HP OpenView Telecom Extensions

Configuration and Maintenance Guide

HP-UX, Solaris, Windows NT®

Manufacturing Part Number : J5121-90004
June 2002

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.

HEWLETT-PACKARD COMPANY
United States of America

Trademark Notices.

Adobe® is a trademark of Adobe Systems Incorporated.

Acrobat® 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.

Contents

Contents

Contents

Tables

Tables

Figures

Figures

Contact Information

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`.

Training Information

For customer training on OV Telecom Extensions for OV Operations and OV Topology Server, contact your HP consulting representative.

For information on current HP OpenView Operations and HP OpenView Network Node Manager product training, visit:

`http://www.education.hp.com`

Submitting Documentation Defects

If you encounter *errors* in the documentation, 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).

Document Conventions

The following typographical conventions are used in the manuals provided with this product.

Font	What the Font Represents	Example
Italic	Book or manual titles and manpage names	See the *HP OpenView Telecom Extensions and Topology Server Concepts Guide* and the *ovsadc*(1M) manpage for more information.
	Emphasis	You *must* follow these steps.
	Variables that you must supply when entering a command	At the prompt type: **`rlogin` *`your_name`*** where you supply your login name.
	Parameters to a method	The *assigned_criteria* parameter returns an ACSE response.
Bold	New terms	The **distinguishing attribute** of this class...
`Computer`	Text and items on the computer screen	The system replies: `Press Enter`
	Command names	Use the `grep` command ...
	Method names	The `get_all_replies()` method does the following...
	File and directory names	`/usr/bin/X11`
	Process names	Verify that `pmd` is running.
	Window/dialog box names	In the `Active Messages` window...
	Menu commands	Select `Actions:Messages->Acknowledge All Messages`.
`Computer Bold`	Text that you must enter	At the prompt, type: **`ls -l`**

Font	What the Font Represents	Example
Keycap	Keyboard keys	Press **Return**.
`[Button]`	Buttons on the user interface.	Click `[NET]`.

In This Book

This manual describes how to configure HP OpenView Integrated Service Assurance for Communication Networks.

It assumes that OV Telecom Extensions for OV Operations has been installed and set up as described in the *HP OpenView Telecom Extensions Installation Guide*. It further assumes that you have read and understood the *HP OpenView Telecom Extensions Concepts Guide*.

This manual guides you through the following tasks:

- Planning and configuring an OVISACN entry solution.
- Configuring a telecom subagent.
- Configuring and deploying telecom templates.
- Troubleshooting the solution.

Audience

This manual is intended for a system integrator or network administrator responsible for initial configuration of the OVISACN solution. It assumes that you have user-level knowledge of the HP-UX operating system and are familiar with using GUI-based applications with mouse and menu-driven interfaces on UNIX workstations.

Manual Organization

This book contains the following chapters:

- Chapter 1, “Configuring an OVISACN Solution,” helps you develop a blueprint of the configuration details needed for a managed network solution.
- Chapter 2, “Working with a Project,” presents an overview of XML and describes the OVISACN XML project files.
- Chapter 3, “Configuring Telecom Subagent Data Collectors and Lookup Tables,” explains the procedure to configure telecom data collectors for OVO agents.
- Chapter 4, “Modifying OVO Message Source Templates,” describes how to build, modify, assign, and install message source templates to configure an OVO agent.
- Chapter 5, “Troubleshooting,” describes tools for troubleshooting problems with the solution.

1 Configuring an OVISACN Solution

This chapter helps you develop a blueprint of the configuration details needed for a managed network solution and points you to the other sections of this manual that describe the implementation of these details.

Solution Deployment Options

The following deployment options are available for HP OpenView Integrated Service Assurance for Communication Networks:

- Entry solution

  For an entry solution, only the OV Telecom Extensions and OVO products are installed and configured. This solution is appropriate for small and moderately-sized networks that have not yet used a telecom-specific network management tool.

  In an entry solution, OV Telecom Extensions adds data collector functionality that communicates with telecom element management systems and network elements. All alarms received from these devices appear in the OVO message browser.

- Standard solution

  For a standard solution, the OV Telecom Extensions, OV Topology Server, and OVO products are installed and configured. This solution is appropriate for most communication service provider networks.

  In a standard solution, OV Telecom Extensions adds data collector functionality that communicates with telecom element management systems and network elements. This data collector forwards the received alarms to the topology server for processing and correlation into problems. These problems appear in the OVO message browser and the topology GUI problems presenter.

NOTE This book only describes the entry solution. For information on configuring a standard solution, see the *HP OpenView Topology Server Configuration Guide*.

Planning Your Solution

This section gives a high-level description of the steps to plan an HP OpenView Integrated Service Assurance for Communication Networks management solution. Throughout the planning process, communication between the system integrator and the network administrators is crucial to designing a network management solution that is appropriate to your environment and network management needs.

Complete this planning process before beginning to configure your solution. Your answers to some questions may require reworking answers to previous questions.

1. Visualize the results.

   Identify end-user needs and define what the solution should look like when completely configured. Consider the following questions:

   - How many operators will the solution support?
   - How many physical locations will operators and administrators manage?
   - How will the work be divided among operators: by equipment type, by location?

2. Assign managed devices to agents.

   Determine how many agents are needed to monitor the network state and which devices each agent will manage. Consider the following questions:

   - What are the device types to be managed?
   - Where are these devices located? Generally, the agent is located near the physical equipment.

3. Identify the alarms to track.

   Examine lists of the alarms that each managed device emits and determine which alarms provide important information regarding the state of your network. Consider the following question:

   - What information helps operators identify problems in the network?

Configuring an Entry Solution

This section gives a high-level description of the steps you should follow to configure an entry solution.

1. Plan the solution before you start working with configuration files.

   For more information, see “Planning Your Solution” on page 22.

2. Create a new project for your configuration files.

   For more information, see “Creating a New Project” on page 39.

3. Create and add an *Agent*.xml file to the *Project*.xml file for each agent in the managed network.

   For more information, see “Adding Agent Files to a Project” on page 40.

4. Define the events to be received.

   a. Edit each *Agent*.xml file to define the data collectors.

      In this file, add details about data collectors, including:

      - The data collectors to be managed. Define data collectors in terms of their sources.
      - The equipment list, consisting of equipment names and record formats.
      - Record formats, which are defined by two strings: beginning and ending markers associated with key events. Use the key event details to create the begin-end markers.

      For more information, see “Configuring a Data Collector” on page 54.

   b. Optionally create one or more lookup tables in the *Agent*.xml files.

      For more information, see “Configuring Lookup Tables” on page 69.

   c. Deploy the *Agent*.xml files.

      For more information, see “Deploying Agent and Data Collection Data” on page 72.

d. Apply the agent configuration files to OVO server.

   For more information, see “Applying Agent and Data Collection Data” on page 73.

5. Configure the received event messages to appear in the OVO message browser.

   a. Modify source templates from the OVO administrator GUI.

      For more information, see “Modifying Basic Templates” on page 90.

   b. Deploy the modified templates.

      For more information, see “Enabling Templates” on page 96.

6. Test the event configuration.

   Simulate the sending of events into the configured data collectors. You should see these events displayed as formatted messages in the OVO message browser.

2 Working with a Project

This chapter describes XML in general terms, explains what a project is, and describes how to:

- Create and modify a new project.
- Validate, deploy, and apply project data.
- Reapply a project.
- Restore backup configuration files.

Overview of XML

HP OpenView Integrated Service Assurance for Communication Networks uses XML files as the means of storing and retrieving most of its configuration data for OV Telecom Extensions for OV Operations.

If you are already familiar with XML and do not need an overview of XML, go to “What Is a Project?” on page 35.

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 science. New technologies are being developed that understand the structure of XML vocabularies in order to operate on the data.

XML is human readable yet structures 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 Consortium. Visit the W3C site for more information: `http://www.w3.org/`

Benefits of 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.
- 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 web technology; like HTML, communication occurs over HTTP.

Sample XML Vocabulary

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>12.95</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 (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).

Sample DTD Vocabulary

```
<!ELEMENT BookCatalog (Book*)>

<!ELEMENT Book (Author, Title, Price)>

<!ATTLIST Book category CDATA #REQUIRED>

<!ELEMENT Author (#PCDATA)>

<!ELEMENT Title (#PCDATA)>

<!ELEMENT Price (#PCDATA)>
```

Sample 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>12.95</Price>
   </Book>
</BookCatalog>
```

How XML Data Is 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 an XML parser that notifies you as it processes the XML.

What Is a Project?

The concept of a project is central to the OV Telecom Extensions configuration process. A project consists of several files that describe the telecom environment. These files are edited in one location and used to generate the actual configuration files the system uses.

Projects add flexibility and conformity to the configuration process. You can modify your project files while the system is running and apply the updated configuration later. In addition, you can incrementally add to the configuration, testing the updated configuration at each step to verify that it is performing as expected.

OV Telecom Extensions processes validate the project files for content accuracy and consistency before generating the configuration files. This validation ensures that the configuration stays consistent across all of the solution.

Project Working Model

There are several steps to working with a project:

1. Edit project files using the telecom configurator or the `ovcfg*` commands.
   - For information on using the telecom configurator, see the telecom configurator online help.
   - For information on using the command line tools to work with a project, see the `ovcfg*` man pages.
   - For information on the contents of the project files, see Chapter 3, “Configuring Telecom Subagent Data Collectors and Lookup Tables.”
2. Validate the project files to ensure that they are structured correctly. This step is optional because all project files are validated as part of project deployment. For information, see “Validating a Project” on page 41.
3. Deploy the project files. This step generates the configuration files that are used by OV Telecom Extensions but does not affect the working configuration. For information, see “Deploying a Project” on page 42.

4. Apply the project. This step consists of several commands that must be completed in sequence. It copies the generated configuration files to the appropriate locations on the OVO server and agent computers. For information, see “Applying Configuration Files” on page 43.

Project File Structure

Project files are stored in a project directory. By default, project directories are stored in the `/etc/opt/OV/Telco/conf/` directory.

The project directory contains several subdirectories:

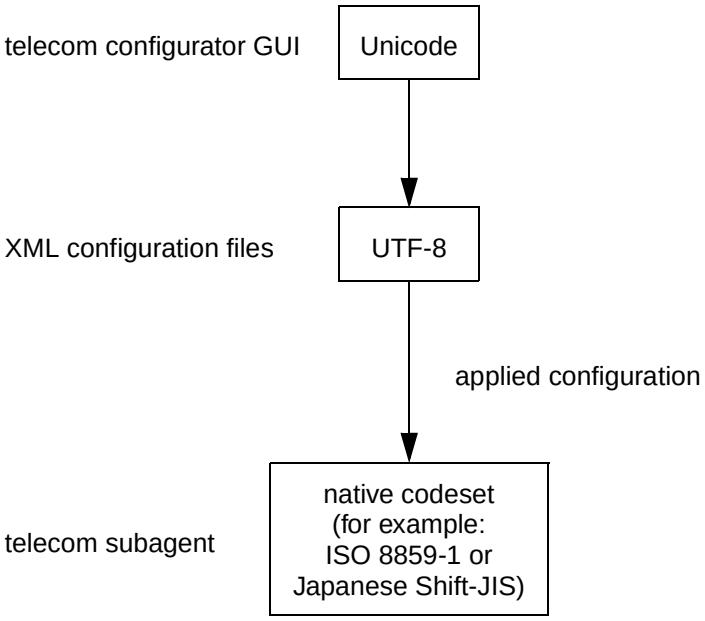
- `XML/` stores the working project files.
  - — *`Project`*`.xml` contains the names of the files in the project. This file has the same base name as the project directory. There is only one *`Project`*`.xml` file per project.
  - — The `agents/` subdirectory contains one *`Agent`*`.xml` file per installed agent. The *`Agent`*`.xml` file contains the definitions of the telecom subagent data collectors.
- `generated/` stores the files that were created during the most recent deploy operation.
- `backup/` stores the project files used during the most recent deploy operation.
- `backup`*`timestamp`*`/` stores previous versions of project files.
- `cmds/` stores optional commands for connecting to and managing a device. The data collector must be configured to use these commands.

I18N Model

OV Telecom Extensions for OV Operations uses UTF-8 data in XML configuration files. When deploying data to the runtime management systems, the management applications convert UTF-8 data to the native codeset. For example, if Shift-JIS is the native codeset on the OVO agent computer, the telecom subagent converts lookup tables from the UTF-8 codeset to the Shift-JIS codeset.

All editing of the XML configuration files must be done in the UTF-8 codeset. The telecom configurator GUI, `ovcfg*` commands, and `vi` editor all use the UTF-8 codeset.

Figure 2-1 I18N Model



The following limitations apply to the OV Telecom Extensions implementation of I18N:

- OV Telecom Extensions does not ship the Java converters necessary to convert non-UTF-8 codeset data into UTF-8 codeset. OV Telecom Extensions assumes that these are part of the JRE on each platform.
- OV Telecom Extensions has been tested in the “mono-lingual” environment, in which all management servers are in the same locale, that OVO supports.
- The only data that can be localized are the record format patterns and lookup tables in the *`Agent`*`.xml` file.

Viewing a Sample Project

To view a sample project and its XML configuration files, use the telecom configurator to view the GPRS project or examine the files in the `/etc/opt/OV/Telco/conf/GPRS/XML` directory.

NOTE The GPRS directory is created only when the optional GPRS demo content is installed. For instructions on installing the GPRS demo, see the *HP OpenView Telecom Extensions Installation Guide*.

Modifying a Project

Use the telecom configurator GUI tool or the `ovcfg*` commands to create and edit a project. The telecom configurator is the recommended configuration tool because it correctly handles I18N and project validation.

To start the telecom configurator:

1. Log on to the OVO server computer as user `root`.
2. Execute:

   `ovcfgsa`

The command line functions are described here. For information about using the telecom configurator, see the telecom configurator online help.

Creating a New Project

To create a new project, execute:

`ovcfgnewproject <`*`project`*`>`

This command creates a new project directory called *`project`* and an XML project file called *`project`*.xml. The project directory is created under the directory defined by the `TELCOCONF` environment variable in `/etc/opt/OV/Telco/telco.env`. By default, `TELCOCONF` is set to `/etc/opt/OV/Telco/conf`. Edit the value for `TELCOCONF` in `telco.env` to change where the project files are saved.

The project directory is populated with template XML configuration files created in its `XML` subdirectory. For more information, see “Project File Structure” on page 36.

Adding Agent Files to a Project

Each agent in the installation must have a corresponding *Agent*.xml file in the XML/agents directory. To add an *Agent*.xml file to the project, execute:

ovcfgproject -addAgent <*path/agent_name.xml*>

where:

- *path* is the full path to the agent file to be added. Relative paths are acceptable.
- *agent_name* is the host name of the agent to be configured.

Validating a Project

Each XML configuration file has an associated DTD file that defines its acceptable structure and syntax. The OV Telecom Extensions DTD files are shipped as contributed material, which means that HP does not directly support these files and their contents may change for future product versions. These files are installed into the `/opt/OV/Telco/contrib/dtds/` directory. Do not edit them.

It is important to validate your XML configuration files against the associated DTD files to ensure that they adhere to the predefined structure before you distribute your project data.

To validate your project, execute:

`ovcfgvalidate <`*`project`*`>`

This command validates each XML file defined in *`Project`*`.xml`. First, global data stored in `TopoModel.xml`, `TopoData.xml`, and `Mappings.xml` is validated, then each configuration file is validated against the global data.

All validation errors appear on screen. A record is stored in `/var/opt/OV/share/log/Telco/configurator.log`.

NOTE This command is optional. When you deploy the project, all configuration files identified in the *`Project`*`.xml` file are validated again.

Deploying a Project

Deploying the project generates configuration files that can be pushed to the OVO server and agent computers in your network.

To generate target configuration files from the XML configuration files, execute:

`ovcfgdeploy -agent [-force] [-verbose] <project>`

`ovcfgdeploy` converts the XML data into separate configuration files used by OVO and OV Telecom Extensions. It also outputs a summary of what has changed and what configuration data must be applied. For more information, see the `ovcfgdeploy` man page.

`The -force` option generates all new target configuration files regardless of whether the deploy should be an update or not.

The `ovcfgdeploy` command places files in three subdirectories of the project directory:

- `generated/`, where the target files are stored.
- `backup/`, where copies of the XML files are stored. This directory contains copies of project files at the time when `ovcfgdeploy` is run.

  A hidden file is also created in the `backup/` directory that is read by the `ovcfgdeploy` command. This file indicates whether the project data is new or needs to be updated only.
- `backup`*`timestamp/`*, where a copy of the backup XML files are stored. This directory contains copies of project files that were located in `backup/` at the time when `ovcfgdeploy` is run. The timestamp reflects the time of the last backup, not the time the last `ovcfgdeploy` was run.

Applying Configuration Files

The apply command copies the generated configuration files to their correct locations on the OVO agent computer. To apply the configuration files, execute:

`ovagt.apply <`*`project`*`> -n <`*`node1`*` [`*`node2 ...`*`]>`

This command pulls the agent configuration files from the OVO server computer to the appropriate file placement on the OVO agent computer. This command must be run every time the agent configuration is changed.

The command to apply the generated data detects whether the configuration files have been updated since the last apply operation. If no changes are necessary, the command quits. For more information, see the `ovoconf.apply` and `ovagt.apply` man pages.

Redeploying a Project

If the configuration process does not proceed as expected, follow these steps to reapply the configuration data:

1. Regenerate target configuration files from the XML files using the `-force` option to `ovcfgdeploy`.
2. If `ovcfgdeploy` reports any errors, correct the errors in the XML files and repeat step 1.
3. Reapply the generated configuration files to the target systems. For more information, see “Applying Configuration Files” on page 43.

Restoring Backup Configuration Files

Backups of project configuration files are automatically created each time `ovcfgdeploy` runs. Backups provide the ability to revert to a previous version of the configuration files.

To restore configuration files from a backup version:

1. Copy all configuration files from either the `backup/` directory or a `backup`*`timestamp`*`/` directory to the `XML/` directory under the project directory.
2. Deploy the project. For more information, see “Deploying a Project” on page 42.
3. Apply the generated configuration files to the target systems. For more information, see “Applying Configuration Files” on page 43.

Maintaining Backup Project Files

Each time you run the `ovcfgdeploy` command, a backup directory containing all of the project files is created. You must decide which backups to save and which to remove. You may want to automate a process that saves the backup directories to a disk and removes them from the OVO server computer periodically, for example, every month.

3 Configuring Telecom Subagent Data Collectors and Lookup Tables

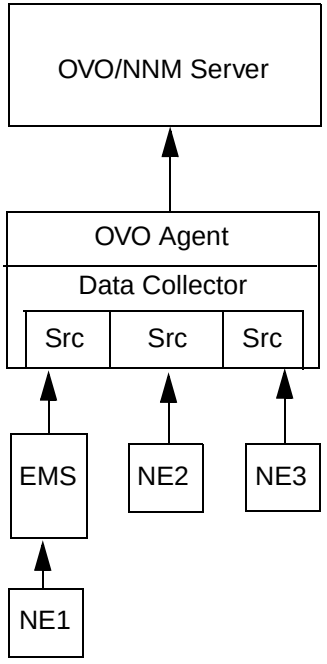
This chapter explains how to configure the telecom subagent data collector for an OVO agent. This process involves configuring the data collector source details, record formats, and lookup tables.

Introduction to Agents and Data Collectors

Network elements and element management systems are integrated into HP OpenView Integrated Service Assurance for Communication Networks at the agent level. Figure 3-1 illustrates the relationship among network elements, a telecom subagent data collector, an OVO agent, and an OVO server.

A managed network solution can have one or more agent computers connected to the management server. Typically, these agent computers are physically deployed at a site near the network elements and element management systems.

Figure 3-1 Data Collectors, OVO Agents, and OVO Server

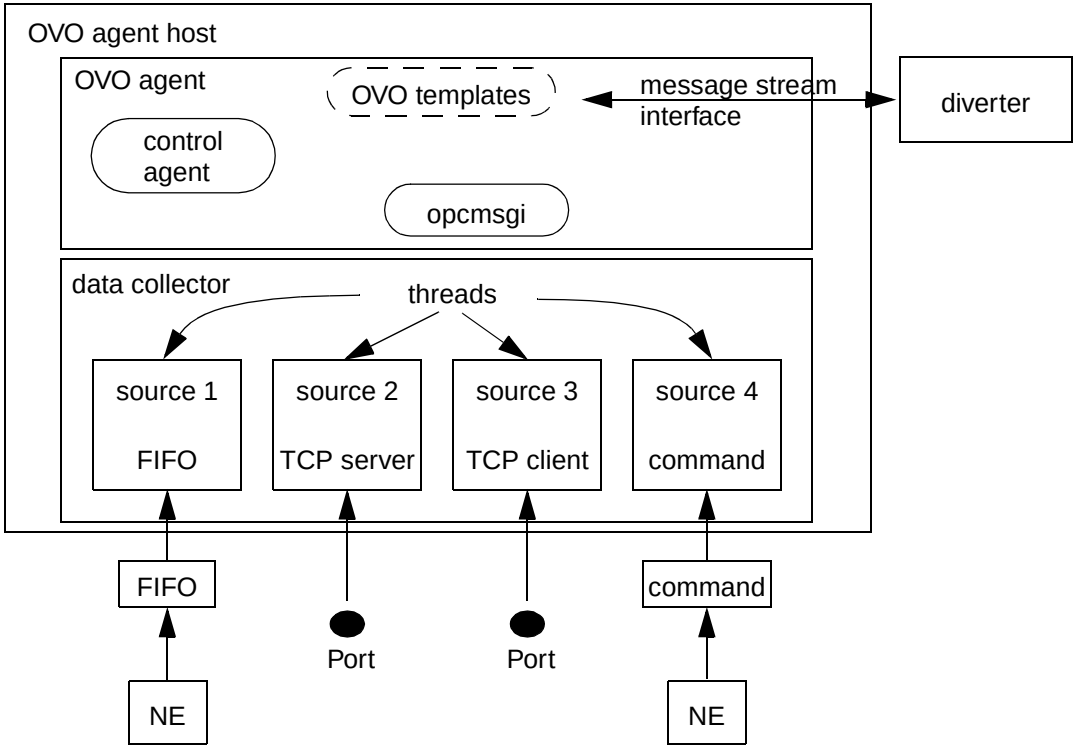


NE	Network Element
Src	Data Collector Source
EMS	Element Mgmt System

An agent receives events from network elements, and forwards them to the OVO server and its message browser. Messages may also be diverted through the telecom diverter. This is turned on by enabling the message stream interface on the OVO agent computer. Use the telecom diverter to take advantage of lookup tables.

Figure 3-2 depicts how the telecom data collector might be configured to receive alarms from telecom devices.

Figure 3-2 Sources, Data Collectors, and Agents



The telecom data collector is part of the OV Telecom Extensions telecom subagent. It must be on the same computer as an OVO agent. A data collector source (thread) connects to a managed device for the purpose of receiving event data. Each source parses the input event stream to break the stream into individual events. The source passes recognized events to

the OVO agent. When auditing is turned on, the source logs all received events (recognized and unrecognized). Each data collector can have multiple sources.

The telecom data collector supports several input connection types:

- A FIFO (named pipe) source opens a connection to a file on the managed device. The data collector source reads the event stream as it is written to the file.
- A TCP server source connects to a TCP port on the agent computer. The data collector source acts as a listener, reading events from devices that connect to this port as this data becomes available.
- A TCP client source connects directly to an assigned host and TCP port. The data collector source reads for event data and receives all data to which it is connected.
- An executed command connects to the managed device and as a proxy to forward events.

Other connection types, such as SNMP, must be configured to communicate directly with the OVO agent.

NOTE Use great care when configuring multiple network elements to a single FIFO source. For more information on FIFO source restrictions, see the `ovsadc(1m)` reference page.

Events received from network elements are processed at the data collector level to determine if they conform to predefined patterns called **record formats**. Events that fit the record format are extracted and sent to the OVO agent for further processing. The events are then presented in the OVO message browser to the operators who are managing those network elements.

Figure 3-3 shows how sample events might be parsed by the data collector. The underlined text indicates the beginning and ending markers for the defined record formats. Thus, only three events are forwarded to the OVO agent for further processing.

Figure 3-3 Sample Events

##CCF# 10/05/01 10:32:25 Switch1 CC=1 EC=ASF #E#
##SCF# 10/05/01 10:32:25 Switch1 Self Check Failure: EC=ASF SV=w #E#
##S# 10/05/01 10:32:25 Switch1 Stats: 813528929 1703199216 618906479 573 #E#
##T# 10/05/01 10:32:25 Switch1 PS=1 Temp: 78 #E#
##SC# 10/05/01 10:32:25 Switch1 System Nominal. #E#

Overview of Data Collector Configuration Data

To configure the telecom subagent to receive events from network elements and element management systems, you must name the data collectors, identify which data collectors are being monitored by which agents, identify how to process events received from devices, and further configure the agents to accept events and process them as programmed.

All agent data collector information is configured and stored in an agent XML file. One agent XML file is needed per telecom subagent computer in the solution configuration.

Each *Agent*`.xml` file contains, among other things:

- Information describing each data collector active for the telecom subagent on the *Agent* computer:
  - A list of names for the data collectors.
  - Input sources to the data collectors.
  - Details on whether the sources are FIFO, TCP/IP, or command.
  - A list of names for the sources.
  - A list of equipment types that can provide data to sources.
  - The begin and end strings, which filter unwanted events.
- Lookup tables, which convert parts of events to a more readable format.

Add and remove content in *Agent*`.xml` files with the command `ovcfgagent`. This command provides options for an administrator to add and remove:

- Data collector definitions
- Input source definitions

- Source detail definitions
- Record format definitions
- Lookup table definitions

Overview of Data Collector Configuration Tasks

For each data collector, configure the following information:

1. Name, source list, and optional data collector details

   For more information, see “Configuring a Data Collector” on page 54.
2. Sources and source details

   For more information, see “Configuring Data Sources” on page 56.
3. Equipment list

   For more information, see “Configuring Equipment” on page 63.
4. Record formats

   For more information, see “Configuring Record Formats” on page 64.

Configuring a Data Collector

Each data collector must have a unique name and at least one data source for receiving events.

Normally, there is only one data collector per agent. When you enter more than one definition of a data collector in an agent configuration file, OVO asks which data collector to start. For more information on configuring multiple data collectors for one agent, see the white paper on configuring multiple data collectors.

If the data collector process stops for any reason, the network management system loses connectivity to all devices to which that data collector was connected. For more information on the OV Telecom Extensions data collector process, see the `ovsadc` man page.

To add a data collector definition to *`Agent`*`.xml`, execute:

`ovcfgagent -addDC` *`<DC name> <source> [-<dcdetails>] xml_file`*

For each data collector, configure the following values:

- Name

  Assign a unique name to this data collector.

  The data collector includes this name in the message that it passes to the OVO agent. For more information, see “Template Fields for Telecom Devices” on page 89.

- Source list

  List the names of data sources that will pass events to this data collector. You must also define these data sources. For more information, see “Configuring Data Sources” on page 56.

- DC details table name (optional)

  Assign a unique name for the table of name-value pairs, which further defines this data collector.

- DC detail values (optional)

  Specify values for zero or more of the following names:

  — `AuditMode`

  Specify whether to log incoming events. All events received on this data collector are logged together in one file unless the audit mode has been enabled separately in a source detail specification. Possible values are `enabled` and `disabled`.

  — `AuditFileMaxSize`

  Specify the maximum size (in bytes) of the audit log file. The log file rolls over to a secondary file when the audit file size is reached. This file is uniquely named and does not override previous log data. If this value is not set, the audit log file grows without bound.

  — `AuditDir`

  Specify the directory in which to store the audit log file(s). If this value is not set, the audit log files are stored in the `/var/opt/OV/log/Telco` directory.

  — `IdleTimeThreshold`

  Specify the number of seconds (in a multiple of 5) for the data source to wait before reporting that it is idle. The value `0` disables idle time reporting. The default value is 5 seconds. If this detail is not configured, the data collector does not report that it is idle.

  — `CmdMode`

  Specify whether the data collector can receive commands from `ovsadccmd`. Possible values are `enabled` and `disabled`. When the command mode is enabled, the data collector creates a source that listens for requests from `ovsadccmd`. By default, command mode is enabled for the data collector.

  The available commands tell the data collector to reset its connection to the telecom device and control data collector tracing and logging. For more information, see the `ovsadccmd` man page.

— `CustomTranslate`

Specify whether the data collector should translate ASCII characters below the value of 40 octal (excluding tab and newline) to strings for their octal values before passing events to the OVO agent. Possible values are `enabled` and `disabled`.

— `HLPTimeout`

Specify the number of seconds (in a multiple of 5) for the data source to wait before timing out and flushing the event buffer. The default value is 5 seconds. For more information, see the `ovsadc` man page.

— `HLPBufferLimit`

Specify the maximum size (in bytes) of the buffer that the data collector will hold while waiting for a recognized end filter. The default value is 8096 bytes. For more information, see the `ovsadc` man page.

Configuring Data Sources

A data collector source defines a connection to a managed device. There are several types of data sources:

- FIFO—The data source connects to the configured file (pipe) and reads the file contents as events. There may be multiple FIFO data sources on one data collector, but only one source can connect to any given file.
- TCP server—The data source connects to the configured port and listens for connections from network elements. When a network element connects, the source reads the data on the port as events. There may be multiple TCP server sources on one data collector, but only one source can connect to any given port on a host.
- TCP client—The data source connects to the configured port and actively reads all incoming data as events. There may be multiple TCP client sources on one data collector, but only one source can connect to any given port on a host.
- Command—The data source executes a command that acts as a proxy to an event source.

For detailed information, see the appropriate section:

- "Configuring a FIFO Data Source" on page 58
- "Configuring a TCP Server Data Source" on page 58
- "Configuring a TCP Client Data Source" on page 58
- "Configuring Command Data Sources to Support Data Collector Customization" on page 60

To add a data source definition to *Agent*.xml, execute:

```
ovcfgagent -addSource <name> <sourcetype> <sourcedetails> <status> <equipment> xml_file
```

For each data source, configure the following values:

- Name

  Use the name from the source list in the data collector configuration. This name must be unique.

  The data collector includes this name in the message that it passes to the OVO agent. For more information, see "Template Fields for Telecom Devices" on page 89.

- Source type

  Specify the type of connection for this source. Possible values are FIFO, TCP, and COMMAND.

- Source details table name

  Assign a unique name for the table of name-value pairs that further defines this data source.

- Status

  Specify whether this data source can receive events. Possible values are yes and no.

- Equipment list

  List the names of equipment types that will pass events to this data source. You must also define these equipment types. The same equipment type may apply to multiple data sources. For more information, see "Configuring Equipment" on page 63.

Configuring a FIFO Data Source

To add a source detail definition to *Agent*.xml, execute:

```
ovcfgagent -addSourceDetail <source detail name> <name> <value> xml_file
```

To define a data source for a FIFO (named pipe) data stream, add the source with the SourceType value FIFO and include the following source detail in the source configuration:

- FIFOName

  Specify the full pathname of the FIFO file the data source should read.

There are also several optional names available for source detail configuration. For more information, see "Optional Data Collector Detail and Source Detail Name-Value Pairs" on page 59.

Configuring a TCP Server Data Source

To add a source detail definition to *Agent*.xml, execute:

```
ovcfgagent -addSourceDetail <source detail name> <name> <value> xml_file
```

To define a data source for a TCP server data stream, add the source with the SourceType value TCP and include the following source details in the source configuration:

- TCPMode

  Assign the value Server.

- TCPPort

  Specify the number of the IP port on which the managed device should connect to the data collector.

There are also several optional names available for source detail configuration. For more information, see "Optional Data Collector Detail and Source Detail Name-Value Pairs" on page 59.

Configuring a TCP Client Data Source

To add a source detail definition to *Agent*.xml, execute:

```
ovcfgagent -addSourceDetail <source detail name> <name> <value> xml_file
```

To define a data source for a TCP client data stream, add the source with the `SourceType` value `TCP` and include the following source details in the source configuration:

- `TCPMode`

  Assign the value `Client`.

- `TCPHost`

  Specify the name of the remote host to which the data source should connect.

- `TCPPort`

  Specify the number of the IP port on which the data source should connect to the remote host.

There are also several optional names available for source detail configuration. For more information, see "Optional Data Collector Detail and Source Detail Name-Value Pairs" on page 59.

Optional Data Collector Detail and Source Detail Name-Value Pairs

The following name-value pairs are optional. Use them as needed to configure any of the supported data collector or source types. When one of these name-value pairs is set as a data collector detail, it applies to all of that data collector's sources unless overridden by a source detail specification. When one of these name-value pairs is set as a source detail, it applies only to that specific data source and overrides any setting for the data collector detail.

For more information, see the `Agent.dtd` man page.

- `AuditMode`

  Specify whether to log incoming events. All events received on this data collector are logged together in one file unless the audit mode has been enabled separately in a source detail specification. Possible values are `enabled` and `disabled`.

- `IdleTimeThreshold`

  Specify the number of seconds (in a multiple of 5) for the data source to wait before reporting that it is idle. The value `0` disables idle time reporting. The default value is 5 seconds. If this detail is not configured, the data collector does not report that it is idle.

- `CustomTranslate`

  Specify whether the data collector should translate ASCII characters below the value of 40 octal (excluding tab and newline) to strings for their octal values before passing events to the OVO agent. Possible values are `enabled` and `disabled`.

- `HLPTimeout`

  Specify the number of seconds (in a multiple of 5) for the data source to wait before timing out and flushing the event buffer. The default value is 5 seconds. For more information, see the `ovsadc` man page.

- `HLPBufferLimit`

  Specify the maximum size (in bytes) of the buffer that the data collector will hold while waiting for a recognized end filter. The default value is 8096 bytes. For more information, see the `ovsadc` man page.

Configuring Command Data Sources to Support Data Collector Customization

The OV Telecom Extensions data collector includes a framework that supports third-party commands for communicating with managed equipment. The commands can perform device-specific functions such as logging on and configuring parameters. Each telecom data collector may have one or more sources (threads) that execute the provided commands. Configure each of these command data sources individually.

Configuring a Command Data Source

When you configure device management through commands, specify the behavior of the command data source and provide the commands to be executed. For each telecom subagent, configure one command source per managed device.

To configure an OV Telecom Extensions data collector to send commands, add the source with the `SourceType` value `COMMAND` and include the following source details in the source configuration:

- `CmdName`

  Specify the name of the command that the data collector should execute. This name includes the path in the project `cmds` directory. For more information, see “Location of Command Files” on page 62.

- `AutoRestart`

  Specify whether the command data source should restart the command if it stops prematurely. Possible values are `enabled` and `disabled`.

- `AlarmFifo`

  Specify a named pipe to which the managed device should write alarm data. This value is optional. By default, the data collector reads `stdout` as the alarm stream.

- `ErrorFifo`

  Specify a named pipe to which the managed device writes log file data. This value is optional. By default, the data collector reads `stderr` as the log file stream.

There are also several optional names available for source detail configuration. For more information, see "Optional Data Collector Detail and Source Detail Name-Value Pairs" on page 59 of the *HP OpenView Telecom Extensions Configuration and Maintenance Guide*.

Command Data Source Behavior

The data source starts the specified command and manages the command process by performing the following functions:

- If the process stops unexpectedly and the `AutoRestart` source detail is set to `enabled`, the data source restarts the process.
- If the process stops unexpectedly and the `AutoRestart` source detail is set to `disabled`, the data source exits but the rest of the data collector continues to run.

  You can manually restart the data source with a source reset. For more information see the `ovsadc` man page.

- When the data collector shuts down, it sends a `SIGTERM` signal to the command process and assumes that the process shuts down normally.

By default, the data collector reads the following data on command processes:

- It reads `stdout` as the alarm stream from the device and processes this data according to the record formats for the agent.
- It reads `stderr` as error data and logs it to the log file.

Requirements of the Commands

The commands started by the data collector may execute any functionality that the managed device supports and may be written in any programming or scripting language that runs on the target agent.

To interact correctly with the data collector, OV Telecom Extensions expects the following behavior from the commands that it executes:

- All commands and supporting content must be able to run out of the hierarchy created on the agent system. For more information, see "Location of Command Files" on page 62.
- Commands must create and manage the connection to the managed device.
- Commands must send all event data via `stdout` or the pipe specified in the source detail configuration.
- Commands must send all log file messages via `stderr` or the pipe specified in the source detail configuration.
- Commands must receive `SIGTERM` and cleanly shut down the connection to the managed device without further interaction from the data collector.
- Commands must avoid introducing delays when forwarding events to the data collector.

  If there is a delay in forwarding the event stream, the data collector may receive only the first part of an event before it times out and resets, losing the event. You can specify the `HLPTimeout` source detail to increase the data collector time out value.

Location of Command Files

For project configuration, include all commands and supporting files in the `/etc/opt/OV/Telco/conf/`*`project_name`*`/cmds/` directory. The `cmds` directory is further divided according to function:

- The `hp-ux11/` subdirectory contains scripts that run on the HP-UX 11.X operating system. This functionality is generic to all agents on this operating system.
- The `solaris/` subdirectory contains scripts that run on the Solaris operating system. This functionality is generic to all agents on this operating system.

- The `agents/` subdirectory contains subdirectories for each agent that has one or more command data sources. These subdirectories contain functionality that is specific to the named agent.

The process of applying the agent configuration copies the contents and structure of the `cmds` directory to the `/var/opt/OV/bin/Telco/` directory on the agent as follows:

1. It copies the contents of the subdirectory for the operating system of the target agent.
2. It copies the contents of the subdirectory for the name of the target agent.

   While the data is being copied to the agent, the directory structures are merged. If there are any files with the same names in both structures, the files in the agent structure overwrite the more general files in the platform structure.

Configuring Equipment

Events are received in a steady stream of input. The telecom data collector uses record formats to determine how to divide this input stream into individual events that are forwarded to the OVO agent for formatting via message source templates. Data that does not fall between recognized begin/end pairs is ignored (or logged if auditing is enabled). It is most efficient to filter out unwanted events at the data collector level. Doing so minimizes the impact these events have on the network management resources.

The equipment configuration lists the record formats of events that the data source can receive for each equipment type.

To add an equipment definition to *`Agent`*`.xml`, execute:

`ovcfgagent -addEquip <`*`name`*`> <`*`status`*`> <`*`record format`*`> `*`xml_file`*

For each equipment type, specify the following values:

- Name

  Use the name from the equipment list in the source configuration. This name must be unique.

  The data collector includes this name in the message that it passes to the OVO agent. For more information, see “Template Fields for Telecom Devices” on page 89.

- Status

  Specify whether the data collector will parse events from this equipment type. Possible values are `yes` and `no`.

- Record format list

  List the names of record formats for this equipment type. The same record format may apply to multiple equipment types. You must also define these record formats. For more information, see "Configuring Record Formats".

Configuring Record Formats

Record formats are positive filters. The more specific the begin/end pair is, the fewer event types pass through the record format. To prevent the telecom data collector from forwarding a certain event type, configure the record formats such that the event you want to ignore does not match any of the record formats. This process may require the configuration of several record formats for the set of events emitted by one type of equipment.

Record formats use regular expressions to specify the text expected at the beginning and ending of each event type. To specify a literal character that has special meaning in a regular expression, precede that character with a single backslash (for example: \$). Record formats can be localized. For more information, see "I18N Model" on page 36.

To add a record format definition to *`Agent`*`.xml`, execute:

`ovcfgagent -addRecordFormat <`*`name`*`> <`*`begin`*`> <`*`end`*`> `*`xml_file`*

For each record format, specify the following values:

- Name

  Use the name from the record format list in the equipment configuration. This name must be unique.

- Begin filter (optional)

  Specify the regular expression that marks the beginning of one or more events. This regular expression may not be a subset of any other begin filters.

- End filter (optional)

  Specify the regular expression that marks the ending of one or more events with the given begin filter. This regular expression may not be a subset of any other end filters.

NOTE The begin and end filters are both optional, but each record format must have at least one begin or end filter. One data source *and* events with an end filter but no begin filter. For more information, see the `ovsadc` man page.

Example Data Collector Configuration

This section includes several tables excerpted from an example configuration of a telecom data collector.

Table 3-1 shows an example definition of a telecom subagent data collector.

Table 3-1 Example Data Collector Definition

Data Collector Name	Sources	DC Details
`DC-DEMO`	`Source1 Source2 Source12 Source15 SourceBSC`	`DC-DEMO.details`

Table 3-2 shows an example definition for the data collector detail listed in Table 3-1.

Table 3-2 Example Data Collector Detail Definition

Name	Value
`CmdMode`	`enabled`

Table 3-3 shows example definitions for the sources listed in Table 3-1.

Table 3-3 Example Sources Definition

Name	SourceType	SourceDetail	Status (Enabled)	Equipment
Source1	TCP	Source1Detail	yes	NokiaBTS
Source2	FIFO	Source2Detail	yes	NokiaBTS
Source12	TCP	Source12Detail	yes	NokiaMSC EricAXE
Source15	TCP	Source15Detail	yes	EricAXE MotorolaBTS
SourceBSC	FIFO	SourceBSCDetail	yes	NMSC MotorolaBTS

Table 3-4 shows example definitions for the source details of Source1 in Table 3-3.

Table 3-4 Example of Source1Detail Definition

Name	Value
TCPMode	Server
TCPPort	9997

Table 3-5 shows example definitions for equipment connected to a data source.

Table 3-5 Example Equipment List Definition

Name	Status (Enabled)	Record Formats
NokiaBTS	yes	NMSC281
NokiaMSC	yes	NokBin GeneralBE
EricAXE	yes	AXEBin GeneralBE
MotorolaBTS	yes	GeneralBE

Table 3-5 Example Equipment List Definition (Continued)

Name	Status (Enabled)	Record Formats
NMSC	yes	NMSC281

Table 3-6 shows example definitions of three record formats from Table 3-5.

Table 3-6 Example Record Format Definitions

Record Format Name	Begin	End
NokBin	MSCBIN	MSCEND\012
AXEBin	AXEBIN	AXEEND\012
NMSC281	#S#281...MSC	#E#\012

Removing Data Collectors

When you delete a data collector definition, also delete supplemental definitions associated with that data collector if they are not being used by other data collectors.

To remove a data collector definition from the *Agent*.xml file:

1. Execute:

   ovcfgagent -remDC <*DC name*> *xml_file*

2. Optionally execute:

   ovcfgagent -remSource <*name*> *xml_file*

3. Optionally execute:

   ovcfgagent -remSourceDetail <*source detail name*> *xml_file*

4. Optionally execute:

   ovcfgagent -remEquip <*name*> *xml_file*

5. Execute:

   ovcfgagent -remRecordFormat <*name*> *xml_file*

6. Deploy the agent configuration.

   For more information, see “Deploying Agent and Data Collection Data” on page 72.

7. Apply the agent configuration.

   For more information, see “Applying Agent and Data Collection Data” on page 73.

Configuring Lookup Tables

Lookup tables are optional. Use lookup tables to reduce the number of message source template conditions that are necessary to process all possible events. You can also use lookup tables to improve the readability of messages.

Create lookup tables when you configure the agent as described here. Use lookup tables in the output fields of a message source template condition. For more information, see "Using Lookup Tables to Format an Output Message String" on page 94.

Lookup tables may have as many columns and rows as you need and may contain any valid string. The first row of the lookup table is the heading row. It contains the column names by which you can search the table. Lookup tables can be localized. For more information, see "I18N Model" on page 36.

To add a new lookup table to *Agent*.xml, execute:

```
ovcfgagent -createLookupTable <name> <column name> <column name> ...
```

To add a new entry to a lookup table in *Agent*.xml, execute:

```
ovcfgagent -addLookupTable <name> <column value> <column value> ...
```

For each lookup table, specify the following information:

- Table name

  This name must be unique.

- Two or more column names

  These are the headings by which you can search.

- Multiple rows

  Specify one value for each column name. Complete each row before starting the next row.

Table 3-7 shows an example of a lookup table for converting event type strings from incoming events to one of the five predefined X.733 event types used by the topology server.

Table 3-7 Example Lookup Table

Input	Output
"COMMUNICATION"	communicationsAlarm
"QUALITY"	qualityofServiceAlarm
"PROCESSING"	processingErrorAlarm
"EQUIPMENT"	equipmentAlarm
"ENVIRONMENTAL"	environmentalAlarm

Removing Lookup Tables

To remove a lookup table definition from the *Agent*.xml file:

1. Execute:

   ovcfgagent -remLookupTable <*name*>

2. Deploy the agent configuration.

   For more information, see “Deploying Agent and Data Collection Data” on page 72.

3. Apply the agent configuration.

   For more information, see “Applying Agent and Data Collection Data” on page 73.

Deploying Agent and Data Collection Data

After entering the agent configuration data in *Agent*.xml, validate and deploy the XML file. The validation process checks to see if the content you entered is consistent and valid against Agent.dtd. The deploy operation generates target configuration files and stores them in generated/agents/ under the project directory.

Execute: ovcfgdeploy -agent <*agent_name*> <*project*>

If the *Agent*.xml file is valid, the command generates the *agent*.xml configuration file.

Applying Agent and Data Collection Data

To pull the agent configuration files stored on the OVO server to the OVO agent computer for processing, use the `ovagt.apply` command.

On the OVO server, execute:

ovagt.apply <*project*> `-n` <*node1* [*node2 ...*]>

This utility takes the configuration file generated by deploying *Agent*`.xml` and places it in the location indicated in Table 3-8. This command also restarts the subagent processes.

Table 3-8 Data Collector Configuration Files

File Name	Content	Destination
agent`.xml`	Data collectors, sources, source details, equipment, record formats, and lookup tables.	`/var/opt/OV/conf/Telco/OVSAAgentCfg.xml` on the agent node

4 Modifying OVO Message Source Templates

This chapter explains how to build, modify, assign, and install message source templates, which are used to configure an OVO agent. It also describes how to configure OVO message source templates for telecom messages.

Introduction to OVO Templates

With the use of record formats in the agent configuration file, a significant number of unwanted alarms have been removed from the system. The data collector forwards the remaining alarms to the OVO agent to be processed by templates. Templates contain user-defined conditions that further filter unwanted alarms from alarms important to the customer. These conditions enable alarms to be parsed so that alarms matching conditions defined in the templates can be processed further or immediately forwarded to a log file.

For messages to be processed properly by templates and displayed in the message browser:

- The node defined in the template must exist in the Node Bank.
- The message group defined in the template must exist in the Message Group Bank.

Templates also reformat alarms according to user-defined rules. Reformatting changes the way messages are viewed in the message browser.

Lookup tables can reduce the number of conditions needed for a template by retrieving values from tables defined in an agent configuration file. Lookup tables are optional and could be used to translate parameters such as error or location codes into more readable strings.

The telecom diverter processes the lookup tables after the templates have been processed. The message stream interface must be enabled for the diverter to function properly. After the diverter processes the lookup tables, the messages are sent back to the agent for further processing.

Message Source Templates

OVO agents are configured via message source templates, which monitor the status of and collect information from telecommunication devices. An agent can format and forward a message only if it is described in a message source template. Message source templates are configured using the OVO administrator GUI.

Message source templates work by identifying strings within messages in message streams. When messages match the conditions defined in the templates, they are processed according to the rules defined in the template and forwarded to the OVO server.

Configuring message source templates for all of your managed devices can be time-consuming. For this reason, HP and third-party partners offer HP OpenView Smart Plug-ins, which contain preconfigured templates for many supported devices.

For more information about OVO templates, see the *HP OpenView VantagePoint Operations for UNIX Concepts Guide*.

Template Administrators

Template administrators are typically responsible for creating and modifying templates. Use the `Add User` window to assign the template administration responsibility to specific network administrators.

Template administrators use the `Message Source Templates` window to build, modify, copy, and delete templates as well as to organize template groups. To prevent multiple template administrators from modifying the same template, each template is locked after it is opened in the `Message Source Templates` window. The lock is released when the template administrator closes the last configuration window.

Template Types

Message source templates can be one of two types:

- Basic templates perform the minimal amount of processing needed to deliver an incoming message to the OVO operator GUI for further action by an operator. Such templates are often referred to as wildcard templates, meaning they use very few conditions to match large numbers of incoming messages. However, basic templates can vary in complexity and sophistication. They can contain any number of conditions and complex pattern matching schemes.

  Basic templates can include all standard OVO template functionality, such as automatic actions, operator-initiated actions, and instruction text. They can also include OV Telecom Extensions lookup tables.

- Topo-smart templates are a superset of basic templates. Typically, topo-smart templates are utilized to deliver messages to the topology server for further processing before forwarding to the OVO server and OVO operator GUI.

  Topo-smart templates include topology information that enables incoming messages to be translated into an X.733 message format needed by the topology server computer.

NOTE This book only describes the use of basic templates. For information on using topo-smart templates in a standard solution, see the *HP OpenView Topology Server Configuration Guide*.

Template Groups

Template groups are collections of templates or collections of other template groups that you create to increase the performance of configuration and management tasks.

If several devices from a particular vendor share a common alarm format, a single template can be constructed to manage these devices. For example, small, medium, and large configurations of a particular Ericsson switch might emit identical alarms.

Managing Message Source Templates

This section describes the process of and the fields for building and modifying message source templates. For information on configuring templates for events from telecom equipment, see “Modifying Basic Templates” on page 90.

Message source templates integrate messages from different message sources into the system. By configuring templates you can determine whether a message is forwarded to the message browser and which actions are to be performed. You can also determine which attributes are displayed with the message.

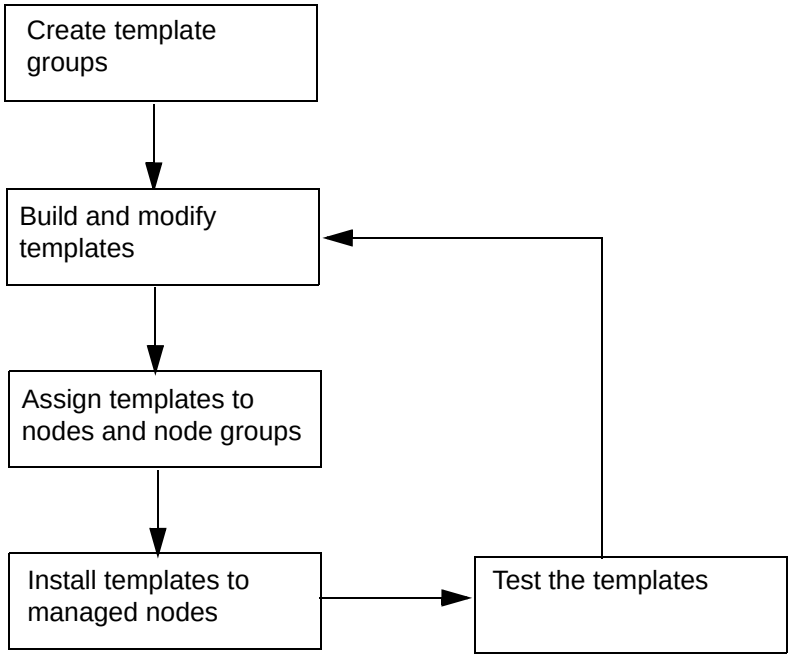
Configure message source templates on the OVO management server. The templates specify the messages and values that you want to collect or monitor. Templates also specify the routine actions that you want to schedule, the filters (conditions) that integrate or suppress messages, and the logging options that you want to use after interception.

Message source templates consist of the following elements:

- Type of message source from which you want to collect messages, such as a log file, a trap, an OVO message interface, or an action. Most templates associated with telecommunications events are message source templates.
- Message conditions and suppress conditions that match a set of attributes and define responses to received messages.
- Options, such as default message logging.

Figure 4-1 illustrates the order of tasks involved in configuring message source templates.

Figure 4-1 Configuring Message Source Templates



Work with templates and template groups in the `Message Source Templates` window. To open the `Message Source Templates` window, launch the OVO administrator GUI. Click `Window:Message Source Templates`.

Use the `Message Source Templates` window to build, modify, copy, and delete templates. The `Message Source Templates` window consists of a logical input and output section. Enter values in fields of the input section to match messages, and enter values in fields of the output section to determine how the message gets filtered and displayed in the message browser.

Work with template groups in the `Templates Groups` list box of the `Message Source Templates` window. This list box shows the hierarchy of template groups and lets you expand and collapse each group.

After templates and template groups are configured, you need to determine which node or node groups should receive the new templates. You can assign templates to nodes or node groups, or template groups to nodes or node groups where the message interception should be performed, and then distribute the new configuration.

The process of assigning a template group to a node is the same process as assigning a template to a node. All nodes within a node group automatically inherit the templates and template groups assigned to the node group. This simplifies assigning templates to new nodes.

After you have defined a new message source template and assigned it to the managed nodes, distribute it to the managed nodes.

NOTE If you delete a template or remove a template assignment, you must distribute the new configuration to the affected managed nodes.

Adding a Message Source Template

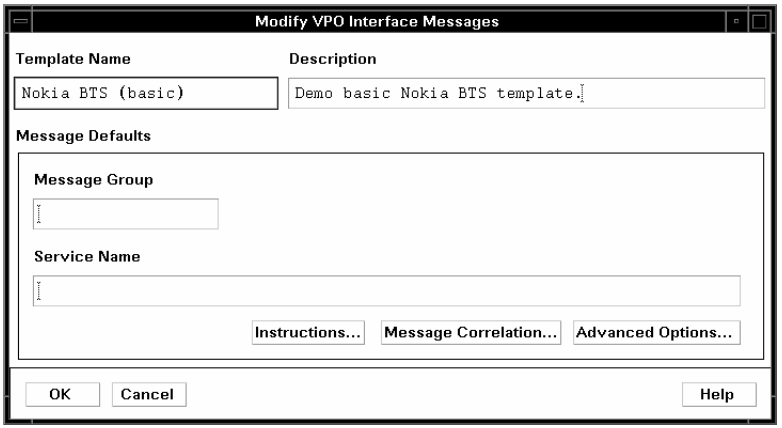
To add a new message source template, perform the following procedure:

1. In the OVO administrator GUI, open the Message Source Templates window, click [`Add Logfile`] and select [`Add Message`].

   The `Add Interface Messages` window shown in Figure 4-2 appears.

2. In the `Add Interface Messages` window, enter the name and description of the new template.

Figure 4-2 Example New Message Source Template



Adding Template Conditions

To add a condition to a message source template, perform the following procedure:

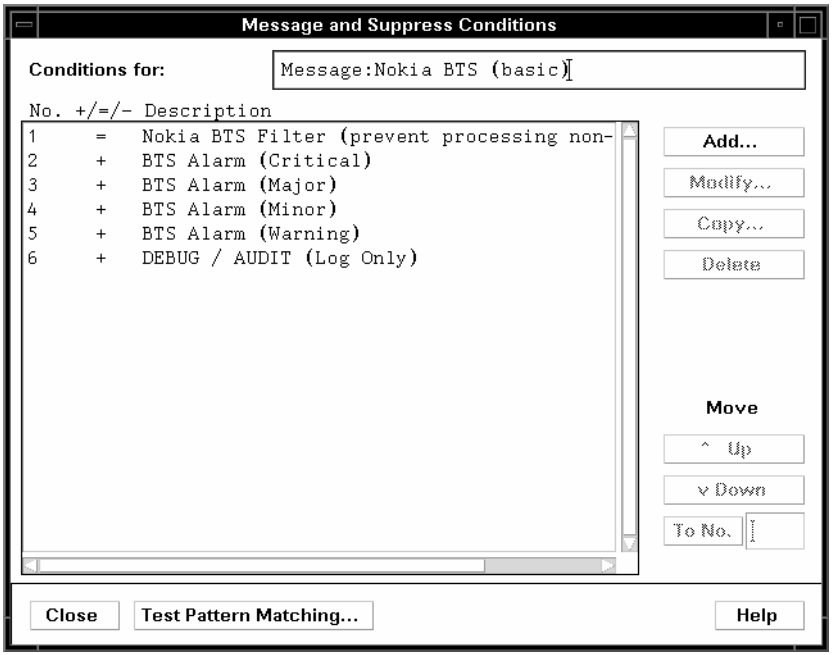
1. In the OVO administrator GUI, open the `Message Source Templates` window, select the template and click [`Conditions`].

   The `Message and Suppress Conditions` window shown in Figure 4-3 appears.

2. In the `Message and Suppress Conditions` window, click [`Add`] to add a new condition definition or [`Modify`] to update an existing condition definition.

   The `Condition` window shown in Figure 4-4 on page 85 appears. For information about this window, see “Modifying a Message Source Template” on page 85.

Figure 4-3 Example Template Conditions



The conditions in this window are processed in the order listed. Thus, the first matching condition determines template output behavior. For example, if your first condition says to suppress all messages, the other conditions listed are not processed and no messages appear in the message browser.

Modifying a Message Source Template

Figure 4-4 shows a condition window for modifying a condition definition. The window is divided into three logical parts: input fields, matching conditions, and output fields.

Figure 4-4 Modifying Template Conditions

Message Source Template Condition Input Fields

The top portion of the `Condition` window contains the input section. Messages are matched according to values stored in the fields listed in Table 4-1. This table describes the key message fields in message source templates and indicates their size limitations.

Table 4-1 Message Source Template Condition Input Fields

Field	Description	Size
Node	Coarse-grained identifier for the source of a message. For example, `software.hp.com`.	254
Application	Medium-grained identifier for a message source. For telecom-specific messages, the value is set to `Data Collector Name:Source Name` by the data collector.	32
Message Group	Group of alarms to which a message belongs. For example, `Telco-Nokia`.	32
Object	Fine-grained message source identifier. For example, `OpenView SID`. For telecom-specific devices, this value is set to the equipment type that matches the record format the data collector used to identify the event.	32
Message Text	Content and/or description of a message/ alarm. Use OVO message expression syntax to define the Message Text. Right-click in the field to view a short list of acceptable expressions. For example, if you want to match messages on the string `"Switch1"`, define the Message Text field to be `<*>Switch1<*>`.	512

Message Source Template Condition Matching Conditions

The matching conditions section of the `Condition` window describes how the conditions are to be treated. The options are listed in Table 4-2.

Table 4-2 Message Source Template Condition Matching Conditions

Option	Description
`Suppress Matched Conditions`	Suppresses all messages matching condition fields in the input section. Messages are not forwarded to the OVO server. If logging is enabled in the template advanced options, messages are stored in a log file.
`Suppress Unmatched Conditions`	Suppresses all messages not matching condition fields in the input section. Messages are not forwarded to the OVO server. If logging is enabled in the template advanced options, messages are stored in a log file.
`Message on Matched Condition`	Forwards all messages matching condition fields in the input section to the message browser.

Message Source Template Condition Output Fields

The lower portion of the `Condition` window contains the output section. The fields in this section correspond to columns in the message browser. Use these fields to reformat the original message into a more readable format. When any of these fields is unspecified, it is populated with the value from the original message. Table 4-3 describes the key output fields in message source templates and indicates their size limitations.

Table 4-3 Message Source Template Condition Output Fields

Field	Description	Size
Node	Coarse-grained identifier for the source of a message. For example, `software.hp.com`. The Node in this field must exist in the Node Bank before a message gets forwarded to the message browser.	254
Application	Medium-grained identifier for a message source. For example, `Oracle`.	32

Table 4-3 Message Source Template Condition Output Fields (Continued)

Field	Description	Size
Message Group	Group of alarms to which a message belongs. For example, `Telco-Nokia`. The Message Group in this field must exist in the Message Group Bank before a message gets forwarded to the message browser.	32
Object	Fine-grained message source identifier. For example, `OpenView SID`.	32
Message Text	Content and/or description of a message/ alarm. Use OVO expression syntax to define the Message Text.	2048
Service Name	Identifier used to associate a message/alarm with a service.	254
Message Type	Identifier of a subgroup within a message group. For telecom-specific alarms to be forwarded to the telecom diverter, the value of this field must being with `Telco_`. The recommended value of this field is `Telco_<`*`eventtype`*`>`, where `<`*`eventtype`*`>`is one of the following values: `communicationsAlarm`, `qualityofServiceAlarm`, `processingErrorAlarm`, `equipmentAlarm`, and `environmentalAlarm`.	32

NOTE

Node, Application, Message Group, and Object are used by administrators to create filtered views in the OVO operator GUI. Use these fields consistently to enable operators to effectively monitor equipment for which they are responsible.

Message Group and Node objects control which messages are assigned to which operators. See the *HP OpenView VantagePoint Operations for UNIX Concepts Guide* for more details.

Template Fields for Telecom Devices

The OVO agent identifies messages based on conditions defined in the templates. Received messages are evaluated based on the content in the Node, Application, Message Group, Object, and Message Text fields. Some fields, such as the Object and Message Text fields, direct the agent to perform pattern matching or string comparison.

Table 4-4 lists the OVO message fields that you can use in message source template conditions to filter messages received by telecommunications data collectors. The example values are for the data collector configuration shown in “Example Data Collector Configuration” on page 65.

Table 4-4 Template Fields Populated by Data Collectors

Field	Value	Example
Application	Data collector name: Source name	`DC-DEMO:Source 1`
Object	Equipment type	`NokiaBTS`
Message Text	Raw event text, including beginning and ending markers that define the event boundaries. These markers match one of the defined record formats.	`#S#281...MSC message text here #E#`

Beginning and ending markers are included in the Message Text field to allow templates to distinguish between alarms from a single device where the alarm text without beginning and ending markers might be identical. With these markers, alarms such as these are clearly different: `POWERALM 24 42 END` and `COMMALM 24 42 END`.

Modifying Basic Templates

Basic templates act as standard OVO message source templates. Table 4-5 lists recommended values of OVO message output fields for basic templates.

Table 4-5 Output Message Fields of Basic Templates

Field	Recommended Value
Node	*<agent hostname>*
Application	Technology tag, such as GSM, SDH
Object	*<Device ID:Component ID>*
Severity	Set as indicated by alarm
Message Text	Formatted message to appear in OVO message browser
Message Group	Telco_*<eventtype>*
Service Name	No recommendation
Message Type	X.733 event type
Message Key	No recommendation

Message Text is the only required message field. If a message field is left blank, information from the raw alarm may be inserted in the message field.

The Message Text field is the key field where received messages are evaluated and transformed into more readable messages. To make messages more readable or to reduce the number of conditions for a message template, lookup tables can be referenced in the Message Text field. For more information, see “Using Lookup Tables to Format an Output Message String” on page 94.

Enabling Messages for OV Telecom Extensions Processing

After the OVO agent applies the message template, most telecom messages should be passed to the telecom diverter for further processing. The diverter is part of the telecom subagent functionality. Lookup tables are used by the diverter process. After completing message processing, the diverter passes the message back to the OVO agent.

To enable message passing to the diverter, first enable the message stream interface and then enable message diversion to the message stream interface.

Enabling the Message Stream Interface

Enable the message stream interface once per telecom subagent computer.

1. In the OVO administrator GUI, open the `Node Bank` window, select the managed node on which to enable the message stream interface.
2. Right-click on the managed node to display its menu.
3. Click `Modify`.

   The `Modify Node` window appears.
4. Click `[Advanced Options]`.

   The `Node Advanced Options` window appears.
5. Under the `Message Stream Interface` option area, select `Enable Output`.
6. Click `[Close]`.
7. Click `[OK]` to enable the message stream interface.

   In a short while, you should see a message in the message browser indicating that the agent system has been updated.

Enabling Message Diversion

Enable message diversion to the message stream interface for *each* template condition. To enable a template condition so that the message matching this condition is passed to the diverter, use the following procedure:

1. In the message `Condition` window, set the `Message Type` name to any name that begins with the string `Telco_`.
2. Click `[Advanced Options]`.
3. In the `Message Stream Interface` section, select `Agent MSI` and `Divert Messages`.

Configuring the Message Stream Interface for Multiple Outputs

The entries in the `/var/opt/OV/conf/OpC/msiconf` file indicate the order in which processes receive events from the agent message stream interface. In this file, higher numbers indicate lower run priority. It is recommended that these numbers always be multiples of 10.

The message stream interface passes each event to the registered processes in the following manner:

- For registered processes that have different priority numbers, events are processed serially. The message stream interface passes each event to the first process and waits for that process to return the event to the message stream interface before passing it to the next process in the list.
- For registered processes that have the same priority numbers, events are processed in parallel. The message stream interface passes each event to these processes simultaneously.

IMPORTANT

If two or more processes receive events in parallel and return any events to the message stream interface, the agent will have multiple copies of the same event that may not be identical. Therefore, ensure that processes that return events to the message stream interface have different priority numbers in the `msiconf` file.

By default, there is no entry for ECS in the `msiconf` file, which means that the ECS runs at a priority of `0` (before the diverter). By default, the run priority for the `ovtopodiverter` is `10`.

If additional installed applications listen on the agent message stream interface, the diverter value can be changed. It is recommended that it always be later than ECS, and that it be a multiple of 10.

To change the order in which processes receive events from the agent message stream interface:

1. On the OVO agent computer, edit the `/var/opt/OV/conf/OpC/msiconf` file.
2. If this file contains an entry for ECS, ensure that its run priority is lower than that of the `ovtopodiverter` process.

   The format of entries in the `msiconf` file is:

   ```
   ovtopodiverter     10     # HP Topology Diverter
   ```

   The # sign indicates a comment.

Using Lookup Tables to Format an Output Message String

Lookup table processing retrieves values from lookup tables defined in the telecom subagent *Agent*.xml file. This processing occurs in the telecom diverter after the message source template has been processed. The use of lookup tables is optional. Use lookup tables to reduce the number of message source template conditions that are necessary to process all possible events. You can also use lookup tables to improve the readability of messages.

Because lookup table are used by the telecom diverter, they are not limited to messages from telecom data collectors. Any message that is received by any data collector on the OVO agent on which there is a telecom subagent can use a lookup table.

OV Telecom Extensions has defined the $SELECT() operator to extend OVO message processing functionality.

Lookup tables are supported in the output message text field and custom message attributes. They use syntax similar (in nature) to SQL. The syntax of the $SELECT() statement is:

$SELECT(*table_column_wanted*,*table_name*,*table_column_known*="*input_variable*"
[,*table_column_known*="*input_variable*"])

The quotation marks in the $SELECT() statement syntax are optional. For more information about the syntax of the $SELECT() statement, see the ovtopodiverter man page.

Observe the following limitations:

- The *table_column_wanted*, *table_name*, and *table_column_known* strings may contain any character *except*:) (, " =
- The *input_variable* string may contain any character *except*:) (,

For example, consider the example shown in Table 4-6.

Table 4-6 Example `employee` Lookup Table

employee_id	site	first_name	last_name
1234	Cupertino	Bob	Smith
1235	Fort Collins	Bob	Smith

To look up an employee ID number based on employee name and location, use the statement:

```
$SELECT(employee_id,employee,first_name="<fname>",last_name="<lname>",
site="<location>")
```

where:

- `employee_id` is the name of the column defined in the *`Agent`*`.xml` file.
- `employee` is the name of the table defined in the *`Agent`*`.xml` file.
- <*`fname`*>, <*`lname`*> and <*`location`*> are variables extracted from the event text in the input message text field of the message source condition.

To use lookup tables:

1. Define the table name and values in the *`Agent`*`.xml` file. For more information, see “Configuring Lookup Tables” on page 69.
2. Modify the agent configuration to enable the agent message stream interface. For more information, see “Enable Message Stream Interface” on page 146.
3. Enable the templates for processing. For more information, see “Enabling Messages for OV Telecom Extensions Processing” on page 91.
4. Edit the output message text to use `$SELECT()`.
5. Deploy the agent configuration. For more information, see “Deploying Agent and Data Collection Data” on page 72.
6. Deploy the templates to the agent. For more information, see “Enabling Templates” on page 96.

Enabling Templates

Templates must be assigned to and installed onto the managed node where the OVISACN data collector is deployed. Assigning templates designates that the templates should go to the target agent computer, but you must install them for the templates to be active. Even when the agent resides on the OVO management server computer, you must install templates on the target managed node.

Use the OVO administrator GUI to assign and install templates. After templates are assigned, they remain assigned. Any modifications to the templates or template assignments, however, require reinstallation on the agent systems. Modifications to templates include adding, modifying, and removing conditions. Modifications to template assignments include adding and removing templates.

Assigning Templates

1. In the OVO administrator GUI, open the `Node Bank window`.
2. Select the managed node for which you want to change the assigned templates.
3. Click `Actions:Agents->Assign Templates`.

   The `Define Configuration` window appears.
4. Click [`Add`].

   The `Add Configuration` window appears.
5. Make sure that the correct managed nodes are selected in the `Node/Node Group` list box.
6. Click [`Open Template Window`].
7. In the `Message Source Template` window, select the template to assign.
8. From the `Add Configuration` window, click [`Get Template Selections`].
9. Click [`OK`] in the `Add Configuration` window.
10. Click [`OK`] in the `Define Configuration` window.

Installing Templates

1. In the OVO administrator GUI, open the `Node Bank window`.
2. Select the managed node on which you want to change the installed templates.
3. Click `Actions:Agents ->Install/Update SW & Config`.

   The `Install/Update Software and Configuration` window appears.
4. Make sure that the correct managed nodes are listed in the `Target Nodes` window.
5. Select `Templates`.
6. Click `[OK]`.

   After a short while, you should see a message in the message browser indicating that the agent system has been updated.

Checking Which Templates Are Installed

You can view all templates installed on an agent system by executing (as the `root` user):

`/opt/OV/bin/OpC/opctemplate`

This command lists all templates with type, name, and status (enabled or disabled). This command is useful for checking whether a template you have assigned to an agent node has been successfully installed on that agent system. Be aware that this command does not indicate which version of the template has been installed. If you have made modifications to any assigned templates, you must reinstall the templates on the managed nodes.

Avoiding Duplicate Messages

While template conditions *are* ordered, templates *are not* ordered. If an incoming message matches one or more conditions of multiple templates, multiple messages appear in the OVO message browser. For example, if a wildcard template is assigned and installed on a system, then every message entering the OVO agent is forwarded to the message browser. If additional templates are assigned and installed on an agent computer, those messages matching the conditions of the templates are also forwarded to the message browser. Thus, duplicate messages appear in the message browser, formatted according to the rules in each matching template.

To avoid duplicate messages in the OVO message browser, disable the templates causing the duplicate messages or create message conditions that suppress messages that are not applicable to these templates.

5 Troubleshooting

This chapter details the utilities and programs to be run and files to be checked should there be any problem while running OVISACN. These utilities can be run by all valid system users unless otherwise specified.

The utilities discussed in this chapter pertain to:

- Checking the log files for errors.
- Changing the tracing level for some telecom management processes.
- Configuring OVISACN self-management.
- Tracking issues when messages are not forwarded to the OVO message browser.

While all HP OpenView products go through exhaustive testing prior to release, occasionally, some defects or problems are discovered after the product is released. Review the product release notes and the available HP OpenView software patches that solve some of the known product defects.

Prerequisites

This chapter assumes that OVISACN is installed according to the instructions outlined in *HP OpenView Telecom Extensions Installation Guide*, with the OVO administrator GUI and OVO operator GUI available to you.

Viewing Log Files

Table 5-1 provides the location and description of the log files associated with HP OpenView Integrated Service Assurance for Communication Networks.

Table 5-1 Useful Log Files

Log File	Description
`/var/adm/syslog/syslog.log`	System log file.
`/var/opt/OV/log/Telco/ovtopodiverter.log`	Diverter log file.
`/var/opt/OV/log/Telco/ovtopodiverter.trc`	Diverter trace file.
`/var/opt/OV/log/Telco/ovsadc.log`	Data collector log file.
`/var/opt/OV/log/Telco/ovsadc.trc`	Data collector trace file.
`/var/opt/OV/share/log/Telco/configurator.log`	Configuration log file for diagnostics on the commands `ovcfgdeploy` and `ovcfgvalidate`.
`/var/opt/OV/log/OpC/opcmsglg`	OVO message log.

Enabling and Disabling Tracing

The `-t` option toggles the tracing level for the `ovsadc` and `ovtopodiverter` processes.

Setting the Tracing Level on the Data Collector

Use the `-t` option to toggle the tracing level for the `ovsadc` process on and off.

The tracing level for the `ovsadc` process can be changed while this process is running.

For more information, see the `ovsadc` man page.

Setting the Tracing Level on the Diverter

Use the `-t` option to set the tracing level for the `ovtopodiverter` process.

For `ovtopodiverter`, the `-t` option is a circular toggle. The first use puts the process in `DEBUG` mode. The second use puts the process in `TRACE` mode. The third use turns tracing `OFF`.

The tracing level for the `ovtopodiverter` process can be changed while this process is running.

For more information, see the `ovtopodiverter` man page.

Configuring OVISACN Self-Management

OV Telecom Extensions provides OVO log file templates for the telecom subagent process. When these templates are in place on the appropriate OVO agent computer, error and warning messages from the telecom subagent process are reported in the OVO message browser.

The OV Telecom Extensions setup process creates the following default configuration:

- The new message group `Telco Admin` contains the self-management messages.
- The new user profile `Telco Admin` has responsibility for the `Telco Admin` message group.
- The new user `telco_admin` has only the `Telco Admin` user profile.
- The `opc_adm` user has the `Telco Admin` user profile in addition to other user profiles.
- The `TelcoAgentSelfMgmt` templates are loaded under the `Telco` template group.

Telecom Subagent Self-Management

The telecom subagent self-management template group contains the following templates:

- `DataCollectorLogs`—Templates for the data collector (`ovsadc`)
- `DiverterLogs`—Templates for the diverter (`ovtopodiverter`)
- `InjectorLogs`—Templates for the injector (`ovtopoinjector`)

Each template provides conditions for the following types of messages:

- Critical
- Error
- Warning
- Information (suppressed by default)

Enabling Telecom Subagent Self-Management

Telecom subagent self-management is enabled by default.

To re-enable telecom subagent self-management on all telecom subagent nodes:

1. Set the value of `ENABLE_AGENT_SELF_MGMT` to `yes`.

   The `ENABLE_AGENT_SELF_MGMT` variable is in the `/etc/opt/OV/share/conf/Telco/ovtelco.setup` answer file.

2. Run the `ovagt.apply` command for each telecom subagent node.

   The `ovagt.apply` command assigns and distributes the telecom subagent self-management templates to the nodes.

NOTE If you have *not* changed the agent configuration, you do *not* need to deploy the agent configuration before doing this step.

To enable self-management for an individual OVO agent node, use the `-t` option to the `ovagt.apply` command. This command line option overrides the value of `ENABLE_AGENT_SELF_MGMT`.

Disabling Telecom Subagent Self-Management

To disable telecom subagent self-management on all telecom subagent nodes:

1. Set the value of `ENABLE_AGENT_SELF_MGMT` to `no`.

   The `ENABLE_AGENT_SELF_MGMT` variable is in the `/etc/opt/OV/share/conf/Telco/ovtelco.setup` answer file.

2. Run the `ovagt.apply` command for each telecom subagent node.

   The `ovagt.apply` command removes the telecom subagent self-management templates from the nodes.

NOTE If you have *not* changed the agent configuration, you do *not* need to deploy the agent configuration before doing this step.

To disable telecom subagent self-management for an individual OVO agent node, use the `-nt` option to the `ovagt.apply` command. This command line option overrides the value of `ENABLE_AGENT_SELF_MGMT`.

Troubleshooting Message Delivery

This section captures some helpful troubleshooting information related to issues with the delivery of messages within OVISACN. It divides the process of message delivery into the tasks done by the component software pieces and then lists the most likely causes of problems for each piece. The focus here is specifically on message delivery issues that are likely caused by configuration issues.

Initial Troubleshooting

If the messages you are attempting to deliver do not appear in the OVO message browser, see “Troubleshooting the Data Collector” on page 109. If messages appear in the OVO message browser, but lookup table processing (`$SELECT()`) is not parsing messages as expected, see “Troubleshooting the Telecom Diverter” on page 107.

OVO User Responsibilities

By default, the user `opc_adm` and the user `telco_op` get the needed permissions from the `Telco_Op` operation profile. Verify that the assigned profile’s permissions are correct. The areas to verify include:

- The message group identifies in the incoming message and the message source templates exists.
- The message group of interest is assigned to the user and the user profile.
- The node identified in the incoming message and the message source templates exists in a node group.

Troubleshooting the Telecom Diverter

The telecom diverter process (`ovtopodiverter`) is responsible for listening on the agent message stream interface (MSI) for telecom-related messages, processing lookup tables, and returning messages to the OVO agent.

Telecom Diverter Checklist

If you believe that the diverter should process messages, but messages are not being properly handled, do the following tasks:

- Verify that the telecom diverter is running by executing:

  `opcragt -status -id 11 <agent hostname>`

- Verify that the agent MSI is enabled on the agent computer. For instructions, see “Enabling the Message Stream Interface” on page 91.
- Verify that the agent MSI is enabled with `Divert Messages` set in the appropriate template conditions. For instructions, see page 92.
- Verify that the `Message Type` field is set to `Telco_<something>` in the appropriate templates.
- Assign and reinstall the templates.
- Verify that the appropriate agent configuration has been successfully deployed and applied.
- Check `/var/opt/OV/log/Telco/ovtopodiverter.log` for error messages.
- Check `/var/adm/syslog/syslog.log` for `ovtopoinjector` error messages.

Lookup Table Processing Checklist

If messages appear in the OVO message browser, but a lookup table `$SELECT()`call does not appear to be processed correctly, do the following tasks:

- Verify that the target column, table name, and index columns specified in the `$SELECT()` call match the values in the *`Agent`*`.xml` file.
- Verify that the agent configuration is deployed and applied successfully. For instructions, see “Deploying a Project” on page 42 and “Applying Configuration Files” on page 43.
- Verify that the agent MSI is enabled on the agent computer. For instructions, see “Enabling the Message Stream Interface” on page 91.

- Verify that the agent MSI is enabled with `Divert Messages` set in the appropriate template conditions. For instructions, see page 92.
- Verify that the Message Type field is set to `Telco_<something>` in the message source templates.
- Look at the processed message and verify that any substituted values are actual entries in your lookup table. Remember that quoted strings will attempt to match the quotes exactly. For example, if the message contains “something”, including the quotes, then for the lookup table to match the entry, “something”, including the quotes, is needed in the index column of the table.

Troubleshooting the Data Collector

The data collector is responsible for receiving a stream of text from a variety of sources and converting it into a message for the OVO agent to process. There are two broad areas that are required for a message to be processed correctly:

- The message must be identified in some configured input stream for the data collector.
- The message must be matched and processed correctly by an OVO message source template.

Data Collector Configuration Checklist

If the data collector is not receiving some or all expected events, do the following tasks:

- Did the agent configuration deploy and apply successfully from your configuration project?
- Check `/var/opt/OV/log/Telco/ovsadc.log` for the initial configuration state and error conditions. Is there any indication of problems? Does it match the intended configuration?
- Verify that the agent is running by executing: `opcagt –status`. If the agent message interceptor is not executing, it typically indicates that no templates are installed on the agent system.

- Enable auditing in your agent source details configuration, and verify that an input stream of data is being processed. If no data is being processed, double check your source details configuration against your target source. Remember to deploy and apply your configuration changes.
- If data is being received, but messages are not being correctly matched, confirm that you have correct begin/end pair definitions and that the pairs are correctly assigned to the appropriate equipment types in the agent configuration.
- If the data collector indicates a run-time error, the executing thread for that source may have stopped executing. Fix the problem; then stop and restart the data collector.

Data Collector OVO Configuration Checklist

For the messages the data collector sends to be processed correctly, a template matching that message must be assigned and installed on the agent computer. To troubleshoot the OVO template configuration, do the following tasks:

- Execute `/opt/OV/bin/OpC/opctemplate` and verify that the intended templates are installed on the system.
- If the template sets the message group, verify that the message group and the agent node are in the target users responsibilities, either directly or via an assigned profile.
- Verify that the agent is running by executing `opcagt –status`.
- Modify the template to enabled logging of matched and unmatched messages, and re-install the templates on the agent system. Then check the agent log file, `/var/opt/OV/log/OpC/opcmsglg`, to see if the incoming message is being processed correctly.
- If the message is suppressed, but this is not part of the template processing, check the node configuration and make sure message reception is enabled for this node.

Glossary

acknowledge An action that causes a message or other notification to be removed from the OVO message browser.

action An operation that can be carried out within an OpenView application. Actions are typically performed on managed objects and can be executed by users through menu items or toolbar buttons. Actions can also be configured to automatically occur in response to an event or notification.

admin group A system-defined user group. Users belonging to this group have supervisory rights over other users.

administrator A user who has privileges and responsibilities to configure and maintain a managed network.

agent A program or process running on a device or computer system that responds to management requests, performs management operations, or sends performance and event notifications.

alarm An announcement of an event that occurred on the network. Alarms are collected from a network element or system and forwarded to the management system for processing.

annotation Text entered by operators, administrators, or automatically after actions that describes actions and tasks that have occurred to solve a given problem.

application A packaged software that provides functionality to accomplish a set of related tasks.

attribute A characteristic or property of an object that can be described through a name-value pair.

authentication A process of determining whether someone or something is who or what it is declared to be. Authentication is commonly done through the use of logon passwords.

automatic action A preconfigured program instruction that is executed in response to an alarm without operator control or intervention.

circuit See correlation circuit.

CMIP (Common Management Information Protocol) A connection-oriented protocol that allows network elements, such as switches, routers, and management agents, to be manipulated via sophisticated messages.

CMISE The services defined for CMIP protocol are known as CMISE.

component A physical object contained within a network element. Components may or may not emit alarms.

configuration The process of customizing the software to a specific managed network.

control agent Also known as `opcctla` in OVO. An agent on each managed node that is responsible for starting and stopping all other agents and processing requests from the management server.

correlation circuit In ECS, a collection of interconnected primitive and compound nodes configured to perform a filtering or correlation activity. Each correlation node is configured appropriately to the correlation requirement. The configuration includes the specification of the event types and the allowed transit delays for those events. A correlation circuit can be loaded into the ECS correlation engine.

correlation engine The ECS component that reads an input event stream, decodes the input events, performs the event correlation, encodes the output events, and returns the output events to the event stream. The rules of event correlation are specified by one or more correlation circuits loaded into the correlation engine.

data collector A process that receives alarms emitted by network elements and forwards these alarms to the agent.

device A piece of equipment that generates alarms when any of its components fails.

discharge An action that causes a message or other notification associated with a problem or situation to be removed from the browser. Messages are usually discharged when the operator has resolved the situation that led to the message.

diverter A process on the telecom subagent that provides additional message translation capabilities, such as table lookup conversions and time base arithmetic.

ECS See Event Correlation Services (ECS).

ECS circuit See Correlation Circuit.

ECS Designer OpenView software product that is used to create and test correlation circuits. It works in two modes: build and simulate.

ECS Engine See Correlation Engine.

element management system (EMS)

Vendor- or device-specific components that provide device-specific interfaces for receiving alarms and monitoring device status.

event An unsolicited notification, such as an SNMP trap or WMI notification generated by an agent or process in a managed object or by a user action. Also known as an alarm.

event correlation The evaluation of multiple events or notifications that are related to a single incident or problem in order to produce a single message. Event correlation is used to reduce the number of messages that are presented to an operator in a message browser.

Event Correlation Services (ECS) The HP Open View Event Correlation Services product, which uses correlation circuits and the ECS engine to filter events.

explodable Describes map icons that result in the display of another map upon double-clicking.

host A server or workstation.

host name The name of the server in the network.

icon One or more on-screen images combined as a unit. Icons represent objects that can be monitored or manipulated by the user or actions that can be executed by the user.

inheritance An object-oriented concept that says when a class of objects is defined, any subclass that is defined can inherit the definitions of one or more general classes.

interceptor An agent process dedicated to collecting alarms from a particular source. The `logfile` encapsulator collects alarms from log files. The `opcmsg` interceptor collects alarms injected using the `opcmsg`(3) API.

IP (Internet Protocol) A network layer protocol used by TCP and UDP protocols. Its main function is to route datagrams among nodes in different networks.

log file Files that store received alarms emitted by managed network elements.

lookup table Information used by the telecom diverter to translate raw alarm fields into readable strings. Lookup tables are configured per telecom subagent.

managed node A computer system or device in a network that is both monitored for status and messages and is manipulated by means of actions in the management software.

management server Provides management services, processes, and a management user interface to clients. The OVO software and relational database reside on the management server.

map A graphical representation of objects in the management environment in a format that shows one or more topological relationships among the objects.

message A structured, readable piece of information that is generated as a result of an event, the evaluation of one or more events relative to specified conditions, or a change in application, system, network, or network element status.

message agent Also known as `opcmsga` in OVO. An agent on a managed node that receives messages from the message sources, and processes and forwards the messages to the management server.

message group A collection of messages that belong to the same task or have some logical relationship and can be treated as a unit.

name-value pair A combination of an attribute identifier and the value of that attribute for a specific object.

network element (NE) A piece of manageable telecommunications equipment that generates alarms when any of its components fails. For example, a network element can be a digital cross connect, an add-drop multiplexer, or a digital loop carrier.

NNM (Network Node Manager) An OpenView software product that discovers and manages IP and IPX networks.

NOC (Network Operations Center) A place from which a network is supervised, monitored, and maintained.

node A connection point for data transmissions.

node group A collection of nodes that can be treated as a unit.

Open Systems Interconnection (OSI) A systems management model that defines the rules for processing and transferring data over networks.

operator-initiated action A preconfigured program instruction that requires user interaction to initiate. Operator actions are often provided to users in relation to an event, message, or other notification.

OSF/Motif GUI A graphical user interface standard that conforms to Open Software Foundation's recommendations.

outstanding alarms Alarms, both unassigned and owned, that have not yet been dismissed.

OV Operations (OVO) An OpenView software product that provides a generic framework for system, applications, and network management. Formerly known as VantagePoint Operations (VPO) and ITO.

ovstart The program that starts up the OpenView processes. This program is (normally) run automatically on system startup and can only be run by the superuser.

ovstop The program that stops OpenView processes. This program is (normally) run automatically on system shutdown and can only be run by the superuser.

OVW HP OpenView Windows is a GUI for integrating network management and system management applications.

own The act of taking or assigning responsibility for resolving a problem or situation associated with a message or other notification.

PDU (Protocol Data Unit) The package used for SNMP requests and responses.

profile A collection of tasks, applications, capabilities, and responsibilities that can be assigned to a user.

project A collection of configuration files and preference definitions that defines a telecom network management solution. The information stored in a project is applied to the OVO server and agent computers.

radio buttons Radio buttons are typically used for setting states or modes. Depressed button state indicates that the parameter is selected.

raw alarms Alarm messages that are emitted from network elements in a managed network, and are not formatted or correlated.

server A computer system that provides a service to other computer systems (clients) on the network.

severity A property of an object indicating the status of the object. Severity is based on the impact of alarms or messages associated with the object.

SNMP (Simple Network Management Protocol) The ARPA network management protocol running above TCP/IP used to communicate network management information between a manager and an agent.

TCP (Transmission Control Protocol) A method or protocol used along with the internet protocol to send data in the form of message units between computers over the Internet.

telecom subagent A process that can be installed on an OVO agent computer to receive events from telecom network elements and element management systems.

template A predefined object with default settings that is used for creating new instances of objects.

UDP (User Datagram Protocol) A communications method or portal that offers a limited amount of service when messages are exchanged between computers in a network that uses the internet protocol. An alternative to TCP.

X.733 A standard alarm format understood by HP OpenView Integrated Service Assurance for Communication Networks. X.733 specifies a well-defined set of alarm fields and values.

XML (Extensible Markup Language) A programming language similar to HTML that provides a common way to describe any kind of data.

Index

V

X

Index