. Log in as **`admin`**.
3. From the Role selection list in the portal banner, select `Create A View`. Go to the [`Add`] button at the bottom of the wide tab column.
4. Select `Yahoo Headlines` and click [`Add`].

Resources for Creating Generic-Based Modules

Once you know the process for creating Generic-based modules, all you need is the supplied sample code and an understanding of the DTD. Below is a listing of resources and where you can find these.

Sample registration file

- *Windows NT/2000:*

```
<install_dir>\integrations\Yahoo_Headlines\OVRegYahooHeadlines.xml
```

- *UNIX:*

```
/opt/OV/SIP/integrations/Yahoo_Headlines/OVRegYahooHeadlines.xml
```

Sample Generic-based modules of Third-Party Integrations

Concord, Keynote, NNM Commands, NNM Reports, Opticom, Service Desk, VP Reporter, Webtrends, and Yahoo Headlines:

Windows NT/2000: `<`*`install_dir`*`>\integrations\`
UNIX: `/opt/OV/SIP/integrations/`

Location of Default Module files

Windows NT/2000: `<`*`install_dir`*`>\registration\defaults\`
UNIX: `/etc/opt/OV/SIP/registration/defaults/`

Sample Portal View files

`cannedDemo.xml` and `integration.xml` located here:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\views\samples\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/samples/`

DTD and Element Definitions

`OVGeneric.dtd` located here:

Windows NT/2000: `<`*`install_dir`*`>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

`OVModuleRegistration.dtd` located in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\registration\`
UNIX: `/etc/opt/OV/SIP/registration/`

Rules for Editing XML Files

“Rules for Direct Editing of XML Files” on page 54.

Designing the Functionality of Your Module

Before you create a Generic-based module, you need to decide on the functionality you want to build into it. Answer for yourself the following questions:

- Do you want to provide for editing of the module configuration from the portal interface?

  For example, you can provide an editing GUI with a choice list of module parameters. You can allow a person with `ViewAdmin` edit permissions to make these types of module configurations. For more information on edit permissions, see Table 14-10 on page 284.

- If you provide an edit GUI, what parameters will you offer?

  For example, parameters can be hosts, various reports, different commands, and for the different commands ability to have different parameters.

- Will the parameters be selected from a options list, or will they be entered into a text field?

  For example, you can create a selection list of commands, or allow the text entry of a command.

- Will your module have multiple submodules?
- Because editing GUIs apply to the entire module and not individual submodules, how will you design the editing interface appropriately for all submodules?
- Do you want to develop an online help topic for your module?

  If you do, keep in mind that, like edit GUIs, the help topic applies to the entire module and not individual submodules.

NOTE To learn more about the kinds of choices you have when designing a Generic-based module, refer to the examples in “Displaying the Output from a Command” on page 214. Four examples illustrate the output of the "ping" command displayed through the portal; but each example uses a different approach.

Creating XML Code for a Generic-Based Module

For information and sample code that is specific to each type of submodule, see:

- "Displaying the Output from a Command" on page 214
- "Displaying HTML from an External File" on page 216
- "Displaying Embedded HTML" on page 218
- "Displaying the Contents of a URL" on page 219

The process of creating the XML code for a Generic-based module can be summed up in the following steps:

1. Find an example of a Generic-based module that is similar to the one you want to create. Make sure you copy the entire `Generic` element.

NOTE For the location of example modules, see "Sample Generic-based modules of Third-Party Integrations" on page 211 and "Sample Portal View files" on page 211. You can also create a module from scratch. For detailed information on the XML elements and attributes, see the documentation that begins on page 237.

2. Copy and rename the XML file that contains the example. Place the file in the defaults directory where the default module code for each registered module is stored:

   Windows NT/2000: `<install_dir>\registration\defaults\`
   UNIX: `/etc/opt/OV/SIP/registration/defaults/`

3. Using an ASCII editor, open the default file and modify the XML code. In addition to the four references cited at the top of this page, the following sections in this chapter describe the functionality that you can add:

   - "Giving Your Module Access to SIP Data Through Variable Substitution" on page 221
   - "Adding an Edit GUI to Your Module" on page 226

- "Adding Status Text, Graphics, and Links to a Submodule Title" on page 227

4. Save the file to the following directory:

   Windows NT/2000: `<install_dir>\registration\defaults\`
   UNIX: `/etc/opt/OV/SIP/registration/defaults/`

Displaying the Output from a Command

Through SIP, you can display the output from program execution. In fact, you can capture anything generated to `stdout`. For example, if you want to run a command, such as `ping` or a command that generates results from a database, and display the output of the command in a portal, you can achieve this through the Generic-based module.

To display the command output, you will define the `Command` element in a Generic element. See the examples in Figure 12-1 through Figure 12-10 and the element definitions in Table 12-5 on page 241.

CAUTION Be careful when presenting the results of a command. A command is executed as:

- *Windows NT/2000:* the local "System" account
- *HP-UX:* `www/www`
- Solaris: `nobody/nogroup`

Make sure that only knowledgeable personnel implement this feature.

Figure 12-1 Example Ping Command

```
<Generic>
  <Submodule>
    <TitleBar title="Ping Command"/>
    <Command commandLine="ping.exe -n 3 localhost"
     expires="5" stripHtmlHeader="no" type="text/plain"/>
  </Submodule>
</Generic>
```

Figure 12-2 Example Ping Command: Three Machines

```
<Generic>
  <Submodule>
    <TitleBar title="Ping Local Host"/>
    <Command commandLine="ping.exe -n 3 localhost"
      expires="5" stripHtmlHeader="no" type="text/plain">
    </Command>
  </Submodule>

  <Submodule>
    <TitleBar title="Ping weminuche"/>
    <Command commandLine="ping.exe -n 3 weminuche.cnd.hp.com"
      expires="5" stripHtmlHeader="no" type="text/plain">
    </Command>
  </Submodule>

  <Submodule>
    <TitleBar title="Ping fcbeyond"/>
    <Command commandLine="ping.exe -n 3 fcbeyond.cnd.hp.com"
      expires="5" stripHtmlHeader="no" type="text/plain">
    </Command>
  </Submodule>
</Generic>
```

Figure 12-3 Example Ping Command with Editing GUI: Option Parameter

```
<Generic>
  <Submodule>
    <TitleBar title="Ping Command"/>
    <Command commandLine="ping.exe -n 3 $HOSTNAME"
      expires="5" stripHtmlHeader="no" type="text/plain">
      <OptionParm name="HOSTNAME"
        prompt="Choose Ping Target" value="Local Host">
          <Option name="Local Host" value="localhost"/>
          <Option name="weminuche"
                  value="weminuche.cnd.hp.com"/>
          <Option name="fcbeyond"
                 value="fcbeyond.cnd.hp.com"/>
      </OptionParm>
    </Command>
  </Submodule>
</Generic>
```

CAUTION Be aware that Figure 12-3 on page 215 and Figure 12-4 below are example only; for security reasons, you may not want to offer a text parameter or options parameters for commands.)

Figure 12-4 **Example Ping Command with Editing GUI: Text Parameter**

```
<Generic>
  <Submodule>
    <TitleBar title="Ping Command"/>
      <Command commandLine="ping.exe -n 3 $HOSTNAME"
        expires="5" stripHtmlHeader="no" type="text/plain">
      <TextParm name="HOSTNAME" prompt="Ping Target"
        value="localhost"/>
      </Command>
  </Submodule>
</Generic>
```

Displaying HTML from an External File

Through SIP, you can display HTML from an external file. For example, if you have a tool that generates reports in the form of a GIF image or text, you can display that report through the portal by referencing the HTML file.

To display HTML from an external file, you will define the `File` element in a Generic element. See the examples in Figure 12-5 on page 217 and the element definitions in Table 12-6 on page 244.

Rules for External HTML

- The file must be on your local machine or network. It cannot require http access. If you want to display files on a remote system, then refer to “Displaying the Contents of a URL” on page 219.
- The file can be HTML or plain text. The `type` should be set to "`text/html`" when the content is HTML. The `type` should be set to “`text/plain`” when the content is plain text.
- The HTML should be well formed.
- The contents of an external HTML file should not include "header" tags such as `<HTML></HTML>`, `<HEAD></HEAD>`, or `<BODY></BODY>`. The portal will automatically provide these tags. If your HTML files include these tags, you can use `stripHTMLHeader`. For more information about `stripHTMLHeader`, refer to Table 12-5 on page 241.

- `fileName` ("/rootdir/*filename*.htm") specifies the file.
- `displayFileInfo` configures a Generic-based module to display the full path to the file and the time it was last modified in the browser. This information will precede the file content.
- `stripHTMLHeader` is used to strip the <HEAD></HEAD> contents before displaying the file. This is useful if the imported file contains a <HEAD> section and the customer's browser cannot handle the reception of multiple <HEAD> elements (one from the portal and one from the file). "no" allows scripts to run within the HEAD tag.
- If you have style sheets in a file, your imported style sheets may be ignored. If you embed content from a URL that defines a cascading style sheet, it will override the SIP cascading style sheet.

Figure 12-5 Example: Displaying HTML from an External File from cannedDemo.xml

```
<Generic>
  <Submodule>
    <TitleBar titleAnchorText="Router Health"
       titleAnchorURL="/OvSipDocs/C/demo/router_details.html"/>
    <File displayFileInfo="no"
      fileName="$SIP_HOME_DIRhtdocs/C/demo/router_health.html"
     stripHtmlHeader="yes" type="text/html"/>
  </Submodule>
</Generic>
```

Figure 12-6 Example of Displaying External HTML Files With Editing GUI: Option Parameters

```
<Generic>
  <Submodule>
    <TitleBar titleAnchorText="Employee Files"/>
    <File displayFileInfo="no"
      fileName="$SIP_HOME_DIRhtdocs/C/demo/$EmpFiles"
      stripHtmlHeader="yes" type="text/html">
      <OptionParm name="EmpFiles"
        prompt="Choose Employee File" value="John">
       <Option name="John" value="john.html"/>
       <Option name="Jane" value="jane.html"/>
      </OptionParm>
    </File>
  </Submodule>
</Generic>
```

Displaying Embedded HTML

Through SIP, you can embed a short HTML snippet in the portal view file. To display embedded HTML, you will define the `EmbeddedHtml` element in a Generic element. See the examples in Figure 12-7 on page 218 and the element definitions in Table 12-8 on page 249.

Rules for Embedded HTML

- Embedded HTML should be well formed.
- Embedded HTML should be brief; if the HTML content is lengthy, refer to "Displaying HTML from an External File" on page 216.
- The contents of embedded HTML should not include "header" tags such as `<HTML></HTML>`, `<HEAD></HEAD>`, or `<BODY></BODY>`. The portal will automatically provide these tags.
- The value you enter for embedded HTML may be any string of text not containing a less-than sign (<) or quotation marks ("), or similar characters. These characters may be inserted using the usual entity references (`&lt;`, and `&quot;`) or by their Unicode values using character references. All raw ampersands (&), those that do not begin a character or entity reference, must also be escaped as `&amp;`. Embedded HTML must not contain: "`]]>`"

Figure 12-7 Example of Displaying Embedded HTML from cannedDemo.xml

```
<Generic>
  <Submodule>
    <TitleBar title="Email Service"/>
    <EmbeddedHtml data="&lt;img height=79 width-239
     src=&quot;/OvSipDocs/C/demo/services/email-card.gif&quot;
     alt=&quot;/Email Service Card - Minor&quot;&gt;"/>
  </Submodule>
</Generic>
<Generic>
  <Submodule>
    <TitleBar title="Summary: HTTP-Web Pages Last 4 Hours"/>
    <File displayFileInfo="no" fileName=
   "$SIP_HOME_DIRhtdocs/C/demo/services/vpis_avail_gauge.html"
    stripHtmlHeader="yes" type="text/html"/>
  </Submodule>
</Generic>
```

Figure 12-8 Example of File and Embedded HTML

```
<Generic>
  <Submodule>
    <TitleBar title="Team Photo"/>
       <File displayFileInfo="no"
           fileName="$SIP_HOME_DIRhtdocs/C/demo/teamphoto.html"
           type="text/html"/>
      <EmbeddedHtml data="&lt;br&gt;&lt;i&gt;Standing
         (from Left): Jane, Joe, Janet, Jack. Sitting
         (from left): Bob, Bill, Betty, Beth;&lt;/i&gt;/>
  </Submodule>
</Generic>
```

Displaying the Contents of a URL

Through SIP, you can display the contents of a specified URL. For example, if you have an HTML file on a different server, or a CGI program, you can display the URL by referencing it in a portal view file.

You can display URLs in one of three ways:

- Inline frames (IFRAME). (Displays in scrollable window. The browser bears the responsibility for displaying the URL.)
- Anchor tags. (Displays a link.)
- Embedded content. (Displays in a non-scrollable window the content of the URL. SIP takes control of the content for display purposes.)

To display the contents of a URL, define the `Url` element in a `Generic` element. See the examples in Figure 12-9 on page 219 and the element definitions in Table 12-7 on page 247.

All three mechanisms can be used by the URL submodule to pass a parameter (user name, role properties, SIP home directory) to the application. You can also make it part of the path to the file or URL that you are accessing.

Figure 12-9 Example of Displaying Content of a URL

```
<Generic>
 <Submodule>
  <TitleBar title="HP E-Services IFRAME on IE,
     Link on Netscape"/>
  <Url anchorText="HP E-Services" displayMethod="inline"
```

```
     href="http://www.hp.com/e-services" inlineHeight="300"
     proxy="no"/>
 </Submodule>
</Generic>
```

Figure 12-10 NNM Commands Integration

```
<Generic>
  <Submodule>
    <TitleBar title="SNMP Query"/>
    <Url href="http://$NNM/OvCgi/webappmon.exe?$report&amp;sel=$target"
         displayMethod="inline" inlineHeight="300">
       <TextParm name="NNM" prompt="NNM Management Server:" value="islandia"/>
       <TextParm name="target" prompt="Target:" value="islandia"/>
       <OptionParm name="report" prompt="Report:"
                   value="Show Capabilities">
           <Option name="Demand Status Poll"
                   value="app=IP+Demand+Poll&amp;act=demandStatusPoll"/>
           <Option name="Demand Poll"
                   value="app=IP+Demand+Poll&amp;act=demandPoll"/>
           <Option name="Show Capabilities"
                   value="app=IP+Demand+Poll&amp;act=capsPoll"/>
           <Option name="Ping" value="app=IP+Tables&amp;act=ping"/>
           <Option name="Remote Ping"
                   value="app=IP+Tables&amp;act=rping"/>
       </OptionParm>
    </Url>
  </Submodule>
</Generic>
```

If you want to activate the use of the IFRAME HTML tag, set the `displayMethod` attribute to `inline` and use the `inlineHeight` attribute to specify the height of the IFRAME. The default is 100 pixels.

NOTE Netscape prior to 6.0 do not support IFRAME; the URL will be displayed as an anchor tag using `anchorText` as the text for the anchor. If you set `displayMethod` to `inline`, an anchor tag will always be present on both Internet Explorer and Netscape.

Proxying Capabilities

The generic module provides limited proxying capabilities when displaying the content of a URL. When the `proxy` attribute is set to `yes` then the SIP server will access the URL directly (over the secure network, for example). This way you can display portal data that otherwise would not be accessible to the end user. In addition, you can also provide the name and password that will be used when accessing the protected data (see `auth` attribute in Table 12-7 on page 247).

Not all URLs will proxy well through the Generic module. If the URL uses JavaScript extensively or if it contains relative links (other than in images, areas, hrefs, frames or target), the page will not display fully correctly or some of its functionality will not be available in the module. In this case, you may need to set the proxy attribute to `no` and make the data system accessible to end users.

Giving Your Module Access to SIP Data Through Variable Substitution

Your Generic-based module can be extended to allow the retrieving of user-specific information from SIP using predefined variables. If placed in the `commandLine`, `fileName`, and `href` attributes, the variables will be replaced by the Generic-based module with current SIP data.

The three variables give your integrated module access to information about the SIP home directory, the user that is logged in, and the role of that user. Note that you can access this information, but you cannot change it.

Table 12-1 Variable Substitution

Variable	Description
`$SIP_HOME_DIR`	This variable will be replaced with the SIP installation directory, for example, `C:\Program Files\HP OpenView\SIP`. This variable can be placed in a `commandLine`, `fileName` or `href` attribute in Command, File, or URL elements respectively. For an example of using `$SIP_HOME_DIR`, see Figure 12-14 on page 224.

Table 12-1 Variable Substitution

Variable	Description
`$OVLOGIN`	This variable will be replaced with the user's login name. You can substitute in the login name as part of a command, filename, or URL. If the command, file, or URL requires a login name, you can use `$OVLOGIN` to substitute in a user login name. For examples of using `$OVLOGIN`, see Figure 12-11, Figure 12-12, and Figure 12-13 on page 223.

`$OVROLE`	Allows applications to get data about user's current role, edit permissions, organizations defined by the MgmtData filter, and the property name. An application can use this information to determine what to display to the user.
	
	This is done by first defining `Role` in the User-Role package file. Specific keywords allow you to access four aspects of the role definition: current role, edit permissions, `MgmtData` filter, and properties:
	
	`$OVROLE[OVName]`
	`$OVROLE[OVEditLevel]`
	(See Figure 12-15 and Figure 12-16 on page 224.)
	
	`$OVROLE[OVOrgs]`
	(See Figure 12-17 on page 224.)
	
	`$OVROLE[<property name>]`
	See Figure 12-18 and Figure 12-19 on page 226.

Login Name Substitution

You can substitute in the login name as part of the command, file name, or URL. If you have an accessible URL that is login-name based, you can substitute in $OVLOGIN as part of the URL.

If you have a CGI program that you want to refer to on another server and the program is keyed off of a user login name, you can substitute `$OVLOGIN` as part of the HTTP string. (See example in Figure 12-11.)

When the Generic-based module sees this href with `$OVLOGIN`, it substitutes in the user name as part of the href and then allows the browser to fetch the document.

Figure 12-11 Example of Login Name Substitution ($OVLOGIN in URL)

```
<Url href="http://webpage.mycompany.com/$OVLOGIN/employeephoto.gif/>
```

In the example in Figure 12-11, the Generic-based module would display an image from the directory named for the SIP user. For example, if the SIP user name is "jan" then the accessed URL would be "`http://webpage.mycompany.com/jan/employeephoto.gif`"

Figure 12-12 Example of Login Name Substitution ($OVLOGIN in Command)

```
<Command commandline="ping $OVLOGIN"/>
```

In the example in Figure 12-12, the Generic-based module would attempt to ping a host with the same name as the logged-in user. For example, if the SIP user name is "john" then the command executed would be "`ping john`".

Figure 12-13 Example of Login Name Substitution ($OVLOGIN in File Name)

```
<File fileName="C:/EmployeeFiles/$OVLOGIN.html"/>
```

In the example inFigure 12-13, the Generic-based module would display a file named for the SIP user. For example, if the SIP user name is "Jan" then the accessed file would be "`C:/EmployeeFiles/Jan.html`"

SIP Installation Directory Substitution

In the example in Figure 12-14 on page 224, the portal will display a file called services.html that is located in the SIP home directory hierarchy.

For example, if SIP is installed in:
`C:\Program Files\HP OpenView\SIP`

Then the displayed file is:
`C:\Program Files\HP OpenView\SIP\htdocs\C\demo\services.html`

Figure 12-14 Example of $SIP_HOME_DIR Used in File Name

```
<Submodule>
    <TitleBar title="Services Purchased"/>
    <File fileName="$SIP_HOME_DIRhtdocs/C/demo/services.html"
         type="text/html" stripHtmlHeader="yes"/>
```

User-Role Substitution

In the example in Figure 12-15, the portal will execute `myapplication` and pass the user's current role and edit permissions as two parameters to `myapplication`. The application can then use this information to determine what to display to this user.

Figure 12-15 Example of $OVROLE[OVNAME] Used in a Command

```
<Command commandline="myapplication $OVROLE[OVName] $OVROLE[OVEditLevel]"/>
```

If the role is `AcmeTechnical` and the edit permissions level is `ViewAdmin`, then the command executed would be:

`myapplication AcmeTechnical ViewAdmin`.

Figure 12-16 Example of $OVROLE[OVNAME] Used in a URL

```
<Url
href="http://webpage.mycompany.com/foo.exe?role=$OVROLE[OVNAME]&amp;permission=$
OVROLE[OVEditLevel]"/>
```

In the example in Figure 12-16, the portal will execute the remote application and pass the user's current role and edit permissions as two parameters to `foo.exe`:

If the role is `AcmeTechnical` and the edit permissions level is `ViewAdmin`, then the URL accessed would be:

```
"http://webpage.mycompany.com/foo.exe?role=AcmeTechnical&amp;permission=ViewAdmin"
```

Figure 12-17 Example of $OVROLE[OVOrgs] Used in a Command

```
<Command commandline="myapplication '$OVROLE[OVOrgs]'"/>
```

In the example in Figure 12-17, the portal will execute `myapplication` and pass the management data associated with the user's current role as a parameter to `myapplication`.

If the management data filter defines the three organizations `Ariston`, `Kalloi`, and `Another Group`, then the command executed would be:

```
myapplication 'Ariston|Kalloi|Another Group'
```

Rules for Substitution of Management Data

- If a role is allowed to see all management data, the value of `$OVROLE[OVOrgs]` will be replaced with "`<ALL>`"
- If a role is not allowed to see any management data, the value of `$OVROLE[OVOrgs]` will be replaced with "`<NONE>`"
- If a role is allowed to see one organization's data, the value of `$OVROLE[OVOrgs]` will be replaced with the organization name.
- If a role is allowed to see the data of multiple organizations, then the organization names are passed as a string separated by the pipe character. For example, "`Ariston|Kalloi|Another Group`".

Figure 12-18 Example of Role Properties Defined in User-Role Package File

```
<Properties>
  <Property name="MgmtApp.login" value="acme/>
  <Property name="MgmtApp.password" value="acmepasswd"/>
</Properties>
```

In the User-Role package file, you would define the role properties as in Figure 12-18. Then, in the Generic-based module code, you would define the URL using `$OVROLE[<property name>]`. In the example in Figure 12-19, the portal will execute the remote application and pass the user's `MgmtApp login` and `password` as two parameters to `MgmtApp.exe`.

NOTE A module, whether supplied or Generic-based, may have need for additional role attributes. The role properties element can contain any number of property elements, and each property element has two attributes: name and value. The value of the name attribute is property name. For value of the value attribute is property value.

The supplied VP-IS module uses Role Properties to map a SIP role to a VP-IS server and customer. To do this, you must define two properties named: `VPIS.server` and `VPIS.customer`. The value of the

`VPIS.server` should be set to the fully qualified name of the VPIS system. The value of `VPIS.customer` should be set to the VP-IS customer name.

Generic-based modules can also access the additional role properties. If your module needs to access role properties information, define the role properties in the role file and use variable substitution to pass the information to your application.

Figure 12-19 Example of $OVROLE[<property name>] Used in a URL

```
<Url
href="http://webpage.mycompany.com/MgmtApp.exe?login=$OVROLE[MgmtApp.login]&amp;
password=$OVROLE[MgmtApp.password]"/>
```

Referring to the `Properties` definition in the User-Role package file in Figure 12-18, the accessed URL would be:

```
http://webpage.mycompany.com/MgmtApp.exe?login=acme&amp;password=acmepasswd
```

Adding an Edit GUI to Your Module

An edit GUI can be valuable for `Command`, `File`, and `Url` submodules. With this feature, you can define parameters and then reference them in the `commandLine`, `fileName`, or `href`. In the reference to the parameter, the tag must be `$<`*`ParameterName`*`>` where `<`*`ParameterName`*`>` is the name of the parameter defined in the element.

For examples, see Figure 12-3 on page 215, Figure 12-6 on page 217, and Figure 12-4 on page 216. Also, see examples in the following directory:

Windows NT/2000: `<`*`install_dir`*`>\integrations\`
UNIX: `/opt/OV/SIP/integrations/`

Refer, also, to the Table 12-9 on page 249 (TextParm Attributes) and Table 12-10 on page 250 (OptionParm Attributes).

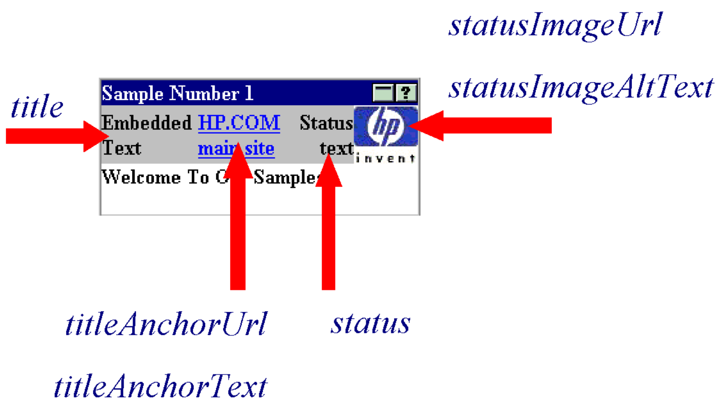
Adding Status Text, Graphics, and Links to a Submodule Title

There are four types of information that can be added to an submodule title bar area. At a minimum, you typically have a title using the `title` attribute. In addition, you can have a link to detailed data, an area for status text, and an area for a status image.

The submodule title is filled in from left to right in the following order:

- `title`
- `titleAnchorUrl|titleAnchorText`
- `status`
- `statusImageUrl/statusImageAltText`

Figure 12-20 Titlebar Attributes of a Submodule



NOTE For detailed information on elements and attributes of the Generic module, see “OVGeneric.dtd” on page 237, Table 12-2 on page 239 and Table 12-4 on page 240.

Adding a Link in the Submodule Titlebar

There are two attributes of the `TitleBar` element that control placing a link to detailed data in the titlebar. These are `titleAnchorUrl` and titleAnchorText. Specify the link's URL in titleAnchorUrl and specify the link's text in titleAnchorText.

For example:

```
<TitleBar titleAnchorText="Server Health"
   titleAnchorUrl="/OvSipDocs/C/demo/server_details.html"/>
```

Adding Status Text and Status Graphics in the Submodule Titlebar

Plain status text can be presented in the submodule titlebar using the status attribute. In addition, an image can be placed in the submodule title using the attributes: statusImageUrl and statusImageAltText. The following example demonstrates both:

```
<TitleBar status="115 alarms"
          statusImageAltText="80% major, 20% Normal"
          statusImage="/OvSipDocs/C/demo/alarm1.gif"
         title="Threshold Alarms"/>
```

Adding Online Help

HP OpenView Service Portal provides the capability to add help for a Generic-based module. You can provide help only for the module as a whole, not for individual submodules.

When adding a Generic module to a portal view file, you can specify a URL to an HTML file that contains help for the module using the `help` attribute.

The help attribute specifies the URL to the help content for this module instance. This attribute allows you to override the default help URL defined in the module registration file. If you place your help topic somewhere under the `/OvSipDocs` directory, your topic will be displayed in the same decorated window in which SIP help topics are displayed. The recommended location is:
`help="/OvSipDocs/C/help/<mod_directory>/<topic.html>"`

For example:

```
<ModuleInstance id="myModule
        title="MyModule"
       help="/OvSipDocs/C/MyHelp.html">
```

Each Generic module will navigate the customer to the specified HTML help file whenever the module’s "?" button is activated.

The Generic module can have multiple `Submodule` elements, hence, all the `Submodule` child elements of a single `Generic` element will reference the same HTML help file.

The `help` attribute must be a URL and can point at documentation from the local system. You are free to present a help page using the most appropriate style for the module.

Registering the Module with SIP

After you have created your module code, you can register it with SIP. This is done through a module registration file. The registration file defines such information as a unique ID for the module, a reference to the servlet, the module's associated help file, and module's capabilities.

1. Create a module registration file by copying, renaming, and modifying one of the existing ones located in the following directory:

   (Windows) `\HP OpenView\SIP\registration\`
   (UNIX) `/etc/opt/OV/SIP/registration/`

2. Edit the registration file to define module-specific information:

   - Enter `title`. This is the title that is displayed in the module's title bar in the portal.
   - Enter `classid`. This required attribute uniquely identifies the module. It must be unique among all registered modules.
   - Enter `implementation`. This required attribute is a reference to the servlet that implements the module. This is the name of the Java class implementing the Generic servlet. For Generic modules, this value is always the same:

```
"/servlet/com.hp.ov.portal.modules.ovgeneric.GmGenericServlet"
```

     For an example, see the Sample Module Registration File, Figure 12-21 on page 231.

   - Enter `outputType`. This attribute is used to determine in what column type to place the module.
   - Set the `add` attribute to "yes" if you want to allow the adding of this module from the user interface. If "yes", SIP will check the user configuration file to determine whether to allow it.
   - Set the `edit` attribute to "yes" if you want to allow editing capabilities from the user interface. If "yes", SIP will check the user configuration file to determine whether to allow it.
   - Enter `help`, a pointer to the default help file for this module.
   - Enter `configDTD`, the name of the module's Document Type Definition (DTD). This is the `OVGeneric.dtd`.

- Enter a value for `defaultConfigXML` only if `add="true"`. This optional attribute provides the path to an XML file with an XML fragment for the module configuration. This XML fragment will be added to the portal view when a user with editing permissions set to `ViewAdmin` adds the module through the module selection Add button.

3. Save the registration file.

Figure 12-21 Sample Module Registration File from Integrations Directory

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE ModuleRegistration SYSTEM "OVModuleRegistration.dtd">

<ModuleRegistration
   vendorName="Hewlett-Packard Company"
   vendorURL="http://www.openview.hp.com"
   description="HP OpenView SIP Keynote Module"
   version="1.0"
   title="Keynote"
   classid="Keynote"
   implementation="/servlet/com.hp.ov.portal.modules.ovgeneric.GmGenericServlet"
   outputType="wide"
   add="yes"
   edit="no"
   defaultConfigXML="defaults/OVDefaultKeynote.xml"
   configDTD=""
/>
```

Figure 12-22 OVModuleRegistration.dtd

```
<!-- OVModuleRegistration.dtd -->
<!-- Copyright (c) 2000 Hewlett-Packard Company -->
<!-- $Revision: /main/BACCHUS/8 $ -->
<!-- $Date: 2001/03/15 00:13 UTC $ -->

<!ELEMENT ModuleRegistration EMPTY>

<!ATTLIST ModuleRegistration
   vendorName          CDATA #IMPLIED
   vendorURL           CDATA #IMPLIED
   description         CDATA #IMPLIED
   version             CDATA #IMPLIED
   category            CDATA "General"
   title               CDATA #IMPLIED
   classid             ID    #REQUIRED
```

```
 implementation       CDATA #REQUIRED
outputType       (narrow|NARROW|wide|WIDE|any|ANY) "narrow"
 add                  (1|0|YES|yes|NO|no) "no"
 edit                 (1|0|YES|yes|NO|no) "no"
 help                CDATA #IMPLIED
 configDTD            CDATA #IMPLIED
 defaultConfigXML     CDATA #IMPLIED
 moduleTimeout      CDATA "90">
```

Restarting the Servlet Engine

After adding or changing a module registration file, you must stop and restart the servlet engine before changes take effect:

- *WindowsNT/2000*:

  From the `Control Panel`, select `Services`. Stop and then restart `Tomcat`. Alternatively, you can use the command line: **`net stop tomcat`** and **`net start tomcat`**.

- *UNIX:*

  As root, stop and restart the web server and servlet engine by running the following:

  Stop on HP-UX: **`/sbin/init.d/ovsip stop`**
  Start on HP-UX: **`/sbin/init.d/ovsip start`**

  Stop on Solaris: `/etc/init.d/ovsip stop`
  Start on Solaris: `/etc/init.d/ovsip start`

Testing the New Module

It is important to test your newly created Generic-based module to determine if you have used the correct syntax and have placed executables and files in locations that are accessible to the Portal. You also want to verify that the functionality is working.

1. Log in as someone with *ViewAdmin* edit permissions. Add the new module from the selection list at the bottom of the tab (if you registered your module with `add="yes"`).
2. Verify that the results in the browser are correct.
3. Modify the XML file, as needed.

Adding the Module to a Portal View

After you create and register your Generic-based module, you can add it to a portal view so that it appears in a portal. Generic-based modules are added in the same way other modules are added: (1) Through the interface (the preferred method), or (2) Through direct editing of the portal view XML file to insert the `Generic` and `Submodule` elements and attributes.

Through the SIP Interface (Preferred Method)

1. Log into SIP as a user with *ViewAdmin* permissions, and switch to the role that is associated with the portal view you want to modify.
2. Create or go to the tab to which you want to add the module.
3. Scroll to the bottom of the tab.
4. Select a module from the drop-down list and click `[Add]`.

Through Direct Editing of the XML

Generic-based modules can be added and configured by directly editing a portal view file, if you do not expect to add this Generic-based module again. Be careful, because you can introduce a lot of XML syntax problems by adding the module this way.

Adding the module to the portal view file manually involves creating a new module instance with an id that is unique among all module instances in the portal view file; then defining the `Generic` and `Submodule` elements and attributes. For information on the `ModuleInstance` element, see Table 9-4 on page 152.

Make a backup of configuration files before you customize them. If you edit the XML file and get incorrect XML syntax, you may want the ability to revert to the previous version of the file.

NOTE For detailed information on elements and attributes of the Generic module, see “OVGeneric.dtd” on page 237, Table 12-2 on page 239 and Table 12-4 on page 240.

1. In an ASCII editor, open the file that has your Generic-based module code, and copy the entire `Generic` element. Default modules are stored in the following location:

   Windows NT/2000: `<`*`install_dir`*`>\registration\defaults\`
   UNIX: `/etc/opt/OV/SIP/registration/defaults/`

   For the location of the Generic-based modules installed with SIP, see "Sample Generic-based modules of Third-Party Integrations" and "Sample Portal View files" on page 211.

2. Open the portal view file to which you want to add the Generic-based module. Portal view files are located in following directory:

   Windows NT/2000: `<`*`install_dir`*`>\conf\share\views\`
   UNIX: `/etc/opt/OV/share/SIP/conf/share/views/`

3. Find the `Sheet` (tab) and `Column` that you want the Generic module to appear on, and create a new `ModuleInstance`. For information on the `ModuleInstance` element, see Table 9-4 on page 152.

4. Paste the `Generic` element into the `ModuleInstance` element.

5. After you make modifications to the XML files, validate your XML syntax. For information on validation tools, see "Validating XML Files" on page 55.

6. Log into the portal to verify that your Generic-based module displays as intended.

OVGeneric.dtd

To create integrated modules based on the Generic module, you need to understand the document type definition (DTD) and specifications behind the module. Each instance of the Generic module is configured based on the DTD. The `OVGeneric.dtd` is in the following directory:

Windows NT/2000: `<install_dir>\conf\share\views\`
UNIX: `/etc/opt/OV/share/SIP/conf/share/views/`

A `Generic` element defines an instance of the Generic module and contains the following:

```
<Generic> consists of one or more:
  <Submodule> consists of:
    <TitleBar>
    <Command> or <File> or <URL>
    <EmbeddedHtml>
  </Submodule>
</Generic>
```

Figure 12-23 **OVGeneric.dtd**

```
<!-- OVGeneric.dtd -->
<!-- Copyright (c) 2000 Hewlett-Packard Company -->
<!-- $Revision: /main/BACCHUS/6 $ -->
<!-- $Date: 2001/02/27 20:45 UTC $ -->
<!ELEMENT Generic (Submodule+)>
<!ELEMENT Submodule (TitleBar?,(Command? | File? | Url?),
    EmbeddedHtml?)>
<!ELEMENT TitleBar EMPTY>
<!ATTLIST TitleBar
    title CDATA #IMPLIED
    titleAnchorUrl CDATA #IMPLIED
    titleAnchorText CDATA #IMPLIED
    status CDATA #IMPLIED
statusImageUrl CDATA #IMPLIED
statusImageAltText CDATA #IMPLIED
>
<!-- EmbeddedHtml - this is HTML that will be embedded
as it is at the end of the submodule, i.e., below the
command output or file. -->

<!-- Notice also that the submodule may display only
```

```
the Embedded HTML if a submodule does not contain
a Command, File or Url -->

<!ELEMENT EmbeddedHtml EMPTY>
<!ATTLIST EmbeddedHtml
    data CDATA #IMPLIED
>
<!ELEMENT Command ((TextParm*, OptionParm*)*)>
<!ATTLIST Command
    commandLine CDATA #REQUIRED
    type CDATA "text/plain"
    expires CDATA "0"
    stripHtmlHeader (yes | YES | no | NO | 0 | 1) "no"
>
<!ELEMENT File ((TextParm*, OptionParm*)*)>
<!ATTLIST File
    fileName CDATA #REQUIRED
    type CDATA "text/plain"
    displayFileInfo (yes | YES | no | NO | 0 | 1) "no"
    stripHtmlHeader (yes | YES | no | NO | 0 | 1) "no"
>
<!-- If displayMethod is "inline" or "anchor," the proxy
attribute is used by the portal. -->

<!-- If displayMethod is "embedded," then proxing is always
done regardless of whether the value is "yes" or "no." -->

<!-- "auth" is used only if "proxy" is set to "yes",
otherwise it is ignored -->

<!ELEMENT Url ((TextParm*, OptionParm*)*)>
<!ATTLIST Url
    href CDATA #REQUIRED
    displayMethod (inline | anchor | embedded) #REQUIRED
    inlineHeight CDATA "100"
    anchorText CDATA "click here"
    proxy (yes | YES | no | NO | 0 | 1) "no"
    auth CDATA #IMPLIED
>
<!ELEMENT TextParm EMPTY>
<!ATTLIST TextParm
    prompt CDATA #REQUIRED
    name CDATA #REQUIRED
    value CDATA #IMPLIED
>
<!ELEMENT OptionParm (Option+)>
```

```
<!ATTLIST OptionParm
    prompt CDATA #REQUIRED
    name CDATA #REQUIRED
    value CDATA #IMPLIED
>
<!ELEMENT Option EMPTY>
<!ATTLIST Option
    name CDATA #REQUIRED
    value CDATA #REQUIRED
>
```

Table 12-2 through Table 12-11 describe the DTD elements and attributes.

Table 12-2 Generic Element

Element	Description
`Submodule`	A child element of the `Generic` element, `Submodule` contains zero or 1 `TitleBar` elements, zero or 1 of the following elements: `Command`, `File`, or `Url`; and zero or 1 `EmbeddedHtml` element.

Table 12-3 Submodule Elements

Element	Description
`TitleBar`	An optional child element of the `Submodule` element, `TitleBar` displays a gray titlebar at the beginning of a Submodule. A `Submodule` can have one `TitleBar`.
`File`	An optional child element of the `Submodule` element, `File` imports a file on your local machine or network. It does not access files that require HTTP access. File can be HTML (`text/html`) or plain text (`text/plain`). For more information see “Displaying HTML from an External File” on page 216.
`Command`	An optional child element of the `Submodule` element, `Command` displays the output of any executable command that generates stdout. For more information, see “Displaying the Output from a Command” on page 214. Note that parameter substitution can be used with the `Command` element, as described in “Giving Your Module Access to SIP Data Through Variable Substitution” on page 221.

Table 12-3 Submodule Elements

Element	Description
`Url`	An optional child element of the `Submodule` element, this element displays the content of a URL. Using `Url`, you can display the contents of a remote file that requires HTTP access, or execute a CGI program. For more information, see “Displaying the Contents of a URL” on page 219.
`EmbeddedHtml`	An optional element that embeds raw HTML. Can be used by itself or in conjunction with command, file, or URL. If used with one of those, the value of EmbeddedHtml is output after the processing of the command, file, or URL and must be defined in the XML file after the command, file, or URL. For more information, see “Displaying Embedded HTML” on page 218.

Table 12-4 TitleBar Attributes

Attributes	Description
`title`	The name displayed in the submodule title bar (e.g., "Billing" or "Trouble Tickets"). The value can be either simple text or HTML. If HTML is used, characters such as "<" and ">" need to be appropriately escaped according the rules for valid HTML.
`titleAnchorUrl`	Defines a URL ("http:...") to be accessed via the submodule titlebar. The attribute `titleAnchorText` must also be set in order for this attribute to have an effect.
`titleAnchorText`	Defines the text to be placed within the anchor tag and displayed in the submodule title bar. The attribute `titleAnchorUrl` must also be set for this to have an effect.
`status`	Status text to be displayed on the right-hand side in the submodule title. The value can be either simple text or HTML.
`statusImageUrl`	Defines a URL ("http:...") of an image to display right justified in the submodule title bar.
`statusImageAltText`	Defines alternate text to display for the image.

For more information on `TitleBar`, see “Adding Status Text, Graphics, and Links to a Submodule Title” on page 227.

Table 12-5 Command Attributes and Elements

Attributes and Elements	Description

`commandLine`	A required attribute, `commandLine` defines the command to use with a submodule.
	
	Example:
	`"commandLine="/usr/sbin/ping hostname 10"`
	
	Optional arguments to a command can be placed as part of the `commandLine` attribute. If the value of `commandLine` contains one of the following, the command referenced by the attribute has the option of using this argument to determine a user context:
	
	`$SIP_HOME_DIR`
	`$OVLOGIN`
	`$OVROLE`
	
	For more information on substitution, see “Giving Your Module Access to SIP Data Through Variable Substitution” on page 221.
	
	Be careful when presenting the results of a command. A command is executed as:
	
	• *Windows NT/2000:* the local "System" account
	• *HP-UX*: `www/www`
	• Solaris: `nobody/nogroup`
	
	Make sure that only knowledgeable personnel implement this feature.

Table 12-5 Command Attributes and Elements

Attributes and Elements	Description
`type`	An attribute that indicates what type of output a command will generate based on standard MIME types. There are two values for this enumerated type: `text/plain`: The output from the command is pure ASCII text. In this mode, the Generic module will prefix the command's output with the HTML tag `<pre>` and will append the HTML tag `</pre>` after the command completes. This wrapping is necessary to support non-HTML output. `text/html`: The output from the command contains HTML text. No wrapping of the output is performed.
`expires`	An attribute that indicates the number of minutes the results of command execution are valid. The results of running a command are cached if `expires` is greater than 0 minutes. The cached results will be used the next time the same command is processed until the number of minutes specified in `expires` has elapsed. When the number of minutes has elapsed, the Generic module will reinvoke the command instead of using previous cache. The default is 0 minutes, meaning that no caching is performed and the command is always executed.

Table 12-5 Command Attributes and Elements

Attributes and Elements	Description
`stripHtmlHeader`	An attribute that is used when the value of `type` equals "`text/html`". This attribute can be set to `yes` in order to display just the contents of the HTML `<body>` tag. All HTML header information from the command is removed from the output stream. This attribute may need to be set if the referenced command will generate an entire valid HTML document including HTML header information. It may not be valid to return more than one set of HTML header tags to the browser. It is important to remove the additional tags from the referenced command. The default value of this attribute is `no`. Possible values: `yes|YES|no|NO|0|1`. Output from the portal generates a complete HTML document including `<Html>`, `<Head>`, and `<Body>` tags. If the output of your command generates its own `<Html>`, `<Head>`, and `<Body>` tags, you will have more than one each of the `<Html>`>, `<Head>`, and `<Body>` tags. If this is the case, set `stripHtmlHeader` to "yes." The default is set to "no" to allow the functioning of any potential scripts that you have placed in the `<Head>` tag. Be aware that a document containing multiple `<Html>`, `<Head>`, and `<Body>` tags may cause unpredictable behavior in the browser. If a problem arises that you suspect is related to header data, change `StripHtmlHeader` to "yes."
`TextParm`	The optional child element of `File`, `Command` or `Url` elements. `TextParm` element defines a parameter that can be used in `fileName`, `commandLine` or `href` attribute. The Portal will provide the UI for an end user to edit this parameter (module must be editable).

Table 12-5 Command Attributes and Elements

Attributes and Elements	Description
`OptionParm`	The optional child element of `File`, `Command`, or `Url` elements. `OptionParm` element defines an option parameter that can be used in `fileName`, `commandLine` or `href` attribute. The option parameter is a parameter with predefined values (options) that a user can choose from. The Portal will provide the UI for a user to choose the value of this parameter from the drop-down list (module must be editable). `OptionParm` element must contain at least one `Option` element.

Table 12-6 File Attributes

Attributes	Description

`fileName`	A required attribute, `fileName` defines the file that can be accessed and displayed. The file needs to reside on the same system or be accessible via the network. The file path can be absolute (starting from the root directory) or you can use `$SIP_HOME_DIR` to reference files relative to the SIP installation directory.
	
	Example:
	`fileName="/tmp/file.html"`
	
	You can take advantage of a portal user context if the value of fileName contains one of the following:
	
	`$SIP_HOME_DIR`
	`$OVLOGIN`
	`$OVROLE`
	
	For more information on substitution, see “Giving Your Module Access to SIP Data Through Variable Substitution” on page 221.

Table 12-6 File Attributes

Attributes	Description
`type`	Indicates the type of file. There are two values for this attribute: `text/plain`: The file content is pure ASCII text. In this mode, the Generic module will prefix the file content with the HTML tag <pre> and will append the HTML tag </pre> after the file content. This wrapping is necessary to support non-HTML files. `text/html`: The file contains HTML text. No wrapping of the content is performed.
`displayFileInfo`	This attribute can be set to "yes" in order to display the full path to the file and the time when the file was last modified. The default value for this attribute is "no." The default is not to display the full pathname or modification time.
`stripHtmlHeader`	When the value of `type` equals "`text/html`", this attribute can be set to "`yes`" to display just the contents of the HTML `<body>` tag. All HTML header information from the file will be removed from the output stream. This attribute may need to be set if the referenced file contains an entire valid HTML document including HTML header information. It may not be valid to return more than one set of HTML header tags to the browser. It is important to remove the additional tags from the referenced file. The default value of this attribute is "no." Possible values: `yes|YES|no|NO|0|1`. Output from the portal generates a complete HTML document including `<Html>`, `<Head>`, and `<Body>` tags. If the output of your command generates its own `<Html>`, `<Head>`, and `<Body>` tags, you will have more than one each of the `<Html>`>, `<Head>`, and `<Body>` tags. If this is the case, set `stripHtmlHeader` to “yes.” The default is set to “no” to allow the functioning of any potential scripts that you have placed in the `<Head>` tag. Be aware that a document containing multiple `<Html>`, `<Head>`, and `<Body>` tags may cause unpredictable behavior in the browser. If a problem arises that you suspect is related to header data, change `StripHtmlHeader` to “yes.”

Table 12-6 File Attributes

Attributes	Description
`TextParm`	The optional child element of `File`, `Command` or `Url` elements. `TextParm` element defines a parameter that can be used in `fileName`, `commandLine` or `href` attribute. The Portal will provide the UI for an end user to edit this parameter (module must be editable).
`OptionParm`	The optional child element of File, Command or Url elements. `OptionParm` element defines an option parameter that can be used in `fileName`, `commandLine` or `href` attribute. The option parameter is a parameter with predefined values (options) that a user can choose from. The Portal will provide the UI for a user to choose the value of this parameter from the drop-down list (module must be editable). `OptionParm` element must contain at least one `Option` element.

Table 12-7 Url Attributes

Attributes	Description

`href`	A required attribute that defines the URL to use with this submodule.
	
	Examples:
	`href="http:/www.hp.com"`
	
	Optional arguments to a URL can be placed as part of the `href` attribute. If the value of `href` contains one of the following, the URL referenced by the attribute has the option of using this argument to determine a user context for the URL:
	
	`$SIP_HOME_DIR`
	`$OVLOGIN`
	`$OVROLE`
	
	For more information on substitution, see “Giving Your Module Access to SIP Data Through Variable Substitution” on page 221.

`displayMethod`	A required attribute that can be set to either `inline`, `anchor` or `embedded`. The `inline` value indicates whether <`IFRAME`> tags should be used. A standard <`A`> tag will also be displayed because browsers other than IE, including all Netscape browsers prior to version 6.0, do not support the <`IFRAME`> tag. The `anchor` value tells the portal to display only a link to the specified URL. The `embedded` value tells the portal to access the specified URL and embed (display) its content into the submodule.
`inlineHeight`	This attribute is only applicable when `displayMethod` is `inline`. Specifies in pixels the height of the IFRAME. Represents the height in pixels that will be provided for an IFRAME. If not supplied, the default is 100 pixels.

Table 12-7 Url Attributes

Attributes	Description
`anchorText`	This attribute indicates the text to display on browsers that do not support the <`IFRAME`> tag, such as versions of Netscape that are earlier than 6.0, or when `displayMethod` is `anchor`. Represents the text to display for a resulting HTML <A> anchor tag. The default value is "click here".
`proxy`	An attribute that indicates whether SIP should proxy the data from the specified URL. If proxy="no", then the end user's browser will fetch data from the specified location. The data should be accessible to the end user. For example, set proxy to "no" when displaying user myYahoo account. If "proxy" equals "yes" then the SIP server will fetch data from the specified location using http request. This allows the SIP server to directly communicate with another server (data source) over a secured network. Use this method to access data that are not visible to the end user, for example Concord or NNM reports. If `displayMethod="embed"`, then proxy is assumed to be "yes." The default value of "proxy" attribute is "no" Possible values: yes, no, YES, NO, 0, 1
`auth`	This attribute is used if `proxy` equals "yes" and the accessed location is protected with web server authentication. "`auth`" specifies the name and password that will be used by the SIP server when accessing the protected data. Name and password should be separated by colon: For example, `auth="name:password"`
`TextParm`	The optional child element of `File`, `Command` or `Url` elements. `TextParm` element defines a parameter that can be used in `fileName`, `commandLine` or `href` attribute. The Portal will provide the UI for an end user to edit this parameter (module must be editable).

Table 12-7 Url Attributes

Attributes	Description
`OptionParm`	The optional child element of File, Command or Url elements. `OptionParm` element defines an option parameter that can be used in `fileName`, `commandLine` or `href` attribute. The option parameter is a parameter with predefined values (options) that a user can choose from. The Portal will provide the UI for a user to choose the value of this parameter from the drop-down list (module must be editable). `OptionParm` element must contain at least one `Option` element.

Table 12-8 EmbeddedHtml Attributes

Attributes	Description
`data`	The value of this attribute contains HTML for output constrained by the conventions outlined in "Rules for Embedded HTML" on page 218.

Table 12-9 TextParm Attributes

Attributes	Description

`prompt`	A required attribute. Specifies the prompt for the parameter entry. Prompt will be displayed during parameter editing and it should clearly identify the parameter for a user.
	
	For example,
	`prompt="Host name:"`
	`prompt="Enter name of NNM station:"`
	
	For more information on parameter editing, see "Adding an Edit GUI to Your Module" on page 226.

`name`	The required attribute. The name of the parameter. This name preceded by the dollar sign $ can be used in the value of `fileName`, `href` or `commandLine` attribute. The SIP portal will replace this `$name` by the parameter value before accessing the file or the URL or executing the command.

Table 12-9 TextParm Attributes

Attributes	Description
value	The default value of the parameter. The parameter ($name) in fileName, href or commandLine will be replaced by this value (the value of the value attribute) before accessing the file or the URL or executing the command.

Table 12-10 OptionParm Attributes

Attributes	Description

prompt	The required attribute. The prompt for the parameter entry. Prompt will be displayed during parameter editing and it should clearly identify the parameter for a user.
	
	For example,
	`prompt="Host name:"`
	`prompt="Choose name of NNM station:"`
	
	For more information on parameter editing, see “Adding an Edit GUI to Your Module” on page 226.

name	The required attribute. The name of the parameter. This name preceded by the dollar sign $ can be used in the value of fileName, href or commandLine attribute. The SIP portal will replace this $name by the parameter value before accessing the file or the URL or executing the command.
value	The name of a default option. The name of an option that define the default value of the parameter. The option with this name must be defined i.e. Option element with this name must be contained in this OptionParm element. The parameter ($name) in fileName, href or commandLine will be replaced by the value of this option before accessing the file or the URL or executing the command.
Option	A child element of OptionParm element. The Option element defines one of options for the parameter, that is, one of the values that the parameter can take. There must be at least one Option element defined (contained in) for an OptionParm element.

Table 12-11 Option Attributes

Attributes	**Description**
`name`	A required attribute of the `Option` element, specifies the name of the option. This name will be displayed in the drop-down selection list to identify the option.
`value`	A required attribute of the `Option` element, specifies the default value of the option. If the parameter has this option value (that is, if the option is selected), the parameter in `fileName`, `href`, or `commandLine` will be replaced by this value (the value of the `value` attribute) before accessing the file or URL or executing the command.

13 Developing a Custom Authentication Provider

Understanding What It Means to Integrate A Custom Authentication Provider

If none of the supplied authentication providers meets your needs, you can write and integrate your own custom authentication provider. SIP provides APIs and sample code to write your own Java class to integrate with SIP. Following is a general idea of what is involved in this type of integration.

SIP Authentication Model

The SIP 2.0 authentication model loads a configured Authentication Provider Java class. If one of the supplied JAVA classes is not sufficient, you can create a Java class that implements a custom authentication provider and integrate it with SIP.

In general terms, creating a custom authentication provider involves:

- Deciding the type of Authentication Provider to develop: Portal Authentication Provider or External Authentication Provider.
- Developing the code.
- Developing the web pages: login and logout pages.
- Making the authentication provider configurable, if relevant.
- Registering the authentication provider with SIP.

Deciding Which Type of Authentication Provider to Develop

First, you need to decide whether you are going to develop a Portal Authentication Provider or an External Authentication Provider. Use the following general principles and supplied samples to guide your decision.

General Principles

You should implement the `PortalAuthenticationProvider` interface if the user's session (the "state" of whether the user is logged in) is held locally, in SIP.

You should implement the `ExternalAuthenticationProvider` interface if the user's session is held externally to SIP.

`PortalAuthenticationProvider` implements the `Authenticate` method, which accepts username and password as input and makes the decision whether password actually authenticates username.

`ExternalAuthenticationProvider` implements the `getAuthenticatedUser` method, which determines whether there is an authenticated user or not (determined through some means known only to it), and if so, returns the username to SIP.

Supplied Authentication Providers As Examples

Use the supplied authentication providers in Table 13-1 on page 256 as examples when developing a custom authentication provider.

Table 13-1 Supplied Authentication Providers

Supplied Authentication Providers	Type and Description
NullAuthenticationProvider	An External Authentication Provider, because the user's state of being logged in is external to SIP (always true) and because the user never inputs a username or password, and the Authentication Provider returns a username (always "anyuser") to SIP.
NoPasswordAuthenticationProvider	A Portal Authentication Provider, because the user's state of being logged in is local to SIP, and because it accepts a username from the user.
FileAuthenticationProvider	A Portal Authentication Provider, because the user's state of being logged in is local to SIP, and because it authenticates based on a username and password input by the user.
LDAPAuthenticationProvider	A Portal Authentication Provider, because the user's state of being logged in is local to SIP, and because it authenticates based on a username and password input by the user. It does perform an authenticated bind with the remote LDAP server, but the LDAP server does not maintain this bind for the duration of the session; the LDAP Authentication Provider merely delegates the password validation task to the LDAP server.
NNMSSOAuthenticationProvider	An External Authentication Provider, because the user's state of being logged in is external to SIP (held by ovsessionmgr, part of NNM). SIP never sees the username and password input by the user, and it returns the username (which it gets from a cookie) to SIP.
WebServerAuthenticationProvider	An External Authentication Provider, because the user's state of being logged in is external to SIP (held by the web server). SIP never sees the username and password input by the user, and it returns the username (which it gets by calling `HttpServletRequest.getRemoteUser()`) to SIP.

Developing the Code

Following is general information that will help you develop your custom authentication provider code.

Development Resources

You have several resources for the development of the authentication provider.

Javadocs

Javadoc for the Authentication Provider interfaces can be found at:

- Windows NT/2000: `<`*`install_dir`*`>\htdocs\javadoc\index.html`
- UNIX: `/opt/OV/SIP/htdocs/javadoc/index.html`

Sample Authentication Providers

Start with one of the two sample Authentication Provider java files:

- `SampleExternalAuthenticationProvider.java`
- `SamplePortalAuthenticationProvider.java`

Located in:

- *Windows NT/2000:* `<`*`install_dir`*`>\samples\authentication`
- *UNIX:* `/opt/OV/SIP/samples/authentication`

Main Program

For testing purposes, it can be helpful to add a main method to your Authentication Provider, which outputs results to standard output.

For a PortalAuthenticationProvider, the main method would interpret its first two command line arguments as username and password, pass them to the Authenticate method, and output detailed tracing.

An ExternalAuthenticationProvider could be tested as a CGI program, so that it can access cookies.

ExternalAuthenticationProvider

An ExternalAuthenticationProvider must implement the `getAuthenticatedUserName` method. Refer to the javadoc for details.

Since an ExternalAuthenticationProvider maintains state with a remote authentication service, if it supports logout, it may need to clear this state when the user logs out. For example, it may need to remove the session cookie.

PortalAuthenticationProvider

A PortalAuthenticationProvider must implement the `authenticate` method. Refer to the javadoc for details.

The logout method of a PortalAuthenticationProvider normally does not need to do anything.

The Web Pages: Login Page and Logout Page

Following are explanations for developing a login page and logout page for your custom authentication provider.

Developing a Login Page

The login page you develop is different based on whether you are developing an External Authentication Provider or a Portal Authentication Provider.

ExternalAuthentication Provider

An ExternalAuthenticationProvider may or may not use a login page. Two models are possible.

If the user has already logged in when the portal is first accessed, the getAuthenticatedUserName method can detect this and return the correct user name. If not, it returns null and the Portal redirects to the login page, which either is an error page ("You have not already logged in") or is the login page for the external authentication server, which gathers whatever parameters it requires and authenticates however it chooses. From SIP's point of view, all that is required is that the next time the getAuthenticatedUserName method is called, it can return a user name. Normally this requires a session cookie to be set. See the Javadoc for the ExternalAuthenticationProvider interface for a detailed description of how to do this.

PortalAuthenticationProvider

The login page associated with a PortalAuthenticationProvider is required to forward to `/ovportal/` with the query parameters `J_USERNAME` and `J_PASSWORD` set. Normally this would be accomplished with a form with "`action=/ovportal/`". The Portal calls the `PortalAuthenticationProvider.authenticate` method with these parameters to authenticate the user.

When developing your login page, start with `/ovportal/jsp/security/login.jsp`.

Located in:

Windows NT/2000:

`<`*`install_dir`*`>\webapps\ovportal\jsp\security\login.jsp`

UNIX: `/opt/OV/SIP/webapps/ovportal/jsp/secuity/login.jsp`

This supplied login page also has the following useful features, which you can copy:

- If Javascript is enabled, it positions the cursor on the username field when the page is first displayed.
- If Javascript is enabled, typing the username and then hitting either tab or newline forwards the user to the password field.
- If Javascript is enabled, typing the password and then hitting newline submits the form.
- If Javascript is not enabled, the login page does work correctly, though it is not as user-friendly.

Developing a Logout Page

The portal redirects to the configured logout page when the user logs out. This page can issue a message, such as, "If you want to be sure none of your confidential data escapes, exit the browser now." Or it can permit the user to login again, with a button linking to `/ovportal/`.

Make the Authentication Provider Configurable (if Relevant)

Your authentication provider may have configuration parameters. See for example the sample configuration file for the LDAP Authentication Provider, located in:

Windows NT/2000:
`<install_dir>\conf\share\authentication\LDAP\LDAP.xml`

UNIX:
`/etc/opt/OV/SIP/conf/share/authentication/LDAP/LDAP.xml`

Register the Authentication Provider with SIP

You must register your authentication provider with SIP by configuring the Authentication element in the file.

Windows NT/2000:
`<`*`install_dir`*`>\conf\framework\OVPortalConfig.xml`

UNIX: `/etc/opt/OV/SIP/conf/framework/OVPortalConfig.xml`

For a detailed description of this registration, refer to Table 6-2 on page 71.

14 Deploying Customer Portals

Introduction

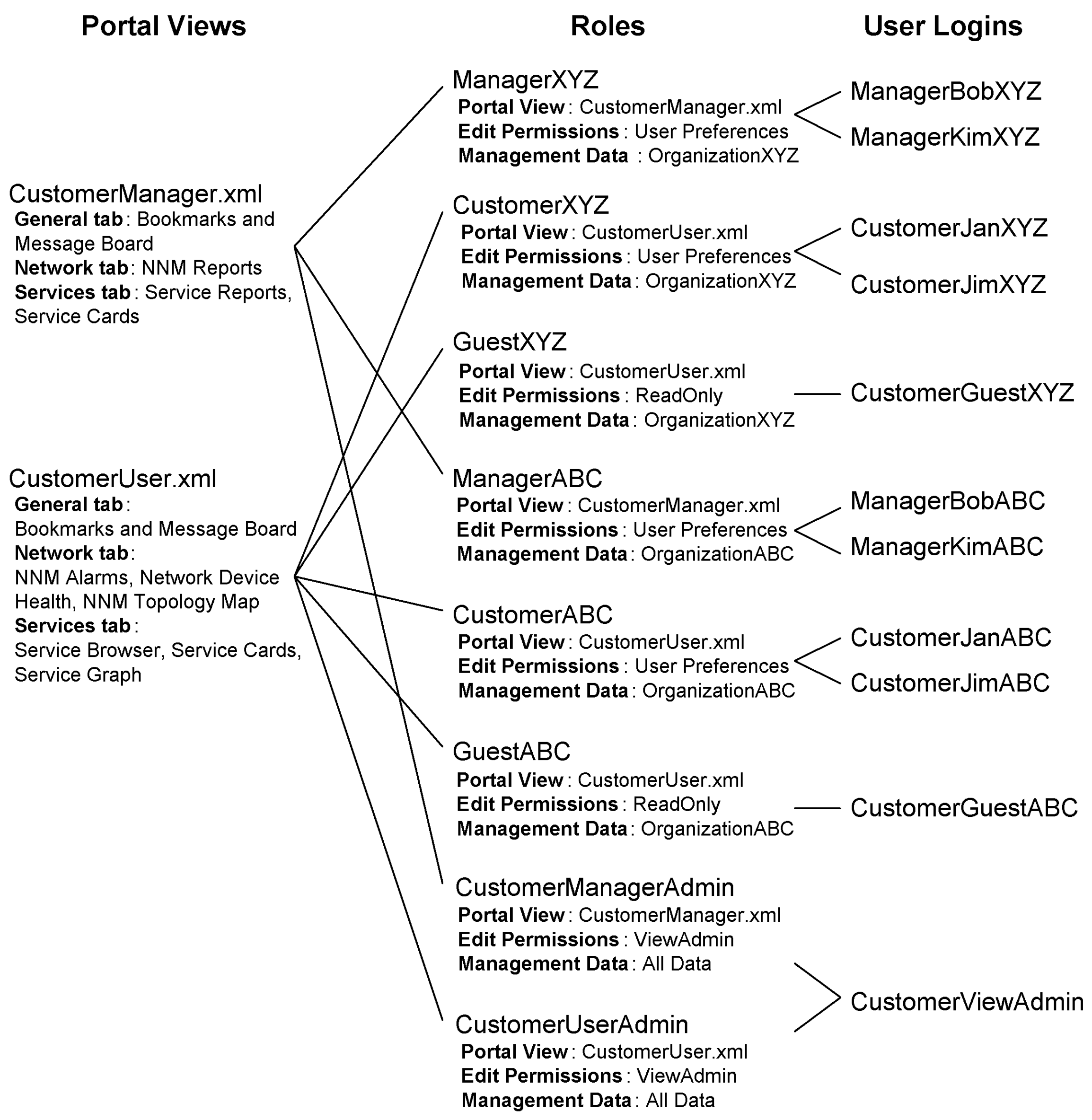
The first step in the deployment of customer portals is determining what the customer (or user) is authorized to see and do through the portal. You will need to answer questions like the following:

- What modules do I want this user see?
- How will the modules be arranged on the tabs?
- The data of which organization is relevant to this user?
- Can the user change the configured modules and tabs, change only their name and color scheme, or simply look at the data and change nothing?

The answers to these questions may be the same or similar for groups or categories of customers. SIP 2.0 gives you a flexible way to create and deploy customer portals—the User-Role Model—that helps reduce the number of configuration files that you will need to maintain for your customers. Whereas portal views determine what modules a user can see, roles determine what data a user can see, and what editing they can do.

For an example of how you might configure a set of roles that would meet the needs of several users, see Figure 14-1 on page 265.

Figure 14-1 Example of Role Configurations


Portal Views
Roles
User Logins
CustomerManager.xml
General tab: Bookmarks and Message Board
Network tab: NNM Reports
Services tab: Service Reports, Service Cards
CustomerUser.xml
General tab: Bookmarks and Message Board
Network tab: NNM Alarms, Network Device Health, NNM Topology Map
Services tab: Service Browser, Service Cards, Service Graph
ManagerXYZ
Portal View: CustomerManager.xml
Edit Permissions: User Preferences
Management Data : OrganizationXYZ
CustomerXYZ
Portal View: CustomerUser.xml
Edit Permissions: User Preferences
Management Data: OrganizationXYZ
GuestXYZ
Portal View: CustomerUser.xml
Edit Permissions: ReadOnly
Management Data: OrganizationXYZ
ManagerABC
Portal View: CustomerManager.xml
Edit Permissions: User Preferences
Management Data: OrganizationABC
CustomerABC
Portal View: CustomerUser.xml
Edit Permissions: User Preferences
Management Data: OrganizationABC
GuestABC
Portal View: CustomerUser.xml
Edit Permissions: ReadOnly
Management Data: OrganizationABC
CustomerManagerAdmin
Portal View: CustomerManager.xml
Edit Permissions: ViewAdmin
Management Data: All Data
CustomerUserAdmin
Portal View: CustomerUser.xml
Edit Permissions: ViewAdmin
Management Data: All Data
ManagerBobXYZ
ManagerKimXYZ
CustomerJanXYZ
CustomerJimXYZ
CustomerGuestXYZ
ManagerBobABC
ManagerKimABC
CustomerJanABC
CustomerJimABC
CustomerGuestABC
CustomerViewAdmin

Understanding the User-Role Model

After a user is authenticated and considered a valid SIP user, a second level of security ensures that the user sees only the data you want him to see, and changes only the things you want her to change.

SIP uses an authorization model called the User-Role Model. By associating users with roles and assigning to each role what you want the user to be able to see and do, you achieve portal security.

All valid portal users are associated with at least one role. A **role** defines what a user can see and do through the portal at a particular point in time. Users that are associated with multiple roles can easily switch between roles through a drop-down list in the interface.

A role consists of four parts:

- **A portal view**
  A portal view is what is displayed by the portal to a user. It is a configured set of modules and how they appear on tabs. It is also a configured set of portal view attributes, such as name in the welcome banner, portal color scheme, refresh rate, default tab, portal header, and portal footer, and others. (Two of these attributes—the name in the welcome banner and the color scheme—can be overridden by the user if they have at least "UserPreferences” editing permissions.)
- **An editing permissions level**
  Editing permission levels define the interactive editing operations that a user can perform through the portal interface. Each level of editing permissions includes all the operations defined by the previous level and adds some additional operations. The three levels are: ReadOnly, UserPreferences, and ViewAdmin. See “Choosing the Level of Edit Permissions for a Role” on page 284 for details.
- **A set of management data**
  Management data refers to the information about services, nodes, and interfaces that is displayed by management modules through the portal. In the User-Role Model, you specify the customers (organizations) whose data will be displayed to the users who are associated with a given role. Only the management data associated with the specified organizations is displayed. The resources associated with the organization are defined in the customer model.
- **An extensible list of role properties**
  Role properties can be used to, among other things, implement a single sign-on solution for logging in to back-end management

applications. It can be used by modules, including the generic module.

Understanding the Three Authorization Models

SIP supports three authorization models:

- Explicit user entry
- User view file (Default Role)
- Default user

Following is an explanation of each model and a decision tree diagram that describes what validity checks occur when a user logs into SIP. Note that the authorization is checked in the above order.

Explicit User Entry

In this case, the user login corresponds to an explicit user entry in the User-Role model. The user gets the initial role defined in the User entry. (You can check if there is an explicit entry for a user in the created roles database by running `test_role_db -u <user>`.)

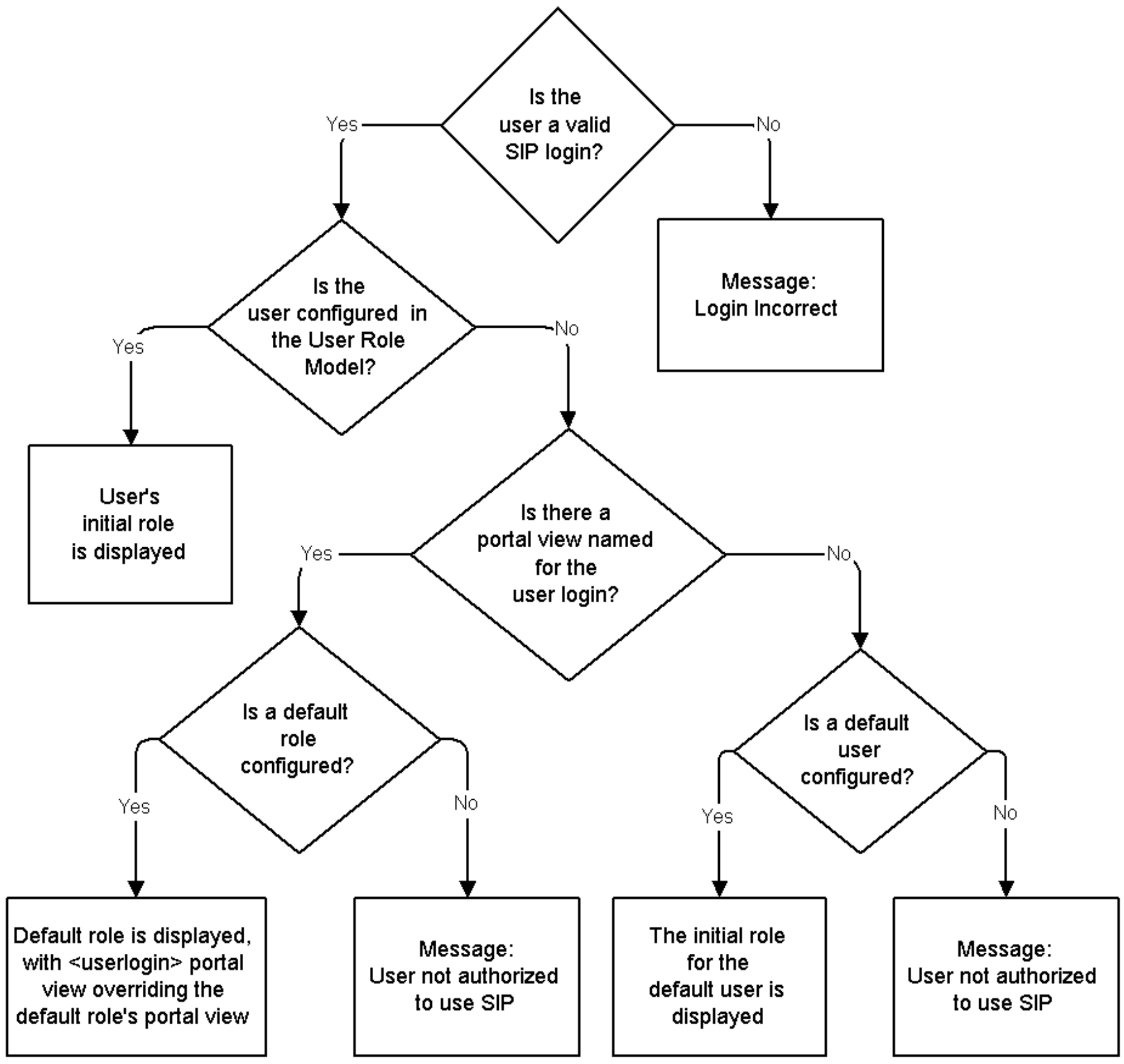
User View File (Default Role)

If there is no explicit user entry for the user login but there is a view file named <login>.xml and there is a default role (indicated by a value of "yes" for the `defaultRole` attribute of a configured role), the user gets the default role with the view file used as the view. This means of authorization is provided as a mechanism similar to SIP 1.0 which allows users to be configured by simply having a configuration file named for the user. This mechanism can be easily disabled by not specifying any default role. (You can see if any default role is configured by running `test_role_db` without any arguments.)

Default User

If neither of the two preceding mechanisms allow the user to be authorized, the user gets the rights of the default user (if there is one configured). (The default user is specified by a value of "yes" for the defaultUser attribute of a User entry.) The user is logged in with the initial role of the default user and can switch to any of the roles assigned to the default user. (You can see if the default user is configured by running `test_role_db` without any arguments.) You can disable this mechanism by not having a default user.

Figure 14-2 User Authorization


Is the user a valid SIP login?
Yes
No
Message: Login Incorrect
Is the user configured in the User Role Model?
Yes
No
User's initial role is displayed
Is there a portal view named for the user login?
Yes
No
Is a default role configured?
Yes
No
Default role is displayed, with <userlogin> portal view overriding the default role's portal view
Message: User not authorized to use SIP
Is a default user configured?
Yes
No
The initial role for the default user is displayed
Message: User not authorized to use SIP

Prerequisites to Deploying Customer Portals

Before you can begin creating roles and associating users with those roles, you need to do the following:

- Decide on a portal view for the given customer. This step assumes that you have already created portal views. If you have not created portal views, see “Introduction to Portal Views” on page 146.
- Make sure the customer and the associated resources are represented in the Customer Model. If the customer’s organization is not yet part of the customer model data sources, see Chapter 7, “Implementing a Customer Model for Mapping Customers to Resources,” on page 81.
- Create user logins for the customer. For more information, see “Configuring User Logins” on page 75.

Understanding User-Role Configuration Files

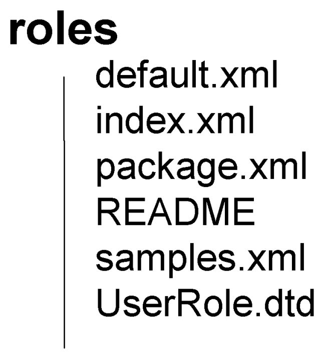
The User-Role Model files are located in the following directory:

Windows NT/2000: `<install_dir>\conf\share\roles\`
UNIX: `/etc/opt/OV/SIP/conf/share/roles/`

Figure 14-3 below depicts the directory structure when SIP is installed:

- `default.xml` - A package configuration file that defines a default user and role used when authorizing valid SIP users who are not explicitly defined in the User-Role Model.
- `index.xml` - Used as an index to all package files that contain user and role information and comprise the User-Role Model.
- `package.xml` - A package file that you can copy and rename as a starting point when creating your own package files.
- `README` - Important notes on updating the User-Role Model.
- `samples.xml` - A package configuration file that defines several sample users and roles for out-of-the-box use of SIP.
- `UserRole.dtd` - A file that defines the XML DTD for the User-Role package files.

Figure 14-3 **roles directory**



Rules for Updating the User-Role Model

- Two types of User-Role configuration files make up the User-Role Model: an index file and package files.
- You can store the User-Role package files in any directory structure under the `roles` directory. Do not, however, put any files in the generated `.db` database directory.
- Never edit any file in the `.db` directory or its subdirectories. These files are automatically generated by the `create_role_db` command and constitute the User-Role database.
- The index file defines all of the package files that make up the User-Role Model. The index file defines the `UserRoleModel` element, shown in Figure 14-4 and described in Table 14-1. You can have only one index file and it must be named `/conf/share/roles/index.xml`.
- The User-Role package files each contain some set of users and roles and together define all the users and roles that make up the User-Role Model. All users and roles can be defined in a single file, or they can be partitioned into multiple files (for example, one for each customer or organization). The package files define the `UserRolePackage` element shown in Figure 14-5 and described in Table 14-2. You can name the package files anything you want as long as you reference them in the index file.
- **After making changes to the User-Role configuration files, you must run the create_role_db command to update the database files**. For more information on the role database, see “Checking the Contents of the Role Database” on page 304.

Figure 14-4 Sample index.xml File

```
<UserRoleModel>
   <UserRolePackageRef href="default.xml"/>
   <UserRolePackageRef href="samples.xml"/>
</UserRoleModel>
```

Figure 14-5 Sample package.xml file

```
<UserRolePackage title="Sample Users and Roles">
```

```
  <!-- Users -->
  <User name="admin" displayName="Admin" initialRole="Welcome">
    <RoleRef href="Welcome"/>
    <RoleRef href="LiveDemo"/>
    <RoleRef href="AcmeTechnical" title="Acme Technical">
       <EditPermission level="ViewAdmin"/></RoleRef>
  </User>

  <!-- Roles -->
  <Role name="Welcome" title="Welcome">
    <PortalViewRef href="welcome.xml"/>
    <EditPermission level="UserPreferences"/>
    <MgmtDataRef href="NoData"/>
  </Role>
  <Role name="LiveDemo" title="Live Demo">
    <PortalViewRef href="samples/liveDemo.xml"/>
    <EditPermission level="ViewAdmin"/>
    <MgmtDataRef href="AllData"/>
  </Role>
  <Role name="AcmeTechnical" title="Technical">
    <PortalViewRef href="samples/technical.xml"
      copy="Acme/technical.xml"/>
    <EditPermission level="UserPreferences"/>
    <MgmtDataRef href="AcmeOrg"/>
  </Role>

<!-- Management Data -->
<!-- This will show data only for the "Acme" organization -->
  <MgmtData name="AcmeOrg">
    <OrganizationFilter>
```

```
      <OrganizationRef href="Acme"/>
    </OrganizationFilter>
  </MgmtData>
```

NOTE Sample User-Role package files can be found in the following two directories:

Windows NT/2000: `<install_dir>\conf\share\roles\`
UNIX: `/etc/opt/OV/SIP/conf/share/roles/`

Windows NT/2000: `<install_dir>\samples\authorization\roles\`
UNIX: `/opt/OV/SIP/samples/authorization/roles/`

Table 14-1 UserRoleModel Elements and Attributes

Elements and Attributes	**Description**
`UserRoleModel`	The root element of the `/conf/roles/index.xml` file. This file is an index to all packages that together make up the User-Role Model. Each package is stored in a separate XML file A User-Role Model consists of one or more packages.
`UserRolePackageRef`	The child element of the `UserRoleModel` element. Each `UserRolePackageRef` element refers to a `UserRolePackage`.
`href`	An attribute of the `UserRolePackageRef` element. Specifies the file containing the User-Role package information. The value of `href` is interpreted as a filename relative to the `conf/share/roles` directory. It can include subdirectories. For example, the `href` value `gold/Apex.xml` is interpreted as the file `conf/share/roles/gold/Apex.xml`. The file so specified must be an XML document with `UserRolePackage` as the root element.

Table 14-2 UserRolePackage Element

Elements and Attributes	Description
`UserRolePackage`	The root element of each User-Role package configuration file. Contains a set of user and role definitions. It can also contain management data definitions. A package can, for example, define all the users and roles for a particular customer or organization. Users in one package can reference `Role` and `MgmtData` definitions in another package.
`title`	An attribute of the `UserRolePackage` element. Optional descriptive string that describes the purpose of the package.
`defaultMgmtData`	An attribute of the `UserRolePackage` element. A reference to a `MgmtData` element. Specifies the management data to use for any role defined in the package that has a management data value of `DefaultMgmtData`. Makes it easy to specify the management data for all roles in the package in one place.

Table 14-3 User Element

Elements and Attributes	Description
`User`	A child element of `UserRolePackage` element. Defines a valid SIP user. `User` element has `RoleRef` child elements. This list specifies the set of roles for the user. The order in which the `RoleRef` elements appear for a user determine the order in which the roles appear in the role drop-down menu in the user interface.
`name`	An attribute of `User`, name is a unique identifier that corresponds to the login name by which the user was authenticated to SIP.
`displayName`	An attribute of `User`, `displayName` is the name of the user that will be displayed in the portal welcome banner in the user interface, unless it is overridden in the user preferences.

Table 14-3 User Element

Elements and Attributes	**Description**
`initialRole`	An attribute of `User`, initialRole is a mandatory attribute that specifies the role that the user will be in when he or she first logs in to SIP. It is a reference to a Role element. Because this is a required attribute, user must be defined as having at least one role. If the “initial role” does not also appear in the list of roles for the user, it is implicitly added to the end of the roles for that user.
`defaultUser`	An attribute of `User`, `defaultUser` is a flag indicating that the user entry should be used as the “default user.” This means that when a user logs in with a valid login name and there is no `User` entry for that login name and there is no portal view file named `conf/share/views/`*`login`*`.xml`, the user has all roles defined for the default user.

Table 14-4 Role Element

Elements and Attributes	**Description**
`Role`	A child element of `UserRolePackage` element. Defines a role. A user can have multiple roles, and multiple users can share a role. The `Role` element has four child elements that together define what a user operating in the role can see and do. Child elements: • `PortalViewRef` • `EditPermission` • `MgmtData`, `MgmtDataRef`, or `DefaultMgmtData` • `Properties`

Table 14-4 Role Element

Elements and Attributes	Description
`name`	An attribute of `Role`, `name` is a unique identifier for the `Role`. The name value is used in the `href` attribute of the `RoleRef` element to refer to this role. Only one role with a given name is allowed in the User-Role Model. Subsequent roles with the same name are ignored. The name must be unique across all User-Role packages.
`title`	An attribute of `Role`, title is a descriptive name for the role. The `title` is the string that appears in the drop-down list of roles presented to this user in the user interface. The title for a role does not need to be unique.
`defaultRole`	An attribute of `Role`, `defaultRole` is a flag indicating that the role should be used as the "default role." This means that when a user logs in with a valid login name and there is no `User` entry for that login name and there is a portal view file named `conf/share/views/`*`login`*`.xml`, that view is displayed to the user and the edit permissions and management data are determined by the default role. When this role is used as the default role (because it can also be used as a normal role), the `PortalViewRef` child element is ignored. Only the first `Role` element is recognized as the default role; any subsequent `defaultRole` flags are ignored.

Table 14-5 RoleRef Element

Elements and Attributes	Description
`RoleRef`	A child element of the `Role` element. Allows multiple users to refer to the same role. The referenced role can be in any of the User-Role package files.

Table 14-5 RoleRef Element

Elements and Attributes	**Description**
`href`	An attribute of `RoleRef`, href is a reference to a `Role` element. It must be the same as the `name` attribute of a `Role` element. Can contain wildcards using Perl5 pattern matching. This allows multiple roles to be associated with the user using a single `href`. All roles that match the pattern specified by the `href` are added to the list of available roles for the user. For example, an `href` value of "`.*`" matches all roles. An `href` value of "`Acme.*`" matches all roles that start with "Acme". If a `RoleRef` with wildcards has an `EditPermissions` child element, the user gets the specified edit permissions for all roles that match the pattern.
`title`	An optional attribute of `RoleRef`, title can be used to override the displayed role name as seen by the current user. If the title is specified for a `RoleRef` that contains wildcards, it serves as a prefix for the title of each of the roles that the regular expression matches.
`EditPermissions`	An optional child element of `RoleRef`. Allows a particular user to override the edit permissions defined on the role itself.

Table 14-6 PortalViewRef Element

Elements and Attributes	**Description**
`PortalViewRef`	A child element of the `Role` element. References the portal view for this role. The portal view determines the set of tabs and modules that the user sees.

Table 14-6 PortalViewRef Element

Elements and Attributes	Description
`href`	An attribute of `PortalViewRef`, href specifies the file containing the `PortalView` element. The value of `href` is interpreted as a filename relative to the `conf/share/views` directory. It can include subdirectories. For example, the `href` value “`NetOperator.xml`” is interpreted as the file “`conf/share/views/NetOperator.xml`. The file so specified must be an XML document with `PortalView` as the root element.

Table 14-6 PortalViewRef Element

Elements and Attributes	Description
`copy`	An optional attribute of `PortalViewRef`, `copy` allows multiple roles to share a portal view file, and yet have a new copy of the portal view created if the portal view is modified (through GUI editing) through one of the roles. For example, you could set up one `NetOperator.xml` portal view file and have 50 roles (one each for 50 customers) share the file. Any changes made in `NetOperator.xml` would be seen by all 50 roles/customers. However, you could use the `copy` attribute to give editing permissions to five of these roles/customers in such a way that changes do not affect the original view file. Note that edit permissions are defined through the level. The "`copy`" attribute just determines where the changes go. When each of these five customers edits the portal view, it would be saved as a new portal view file under the file name specified by the `copy` attribute. Thus, the `copy` attribute specifies the filename to use to save a modified version of the original portal view. The value of `copy` is interpreted as a filename relative to the `conf/share/views` directory. It can include subdirectories. For example, the `href` value "`Apex/NetOperator.xml`" is interpreted as the file "`conf/share/views/Apex/NetOperator.xml`. The file specified by "copy" may or may not exist. SIP will create it as needed. Once the "copy" version of the portal view has been created, the role will use that file as the view file for the role (that is, it will no longer use the portal view file specified by the `href` attribute). Note that the `create_role_db` command creates view directories required by "copy" view files if they do not already exist. In the example just cited, it would create the directory `conf/share/views/Apex`.

Table 14-7 EditPermissions Element

Elements and Attributes	**Description**
`EditPermissions`	A child element of the `Role` element. Specifies the editing operations the user can perform through the SIP graphical user interface (GUI).
`level`	An attribute of `EditPermissions`, `level` specifies the level of editing operations permitted. Each level consists of an implicit set of operations. The levels are ordered. Each level allows all operations defined by the previous level. The three levels are: • `ReadOnly` • `UserPreferences` • `ViewAdmin` For an explanation of each level see Table 14-10 on page 284.

Table 14-8 MgmtData Element

Elements and Attributes	**Description**
`MgmtData`	A child element of `UserRolePackage` element. Specifies what management data will be presented through the portal. The `MgmtData` element can appear directly with a `Role`, or it can be defined independently and referenced by multiple roles. The `MgmtData` element is essentially a list of organizations (or customers) as defined in the customer model in the `OVPortalConfig.xml` file. This list of organizations is translated at run-time into a list of resources (services, nodes, interfaces) that determine what information is presented to the user. For examples of valid `MgmtData` element values, see Table 14-11 on page 287.

Table 14-8 MgmtData Element

Elements and Attributes	Description
`name`	An attribute of `MgmtData`, `name` is the unique identifier for the `MgmtData`. This is the value used in the `href` attribute of the `MgmtDataRef` element to refer to this management data specification. Only one management data specification with a given name is allowed in the User-Role Model. If a subsequent one is found with the same name, it is ignored.
`OrganizationFilter`	An optional child element of the `MgmtData` element. Allows multiple organizations (or customers) to be specified for the “management data.” Tip for SIP 1.0 users: `OrganizationFilter` is the same as the SIP 1.0 `CustomerFilter`.
`orgName`	An attribute of `MgmtData`, `orgName` specifies the name of an organization (or customer) to use to determine the management data to present. This is provided as a shortcut so that a `OrganizationFilter` element does not need to be defined if there is only a single organization for the “management data” definition. If both the `orgName` attribute and a child `OrganizationFilter` is specified, the organization specified by “`orgName`” is added to the list of organizations listed in the `OrganizationFilter`.
`MgmtDataRef`	A child element of a `Role` element. MgmtDataRef is a way of specifying the management data for a role. It allows multiple roles to refer to the same “management data.” The referenced “management data” can be in defined in any of the User-Role package files.
`href`	An attribute of `MgmtDataRef`, `href` is a reference to a `MgmtData` element. It must be the same as the `name` attribute of a `MgmtData` element.

Table 14-8 MgmtData Element

Elements and Attributes	**Description**
`DefaultMgmtData`	A child element of the `Role` element. An indication that the role should use the `MgmtData` element specified by the `defaultMgmtData` attribute of the `UserRolePackage` element. This element is used to make this “inheritance” explicit.

Table 14-9 Properties Element

Elements and Attributes	**Description**
`Properties`	An optional child element of the `Role` element. A list of `Property` elements.
`Property`	A child element of the `Properties` element. Specifies a role property name and value that can be used by modules, including the generic module.
`name`	An attribute of `Property`, specifies the name of the role property.
`value`	An attribute of `Property`, specifies the value of the role property.

Creating Roles

Roles are defined in User-Role package configuration files. These files contain some set of users and roles and together define all the users and roles that make up the User-Role Model. All users and roles can be defined in a single file, or they can be partitioned into multiple files (for example, one for each customer or organization).

The package files define the `UserRolePackage` element shown in Figure 14-5 and described in Table 14-2. You can name the package files anything you want as long as you reference them in the index file.

Creating a User-Role Package and Adding It to the User-Role Model

1. In the `roles` directory, create a User-Role package file by copying and renaming the supplied `package.xml` file. This file is supplied as a starting point for new User-Role packages and is located in the following directory:

   Windows NT/2000: `<`*`install_dir`*`>\conf\share\roles\`
   UNIX: `/etc/opt/OV/SIP/conf/share/roles/`

2. Add the new package to the User-Role Model by adding an entry for it in the following file:

   Windows NT/2000: `<`*`install_dir`*`>\conf\share\roles\index.xml`
   UNIX: `/etc/opt/OV/SIP/conf/share/roles/index.xml`

   For example:

   `<UserRolePackageRef href="`*`new_package`*`.xml"/>`

3. Save and close the `index.xml` file.

4. In the package file, define the new roles, referring to the following instructions and to the `Role` element information in Table 14-4 on page 275.

Choosing the Portal View for a Role

A portal view defines what someone in a given role is allowed to see through the portal. Assuming that you have already created portal views for the roles that you are now creating, determine which portal view you want to associate with each role. Portal view files are stored in the following directory:

Windows NT/2000: `<install_dir>\conf\share\views\`
UNIX: `/etc/opt/OV/SIP/conf/share/views/`

If you have not yet created portal views, see “Introduction to Portal Views” on page 146.

For a description of the `PortalViewRef` element and attributes, see Table 14-6 on page 277.

Configuring a Role So Changes Are Not Made to an Original Portal View File

Refer to the `copy` attribute of the `PortalViewRef` element in Table 14-6 on page 277.

Choosing the Level of Edit Permissions for a Role

A role must have an edit permissions level associated with it. Whereas a portal view defines what someone in a role is allowed to *see* through the portal, edit permissions defines what they are allowed to *do* through the portal—what level of editing they are allowed to perform through the user interface.

For a description of each edit permissions level, see Table 14-10 on page 284. For a description of the EditPermissions element and its attributes, see Table 14-7 on page 280.

Table 14-10 Editing Permission Levels

Level	Description
ReadOnly	A user with `ReadOnly` edit permissions can make no persistent editing changes. This user will see changes that have session scope, such as last tab and rollup/rolldown per module instance. This user will not see the `[Options]` button on the main portal page.

Table 14-10 Editing Permission Levels

Level	**Description**
`UserPreferences`	A user with `UserPreferences` edit permissions can edit the following portal attributes: • Display Name (in the portal welcome banner) • Color Schemes These changes have effect only for the user. Other users of the role will see either their own user preferences or the defaults defined in the portal view file. User preferences are saved in the file `conf/share/users/`*`login`*`.xml`. Note that because changes to “display name” and “color scheme” through the GUI are modified as user preferences, the default “display name” and “color scheme” in the portal view file can only be changed by directly editing the portal view XML file.

Table 14-10 Editing Permission Levels

Level	Description
`ViewAdmin`	A user with `ViewAdmin` edit permissions can perform all the operations defined for the UserPreferences level, as well as edit the following portal attributes: • Refresh rate • Show Display Name • Show Date/Time • Default tab In addition, a user with ViewAdmin permissions can perform the following edit operations: • Add, delete, rename tabs. • Add module instances from a list of configured module instances. • Delete module instances. • Add any module, resulting in a module instance with the default configuration. • Configure module instance attributes. The changes made by this user have effect for all roles that reference the portal view being edited. These changes are saved in the portal view file: `conf/share/views/`*`viewname`*`.xml`.

NOTE Operations that are not supported by the Edit GUI (e.g., reordering tabs and modules) need to be done by direct editing of the XML files.

Overriding the Edit Permissions Level

If you want to change the edit permissions level for a user but you do not want to create a new role, you can override the edit permissions defined on the role itself. For more information, see the EditPermissions element in Table 14-5 on page 276.

Defining Management Data for a Role

Before you define the management data for a role, you need to verify that the customer and their associated resources are defined in the Customer Model. In the OVPortalConfig.xml file, the CustomerModel hrefs indicate the names and locations of the customer model files. Look in those files and other customer model data sources and verify that the customer (organization) and resources are defined.

Table 14-11 Examples of Valid MgmtData Element Values

Data to Be Presented	
All Data	All known data should be presented; no data should be filtered out. This is a MgmtData element that has no orgName attribute and no OrganizationFilter child element. `<MgmtData/>`

No Data	No data should be presented. This is a MgmtData element that has no orgName attribute and an empty OrganizationFilter child element.
	
	`<MgmtData>`
	`  <OrganizationFilter/>`
	`</MgmtData>`

Data for a Single Organization	Data for a single organization (or customer) should be presented. This is a MgmtData element that has an orgName attribute and no OrganizationFilter child element. `<MgmtData orgName="Acme"/>`

Table 14-11 Examples of Valid MgmtData Element Values

Data to Be Presented	

Data for Multiple Organizations	Data for multiple organizations (or customers) should be presented. This is a `MgmtData` element that has a non-empty `OrganizationFilter` child element.
	
	`<MgmtData name="GoldCustomers">`
	  `<OrganizationFilter>`
	    `<OrganizationRef href="Acme"/>`
	    `<OrganizationRef href="Nabob"/>`
	    `<OrganizationRef href="Aureum"/>`
	  `</OrganizationFilter>`
	`</MgmtData>`

Setting Role Properties

Refer to “User-Role Substitution” on page 224.

Creating Users

Like roles, users are defined in User-Role package configuration files. Again, these files contain some set of users and roles and together define all the users and roles that make up the User-Role Model. All users and roles can be defined in a single file, or they can be partitioned into multiple files (for example, one for each customer or organization).

Creating Users

1. In the `roles` directory, open the User-Role package file in which you want to define the users. This file is somewhere under the following directory:

   Windows NT/2000: `<install_dir>\conf\share\roles\`
   UNIX: `/etc/opt/OV/SIP/conf/share/roles/`

2. In the package file, define the new users, referring to the following instructions and to the `User` element information in Table 14-3 on page 274. For example:

   ```
   <User name="operator" displayName="Operator"
   initialRole="NOC"/>
   ```

   End the user XML definition with either “`/>`” or `</User>`.

NOTE Each user must have an `initialRole`. This attribute is mandatory and specifies the role that the user will be in when he or she first logs in to SIP.

Assigning Roles to Users

After you have defined the roles, you can assign to each user which roles (in addition to the `defaultRole`) you want them to have.

- Use the `RoleRef` element and `href` and `title` attributes to associate a role with a given user. For example:

  ```
  <User name="operator" displayName="Operator"
        initialRole="NOC">
    <RoleRef href="NOC"/>
  ```

```
  <RoleRef href="AcmeTechnical" title="Acme Technical"/>
  <RoleRef href="AcmeBusiness" title="Acme Business"/>
</User>
```

The `title` attribute is only needed if you want to override the `title` defined for the role.

Assigning Multiple Roles Using Wildcards

The `RoleRef` `href` can contain wildcards using Perl5 pattern matching. This allows multiple roles to be associated with the user using a single `href`. All roles that match the pattern specified by the `href` are added to the list of available roles for the user. For example, an `href` value of "`.*`" matches all roles. An `href` value of "`Acme.*`" matches all roles that start with "Acme".

If a `RoleRef` with wildcards has an `EditPermissions` child element, the user gets the specified edit permissions for all roles that match the pattern.

Updating the User-Role Model

Updates to the User-Role Model are made by running the `create_role_db` command, as described below. The changes are immediately available to the portal.

1. Any time you update the User-Role configuration files, you must run `create_role_db`.

NOTE For the command to work from outside the `bin` directory, add the following to your PATH variable:

Windows NT/2000: `%SIP_HOME\bin`
UNIX: `/opt/OV/SIP/bin`

2. Correct any errors that are detected, and run the command repeatedly until the User-Role Model is satisfactory.

Verifying that the Customer Portal Works as Expected

After you have configured the users and roles and updated the User-Role Model, you can log in as the user and look at the portal to verify that it is configured correctly.

SIP provides a module called User/Role Information that can be used to verify that things are as expected.

In the portal, switch to the role that you want to test. From the module [Add] button at the bottom of the Tab, add the `User/Role Information` module.

NOTE There is a `details` attribute on the User/Role Information module that can be set to “yes” to get more detailed information (e.g., the name of the view file associated with the role).

15 Maintaining SIP

Chapter Contents

Administration tools are available to help you troubleshoot portal problems that may arise. Tools include: log files, tracing, and a CGI token to help you troubleshoot performance problems on a per-module basis. A graphical view of the Service Information Portal directory structure is also available.

Using the Portal Log File for Troubleshooting

A portal log file is available to help you troubleshoot problems with the portal. It is located here:

Windows NT/2000: `<install_dir>\log\sip.log`
UNIX: `/var/opt/OV/SIP/log/sip.log`

This file consists of one-line entries that are of WARNING or ERROR severity level. (You cannot change the logging level.)

1. "error" - logged when a problem prevented the framework or module from completing a task.
2. "warning" - logged to indicate that a problem occurred, but the framework or module was able to continue, possibly using a default value or some assumption

Each line of the `portal_log` file is in a standard format, beginning with "error" or "warning":

```
error MODULENAME THREAD_NUM TIMESTAMP MESSAGE
```

as in this example:

Figure 15-1 Example of Error Entry in sip.log File

```
error NMConfig  Thread-21  987715299842  There are no NNM stations configured
                                         in nmConfig.xml. At least one station
                                         must be configured for NNM modules to
                                    operate.
```

Maintenance of the sip.log File

The `sip.log` file rolls over to `sip.log.old` under two circumstances:

- when SIP is restarted
- when the log file reaches its maximum size.

Note that only one back copy of this file is kept.

The maximum file size is controlled via the `maxLogSize` attribute in `OVPortalConfig.xml`. This attribute specifies the maximum log file size in bytes. The DTD specifies "10000000" (10 million) as the default value, so the attribute is not required.

Using Portal Tracing for Troubleshooting

Turning portal tracing on supplies a large amount of extra data that may be used for debugging problems. It is primarily intended for use by HP support.

When tracing is turned on, it is turned on portal-wide for all users.

To change the tracing level for the portal, edit the `OVPortalConfig.xml` file, located in the following directory:

Windows NT/2000: `<install_dir>\conf\portal\framework\`
UNIX: `/etc/opt/OV/SIP/conf/portal/framework/`

By default, the tracing level is set to "none":

```
<OVPortalConfig
   tracingLevel="none"/>
```

The possible levels are: none, error, warning, tracing, and verbose.

Setting the trace level to "tracing" or "verbose" will cause entries to be written to the portal trace file. This file is located at:

Windows NT/2000: `<install_dir>\log\sip.trace`
UNIX: `/var/opt/OV/SIP/log/sip.trace`

A trace level of "tracing" results in entries that record key steps during the portal’s execution. A trace level of "verbose" does the same, but with greater detail.

Maintenance of the trace.log File

The `trace.log` file rolls over to `trace.log.old` under two circumstances:

- when SIP is restarted
- when the log file reaches its maximum size.

Note that only one back copy of this file is kept.

The maximum file size is controlled via the `maxLogSize` attribute in `OVPortalConfig.xml`. This attribute specifies the maximum log file size in bytes. The DTD specifies "10000000" (10 million) as the default value, so the attribute is not required.

Monitoring the Size of NNM’s snmpCollect Database

If automatic configuration of NNM SNMP data collection has been enabled, periodic trimming of the NNM SNMP data collector database will be necessary. For details, see "Monitoring the Size of NNM’s snmpCollect Database" in the *Configuring NNM* manual (`Configuring_NNM.pdf`).

Determining the Amount of Time It Takes to Display Each Module

If you are having performance problems with a customer's portal, you can determine the amount of time each module takes to load by using the Timer CGI token.

In your internet browser, run:

`http://<yourmachinename>/ovportal?Timer=yes` (or `true`)

This will show the time in milliseconds to display each module. To turn off timing, run:

`http://<yourmachinename>/ovportal?Timer=no` (or `false`)

Certain SIP modules require significant initial load times because a large amount of data is passed over the network when the module is first accessed. As a result, the first person displaying these modules experiences the longest delay. To enhance your customer's experience, log into the portal following any restart of SIP and open any portal view that contains the following modules (if used in your environment):

- A Network Device Health module
- One of the following modules:
  - Service Graph
  - Service Browser
  - Service Health
  - Service Card

If you open it first, the required information is already cached when your customers access their portals. Your customers won't experience the delay. It is not necessary to perform a log in for each portal user. A single log in, viewing the above listed modules, is sufficient.

NOTE If a tab page containing one of the NNM modules remains blank with the progress bar partially loaded, one of the NNM management stations (that SIP gathers data from) may be in the early phase of an NNM backup procedure.

Wait for NNM's backup to proceed beyond the `ovpause` state. The modules display when the NNM management station issues an `ovresume` command. If the browser timeout limit is exceeded while you are waiting, you must press `[Refresh]` to display the modules.

For more information, see the Troubleshooting section of *Presenting NNM Data* (`Presenting_NNM_Data.pdf`).

Removing a Customer Portal

You have two options for removing a customer from the portal:

- Remove only the customer’s access.
- Remove the customer’s configuration entries in addition to the customer’s access.

Removing the Customer’s Access

If the goal is solely to prevent the customer from accessing the portal, you simply need to remove the appropriate user(s) from the authentication mechanisms access list. For instance, if password file authentication is used, remove the user(s) from the password file. For information about authentication providers supported by SIP, see Chapter 6, “Configuring an Authentication Provider,” on page 63.

Removing the Customer’s Configuration and Access

In addition to option 1, you may:

1. Remove the appropriate user(s) from the portal User-Role package file(s). This removed the authorization for those users.
2. Remove any roles in the User-Role package files(s) created solely for this customer. If you do this, you should also do step 1.
3. Run the `create_role_db` command to update SIP’s User-Role database.
4. Remove this customer’s organization entries from the customer model. If you do this, you should also do steps 1 and 2.
5. Remove any Portal View files created solely for this customer. If you do this, you should also do steps 1 and 2.

Be careful not to remove Roles or Portal View files used by other portal users. For this reason, and the possibility that you may want to reactivate this customer’s portal in the future, you may wish to leave these configurations in place. As long as authentication access has been removed, the customer will not have access to the portal.

NOTE If automatic NNM SNMP data collection configuration has been enabled, no special action is necessary other than option 1 -- disallowing access to the portal. In time (by default 30 days), the automatic configuration mechanism will detect that the customer's data is no longer needed and will remove the corresponding configuration entries -- providing no other customer needs them.

Uninstalling HP OpenView Service Information Portal

Be aware that uninstalling SIP will remove all of your portal configurations. If you want to save your customized files and directories, see “Removing and Reinstalling Service Information Portal” on page 303 BEFORE uninstalling SIP.

On Windows NT and 2000

The state of your IIS Admin Service after uninstall will be the same as before uninstall.

1. `Start`: `Settings -> Control Panel -> Add/Remove Programs`.
2. Scroll down and select HP OpenView SIP 2.0 and click [`Add/Remove`].

On HP-UX and Sun Solaris

1. As root, uninstall the software by running the command: **`/var/opt/OV/SIP/install/removesip`**
2. When prompted to continue with the removal, type "y" and press **Enter**.

Removing and Reinstalling Service Information Portal

When SIP is uninstalled, all files and directories are removed, including any files that you added or customized. If you want to save your customized files, use the recommended steps below.

1. If you have files and customizations that you want to save, save them to a directory structure outside the SIP directory structure.

   When the portal is reinstalled, your modified and customized files in and below the following directories are automatically overwritten. Here are the directories that most likely contain customized files:

   - etc (password file)
   - conf (configuration files)
   - registration (integrated modules)
   - jsp (custom headers, footers, and login pages)
   - htdocs (customized help topics, web pages, .css files, etc.)
2. Uninstall SIP, as instructed on page 302.
3. Selectively copy the customized files that you want to keep back to the SIP directory structure.

NOTE Be aware that you may need to change the permissions on the files that you move back into the SIP directory structure. It is necessary for certain files to be writable by the web server for GUI editing to work properly.

Checking the Contents of the Role Database

After running `create_role_db`, you can run the `test_role_db` command to verify the contents of the created role database.

- To find out the configured default user and default role, run:
  `test_role_db`
- To see the database entry for a particular user, run:
  `test_role_db -u <`*`user`*`>`
- To see the database entry for a particular role, run:
  `test_role_db -r <`*`role`*`>`

Glossary

A-B

authentication The process by which a user identifies and validates him/herself to the system.

authentication provider A configured component of the system that authenticates users that attempt to use the system.

authorization The granting of access privileges to an authenticated user that determines what the user can see and do while logged into the system.

C

configuration The combination of settings of software parameters and attributes that determine the way the software works, the way it is used, and/or how it appears.

configuration file A file that contains specifications or information that can be used for determining how a software program should look and operate.

configure To define and/or modify specified software settings to fulfill the requirements of a specified environment, application and/or usage.

current role The currently selected role for a logged in SIP user. The Role drop-down list box in the portal button bar shows the current role, and allows it to be changed.

customer data filtering The use of attributes as a mask for constraining the data that is to be acted on, used, or displayed in the user interface. In HP OpenView Service Information Portal, the information that is presented in the user interface for a given user login is filtered by the specific management data that is associated with a given role. Customer data filtering can also be described as “customer segmentation.”

customer model A mapping of customers to resources, where resources are associated hosts, interfaces, and services. A customer model can be defined in several XML files, or a mix of programs and files.

SIP 2.0 uses the so-called “Simple Customer Model” that is defined via an XML DTD.

customer model data source A configured URL or file that provides the mappings or partial mappings of customers to resources.

customize To design, construct and/or modify software to meet the needs and preferences of a particular customer or user. For HP OpenView Service

Information Portal, customizing is synonymous with assigning to customers what will be displayed to them. Customization tasks include customizing content, tabs, and tab layout, customer filtering, and the setting of options.

customization The process of designing, constructing and/or modifying software to meet the needs and preferences of a particular customer or user.

D

default role Any role in the User-Role Model that has the "defaultRole" attribute set to "yes." The default role is the role that is selected when a user logs into the portal and does not have a role configured for them. This is used only if there is no explicit user entry but there is a portal view file named for the login user. The default role is effectively a way to enable the "user-specific view file" mechanism for logging in. If no default role exists, there is no role to associate with the "user-specific view file" and the user is not authorized to use it. Only one role may be specified as the default role.

default user Any user in the User-Role Model that has the "defaultUser" attribute set to "yes". This user is selected when no user is found for a login after a user was authenticated.

E

edit permissions That which determines the editing operations that are available to a user through the program interface.

edit permissions level A group of operations that a user is authorized to perform through the program interface. Each level includes all the operations defined by the previous level and adds some additional operations.

extensible Software functionality whose capability, scope or effectiveness can be increased.

extend The act of increasing the capabilities, scope, and/or effectiveness of a program. The capabilities of HP OpenView Service Information Portal can be extended through the generic module and through the writing of XML.

F

filter A set of attributes and values that act as a pattern or mask through which data is passed. Filters allow matching-relevant information to be extracted and acted on while non-matching-irrelevant information is blocked.

L

login The string used to identify a user for authentication purposes.

M

Management Data Filter The security mechanism that defines what data is displayed through the portal.

message A communication using text and/or images. In HP OpenView Service Information Portal, messages are presented to customers via the message board module.

Message Board A module that is used for presenting messages to customers.

message content The information that is presented in a message. Message content may include text and/or graphics.

module A self-contained software component that performs a specific type of task or provides for the presentation of a specific type of data. Modules can interact with one another and with other software. In HP OpenView Service Information Portal, modules present specific sets of functionality to the user through the portal framework. Examples of modules include the Message Board, Service Browser, Network Device Health, and the Alarm Module.

module instance An instantiation of the module in a portal view file. Module Instance will likely differ from other instances in the portal view file by changing the XML description for that instance. For example, one could have an instance of the Alarms module displaying "All Alarms" and another instance displaying "Router Alarms."

P

page A single display or presentation of information on the World Wide Web. Typically a web page consists of an HTML file, referenced graphics files, and associated scripts.

portal A web site that provides a variety of different types of information and which serves as a gateway to other web sites. HP OpenView Service Information Portal consists of the framework and modules. It provides information and access to other websites through the modules and submodules.

portal framework A program that acts as the basic structure to support other software modules or programs that provide additional functionality for the user. In HP OpenView Service Information Portal, the framework provides a mechanism for the modules to present information to the user. The framework also provides the structure for customization, configuration and extension of the portal's functionality.

portal view Consists of modules, tabs, and view properties. Each user account that is set up by an administrator has one or more roles, and each role is associated with one portal view. A portal view can be shared by multiple users.

portal view file A configuration file that contains specifications or information that can be used for determining how software should look to a given user. In HP OpenView Service Information Portal, portal view files are XML files that contain all information needed to render a portal view.

R

role That which defines what a user can see and do through the portal at a particular point in time. A role can be shared by multiple users.

role properties An extensibility mechanism used to provide authorization information associated with a role and that is not defined in the predefined role XML elements.

S

skin A setting that controls the visual appearance of the user interface. Skins can determine the color scheme, fonts, graphics, and other attributes presented in the user interface. The skins in the HP OpenView Service Information Portal are based upon W3C's Cascading Style Sheets. Existing skins may be extended, or new ones added to the 'css' files located under the `htdocs/styles` directory.

submodule A portion of a software module that provides a subset of the functionality provided by the module. A submodule performs a specific task or presents a specific set of data. In HP OpenView Service Information Portal submodules present different variations of the type of data presented by the Module. For example, one submodule of the Network Device Health Module presents Network health for Routers while another submodule presents Network health for Servers.

T

tab A page in the user interface that has a small index-card like projection. The projection typically presents the name for the page and allows navigation to the page by clicking. In HP OpenView Service Information Portal the main portal pages have tabs. Service Information Portal provides one tab, but multiple tabs can be created using the Customize Content option on the Options page. Similar types of modules can be grouped together using tabs.

U

user preferences The attributes that are associated with a specific user. In Service Information Portal, user preferences control the name that appears in the portal header, and the color scheme, or “skins” that control the portal colors and fonts.

User-Role Model An authorization model that achieves security by associating users with roles and assigning to each role what the user is able to see and do. The User-Role Model consists of all User Role Package files.

Index