HP OpenView Service Assurance for Communication Networks

Configuration Guide

HP-UX, Solaris

Manufacturing Part Number: J5119-90002

October 2001

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

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Figures

Figures

Tables

Tables

Tables

Tables

1 Introduction

This manual is intended for a system integrator or network administrator in charge of configuration. It assumes that the administrator has user-level knowledge of the HP-UX operating system and is familiar with using GUI-based applications with mouse and menu-driven interface on UNIX workstations.

What This Manual Covers

This manual covers the configuration process for HP OpenView Service Assurance for Communication Networks. These steps must be followed after the OVSACN system has been installed and setup as described in *HP OpenView VantagePoint Operations for UNIX Installation Guide*.

This manual guides you through the following tasks:

- Modifying OVSACN configuration files.
- Configuring a telecom subagent.
- Deploying telecom templates.
- Interfacing network elements.
- Managing a network by applying system distribution rules.
- Synchronizing data between OV Operations and OV Topology Server.
- Adding operator access to the OV Topology Server topology GUI.
- Creating operator profiles in OV Topology Server.

Getting Additional Information

Both printable and online documentation are available. Table 1-1 lists printable documents available to you via a web browser from the OVO server after OV Telecom Extensions for OV Operations and the documentation files are installed. Documents can be found at:

http://*<OVO servername>*:8880/Telco/index.html

Table 1-1 **OVSACN Documentation**

Document Title	Description
HP OpenView Service Assurance for Commuinication Networks Concepts Guide	Provides a brief overview of the product architecture
HP OpenView Service Assurance for Communication Networks Installation Guide	Details the steps necesary to install and setup the product.
HP OpenView Service Assurance for Communication Networks Quick Start Guide	Provides an overview to the product features and benefits and highlights 10 useful tasks users can do with the product.
HP OpenView Service Assurance for Communication Networks Configuration Guide	Details the steps necessary to configure a cusotmer's managed network.
HP OpenView Service Assurance for Communication Networks Customization and Maintenance Guide	Details the steps necessary to customize operator views and maintain the OV Topology Server.
HP OpenView Service Assurance for Communication Networks Operator Online Help	Provides operators help on navigating between the OVO operator GUI and the topology GUI as well as customizing the topology GUI.

If The Online Documentation Is Not Installed

If the online documentation has not been installed on a particular system, you may find it installed on other systems, or you can install the documentation from the User Documentation CD-ROM that is shipped with OV Telecom Extensions for OV Operations. For more information on installing the online documentation, see *HP OpenView Service Assurance for Communication Networks Installation Guide*.

Reference Pages

Reference pages for most of the commands and files needed by network administrators to configure, customize, and maintain OV Telecom Extensions for OV Operations and OV Topology Server are provided in two formats: HTML and nroff.

- HTML reference pages for OV Telecom Extensions for OV Operations and OV Topology Server can be viewed by visiting:
  http://*<OVO servername>*:8880/Telco/man/index.html
- Nroff reference pages for OV Telecom Extensions for OV Operations and OV Topology Server can be viewed in the UNIX environment by typing:
  `man <commnad name>`

2 Understanding the Network Management Environment

This chapter explains the key terms and concepts associated with managed network environments. This is a great overview chapter for any network administrator who is new to the telecom management domain.

If you are familiar with the network management environment, skip to Chapter 3, “Understanding OV Telecom Extensions for OV Operations and OV Topology Server.”

Network Management and HP OpenView Products

The management of modern telecommunication networks is an inherently complex task, involving diverse and geographically dispersed hardware, software, and users. Such networks seethe with change: nodes come and go frequently, the capabilities of different nodes evolve, and resource consumption can vary dramatically.

Any of all of these can occur quickly. The people responsible for managing this environment need quick access to reliable network data, tools to analyze the data, and mechanisms to make changes in the network at any moment. The sheer volume of information flowing through a network compounds the complexity. The growth in voice and data networks has created a critical need to monitor and control network resources for utmost effectiveness and productivity.

OV Topology Server coupled with OV Telecom Extensions for OV Operations consist of a set of powerful, comprehensive, and flexible network management functions integrated to deliver problem solutions closely tailored to the needs of the Network Operations Centers (NOC) in the wireline and wireless telephony environment. OV Topology Server provides monitoring and control of the network element managers and problem management functions in multivendor, geographically-dispersed network environments. Its open, fully-distributed, and standards-based architecture provides a flexible framework for integrating and supporting multiple best-in-class operations support applications, such as performance management and equipment provisioning from HP partners and third-party vendors.

Telecommunications Management Network

OV Topology Server is based on Telecommunication Management Network (TMN) standards, which was developed by ITU-U to specify management of telecommunications networks. TMN standards allow TMN and non-TMN applications to be integrated easily.

The benefits of using TMN standards are to:

- Allow network management to be distributed.
- Make the network more robust.
- Integrate equipment from many suppliers in the same TMN network management structure.

The design of the *HP OpenView Telecom DM* platform derives from the management model created by ISO – the OSI management model – extended to include TCP/IP networks.

The OSI model is designed to satisfy the needs for distributed applications. To do so, it adopts an object-oriented approach, modeling all the resources to be managed as **managed objects**. These resources include communications hardware (such as modems, bridges, and gateways) and software (such as LAN and database software).

Object Orientation in the OSI Model

Communication protocols, including OSI and TCP/IP, were developed to provide communications between systems. But, for distributed network management, communication protocols alone are insufficient. Distributed applications require standard descriptions of the resources to be managed.

The OSI standards adopt an object-oriented model for encapsulating these resources and standardizing the interfaces they present to the network. The object-oriented model promotes the following attributes in network management applications:

- **Modularity**: the necessary functionality is decomposed into clearly defined and readily implemented modules.
- **Extensibility**: distributed applications can build upon the functions and services of existing applications.

The TMN standards encourages an object-oriented approach to modeling and solving network management problems. Everything on the network is treated as a managed object, and is manipulated through cooperating distributed processes.

The object-oriented paradigm differs from the procedural model. In object-oriented management applications, code and data are fused into a single entity called a managed object. A managed object is composed of a set of **attributes** which characterize it, **actions** that it can perform, and **notifications** that it can emit spontaneously.

The object-oriented paradigm naturally supports asynchronous processing. That is, a process doesn't necessarily have to block simply because it has issued a request message to some managed object. It can continue processing, perhaps corresponding with other managed objects, while awaiting the object's response.

Elements of Network Management

Before the distribution model for HP OpenView Service Assurance for Communication Networks is described, let's first cover some basic concepts regarding network management communications:

- Managed object
- Managed object class
- Managed object instance

Managed Object

A managed object is an abstraction that models some resource in the real world. A managed object represents some resource, either physical or logical, that is available somewhere on the network, and that has the capacity for being managed.

For example, managed objects might include:

- A computer workstation
- A network interface card attached to a computer workstation
- A collection of free disk space data for nodes on a network

The aspects of a resource that are related to the management of that resource are accessible through the managed object abstraction; other aspects are not.

For example, consider a printer modeled a managed object. The management properties of the printer object include its model, serial number, location, toner level, and so on. It also processes certain management functions, like reset, self-test, and so on. Its color probably is not of interest, and is not included in the managed object abstraction.

Managed Object Class

Managed objects are classified into groups for ease of management and monitoring. The classification is based on the role of the managed object within the managed network. An object class consists of a definition that fully describes what each member of the group looks like and how it behaves.

For HP OpenView Service Assurance for Communication Networks, the managed object classes (MOCs) for a network environment are:

- **Network Class**

  This logical class is the highest class of managed objects. In the containment tree, all other managed objects are contained under a network object. Network class objects can be embedded within other network class objects. Network, network element, and connection class objects can be contained directly under a network class object.

  Within a managed network, there is one special network called **root**. This is a default network class created at installation. All user-defined, first-level network classes are contained under this root object class, which is used as a point of reference for all other managed object classes. This root class has no significance outside the configuration process.

- **Network Element Class**

  This class consists of devices that can emit alarms. In the containment tree, the highest level of managed objects that emit alarms are referred to as network elements. Within a network, managed objects of component and link classes are contained under the network element.

- **Component**

  Component class objects can be contained under network element classes, termination points, and other component class objects within the containment tree. Instances of this class may or may not emit alarms.

- **Termination Point**

  Termination point class objects are those objects that form the end points of the connection between managed object classes.

- **Connection**

  Connection class objects indicate physical links between the end points of two managed objects. The end points of the objects are termination point class objects, which identify the two managed objects that the connection class object connects.

  A connection class object can be contained under any of the above classes of managed objects. The connection class object must be contained under the lowest common parent class object.

For example, suppose the following are two networks:

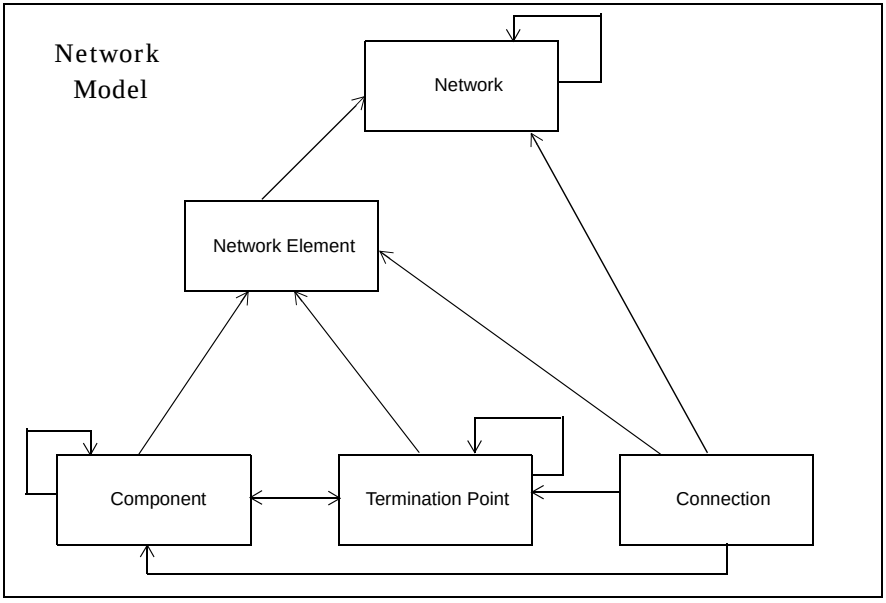
NW/NE1/COM1/COM2/TP1

NW/NE1/COM1/TP2

Then, the connection class connecting the two networks must be created under COM1.

Figure 2-1 provides a diagram of the containment tree supported by OV Topology Server.

Figure 2-1 Supported Network Object Model



Each managed object class has a **naming attribute** assigned to it. Naming attributes describe what managed object classes can look like.

Classes of the same object type can share naming attributes, but naming attributes must be unique across object types. For example, all Component-type classes can share the same naming attribute, but classes of different types (Component, Network Element, and so on) must have unique naming attributes.

Managed Object Instance

An object instance is an individual member of the object class. While all

instances of the same class have the same look and behavior, the values vary from instance to instance.

All managed object instances have one key attribute that distinguishes them from the other instances of the same class. For example, consider an object class of automobiles. This class describes automobiles in general. Assume two instances of automobiles are constructed from the object class. These two automobiles could appear to have identical attribute values, such as model, year, color, and engine size. A key attribute, however, that distinguishes the two automobiles is a manufacturer's vehicle identification number.

Relationship Among Object Instances and Object Classes

This section covers the major relationships among object classes and among object instances. Each of these relationships is hierarchical, and and therefore can be represented as a tree.

The three relationships that provide a structured framework for defining and using objects are:

- Registration
- Containment
- Inheritance

They constitute a common knowledge base in the network and systems management community. Each of these relationships is described in detail in the following sections.

The Registration Relationship

The **registration relationship** is a hierarchy of unique identifiers for object classes, actions, events, and attributes. The registration relationship pertains to these elements and not to object instances. As various object definitions are added to the network community, each is assigned a unique identification number.

An object class identifier (registration ID) is made of a series of integers that describe the path from the root of the registration tree to the node to be identified. Each node of this tree has an associated registration authority that determines how numbers in the subtree beneath it are allocated. For example, the path 1.3.6.1 defines a branch of the registration tree which is administered by the Internet Activities Board. This organization has designated a subbranch (1.3.6.1.4.1.11) for which Hewlett-Packard is the registration authority. This gives HP responsibility for new registration IDs defined under this subbranch.

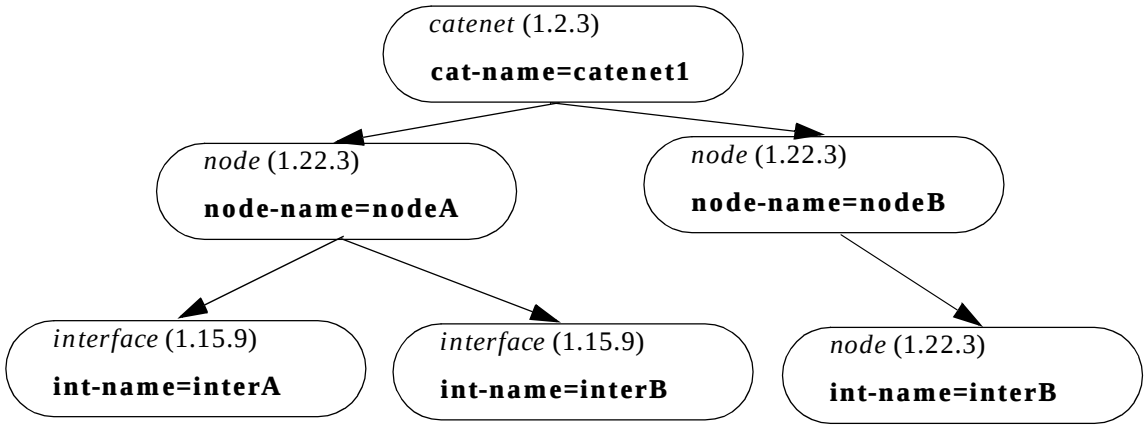
The registration relationship comes into play frequently during object definition and use. It is used during design to construct definitions which are uniquely registered with a central authority. Registration numbers are also used at application execution to identify the class of a managed object instance.

The Containment Relationship

The containment relationship describes how each object instance is related to other object instances within a particular environment. All object instances are contained within other object instances. Note that this relationship pertains to object *instances*, not object *classes*.

Figure 2-2 depicts an example of a containment relationship.

Figure 2-2 The Containment Relationship



The containment relationship is the hierachy of which objects instances are contained in others. Note that the containment relationship implies that an object exists only if the object instance which contains it exists.

Since every object class has a distinguishing attribute that is used to uniquely identify each instance of the of the class, An object instance's distinguishing attribute, together with the value of that attribute, is called the **relative distinguished name** (**RDN**) of the object instance.

By traversing the containment tree from the root to an object instance, you can construct a sequence of RDNs that is unique to the object instance. This unique sequence is called the **fully distinguished name** (**FDN**) of the object instance.

Using Figure 2-2, the following statements are true for the *node* object class:

- The *node-name* is the distinguishing attribute of the *node* object

class.

- The RDN of the node object instance named *nodeB* is `/node-name="nodeB"`.
- The FDN of the node object instance named *nodeB* is `/cat-name="catenet"/node-name="nodeB"`.

The RDN of an object instance is unique only under its parent. One the other hand, the FDN of an object instance is globally unique.

The Inheritance Relationship

In object-oriented programming, inheritance is the property by which one object class is defined as an extension of another. The new class has all the properties or attributes of the parent, plus any additional defined properties.

The **inheritance** relationship describes which object classes are derived from which others. When one object class is derived from another object class, it inherits all the attributes and actions of that base class. Thus, the inheritance relationship is used to define new object classes based on existing ones. Object classes are arranged in a hierachy.

For example, suppose you had to define a new object class called ethernet. You would find that an object class called network already exists and contains many of the attributes of interest. You could simplify your work by borrowing the attributes and actions from the network object class, and extending them to support your new functionality. The property which allows new object classes to be created from previously developed object classes is called inheritance.

Summary of Relationships

Table 2-1 summarizes the relationships and their purpose.

Table 2-1 The Relationships Among Object Classes and Instances

Relationship	Purpose
Registration	Registration applies to the definition of object classes. The class of an object instance is specified by a registration ID number, assigned by a registration authority.

Table 2-1 The Relationships Among Object Classes and Instances

Relationship	Purpose
Containment	Managed objects can contain other managed objects. This hierarchy of containment is used to derive a unique name for each object instance.
Inheritance	Managed object classes can derive their attributes, actions, events by inheriting the definitions of attributes, actions, and events from other objects. This relationship defines the hierarchy of derivation for object classes.

Managers, Agents, and Managed Objects

NOTE

The use of the terms managers and agents in this section are based on terminology defined for Telecommunication Management Network (TMN). Be aware that OVO uses these terms in a different manner. For more information on how OVO defines managers and agents, see the *HP OpenView VantagePoint Operations for UNIX Concepts Guide*.

The environment to be managed is distributed, so it follows that the management activities must also be distributed. This idea is crucial, and its is the basis for the concepts of **managers** and **agents**. It is through the distinct entities called managers and agents that the distribution of management activities is achieved. This is similar to the client-server model, where managers are akin to clients and agents are akin to servers.

- An agent is the part of a distributed management application that supervises one or more managed objects. The agent receives and services requests for the operations that are defined for the managed objects it represents. The agent is also responsible for emitting notifications (events or traps) when it detects special conditions in the managed object.
- A manager is the part of a distributed management application that issues requests for operations and that receives notifications. That is, a manager uses the services of one or more agents. Managers also interact directly with the user interface; agents do not. A manager receives requests from the user interface and displays information back to the user. Managers do not manage objects directly; rather they issue requests to agents.

Note how these definitions express how agents and managers interact without limiting these interactions. That is, an agent is apt to respond to the requests of multiple managers, and a manager is apt to request the services of multiple agents. Thus, when a manager is created to make use of one or more existing agents, the new manager reuses the investment that created each agent. In defining managers and agents, you produce flexible systems that exploit the distributed nature of the HP OpenView Telecom DM platform.

It is important to distribute the functionality of a solution appropriately between managers and agents. Table 2-2 provides guidelines.

Table 2-2 Manager and Agent Division of Functionality

Element	Description
Manager	• Interfaces with users • Uses the services of one or more agents • Does not generate events • Is not required to be running all of the time
Agent	• Represents information and functions of possible interest to more than one manager • Does not interface with users • Is always running • Interfaces with some type of managed objects • Detects special conditions in managed objects and emits corresponding events.

Manager Functionality

A manager is essentially a proactive entity that acts as the tool of a user to obtain, analyze, present, and act upon information about the networked environment. This includes:

- Collecting, storing, and processing raw data.
- Presenting meaningful information to the user in an understandable form.
- Obtaining user requests and acting upon them. User requests might include changes in certain elements of the environment or gather additional data.

In addition to their proactive role, most managers are alert to unsolicited messages that may indicate a change in status somewhere in the environment. Thus, while most of the activities of a manager place it in the role of a requestor, it takes the role of a responder when it receives (asynchronously) an event report.

Agent Functionality

An agent is essentially a reactive entity that waits for and services the requests of one or more managers. An agent is a manager's window into the management aspects of the managed object.

Thus, agents are fundamentally request-driven, responding to requests from managers. When an agent receives a request, it needs to invoke the appropriate functions to handle the request.

In addition to their reactive role, most agents are also responsible for flagging certain special events that occur in the objects they manage. These event messages receive special treatment and are relayed to any manager that has expressed interest in receiving them.

Thus, while most of the activities of an agent place it in the role of a responder, it takes the role of a requestor when it asynchronously issues an event report.

Object Managers as Complex Agents

Up to now, this chapter has presented a fairly simple two-process model, where agents directly represent a resource modeled as an object, and managers make direct use of those agents. The reality of many networks is more complicated, involving several layers of processes between the manager (as it is defined here) and the resources being managed.

In general, the mapping between an agent and its managed object is considered private; the implementor of the agent uses whatever tools are available and appropriate to interact with the actual resource. Suppose, however, that some agent (called A1) has access to the resources it manages via some other agent (called A2). The manager views A1 as just another agent, but the second agent A2 views A1 as a manager.

For the purposes of this discussion, an agent like A1 described above is called an **object manager**. An object manager has the characteristics of both an agent and a manager, except that it typically has no user interface component. In other words, an object manager has both proactive and reactive roles.

Once the simple two-process model has been extended this way, there are innumerable applications, some of which are not strictly hierarchical. Among the more common applications are **proxy** agents and **mediation**

devices.

Table 2-3 Proxies and Mediation Devices

Type	Description
Proxy	A proxy is a special agent that acts as a dedicated translator for a resource whose actual agent uses a foreign protocol. A proxy takes requests in one language (such as CMIS), translates them to the foreign language, and submits the translated request to the actual agent. Any response from the actual agent is picked up by the proxy, translated into the language of the request (CMIS, in this case), and sent to the manager.
Mediation Device	A mediation device is an object manager whose managed object is typically a large subnet, and which represents that subnet to a higher-level manager. Mediation devices usually appear only in very large networks, and may or may not include the translation function of a proxy. The function of a mediation device is to distill the massive quantities of information available on subnets, extract the most important and relevant data, and make the data available to a higher-level manager. At the option of a manager, a mediation device must pass through detailed information about the subnet it represents. In addition, a manager may bypass the services of the mediation device, and converse directly with lower-level object managers and/or agents in the subnet.

These two options do not exhaust the possibilities for using object managers. However, be aware that the design, implementation, maintenance, and administrative costs associated with highly complex designs are high.

Network Management Communication Model

This section describes the two widely-accepted network management protocols.

- CMIP, the Common Management Information Protocol, is designed for managing objects in an OSI environment. The services defined for CMIP are known as the Common Management Information Services, CMIS.
- SNMP, the Simple Network Management Protocol, is designed primarily for managing objects in TCP/IP environment.

CMIP, CMIS

CMIP is a connection-oriented protocol that allows network elements such as hosts, terminal servers, gateways, and management agents to be manipulated via sophisticated messages. The services defined for the CMIP protocol are known as CMIS.

SNMP

SNMP is a transaction-oriented, stateless protocol that allows network elements such as hosts, terminal servers, gateways, and management agents to be queried directly. Because it has low network overhead when making such queries, it provides an inexpensive way of gathering network statistics. It is therefore ideal for real-time monitoring and many other applications.

Object Modeling Overview

Automated access to a manageable object requires a clear statement of the external view of the object. Object modeling is the task of formally describing the important characteristics and operations of an object class. Once an object class is modeled, object managers and agents can be written to manage the objects modeled.

- It is possible to have one or more agents that manage instances of the class you modeled.
- It is possible to have one or more managers that interact with the agents, and through them manage such object instances.

Furthermore, specialized tools can use the model to perform other useful tasks, such as automatically generating much of the specialized code necessary for building managers and agents, or automatically building database schemas for the object class.

Therefore, object modeling is crucial step in creating managed objects and management solutions. This section outlines basic concepts of object modeling:

- Describe the components of an object class definition: attributes, actions, behaviors, and notifications.
- Create a package template to integrate the components into a package.
- Bind the various packages together with the managed object class template.
- Use the name binding template to define the distinguishing attribute of the managed object class.
- Use the inheritance property to construct new object class definitions.
- Use the registration tree to uniquely identify an object instance.
- Use metadata to make object definitions available dynamically.
- Use a formal process to register the object definitions for use by others.

3 Understanding OV Telecom Extensions for OV Operations and OV Topology Server

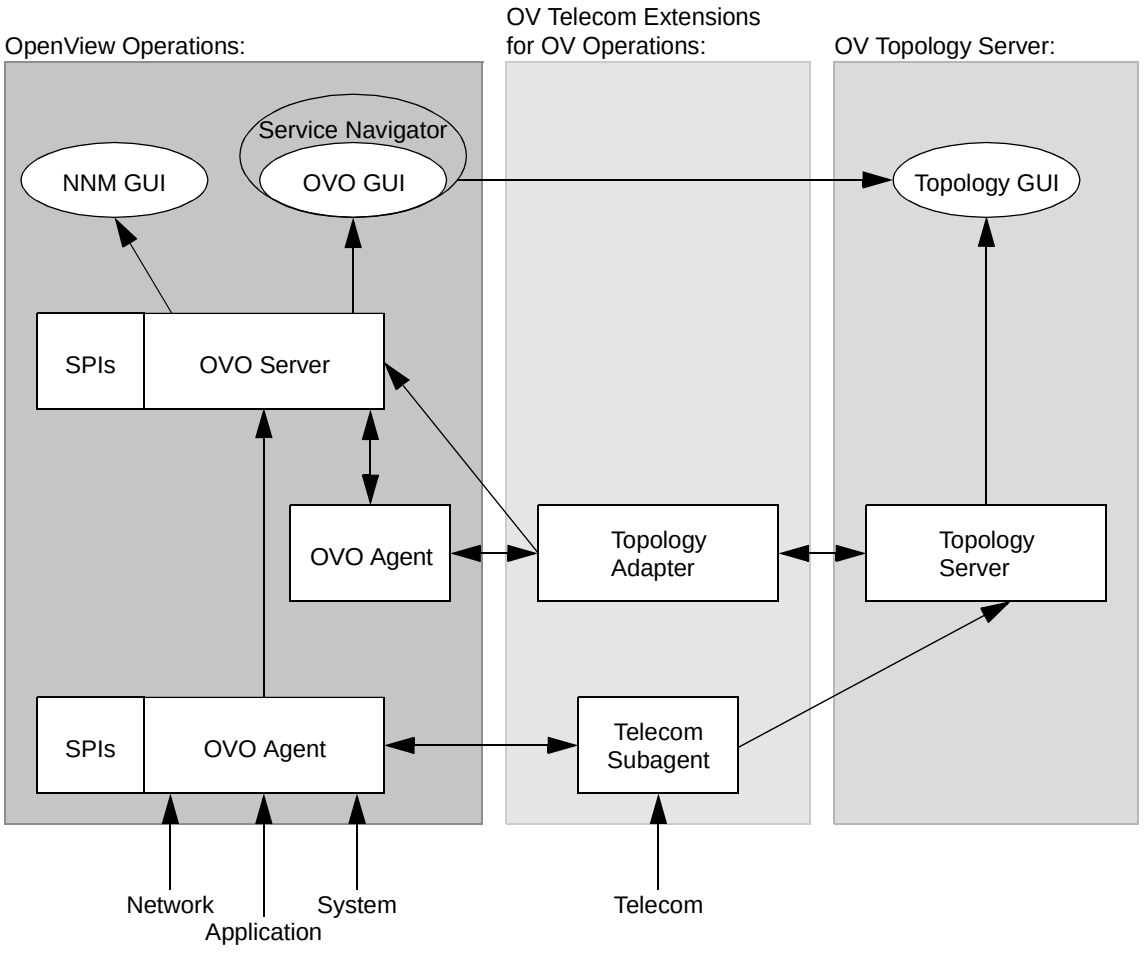
Understanding OV Telecom Extensions for OV Operations and OV Topology Server

This chapter explains the product architecture for OV Telecom Extensions for OV Operations and OV Topology Server. The topics covered in this chapter build upon concepts described in the *HP OpenView Service Assurance for Communication Networks Concepts Guide*.

Product Architecture

HP OpenView Operations (OVO), including HP OpenView Network Node Manager and HP OpenView Service Navigator, serves as the foundation for HP OpenView Service Assurance for Communication Networks (OVSACN). The two add-on products to OVO that comprise OVSACN are OV Telecom Extensions for OV Operations and OV Topology Server. This section describes the major components of each add-on product and how they work together to provide an integrated solution for communication service providers.

Figure 3-1 OVSACN Product Architecture



OV Telecom Extensions for OV Operations

The main components of OV Telecom Extensions for OV Operations are:

- Telecom subagent
- Telecom injector and divertor
- Telecom adapter

Telecom Subagent

The main component of OV Telecom Extensions for OV Operations is the telecom subagent. When OV Telecom Extensions for OV Operations is integrated with OVO, messages from telecom devices are processed and displayed in the message browser.

The telecom subagent acts as a data collector to receive messages from telecom network elements and telecom element management systems. The telecom subagent formats the received messages using the configured message source templates.

When the optional OV Topology Server is configured, the telecom subagent maps received messages to ITU X.733 alarm reporting format. The formatted messages are then forwarded to the topology server over socket connections or the OV DM TMN infrastructure. Multiple telecom subagents can be connected to one topology server.

The supported data collectors are:

- TCP, both active, passive
- FIFO
- SNMP, with the OVO injector
- Custom data collectors developed by solution developer

NOTE CMIP data collectors are supported, but only when CMIP messages are received and processed directly by the topology server.

Telecom Injector and Divertor

When the OV Topology Server is configured, the telecom injector and divertor make decisions on whether messages need to be formatted and forwarded to the topology server for further processing. If the OV Topology Server is not configured then telecom-specific messages are forwarded to the OVO server and the message browser.

The topology injector and divertor tap into the agent message interface to provide the following capabilities:

- Map values in message fields based on table lookups.
- Perform time base arithmetic to adjust time stamps.

- Provide local topology or intradevice-based correlation when the topology implicit in the messages is sufficient for correlation behaviors.
- Forward messages to the topology server for further processing when OV Topology Server is installed, or forwards messages to the OVO agent for continued handling.

Telecom Adaptor

When the OV Topology Server is configured, the telecom adaptor forwards problem notifications from the topology server to the OVO agent for inclusion in the message browser. It maps source topology elements to nodes in the OVO node bank for maintaining the two OVO GUIs (operator and admin GUIs). The telecom adaptor also maintains the synchronization between the message browser and the problems presenter by sending messages directly to the OVO server.

The telecom adaptor provides the following capabilities:

- Message creation based on problem notifications from the topology server.
- Message and problem state synchronization, including own, severity, and discharge.
- Assignment of messages to services based on topology, when Service Navigator is deployed on the OVO system.
- Maintenance of service state based on topology state, when Service Navigator is deployed on the OVO system.
- An element-to-service mapping when Service Navigator is deployed on the OVO system.

OV Topology Server

OV Topology Server main components are a topology server and a topology GUI.

The topology server consists of a FM Server and a GUI Server.

FM Server

Each telecom subagent is linked to one topology server, more specifically one FM Server. The main functions of the FM Server include:

- Logging messages received from the telecom subagent into the Oracle™ relational database.
- Correlating the messages based on user-configured criteria. At this level, correlation is applied in the context of the network topology.
- Consolidating the correlated messages into problems based on the following common attributes:
  - — Managed object instance
  - — Alarm type
  - — Probable cause
  - — Specific problem
- Determining managed object status based on the event messages received and distributed to the topology GUI. Status calculation is governed by user-definable status propagation rules such that the status of the network reflects the health of the network elements.
- Processing and sending requests to and from the topology GUI.

To manage this functionality, the FM Server has two major components:

- The **FM Server modules,** which consist of:
  - — **Alarm Logger**, which logs the alarms received from the agent.
  - — **FMS Event Correlator, (OEMF-EC) and the Event Correlation Service (ECS),** which correlate the alarms received.
  - — **Network Status Monitor,** which propagates the alarm status for the network object symbols on the map presenter.
- The **FM Server CORBA Interface modules**, which present the unified view to the entire network managed by a set of FM Servers. Each interface module consists of a set of CORBA object servers that support and implement the network object model.

  CORBA object servers contain and manage the objects entirely on their own, and are also called collection managers. There are two types: **unified service collection managers** and **replicated collection managers**.

  The unified service collection managers participate in and provide the unified services. They usually cooperate with peer collection managers to provide a unified service.

The replicated collection manager replicates the collection manager and its contained objects across all peer machines.

CORBA Server: The following are the collection managers in the FM Server:

— MO Manager, a unified collection manager that manages the managed CORBA objects. Each MO Manager manages a subset of the managed objects in the entire managed network.

— MOC Manager, a replicated collection manager that manages MOC CORBA objects. The same set of managed object classes is replicated across the entire managed network.

— Problem Manager, a unified collection manager that manages Problem CORBA objects.

— OutagePlan Manager, a unified collection manager that manages the OutagePlan CORBA objects.

— Alarm Manager, a unified collection manager that supports alarm queries and management interfaces.

— OM Event Manager, a unified collection manager that manages OM Event CORBA objects.

GUI Server

The GUI Server consists of a set of CORBA application handlers that interface between the FM Server and the topology GUI. The application handlers perform the user functions with one application handler for each user function.The application handlers are:

- **Problem Handler.** This handler handles problem events such as owning, disowning, and discharging problems, viewing problem details, annontating problems, associating trouble tickets with problems, and querying problems, network elements, and alarm history.
- **Map Handler.** This handler enables topology browsing and topology maintenance, such as creation and deletion of managed objects and managed domain management.
- **Outage Plan Handler.** This handler provides the interface for operators to schedule and manage outage plans for the managed objects.
- **OM Event Handler.** This handler facilitates management of the

object management event life cycle. This involves the functions of owning, disowning, and discharging object management events.

Topology GUI

The topology GUI provides the graphical user interface for presenting network details to an operator. For each handler on the GUI Server, there is a corresponding presenter on the topology GUI that displays the related information to the operator. The presenters are:

- **Problem Presenter.** This presenter displays a list of problems to the operator in tabular form. Problems relate to the subset of the managed network to which the operator has access. Through this presenter, an operator can view the problem details, the problem history, and the network element history. Operators can set up filters to define the type of information they wish to view, and within the purview of their access rights, this information is displayed. It also allows users to annotate problems and query problems and problem history.
- **Map Presenter.** This presenter displays the managed network in graphical maps and uses color-coding and graphical symbols to indicate the network objects that have associated problems.
- **Outage Plan Presenter.** This presenter defines, monitors, and maintains the outage plans for network elements in a managed network.
- **Chart Presenter**. This presenter displays statistics about problems via charts.
- **OM Event Presenter.** This presenter facilitates management of object management event messages. This involves the functions of owning, disowning, and discharging object management events.

HP OpenView Telecom Smart Plug-Ins

Smart Plug-ins (SPIs) enable customers to customize their solution based on their target environment by providing message source templates specifically designed to capture messages of a certain form from managed devices. Message source templates identify the beginning and ending markers for messages in message streams, and come in two flavors:

- Basic templates perform minimal functionality, enough to capture

messages from particular network elements and format them into a more human-readable format.

- Topo-Smart templates are specially customized for particular network elements, providing formatting capabilities and ways of identifying messages as related to topology objects. A topo-smart template can work with or without OV Topology Server.

For example, Smart Plug-ins might be provided for Nokia Wireless environment. These Smart Plug-ins, when installed, include the definition of the required managed object classes, message source templates, and application integrations for devices typical in a Nokia wireless setup.

How the Components Work Together

This section explains how all of the components of HP OpenView Service Assurance for Communication Networks work together. In particular, dependencies among components are highlighted.

Data Collection Process

Let's first talk about the data collection process. A data collector is configured on an OVO agent system, and can contain several user-defined sources. Sources are used to collect alarms from telecom devices. The data collector collects all alarms and processes them based on user-configured rules.

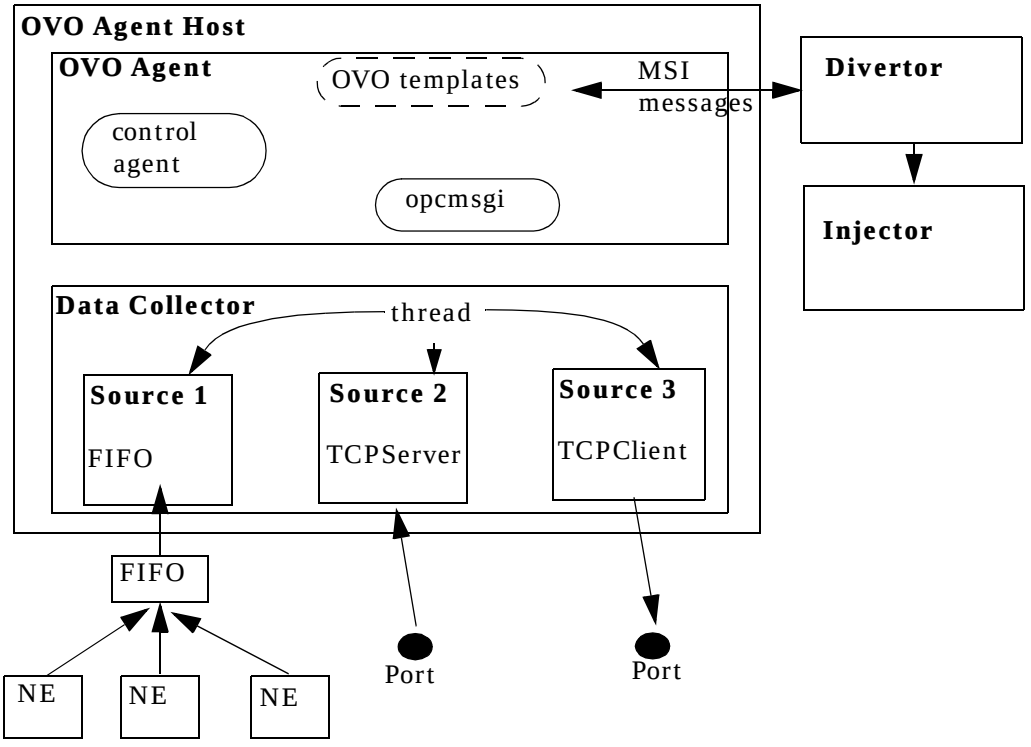
Figure 3-2 depicts how the data collector might be configured to receive alarms from telecom devices.

Multiple sources can be connected to a single data collector. For each source, FIFO and TCP/IP are acceptable input connection types. Other acceptable connection types, such as SNMP, need to be configured to communicate directly with the OVO agent.

- For a source acting as a TCP Server, a unique local TCP port is made available. All messages from devices which connect to this port are received by the source. The source in this case acts as a listener, reading data as it comes in.
- For a source acting as a TCP Client, the source connects directly to an assigned host and port. The source reads all data to which it is connected.
- For a source acting as a FIFO connection, there is a unique file located on a local box to which network elements are connected.

Consider each source as a thread. Thread processes include reading received alarms, parsing messages using a high level parser, and forwarding alarms when alarms conform to parsing rules. Otherwise, alarms are logged to files, when auditing is turned on.

Figure 3-2 Sources, Data Collectors, and Agents



To improve the performance of your system, use the OVSACN data collector to filter unwanted alarms before they proceed to the OVO agent. View a log of alarms emitted from the network elements connected to the data collector, and decide which alarms are noteworthy and which alarms should only appear in a log file.

To reduce the number of unwanted alarms at the data collector level, edit the record formats section of the agent configuration file. Record formats define the beginning and ending markers of alarms. All messages received by the data collector not matching the defined record formats are stored in a log file, when auditing is on. You can define multiple record formats for a single data collector.

Figure 3-3 shows how sample alarms might be parsed by the data collector. The underlined text indicates the beginning and ending markers for the defined record formats. Thus, only three messages are

forwarded to the OVO agent for further processing.

Figure 3-3 Sample Alarms

##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#

Template Processing

At this stage, 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 which further filter unwanted alarms from alarms of importance 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.

In order for messages to be processed properly by templates and displayed in the message browser:

- The node defined in 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.

Use table lookups to reduce the number of conditions needed for a template. Table lookups retrieve values from tables defined in an agent configuration file. Table lookups are optional and could be used to translate parameters such as error codes, location codes, and so on into more readable strings.

The telecom divertor processes the table lookups after the templates have been processed. The Message Stream Interface must be enabled for

the divertor to function properly. After the divertor processes the table lookups, the messages are either sent back to the agent for further processing or forwarded to the telecom injector when OV Topology Server is configured.

Topology Server Processing

After templates, and optionally table lookups, are processed, messages are reviewed to see if they are telecom alarms. The divertor identifies messages as telecom alarms when `Telco_` is stored in the Message Type field. Messages that have XML content embedded (XML X.733 fields entered in Message Text field of templates) are forwarded to the injector, which retrieves telecom-specific messages and forwards them to the topology server. When OV Topology Server is not configured, all messages are forwarded to the OVO server and up to the message browser.

Messages that are forwarded to the topology server are correlated based on event correlation circuits or rules. Typically, the types of correlation performed are root-cause/related and time-based correlation. Further processing is performed on the topology server and the messages are forwarded to the topology GUI. Messages are also forwarded to the OVO server and to the message browser.

User Access

Administrators assign operation profiles to users, which limit the types of messages that can be displayed in the users' message browsers. For example, during configuration a user profile called `Telco_Op` is created, which only has access to the topology-specific applications, nodes, messages groups, and service definitions. Thus, a user who logs in with the profile `Telco_Op` does not see system or NNM-type alarms in the message browser.

4 Understanding Configuration Tools

This chapter provides an overview of the configuration tools needed to set up a managed network within HP OpenView Service Assurance for Communication Networks. The overview includes:

- Description of the Telecom Configurator.
- Description of the files to be manually configured.

NOTE The tool used to define the operation profiles and user permissions is defined in Chapter 18, “Additional Topology Setups.”

Telecom Configurator

This GUI-based configuration utility enables the definition of an OV Topology Server and OV Telecom Extensions for OV Operations installation. It is used to design, develop, and deploy configuration data needed for the creation of a topology-oriented management solution.

The Telecom Configurator enables a network administrator to create, view, modify, validate, and deploy configuration data to the OVO server system.

About the Telecom Configurator

Only the network administrator can use the Telecom Configurator to configure monitored network and message class details.

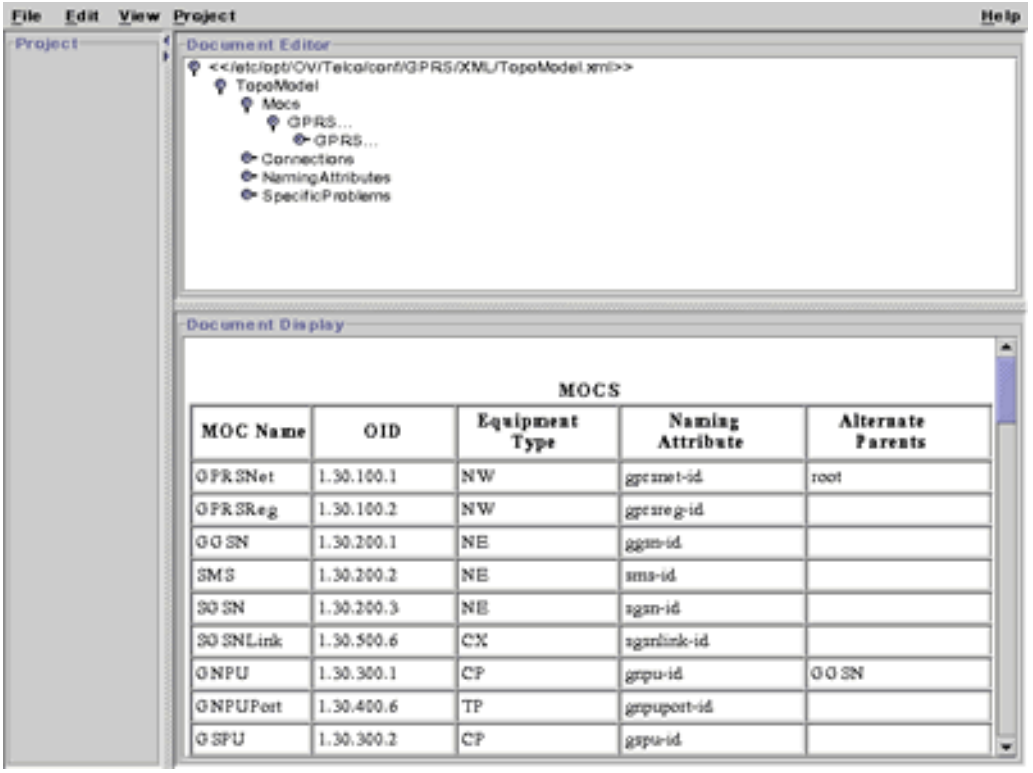
Start the Telecom Configurator as user `root` by typing: `ovcfgsa`

The Telecom Configurator is displayed (as shown in Figure 4-1) which consists of a menu bar, the project area, the document editing area, and the document display area.

The document editing area is where all modifications to the content of the XML configuration files takes place. To add, remove, or modify entries on the configuration data, select the object of interest in the document editing area and apply the appropriate command from the menu bar.

The document display area displays the resulting configuration data in HTML format. Please be patient as the document display area updates your configuration data in HTML format.

Figure 4-1 The Telecom Configurator



The menu bar contains the following functions to maintain the configuration tables. The following table briefly describes these functions:

File Menu Has the functions `New`, `Open`, `Close`, `Validate`, `Save`, `Save As`, `Revert`, `Validate`, and `Exit`. These functions relate only to files.	
`New`	Creates a new configuration file for editing, or a new project. A *`project`*.xml file is created with the same name as the project.
`Open`	Opens an existing configuration file of a project for editing.
`Close`	Closes an open configuration file. A dialog box displays asking if you want to save changes to the file before closing.
`Save`	Saves the configuration details you have entered in the open file. While the information entered is committed, the information *is not saved* into the configuration file until you select `File:Save`. Selecting this function saves all changes made from the moment the configuration file was opened until `File:Save` is invoked, or from the last time `File:Save` was invoked to this one.
`Save As`	Saves the configuration details you have entered in the open file to file of a new name.
`Revert`	Reverts to previous saved version of the open file.
`Validate`	Checks the open configuration file against its appropriate DTD file. Validating a file only checks for DTD syntax validity; you must validate the project in order to check the semantic validity of the configuration files.
`Exit`	This function is used to terminate the configurator session.
Edit Menu Has the functions `Cut`, `Copy`, `Paste`, `Add`, and `Modify`. They are used to maintain the configuration data in the configuration files. Select an object in the document editing area before clicking one of these functions.	
`Cut`	Cuts the contents of the selected row into the edit buffer. It overwrites the current contents of the buffer and the next Paste function pastes the contents cut by this action.

Copy	Copies the contents of the selected row into the edit buffer. It overwrites the current contents of the edit buffer, if any. The Edit:Paste operation pastes the contents of this Edit:Copy to the selected row.
Paste	Pastes the contents of the edit buffer under the selected row. You can only paste information from the last cut or copy operation.
Add	Inserts a a new entry to the configuration file under the selected row based on the type of object selected in the document editing area. Clicking Add opens a configuration window.
Modify	Modify the contents of an entry in the configuration file. Clicking Modify opens a configuration window. A shortcut to this function is to double-click a node.
View Menu Has the option of Console, which launches a read-only window containing all messages outputted by the Telecom Configurator. Open the Console window to view error messages when validating files and projects.	
Project Has the functions Add File, Add Current File, Remove File, Rename File, Deploy, and Validate. These functions relate only to configuration files added to a project.	
Add File	Adds a file to the project directory and the project.xml. The Telecom Configurator can identify which type of configuration file it is and place it it in the right spot in the project.xml file.
Add Current File	Adds the current file to the project directory and the project.xml file. The Telecom Configurator can identify which type of configuration file it is and place it in the right spot in the project.xml file
Rename File	Renames a configuration file in the project directory and the project.xml file.
Remove File	Deletes a file from project.xml file only.
Deploy	Generates target configuration files from all of the configuration files defined for that project. The deploy menu item validates the configuration files within the selected project and generates the target configuration files.
Validate	Checks the validity of the configuration files against the respective DTDs and the semantics of the configuration files.

Role of the Telecom Configurator

Use the Telecom Configurator to configure the following information:

- **Managed objects**. These are all the objects that are to be monitored by OV Topology Server. These objects are defined by their fully distinguished names (FDNs) and the topological tree.
- **Network object model**. This enables the classification of the various devices being monitored into five object classes - network, network element, termination point, connection, and component. The configuration information for each class includes class name, object type, attributes, subcomponents, type of parsing applicable, message configuration, and connection points. The highest class of objects that can emit event messages is classified as network elements.
- **Partitions**. A managed network is divided into partitions, where each partition is a subset of the managed objects in an installation belonging to a single location. All managed objects in the network should belong to at least one partition.

  Partitions can be as large as the location. Partitions can be defined independently and then linked to locations.
- **Locations**. Each installation can be divided into as many locations as the number of topology servers in it. Each topology server, with the telecom OVO agents linked to it, forms a location within an installation. To define a location, the topology server and part of the managed network (or the partitions) it monitors should be defined using the Telecom Configurator.
- **Telecom OVO subagent**. The format to be used to transfer messages to the topology server and the ports connected to the telecom subagent must be configured for each telecom OVO agent. Port specification includes the characteristics of each port linked to the telecom subagent, such as time zones and attributes of the message transfer from the port to the data collector (protocol, data type). This facility is crucial for legacy systems that do not transmit sufficient information in their alarm messages. OV Telecom Extensions for OV Operations adds port specific information to alarm messages from legacy systems into ITU-T X.733 alarm reporting format.
- **Message Parsing**. The OVO agent is responsible for filtering and formatting messages for OV Topology Server, when it is configured. Messages are formatted according to information in configured

message source templates. The OVO agent forwards relevant information in the form of messages to the OV Topology Server.

Topology Server Configuration Through Files

The following information can be configured by editing relevant files:

- **Event/Alarm Correlation Rules.** Event correlation filters superfluous messages based on user-configured criteria. OVC/Assurance provides two event correlation tools: Event Correlator (OEMF-EC), a rules-based correlation tool, and HP OpenView Event Correlation Services (ECS), a circuit or filtered-based correlation tool. Either tool can be configured and used transparent to an operator.

  Event and alarm correlation rules define how events and alarms are to be processed in the Event Filter and the OEMF-EC or ECS correlation tool. The option of filtering all repeated alarms or those which match a user-defined pattern is provided.

- **Status Propagation Rules**. Status propagation gives operators a bird's eye view of the status of a monitored network. Status propagation rules define how the status of lower level object instances in the containment hierarchy is to be propagated to object instances above them and displayed on the Map Presenter. For example, an object can show critical status if more than 50 percent of its subcomponents have critical or marginal status.

5 Understanding Configuration Files and XML

What You Need to Know About OVSACN and XML

HP OpenView Service Assurance for Communication Networks uses XML files as the means of storing and retrieving most of the configuration information. XML is used to represent configuration data for OV Topology Server and OV Telecom Extensions for OV Operations. XML is used to store configuration files on the file system.

This chapter describes XML in general terms, and explains how OVSACN uses it for various types of configuration information.

If you are already familiar with XML and do not need an overview of XML, go to “Use of XML for OVSACN Configuration Files” on page 78.

Overview of XML

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

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

Therefore, XML is an enabling technology for creating vocabularies that represent data that you can easily share with others. A significant number of XML vocabularies are being written and used today, such as domain-specific markup languages for math, music, and 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 am 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.

Use of XML for OVSACN Configuration Files

OVSACN configuration files can be categorized according to five configuration types. Table 5-1 lists the XML configuration types and the names of the XML files for the associated configuration types.

Table 5-1 XML Configuration Files

Configuration Type	Template Configuration File
Topology Model Configuration	`TopoModel.xml`
Topology Instance Configuration	`TopoData.xml`
Agent Configuration	`Agent.xml`
Mapping Configuration	`Mappings.xml`
Project Configuration	*`Project`*`.xml`

When a new project is created, the XML configuration files listed in Table 5-1 are created and stored in a project directory. 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`.

For more information on working with projects and project configuration files, see Chapter 7, “Create a New Project,” on page 101.

Topology Model Configuration

The topology model configuration file, `TopoModel.xml`, holds definitions of the topology model for a managed network.

In this file, the following data must be configured:

- Managed object classes
- Connections between managed object classes
- Naming attributes
- Specific problem definitions

Topology Instance Configuration

The topology instance configuration file holds definitions to instances of managed object classes called managed object instances as well as distribution rules for the managed object instances.

In this configuration file, the following data must be configured:

- Managed object instances
- Distribution rules for managed object instances, including
  - Partition name
  - Partition mask value
  - Location name

Agent Configuration

The agent configuration files hold specific configuration details about the managed objects instances within the managed network as well as the OVO agent hosting a telecom subagent. An agent XML file should be configured for each OVO agent with a telecom subagent.

In these files, the following data must be configured:

- Data collector
- Source (input)
- Source detail definition
- Equipment list
- Record format for each equipment
- Network elements in the agent's scope
- Table lookup definitions
- Connections

Mapping Configuration

The mapping configuration files hold specific configuration data for the various types of mappings, including:

- Element to service maps, which identifies which topology objects are

mapped to services in HP OpenView Service Navigator.

- Shortname FDN maps, which enables the matching of lower physical topology objects with the virtual upper topology, in order to create the full TMN containment tree.
- FDN node maps, which help organize telecom messages between OVO and OV Topology Server.

You can only define mappings for previously configured services and topology data.

Project Configuration

The project configuration file maintains a list of all of the configuration files needed for a project. The project configuration file and its listed XML files must exist before the configuration data can be installed on the appropriate systems.

This configuration file also contains settings and preferences that control the behavior of the OVSACN network configuration tool.

How XML Files are Updated

- Edit XML Files
- Deploy XML Files
- Apply XML Files

Projects

All of the XML configuration files needed to configure an entire managed network should be stored in one directory, called a project, and added to a *Project*.xml file.

Editing XML Files

Use command line tools provided with this product to enter and update configuration data for your managed network.

Alternatively, use the Telecom Configurator to perform the same functionality. The Telecom Configurator is an XML editor and configuration tool provided with OV Telecom Extensions for OV Operations and OV Topology Server. See “Telecom Configurator” on page 63 for more details on how to use the Telecom Configurator to edit configuration files.

To start the Telecom Configurator, execute:
`/opt/OV/Telco/bin/ovcfgsa`

NOTE Configuration steps outlined in the following chapters are first described using the command line tools. An additional section describes how to perform those same steps using the Telecom Configurator.

Deploying XML Files

After configuration data is entered, deploy the XML configuration files to the designated components.

A network administrator can validate the XML configuration files and generate the target configuration files using the `ovcfgdeploy` command.

The deploy command informs the user of the changes from the previous deploy and highlights the areas of configuratin to apply. See its man page for more details.

From the Telecom Configurator, click `Project:Deploy` to generate target configuration files from the configured XML files. The deploy menu command validates the XML files within the selected project and generates the target configurations files.

Applying XML Files

After configuration files are deployed, apply the configuration data to the designated systems. The commands to apply data from configuration files first copy the files to the respective OV Telecom Extensions for OV Operations and OV Topology Server components, and then install the configuration data.

Use the apply commands to copy and install configuration data to the respective OV Telecom Extensions for OV Operations and OV Topology Server components. You cannot do this from Telecom Configurator.

The commands to apply are:

- `ovtopomodel.apply`
- `ovtopodata.apply`
- `ovoconf.apply`
- `ovagt.apply`

Rules for Editing XML Configuration Files

- Validate the XML after you modify it.
- Be careful not to lose changes made through the GUI. This can happen when you and/or another network administrator edit through the XML and/or the configurator GUI at the same time.

Backing Up XML Files

Backups of XML configuration files are automatically created after each deploy. Backups provide the ability to revert to a previous version of the configuration files.

Validating XML Files

The Telecom Configurator can detect and report an invalid XML configuration file within a project. After making modifications to an XML file in your project, you may want to validate you XML syntax.

To validate your XML files within a project from the Telecom Configurator, select your project and click `Project:Validate`. `Validate` checks whether the XML file is both well-formed and valid.

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

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

To validate a particular file in your project, click `File:Validate`. Validating a configuration file with this command only checks for validity against its corresponding DTD.

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

Avoiding Loss of Changes

If you are editing the configuration files with the Telecom Configurator and editing the configuration files directly at the same time, be careful not to lose the changes made through the interface by writing out the file over the interface changes. This is especially important to remember if more than one network administrator will be editing the configuration files for a project at the same time.

6 Planning Solution Configuration

This chapter helps you plan and develop a blue print of the configuration details needed for a managed network solution.

Checklist for Solution Configuration

This section gives a high-level description of the steps you should follow to plan a HP OpenView Service Assurance for Communication Networks management solution. At this early stage, communication between the system integrator and the administrators (end-users) is important to ensure the needs of the customer are met and the structure of the network to be managed is designed properly.

1. Analyze requirements.

   Identify end-user needs and arrive at a definition of what your solution should look like when completely configured. Consider the following questions:

   - How many operators will the solution configuration need to support?
   - How many physical locations will operators and administrators be managing?
   - How do you want to divide up the work among operators – Per equipment type? Per location?

2. Define agents and element management systems.

   A solution configuration can contain one or more agents. Each agent monitors a set of network devices disjoint from devices being monitored by other agents.

   Agent configuration consists of:

   - Defining data collectors.
   - Listing equipment.
   - Defining record format.

   An agent receives messages from network elements and converts them into X.733 telecom alarms. An agent is configured with a set of network elements and element management systems that it is responsible for. Agent configuration defines an object instance for each network element and intradevice topology that is modeled in the topology server. The device object instances are the lower physical part of the TMN containment tree.

3. Define alarm formatting rules.

   System integrators review raw log files of alarms with end-users to key alarms to identify and analyze key alarms for requirements of mappings. Mappings should be configured so that alarms from telecom devices appear in the message viewers as operators would prefer to view them.

4. Define the network object model.

   As a solution is designed, one of the main deliverables of a design is the definition of the object classes for the object instances that are populated in the topology server. Your object model is derived from your inventory of equipment. You need to define a managed object class (MOC) for each object type that you want to create in the topology server. Note that the topology model will determine (limit) how you can manage your network.

   You can define managed object classes for upper level topology objects that are logical and have no physicalness to them. You can also create managed object classes for network elements and intradevice topology within a network element.

   When modeling objects in a network, consider these questions:

   - What resources on the network need to be managed (physical and logical)?
   - How will each resource be managed?
   - What characteristics of each resource should be managed?
   - What similar resources have already been modeled as objects?

5. Configure the upper level topology.

   When the managed object classes and network topology are defined, you must enter this information in the configuration files provided with the product. Specifically, you must:

   - Define the equipment types and managed objects.
   - Define the containment relationship of the managed objects.
   - Define the managed object classes in terms of their registration IDs, naming attributes, specific problems, and so on.
   - Define the partitions, locations, and root instances.

6. Plan the lower level topology.

If customers agree to the upper topology model presented, then the next step is to plan the configuration of the telecom agents and the element management systems. Specifically, you must plan:

- The types of alarms to configure of interest to customers.
- The definition of network elements.
- How alarms are going to be processed.
- The number of equipment per element management system.
- The number of equipment per agent.
- The level of equipment to be managed (I.e., switches, cards, ports).

7. Stitch the upper and lower topology together.

   In OVSACN, the TMN containment tree can be thought is of as consisting of two parts, the upper virtual topology and lower physical topology. The configuration of these two topologies is defined in separate configuration files. So, the third step of stitching the two topologies together is necessary in order to create the full TMN containment tree. This is accomplished by defining network element shortname-to-FDN mappings.

   Each agent in OVSACN is responsible for an island of managed objects. Minimally, these objects are network elements. Each network element is assigned by the agent a shortname that is used for the object label on the topology objects in the map presenter. The network element shortnames are also used to map the network element (physical element) to the virtual upper topology. The shortname mapping table, `ShortNameFDNMap`, contains one entry for each network element that is part of the containment tree. An entry contains the shortname and a corresponding FDN of the upper topology object that represents the network element. The intradevice topology within the network element is then assumed to be under the network element object in the containment tree.

8. Define system distribution details.

   When using OV Topology Server as part of the network management solution, topology objects are defined in a logical containment tree and classified into the following distribution according to the desired management strategy.

   - Partitions

Partitions are logical categories designed to help administrators assign the management of object instances to operators according to their skill sets. A partition can belong to only one location.

- Locations

  You must define the names of the locations and the partitions contained under each location. Only one location can be defined per topology server.

- Root instances

  A root object instance is a logical instance added to the system distribution definitions that is not part of the actual managed topology. A root instance is the highest level instance in a distribution containment tree that determines the path of navigation for all object instances contained under it. Multiple root instances are allowed.

9. Configure operator access management.

   Devise a plan for how operators are to manage the health of the network. This involves determining:

   - The managed object domains each operator will have access to.
   - The tasks and applications available to each operator.
   - The number of operator profiles for a managed network.
   - How operator profiles will be assigned to operators.

10. Define event correlation rules.

    Classify the alarms for correlation, distinguishing between root cause and related alarms.

11. Define status propagation.

    Define the status propagation of object classes.

Configuration Options

This section describes the two configuration options available for HP OpenView Service Assurance for Communication Networks.

- Entry configuration
- Standard configuration

Much of the configuration details are stored in XML configuration files. You can edit these configuration files via the Telecom Configurator or the command line interface.

Entry Configuration

An entry level configuration consists of deploying the OV Telecom Extensions for OV Operations and HP OpenView Operations.

In this approach, OV Topology Server is not deployed, and, hence, the topology server and topology GUI are not available. Without the topology server, messages are not correlated to problems and root-cause/related correlation is not available. All messages received from the telecom devices are forwarded to the message browser.

Standard Configuration

A standard configuration consists of the deployment of all the HP OpenView Service Assurance for Communication Networks components. This approach is appropriate for most communication service provider deployments.

In this approach, a remote telecom subagent collects telecom messages and forwards them to the topology server for correlation into problems. Problem correlation reduces the number of inputs a Network Operation Center (NOC) operator must consider and increases the information available in each displayed message. After problems are produced from the topology server, they are injected in the OVO system. This approach provides a higher level of abstraction into OVO and off loads the OVO server because it has fewer inputs to handle.

Entry Configuration

For entry configurations, only the OV Telecom Extensions for OV Operations is deployed with OVO. These customer seeks to receive telecom-specific alarms from a new set of data collection elements not provided with the OVO product.

A minimal amount of work needs to be done in order to get the new data collectors up and running. In particular, entry configurations consist of the following steps:

- Create a new project to hold the agent configuration files.

  See “Creating a New Project” on page 104.

- Create an Agent.xml file for each agent in the managed network.

  Use the agent configuration details defined during the planning stage to create the correct number of Agent.xml files.

  See “Planning Agent Configuration” on page 124.

- 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, which are defined by equipment names and record formats.

  — Record formats, which are defined by providing two strings: beginning and ending markers associated with key alarms. Use the key alarm details during the planning stage to create the begin-end markers.

  See “Configuring Data Collectors” on page 125.

- Deploy Agent.xml files.

  See “Deploying Agent and Data Collection Data” on page 133.

- Apply Agent.xml files to OVO server.

  See “Applying Agent and Data Collection Data” on page 134.

- Modify source templates from the OVO admin GUI.

Use the key alarm configuration details from the planning stage to modify source templates, if necessary, to capture and process alarms of importance to customer and display them in the iNOC Console.

See “Modifying Basic Templates” on page 154 for instructions on how to create basic templates.

- Deploy basic templates.

  See “Deploying Templates” on page 163.

- Test your configured system up to this point. You should see alarms coming from the configured data collectors in the message browser.

Standard Configuration

This section gives a high-level description of the steps you should follow to configure a standard configuration solution. The steps listed here reflect the configuration of both OV Telecom Extensions for OV Operations, OV Topology Server.

NOTE The steps outlined in this section reflect one way to configure your managed network. You are not required to configure your network in this order.

Standard configurations are an extension of entry configurations. You should first complete the entry configuration and check that your data collectors are working and forwarding alarms to the message browser before configuring OV Topology Server.

Follow the following guidelines to configure a standard configuration.

Create a Project

Create a new project to hold all of the configuration XML files.

See “Creating a New Project” on page 104.

Configure Agent Connectivity to Network Elements

1. Create an Agent.xml file for each agent in the managed network.

   Use the agent configuration details defined during the planning stage to create the correct number of Agent.xml files.

   See “Planning Agent Configuration” on page 124.

2. 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, which are defined by equipment names and

record formats.

- Record formats, which are defined by providing two strings: beginning and ending markers associated with key alarms. Use the key alarm details during the planning stage to create the begin-end markers.

See “Configuring Data Collectors” on page 125.

3. Add any new XML files to the Project.xml file.

   Edit Project.xml to include the new agent.xml files, if necessary.

   See “Adding XML Files to Project” on page 105.

4. Deploy Agent.xml files.

   See “Deploying Agent and Data Collection Data” on page 133.

5. Apply Agent.xml files to OVO server.

   See “Applying Agent and Data Collection Data” on page 134.

6. Modify source templates from the OVO admin GUI.

   Use the key alarm configuration details from the planning stage to modify source templates, if necessary, to capture and process alarms of importance to customer and display them in the iNOC Console.

   See “Modifying Basic Templates” on page 154 for instructions on how to create basic templates.

7. Deploy basic templates.

   See “Deploying Templates” on page 163.

8. Test your configured system up to this point. You should see alarms coming from the configured data collectors in the message browser

Define Topology Model and Upper Topology

1. Edit TopoModel.xml file to define the network object model.

   Use the equipment inventory list defined during the planning stage to create:

   - Managed object classes in terms of their registration IDs, equipment types, naming attributes, and parents.
   - Connection definitions in terms of their source and destination ports.

- Naming attribute definitions in terms of their registration IDs and name type.
- Specific problems in terms of their registration IDs and affected managed object classes.

See “Entering Network Object Model Data” on page 172.

2. Edit TopoData.xml to configure the upper level topology.

   Most of the upper topology configuration is entered in the TopoData.xml file. In this file, add details about upper topology objects, including:

   - Its associated managed object class.
   - Its shortname.
   - Its relative distinguished name (RDN)
   - Its associated domain.

   See “Entering Object Instance Data” on page 185.

3. Deploy the upper topology configuration.

   After the upper level topology information is configured, deploy the configuration data and check with customers that this topology is correct before continuing on to the lower level topology configuration.

   The deploy command translates the XML configuration files listed in the Project.xml file and generates additional configuration files for the OVO server, the topology server, and each agent. These configuration files live on the OVO server until they are pushed to the appropriate file systems using the apply command.

   See “Deploying Network Object Model” on page 178 and “Deploying Object Instance Data” on page 189.

4. Apply the upper topology configuration.

   After the upper level topology information is deployed, apply the configuration data.

   From the topology server, run the following apply commands to apply the topology configuration data to the topology server:

   - `ovtopomodel.apply`
   - `ovtopodata.apply`

See “Applying Network Object Model” on page 179 and “Applying Object Instance Data” on page 190.

Configure Agent Topology Mappings

1. Edit each Agent.xml file to add the lower topology data.

   In these files, you need to define the agent topology instances, which are defined by their shortnames, associated managed object classes, relative distinguished names (RDN), domain, and related time zone.

   See “Entering Agent Topology Instance Data” on page 198.

2. Edit the Mappings.xml file to stitch the upper and lower topology together.

   Use the Mappings.xml file to define mappings relevant to how messages flow between the OVO server and the topology server.

   There are three types of mappings in this file to configure:

   - Shortname-to-FDN mapping, which makes connections between the lower physical topology objects to the virtual upper topology objects. Here, you assign a fully distinguished name to a shortname.
   - FDN-to-Node mapping, which maps telecom alarms to entries in the OVO node bank. To define this mapping, assign a node name to an FDN. This mapping blends OVO and OV Topology Server by synchronizing the telecom messages and state changes between the two servers.
   - Service-to-element mapping, which identifies which topology objects are mapped to services in Service Navigator. When a topology object state changes and it is classified as affecting a service, OV Topology Server sends a notification to OVO and changes the state of the affected services accordingly.

   See “Entering Mappings Data” on page 209.

3. Edit TopoData.xml to define roots and partitions.

   Use the system distribution details defined during the planning stage to create root instances and their associated:

   - Partition
   - Location

- Partition mask

See “Entering System Distribution Rules” on page 224.

4. Deploy the topology configuration.

   After the lower level topology information is configured, deploy the configuration data.

   The deploy command translates the XML configuration files listed in the Project.xml file and generates additional configuration files for the OVO server, the topology server, and each agent. These configuration files live on the OVO server until they are pushed to the appropriate file systems using the apply command.

   See “Deploying a Project” on page 109.

5. Apply the topology configuration.

   After the lower level topology information is deployed, apply the configuration data.

   From the topology server, run the following apply commands to apply the topology configuration data to the topology server:

   - `ovtopomodel.apply`
   - `ovtopodata.apply`

   From the OVO server, run the following apply commands to apply the agent and OVO server configuration data to the OVO server:

   - `ovoconf.apply`
   - `ovagt.apply`

   See “Applying a Project” on page 110 for more detailed instructions

Complete the Configurtation Process

1. Define event correlation rules.

   The events can be correlated using either OEMF-EC or *HP Event Correlation Services* (ECS). ECS is the default correlation tool, but either product can be configured to be the primary correlation tool.

   See Chapter 15, “Configuring Event Correlation,” on page 229.

2. Define status propagation rules.

OV Topology Server provides status propagation to propagate the change in status of the lower-level managed objects to the status of the managed objects above them.

See Chapter 16, “Configuring Status Propagation,” on page 269.

3. Define operator access management.

   Administrators need to define and assign user names for those individuals who will be monitoring the managed network. Users are then assigned profiles, which limit the type of messages that a user can handle in the message browser.

   See Chapter 17, “Configuring Operator Environments,” on page 283.

7 Create a New Project

This chapter describes how to:

- Create a new project.
- Add template XML configuration files to the project.
- Define preferences and settings for the project.
- Validate the project.
- Interpret the configuration data stored in the project.
- Distribute the configuration data to the appropriate systems.

Introduction

The concept of a project is central to the solution configuration process. All XML configuration files for a managed network solution are stored in a project directory and are listed in a *Project*.xml configuration file. Only those configuration files defined in a project can be interpreted and distributed to the OVO server and OV Topology Server.

Using Multiple Projects

For your convenience, there is no limit to the number of projects you can define. Typically, one project is defined per managed network. However, if your managed network is distributed, you could have one project defined per topology server.

Be careful with multiple projects though. When applying a second project, the previous project applied is overwritten.

Viewing a Sample Project

To view a sample project and its XML configuration files, go to `/etc/opt/OV/Telco/conf/GPRS`.

NOTE The GPRS directory is created only when the GPRS demo content is optionally installed. See *HP OpenView Service Assurance for Communication Networks Installation Guide* for instructions on how to install the GPRS demo.

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, including:

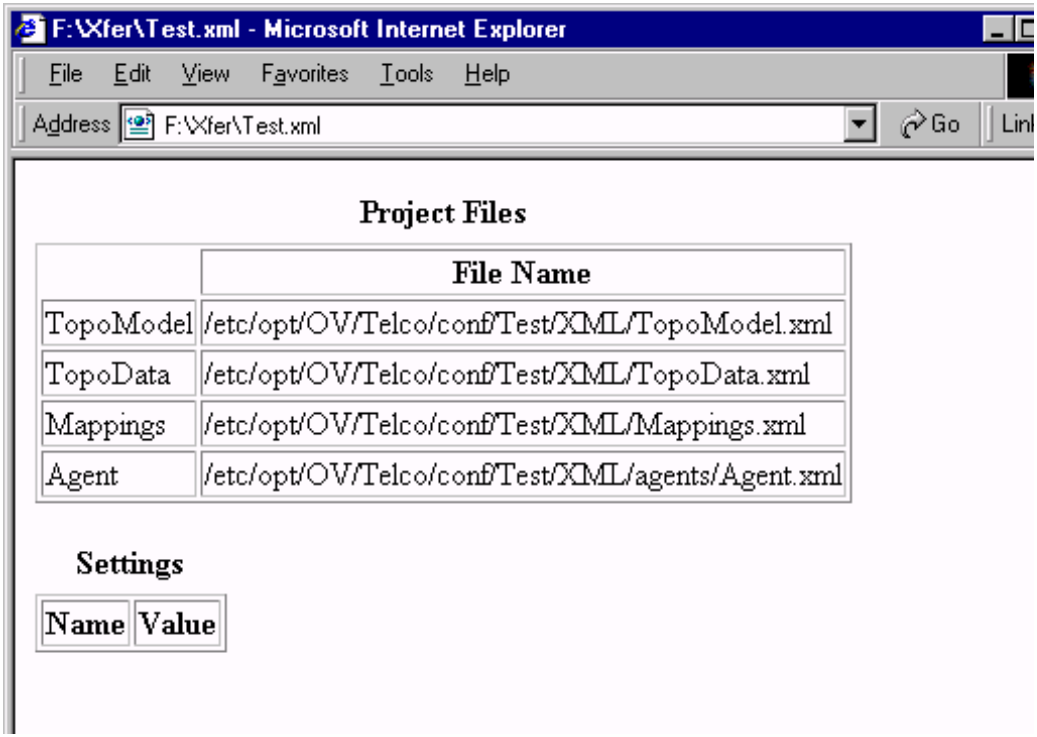
- `TopoModel.xml`, which contains the definitions to the managed object classes needed to configure a telecom network. Only one TopoModel.xml file should be defined per project.
- `TopoData.xml`, which contains the managed object instance definitions. Only one TopoData.xml file should be defined per project.
- `Mappings.xml`, which contains mapping information needed to connect OV Topology Server to OVO. Only one Mappings.xml file should be defined per project.
- An `agents` subdirectory containing one `Agent.xml` file. Multiple agent XML configuration files are allowed. One agent XML file should be defined per agent installed.

Adding XML Files to Project

Template XML configuration files are created and stored in the project directory after a project is created. It is recommended that you use the names of the template XML files, excluding the Agent.xml file.

By default, the template XML filenames are placed in the *project*.xml file. See Figure 7-1 for a sample Project.xml file that is generated after a new project is created.

Figure 7-1 Project.xml Sample File



To change the name of TopoModel.xml, TopoData.xml, or Mappings.xml, execute:

`ovcfgproject [-setTopoModel | -setTopoData | -setMappings] <`*`new_name.xml`*`>`

where *new_name* is the full path to the updated XML file. Relative paths are acceptable.

Adding XML Files to Project

Each agent in the installation must have a corresponding XML configuration file. To add an agent XML file to the project, execute:

`ovcfgproject -addAgent <filename.xml>`

where *filename* is the full path to the agent configuration file to be added. Relative paths are acceptable.

Editing Preferences and Settings

To add preferences or settings to a project, execute:

`ovcfgproject -addSetting <name> <value>`

NOTE At this time no name-value pairs are available for editing or adding to *Project*.xml.

Validating a Project

A set of DTD files that describe the syntax required for the XML configuration files are installed with this product. These DTD files cannot be edited, and are stored in `/etc/opt/OV/share/conf/Telco/dtds`.

Each XML configuration file has an associated DTD file that defines its acceptable syntax. It is important to validate your XML configuration files against the associated DTD files to make sure they adhere to the predefined structure before distributing your project configuration data.

To validate your project, execute:

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

This command validates each XML file defined in *`Project`*.xml. First, global configuration data stored in TopoModel.xml, TopoData.xml, and Mappings.xml are validated, then each agent configuration file is validated against the global configuration 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 in the sense that when the deploy command is called, all configuration files identified in the *`Project.xml`* file are again validated.

Deploying a Project

A network administrator can validate the XML configuration files and generate the target configuration files using the ovcfgdeploy command.

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

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

ovcfgdeploy converts the XML configuration data into separate configuration data files needed by OVO and OV Topology Server. See its man page for more details.

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 configuration files are stored. This directory contains copies of project configuration 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 tells whether the configuration data is new or needs only to be updated.
- backup*timestamp*, where a copy of the backup XML configuration files are stored. This directory contains copies of project configuration files that were located in /backup at the time when ovcfgdeploy is run. The timestamp reflects the time of the last backup, and not the time the last ovcfgdeploy is run.

Applying a Project

After a project is deployed, run the required apply commands on the target systems to copy and install the project configuration data to their proper destination. If data on the topology server becomes inconsistent, trying cleaning the topology data on OV Topology Server before applying the configuration files.

Removing OV Topology Server Data

Use the command `/opt/OEMF/V5.0/TopoSrv/bin/ovtoposrv.clean` to remove all topology data from the topology server system. See its man page for more details.

It is normally used after `ovcfgdeploy` has deployed the new configuration data to the topology server and an administrator has applied the data to the topology server under one of the following conditions:

- A new project is started.
- MOCs are deployed in a non-additive mode resulting in some topology data not being valid.
- A distribution rule is modified in a non-additive mode resulting in topology data not being valid.
- A new topology server built up from scratch.

Applying Configuration Files

Four apply commands are provided to pull the configuration files to the OVO server and the topology server:

- **On the topology server:**
  `ovtopomodel.apply <project>`
- **On the topology server:**
  `ovtopodata.apply <project>`
- **On the OVO server:**
  `ovoconf.apply <project>`
- **On the OVO server:**

```
ovagt.apply <project>
```

NOTE These commands must be run in the order listed above.

Table 7-1 lists each of the apply commands, what the commands do, and on which server the commands must be run.

Table 7-1 Description of Apply Commands

Apply Command	Location to Run Command	Description
`ovtopomodel.apply`	Topology server	Pulls configuration files related to the network object model from the OVO server to the topology server.
`ovtopodata.apply`	Topology server	Pulls configuration files containing the definitions of object instances from the OVO server to the topology server.
`ovoconf.apply`	OVO server	Adds modifications to the node bank and node assignments defined by the Mappings.xml file. Also notifies the telecom adapter of any new mappings.
`ovagt.apply`	OVO server	Pulls the agent configuration files from the OVO server to the appropriate file placement on the OVO server.

See their man pages for more details.

NOTE The apply commands detect if configuration files have not been updated since the last apply, and exits if no changes must be applied.

Redeploying a Project

If for some reason the configuration process does not apply as expected, follow these steps to reapply the configuration data:

- Execute the following command to generate target configuration files from the XML configuration files:

  `ovcfgdeploy -a -force <project>`

- Execute the following commands to apply the generated configuration files to the target systems:

  — **On the topology server:**
  `ovtopomodel.apply <project>`

  — **On the topology server:**
  `ovtopodata.apply <project>`

  — **On the OVO server:**
  `ovoconf.apply <project>`

  — **On the OVO server:**
  `ovagt.apply <project>`

Removing a Project

To remove a project that has been configured on a system:

- Remove the project directory from system. By default, the project directory is placed in `/etc/opt/OV/Telco/conf`.
- Edit the file `/etc/exports` and remove the line regarding your project.
- After editing `/etc/exports`, re-source the file by executing:
  `export fs -a`
  on the OVO management server.
- Clean the topology server by executing:
  `/opt/OEMF/V5.0/TopoSrv/bin/ovtoposrv.clean`
  on the topology server.
- Remove nodes generated by deploying and applying the `Mappings.xml` file from the OVO Node Bank with the OVO admin GUI.
- Additional steps from the OVO admin GUI might include:
  - — Uninstalling templates created specifically for that project.
  - — Removing message groups from the Message Group Bank created specifically for that project. For example, when the GPRS demo is deployed and applied, the `Telco-Nokia` message group is created and placed in the Message Group Bank.
  - — Removing applications from the Application Bank created specifically for that project. For example, when the GPRS demo is deployed and applied, the `GPRS Demo` application is created and placed in the Application Bank.

Using the Telecom Configurator

Start the Telecom Configurator as user `root` by typing: `ovcfgsa`

Creating a New Project

From the Telecom Configurator, click `Project:New` to create a new project directory and project file, *`Project`*.xml. By default, template XML configuration files are created in the project directory and added to the *`Project`*.xml file.

Editing Project XML Files

For more information on the commands available in the Telecom Configurator, see “Telecom Configurator” on page 63.

Validating a Project

Each XML configuration file has an associated DTD file that defines its acceptable syntax. It is important to validate your XML configuration files against the associated DTD files to make sure they adhere to the predefined structure before distributing your project configuration data.

From the Telecom Configurator, click `Project:Validate` to validate each XML file defined in *`Project`*.xml. First, global configuration data stored in TopoModel.xml, TopoData.xml, and Mappings.xml are validated, then each agent configuration file is validated against the global configuration data.

All validation errors display in the Telecom Configurator console window.

NOTE This command is optional in the sense that when the deploy command is called, all configuration files identified in the *`Project.xml`* file are again validated.

Deploying XML Files

After configuration data is entered, deploy the XML configuration files to

the designated components.

From the Telecom Configurator, click `Project:Deploy` to generate target configuration files from the configured XML files. The deploy menu command first validates the XML files within the selected project and then generates the target configuration files.

Applying XML Files

After configuration files are deployed, apply the configuration data to the designated systems. This must be done outside of the Telecom Configurator. See “Applying a Project” on page 110 for more details on applying project configuration data.

8 Configuring Data Collectors and Agents

This chapter explains the procedure to configure telecom data collectors for OVO agents. This involves configuring the data collection source details, record formats, shortnames, and lookup tables.

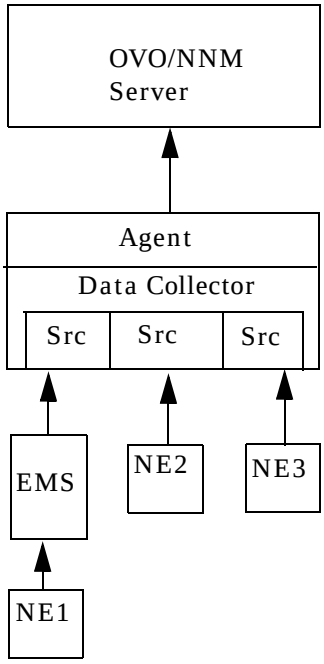
The above information can be configured using the Telecom Configurator or command line tools.

Introduction

Network elements are integrated into HP OpenView Service Assurance for Communication Networks at the agent level. Figure 8-1 illustrates the relationship among network elements, data collectors, agents, and OVO servers.

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

Figure 8-1 Data Collectors, OVO Agents, and OVO Server



DC	Data Collector
NE	Network Element
Src	DC Data Source
EMS	Element Mgmt System

Figure 8-2 depicts how the data collector might be configured to receive alarms from telecom devices.

Multiple sources can be connected to a single data collector. For each source, FIFO and TCP/IP are acceptable input connection types. Other acceptable connection types, such as SNMP, need to be configured to communicate directly with the OVO agent.

For a source acting as a TCP Server, a unique local TCP port is made available. All messages from devices which connect to this port are received by the source. The source in this case acts as a listener, reading data as it comes in.

For a source acting as a TCP Client, the source connects directly to an assigned host and port. The source reads all data to which it is connected.

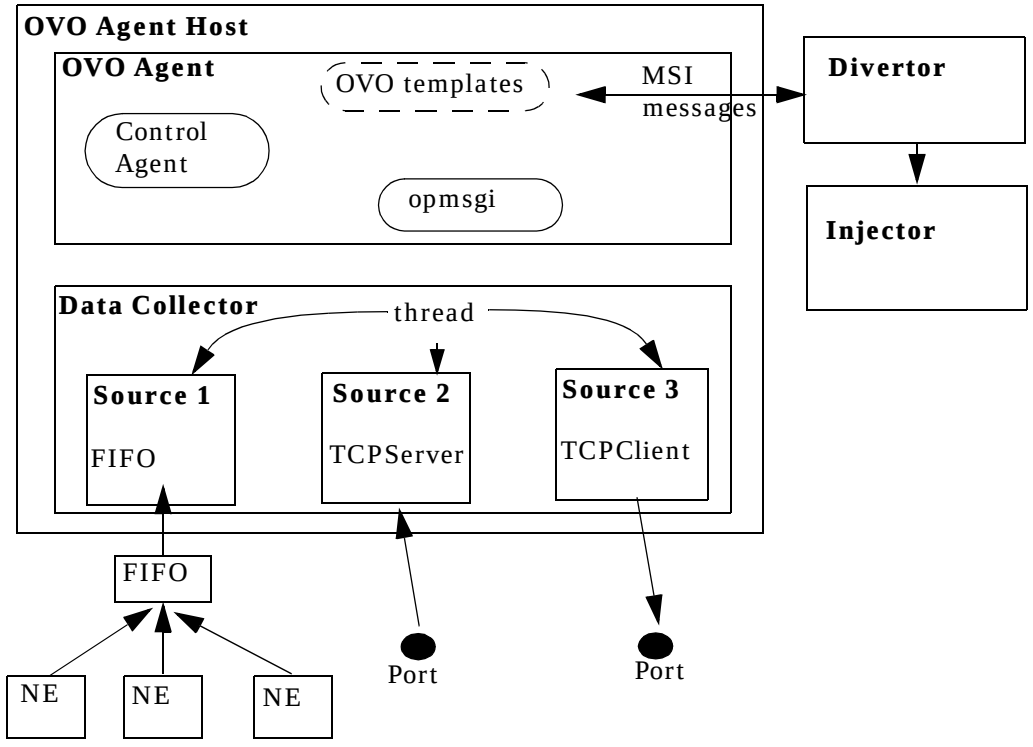
For a source acting as a FIFO connection, there is a unique file located on a local box to which network elements are connected.

NOTE

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

Consider each source acting as a thread. Thread processing consists of reading received alarms, parsing messages using a high level parser, and forwarding alarms when alarms conform to parsing rules. Otherwise, alarms are logged to files, when auditing is turned on.

Figure 8-2 Sources, Data Collectors, and Agents



Received messages from network elements are processed at the data collector level to determine if they conform to predefined patterns called **record formats**. Messages that fit the pattern are extracted and sent to the agent for further processing. The messages are then processed and presented in the iNOC Console to operators who are managing those network elements.

Figure 8-3 shows how sample alarms might be parsed by the data collector. The underlined text indicates the beginning and ending markers for the defined record formats. Thus, only three messages are forwarded to the OVO agent for further processing.

Figure 8-3 Sample Alarms

##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#

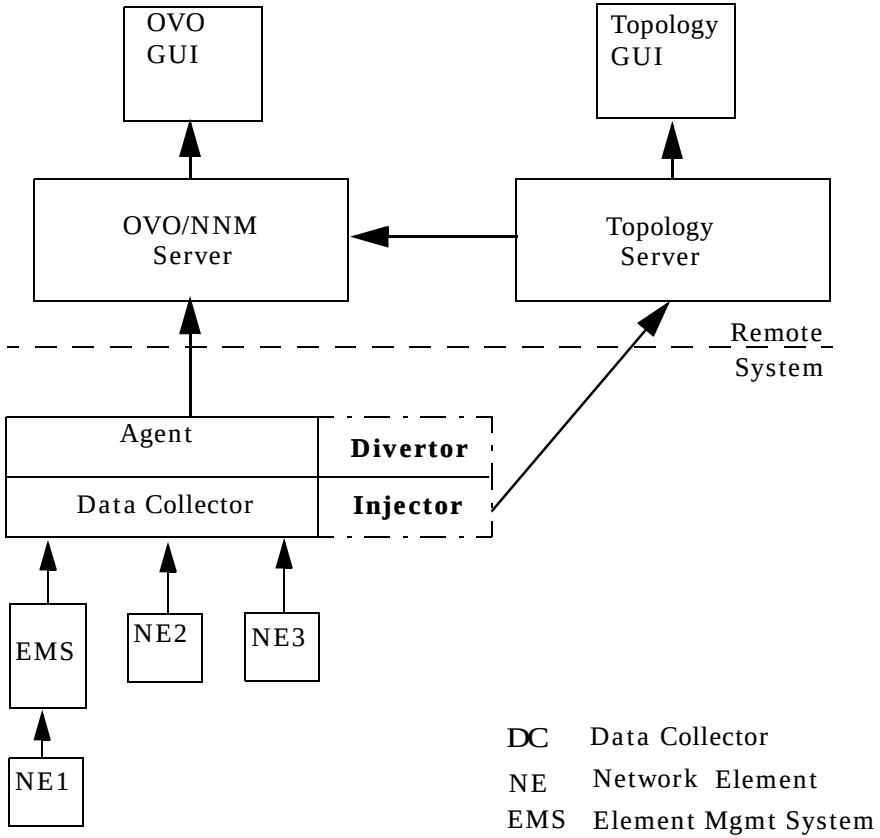
##T# 10/05/01 10:32:25 Switch1 PS=1 Temp: 78 #E#

##SC# 10/05/01 10:32:25 Switch1 System Nominal. #E#

For entry configurations, an agent receives messages from network elements, and forwards them to the OVO server and its message browser. Messages at the entry level may also be diverted through the telcom diverter. This is turned on by enabling the MSI at the agent system. Use the telecom divertor to take advatage of table lookups and timezone conversion.

For standard configurations, messages are diverted out of the OVO agent via the MSI. In turn, the messages are handled by the telecom diverter and telecom injector, converted into X.733 telecom alarms, and forwarded to the topology server. After processing the messages, the topology server forwards the messages back to the OVO server via teh telecom adaptor process, `ovtopoadaptor`. Figure 8-4 depicts a product architecture for a standard configuration.

Figure 8-4 Data Collectors and OV Topology Server



Planning Agent Configuration

To configure a system to receive messages from network elements and element management systems, you need to define the network element instances, name the data collectors, identify which data collectors are being monitored by which agents, identify how to process messages received from devices, and further configure the agents to be able to accept messages and process them as programmed.

All agent topology data is configured and stored in an agent XML file. One agent XML file is needed per agent system in the solution configuration.

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

- A list of names for the data collectors.
- Input sources to the data collectors.
- Details on whether the sources are FIFO or TCP/IP.
- 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 messages.
- Lookup tables, which convert parts of messages to a more readable format and help in the construction of Topo-Smart templates.
- A list of network elements, components, and terminiation points.

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
- Network element definitions
- Record format definitions
- Lookup table definitions
- Topology elements

Configuring Data Collectors

An *Agent*.xml consists of at least six tables you must complete:

- DCs
- Sources
- Source Details (One table for each source detail defined in the Sources table.)
- Equipment List
- Record Formats
- Lookup Tables (One for each lookup table definition needed.) (optional)

NOTE Input is mandatory in all columns of the tables unless otherwise stated.

Configuring Data Collectors

Complete the definition of the data collectors. This data forms the base of the rest of the data collection information to be configured.

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

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

Table 8-1 provides a detailed description of the parameters of `ovcfgagent`.

Table 8-1 Description of DCs Table

Parameter	Description
DC Name	The unique name for the data collector inside the agent.
Source	A list of the input source names for the data collector.
DC details	Optional. Provides a list of data collector details associated with the data collector.

Table 8-1 Description of DCs Table

Parameter	Description
XML File	The name of the agent XML configuration file for which the data collector definition is to be added.

IMPORTANT

Data collectors need to be registed with OVO in order to function properly. By default, one data collector per agent is registered for you. For most configurations, one data collector is sufficient.

When you enter more than one definition of a data collector in an agent configuration file, OVO outputs a message asking which data collector should be started. For more information on configuring more than one data collector on an agent, see the white paper on configuring multiple data collectors.

Table 8-2 provides sample definitions for three data collectors for an agent.

Table 8-2 Sample DCs Table

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

Configuring Sources

Input sources are defined in *Agent*.xml by five required attributes:

- Name
- Source Type
- Source Detail
- Status
- Equipment

To add an input source definition to *Agent*.xml, execute:

ovcfgagent -addSource *<name> <source type> <source details> <status> <equipment> xml_file*

Table 8-1 provides a detailed description of the parameters of ovcfgagent.

Table 8-3 Description of Sources Table

Parameter	Description
Name	The name for the input source.
Source Type	The type of source; enter either TCP or FIFO.
Source Details	A list of source detail names associated with this input source.
Status	Indicates whether this input source is enabled to receive alarms or not; enter Yes or No.
Equipment	A list of equipment types that can be expected to provide data to this source.
XML File	The name of the agent XML configuration file for which the data collector definition is to be added.

Table 8-4 provides sample definitions for the sources listed in the DCs table.

Table 8-4 Sample Sources Table

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

Table 8-4 Sample Sources Table

Name	SourceType	SourceDetail	Status (Enabled)	Equipment
SourceBSC	FIFO	SourceBSCDetail	Yes	NMSC MotorolaBTS

Configuring Source Details

Source details define connections details, consisting of possible name-value pairs for the source. If a source detail is not already defined, then a new one is created. Otherwise, name-value pairs are appended to the existing table.

Possible entries for *Name* are:

- FIFOName – Name of the FIFO source to read. It only applies to FIFO sources.
- TCPMode – Mode in which the data collector is to operate. The data collector can act as either a Server or a Client. When the data collector is identified as a Client, it connects to the port on the remote host specified by TCPHost. When the data collector is identified as a Server, the host connects to its port specified in TCPPort.
- TCPHost – Name of the remote host to be connected to the data collector, when TCP functionality is in client mode.
- TCPPort – The IP port that is to be used for the TCP connection.

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

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

Table 8-1 provides a detailed description of the parameters of ovcfgagent.

Table 8-5 Description of Source Details Table

Parameter	Description
Source Detail Name	Name of input source detail.

Table 8-5 Description of Source Details Table

Parameter	Description
Name	The name of the detail entry.
Value	The value of the detail entry.
XML File	The name of the agent XML configuration file for which the data collector definition is to be added.

Table 8-2 provides sample definitions for source details of a data collector.

Table 8-6 Example of Source1Detail Table

Name	Value
`TCPMode`	`Server`
`TCPPort`	`9997`

Configuring Equipment List

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

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

Table 8-1 provides a detailed description of the parameters of `ovcfgagent`.

Table 8-7 Description of Equipment List Table

Parameter	Description
Name	The name of the equipment type.
Status	Indicates whether the equipment is enabled to receive alarms; enter `Yes` or `No`.
Record Format	A list of record formats which describe the initial filters (begin/end) strings used for this piece of equipment. Note that more than one record format can be identified for each equipment type.

Table 8-7 Description of Equipment List Table

Parameter	Description
XML File	The name of the agent XML configuration file for which the data collector definition is to be added.

Table 8-6 provides sample definitions for equipment connected to a data collector.

Table 8-8 Sample Equipment List Table

Name	Status (Enabled)	Record Formats
NokiaBTS	Yes	NMSC281-DEMO
`NokiaMSC`	`Yes`	`NokBin GeneralBE`
`EricAXE`	`Yes`	`AXEBin GeneralBE`
`MotorolaBTS`	`Yes`	`GeneralBE`
`NMSC`	`Yes`	`NMSC281`

Configuring Record Formats

This section describes the information required for recognizing and classifying the messages received from an input source (the network element class that emitted the message).

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

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

Table 8-1 provides a detailed description of the parameters of `ovcfgagent`.

Table 8-9 Description of Record Formats Table

Parameter	Description
Name	The name of the record format.

Table 8-9 Description of Record Formats Table

Parameter	Description
Begin	The beginning filter string for this record format. This is the regular expression that identifies the beginning of the message packet.This information is not mandatory. However, one of the two columns--Begin or End--must have information entered.
End	The end filter string for this record format. This is the regular expression that indicates the end of the message packet. This information is mandatory if no information is entered in the Begin column.
XML File	The name of the agent XML configuration file for which the data collector definition is to be added.

Table 8-10 provides sample definitions for three record formats.

Table 8-10 Sample Record Format Table

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

Configuring Lookup Tables

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

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

Table 8-1 provides a detailed description of the parameters of `ovcfgagent`.

Table 8-11 Description of Lookup Tables

Parameter	Description
Name	Unique identifier for the lookup table.
Column Name	Identify a name for a column in the lookup table

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

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

Table 8-12 provides an example of a lookup table defined for the converting of event type strings from incoming messages to an one of the five pre-defined X.733 event types used by the topology server.

Table 8-12 Sample Lookup Table

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

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 deployment process generates target configuration files and stores them in `generated/agent` under the project directory.

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

If *Agent*.xml is valid, then the command generates the following configuration files: *agent*.xml, *agent*`.toimport`, and `delta-`*agent*`.toimport`. Next, these files should be pushed onto the agent identified by the name of the *agent*`.xml` file.

Applying Agent and Data Collection Data

To pull the agent topology configuration files stored on the OVO server to the proper servers for processing, use the `ovagt.apply` and `ovtopodata.apply` commands.

NOTE

When the MSI is enabled, the apply commands should be run in the order listed below. Otherwise, the topology server could receive messages from the agent for which it doesn't know how to process.

On the topology server, execute:

`ovtopodata.apply <project>`

On the OVO server, execute:

`ovagt.apply <project> -n <node1 node2 ...>`

This utility takes the configuration files generated by deploying *Agent*.xml and places them in the location indicated in Table 8-13. This command also restarts the subagent processes.

Table 8-13 Object Model 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
agent.toimport	Network element and topology import data	`$FMSVAR/import/ManagedObject/`
delta-*agent*.toimport	Updated topology data since the last deployment	`$FMSVAR/import/ManagedObject/`

Using the Telecom Configurator

Optionally, you may enter the agent configuration data using the Telecom Configurator.

9 Modifying Templates

This chapter explains the procedure to create, modify, assign, and install message source templates, which are used to configure an OVO agent.

Introduction

At this stage, 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 which further filter unwanted alarms from alarms of importance 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.

In order for messages to be processed properly by templates and displayed in the message browser:

- The node defined in 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.

Use table lookups to reduce the number of conditions needed for a template. Table lookups retrieve values from tables defined in an agent configuration file. Table lookups are optional and could be used to translate parameters such as error codes, location codes, and so on into more readable strings.

The telecom divertor processes the table lookups after the templates have been processed. The Message Stream Interface must be enabled for the divertor to function properly. After the divertor processes the table lookups, the messages are either sent back to the agent for further processing or forwarded to the telecom injector when OV Topology Server is configured.

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 only format and forward a message that is described in a message source template. Message source templates are configured using the OVO admin 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. When the optional OV Topology Server is installed and configured, messages matching markers defined in the telecom-specific templates are processed and forwarded to the topology 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 Telecom Smart Plug-Ins, which contain preconfigured templates for many supported devices.

See the *HP OpenView VantagePoint Operations for UNIX Concepts Guide* for more background information regarding OVO templates.

Template Administrators

Template administrators are typically responsible for creating and modifying templates. Configure template administrators in the same way as OVO operators, with the Add User window.

Template administrators use the `Message Source Templates` window to create, modify, copy, and delete templates as well as organize template groups. To prevent multiple template administrators 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
- Topo-Smart

Basic Templates

Typically, 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 in order 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, as well as OVSACN lookup tables.

Topo-Smart Templates

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.

In order for messages from particular network elements to be processed and forwarded to the topology server, additional information in the Message Text field must be entered. This additional information enables incoming messages to be translated into an X.733 message format.

Template Groups

Template groups are collections of templates or other template groups. An administrator groups templates in order to increase the performance of configuration and management tasks.

HP OpenView Service Assurance for Communication Networks delivers template groups specific to individual vendors. For example, an Ericcson template group contains a collection of templates, each of which understands alarms emitted by a particular Ericcson device.

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

A vendor template group may contain template groups as well as individual templates. Such template groups might be useful if several devices from a single vendor are somehow related. For example, power supplies could be placed in an Ericcson Power Supplies template group.

Understanding X.733 Message Format

X.733 is a format of alarm reporting. HP OpenView Service Assurance for Communication Networks transfers messages to the topology server based on X.733 message format. Alarms received by the agent are mapped to an X.733 format before being forwarded to the topology server

for further processing. User-configured templates on the agent convert messages into an X.733 format for the topology server.

OVSACN uses the following X.733 fields:

- MANAGEDOBJECTCLASS
- MANAGEDOBJECTINSTANCE
- EVENTTYPE
- EVENTTIME
- PROBABLECAUSE
- PERCEIVEDSEVERITY
- SPECIFICPROBLEM
- ADDITIONALINFORMATION

Table 9-1 describes the intended usage of these fields.

Table 9-1 X.733 Event Format Fields

Field	Description
Managed Object Class	This is the unique name that distinguishes the object class.
Managed Object Instance	This is the unique name of the managed object instance.
Event Type	This is the name of the format to be used from the standard mapping type. The event types supported are `communicationsAlarm`, `qualityofServiceAlarm`, `processingErrorAlarm`, `equipmentAlarm`, and `environmentalAlarm`.
Event Time	This is the time the alarm occurred.
Probable Cause	This is an indication of hte cause of the alarm. Acceptable values are listed in Appendix A, “List of Probable Causes,” on page 369. For other details, refer the *CCITT Document for X.733*.
Perceived Severity	This is the severity of the incoming alarm. The formats supported are `Critical`, `Major`, `Minor`, `Warning`, `Clear`, and `Indeterminate`.
Specific Problem	This is a string found in the alarm which provides additional details about the cause of the problem.

Table 9-1 X.733 Event Format Fields

Field	Description
Additional Information	This field contains any remaining text of the alarm which does not fall into one of the above fields.

The information for these fields is extracted from the text of the alarms. For example, take the following alarm:

```
ALM MSC Port08 *** 08-03-96 10:05:00 transmitFailure “faulty
tansciever” 1005 NW ALMEND\012
```

This alarm could be parsed as listed in Table 9-2.

Table 9-2 Extracting Text from Alarms to X.733 Alarm Format

Alarm Text	X.733 Field
`MSC`	MANAGED OBJECT CLASS
`Port08`	MANAGED OBJECT INSTANCE
`***`	PERCEIVED SEVERITY
`08-03-96:10:05:00`	EVENT TIME
`transmitFailure`	PROBABLE CAUSE
`“faulty transceiver”`	SPECIFIC PROBLEM
`1005 NW`	ADDITIONAL INFORMATION

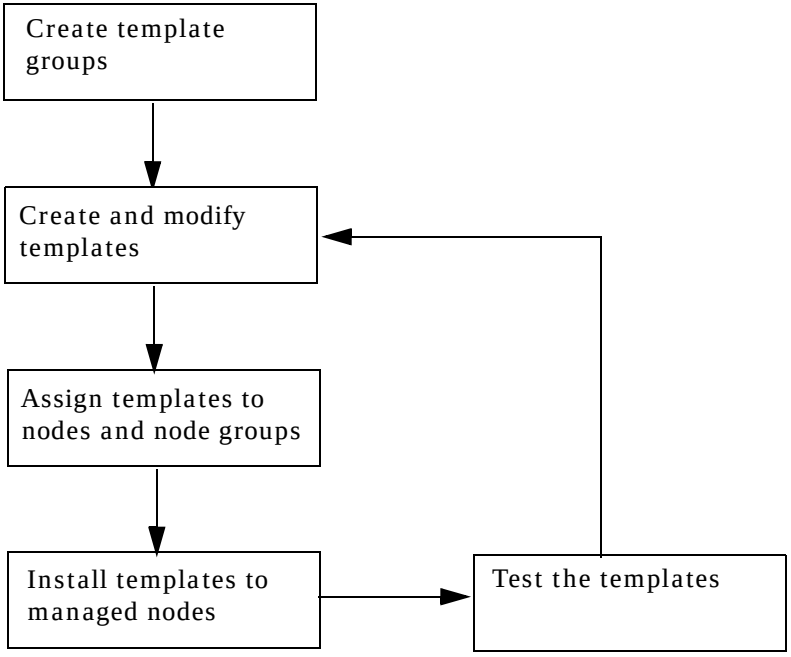
`communicationsAlarm` would also be assigned to the `EVENTTYPE` field during template configuration.

Managing Message Source Templates

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, with which attributes the message is displayed, and whether actions are to be performed.

Figure 9-1 shows the task flow of template administrators from setting up templates to installing templates to managed nodes.

Figure 9-1 Configuring Message Source Templates



Template administrators use the `Message Source Templates` window to create, modify, copy, and delete templates. The `Message Source Templates` window consists of a logical input and output section. Administrators 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.

Administrators 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, the administrator needs 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, you then have to 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.

Templates and the Template Editor

This section describes the fields associated with creating and modifying message source templates. See “Modifying Basic Templates” on page 154 for information on how to use these fields to define basic templates. See “Modifying Topo-Smart Templates” on page 155 for information on how to use these fields to construct Topo-Smart templates.

The Template Editor

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

Message source templates are configured 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.
- Message conditions and suppress conditions that match a set of attributes and define responses to received messages.
- Options, such as default message logging.

Adding a Message Source Template

To add a new message source template, click the pull down menu item [**`Add Log..`**] and select [**`Add message..`**]. An `Add ITO Interface Messages` window displays. Enter the name of the template you want to create and a description in the `Template Name` and `Description` text boxes, respectively.

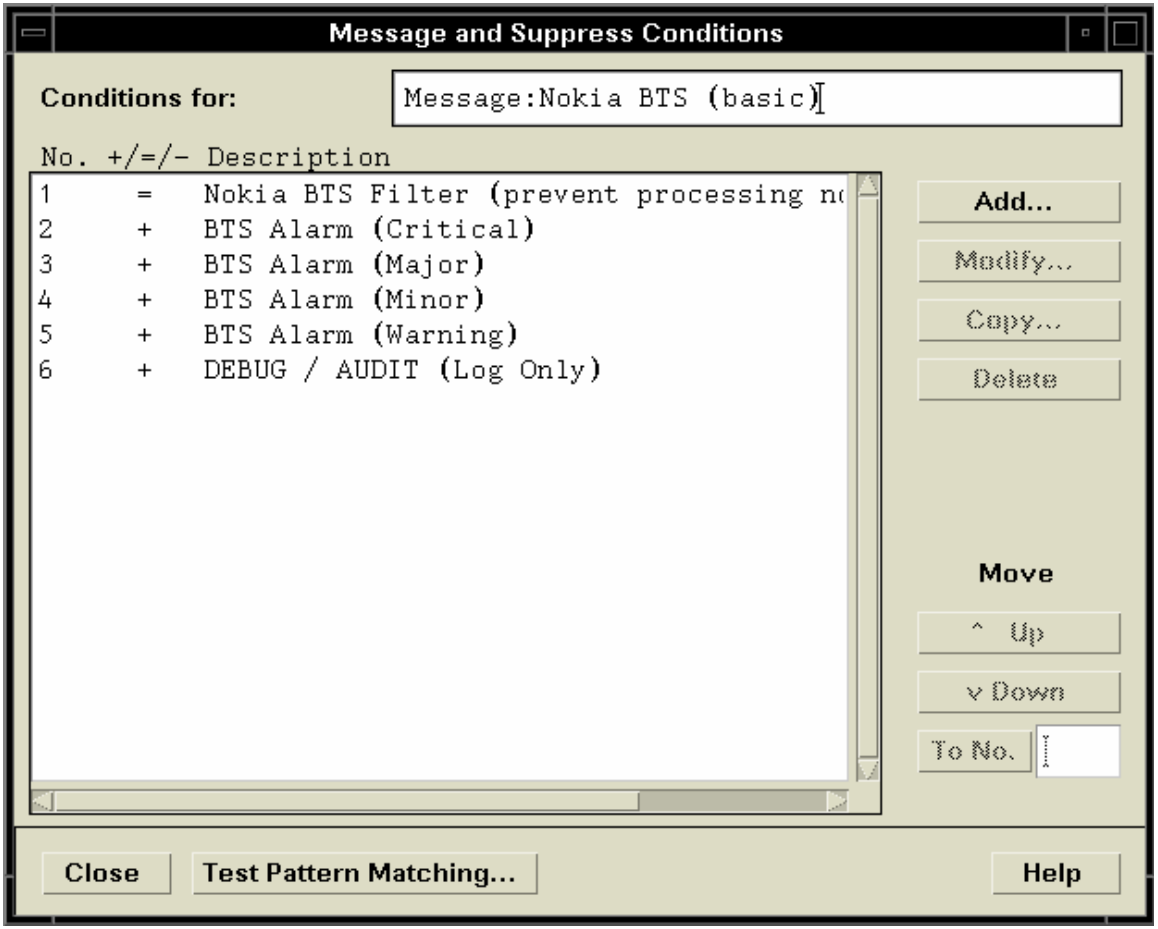
Optionally, you can assign a message group and service name to the template. Typically, assigning a service to an element can be done easily by editing the Mappings configuration file. See “Entering Mappings

Data" on page 209.

Adding Template Conditions

From the Message Source Templates window, click [**Conditions**] to open the Messages and Suppress Conditions window for a selected message source template. See Figure 9-2. In this window, define the conditions for the template, meaning the conditions for which you are going to process messages. Click [**Add**] to add a new condition definition, or double-click an existing condition definition to modify.

Figure 9-2 Template Conditions for Basic Template



The conditions in this window are processed in order listed. Thus, the

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

Modifying a Message Source Template

Figure 9-3 on page 152 shows a condition window to modify a condition definition. The window is divided into three logical parts: input, matching conditions, and output sections.

The top portion contains the input section. Messages are matched according to values stored in the fields listed in Table 9-3. Table 9-3 lists the key message fields in message source templates, their size limitations, and their intended usage.

Table 9-3 Input Fields for Templates

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 example, Oracle. 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. You define which Message Group in which the message gets placed. Note that 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. For telecom-specific devices, this value is set to the equipment type which generated the message by the data collector.	32

Table 9-3 Input Fields for Templates

Field	Description	Size
Message Text	Content and/or description of a message/ alarm. Use regular express syntax to define the Message Text. Right-click in the field to view a short list of acceptable regular expressions. For example, if you want to match messages on the string "Switch1", then define the Message Text field to be <*>Switch1<*>.	512

The matching conditions section describes how the conditions are to be treated. The options are listed in Table 9-4.

Table 9-4 Matching Conditions for Templates

Option	Description
Suppress Matched Conditions	Suppresses all messages matching condition fields in the input section. Messages are stored in a log file, rather than displaying in message browser.
Suppress Unmatched Conditions	Suppresses all messages not matching condition fields in the input section. Messages are stored in a log file, rather than displaying in message browser.
Message on Matched Condition	Forwards all messages matching condition fields in the input section to the message browser.

The lower portion 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 are unspecified, values are copied from the original message. Table 9-5 lists the key message fields in message source templates, their size limitations, and their intended usage.

Table 9-5 Output Fields of Templates

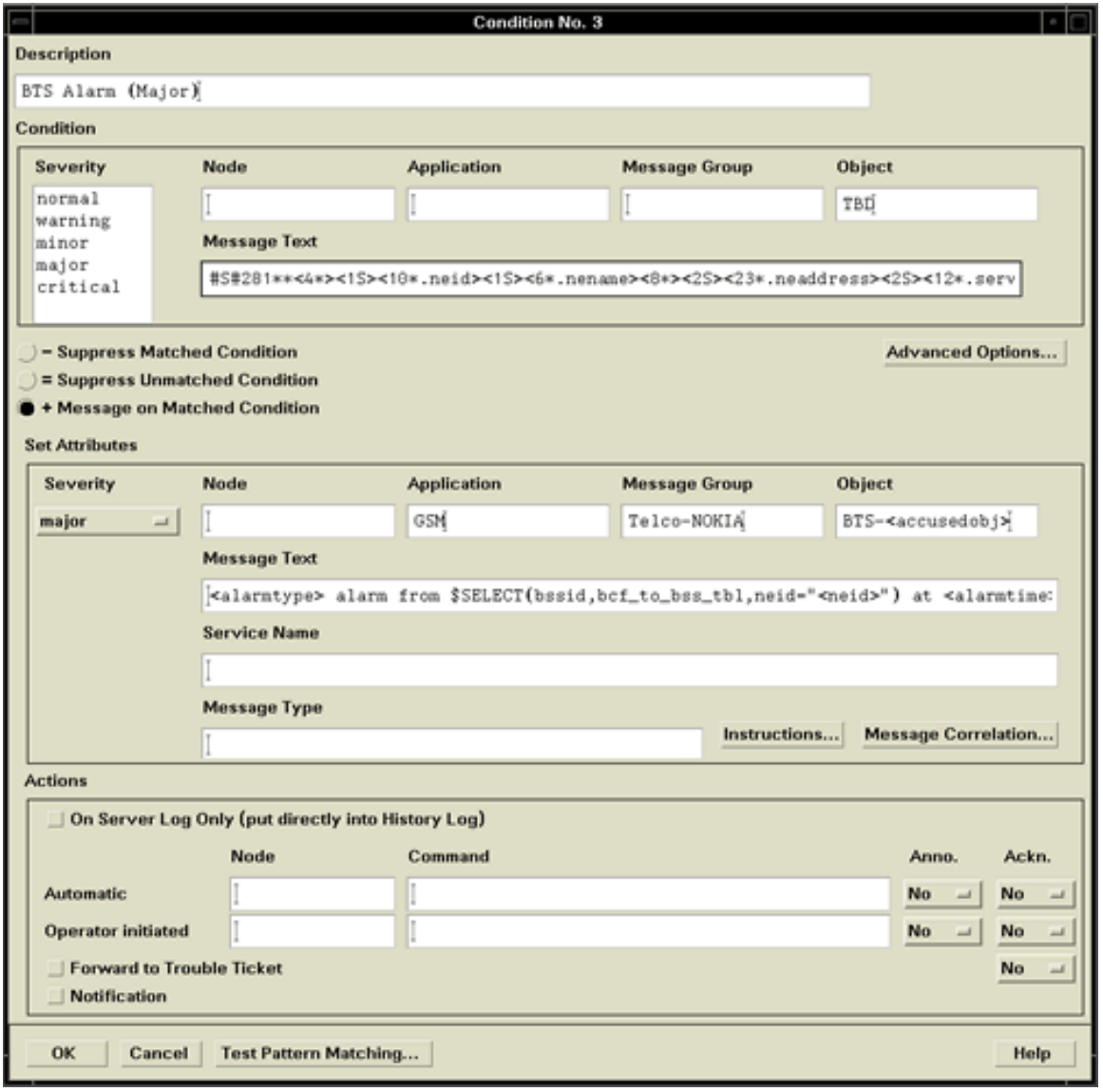
Field	Description	Size
Node	Coarse-grained identifier for the source of a message. For example, software.hp.com.	254
Application	Medium-grained identifier for a message source. For example, Oracle.	32
Message Group	Group of alarms to which a message belongs. For example, Telco-Nokia.	32
Object	Fine-grained message source identifier. For example, OpenView SID.	32
Message Text	Content and/or description of a message/ alarm. Use regular express syntax to define the Message Text. For standard configurations, additional programming is needed for this field. See “Modifying Topo-Smart Templates” on page 155.	2048
Service Name	Identifier used to associate a message/alarm with a service. Any value set here may be overridden by the telecom adaptor depending information in Mappings.xml.	254
Message Type	Identifier of a subgroup within a message group. In order for telecom-specific alarms to be forwarded to the topology server, `Telco_` is required to be entered in this field. It is recommended that `<eventtype>` follow `Telco_`. Event type can be one of five acceptable values: `communicationsAlarm`, `qualityofServiceAlarm`, `processingErrorAlarm`, `equipmentAlarm`, and `environmentalAlarm`.	32

NOTE Be aware that Node, Application, Message Group, and Object are used by administrators to create filtered views in the OVO operator GUI. Consistent use of these fields is paramount to enabling operators to

effectively monitor equipment for which they are responsible.

NOTE 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.

Figure 9-3 Modifying Basic Template Conditions



Template Fields For Telecom Devices

OVO templates identify received messages as important based on conditions defined in the templates. Received messages are evaluated based on the content contained in the Node, Application, Message Group, Object, and Message Text fields. Sometimes, templates perform pattern matching, such as in the Object and Message Text fields. For other fields, string comparison is performed. For example, the value in the field Application might be compared with the string NNM for possible matching.

Table 9-6 lists the OVO message fields populated by the telecommunications data collectors defined in your managed network to identify important received messages. These fields are the only fields that a condition can be used as input.

Table 9-6 **Template Fields Populated by Data Collectors**

Field	Value
Application	Data collector name: Source name
Object	Equipment type
Message Text	Raw alarm text, including beginning and ending markers.

NOTE The managed object class name provided in the Object field allows templates to distinguish among syntactically identical alarms from different devices. For example, two devices could produce an alarm such as `ALARM 24 42 END` where the significance of the two contained integer values is quite different depending on which device sent the alarm.

NOTE 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

Table 9-7 lists recommended values of OVO message fields for basic templates.

Table 9-7 Message Fields for 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	Nearly-raw alarm text
Message Group	Telco_*<equipment vendor>*
Service Name	No recommendation (value may be overridden by telecom adaptor)
Message Type	X.733 event type
Message Key	Reserved for use by the telecom adaptor

No value is required to be entered for a message field, excluding Message Text. If a message field is left blank, then information from the raw alarm may be inserted in the message field.

The Message Text field is the key field where received messaged are evaluated and transformed into more readable messages. To make messages more readable, lookup tables can be referenced in the Message Text field. See “Adding Table Lookup Definitions” on page 158 for details on how to incorporate them.

Modifying Topo-Smart Templates

Table 9-8 lists recommended values of OVO message fields for Topo-Smart templates.

Table 9-8 Message Fields for Topo-Smart 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	<readable text >`!:::!` <XML X.733 fields>
Message Group	`Telco_<`*equipment vendor*>
Service Name	No recommendation (value may be overridden by telecom adaptor)
Message Type	Telco_<X.733 event type>
Message Key	Reserved for use by the telecom adaptor

No value is required to be entered for a message field, excluding Message Text. If a message field is left blank, then information from the received message may be inserted in the message field.

Topo-Smart templates require similar information to basic templates. The only exception is that the Message Text field is more elaborate in order to fully translate the message into an X.733 message.

To make messages more readable, lookup tables can be referenced in the Message Text field. See “Adding Table Lookup Definitions” on page 158 for details on how to incorporate them.

To format messages for further OV Topology Server processing, X.733 fields are appended to the Message Text field, preceded by `!:::!`.

X.733 Fields to XML Elements

Table 9-9 lists the X.733 fields and the corresponding XML elements used to capture the X.733 field values. See “Understanding X.733 Message Format” on page 141 for a description of each of the X.733 fields.

Table 9-9 X.733 Attributes and Corresponding XML Elements

X.733 Attribute	XML Element
Managed Object Class	<MOC>
Managed Object Instance	<MOI>
Event Type	<ETY>
Event Time	<ETI>
Probable Cause	<PC>
Perceived Severity	<PSEV>
Specific Problem	<SP>
Additional Information	<ADDI>

These X.733 fields (and XML tags) are reserved to OVSACN, and can only be used as described in this section.

NOTE All XML elements, with the exception of <ETI> and <ADDI>, are mandatory.

Table 9-10 provides details on the specific type of information required for the XML elements in the Topo-Smart Templates.

Table 9-10 XML Elements for Topo-Smart Templates

Field	Description
<MOC>	Name of the object class.

Table 9-10 XML Elements for Topo-Smart Templates

Field	Description
<MOI>	Shortname:RDN The <MOI> element value consists of the network element shortname followed by its RDN (*shortname:RDN*). On OV Topology Server, the shortname is looked up and replaced with its associated RDN. Thus, the lower topology RDN is appended to the upper topology RDN to form the FDN for this instance on the topology server.
<ETY>	communicationsAlarm, qualityofServiceAlarm, processingErrorAlarm, equipmentAlarm, or environmentalAlarm.
<ETI>	This is the time the alarm occurred. If left blank, then the current time is inserted. See "Adding Time Arithmetic Definitions" on page 160 for more details on specifying event times.
<PC>	See Appendix A, "List of Probable Causes," on page 369 for enumerated types. For other details, refer the *CCITT Document for X.733*.
<PSEV>	Critical, Major, Minor, Warning, Clear, or Indeterminate.
<SP>	Can be set to any string.
<ADDI>	Can be set to any string.

NOTE The Template Editor displays warnings that <MOC>, <MOI>, and so on are not valid. These warnings are displayed for every Topo-Smart template you create, and can be ignored. This is due to the fact that OVO does not recognize the XML tags.

Example 9-1 Sample XML Elements Appended to Message Text Field

```
<?xml version="1.0"?>
<alarm>
<MOC> DXC-COMP </MOC>
<MOI> <device ID>:/dcx-id="DXC1"/dxccomp-id="COPY<comp" </MOI>
```

```
<ETY> equipmentAlarm </ETY>

<ETI> HHMMSS_WITH_COLON,<time>;YYYYMMDD_WITH_HYPHEN,<date>
</ETI>

<PC> communicationsSubsystemFailure </PC>

<PSEV> $SELECT(table-dxc-severity,"sev=<severity>") </PSEV>

<SP> DXC-AUD FRADTS MS <comp> COND <problem> </SP>

<ADDI> $SELECT(table-dxc-comp46-descritpion,"prb=<problem>")
</ADDI>

</alarm>
```

NOTE The text in the example on page 157 should be entered in the Message Text field in the Template Editor as one long line (no line breaks). Optionally, you can edit the templates outside of the OVO admin GUI. See the `opccfgupld` and `opccfgdwn` man pages for more information on how to edit templates outside of the OVO admin GUI.

IMPORTANT After the template is created, click [**Advanced Options**] from the `Condition` window. The `Message Condition Advanced Options` window displays. Check **`Agent MSI`** and click **`Divert Messages`**. Click [**OK**] to accept the changes.

Adding Table Lookup Definitions

Lookups retrieve values from tables defined in the telecom subagent configuration file. The processing of table lookups occurs in the telecom divertor after the template has been processed. Lookups are optional and could be used to translate parameters such as error codes, location codes, and so on into more readable strings.

Lookups add value to OVO by allowing table definitions for multiple message types. Because table lookups occur in the telecom divertor, they are not limited to messages from telecom data collectors.

HP OpenView Service Assurance for Communication Networks has defined the following `$SELECT()` operator to extend OVO message processing functionality. In the short term, this syntax is supported by

an MSI component. Eventually, the functionality of the MSI component will be merged into individual agent interceptors.

Lookups are supported in the Message Text field using syntax similar (in nature) to SQL. The following example provides a quick introduction to how the `$SELECT( )` operator is intended to work.

Consider a relational database table as shown in Table 9-11.

Table 9-11 Table Lookup Sample Table

Employee ID	Location	First Name	Last Name
1234	Cupertino	Bob	Smith
1235	Fort Collins	Bob	Smith

where one might perform the following SQL query:

```
SELECT employee_id FROM employee
WHERE (fname = ‘Bob’ AND lname = ‘Smith’ AND location =
‘Cupertino’);
```

The equivalent syntax in HP OpenView Service Assurance for Communication Networks is:

```
$SELECT(employee_id,employee,fname=”<fname>”,lname=”<lname>”,
location=”<location>”)
```

where *<fname>*, *<lname>* and *<location>* are variables extracted in the Condition Message Text field.

Table 9-12 Table Lookup Formal Definition

Parameter	Value
[1] SelectOp	`::= ‘$SELECT(‘ ColumnName ‘,’ TableName ‘,’ KeyValuePair {KeyValuePair} ‘)’;`
[2] ColumnName	`::= Alpha {AlphaNum};`
[3] TableName	`::= Alpha {AlphaNum};`
[4] KeyValuePair	`::= Key ‘=”’ Value ‘”’;`
[5] Key	`::= Alpha {AlphaNum};`

Table 9-12 **Table Lookup Formal Definition**

Parameter	Value
[6] Value	`::= ['<'] Alpha {AlphaNum} ['>'];`
[7] AlphaNum	`::= Alpha | Num;`
[8] Alpha	`::= 'a' | 'b' | … | 'y' | 'z' | 'A' | 'B' | … | 'Y' | 'Z';`
[9] Num	`::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9';`

Adding Time Arithmetic Definitions

The `<ETI>` (event time) XML element in Topo-Smart templates (if present and non-empty) must contain information in the following format:

`<Time Format>,<Time Data>;<Date Format>,<Date Data>`

where:

- `<Time Format>` and `<Date Format>` are each one of the valid terminal symbols given in Table 9-13.
- <*`Time Data`*> and <*`Date Data`*> are OVO message variables which contain data extracted from the actual message.

If the element is missing or empty, meaning, `<ETI></ETI>`, the current time is used as the event time. The time may be adjusted based on time zone information contained in the agent configuration file.

The following content for the `<ETI>` element would set the time to August 29, 2001, 6:00 pm local time (assuming no time zone adjustments in the agent configuration file and appropriate values for <*`date`*> `[08/29/01]` and <*`time`*> `[18:00:00]`):

`<ETI>HHMMSS_WITH_COLON,<time>;MMDDYY_WITH_SLASH,<date></ETI>`

Table 9-13 Time Arithmetic Formal Definition

Parameter	Value
[1] TimeDataInfo	::= ‘<ETI>‘ TimeFormat ‘,’ TimeData ‘;’ DateFormat ‘,’ ‘DateData ‘</ETI>| ‘<ETI></ETI>’| ‘<ETI/>’;
[2]TimeData	::= 2 * (2 * Num, [TimeSep]) [2 * Num];
[3]DateData	::= (2 * Num | 4 * Num) [DateSep] 2 * Num [DateSep] (2 * Num | 4 * Num);
[4]Num	::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’;
[5]Time Sep	::= ‘:’ | ‘/’;
[6]Date Sep	::= ‘:’ | ‘/’ | ‘-‘;
[7]Time Format	::= ‘HHMMSS_WITH_COLON’ | ‘HHMM_WITH_COLON’ | ‘HHMMSS_WITH_SLASH’ | ‘HHMM_WITH_SLASH’ | ‘HHMM’ | ‘HHMMSS’;

Table 9-13 Time Arithmetic Formal Definition

Parameter	Value
[8]Date Format	::= ‘DDMMYY_WITH_COLON’ | ‘DDMMYYYY_WITH_COLON’ | ‘MMDDYY_WITH_COLON’ | ‘MMDDYYYY_WITH_COLON’ | ‘YYMMDD_WITH_COLON’ | ‘YYYYMMDD_WITH_COLON’ | ‘DDMMYY_WITH_SLASH’ | ‘DDMMYYYY_WITH_SLASH’ | ‘MMDDYY_WITH_SLASH’ | ‘MMDDYYYY_WITH_SLASH’ | ‘YYMMDD_WITH_SLASH’ | ‘YYYYMMDD_WITH_SLASH’ | ‘DDMMYY_WITH_HYPHEN’ | ‘DDMMYYYY_WITH_HYPHEN’ | ‘MMDDYY_WITH_HYPHEN’ | ‘MMDDYYYY_WITH_HYPHEN’ | ‘YYMMDD_WITH_HYPHEN’ | ‘YYYYMMDD_WITH_HYPHEN’ | ‘DDMMYY’ | ‘DDMMYYYY’ | ‘MMDDYY’ | ‘MMDDYYYY’ | ‘YYMMDD’ | ‘YYYYMMDD’;

Deploying Templates

Templates must be assigned to the managed node where the OVSACN data collector is deployed. Assignment designates that the templates should go to the target system, but you must install them in order for the templates to be active. You must install templates on the target managed node even when the agent lives on the management server.

Use the OVO admin GUI to assign and install templates. Once 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. Start HP OpenView Operations.
2. Go to the `ITO Node Bank` window.
3. Select the managed node you wish to deploy the agent onto.
4. Via the menu, click **`Actions:Agents ->Assign Templates`**. This opens the `Define Configuration` window.
5. Click [**Add**]. The `Add Configuration` wondow displays.
6. Make sure that the correct managed nodes are selected in the `Node/Node Group` list box.
7. Click [**`Open Template Window`**].
8. From the `Message Source Template` window, select the template to assign.
9. From the `Add Configuration` window, click [**`Get Template Selections`**].
10. Click [**OK**] in the `Add Configuration` window.
11. Click [**OK**] in the `Define Configuration` window.

Installing Templates

1. Start HP OpenView Operations.
2. Go to the `ITO Node Bank` window.
3. Select the managed node you wish to deploy the agent onto.
4. Via the menu, click **`Actions:Agents ->Install/Update SW & Config`** . This opens the `Install/Update ITO Software and Configuration` dialog box.
5. Make sure that the correct managed nodes are listed in the `Target Nodes` window.
6. Check **`Templates`**.
7. Click **`[OK]`** to cause the templates to be deployed onto the managed node. Afterwards, you should see a message in the message browser indicating that the agent system has been updated.

Checking What Templates Are Deployed

You can view all templates installed on an agent system by executing:

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

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

Avoiding Duplicate Messages

Templates are not ordered, meaning that if messages match conditions of multiple templates, then multiple messages are displayed in the message browser. For example, if a wildcard template is assigned and installed on a system, then every message entering the agent is forwarded to the message browser. Furthermore, if additional templates are assigned and installed on a system, then 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 rules in the templates.

10 Configuring Network Object Model

With HP OpenView Service Assurance for Communication Networks, you can consolidate and integrate alarms from multi-vendor and multi-type equipment and manage them in a single, consistent environment. The equipment needs to be clearly defined and configured.

In this chapter, learn to list the managed objects in the network to be monitored, classify them, and define the containment relationship and connectivity among them.

Defining Network Object Model

All network object model definitions are stored in TopoModel.xml. This XML file contains the definitions to the managed object classes needed to configure a telecom network. In particular, TopoModel.xml defines four kinds of elements:

- Managed object classes (MOCs)
- Connections
- Specific problems
- Naming attributes

To view a sample TopoModel.xml file, go to `/etc/opt/OV/Telco/conf/GPRS/XML`.

NOTE The GPRS directory is created only when the GPRS demo content is optionally installed. See *HP OpenView Service Assurance for Communication Networks Installation Guide* for instructions on how to install the GPRS demo.

To define your network object model, you need to:

Step 1 Determine managed object classes for your managed network.

Step 2 Enter the managed object class information in TopoModel.xml.

Step 3 Validate TopoModel.xml file.

Step 4 Deploy the XML file and its containing network object model data.

Step 5 Apply the configuration data to the appropriate servers.

Determining Network Object Model

Paramount to defining a network object model that accurately captures the network to be monitored is defining the managed object classes that represent that network.

All of the telecom objects to be monitored must be classified into one of the following managed object class types:

- Network
- Network element
- Connection
- Termination point
- Component

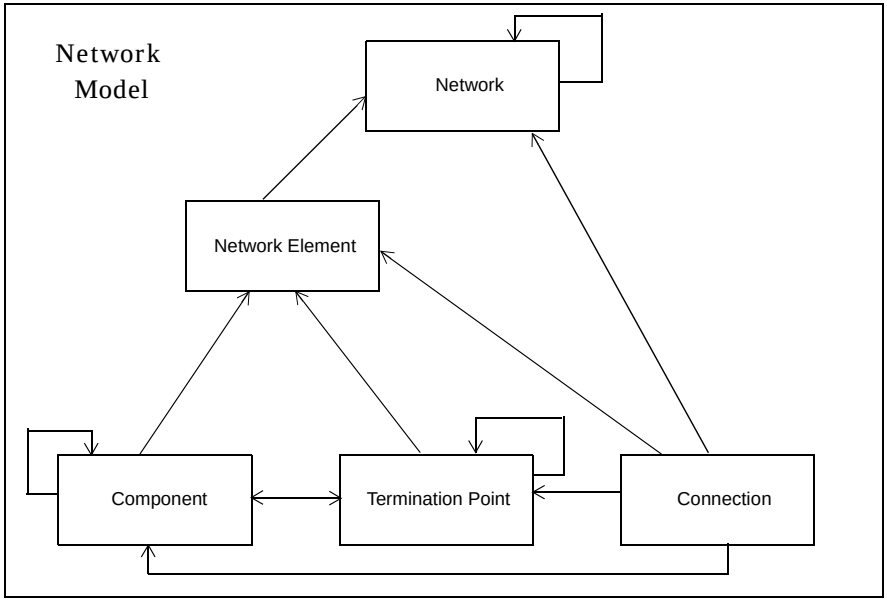
Within a monitored network, the managed object classification depends on the role of the managed object within the network.

- The **Network (NW)** class refers to a logical grouping of objects that contain other devices. In the containment tree, the network element objects that emit alarms reside under network type objects. A network type can have another Network embedded within it.
- Devices that emit alarms are classified as **Network Elements (NE)**. In the containment tree, the highest level of managed objects that emit alarms is referred to as network elements. Within the network, the component classes and connection classes reside under the network element class.
- The **Component (CP)** class can reside under the network element classes, termination point classes, or other component classes within the containment tree. Objects of this type may or may not emit alarms.
- The **Connection (CX)** class form the physical link between the end points of two network elements. The end points of the network element are termination point objects. These termination point objects identify the two network elements this Connection object connects.
- The **Termination Point (TP)** class are those components that form one end of the connection between network elements. Termination

point objects can connect two network elements, components, or termination point object. These network elements, components, or termination points could be either within the same network or in two different networks.

Figure 10-1 depicts the network model with the containment relationship.

Figure 10-1 Network Model with Containment Hierarchy



The containment relationship depicted in the above diagram is represented in tabular form in Table 10-1 on page 169. The top row lists the object types supported by OVSACN. An X below the parent managed object class indicates the object types that can be contained under them. For example, the X marks in the NW (network) column indicate that network, network element, and connection objects can be contained under a network object.

Table 10-1 Relationships Among Child and Parent MOCs

Child Managed Object Classes	NW	NE	CP	TP	CX
Network (NW)	x				
Network Element (NE)	x				

Table 10-1 Relationships Among Child and Parent MOCs

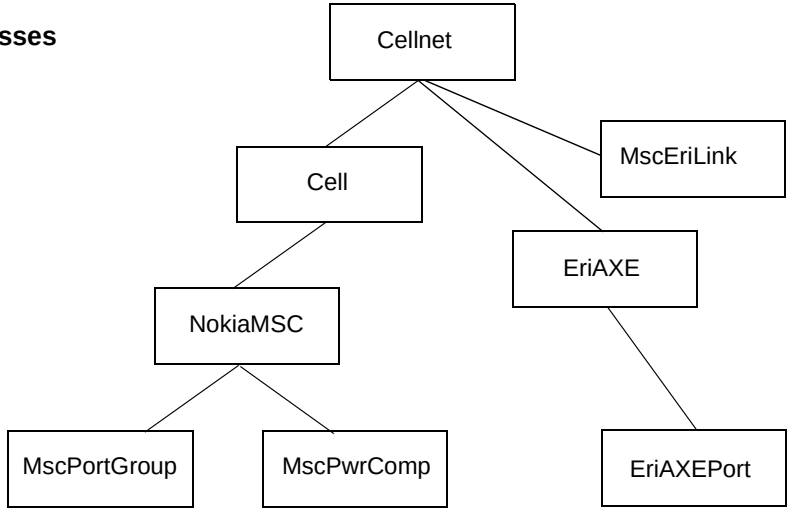
Child Managed Object Classes	NW	NE	CP	TP	CX
Component (CP)		x	x	x	
Termination Point (TP)		x	x	x	
Connection (CX)	x	x	x	x	

Using the containment relation, draw the network object model with the objects that you intend to configure. Figure 10-2 depicts a portion of a network used in a sample configuration.

Figure 10-2 Sample Network Object Model

Managed Object Classes

Cellnet (NW)
Cell (NW)
EriAXE (NE)
NokiaMSC (NE)
MscPortGroup (TP)
MscPwrComp (CP)
EriAXEPort (TP)
MscEriLink (CX)



Configuration of managed object classes supports the following features:

- **Multiple Name Binding**

  Multiple name binding allows one managed object class to be contained under multiple parent managed object classes, thus reducing the number of managed object classes to be configured.

  For example, all network elements contain power supply components. Instead of configuring individual power supply components for each

network element, you can define one “Power Supply” managed object class that can be contained under all network element managed object classes.

Note that this refers only to the managed object class. A managed object instance may have only one parent object instance.

- **Non-unique Naming Attributes**

  Each managed object class has a **naming attribute** assigned to it. Naming attributes describe what managed object classes can look like.

  Naming attributes of one managed object class can be reused for other managed object classes of the same object type. The naming attribute in managed object class definition is unique across object types, rather than being unique across object classes.

  For example, all object classes of object type Component could have the same naming attribute. Using the same naming attributes requires less time and effort to set up the component classes.

- **Recursive Containment**

  Recursive containment refers to a managed object class being contained under itself or one of its child class managed object classes.

Entering Network Object Model Data

After drawing a diagram of your managed network in terms of managed object classes, enter the information contained in the model into TopoModel.xml.

TopoModel.xml consists of four tables you must complete:

- MOCs
- Connections
- Specific Problems
- Naming Attributes

NOTE Input is mandatory in all columns of the tables unless otherwise stated.

- **MOCs**

  Complete the definition of the managed object classes. This data forms the base of the network model to be configured.

  To add a managed object class to TopoModel.xml, execute:

  `ovcfgtopomodel -addMoc` *`<MOC name> <oid> <equipment type> <alternate parents> <naming attribute>`* *`xml_file`*

  Table 10-2 provides a detailed description of the arguments of `ovcfgtopomodel`.

Table 10-2 Description of MOCs Table

Parameter	Description
Managed Object Class Name	This is the unique name that distinguishes the object class. Enter an alphanumeric name (up to 32 characters) with initial alpha character. Hyphens and underscores are allowed. The class name `root` is reserved for internal use and cannot be used.

Table 10-2 Description of MOCs Table

Parameter	Description
Object Identifier (OID)	This is the object identifier (OID) – the registration identity assigned to the object class. This is a global identifier written in dot notation. For example: `1.2.3.2.` The value begins with the number 0, 1 or 2 and ends with a digit. The numbers must not include leading zeros. The second subidentifier must be less than 40.
Equipment Type	This defines the managed object class type. Chose one of the enumerated types: **`NW`** (network), **`NE`** (network element), **`TP`** (termination point), **`CX`** (connection), or **`CP`** (component).
Naming Attribute	Each object is linked to one naming attribute that uniquely identifies an attribute for the object type. The alphanumeric name for the attribute with an initial alpha character can have a maximum of 32 characters. Hyphens and underscores are allowed. By default, if no naming attribute is assigned to an object class, then *`name`*`-id` is assigned, where name is the name of the managed object class. The naming attribute must be unique, and its details must be entered in the Naming Attributes table.
Alternate Parents	This is the object class name of the parent object class. It is the name of the object class that contains the object class being defined. The alternate parent must have a corresponding record in this table.
XML File	The name of the configuration file to which managed object classes are added.

NOTE After managed object class information are entered, make changes cautiously to them as there may be ramifications. For example, deleting a network element class causes all components, termination points, and connections defined under this network element to become invalid.

Table 10-3 provides an example of managed object class definitions based on the topology model defined in Figure 10-2 on page 170.

Table 10-3 Managed Object Classes Example

MOC Name	Object Identifier	Equipment Type (NW, NE, CP, TP, CX)	Naming Attribute	Alternate Parents
Cellnet	1.3.6.1.200.200	NW	cellnet-id	root
Cell	1.3.6.1.200.201	NW	cell-id	
NokiaMSC	1.3.6.1.200.202	NE	nokia-id	
EriAXE	1.3.6.1.200.206	NE	eriaxe-id	
MscPortGroup	1.3.6.1.200.209	TP	mscportgrp-id	
MscPwrGroup	1.3.6.1.200.210	CP	mscpwrgrp-id	NokiaMSC
EriAXEPort	1.3.6.1.200.223	TP	eriaxeport-id	
MscEriLink	1.3.6.1.200.225	CX	mscerilink-id	

- **Connections**

  Complete the definition of managed object classes defined as connections (CX). Each connection class must define a source and destination managed object class name.

  To add a connection definition to TopoModel.xml, execute:

  `ovcfgtopomodel -addCX` *`<MOC name> <source> <destination> xml_file`*

  The information to be entered is detailed in Table 10-4:

Table 10-4 Description of Termination Point Components

Parameter	Description
Connection MOC	The name of the connection object class. This managed object class must be defined in the MOCs table in TopoModel.xml.

Table 10-4 Description of Termination Point Components

Parameter	Description
Source	This indicates the termination point class from which the connection originates.
Destination	This is the end connection point. It indicates the termination point class at which the connection to the second network is made.
XML File	The name of the configuration file to which connection object classes are added.

Table 10-5 provides an example of a connection managed object class definition based on the topology model defined in Figure 10-2 on page 170.

Table 10-5 Connections Example

Connection MOC	Source	Destination
MscEriLink	MSCPrtGroup	EriAXEPort

- **Naming Attributes**

  Complete the definition of naming attributes: object identifier and name type. The object identifier is a dot notation that uniquely identifies the attribute. The name type is either a string or an integer.

  To add a naming attribute definition to TopoModel.xml, execute:

  `ovcfgtopomodel -addNA <MOC name> <oid> <name type> xml_file`

  Information in Table 10-6 is required for configuring naming attributes:

Table 10-6 Description of Naming Attribute Components

Parameter	Description
Naming Attribute	This is the name of the naming attribute as defined in the MOCs table.

Table 10-6 Description of Naming Attribute Components

Parameter	Description
OID	The object identifier of the naming attribute. This unique object identifier (up to 60 characters) is stated in dot notation. It must begin with 0, 1 or 2, and must end with an integer. It can have a maximum of 60 characters. The subidentifiers must not include leading zeros. The second subidentifier must be less than 40.
Name Type	This is the syntax in which the naming attribute is specified. the syntax is either `string` or `integer`.
XML File	The name of the configuration file to which naming attributes are added.

For example, Table 10-7 provides an example of naming attribute definitions based on the topology model defined in Table 10-3 on page 174.

Table 10-7 Naming Attributes Worksheet

Naming Attribute	Registration ID	Name Type
cellnet-id	1.3.6.1.100.200	string
cell-id	1.3.6.1.100.201	string
nokiamsc-id	1.3.6.1.100.202	string
eriaxe-id	1.3.6.1.100.206	string
mscportgrp-id	1.3.6.1.200.209	string
mscpwrgrp-id	1.3.6.1.200.210	string
eriaxeport-id	1.3.6.1.200.223	string
mscerilink-id	1.3.6.1.200.225	string

- **Specific Problems**

  Complete the definition of the specific problem elements. Each specific problem is defined by its object identifier.

  To add a specific problem definition to TopoModel.xml, execute:

```
ovcfgtopomodel -addSP <MOC name> <oid> <Affected MOCs>
xml_file
```

Table 10-8 Description of Specific Problems

Parameter	Description
Specific Problem	*Applicable only to network element, termination point, and component classes.* This is the name of the specific problem the object class. More than one object can be linked to the same specific problem. It can have a maximum of 64 characters.
Object Identifier	The object identifier (OID) of the specific problem named. This dot notation can have a maximum of 60 characters. The sub-identifiers must not have leading zeros and the second sub-identifier must be less than 40.
Affected MOCs	An optional attribute that consists of list of managed object class names affected by this specific problem definition. Each MOC is separated by a whitespace.
XML File	The name of the configuration file to which specific problems are added.

Deploying Network Object Model

After entering the network object model data in TopoModel.xml, validate and deploy the XML file by executing `ovcfgdeploy`. The validation process checks to see if the content you entered is consistent and valid against TopoModel.dtd. The deployment process generates target configuration files and stores them in `generated/` under the project directory.

Execute: `ovcfgdeploy -topomodel <project>`

This command checks the consistency of managed object class data, makes sure all managed object classes in connections and specific problems are defined, and checks that there is a naming attribute defined for each managed object class.

If TopoModel.xml is valid, then the command generates the following configuration files: `fmpmap.conf`, `ocinfo.conf`, and `oid.conf`. Next, these files should be pushed onto the topology server for processing by OV Topology Server.

Applying Network Object Model

To pull the topology model configuration files stored on the OVO server to the topology server for processing by the topology server, use the `ovtopomodel.apply` command.

On the topology server, execute:

`ovtopomodel.apply` *<project>*

Answer `Yes` when prompted whether to synchronize the system distribution rules and update your configuration data on the topology server.

This utility takes the configuration files generated by deploying TopoModel.xml and places them in the location indicated in Table 10-9.

Table 10-9 Object Model Configuration Files

File Name	Content	Destination
`fmpmap.conf`	managed object classes, connection definitions, naming attributes	`$FMSETC/share/newconf`
`oid.conf`	managed object classes, connection definitions, specific problem definitions	`$FMSETC/share/newconf`
`ocinfo.map`	specific problem definitions	`$FMSETC/share/newconf`

NOTE If the object model has significantly changed from the previous deployed object model, then you must clean the topology server before applying the configuraiton data by executing `ovtoposrv.clean`. For example, adding a new managed object class does not warrant the topology server to be cleaned; however, changing a naming attribute or modifying a managed object class does. For more information on cleaning the topology server, see “Removing OV Topology Server Data” on page 110.

Using the Telecom Configurator

Optionally, you may enter the network object model data using the Telecom Configurator. See the `ovcfgsa(1m)` reference page for more information.

11 Configuring Managed Object Instances

In this chapter, learn to define managed object instances within a network to be monitored.

This chapter describes the utilities used and procedures to:

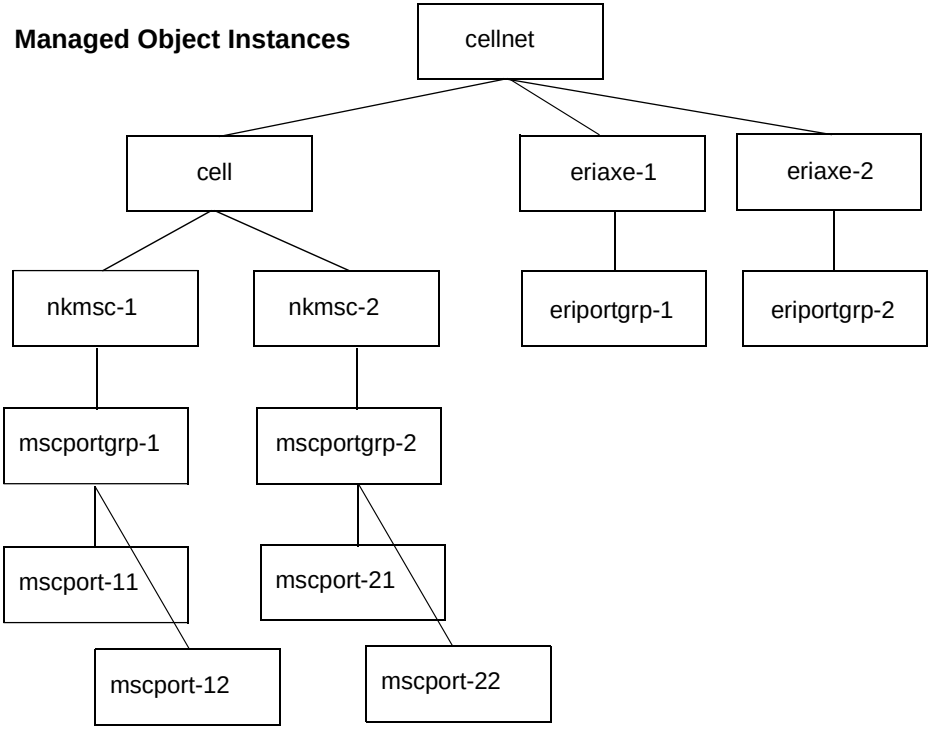
- Create managed object instances in the topology database.
- Create link object maps.

Drawing the Containment Tree

The entire network domain to be managed by OV Topology Server can be structured in a manner similar to a directory structure on a computer. Each directory, subdirectory, and containing file represents a managed object instance that is to be monitored.

Given your network object model defined in “Defining Network Object Model” on page 167, the next step is to draw a containment tree for the managed network.

Figure 11-1 Sample Containment Tree



Defining Network Object Instances

All managed object instance definitions are stored in TopoData.xml. This XML file contains the definitions of the managed object instances needed to monitor an entire network. In particular, TopoData.xml requires the following information to be entered for each managed object instance:

- Managed object class name (MOC)
- Relative distinguished name (RDN)
- Short name
- Domain name (optional)

To view a sample TopoData.xml file, go to `/etc/opt/OV/Telco/conf/GPRS/XML`.

NOTE The GPRS directory is created only when the GPRS demo content is optionally installed. See *HP OpenView Service Assurance for Communication Networks Installation Guide* for instructions on how to install the GPRS demo.

To define your managed object instances, you need to:

Step 1 Retrieve the list of managed object class names defined for your managed network as this is needed to define managed object instances.

Step 2 Retrieve the drawing of the containment tree as this provides a complete list of all the managed object instances to be defined.

Step 3 Enter the managed object instance information in TopoData.xml.

Step 4 Validate TopoData.xml file.

Step 5 Deploy the XML file and its containing network object model data.

Step 6 Apply the configuration data to the appropriate servers.

Entering Object Instance Data

After drawing a containment tree of your managed network in terms of managed object instances, enter the information contained in the diagram into TopoData.xml.

TopoData.xml consists of four columns you must complete for each managed object instance:

- Managed object class name (MOC)
- Relative distinguished name (RDN)
- Short name
- Domain name (optional)

NOTE Input is mandatory in all columns of the table unless otherwise stated.

Two types of upper topology managed object instances can be defined in TopoData.xml: Network and connection upper topology object instances. Instructions to define each type of managed object instance are described next.

- **Network Object Instances**

  Complete the definition of the network upper topology object instances.

  To add a network managed object instance to TopoData.xml, execute:

  `ovcfgtopodata -addInst` *`<parent FDN> <MOC name> <RDN> <shortname> [domain] xml_file`*

  Table 11-1 provides a detailed description of the arguments to `ovcfgtopodata`.

Table 11-1 Description of Upper Topology Table

Parameter	Description
Parent FDN	The FDN of the parent object instance that contains this managed object instance.

Table 11-1 Description of Upper Topology Table

Parameter	Description
Managed Object Class Name	This is the name of the managed object class type associated with this managed object instance.
RDN	The relative distinguished name of the managed object instance.
Shortname	An assigned alpha identifier for the managed object instance. This name is does not have to be unique.
Domain	The domain name for this instance and all its descendants.
XML File	The name of the configuration file to which managed object instances are added.

Table 11-2 provides an example of managed object instance definitions based on the containment tree defined in Figure 11-1 on page 183.

Table 11-2 Network Object Instances Example

Parent FDN	MOC Name	RDN	Shortname	Domain
	Cellnet	/cellnet-id=”Wireless1”	Wireless1	
/cellnet-id=”Wireless1”	Cell	/cell-id=”Wireless2”	Wireless2	
/cellnet-id=”Wireless1”/cell-id=”Wireless2”	NokiaMSC	/nokiamsc-id=”Nokia”	Nokia	
/cellnet-id=”Wireless1”	EriAXE	/eriaxe-id=”Erikson”	Erikson	
/cellnet-id=”Wireless1”/cell-id=”Wireless2”/nokiamsc-id=”Nokia”	MscPortGroup	/mscportgroup-id=”MSCPort”	MSCPort	

Table 11-2 Network Object Instances Example

Parent FDN	MOC Name	RDN	Shortname	Domain
/cellnet-id=”Wireless1”/cell-id=”Wireless2”/nokiamsc-id=”Nokia”	MscPwrGroup	/mscpwrgroup-id=”MSCPwr”	MSCPwr	
/cellnet-id=”Wireless1”/eriaxe-id=”Erikson”	EriAXEPort	/eriaxeport-id=”EriPort”	EriPort	

- **Connection Object Instances**

  Complete the definition of managed object instances defined as connection instances. Connection instances are slightly different from other managed object instances. Connections must also identify source and destination points for the connections.

  To add a connection instance definition to TopoData.xml, execute:

  `ovcfgtopodata -addCX <parent FDN> <MOC name> <RDN> <shortname> <from_FDN> <to_FDN> [domain] xml_file`

  The information to be entered is detailed in Table 11-3:

Table 11-3 Description of Upper Topology Table

Parameter	Description
Parent FDN	The FDN of the parent object instance that contains this managed object instance.
Managed Object Class Name	This is the name of the managed object class type associated with this managed object instance.
RDN	The relative distinguished name of the managed object instance.
Shortname	An assigned alpha identifier for the managed object instance. This name is does not have to be unique.
From FDN	The fully distinguished name of the termination point instance from which the connection originates.

Table 11-3 Description of Upper Topology Table

Parameter	Description
To FDN	The fully distinguished name of the termination point instance at which the connection to the second object is made.
Domain	The domain name for this instance and all its descendants.
XML File	The name of the configuration file to which connection object instances are added.

Table 11-4 provides an example of connection object instance definitions based on the containment tree defined in Figure 11-1 on page 183.

Table 11-4 Connection Object Instances Example

Parent FDN	MOC Name	RDN	Shortname	From FDN	To FDN	Domain
/cellnet-id=” Wireless1”	MscEriLink	/mscerilink -id=”MscE riLink”	MscEriLink	/cellnet-id= ”Wireless1” /cell-id=”Wi reless2”/nok iamsc-id=” Nokia”/msc portgroup-i d=”MSCPor t”	/cellnet-id= ”Wireless1” /eriaxe-id=” Erikson”/eri axeport-id= ”EriPort”	

Deploying Object Instance Data

After entering the object instance data in TopoData.xml, validate and deploy the XML file. The validation process checks to see if the content you entered is consistent and valid against TopoData.dtd. The deployment process generates target configuration files and stores them in `generated/` under the project directory.

Execute: `ovcfgdeploy -topodata <project>`

This command first checks the consistency of managed object instance data. The validation checks to see whether:

- Configuration files generated from a previous deployment have been applied on the topology server.
- TopoData.xml has any updates since the last deployment.
- TopoData.xml contains valid managed object class names.

NOTE No validation is performed on connection object instances to see if the two termination points are defined on the agents.

If TopoData.xml is valid, then the command generates the upper topology data files to be applied to the topology server.

The configuration files that are generated are: `upper.toimport`, `links-upper.toimport`, `delta-upper.toimport`, and `delta-links-upper.toimport`. Next, these files should be pushed onto the topology server for processing by OV Topology Server.

Applying Object Instance Data

To push the upper topology configuration data stored on the OVO server to the topology server for processing by OV Topology Server, use the `ovtopodata.apply` command.

On the topology server, execute:

`ovtopodata.apply` *<project>*

This utility takes the configuration files generated by deploying TopoData.xml and places them in `$FMSVAR/import/ManagedObject`. Table 11-5 describes the contents of these files.

Table 11-5 **Object Instance Configuration Files**

File Name	Content
`upper.toimport`	All network object instances in upper level.
`links-upper.toimport`	All connection object instances in upper level.
`delta-upper.toimport`	All network object instances updated since last deployment of TopoData.xml file.
`delta-links-upper.toimport`	All connection object instances updated since last deployment of TopoData.xml file.

The `ovtopodata.apply` command also pulls in information from the Mappings.xml file and all import files generated by the `ovcfgdeploy` command. Table 11-6 list the possible files to be pulled in by `ovtopodata.apply`.

See the `ovtopodata.apply` man page for more details.

Table 11-6 Other Configuration Files Read

File Name	Content	Destination
`Mappings.xml`	Shortname to upper level topology mappings.	Mappings.xml
`<agent>.toimport`	Lower level network element topology instances.	`$FMSVAR/import/ManagedObject/`
`delta-<agent>.toimport`	All network element, lower topology instances updated since last apply.	`$FMSVAR/import/ManagedObject/`

NOTE In general, the solution developer should get the the network object model and the upper topology instances to a stable point before proceeding to add the lower topology instances. This reduces the time to apply all of the configuration data when an upper topology instance or managed object class is modified.

Using the Telecom Configurator

Optionally, you may enter the network object model data using the Telecom Configurator.

12 Configuring Agent Topology

This chapter explains the procedure to configure lower topology object instances.

The above information can be configured using the Telecom Configurator or command line tools, which are described in this chapter.

Planning Agent Topology Configuration

An agent is configured with a set of network elements for which it is responsible. The network element information is stored in an *Agent*.xml, where the name of the file is the name associated with the agent, such as a hostname. One *Agent*.xml file is needed for each agent in the solution configuration.

Agent.xml files contain object instances for each network element and intra-device topology that is modeled in the topology server. The data collection object instances represent the lower topology of the TMN containment tree.

Defining Network Element Instances

All network element instance definitions (lower topology) are stored in *Agent*.xml. This XML file contains the definitions to the network element instances needed to monitor telecom devices. In particular, *Agent*.xml requires the following information to be entered for each network element instance:

- Network element shortname
- Managed object class (MOC)
- Relative distinguished name (RDN)
- Domain (optional)
- Time zone (optional)

To view a sample *Agent*.xml file, go to `/etc/opt/OV/Telco/conf/GPRS/XML/agents`.

NOTE The GPRS directory is created only when the GPRS demo content is optionally installed. See *HP OpenView Service Assurance for Communication Networks Installation Guide* for instructions on how to install the GPRS demo.

To define your network element instances, you need to:

Step 1 Retrieve the list of managed object class names defined for your managed network as this is needed to define network element instances.

Step 2 Retrieve the drawing of the containment tree as this provides a complete list of all the network element instances to be defined.

Step 3 Enter the network element instance information in *Agent*.xml.

Step 4 Validate *Agent*.xml file.

Step 5 Deploy the XML file and its containing network element instance data.

Step 6 Apply the configuration data to the appropriate servers.

Entering Agent Topology Instance Data

Examine the drawing of your containment tree for your managed network, and identify the lower topology of your TMN containment tree. Enter the instance information contained in the diagram into *Agent*.xml.

Agent.xml consists of and Agent Topology table with five columns you must complete for each managed object instance:

- Managed object class name (MOC)
- Relative distinguished name (RDN)
- Short name
- Domain name
- Time zone (optional)

NOTE Input is mandatory in all columns of the table unless otherwise stated.

- **Network Element Instances**

  Complete the definition of the network element instances.

  To add a network element object instance to *Agent*.xml, execute:

  `ovcfgagent -addNE <RDN> <MOC name> <shortname> <domain> [<timezone>] <xml_file>`

  This command adds a new network element definition into the agent topology.

  Table 12-1 provides a detailed description of the arguments of `ovcfgagent`.

Table 12-1 Parameters for Network Element Instances

Parameter	Description
RDN	The relative distinguished name of the managed object instance.

Table 12-1 Parameters for Network Element Instances

Parameter	Description
Managed Object Class Name	This is the name of the managed object class type associated with this managed object instance.
Shortname	An assigned alpha identifier for the network element instance. This name has to be unique among all network elements across the managed network. Validation of uniqueness is performed within an agent, but not across agents.
Domain	You may add a new domain name for this network element. By default, the network element inherits all of the domains from its parent. If no new domain name is to be added, use "" in this column.
Time zone	The time zone in which the network element is physically located.
XML file	The name of the agent XML file where network elements instances are to be added.

NOTE The shortname assigned to this network element is important to remember. In the Mappings.xml file, the shortname is mapped to the fully distinguished name (FDN) of the upper topology object for this network element to create the complete FDN for the network element.

- **Managed Object Instances**

  Once network element instances are defined, you should then define all component and terminal point object instances in the lower topology containment tree. To add a managed object instance to *Agent*.xml, execute:

  ovcfgagent -addMOI *<parent RDN> <RDN> <MOC> [<shortname>] <xml_file>*

  Table 12-2 provides a detailed description of the arguments of

ovcfgagent.

Table 12-2 Parameters for Managed Object Instances

Parameter	Description
Parent RDN	The RDN of the parent managed object instance that contains this component object instance. The RDN path starts with the RDN of the network element in this agent for this component and ends with the RDN of the parent component, if any.
RDN	The relative distinguished name for this component object instance. The RDNs must be unique among siblings.
Managed Object Class Name	This is the name of the managed object class type associated with this component object instance. The MOC type for this instance must either be a component or a terminal point.
Shortname	An assigned alpha identifier for the managed object instance. If a shortname is not entered then one is assigned the value of the naming attribute from the RDN.
XML file	The name of the agent XML file where managed object instances are to be added.

- Connection Object Instances

  Complete the definition of managed object instances defined as connection instances. Connection instances are slightly different from other managed object instances in that connections must also identify source and destination points for the connections.

  To add a connection instance definition to *Agent*.xml, execute:

  ```
  ovcfgagent -addCx <parent RDN> <RDN> <from> <to>
  [<shortname>] <xml_file>
  ```

  The information to be entered is detailed in Table 12-3:

Table 12-3 Parameters for Connection Instances

Parameter	Description
Parent RDN	The RDN of the parent object instance that contains this managed object instance. The RDN path starts with the RDN of the network element and ends with the RDN of the parent object instance.

Table 12-3 Parameters for Connection Instances

Parameter	Description
RDN	The relative distinguished name of the managed object instance. The RDN must be unique among sibling instances.
Managed Object Class Name	This is the name of the connection managed object class associated with this managed object instance.
From	The relative distinguished name of the termination point instance from which the connection originates.
To	The relative distinguished name of the termination point instance at which the connection to the second object is made. The RDNs of the two connection points must be different.
Shortname	An assigned alpha identifier for the managed object instance. If a shortname is not entered then one is assigned the value of the naming attribute from the RDN.
XML file	The name of the agent XML file where managed object instances are to be added.

NOTE For those connections whose parents are network instances, use `ovcfgtopodata` to create the connection.

Deploying Agent Object Instance Data

After entering the agent topology 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 deployment process generates target configuration files and stores them in `/generated/agents` under the project directory.

Execute: `ovcfgdeploy -agent <project>`

This command first checks the consistency of agent object instance data. The validation checks to see whether:

- Configuration files generated from a previous deployment have been applied on the topology server.
- *Agent*.xml has any updates since the last deployment.
- *Agent*.xml contains valid managed object class names.

NOTE No validation is performed on the managed object class type of the *to* and *from* parameters for connection instances.

If *Agent*.xml is valid, then the command generates the lower topology data files to be applied to the topology server.

The configuration files that are generated are: *agent*.xml, *agent*`.toimport`, and `delta-`*agent*`.toimport`. Next, these files should be pushed onto the topology server for processing by OV Topology Server.

Applying Agent and Object Instance Data

To push the agent topology configuration files stored on the OVO server to the listed managed nodes, use the `ovagt.apply` and `ovtopodata.apply` commands.

On the topology server, execute:

`ovtopodata.apply <project>`

On the OVO server, execute:

`ovagt.apply <project> -n <node1 node2 ...>`

This utility takes the configuration files generated by deploying *Agent*.xml and places them in the location indicated in Table 12-4. This command also restarts the subagent processes.

Table 12-4 Agent Configuration Files

File Name	Content	Destination
agent.xml	Data collectors, sources, source details, equipment, record formats, adn lookup table data.	/var/opt/OV/conf/Telco/OVSAAgentCfg.xml on the managed node
agent.toimport	Network element and topology import data.	$FMSVAR/import/ManagedObject/
delta-*agent*.toimport	Updated topology data since the last deployment.	$FMSVAR/import/ManagedObject/

Using the Telecom Configurator

Optionally, you may enter the lower topology instance data using the Telecom Configurator.

13 Configuring Mappings

In this chapter, learn to define mappings.

This chapter describes the utilities used and procedures to:

- Add service-to-element mapping definitions.
- Add shortname-to-FDN mapping definitions.
- Add FDN-to-node mapping definitions.

Defining Mappings

All mapping definitions are stored in Mappings.xml. This XML file contains the definitions of the mappings to stitch the upper topology to the lower topology, map services to topology elements, and map topology objects to OVO nodes. In particular, Mappings.xml requires the following information to be entered:

- Service-to-element mapping
- Shortname-to-FDN mapping
- FDN-to-Node mapping

To view a sample Mappings.xml file, go to `/etc/opt/OV/Telco/conf/GPRS/XML`.

NOTE The GPRS directory is created only when the GPRS demo content is optionally installed. See *HP OpenView Service Assurance for Communication Networks Installation Guide* for instructions on how to install the GPRS demo.

Service-to-Element Mappings

Service-to-element mapping tables identify which topology objects are mapped to services in Service Navigator. Many customers define complex service definitions that contain IP network components, system and application components, and telecom topology components.

The telecom adaptor reads this mappings file and when a topology object changes state, and sends a message to Service Navigator indicating a service has been affected.

Topology objects can map to multiple services and vice versa.

Shortname-to-FDN Mappings

Shortname-to-FDN mapping stitches the physical topology objects to the TMN containment tree. In OVSACN, the TMN tree is split into the lower physical part (defined in the agent configuration files) and the virtual

upper part (defined in the TopoData.xml configuration file). This mapping stitches the two halves together in order to create the full TMN containment tree.

In the agent configuration file, network elements are defined by a shortname, which is used as the label for the object on the topology map. OVSACN translates this shortname to the RDN of that network element. In the TopoData.xml configuration file, each upper topology object of the TMN containment tree is assigned its RDN. The Shortname-to-FDN mapping combines the RDN of the upper topology object and the RDN of the lower topology object to create the FDN of the network elements in the TMN containment tree.

FDN-to-Node Mappings

The FDN-to-Node mapping maps telecom alarms to entries in the OVO node bank. Telecom problems and state changes are synchronized between OVO and OV Topology Server.

In OVO, the node bank contains a group of nodes to which messages are forwarded. Operators are then assigned certain nodes in the network to manage. In order for operators monitoring OVO to view telecom messages, the topology server posts telecom messages to specific nodes. The node assignment is handled in the FDN-to-Node mappings table.

Planning Mappings

To define your mapping definitions, you need to:

Step 1 Retrieve the list of managed object class names defined for your managed network as this is needed in the mappings tables.

Step 2 Retrieve the drawing of the containment tree as this provides a complete list of all the managed object instances. Also, FDNs and shortnames are used in mappings tables.

Step 3 Enter the mapping definitions in Mappings.xml.

Step 4 Validate Mappings.xml file.

Step 5 Deploy the XML file and its containing mappings data.

Step 6 Apply the configuration data to the appropriate servers.

Entering Mappings Data

Mappings.xml consists of three main mappings tables. You must complete the following tables:

- Service-to-element mapping (optional)
- Shortname-to-FDN mapping
- FDN-to-Node mapping

NOTE Input is mandatory in all columns of the table unless otherwise stated.

- **Service-to-Element Mappings**

  To add a service-to-element definition to Mappings.xml, execute:

  `ovcfgmappings -createServiceMap <service_name> <service_label> <xml_file>`

  This command creates the infrastructure for the mapping table.

  To add a service-to-element definition to the service-to-element mappings table, execute:

  `ovcfgmappings -addServiceMapElement <service_name> <shortname> <MOC> <FDN> <xml_file>`

  Table 13-1 provides a detailed description of the arguments of `ovcfgmappings` command.

Table 13-1 Description of Service-to-Element Maps

Parameter	Description
Service Name	Uniquely identifies a service in Service Navigator. This service must exist in Service Navigator.
Service Label	Label associated with a service.

Table 13-1 Description of Service-to-Element Maps

Parameter	Description
Shortname	An assigned alpha identifier for the network element instance. This name has to be unique among all network elements across the managed network. Validation of uniqueness is performed within an agent, but not across agents. The shortname should be one of the shortnames defined for lower topology objects in the agent configuration file.
MOC	This is the name of the managed object class type associated with this managed object instance.
FDN	The fully distinguished name of the managed object instance.
XML File	The name of the mappings XML configuration file.

- **Shortname-to-FDN Mappings**

  The shortname-to-FDN mapping table contains one entry for each network element defined as part of the TMN containment tree. An entry in this mappings table contains a shortname and a corresponding FDN.

  To add a shortname-to-FDN definition to Mappings.xml, execute:

  `ovcfgmappings -addShortNameFdnMap <shortname> <FDN> <xml_file>`

  Table 13-1 provides a detailed description of the arguments of `ovcfgmappings`.

Table 13-2 Description of Shortname-to-FDN Maps

Parameter	Description
Shortname	An assigned alpha identifier for the network element instance. This name has to be unique among all network elements across the managed network. Validation of uniqueness is performed within an agent, but not across agents. The shortname should be one of the shortnames defined for lower topology objects in the agent configuration file.

Table 13-2 Description of Shortname-to-FDN Maps

Parameter	Description
FDN	The fully distinguished name of the managed object instance.
XML File	The name of the mappings XML configuration file where shortname-to-FDN mappings are added.

- **FDN-to-Node Mappings**

  To add a FDN-to-Node definition to Mappings.xml, execute:

  `ovcfgtopodata -addFdnNodeMap <FDN> <Node name> <XML file>`

  The information to be entered is detailed in Table 13-3:

Table 13-3 Description of Upper Topology Table

Parameter	Description
FDN	The fully distinguished name of the container object instance for the topology object being mapped to an OVO node.
Node Name	The name of the OVO node to which the topology object is being mapped.
XML File	The name of the mappings XML configuration file where FDN-to-Node mappings are added.

Deploying Mappings Data

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

Execute: `ovcfgdeploy -mappings [-force] <project>`

This command first checks the consistency of managed object instance data. The validation checks to see whether:

- Configuration files generated from a previous deployment have been applied on the topology server.
- Mappings.xml has any updates since the last deployment.

If Mappings.xml is valid, then the command generates a configuration file to be applied to the OVO server.

The configuration file that is generated is: `ClientMaps.processed_pdb`. Next, this file should be applied to the OVO server for processing by OVO.

Applying Mappings Data

To apply the mappings configuration data on the OVO server, use the `ovtopodata.apply` and `ovoconf.apply` commands.

On the topology server, execute:

`ovtopodata.apply <project>`

On the OVO server, execute:

`ovoconf.apply <project>`

This utility takes the configuration file generated by deploying Mappings.xml and places them in the location indicated in Table 13-4.

Table 13-4 Mappings Configuration Files

File Name	Content	Destination
`ClientMaps.processed_pdb`	Information needed to generate the topology maps.	`$FMSVAR/import/ManagedObject`

The `ovtopodata.apply` command also pulls in information from the Mappings.xml file generated by the `ovcfgdeploy` command. Table 13-5 list the file to be pulled in by `ovtopodata.apply`.

See the `ovtopodata.apply` man page for more details.

Table 13-5 Other Configuration Files Read

File Name	Content	Destination
`Mappings.xml`	Shortname to upper level topology mappings.	`$FMSVAR/system/ShortNameMap.xml`

Using the Telecom Configurator

Optionally, you may enter the mappings data using the Telecom Configurator. For more information on how to use the Telecom Configurator, see the `ovcfgsa(1m)` reference page.

14 Configuring System Distribution Rules

This chapter describes how to define and distribute system distribution rules for a managed network. In particular, this chapter explains how to:

- Draw a containment tree for a managed network.

  This describes the procedure to define the managed object containment tree.

  See section “Drawing the Containment Tree” on page 183 for details.

- Define partitions.

  This describes the procedure to define partitions in the containment tree.

  See section “Defining Partitions” on page 221 for a detailed description.

- Define locations.

  A installation is divided into locations.

  See section “Defining Locations” on page 223 for procedures to define and maintain locations by their partitions.

- Enter system distribution rules into XML configuration file.

  See section “Entering System Distribution Rules” on page 224 for procedures to add definitions of locations and partitions and associated root elements.

- Distribute system distribution data to configuration files on OV Topology Server.

  See section “Distributing System Distribution Rules” on page 226 for procedures to deploy and apply system distribution configuration data to the topology server.

Configuring System Distribution Rules

System distribution rules help you configure:

- Managed object containment tree
- Partitions
- Locations

Whereas the network object model defines the logical objects in your managed network, the managed object containment tree defines the physical objects in your managed network. These physical objects are called managed object instances.

In a managed network, the number of managed object instances that are to be monitored can be quite large, up to a million. Thus, it is necessary for management purposes to divide the managed object instances into logical regions and distribute the work load among operators. These logical regions of distribution are called partitions and locations.

Figure 14-1 shows a sample partition and location.

Figure 14-1 Sample Location Showing Partitions

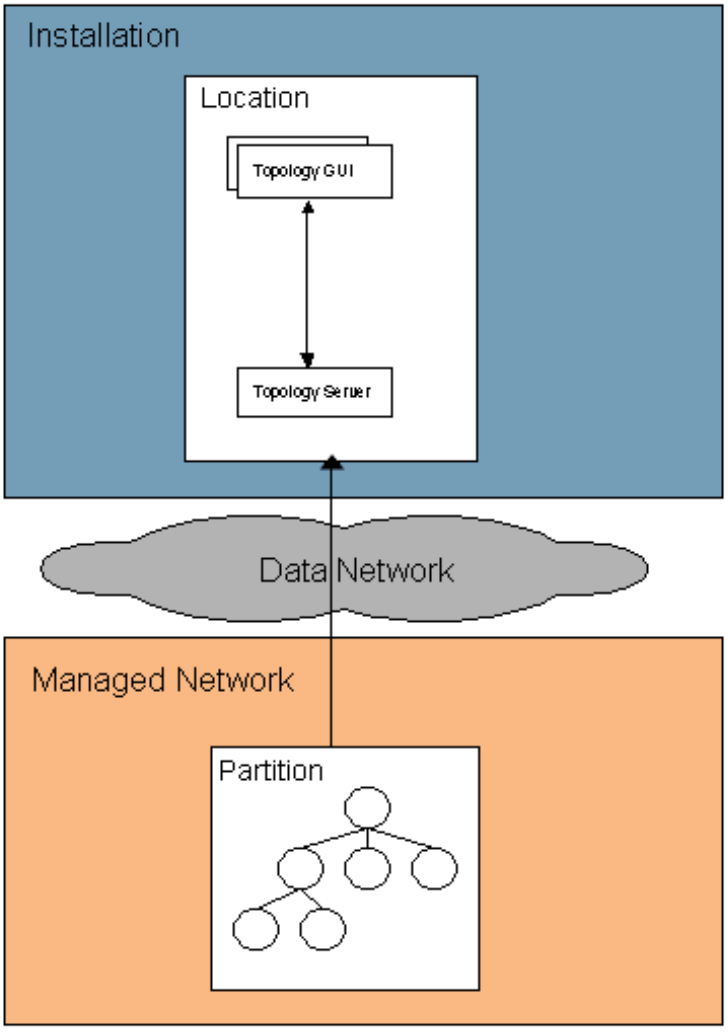
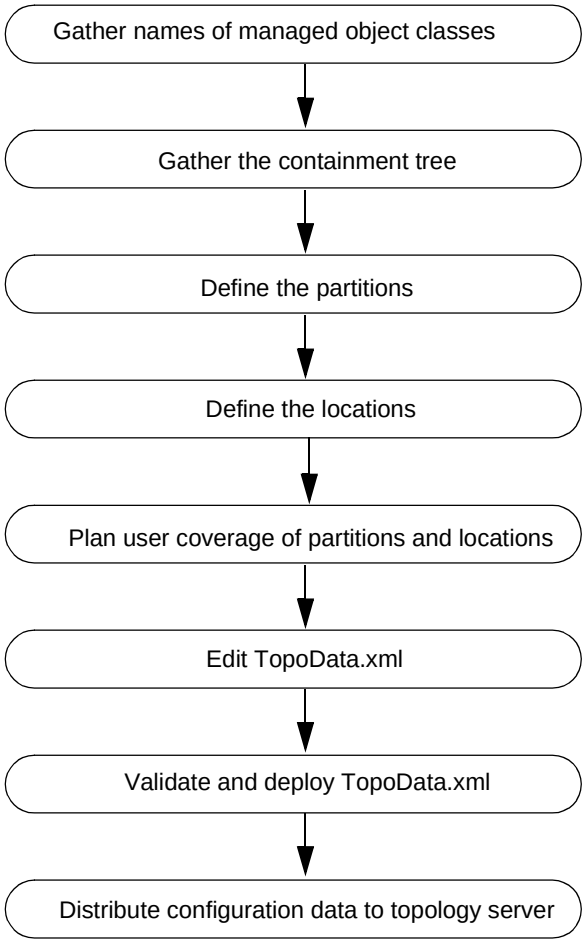


Figure 14-2 depicts a flow-chart of the steps involved in configuring the system distribution rules.

Figure 14-2 Steps in System Distribution Rules Configuration



Planning System Distribution Rules

To ensure smooth installation and configuration, pre-installation planning of the following details is essential:

Planning the configuration can be divided into two steps:

Step 1 **Defining Partitions**. This involves assigning branches of the managed objects topology tree to partitions to assist in distributing the monitored network among the locations.

Step 2 **Defining Locations**. Define the location by its partitions. This is the last step in planning the system distribution. Always keep this list current.

NOTE If the installation does not have a topology and managed objects of its own, the location and partition should be defined for the region, but no managed objects must be associated with the partition.

The following sections explain the planning steps in more detail.

Defining Partitions

With the your managed network object list, partitions can be defined.

Define a partition for each logical group of managed object instances. For example, you might define one partition for all the Nokia network elements and another partition for all of the EriAXE network elements as shown in Figure 14-3. You can just as easily define a partition for all the port elements in a managed network, a partition for all of the switch elements, and so on. Defining partitions depends on how the managed network is planned to be monitored.

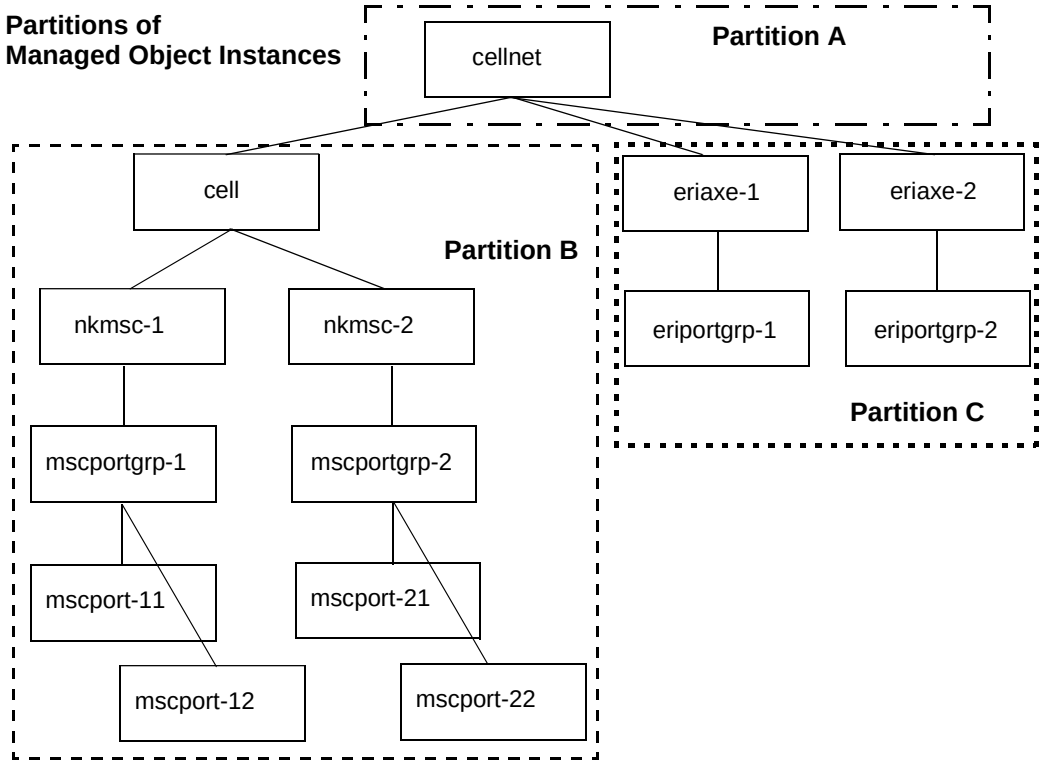
The following points about partitions are key:

- A partition is defined by the object(s) at the highest level of the containment tree. For example, in Figure 14-3, Partition A is defined by the root object instance `cellnet`. Partition B is defined by the network object instance `cell`. Partition C is defined by two network element instances: `eriaxe-1`, `eriaxe-2`.
- By default, all descendants of those objects defining the partition are automatically defined to belong to that partition.
- A managed object instance can belong to only one partition.
- Descendants of objects within a partition can be used to create new partitions. When this occurs, the objects used to define the new partitions as well as their descendants no longer belong to the previous partition.
- Only one partition is allowed per topology server.

NOTE

For most HP OpenView Service Assurance for Communication Networks installations, you need only define one partition. By defining the root elements of the containment tree within a managed network, all descendants are then automatically defined to belong to that partition.

Figure 14-3 Sample Partitioning



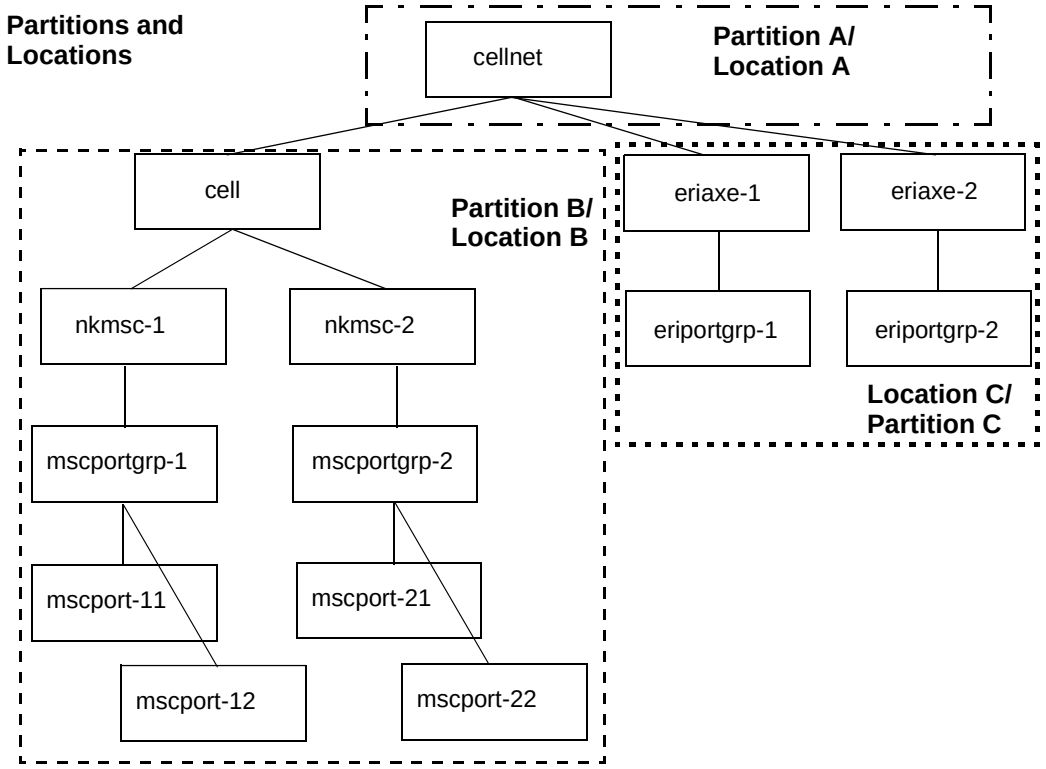
Defining Locations

A location is a cluster of one or more machines that offer OV Topology Server services. An entire managed network can be divided into locations, where each location contains only one topology server. For most HP OpenView Service Assurance for Communication Networks installations, only one topology server is deployed. Hence, only one location is defined.

A location is identified by the host name of its topology server.

Each location can have only one partition. By default, the location name is the same as the partition name.

Figure 14-4 Sample Locations and Partitioning



Entering System Distribution Rules

All partition and location definitions are stored in TopoData.xml. This XML file contains the definitions of the managed object instances used to define partitions and locations. In particular, TopoData.xml contains a table called Roots, which requires the following information:

- Root FDN
- Partition Name
- Location Name
- Partition Mask

Table 14-1 Roots and Partitions Table

Parameter	Description
Root FDN	This is the fully distinguished name of the managed object instance defining the partition.
Partition	The unique name of the partition for the topology server. Partition names have no significance outside the TopoData.xml file.
Location	This is the unique name of the location to be created. The location name is defined as the hostname of the topology server. By default, the location name is the same as the partition name.
Partition Mask	Mask value associated with a partition. It is an unsigned integer. The first partition is assigned mask value `0`. Each additional partition defined is assigned a partition mask increased by `1`.

Adding a Partition/Location

To add a root element, execute:

`ovcfgtopodata -addRoot <partition> [<location>]`

This command adds a new root definition to the TopoData.xml table and to the topology model configuration. *Location* is optional because it is typically defined to be the same as *partition*.

See its man page for additional details on adding and removing definitions of partitions.

Adding Roots to Partitions

After partitions have been defined, assign at least one managed object instance to each partition. The assignment of managed object instances to partitions is defined by the fully distinguished names (FDN) of the managed object instances. Note that when a managed object instance is assigned to a partition, all descendants of the managed object instance defined in the managed network containment tree are also assigned to that partition automatically.

To assign a managed object instance to a partition, execute:

`ovcfgtopodata -addAuth <partition> <root FDN>`

See its man page for additional details on adding and removing managed object instances from partitions.

Distributing System Distribution Rules

After entering the system distribution rules in TopoData.xml, you should perform the following:

- Validate TopoData.xml file.
- Generate configuration files from TopoData.xml file.
- Push generated configuration files to appropriate systems.

Deploying System Distribution Rules

After entering the system distribution rules data in TopoData.xml, validate and deploy the XML file. The validation process checks to see if the content you entered is consistent and valid against TopoData.dtd. The deployment process generates target configuration files and stores them in `generated/` under the project directory.

Execute: `ovcfgdeploy -topomodel <project>`

The `-topomodel` option is employed rather than the `-topodata` option, because it's the `-topomodel` option that reads the TopoData.xml file for system distribution rule information.

This command first checks the consistency of data. If the XML file is valid, then the command generates the following configuration file: roots_topomodel. This is a temporary file. Each line of file consists of a location, partition, partition mask, and object FDNs for that partition separated by a whitespace.

Applying System Distribution Rules

To pull the system distribution rules configuration file stored on the OVO server to the topology server for processing by OV Topology Server, use the `ovtopomodel.apply` command.

On the topology server, execute:

`ovtopomodel.apply <project>`

This utility reads each line of the configuration file generated by deploying TopoModel.xml and generates related SQL commands. The system distribution rules are applied to the topology server.

Use `-a` to add new configuration data into the system or `-r` to override the existing configuration data in the system with the new data.

If the topology server hostname is not defined as a location in the system distribution rules configuration file, then the apply command exits.

Table 14-2 lists the system distribution files that are generated on the topology server after apply command is run.

Table 14-2 System Distribution Configuration Files

File Name	Destination
`partition_to_location.conf`	`$FMSETC/share/conf`
`partition_fdnmask.conf`	`$FMSETC/share/conf`
`fdn_to_partition.conf`	`$FMSETC/share/conf`

Using the Telecom Configurator

This section describes how to use the Telecom Configurator to define system distribution rules.

All system distribution rules must be defined in TopoData.xml.

15 Configuring Event Correlation

This chapter explains how the event correlation tool is setup for OV Topology Server. It describes:

- The process of setting up and using Event Correlation Services (ECS).
- The process of setting up and using Event Correlator (OEMF-EC).
- The difference between the two supported event correlation tools.

The customized configuration files must be placed under the `$FMSETC/share/newconf` directory on the primary location FM Server.

Editing the files: These files can be edited using any standard text editor. Only user `oemfadm` can edit them. Other users have read-only access to these files.

NOTE

You can include your comments in all the files discussed in this chapter. In the Correlation Gateway file and all OEMF-EC configuration files, comments must be on new lines, prefixed with a # sign. So, each line starting with # in the first column is read as a comment.

OV Topology Server configuration files must be customized. Such server-specific files must have the filename suffixed with the server's hostname as part of the filename extension. In this chapter, it is indicated as *`serverhost`*.

You must follow the punctuation rules included in the files.

Event Correlation Options

To correlate event messages, OV Topology Server provides a choice of two event correlation tools:

- HP OpenView Communications Event Correlation Services (ECS)
- HP OpenView Communications Event Correlator (OEMF-EC)

See “Correlating Messages on Topology Server” on page 239 for a brief description on how event messages are correlated on the topology server.

Planning for Event Correlation

The events can be correlated using either OEMF-EC or *HP Event Correlation Services* (ECS). ECS is the default correlation tool, but either product can be configured to be the primary correlation tool. The event correlation information is configured using the Correlation Gateway file. (Refer to "Setting Up The Correlation Tool" on page 242 for details of the file.) Regardless of the correlation tool configured, you can define and customize the correlation rules and circuits.

Event correlation rules enable you to list the alarms that can be received by the data collectors and their interdependence.

For event correlation, Table 15-1 provides information that is required to correlate alarms.

Table 15-1 Event Correlation Details

Column Head	Description
Rule Type	This is the type of rule being defined. It is defined by the type of alarm it describes. This can be transient, root cause /related, or threshold alarm type. See the section, "Maintaining the Correlation Rules" on page 245 for a definition of these types of alarms.
Event type	This is the CMISE event, defined in the X.733 document, to which the rule applies. Refer to Appendix A, "List of Probable Causes," on page 369 for a complete list of the event type supported.
MOC	This is the managed object class name of the object that emits the alarm.
Probable cause	This is the probable cause of the alarm which is to be correlated. The probable cause must be as defined in the X.733 document.
Specific problem	This is the specific cause of the problem which is to be correlated.
Time-out	This is the time-slot, or duration in seconds, within which the filtering or correlation of the alarms is effective. This parameter is optional and defaults the time frame given in the default time frame stated in the Default EC Rules Files (`fmpmdevcd.conf` and `ovfmpevcd.conf`). A negative input in this field indicates a sliding window for correlation.
Threshold	This is the threshold limit used for threshold filtering type. It must a positive integer.

Table 15-1 Event Correlation Details

Column Head	Description
Correlated at	This is the server at which this rule is applicable. Depending on whether this rule is applicable on the MD or FM Server, it has to be entered in the `mdecrules.conf` or the `fmpecrules.conf` file.

Types of Correlation Rules

OV Topology Server provides three types of correlation regardless of the event correlation tool you configure. This section describes the types of correlation that can be performed using OEMF-EC and the built-in ECS circuit.

OV Topology Server executes three types of correlation, *correlation of transient alarms*, *correlation of root-cause/related alarms*, and *correlation of alarms through a user-defined threshold*. The filtered alarms can be classified as *root-cause/related alarms* and *transient alarms* across networks. The following paragraphs explain the concepts of these three types of alarms.

Transient Alarms

All alarms coming from the same managed object class and managed object instance ID with the same probable cause and specific problem but different severity levels and are repeated within a specified time period are considered transient.

For example, suppose any alarm repeated within a time frame of 10 minutes is considered transient. If the first alarm is received at 1:00 pm, then the alarm recurs before 1:10 pm, it is considered transient. This type of transient alarm monitoring is referred to as the *fixed window* transient filtering.

You could define the time frame or *window* to move forward with every recurrence of an alarm. In this case, you define a *sliding window* filtering. The time frame for checking transient alarms moves forward each time the alarm is repeated within the stated timeframe. When the alarm is received within the transient check window, the window is set forward to start at the time at which the alarm is repeated.

Using the above example with a sliding window, if an alarm is repeated at 1:05 pm, then the check for same alarm is maintained until 1:15 pm. The time-window is moved forward with each occurrence of the alarm. When the alarm is no longer transient (in this case, not repeated within 10 minutes), the count of number of recurrences of the alarm is sent to the topology server.

Root-Cause and Related Alarms

Two alarms, A and B, are defined as root-cause and related alarms respectively, if the fault that triggers A causes the fault that triggers B. Within OV Topology Server, B is recognized as an alarm related to A, if B is received within a specific period after A was received.

When a *root-cause alarm* is received, a check is maintained for its related alarm for the given time frame. If the alarm defined as *related alarm* is received within a period defined, then the two alarms are correlated, and only the root-cause alarm is displayed in the topology GUI problem presenter. The number of related alarms received within the given timeframe is stated under the column `Related Alarm Count` for a root-cause alarm. This is an alarm count on the topology server. The concepts of *fixed-window* and *sliding-window* are applicable for these types of alarms.

Alarms are reported to the Problem Manager on receipt. However, the count reported for these alarms depends on whether the time window is fixed or sliding. For fixed windows, the count of alarms (transient or related) is reported to the Problem Manager at the end of the time-window. In the case of sliding windows, the alarm count is sent to the Problem Manager when an alarm recurrence within a timeframe stops.

User-Defined Threshold Filtering

Threshold filtering is another correlation process provided by OV Topology Server to reduce spurious alarms in the alarm processing modules. This type of correlation filters alarms through a user-defined threshold before reporting its presence. For example, an alarm (A) is not considered noteworthy unless it is repeated at least a specific number of times (7) within a specific time period (20 seconds). So, alarm A is sent to the topology GUI only on its seventh occurrence within 20 seconds. If it does not occur 7 times within 20 seconds, the alarm is not sent to the topology GUI. The concepts of fixed-window and sliding-window are applicable for these alarms.

For any alarm, the related and root-cause alarm filtering rule takes precedence over the threshold rule. Similarly, both rules take precedence over transient filtering.

Comparing ECS and OEMF-EC

There are a few differences between the way ECS and OEMF-EC operate. Table 15-2 lists the differences in their operational behavior.

Table 15-2 ECS and OEMF-EC Comparison

ECS	OEMF-EC
ECS correlates based on the correlation circuits which can be created using the GUI-based ECS Designer. These circuits can be viewed graphically, and the built-in simulator can be used to visually verify the circuit. The resultant output codes are in binary.	OEMF-EC correlates based on event correlation rules. These rules are written into ASCII files and can be edited with any standard text editor.
ECS takes into account a **transit delay time**. The transit delay time is a user-definable period that can be configured for each circuit. Any event whose creation time falls outside the transit delay time window is passed through without being correlated by ECS. ECS passes the following events without correlating: • Buffered alarms not able to be received by the topology server. • Alarms delayed long in transit due to an alarm storm or the like.	OEMF-EC does not take into account the transit delay time. It receives, correlates, and processes all alarms at the time received, regardless of the delay in receipt of the event message.

Table 15-2 ECS and OEMF-EC Comparison

ECS	OEMF-EC
ECS filters event messages based on the following criteria before performing correlation: • The event creation time is greater than the current time minus the transit delay time. • The input event type must match the configured event type. • The input event encoding type must match the configured event encoding type.	OEMF-EC correlates all alarm messages received.
In the event the topology server goes down and restarts, ECS does not correlate the buffered alarms. This is because the hold time (or the transit delay time) for the alarm messages is exceeded.	In the event the topology server recovers from an abnormal shutdown, OEMF-EC correlates all the alarms received irrespective of hold time for the alarm. If the topology server is started in recovery mode, then all the non-discharged alarms are replayed and correlated. Buffered alarms transferred to the topology server are also correlated by OEMF-EC before being forwarded to the topology GUI.

Using ECS as the correlation tool within OV Topology Server provides you with other features not provided by OEMF-EC, including:

Dropping Alarms in the ECS Engine: You can drop an alarm within the ECS Engine on the topology server, if required. If so, this alarm is logged in the alarm database and not forwarded to the topology GUI problems presenter. Though this introduces an orphan alarm, an alarm not associated with any problem, orphan alarms are removed when you backup the alarm database using `fmsalmbackup`.

Generate New Alarms Based on Alarms Received: ECS circuits can be designed to create new alarms based on the alarms received. Both the newly created alarms and those received can be channeled through the Correlation Gateway to the topology server and exported to other user

applications to be processed as required. Using the ECS functionality and APIs, you can write your own annotate manager to send the alarm information from ECS to any other application. For example, you could generate mail based on the alarms received and send them electronically to the service center.

Note that the ECS generated alarms must follow the defined alarm format in order to be processed. This format is listed in the section “Alarm Format” on page 261.

Note also that if any of the alarm fields are modified by the ECS circuit, the modification is not reflected in the database. So, if the modification needs to be reflected in the alarm database for consistency purposes, then, instead of modifying any specific field, a secondary alarm must be created with the new values.

Correlating Messages on Topology Server

OV Topology Server provides correlation or filtering technology that enables users to eliminate superfluous messages according to specific user-definable criteria. Event correlation can dramatically reduce the number of events and improve the value of events displayed in the alarm browser. Instead of displaying the whole event storm typically generated by equipment and link failures, a correlated event stream displays only the most meaningful alarms, resulting in faster and easier identification of network problems.

OV Topology Server provides HP OpenView Communications Event Correlation Services (ECS) and Event Correlator (OEMF-EC) as two correlation tools to filter received alarms. ECS is the default correlation tool.

A **Correlation Gateway** enables event channeling to the configured correlation tool. The correlation tool is specified in the Correlation Gateway files.

If the correlation tool is OEMF-EC, then the Correlation Gateway channels the event messages to it. The OEMF-EC correlates the messages and passes the results to the next module.

If the configured correlation tool is ECS, then Correlation Gateway receives the correlated events and channels the results to the next module.

ECS can also be configured to generate secondary alarms that supplement the alarm information generated on a monitored network. The Correlation Gateway recognizes these alarms and reroutes them to the Alarm Logger for logging. On receiving these logged secondary alarms from the Alarm Logger, the Correlation Gateway reroutes them to the Problem Manager.

Using OEMF-EC as the Correlation Tool

OEMF-EC is rules-based, where the rules are maintained in an ASCII file. This correlation tool can be used by the topology server. OEMF-EC enables the following types of correlation to be performed:

- Transient and repeated alarm correlation.

- Root cause/related alarm correlation.
- Threshold alarm correlation.

On the topology server, OEMF-EC accesses the topology database, and correlates events across networks based on physical and logical links created by the system administrator/integrator.

To customize correlation using OEMF-EC, edit the event correlation configuration files.

Using ECS as the Correlation Tool

OV Topology Server offers the flexible and powerful ECS as a correlation tool. ECS provides advanced correlation capabilities and high performance. It also has the advantages of a GUI for configuring the correlation criteria and more flexibility in specifying the correlation criteria.

OV Topology Server provides a correlation filter or circuit for your use. This circuit can perform three types of correlation depending on the information provided in the factstores.

This circuit enables the following types of correlation to be performed:

- Transient and repeated alarm correlation.
- Root-cause / related alarm correlation.
- Threshold alarm correlation.

The pre-built circuit in ECS and the OEMF-EC correlation rules provide the same correlation capabilities. See the section “Types of Correlation Rules” on page 234 for more details. ECS, however, provides the flexibility to create new correlation circuits.

The major components of ECS include:

- **Correlation Engine (ECS Engine)**

  A process that transforms event streams according to the installed circuits and data stored in the fact and data stores. Data for specific functions can be retrieved from an external system via the annotation manager. The retrieved data is fed back to the correlation engine without impacting the other correlation activity. The ECS engine is installed when installing the topology server.

- **Configuration and Management Utility (ECS Engine**

Manager)

A utility that loads and unloads correlation circuits. This utility also manages event logging, engine logging, tracing and drill down to events in the log files among other functions. Refer to the *ECS Administration Guide* for details.

- **Correlation Designer and Simulator (ECS Designer)**

  A highly flexible graphical user interface that allows the creation, visualization, verification, and testing of correlation circuits to be loaded into the correlation engines distributed in a network. A correlation simulator enables administrators to view the events passing through the correlation engine, simplifying the testing of correlation circuits. The ECS Designer supports a hierarchical circuit design, allowing circuits to be saved as reusable components called compound nodes that enable either a top-down or bottom-up design. The ECS Designer is an optional product that must be installed if you plan to create your own circuits.

  With ECS Designer, a set of custom processing elements, or correlation nodes, are defined. Each type of correlation node performs a specific correlation function. These nodes can be combined to build simple or complex circuits (event correlations) through which the events flow.

ECS provides a dynamically configurable, real-time, indexed datastore, accessible externally to the correlation engine. ECS also provides a factstore to store the relational data among objects.

ECS also contains the following components:

- **OV Topology Server PFF Encoder/Decoder**

  The OV Topology Server PFF Encoder/Decoder converts event messages to a form that is readable by ECS. On receiving a correlated event from ECS, the Encoder/Decoder converts the message to the OVC/Assurance proprietary format.

- **ECS Integration Library**

  This module runs on the FM Server to retrieve topology information about the network elements emitting alarms. It retrieves information on physical links, logical links, and state of the object (whether on outage) that emits alarms.

For more information on using ECS with OV Topology Server, see the section “Using the Default ECS Circuit” on page 252.

Setting Up The Correlation Tool

A Correlation Gateway file is provided with the product to configure the event correlation tool. During installation, this file is created under `$FMSETC/share/newconf` directory on the topology server.

The correlation tool (or `MODE`) is entered in this file, and can be modified, when required. Thus, the Correlation Gateway file also facilitates the flexibility of using either correlation tool.

When ECS is defined as the correlation tool, this file also indicates the ECS Engine number.

To edit the Correlation Gateway File:

1. Log on as user `oemfadm` on the primary topology server machine.
2. Change directory to the configuration files directory:

   `cd $`*`FMSETC`*`/share/newconf`
3. Open the Correlation Gateway file using any standard text editor.

   To change the Correlation Gateway File, edit the file `ovfmpcgw.conf.`*`serverhost`*. Refer to the file details provided below.
4. Run the `fmscfgsync` utility to distribute the updated configuration.

Correlation Gateway File Details

File

`ovfmpcgw.conf.`*`serverhost`*

Where *`serverhost`* is an optional extension. It specifies the name of the server host if the file is created for a specific server.

Format

MODE = OEMFEC | ECS;

ECS_INSTANCE = *number*;

[AdminStateFiltering = ON | OFF;]

ECS_HA_COUPLED = 0;

MIGRATE2ECS = 0 | 1;

Where:

MODE

A parameter that indicates the correlation tool to be used with the topology server. The valid values are `ECS` and `OEMFEC`. This parameter is set to `ECS` when OV Topology Server is installed, which indicates that ECS is used to correlate the event messages.

You can set this parameter to `OEMFEC` to use OEMF-EC. In this case, when the topology server is started (or when the `fmseccfgupd` is run), it loads the event correlation configuration files and correlates the events per the defined rules. If the entry against this parameter is invalid, then the system uses the default OEMF-EC rules.

ECS_INSTANCE

A parameter that indicates the instance of ECS Engine to be used by the topology server. The valid values are any positive integer between 1 and 99. If the ECS engine instance is not specified, then the default value for the topology server is `1` (one). *The value of this parameter must be unique within the machine to avoid conflict when using the ECS engine.*

It is desirable if the ECS engine instance is explicitly specified to prevent potential conflicts with other ECS engines running on the same machine. Also, avoid using `1` when using OV DM TMN on the same machine, since instance `1` is used by OV DM TMN.

AdminStateFiltering

An optional parameter that is applicable only on the FM Server (hence must be entered only in the file `ovfmpcgw.conf`). It is used to state whether the ECS must filter alarms from network elements on outage. The valid values are `ON` and `OFF`. The default value is `OFF`.

When this parameter is set to `OFF`, alarms from objects on an activated outage plan are correlated. If the parameter is set to `ON`, then alarms from objects on an activated outage plan are *not correlated*, but channeled directly to the next module, that is the Problem Manager.

ECS_HA_COUPLED

A parameter that is applicable only to high availability (HA) topology servers. It is used to indicate the behavior of HA topology server if the ECS Engine stops while the server is running. The valid values are `0` (zero) and `1` (one). The default value is `0`. Note that any value other than `1` is taken as `0`.

MIGRATE2ECS

A parameter that can be set to `0` or `1`. Set to `1` if the OEMF-EC configuration files have been converted to ECS factstore files and the default built-in circuit is to be used for correlation. Set to `0` if a custom circuit is to be used by the ECS Engine, or ECS is not being used as the correlation tool. The default value for this parameter is `0`.

If in the Gateway Configuration File, the `MODE` is set to `ECS` and `MIGRATE2ECS` is set to `0`, then after the topology server has been started, the administrator must use the ECS utility **ecsmgr** to load the circuit, factstore, and datastore, as applicable. Refer to the *ECS Administration Guide* for details on using this utility.

Using OEMF-EC as the Correlation Tool

Maintaining the Correlation Rules

Alarms are correlated in two stages of message processing. On the telecom agent, the event filter module correlates and filters event messages received from managed object instances that are contained within the same network element - as defined while configuring the managed object classes. On the topology server, the alarm correlator correlates alarms received from within the same network. Besides this, messages from devices that are physically or logically linked can also be correlated. (Physical links are established during network configuration. A command-line utility is also provided to establish logical connections. See the man page for `fmslogconadmin` for more details about this command.

Customizing OEMF-EC Rules

This section provides details on the fields to edit to customize correlation performed by OEMF-EC. The files to be edited are:

Correlation Gateway Configuration File

This file is used to specify the correlation tool to be used. There is one file for the topology server. See “Setting Up The Correlation Tool” on page 242 to set the event correlation tool to OEMF-EC.

Event Correlation Global Rules File

This file indicates the type of filtering that must be executed. There is one file on the topology server. This file must be created and customized if you wish to use OEMF-EC as the event correlation tool.

Event Correlation Rules File

This file is used to maintain the event correlation rules. There is one file on the topology server. This file must be created and customized if you wish to use OEMF-EC as the event correlation tool.

To customize the correlation:

1. Log on as user **oemfadm** on the primary FM Server.
2. Go to directory `$FMSETC/share/newconf`.
3. Edit the configuration file.
4. Save the file.
5. Shutdown the topology server.

NOTE The Event Correlation Global Rules and Event Correlation Rules files need be maintained only if the OEMF-EC is used as the correlation tool.

Event Correlation Global Rules File Details

File `ovfmpevcd.conf.`*`serverhost`*

Where *`serverhost`* an optional extension. It specifies the name of the server host if a set of correlation rules are created specific to a host.

Format *`RulesType`*`Filtering = ON|OFF;`

`RulesType``Timeout = nn;`

Where:

`RulesType``Filtering`

Is the type of filtering rule being defined. The *`RulesType`* can be one of the following:

`Transient`	Indicates that the rule applies to transient alarms.
`AdminState`	Indicates that the rule applies to network elements in administrative state; that is, the network elements are on outage. See the note on AdminStateFiltering on page 247.
`Threshold`	Indicates whether threshold filter is turned ON or OFF.

If any of these filtering rules is turn OFF, Event Correlator does not execute that type of filtering.

`ON|OFF`

Enter ON for the *RulesType*`Filtering` if the filtering must be applicable to all alarms. Set the *RulesType*`Filtering` to OFF if this type of filtering is applicable only to those alarms defined in the EC rules file.

RulesType`Timeout`

Is the default duration in seconds during which the filtering or the correlation rule of alarms is effective. If the time is not defined, then the default event correlator fixed time frame is used. The default time fixed by the event correlator is 20 seconds.

The *RulesType* can be one of the following:

`Transient`	Indicating that the rule applies to transient alarms.
`Related`	Indicating that the rule applies to related alarms.
`Threshold`	Indicating that the rule applies to threshold filtering.

nn

Is the time window value in seconds. The minimum time that can be specified is one second. However, due to system restrictions, it is recommended to specify at least twenty seconds. A small timeout window, less than twenty seconds, coupled with a small threshold value can cause unexpected behavior.

By default, the time stated is taken to be a *fixed* window timeframe. If the time specified is prefixed with a minus sign (-), then the window for this rule is considered *sliding*.

Note on AdminStateFiltering:

If `AdminStateFiltering` is set to `ON`, then alarms from devices in administrative state are not correlated. All alarms received from such devices are sent to the next module. To correlate alarms from devices in administrative state, set the `AdminStateFiltering` to `OFF`.

Example 15-1 Sample of the Global EC Rules File (ovfmpevcd.conf.serverhost)

```
#Transient filtering done on all problems if this flag is set to ON.

#If set to OFF, transient filtering is done only on problems specified in
#the ecrules.conf files.
TransientFiltering = ON;

#Events from devices in non-operational states are not correlated if this flag
#is set to ON.
#If set to OFF, alarms from administrative state devices are correlated.
AdminStateFiltering = ON;

#Default timeout for all transient rules
TransientTimeout = 20;

#Default timeout for all related rules
RelatedTimeout = 25;
```

Event Correlation Rules File Details

File

`fmpecrules.conf.`*`serverhost`*

Where *`serverhost`* is the name of the host to which this files belongs.

Format

`RulesType`, *`EventType`*, *`MOC`*, *`ProbableCause`*, [*`SpecificProblem`*], [*`Timeout`*], [*`Threshold`*];

Where:

`RulesType`

Is the type of rule being defined. This can be one of the following:

`TRANSIENT`	Indicates rule defining transient alarms
`THRESHOLD`	Indicates rule defining the alarm threshold
`ROOTCAUSE`	Indicates rule defining root cause alarms
`RELATED`	Indicates rule defining related alarms

Each `ROOTCAUSE` type rule must have at least one corresponding `RELATED` type rule.

`EventType`

Is the kind of event to which the rule applies. This event type of the five defined X.733 event types.

`MOC`

Is the managed object class that emits this event type.

`ProbableCause`

Is the probable cause of the alarm as defined by X.733. Refer Appendix A, “List of Probable Causes,” on page 369 for the supported list.

SpecificProblem

Is the specific problem of the event/alarm being emitted. This is an optional parameter.

If this parameter is not specified, then the rule is applicable to all event messages that match the other parameters that are specified.

Timeout

Is the duration in seconds during which the filtering or the correlation of an alarm is effective. This parameter is optional.

By default, the time stated is taken to be a fixed window timeframe. If the time specified is prefixed with a minus sign (-), then the window for this rule is considered *sliding*.

The default timeout value is specified in the files `fmpmdevcd.conf` and `ovfmpevcd.conf` on each server. The default value is used if no timeout is specified.

Threshold

Is the threshold level of an alarm, and is applicable and mandatory to the rulestype THRESHOLD. It must be a positive integer.

NOTE

Any rule specified for an alarm condition in the EC Rules Files takes precedence over the rule set in the Global Rules files.

Each rule must start on a new line.

Example 15-2 Sample of the EC Rules Files (fmpecrules.conf.serverhost)

```
#Example of root cause alarm filtering:
ROOTCAUSE,equipmentAlarm,NokiaMSC,powerProblem,power-problem,;
RELATED,equipmentAlarm,MscPwrComp,powerProblem,power_problem,;

#Example of transient alarm filtering:
TRANSIENT,equipmentAlarm,NokiaMSC,floodDetected,flood_detected
,20;
```

Alarm Types

The alarm types and their time windows are defined in the files described in Table 15-3.

Table 15-3 Alarm Type Details Files

Filename	Description
`ovfmpevcd.conf.`*`serverhost`* (on the FM Server)	The Global Rules Files contain the values of parameters, alarm type and time-windows, to be used by OEMF-EC for any alarm condition for which a rule is not specified in the EC Rules Files.
`fmpecrules.conf.`*`serverhost`* (on the FM Server)	The EC Rules Files contain the alarm patterns for filtering the different types of alarms.

NOTE Any rule specified for an alarm condition in the EC Rules Files takes precedence over the rule set in the Global Rules files.

Each rule must start on a new line.

Using the Default ECS Circuit

This section is divided into two parts. The first describes the procedure to convert the OEMF-EC configuration to ECS configuration. The second part describes usage of the default circuit available when the topology server is installed. The default circuit provides the correlation offered by OEMF-EC.

ECS Documentation

A library of ECS manuals in printable and web-browsable format are available at http://docs.hp.com. The printable Adobe Acrobat files can be installed from the ECS CD-ROM.

The ECS documentation set consists of several manuals. You should start with the following manuals:

- *HP Open View Event Communications Correlation Service Installation Guide* – This book guides you through the process of installing HP Open View Event Correlation Services on HP-UX, Windows NT/2000, and Solaris.
- *HP Open View Communications Event Correlation Services Administrator's Guide* – This book explains how to manage the ECS Engine. It provides information on how to initialize, start and stop, operate, and troubleshoot the ECS Engine.

ECS Files

ECS adds files to the existing directories of OV Topology Server. Files are added to `$FMSETC/share/newconf`.

Circuit Files

ECS uses these types of files:

`correlation.eco` – Compiled correlation file (not editable). This file uses the other files (datastores and factstores) to execute correlation.

`correlation.ds` – Datastore file containing externally configurable parameter values (text file).

`correlation.fs` – Factstore file containing topological or other information describing relationships (text file).

`correlation.param` – Parameter file that describes the parameters configurable by the `ECS Event Configuration` window (text file).

Setting up ECS on Topology Server

As part of the topology server installation, an ECS circuit is installed which emulates the correlation provided by the OEMF-EC rules. To use this circuit for correlation, a factstore must be created just as correlation rules need to be defined to use OEMF-EC. For the definition of the ECS factstore, see “Format for the Default Factstore” on page 256.

To convert installations that have been using OEMF-EC as the correlation tool to ECS:

- Convert OEMF-EC rules to ECS format by running `migrateEc2Ecs`, a conversion utility provided to convert the existing rules into a format that is readable by the ECS default circuit.
- Configure the Event Correlation Gateway file appropriately.
- Run the synchronization configuration utility, `fmscfgsync`.
- Restart the servers to enable the new event correlation tool.

Converting OEMF-EC Rules to ECS Format

To convert OEMF-EC rules to the ECS format:

1. Log on as `oemfadm` on the primary topology server.
2. Run the `migrateEc2Ecs` utility: **`$FMSETC/util/migrateEc2Ecs`**

   This creates the factstore `$FMSETC/share/newconf` directory. The factstore files are named `fmsecsfact.fs[.`*`serverhost`*`]`.

   Informative messages are displayed during the processing, and on completion the following message is displayed:

   ```
   Migration of EC configuration files to ECS FactStore is
   done.
   ```

   Note that the files are converted in pairs of global rules and specific rules.

   Note that if the utility had already been run and the OEMF-EC rules had been converted to factstore files, the following messages are displayed:

   ```
   ==> Migrating FMS-EC-Rules files to ECS FactStore file...
   INFO : FactStore file [./fmsecsfact.fs] already exists.
   Will not re-create.
   ```

Configuring Event Correlation Gateway File for ECS

The Event Correlation Gateway file must be configured to use ECS. The file should indicate whether it is using the migrated OEMF-EC rules or the default ECS circuit. Edit `$FMPETC/share/newconf/ovfmpcgw.conf.`*`hostname`* on the primary topology server as described in “Correlation Gateway File Details” on page 242. This file should include at least:

```
MODE=ECS
MIGRATE2ECS=1
ECS_INSTANCE=1
```

Note that a unique number must be entered for the `ECS_INSTANCE`.

NOTE

The **MIGRATE2ECS** parameter indicates whether the OEMF-EC rules have been migrated to ECS, and whether the ECS circuit and factstores must be loaded after the server, corresponding to this file, is started.

If the correlation mode is configured for ECS and `MIGRATE2ECS` parameter is set to `1`, then the default ECS circuit and its migrated factstore are loaded when the server is started.

Configure the correlation mode for ECS and `MIGRATE2ECS` to `0` when the ECS circuit is a custom circuit. After the topology server is started, use the `ecsmgr` utility to load the circuit, factstore and datastore, if required.

Restarting the Server

Shutdown and restart the topology server as user `root`.

Format for the Default Factstore

The interface to the ECS circuits is the factstore. The following is the format of the factstore created by the `migrateEc2Ecs` utility. Comments can be added to the file by entering a double dash (--) beginning with the first column of the row. *Note that # is not an acceptable comment sign for this file.* See "Sample Factstore File for Topology Server" on page 260 for an example of a factstore file.

The first line of this file must contain the filename, time stamp and version in the following format:

#*filename*#*timestamp*#*version*#1

The following is the description of the parameters in the above line:

filename	This is the name of this factstore file you are viewing. It will be of the format *servertype*`ecsfact.fs`, where the *servertype* is either `fms` or `md`. Note that if the factstore file is server specific, then, it will have an additional extension indicating the server hostname.
timestamp	This is the UNIX timestamp showing the date and time when the file was created.
version	This provides the version number (*#.#*) of the file. It is recommended that it be updated whenever changes are made to the file.
1	This is for internal use, and must not be changed.

For global rules, the following syntax is used:

```
ADD FACT ("MD_OR_FMS", 1, "server_type")
ADD FACT ("rulestype", 1, "[ON|OFF]")
ADD FACT ("DEFAULT_filter-type_WINDOW", 1, time-seconds)
```

Where:

server_type	The value is `FMS`, which indicates that this file is for the topology server (containing the FM Server).
rulestype	This is used to indicate the type of correlation in the global rule. It can be: `TRANSIENT_FILTERING_ALL` to indicate whether, by default, transient filtering must be executed on all alarms.

ADMINSTATE to indicate whether, by default, alarms from objects in administrative state must be correlated.

THRESHOLD to indicate whether, by default, threshold filtering must be executed on all alarms.

filter-type This indicates the type of correlation in the default correlation window definition. It can be:

TRANSIENT to indicate transient filter window.

ROOTCAUSE to indicate root cause filter window.

THRESHOLD to indicate window for threshold filtering.

time-seconds This is the default time window of the correlation type defined by *filter-type*. The time must be defined in seconds. Note that a negative integer would indicate a sliding window.

Besides this, there are two more global rules:

ADD FACT ("DEFAULT_THRESHOLD", 1, *threshold-limit*)
ADD FACT ("NETWORK_DELAY", 1, *delay-seconds*)

Where:

threshold-limit

This is the number of alarms that set the threshold limit. By default, the threshold correlation filter suppresses all alarms belonging to a problem condition if the number of alarms that arrive for the problem condition, within the correlation time-out window, is less than the *threshold-limit*.

delay-seconds

This is the maximum transit delay time allowed for the alarm. The time must be defined in seconds.

The MOC-specific correlation rules are defined as follows. Each line is a event correlation filter. All punctuations (the regular and square brackets, commas, and quotation marks) in the following syntax must be included in the fact file. *Note that in the following syntax the square brackets are part of the punctuation and do not indicate optional parameters. Curly brackets are used to indicate used optional parameters.*

ADD FACT ("RC", 1, [*pc-info*, [*related-pc-info{, t-window*}]
{, [*related-pc-info, t-window*], ...}])

ADD FACT ("TR", 1, [*pc-info*, *t-window*])
ADD FACT ("TH", 1, [*pc-info*, *threshold-limit*, *t-window*])

Where:

"RC" "TR" "TH"

This defines the type of correlation executed by that "**ADD FACT**" line. **RC** indicates root-cause related type of correlation. **TR** indicates transient correlation, and **TH** indicates threshold correlation.

pc-info

This is the description of the problem condition which must be correlated. The description is defined as follows:

"*eventtype*, *moc*, *probable-cause*, *specific-problem*"

The *moc* is the name of the network element class. The *eventtype*, *probable-cause*, and *specific-problem* are those defined for the managed object class defined by *moc* in this statement. This definition of the pc-information must be enclosed by double-quotes and the fields within the pc-information are delimited using a comma and a space (**,**).

related-pc-info

This is applicable only to the *rootcause* type of correlation and is the definition of the problem condition of the related alarm. The definition is as follows:

"*eventtype*, *moc*, *probable-cause*, *specific-problem*"

The *moc* is the name of the network element class. The *eventtype*, *probable-cause* and *specific-problem* are those defined for the managed object class defined by *moc* in this statement. The fields must be delimited using a comma and a space (**,**)

This definition of the related alarm must be enclosed by double-quotes. The related alarm and its time

window, if any, must be enclosed within square brackets as indicated in the syntax. Note that if there are multiple related alarms, then each of them must be listed on a new line.

threshold-limit

This is the threshold limit specific to the **pc-info** specified in this filter.

t-window

This is the time window of the correlation defined for this filter. The time must be defined in seconds. A negative integer would indicate a sliding window.

Example 15-3 Sample Factstore File for Topology Server

```
#./fmsecsfact.fs#<Thu Mar 29 11:43:48 2001>#<1.0>#<1>

-- The meaning of the FACT values can be inferred from
-- their names.

ADD FACT (“MD_OR_FMS”, 1, “FMS”)

ADD FACT (“TRANSIENT_FILTERING_ALL”, 1, “ON”)
ADD FACT (“ADMINSTATE”, 1, “ON”)
ADD FACT (“DEFAULT_TRANSIENT_WINDOW”, 1, 20)
ADD FACT (“DEFAULT_THRESHOLD_WINDOW”, 1, 20)
ADD FACT (“DEFUALT_ROOTCAUSE_WINDOW”, 1, 20)


-- Factstore entry for ROOTCAUSE/RELATED rules

ADD FACT (“RC”, 1, [“equipmentAlarm, NokiaMSC, powerProblem,
power_problem”, [
[“equipmentAlarm, MscPortSubA, powerProblem, power_problem”]
]])

ADD FACT (“RC”, 1, [“equipmentAlarm, NokiaMSC, powerProblem,
power_problem”, [
[“equipmentAlarm, MscPwrComp, powerProblem, power_problem”]
]])

ADD FACT (“RC”, 1, [“equipmentAlarm, NokiaBTS, powerProblem,
power_problem”, [
[“equipmentAlarm, BtsPwrComp, powerProblem, power_problem”]
]])

ADD FACT (“RC”, 1, [“equipmentAlarm, NokiaBSC,
processorProblem, processor_problem”, [
[“equipmentAlarm, BscPort, processorProblem,
processor_problem”]
]])

ADD FACT (“RC”, 1, [“equipmentAlarm, BtsSub1, softwareError,
software_error”, [
[“equipmentAlarm, BtsSub3, softwareError, software_error”]
]])

ADD FACT (“RC”, 1, [“equipmentAlarm, NokiaMSC, congestion,
congestion”, [
[“equipmentAlarm, NokiaBSC, congestion, congestion”]
]])
```

Creating Circuits with ECS Designer

You must have the ECS Designer installed in order to create new circuits. This section lists the alarm format needed when using the ECS Designer to create event correlation circuits. See *HP OpenView Communications Service Assurance Installation Guide* for details on installing and setting up the ECS Designer.

Alarm Format

This section describes the setup required to use the ECS Designer, and the OV Topology Server alarm format used in the context of the ECS Designer. The message fields are referenced in string form in the ECS circuit nodes. The following table lists the data type restrictions for the OVC/Assurance alarm format fields. While the alarm contains other fields, those fields are not for extraction. For the details of the data type restrictions refer to the chapter on Data Types in the *HP OpenView Communications ECS Designer's Reference*.

Table 15-4 Data Type Restrictions for Alarm Format Fields

Field Name	Data Type	Description	Max Size (bytes)
ServicePrimitive	`OEMF_SERVICEPRIMITIVE_TOK`	Defines the service primitive of the current event message. The valid service primitive is: `M_EVENT_FMP_REPORT`	Fixed - 4
Format	`OEMF_FORMAT_TOK`	Defines the format of the current event message. The valid value is `FMP_SPECIFIC_FORMAT`	Fixed - 4

Table 15-4 Data Type Restrictions for Alarm Format Fields

Field Name	Data Type	Description	Max Size (bytes)
Mode	`OEMF_MODE_TOK`	Defines the mode of the event message. The possible values are: • `Confirmed` • `Unconfirmed`	Fixed - 4
ManagedObject Class	String	Defines the managed object class of the object sending the event message. The valid values are those configured using the OVC/Assurance Configurator. Maximum length is 32 characters.	32
ManagedObject Instance	String	Defines the managed object instance of the object sending the event message. The valid values are those configured into the OVC/Assurance system. Maximum length is 198 characters excluding the Null character.	256

Table 15-4 Data Type Restrictions for Alarm Format Fields

Field Name	Data Type	Description	Max Size (bytes)
EventType	`OEMF_EVENTTYPE_TOK`	Defines the event type of the event message. The possible values are: • `communicationsAlarm` • `qualityOfServiceAlarm` • `processingErrorAlarm` • `equipmentAlarm` • `environmentalAlarm`	Fixed - 4
EventTime	Time	Defines the event time of the event message.	Fixed - 4

Table 15-4 Data Type Restrictions for Alarm Format Fields

Field Name	Data Type	Description	Max Size (bytes)
ProbableCause	`OEMF_PROBABLECAUSE_TOK`	Defines the probable cause in the event message. The valid values are the X.721 and M3100 values for the Event type listed under EventType. These values are listed in the text file `oemf_ecs_prob_cause.txt` in `$FMSDIR/ReleaseNotes` for reference. Note that the probable causes for M3100 are prefixed with `M3100_`. These values must be entered as is in the factstore. However, this prefix will not be displayed in the Problem Presenter, under the Probable Cause column.	Fixed - 4

Table 15-4 Data Type Restrictions for Alarm Format Fields

Field Name	Data Type	Description	Max Size (bytes)
PerceivedSeverity	`OEMF_PERCEIVEDSEVERITY _TOK`	Defines the perceived severity in the event message. The valid values are: • `Indeterminate` • `Critical` • `Major` • `Minor` • `Warning` • `Cleared`	Fixed - 4
BackedUpStatus	`OEMF_BACKEDUPSTATUS _TOK`	Defines the back up status of the event message. The valid values are: • `NotBackedUp` • `BackedUp`	Fixed - 4
TrendIndication	`OEMF_TRENDINDICATION_TOK`	Defines the trend of the alarm severity change. The valid values are: • LessSevere • NoChange • MoreSevere	Fixed - 4

Table 15-4 Data Type Restrictions for Alarm Format Fields

Field Name	Data Type	Description	Max Size (bytes)
SpecificProblem	String	Defines the specific problem of the alarm. The valid values are those configured into the OVC/Assurance system.	64
NotificationIdenti-fier	Integer	Defines the notification identifier in the event message. This is a system defined value present in the system.	Fixed - 4
AdditionalText	String	Defines the user specified additional text in the event message. Maximum length is 256 characters.	4096
AdditionalInform-ation	String	Defines the user specified additional information in the event message. Maximum length is 4096 characters.	4096
ERId	Integer	Internal use. Do NOT modify.	Fixed - 4

Table 15-4 Data Type Restrictions for Alarm Format Fields

Field Name	Data Type	Description	Max Size (bytes)
AlarmId	**String**	Internal use. Do NOT modify.	128
CorResult	`OEMF_CORRESULT_TOK`	Internal use. Do NOT modify.	Fixed - 4
AdminState	`OEMF_ADMIN_TOK`	Internal use. Do NOT modify.	Fixed - 4
TransientCount	Integer	Internal use. Do NOT modify.	Fixed - 4
RelatedCount	Integer	Internal use. Do NOT modify.	Fixed - 4
PortName	String	Internal use. Do NOT modify.	Internal use. Do NOT modify.
Orefld	String	Internal use. Do NOT modify.	4096
NetworkMoc	String	This is the MOC of the parent network of the object that has emitted the alarm.	256
NetworkEquip Moc	String	This is the MOC of the parent network element of the object that emitted the alarm.	256
NetworkEquip Shortname	String	This is the short name of the parent network element of the object that emitted the alarm.	4096

Modifying Alarms with ECS Designer

Messages correlated by ECS can be modified using the Modify node of the ECS Designer within a circuit. When the Modify node is used, depending on the part of the message modified, a new alarm may be generated. If the Modify node is used to change the value of any of the following key fields, then a new alarm is generated:

- ManagedObjectClass
- ManagedObjectInstance
- ProbableCause
- SpecificProblem

Since alarms are identified and correlated by the value of these four key fields, any change in these fields defines a new alarm. ECS outputs both alarms. However, the Alarm Manager identifies alarms by the Alarm ID and the key fields. So, the Alarm Manager accepts the first alarm that reaches it, and rejects the second alarm since both the original alarm and the modified one have the same Alarm ID. The Alarm Manager logs a message in the `syslog` file to indicate that a message was not processed.

You can set ECS to discard the original message and process the modified one. However, if you wish to process both the original message and the modified one, then use the Create node in the circuit to create the new (modified) message and process the new message and the original one.

Choosing ECS Designer

Developing your own event correlation circuits is a powerful option, but not a trivial task. You should carefully consider whether designing circuits is something you want to take on or contract with a specialist to supply the circuits you need.

If you need only one or two special circuits, then it probably not worthwhile to develop them yourself. In this case, you should consider contracting with a specialist to develop them for you since it would be quicker, simpler, and cheaper.

16 Configuring Status Propagation

This chapter describes the procedure to customize status propagation, using the Status Propagation Configuration file. This file can be edited using any standard text editor and must be saved under the `$FMSETC/share/newconf` directory.

This file must be edited only on the topology server under the `oemfadm` login. All other users have read-only access to the file.

About Status Propagation

To enable an operator viewing a high level map receive a clearer picture of the status of the network equipment, the status of the object is propagated to their parent objects. By default, the most severe of the child object status is propagated to its parent object. Thus enabling the operator viewing a high level map to know the presence of any alarms in the lower level objects.

Status propagation is based on the actual alarms that are received by OV Topology Server. It is not influenced by any operation profile or problems presenter filters. The problems presenter, on the other hand, only presents a view into the problems managed by the topology server. This view is shaped by the domain, the problems presenter filter, and the operation profile filter. Operators use filters to reduce the amount of information they need to attend to and focus on the essential information. A user may observe inconsistency between the status in a map and problem in the problems presenter if filters are applied. To make the map consistent with the problems presenter, remove all problems presenter and operation profile filters.

This status propagation rule can be customized using the Status Propagation Rules file. This is described in the section “Setting up Status Propagation Rules” on page 276.

The status propagation rules can be defined for each object class and status combination. They govern the propagation of alarm status from child object classes to their parent object class. All classes stated in the rules are cross validated against the object classes. Any operational status can be used for propagation except `Normal`.

Object status can be propagated from the lowest level non-map objects to the network element object on the map presenter, irrespective of the number of levels of non-map objects between the object instances emitting the alarm and the network element on the map.

For status propagation, only component class and termination point class objects can be non-map objects, the network and network element class objects MUST BE MAP OBJECTS.

For any class that has not been defined as a parent-class in any of the status propagation rules, the default propagation rule is applied. The default rule sets the parent object status to the most severe of its child

objects status.

If there are several propagation rules for a specific parent class resulting in different levels of severity status, then the parent status is set to the most severe status resulting from the rules. For more details on the default rules refer to the section “Default Propagation Rules”.

Default Propagation Rules

There are two default rules for status propagation.

- For any class that has not been defined as a parent-class in any of the status propagation rules, the class status is set to the most severe of its child class status.
- If there are several propagation rules for a specific parent class resulting in different status, then the parent status is set to the most severe status resulting from the rules.

It is important to note that:

- The status propagation rules are defined by the object class and not the severity.
- The default status propagation rules apply only to those object classes that have not been defined as a parent class in any of the status propagation rules.

  If one or more rules have been defined for a given parent class, no other propagation will occur for that object class. For example, if one rule is defined such that three or more minor alarms propagate to set the parent class object to major. Then no other change in any of its child class objects of any level of severity will be propagated to this parent class object.
- Propagation rules that are deleted, modified, or added in `sprules.conf` are only effective upon running the `configuration synchronization` utility, `fmscfgsync`.
- The object status due to existing problems from previous events will not be changed by restarting the topology server with modified propagation rules.

Planning

OV Topology Server provides status propagation to propagate the change in status of the lower-level managed objects to the status of the managed objects above them. See “Setting up Status Propagation Rules” on page 276 for a detailed explanation of the Status Propagation Rules file.

To set the status propagation rules, you define the parent managed object class, the child managed object class, and the status of the parent managed object class in relation to the percentage or number of a particular status of the child managed object class. For example, in a network object made up of multiple components, if two of the components become critical, the network object can be configured to display minor status.

The change in status of more than one class of objects can be reflected in the status of a parent object class. This is achieved by linking the change to the status of both the child object classes to that of the parent object class in a single statement, as shown in the following example from the $*FMSDIR*/sampleconf/sprules.conf:

```
Cell IS CRITICAL IF_MORE_THAN 0 MotorolaBTS SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MINOR
AND IF_MORE_THAN 0 NokiaBSC SUBOBJECTS ARE_OR_MORE_SEVERE_THAN
MINOR AND IF_MORE_THAN
0 NokiaBTS SUBOBJECTS ARE_OR_MORE_SEVERE_THAN MINOR

NokiaBTS IS CRITICAL IF_MORE_THAN 18% BtsSub1 SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN
MINOR

NokiaBTS IS CRITICAL IF_MORE_THAN 0 BtsPort SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MINOR
```

In the above sample, the objects of the Cell object class become critical if one or more of the MotorolaBTS, NokiaBSC and NokiaBTS become minor or more severe than minor.

All levels of severity can be used for propagation rules except Clear.

Default Propagation Rules

The following default propagation rules apply to all:

- For any class that has not been defined as a parent-class in any of the status propagation rules, the class status is set to the most severe of its child class status.
- If there are several propagation rules for a specific parent class resulting in different status, then the parent status is set to the most severe status resulting by the rules.

NOTE If specifying the threshold as a percentage, the change in the parent status takes place only when the change in the percentage of the child class objects is an integer. For example, in the default rule, the parent class object changes status color only when 51% of the child class objects have reached the status stated in this statement.

You can edit the Status Propagation Rules file as explained in the section, “Setting up Status Propagation Rules” on page 276.

Table 16-1 is an example of the information stored in the status propagation rules stated in `$FMSDIR/sampleconf/sprules.conf`. The column Threshold is used to enter the number of child class objects, whose status affects the parent class object.

Table 16-1 Status Propagation Rules Example

Parent Class	Status	Threshold	Child-Class	Child Status	And (Y/N)
Cellnet	Minor	0	EriAXE	Minor	N
Cellnet	Critical	1	EriAXE	Minor	N
Cell	Minor	0	MotorolaBTS	Minor	N
Cell	Critical	0	MotorolaBTS	Minor	Y
		0	NokiaBSC	Minor	Y
		0	NokiaBTS	Minor	N
MscPortGroup	Critical	0	MscPort	Minor	N
NokiaBTS	Critical	18%	BtsSub1	Minor	N

Setting up Status Propagation Rules

The status propagation rules file is used to maintain the rules governing object status propagation. These rules define the relation between the status of the parent object class to the status of its child object class.To customize status propagation:

1. Log in as **oemfadm** on the topology server.
2. Change directory to `$FMSETC/share/newconf`.
3. Edit the `sprules.conf` file as described below.
4. Save the file.
5. Verify that the file is error free using the status propagation rules verification utility. This utility is described under “Verifying the Status Propagation Rules” on page 281.
6. Shut down the topology server.
7. Run the `System Configuration Synchronization` utility.
8. Start up the topology server.

The file name and format are as follows:

NOTE

Comments can be included in this file on separate lines. These lines must have the # sign in the first column to indicate that they are comments. So, each line starting with # in the first column, will be taken as a comment.

All punctuations included in the syntax are mandatory.

Filename

sprules.conf

Syntax

```
parent-class IS status IF_MORE_THAN threshold
child-class SUBOBJECTS ARE_OR_MORE_SEVERE_THAN
child-status [AND IF_MORE_THAN threshold
child-class SUBOBJECTS ARE_OR_MORE_SEVERE_THAN
child-status [AND ...]]
```

Where:

parent-class Is the object class for which the rule is being defined.

NOTE

Once a rule is defined for a given parent class, the default rules no longer apply for that managed object class. For example, suppose a rule is defined to set the NokiaBTS to severity level minor if three or more of its child class BtsSub1 are of severity level major. Then, no other status will be propagated to the NokiaBTS. *Change in status of any other class of child object will not affect the status of NokiaBTS*.

Hence, you may consider defining propagation rules for all other levels of severity as well as all other child classes for NokiaBTS, as the default rules no longer apply for it.

status Is the status to which the parent object class will change. Valid values are:

- Unknown
- Critical
- Major
- Marginal (or Minor)
- Warning

threshold Can be a number or percentage (no decimals allowed) of child components that change the status of the parent object class. If this is defined as a percentage, then suffix the number (no decimals) entered with the percent symbol (%).

If the rule is stated as a percentage, when calculating the percentage of child objects, the result is rounded down to the nearest whole number. For example, take a propagation rule that states that the parent class, NokiaBTS, is critical if more than 18% of its child class, BtsSub1, is minor or more severe than minor.

If 17 of a total of 90 BtsSub1 class objects turn critical, the percentage of BtsSub1 that are critical is 18.88. Rounding down to the whole number, the percentage of child class objects that are critical is 18. So the status will NOT be propagated to the parent class object. However, when 18 BtsSub1 class objects (20%) turn critical, the parent class object will turn critical.

child-class Is the object class whose status changes the status of the parent class.

child-status Is the status of the child object class that causes a change in the status of the parent object class. Valid values for the status for the child object class are:

- Unknown
- Critical
- Major
- Marginal (or Minor)
- Warning

As indicated in the file format, you can have more than one condition for the change of status of the parent object class to a specific severity. Each of these conditions can be linked to the other using the conjunction AND in the statement, as shown in the sample below.

Example 16-1 Sample of the Status Propagation Configuration File (sprules.conf)

```
Cellnet IS MARGINAL IF_MORE_THAN 0 SECLINK SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS CRITICAL IF_MORE_THAN 1 SECLINK SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MAJOR

Cellnet IS MARGINAL IF_MORE_THAN 0 Cell SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL
```

```
Cellnet IS CRITICAL IF_MORE_THAN 2 Cell SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS MAJOR IF_MORE_THAN 0 MotorolaMSC SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS CRITICAL IF_MORE_THAN 1 MotorolaMSC SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS MAJOR IF_MORE_THAN 1 NokiaBSC SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS CRITICAL IF_MORE_THAN 1 NokiaBSC SUBOBJECTS
RE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS MARGINAL IF_MORE_THAN 0 NokiaMSC SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS CRITICAL IF_MORE_THAN 1 NokiaMSC SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS MARGINAL IF_MORE_THAN 0 LINK SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS CRITICAL IF_MORE_THAN 1 LINK SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS MARGINAL IF_MORE_THAN 0 EriAXE SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cellnet IS CRITICAL IF_MORE_THAN 1 EriAXE SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cell IS MARGINAL IF_MORE_THAN 0 NokiaBTS SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cell IS CRITICAL IF_MORE_THAN 1 NokiaBTS SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cell IS MARGINAL IF_MORE_THAN 0 MotorolaBTS SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

Cell IS MAJOR IF_MORE_THAN 0 MotorolaBTS SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL AND
IF_MORE_THAN 0 NokiaBSC SUBOBJECTS ARE_OR_MORE_SEVERE_THAN MARGINAL AND
```

IF_MORE_THAN 0 NokiaBTS SUBOBJECTS ARE_OR_MORE_SEVERE_THAN MARGINAL

```
NokiaBTS IS CRITICAL IF_MORE_THAN 18% BtsSub1 SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

NokiaBTS IS CRITICAL IF_MORE_THAN 0 BtsPort SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

NokiaBTS IS MAJOR IF_MORE_THAN 0 BtsPwrComp SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MARGINAL

NokiaMSC IS CRITICAL IF_MORE_THAN 0 MscPwrComp SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN MAJOR

MscPortGroup IS MAJOR IF_MORE_THAN 0 MscPort SUBOBJECTS
ARE_OR_MORE_SEVERE_THAN Marginal
```

Verifying the Status Propagation Rules

A utility is provided to enable you to check that the status propagation rules are correct before copying the file into the `$FMSETC/share/newconf` directory. This utility, `fmssprulesverify`, reads the file and ensures that the syntax for each rule is correct and validates all the object classes specified in the file against those configured. The utility exits on error and displays the error in the rule. The syntax for the utility is as follows:

`fmssprulesverify [-p] -f` *`filename`* <Return>

Where:

`-p`	Is used to indicate that a report be generated on the standard output - the terminal.
`-f`	Is used to name the file which must be verified.
`filename`	Is the name of the file to be verified. Enter the UNIX path of the file. This option enables you to make the changes to the status propagation rules in an alternative file and verify it before copying it to the configuration files directory.

When you execute this command, the utility reads each of the rules entered in the file checking the syntax and validating all the object classes (parent-class and child-class) against the configured object classes.

If all the rules are in perfect order, then the utility will display the following message at the end of its run:

```
SP-Rule file name is filename
SP-Rule file validated successful
Number of SP-Rules = nn
```

Where *`filename`* is the name of the file that was verified and *`nn`* the number of rules in it.

If there are any errors in the file and you had not used the -p option, then the utility will exit and display the rule that has the problem along with the error in it. For example:

```
SP-Rule file name is sprules.conf
syntax error
[ERROR]: -- At line number 1 in sprules.conf file.

[ERROR]: -- Last Token Parsed : ARE_OR_MORE_SEVEREE_THAN

SP-Rule file validated failed
```

This indicates that the line number one has an error. The error is in the syntax ARE_OR_MORE_SEVEREE_THAN. The correct syntax is ARE_OR_MORE_SEVERE_THAN. Correct the typographical error and save the file and run this utility again.

If there is an error in the object class name, then the error displayed will be as follows:

```
Rule file name is sprules.conf
[ERROR]: ClassName :[SECLINK1] is not configured.
[ERROR]: -- At line number 1 in sprules.conf file.

[ERROR]: -- Last Token Parsed : SECLINK1

SP-Rule file validated failed
```

The second line of the error message indicates the error is in the object class name. Correct the name, save the file and run the utility on the file again, till you receive the message indicating that the file validation was a success.

If you had used the -p option, then the utility will display each rule as it verifies it as shown below:

```
Rule file name is sprules.conf

1) Cellnet IS MINOR
IF_MORE_THAN 0 SECLINK SUBOBJECTS ARE_OR_MORE_SEVERE_THAN
CRITICAL

Rule file validation success
Number of Rules = 1
```

17 Configuring Operator Environments

This chapter describes the procedures to:

- Provide operator access to the iNOC Console.
- Create operation profiles.
- Assign operation profiles to users.

About the iNOC Console

The iNOC Console is a collection of graphical user interfaces through which operators:

- View all processed messages via the message browser.
- View correlated topology messages via the problems presenter.
- View the topology of the network and symbolic representation of the network objects via the map presenter.
- Set up and monitor outage plans for managed object instances on outage via the outage plan presenter.

These GUIs are described in the online help installed with the iNOC Console.

Prerequisites

The prerequisites for setting up the iNOC Console are:

- A completed installation and configuration of the OVO server.
- A completed installation and configuration of the topology server.
- A completed installation and configuration of databases.
- A completed installation of the iNOC Console components.

Smart Navigations

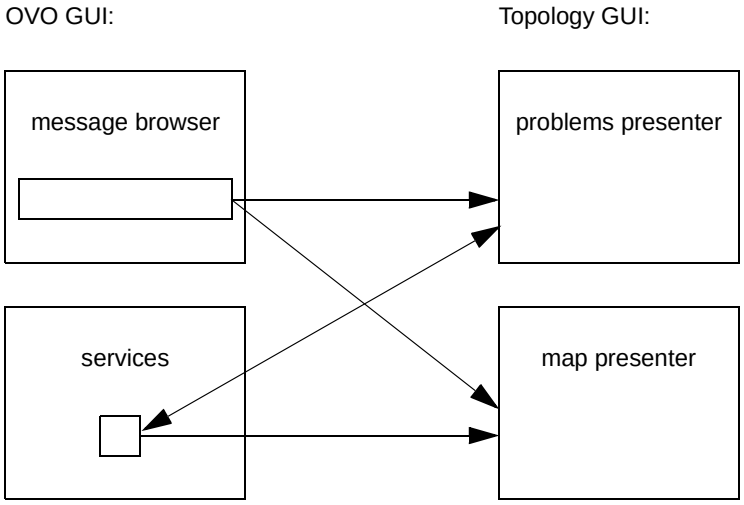
The following cross launches from the OVO operator GUI to the topology GUI are supported:

- Given a problem message in OVO, open a topology GUI map presenter containing the element or object that caused the problem.
- Given a problem message in OVO, show the events correlated in that problem and the problem history in the topology GUI problem presenter.
- Given a service in Service Navigator, open the topology GUI problem presenter with all the problems associated with elements mapped to that service.

- Given a service in Service Navigator, open a topology GUI map that shows the elements affecting that service.
- Given an element in a topology GUI map, show a list of services with which that element is associated.

Figure 17-1 shows the possible cross launches from the OVO operator GUI to the topology GUI.

Figure 17-1 Relationship Between GUIs



NOTE These smart navigations are only valid when the OVO operator GUI (Java GUI) is used. The OVO admin GUI (native GUI) does not permit launching to the topology GUI.

Users and Operation Profiles

The iNOC Console supports several types of users: integrators, administrators, and operators. Each user type may have one or more **user name**s defined to allow **users** to logon to the various GUIs: OVO operator GUI, OVO admin GUI, NNM admin GUI, or the topology GUI.

Users also have assigned to them **operation profiles**. These profiles

determine which applications, tasks, and objects users can access. Users can be assigned one or more operation profiles. In OVO, the concept of operation profiles is collective, meaning operators assigned to multiple operation profiles inherit all of the permissions of the collective profiles. In OV Topology Server, the concept of operation profiles is restrictive, menaing only one operation profile is accepted when launching the topology GUI.

Integrators and administrators can log onto the two iNOC Console using the default administrator user `opc_adm`. When the `opc_adm` user launches the topology GUI from the OVO operator GUI, he is, by default, logging onto the topology GUI as user `oemfadm` with operation profile `admin`. In OVSACN, this default administrator has full permissions and responsibilities.

Operators can log onto the iNOC Console using the default operator user `telco_op`. This user is assigned all topology GUI launching and navigation permissions. When `telco_op` user launches the topology GUI, he is logged onto the this GUI as user `telco_op` with operation profile `Telco_Op`.

It is the role of integrators/administrators to define additional user names and operation profiles.

NOTE The iNOC users must be configured with the same user name on both the OVO management server and the topology server in order to successfully launch and perform navigations from the OVO operator GUI to the topology GUI.

Topology GUI Login

When launching the topology GUI from the OVO operator GUI, OVSACN needs the following information defined:

- Topology server hostname – By default, the value of the topology server hostname is read from the first line of `hostfile.dat`.
- User name – By default, the user name is assumed to be the same as the OVO user name, except when OVO user is `opc_adm`. In this case, the default user name is `oemfadm`.
- User password – By default, the password is ignored.

NOTE

The default iNOC Console configuration relies solely on the OVO user and password authentication process to launch the topology GUI. If additional authentication is required to launch the topology GUI, then set the `FORCE_TELCO_GUI_LOGIN_AUTHENTICATION` environment variable to `Yes`. For more information about changing authentication, see "Adding Authentication" on page 324.

- Operation profile – By default, the operation profile is the first profiles in the user's assigned operation profile list.

About Configuring Operator Access

This section describes the tool used to configure new users and operation profiles on the topology server, the configuration files where additional users and operation profiles can be specified, and the iNOC default operator user and profile.

First, let's discuss the general process to configure operator access. To configure operator access to the iNOC Console:

- On the OVO management server system:
  - Create a new OVO user with the `User Profile Bank` window.
  - Create a new OVO user profile with the `User Bank` window.
  - Assign OVO user profiles to OVO user with the `User Bank` and `User Profile Bank`.
- On the topology server system:
  - Create OS user matching the name of the OVO user via SAM.
  - Create a new topology operation profile with the Operation Profile Configurator.
  - Create a new topology user matching the name of the OVO user with the Operation Profile Configurator.
  - Assign topology operation profiles to the topology user with the Operation Profile Configurator.

Operation Profile Configurator

Use the Operation Profile Configurator to define operation profiles to allow operator access to the data managed by the topology server. Operation profiles present specific information about the managed network via the topology GUI presenters.

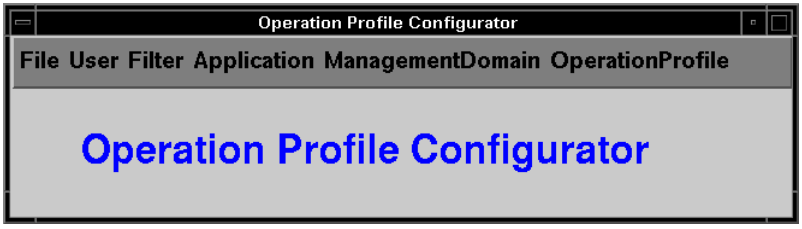
The Operation Profile Configurator can be invoked and used only by the user `oemfadm` or any user with the same user access. Before running this utility, close any topology GUI presenters that are open under the user `oemfadm`.

To invoke the Operation Profile Configuration utility, execute:

`fmsopcfg`

This command opens the Operation Profile Configurator window shown in Figure 17-2. This window includes the window title and the menu bar. The text area of the window contains the name of the configuration utility and does not accept any text entries.

Figure 17-2 Operation Profile Main Menu Screen



The Operation Profile Configurator windows consist primarily of text boxes, lists, and the following buttons as applicable:

`[Add]`	Adds the name entered in the text box to the list.
`[Delete]`	Deletes the item selected from the list.
`[Modify]`	Modifies the properties of the item selected in the list.
`[Apply]`	Saves the unsaved changes made in the current session.
`[Reset]`	Reverts the configuration to the status of the last saved configuration. All changes made since the last `Apply` function are removed.
`[Close]`	Closes the current window.
`[-->]`	Moves the selected item from the left pane to the right pane.
`[<--]`	Moves the selected item from the right pane to the left pane.

The **Operation Profile Configurator** maintains user profiles that define operator access to the topology GUI presenters and a subset of the managed network and its associated messages. Use this configuration utility to configure:

- Operation Profiles

Filters, applications, tasks, management domains, and work schedules are assigned to operation profiles. Each user can have assigned multiple operation profiles.

- Filters

  Filters restrict the operator's access to problems based on the alarm type, probable cause, specific problem, and severity. Each user can be associated with one or more filters. Multiple filters can be associated with each operation profile.

- Applications and task sets

  These are a set of tasks to which the user can be assigned access. This defines all the applications and the tasks within the applications that are available for each operation profile. The application and tasks correspond to GUI application and tasks. Applications and their tasks can be defined into `application domains`. Multiple application domains can be linked to each operation profile.

- Management Domains

  The monitored topology is divided into logical management domains for operator assignments. The managed object (MO) management domain is a domain explicitly defined by the network administrator. Each MO management domain is associated with one or more managed objects. Multiple management domains can be associated with an operation profile.

Default Operator and Operation Profile

Telco_Op is the name of the preconfigured, default topology GUI operator operation profile. This profile provides its members access to topology problems, managed objects, and outages as well as the applications and tasks that can be applied to these objects.

Operators logged on as user `telco_op` with the profile `Telco_Op` see only topology-specific messages in the OVO message browser. From the OVO operator GUI, they can launch the topology GUI presenters and send demo alarms. From the topology GUI, operators have read-only access to the maps and all topology objects in the map presenter and nearly-full access to problems in the problems presenter. Operators may also view topology outage plans applied to network elements.

When OV Topology Server is optionally installed and configured, the following OVO configuration is evident:

- `Telco iNOC` application group is created that contains applications to launch the topology GUI and its presenters.
- `Telco_Op` profile is created and added to the User Profile Bank.
- `Telco iNOC` application group is assigned to the `Telco_Op` profile.
- `telco_op` user is created.
- `Telco_Op` profile is assigned to the `telco_op` user.
- `Telco_Op` profile is assigned to the `opc_adm` user.

When OV Topology Server is optionally installed and configured, the following topology server configuration is evident:

- `Telco_Op` profile is created that provides access to problems and outages associated with all managed objects in the domain.
- `telco_op` OS user is created.
- `telco_op` user name is created.
- `Telco_Op` profile is assigned to `telco_op` user.
- `Telco_Op` profile is assigned to `oemfadm` user.

NOTE The creation and assignment of the users and operation profiles must occur on each management server system in order for the iNOC Console to function as documented.

Configuration Files for Users and Profiles

The integrated login functionality of the iNOC Console simplifies and hides the specifics of the topology GUI login from users. However, two configuration files are provided with OV Topology Server to facilitate customizing the login process and changing the operation profile to be used.

topologin.user is an ASCII file that specifies the topology GUI login information. user is the name of the user logged onto the iNOC Console. This file is optional and must reside in the $HOME directory on Unix systems and in the %TEMP% directory on NT systems.

`topologin.`*`user`* is read before the topology GUI is launched and must contain all login information required by the topology GUI, including:

- Hostname of the topology server
- User name
- User password
- Operation profile for the topology GUI

topoprofile.user is an ASCII file that specifies the operation profile to be used when launching the topology GUI. user is the name of the user logged onto the iNOC Console. This file is optional and must reside in the $HOME directory on Unix systems and in the %TEMP% directory on NT systems.

The `topoprofile.`*`user`* file should contain a single line, specifying the name of the operation profile to be used. When a `topoprofile.`*`user`* file is not present, then the user is logged on with the first operation profile configured for that user.

NOTE If both the `topoprofile.`*`user`* and `topologin.`*`user`* files are present, then the `topologin.`*`user`* file takes precedence.

Configuring iNOC Users

For access to the iNOC Console, users must set up the following information:

- A user name and password for access to the OVO operator GUI.
- A user name for access to the topology GUI that matches the OVO user name.
- An OS user on the topology server that matches the OVO user name.

Creating OVO Users

To create a new OVO user:

1. From the OVO management server, log on to the OVO admin GUI as user `opc_adm`.
2. Open the `User Profile Bank` window by clicking **`Window:User Profile Bank`**.
3. Optionally, define a new OVO user profile by:
   - Clicking **`Actions:User Profile -> Add`**
   - Assigning necessary message, application, and node groups to the new profile.
   - Assigning the `Telco iNOC` application group to the new profile, which allows the launching of the topology GUI presenters.
   - Saving the new profile
4. Click **`Window:User Bank`**.
5. Click **`Actions:User -> Add`**.
6. Assign a user name and password.
7. Assign appropriate capabilities to the new user.
8. Click [**`Profiles`**]. The `Profiles of New User` window displays.
9. Drag and drop user profiles from the `User Profile Bank` window to the `Profiles of New User` window.
10. Save and close all windows.

Creating Topology GUI Users

To access the topology GUI, a user must have the following information defined in the Operation Profile Configurator:

- OV User Name—The user ID for identifying the user within the topology server.
- User Name—The user's login ID. Each login ID is associated with a hostname.
- Host Name—The host name of the machine from which the user can log onto the topology GUI.
- Operation Profile—An operation profile must already exist in order to assign it to a user.

To create a new topology GUI user on the topology server:

1. From the topology server management server, log on as `oemfadm`.
2. Launch the Operation Profile Configurator, **`fmsopcfg`**.
3. Click **`User:User Maintenance`**.
4. Click [**`Add`**]. The `User Modification` dialog displays.
5. Click the `Operation Profile` tab to display the `User Association` page. Use this page to assign topology profiles to the new user.
6. Select an appropriate profile, and click **`-->`**.
7. Select the `Attributes` tab to display the `User Attributes` page.
8. Enter the name of the user in the `OVuser Name` text entry box.
9. Enter the topology server hostname for that user in the `Host Name` text entry box.
10. Click [**`Add`**]. This action adds the login details to the scroll box under the columns `User Name` and `Host Name`.
11. Click [**`Apply`**].
12. Close all windows and the Operation Profile Configurator.

Creating OS Users

An OS user name should be defined on the topology server with the same

name as the OVO user created on the OVO management server. To create a new OS user:

- Invoke SAM as user root.
- Double-click **Accounts for Users and Groups**.
- Double-click **users**.
- Click **Actions: Add**.
  - Enter the name of the OVO user in the Login Name field.
  - Enter the next available value for the User ID field.
  - Enter an appropriate directory for the Home Directory field.
  - Enter **oemf** in the Primary Group Name field.
  - Enter an appropriate value for the Start-Up Program field.
  - Enter an appropriate password in the Password field.
- Exit **Users**.
- Double-click **Groups**.
- Select Users group name.
- Click **Actions: Add**.
  - Enter the OS user name in the Login Name text entry box.
  - Click [**Add**].
  - Repeat as needed.

NOTE When the iNOC Console is running on Unix systems, the OS user name must also be a member of the oemf OS group in order to launch the topology GUI.

Deleting a User

To delete an operator's access to the iNOC Console:

- On the OVO management server, delete the user name from the User Bank window.

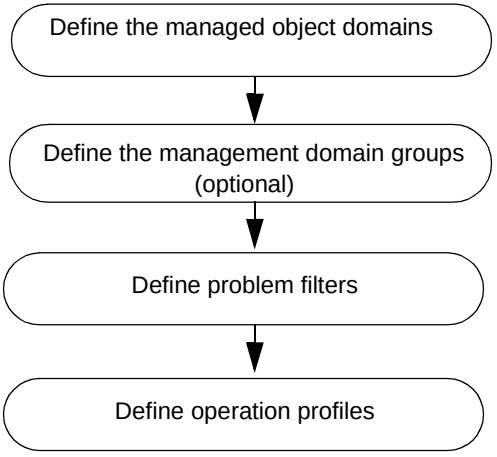
- On the topology server, delete the user name from the list of users in the Operation Profile Configurator using the function `User:User Maintenance`.
- On the topology server, remove the operator's HP-UX login ID from the topology server host.

Configuring Operator Access

Operators use the iNOC Console to monitor the function of the network and trigger applications based on the state of the network. Each operator may monitor and manage a different part of the network depending on how administrators distribute the workload of the managed network. This distribution is accomplished through **operation profiles**, or roles. Operation profiles determine which applications, tasks, and objects operators are permitted to access.

Figure 17-3 presents a flowchart of the planning steps for configuring topology access that must be completed before entering the configuration into the system. See the OVO manuals for configuring operator access to the OVO operator GUI.

Figure 17-3 Steps in Planning Operator Access to Topology GUI



Managed object domains. Managed object domains make it possible to assign multiple managed objects to an operation profile in one action. A managed object domain consists of a set of managed objects grouped together for management purposes. Any number of objects can belong to the same managed object domain. All child objects of these managed objects also belong to the same managed object domain.

Management domain groups. Like managed object domains, management domain groups make it possible to assign multiple

managed objects to an operation profile in one action. The management domain group is a management group of managed object domains. It is created only for ease of management in assigning domains to operation profiles.

Problem filters. Problem filters define the nature of alarms that operators can view.

Operation profiles. Managed object domains, management domain groups, applications, tasks, and filters are grouped together to form an *operation profile*.

Define the Managed Object Domains

A managed object domain consists of a set of managed object instances grouped together for management purposes.

You can define managed object domains using the `Domain` parameter in the `TopoData.xml` configuration file. For information on defining managed object domains with `TopoData.xml`, see "Entering Object Instance Data" on page 185.

Alternatively, you can define managed object domains with the Operation Profile Configurator. All domains defined using the `TopoData.xml` file appear in the Operation Profile Configurator in the `Available MO Management Domains` list box.

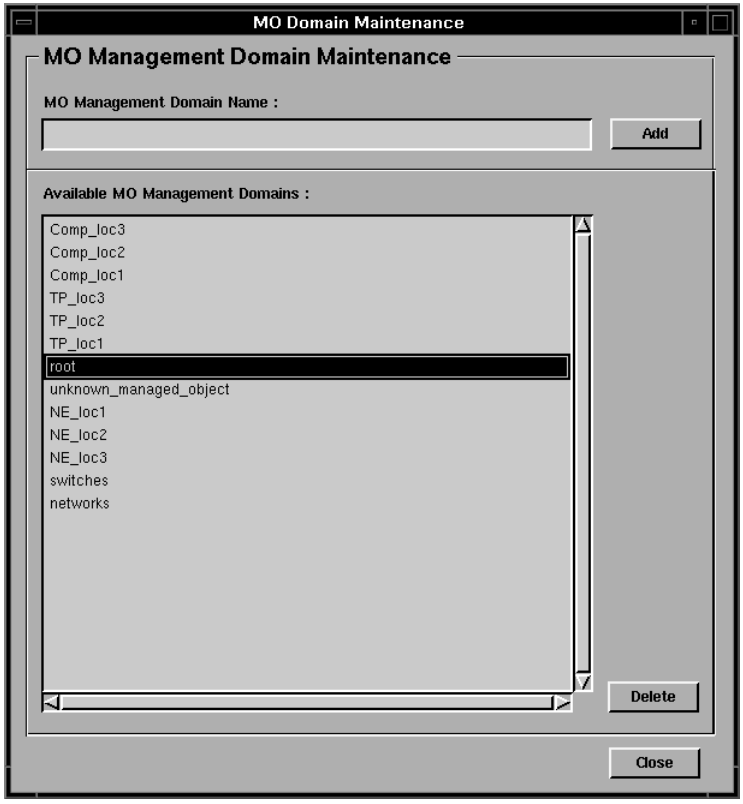
When using the Operation Profile Configurator, the information listed in Table 17-1 is required.

Table 17-1 Managed Object Domains

Parameter	Description
Managed Object Domain Names	This is the name of the managed object domain. It is suggested that logical names, by location or object type, be used for easy identification of the domain. *Note that you must not assign MOC names or managed object instance names as managed object domain names.* There is no restriction on the number of domains that can be created.
Managed Object Domain Object Names	These are the names of the managed objects contained within the managed object domain. These are the object instances as seen on the map presenter. You can list them by their shortname or map label. There is no restriction on the number of managed objects that can belong to the same domain. However, a managed object must be assigned to only one managed object domain. Any number of objects can belong to the same managed object domain. All child objects of these managed objects also belong to the same managed object domain.

To add a managed object domain, click `ManagementDomain:ManagedObject`. The window shown in Figure 17-4 displays.

Figure 17-4 Managed Object Management Domain Maintenance Window



You can add and delete management object domain names from the `Managed Object Management Domain Maintenance` window. Any action executed in this screen is immediately saved.

Two default domains, `unknown_managed_object` and root, are created during the installation process. The `unknown_managed_object` domain contains any managed object specified in a problem message that is not present in the topology database. The `root` and `unknown_managed_object` managed object domains are assigned to the `admin` operation profile as well as the `Telco_Op` operation profile.

All domains defined using the `TopoData.xml` file also appear in the `Available MO Management Domains` list box.

Adding a Managed Object Domain

To add a management domain:

1. Enter the name of the management domain in the `Managed Object Management Domain Name` text entry box in the `Managed Object Management Domain Maintenance` window.
2. Click **[Add]**.

   This action creates the management domain and adds its name to the `Available MO Management Domains` list box.

When a management domain is created in the Operation Profile Configurator, a client map named `domain-name_domain` is created in the map presenter. This map is also referred to as Managed Object Management Domain Map. It is a shared client map and is maintained by the user `oemfadm`.

Define Management Domain Groups (optional)

A management domain group is a logical grouping of managed object domains. This grouping is specifically used to assign managed object domains to operation profiles and has no other relevance. *Creating and using management domain groups is optional.*

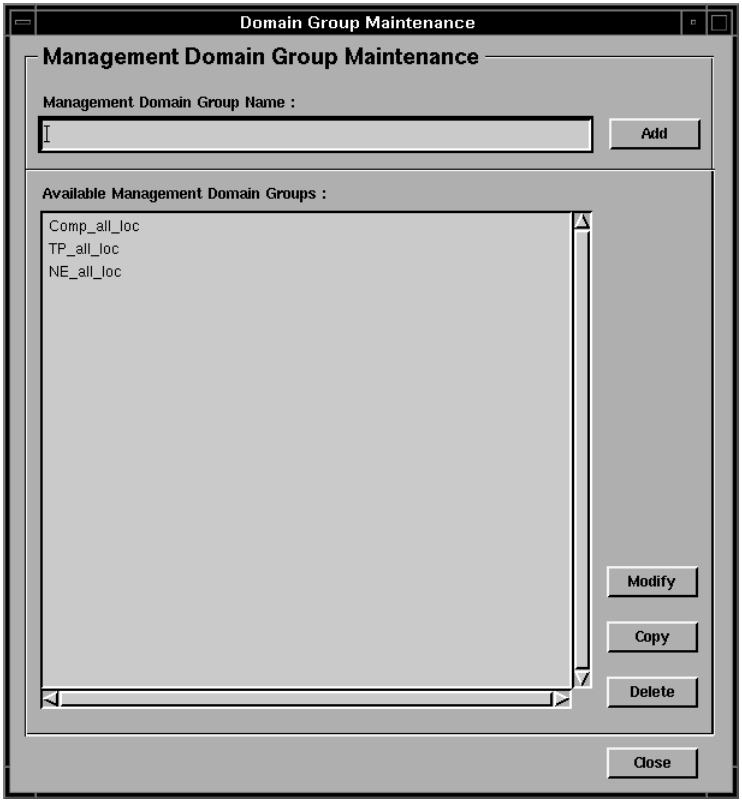
When defining management domain groups, the information listed in Table 17-2 is required.

Table 17-2 Management Domain Group

Information Head	Description
Management Domain Group	List a set of unique names for the management domain groups.
Managed Object Domain Names	These are the names of the managed object domains that belong to the management domain group defined above. These names should have been listed in the Managed Object Domains Worksheet completed in Step 1. A managed object domain can belong to more than one management domain group. There is no limit on the number of managed object domains that can belong to a management domain group.

To add a management domain group, click `ManagementDomain:Group`. The window shown in Figure 17-5 displays.

Figure 17-5 Management Domain Group Maintenance Window



Use the Domain Group Maintenance window to add, delete, and modify management domain groups. You can also copy an existing management domain group and create another group by making modifications to it. Any action executed in this screen is immediately saved.

Adding a Management Domain Group

Adding a management domain group is a two step process. Create the management domain group by adding its name. Then associate the required managed object management domains to the management domain group.

To create a management domain group:

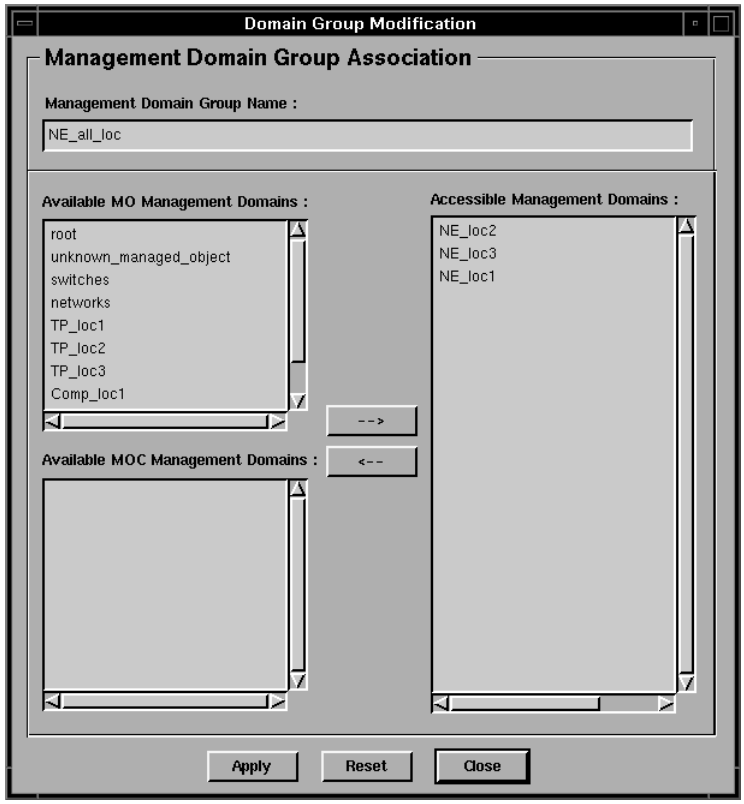
1. Enter the name of the management domain group in the Management

`Domain Group Name` edit box.

2. Click `[Add]`.

   This action creates the management domain group name in the text box and opens the `Management Domain Group Association window` shown in Figure 17-6.

Figure 17-6 Management Domain Group Association Window



This window displays the list of managed object management domains that are available.

To associate the managed object management domains with the management domain group:

1. Select the managed object management domains by name from the `Available MO Management Domains` list.

Multiple managed object management domains can be selected by pressing **Shift** and clicking on the required domain names or by clicking within the list and dragging the mouse through the list of required names.

2. Click **[-->]** to associate the managed object management domains with the specified management domain group.

   This action adds the managed object management domain name to the `Accessible Management Domains` list.

3. Click [**Apply**] at the bottom of the window to save the association of the managed object management domains with the specified management domain group.

To close this window and return to the `Management Domain Group Maintenance` window, click [Close].

You are prompted to confirm exit if you have not *applied* all of your changes.

Define Problem Filters

Within OV Topology Server, access to network problems can be defined using **problem filters**. Problem filters are defined by the following:

- **Alarm Type**. You can assign access to any or all of the five X.733 defined event types. If event types are selected as attributes, the operation profile provides access to only those alarms with the specified event type.
- **Probable Cause Type**. OV Topology Server provides message mapping by global and local variables. If probable cause types are selected as attributes, the operation profile provides access to only those alarms with the specified probable cause.
- **Probable Cause**. Among the alarm types assigned, probable causes can be selectively accessed.
- **Specific Problems**. If specific problems are selected as attributes, the operation profile provides access to only those alarms with the specified specific problem. These specific problems are defined in the Topo-Smart templates.
- **Alarm Severity**. Operation profiles can be associated with specific severity levels. If severity levels are selected as attributes, the operation profile provides access to only those alarms with the specified severity level.

Table 17-3 provides a description of the information needed to configure problem filters.

Table 17-3 Problem Filters

Information Head	Description
Filter Name	This is the name of the filter. It is internal to the Operation Profile Configurator. Enter any unique name to identify the filter you are defining.
Alarm Type	One of five X.733 event types. More than one event type can be assigned to a filter.

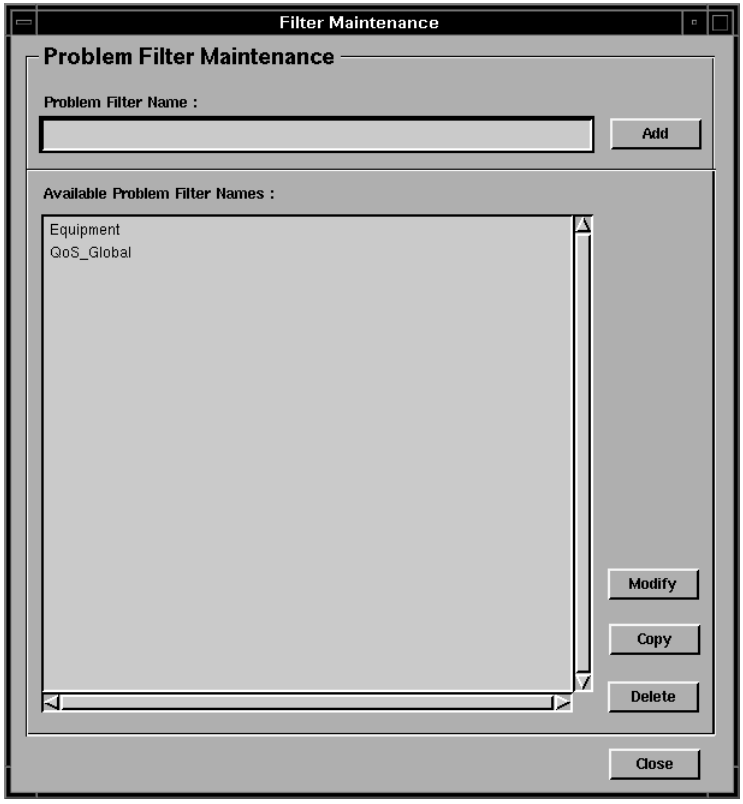
Table 17-3 Problem Filters

Information Head	Description
Probable Cause	This the probable cause that relates to the event type specified in the previous column. For each probable cause, indicate whether it is a global variable or a local one. Multiple probable causes can be selected.
Specific Problem	This is the specific problem that relates to the probable cause that is specified in the previous column. Multiple specific problems can be specified.
Severity	One of six X.733 severity levels. Multiple levels of severity can be associated with one operation profile.

To add a problem filter definition, click **`Filter:ProblemFilter Maintenance`** from the Operation Profile Configurator. The `Problem Filter Maintenance` window displays as shown in Figure 17-7.

Use the `Problem Filter Maintenance` window to add, delete, and modify problem filters. You can also copy an existing filter to create a new one by making modifications to it. Any action executed in this screen is immediately saved.

Figure 17-7 Problem Filter Maintenance Window



Adding a Problem Filter

Adding a problem filter is a two step process. Create the filter by adding its name. Then define the filter by the alarm type, probable cause, and severity details.

To create a problem filter:

1. Enter the name of the problem filter in the `Problem Filter Name` edit box.
2. Click **[Add]**.

   This action adds the problem filter name to the list and opens the `Problem Filter Association window`.

This screen has two tabs:

- `OperationProfile`

  Use this tab to associate the current problem filter with existing operation profiles. For information on using this tab, see "Associating Problem Filters with an Operation Profile" on page 319.

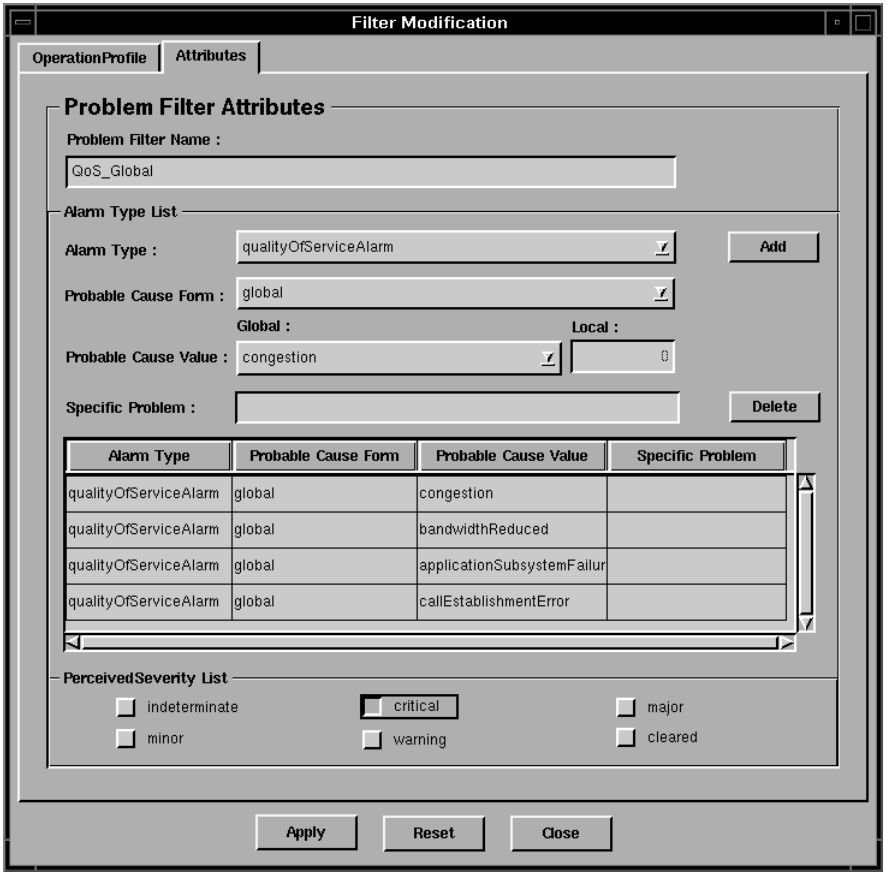
- `Attributes`

  Use this tab to define the problem filter. This tab is described below.

3. Click the `Attributes` tab to define the attributes for the problem filter.

   The window shown in Figure 17-8 displays. Use this window to define the alarm type, probable cause, and severity level associated with this problem filter.

Figure 17-8 Problem Filter Attributes Window



To associate the alarm type for this problem filter:

1. Select the alarm type from the `Alarm Type` list.

   The list contains the five X.733 event types.

2. Select the probable cause type from the `Probable Cause Form` list.

   The list contains two options: global and local. Select `global` to associate this problem with X.721 probable causes.

   Select `local` to associate this problem with M3100 probable causes.

3. Depending on the probable cause form you selected in step 2, select the probable cause value. Multiple selections are allowed.

   If you selected `global` in step 2, select the probable cause values from

the list under the `Global:` section of the `Probable Cause Value`.

If you selected `local` in step 2, enter the local probable cause values under the `Local:` section of the `Probable Cause Value`.

4. Click **[Add]** to save the association of the alarm type with the problem filter.

   The alarm type list details are displayed in the scroll box in table form.

   Repeat the above steps for all alarm type settings for this problem filter.

5. Use the `Perceived Severity List` to select the levels of alarm severity to add to the problem filter. This setting applies to all alarm types associated with the problem filter.

6. Click **[Apply]** in this window to save the additions made to the problem filter attributes.

Define Operation Profiles

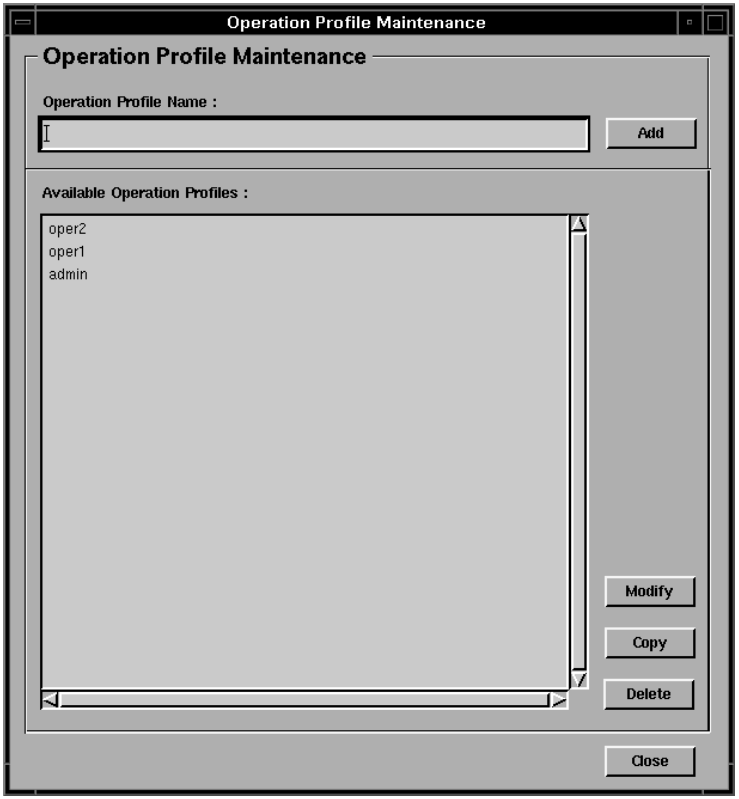
Managed object domains, management domain groups, applications, and filters are grouped together to form an *operation profile*. The types of information required to define an operation profile is described in Table 17-4.

Table 17-4 Operation Profiles

Information Head	Description
Operation Profile Name	This is a unique name for the set of functions for which access is being defined.
Application Name	A set of four applications, each providing various tasks. Operators need access to these tasks to be able to execute them. Access to the applications is assigned to the operators through the operation profile. Multiple applications can be linked to a single profile.
Task Name	These are the tasks that are associated with the application name specified in the previous column. Select the desired tasks for this operation profile. **NOTE** All operation profiles must include the `managed_object_application` with the task `get`.
Management Domain Group Name	This is the name of the management domain group. Multiple management domain groups can be linked to a single profile. If, for any management domain group, you do not want to include *all* the linked managed object domains, then you can list those you want in brackets. These can be individually assigned from the management domain group.
Managed Object Domain Name	Use this column to specify any managed object domain that you want to link to this profile that is not covered by the management domain group specified in the previous column.
Filter	Multiple filters can be linked to a single profile.

To add an operation profile definition, click **`OperationProfile:OperationProfile Maintenance`**. The `Operation Profile Maintenance Window` shown in Figure 17-9 displays.

Figure 17-9 Operation Profile Maintenance Window



The `admin` profile is automatically assigned access to all domains. This operation profile should be assigned to users who require a complete picture of the monitored network. It is recommended that no access be deleted from this user's access domain. This guards against any part of the network being left unmonitored.

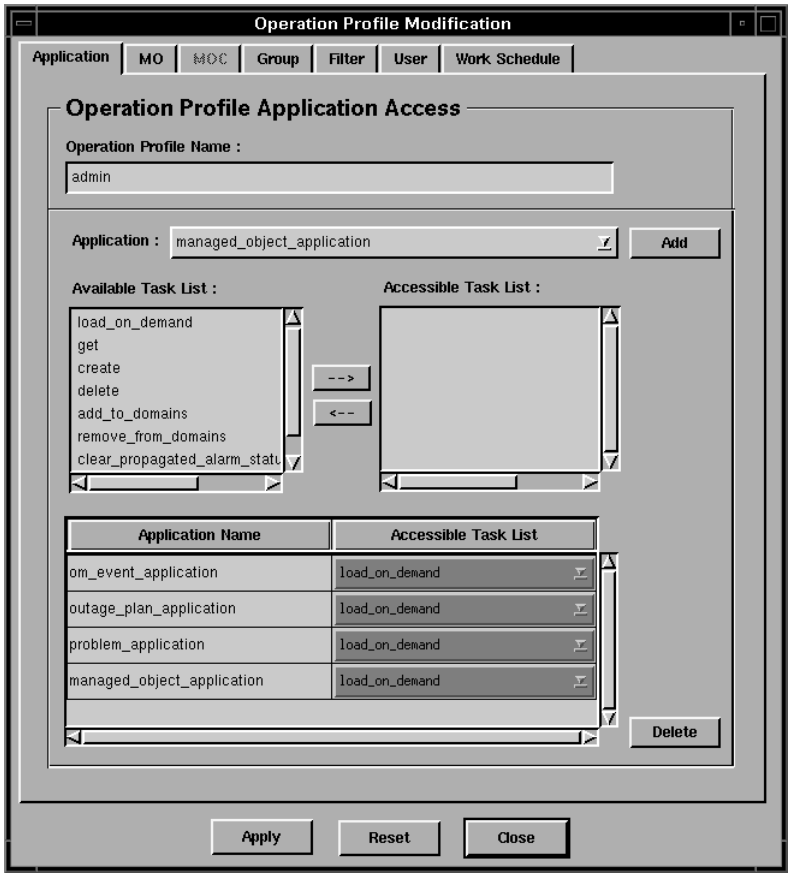
To create a new operation profile:

1. Enter a name for the operation profile in the `Operation Profile Name` text box.
2. Click **`[Add]`**.

   The `Operation Profile Modification` window shown in Figure 17-10 appears.

NOTE It is often faster to copy the `admin` profile and modify the copy to the desired settings than it is to create a new profile and assign all desired tasks.

Figure 17-10 Operation Profile Modification Window



This window has seven option tabs that enable association of access rights for the operation profile shown in the `Operation Profile Name` text box. The seven option tabs are used to associate the following information with the above named operation profile:

`Application` Applications and the tasks under them.

MO	Individual managed object domains.
MOC	Not used.
Group	Management domain groups.
Filter	Problem filters.
Users	Topology GUI users.
Work Schedule	Work schedule.

3. Save the operation profile by clicking **[Apply]** in the window. All additions, modifications, and deletions made in any of the tabs are saved.

Associating Applications with an Operation Profile

To associate an application with an operation profile, follow these steps:

1. Click the Application tab in the Operation Profile Modification window.
2. Select the application to be associated from the Application list. This list shows the applications configured with the Application:Application Maintenance function.

   There are four applications:

   - managed_object_application

     This is the application name associated with the actions taken with respect to managed objects via the managed object manager. This application opens the map presenter.

   - problem_application

     This is the application name associated with actions taken with respect to problems for managed objects via the problem manager. This application opens the problem presenter.

   - outage_plan_application

     This is the application name associated with actions taken with respect to outage plans for managed objects via the outage plan manager. This application opens the outage plan presenter.

   - om_event_application

     This is the application name associated with actions taken with respect to OM events for managed objects via the OM Event

manager. This applicationopens the OM event presenter.

For more information on the applications and associated tasks, see Appendix B , “Applications and Tasks.”

3. Select the tasks to be associated with this operation profile. Multiple selection is allowed.
4. Click **[-->]** to move the tasks to the Accessible Task List list.
5. Select any tasks that must be removed from the Accessible Task List list, and click **[<--]** to move the tasks back to the Available Task List.

   This actions makes the tasks unavailable to the current operation profile.
6. Click **[Add]**.

   This action copies the application name and the associated tasks to the lower list. The information is listed under the column heads Application Name and Accessible Tasks List.

Repeat steps 2 through 6, until all required applications and tasks have been associated with the operation profile being defined.

WARNING **Do NOT modify or delete the applications and their tasks under any circumstances.**

Associating Management Domain Groups with an Operation Profile

To associate a management domain group with an operation profile:

1. Click the Group tab in the Operation Profile Modification window.
2. Select the management domain groups to be associated with the operation profile from the Available Management Domain Groups list. Multiple selections are allowed.
3. Click **[-->]** to move the selected management domain groups to the Accessible Management Domain Groups list.

   All management domain groups listed in the Accessible Management Domain Groups list are accessible to the operators linked to this operation profile being configured.

Although many management domain groups are no longer listed in the Available Management Domain Groups list, they are still available to other operation profiles. A management domain group can belong to more than one operation profile.

4. Select any management domain groups that you do not wish to associate with this operation profile from the Accessible Management Domain Groups list, click **[<--]** to move them out of this list and into the Available Management Domain Groups list.

   This action makes the selected management domain groups unavailable to the current operation profile.

Associating Managed Object Domains with an Operation Profile

There may be some managed object domains, not covered by the management domain groups, that must be associated with the operation profile. The process of adding the managed object management domains to the operation profile is similar to that of adding the management domain groups.

To associate these managed object domains individually to the operation profile:

1. Click the Managed Object tab in the Operation Profile Modification window.
2. Select the managed object management domains to be associated with the operation profile from the Available MO Management Domains list. Multiple selection is allowed.
3. Click **[-->]** to move the selected managed object management domains to the Accessible Management Domains list.

   All managed object management domains listed in the Accessible Management Domains list are accessible to the operators linked to this operation profile.

   Although many managed object management domains are no longer listed in the Available MO Management Domains list, they are still available to other operation profiles. A managed object management domain can belong to more than one operation profile.
4. Select any managed object management domains that you do not wish to associate with this operation profile from the Accessible Management Domains list, and click **[<--]** to move them out of this list and into the Available MO Management Domains list.

This action makes the selected managed object management domains unavailable to the current operation profile.

Associating Problem Filters with the Operation Profile

The process of adding problem filters to the operation profile is similar to that of adding the management domain groups and managed object management domains. To associate problem filters with the operation profile:

1. Click the `Filter` tab in the `Operation Profile Modification` window.
2. Select the problem filters to be associated with the operation profile from the `Available Filters` list. Multiple selections are allowed.
3. Click **`[-->]`** to move the selected problem filters to the `Accessible Filters` list.

   The problem filters in the `Accessible Filters` list are used for the operators linked to this operation profile.

   Although many problem filters are no longer listed in the `Available Filters` list, they are still available to other operation profiles. A problem filter can belong to more than one operation profile.
4. Select any problem filters that you do not wish to associate with this operation profile from the `Accessible Filters` list, and click **`[<--]`** to move them out of this list and into the `Available Filters` list.

   This actions makes the selected problem filter unavailable to the current operation profile.

Associating Problem Filters with an Operation Profile After you have created problem filters and operation profiles, you can associate operation profiles with a problem filter and you can view the operation profiles that are currently associated with a filter.

To associate operation profiles with problem filters:

1. Click **`Filter:ProblemFilter Maintenance`**.

   The `Problem Filter Maintenance` window appears.
2. Select the problem filter of interest from the `Problem Filter Name` list.
3. Click **`[Modify]`**.

The Problem Filter Modification window displays.

4. Click the OperationProfile tab to view the associated operation profiles.

   The Available Operation Profiles list shows the operation profiles currently configured in the system. The Associated Operation Profiles list shows the operation profiles associated with the problem filter.

Associating Users with an Operation Profile

The process of associating users to the operation profile is similar to that of adding problem filters.

To associate users with the operation profile:

1. Click the User tab in the Operation Profile Modification window.
2. Select the user names to be associated with the operation profile from the Available Users list. These are OV user names created using the User Maintenance function.

   Multiple selection is allowed.
3. Click **[-->]** to move the selected user names to the Accessible Users list.

   The user names in the Accessible Users list are linked to this operation profile.

   Although many user names are no longer listed in the Available Users list, they are still available to other operation profiles. Users can linked to more than one operation profile.

   This User tab enables you to associate the currently opened operation profile with multiple users.

   To associate multiple operation profiles with a user, use the User:User Maintenance function.
4. Select any user names that you do not wish to associate with this operation profile from the Accessible Users list, and click **[<--]** to move them out of this list and into the Available Users list.

   This actions disassociates the user from the current operation profile.

Associating Work Schedules with an Operation Profile

Each operation profile can have a work schedule defined for it. The work schedule covers the following information:

- Whether there is a time restriction for operators using this profile. If there is a time restriction, the period during which this profile is accessible should be defined. *Operators accessing the topology GUI with this profile can log on and initiate any function only during the specified period.*
- If there is a time restriction, the time zone for the timing provided.

Define Work Schedule You can specify the following information regarding the operator login:

- Days of the week when the operator can log on.
- Time of the day when the operator can log on.
- The time zone associated with the above schedule.

Define the work schedule for the operation profile.

Table 17-5 describes the information needed to define a work schedule:

Table 17-5 Work Schedule

Parameter	Description
Operation Profile Name	This is the name of the operation profile for which the work schedule is being defined.
Time Zone	Indicates the time zone to which this work schedule relates.
Login Schedule	This time restriction for the operational profile is optional. You can restrict the access of the operators linked to this operation profile by specifying the day, and period within the day when this operation profile is operational. *Operator can trigger operations only during the specified period.* After the operator's scheduled time passes, the operator cannot initiate any actions. Specify the login schedule stating the following information: • Days of the week when this operation profile is operational. • The start time and the end time for the period during the day when this operation profile is operational.

Associating Work Schedule with Profiles To associate a work schedule with the operation profile:

1. Click the `Work Schedule` tab in the `Operation Profile Modification` window.
2. Select the `Time Restriction` option button.
3. For each time window, enter the start time and the end time by the day of the week and the time. Use the arrow keys beside the selection boxes to select the appropriate times.

   For each time period, click **`[Add]`**. The time window appears in the scroll box.

   If required, you can delete any time definition. Select the timing from the scroll box and click **`[Delete]`**.

   After defining the access time, save the work schedule.

NOTE

The operator can only log in during the period for which the operator is provided access. If the access time expires while the operator is logged in, the operator will not be able to make any updates through the GUI Client presenters.

4. Select the time zone from the `Timezone` list.

Assigning Profiles to Users

To assign operation profiles to the operators, use the User:User Maintenance function or the OperationProfile:Operation Profile function of the Operation Profile Configurator to link the operator's user name (login ID) to the operation profiles.

Associating Operation Profiles with Users

After you have created operation profiles, you can associate operation profiles with users. You can also view the operation profiles that are currently associated with a user.

To associate operation profiles with a user:

1. Click **User:User Maintenance**.

   The User Maintenance window displays.

2. Select the user name of interest from the User Names list.

3. Click **[Modify]**.

   The User Modification window appears.

4. Click the OperationProfile tab.

   The Available OperationProfile list shows the operation profiles currently configured in the system. The Associated OperationProfile list shows the operation profiles associated with the OV User.

   This function is very useful when creating new users and maintaining them. You can select multiple profiles from the Available OperationProfile list and use **[-->]** to make them accessible to the selected OV user.

   The association is saved on clicking **[Apply]** and is immediately effective.

Customizing Topology GUI Login

To customize the topology GUI logon process add a `topologin.<user>` file or a `topoprofile.<user>` file in the system. Use the `topologin.<user>` file to log onto the topology GUI with a different user name than the OVO user login or to select an operation profile other than the default. Use the `topoprofile.<user>` file to log onto the topology GUI with a different operation profile than the default operation profile configured for that user.

Adding Authentication

By default, no additional login authentication is required by operators to launch the topology GUI from the OVO operator GUI. Administrators can add additional authentication by doing the following configuration steps:

- To add authentication, modify either the `guicstart` script on UNIX systems or the `%OVCAROOTDIR%\bin\start_ovcsa_gui.bat` script on NT systems to include:

  ```
  export FORCE_TELCO_GUI_LOGIN_AUTHENTICATION=yes
  ```

- Decide whether the authentication should be command line driven or via an interactive login GUI.

For command line authentication:

- Edit `%TEMP%/topologin.<user>` file to contain the topology GUI hostname, OS user name, OS password, and operation profile. Each of the entries should be included in the order specified and on a separate line.
- When the `topologin.<user>` file is present on the system, OV Topology Server reads this file to launch the topology GUI.

For an interactive login GUI authentication:

- On the OVO admin GUI, register a new application or change the executable of the existing `Launch UX GUI` function to `/opt/OEMF/V5.0/GUIC/oemf/util/login_telco_gui.ksh`. This displays the topology GUI login panel.
- On NT systems, register a new application or change the executable

of the existing `Launch NT GUI` function to `%OVCAROOTDIR%\bin\login_telco_gui.bat`. This displays the topology GUI login panel.

- Unregister any unwanted GUI launch applications.

NOTE When `FORCE_TELCO_GUI_LOGIN_AUTHENTICATION` is on, and neither `topologin.<user>` file is present nor the GUI launch application was changed, then authentication fails.

Changing Topology GUI profiles

By default, the topology GUI autoselects the first operation profile defined for that user. If an operator prefers to launch the topology GUI using a different operation profile, then edit the `%TEMP%/topoprofile.<user>` file. In this file, specify only the operation profile to be used when the topology GUI is launched from the OVO operator GUI.

18 Additional Topology Setups

This chapter provides instructions on some of the common OVO tasks needed during the configuration process as well as other configuration steps you may want to employ for your managed network.

Configuring Message Groups

The Message Groups window contains symbols for the message groups for which a particular operator is responsible. In this window, you can review the status of each group and select specific groups for message review. OVO uses message groups to combine management information about similar or related managed objects under a chosen name, and provide status information on a group level.

Message groups are a convenient way of categorizing messages. Messages belonging to the same function or task can be collected into a single group. For example, the message group Backup can contian all messages that relate to the backing up and storing of data, such as message originating from a netowrk backup program, pieces of hardware used in the backup program, and so on.

Message groups are then assigned to operators, who see and manage only those groups assigned to them. Messages are organized into groups to simplify message management, and to let operators work in a task-oriented way. For example, one operator can be responsible for backups and output, while another operator can be responsible for security aspects.

To add a message group to the Message Group Bank:

1. Click **Window: Message Group Bank**. The ITO Message Group Bank window displays.
2. Click **Action:Message Group -> Add** to add new message groups more relevant to your environment. The Add Message Group window displays.
3. In the Add Message Group window, the name, label, and description of hte new message group. Be sure that the new message group does not conflict with an existing OVO message group.
4. Click [**OK**].

Enable Message Stream Interface

1. Start HP OpenView Operations.
2. Go to the `ITO Node Bank` window.
3. Select the managed node on which to enable the MSI.
4. Right-click on the managed node to display its menu.
5. Click `Modify...` to open the `Modify Node` dialog box.
6. Click `Advanced Options...` to open the `Node Advanced Options` dialog box.
7. Under the `Message Stream Interface` option area, check `Enable Output`.
8. Click `Close`.
9. Click `[OK]` to enable the MSI. Shortly after this point you should see a message in the message browser indicating that the agent system has been updated.

Configuring Maps in Map Presenter

A selection of maps is created automatically when the project XML configuration files are deployed and applied:

- Server maps are created based on the actual topology in the containment hierarchy.
- One shared, client-based domain map is created for each domain defined in XML configuration files.

The administrator or operator can create additional client and filter maps that provide a logical view of the managed objects and include all objects that exist below the specified origin and pass the filter criteria, respectively.

Map Ownership

By default, maps created through the map presenter are available only to the user and role of the session when the maps are made. To make a map available to all roles, toggle the `MapGadget:For All Roles` menu command.

Maps created by the user `oemfadm` with the role `admin` are available as shared maps to all users.

Maps created by the user `oemfadm` are available as shared maps to all users with the role of the session when the maps are made.

Creating a Filter Map

Filter maps are created with the map presenter. A filter map displays all server objects below a specified origin that pass the user-defined filter criteria. Hierarchical relationships are not displayed. For example, the filter map can display all objects in the managed network that have a perceived severity of critical.

To create a filter map:

Step 1. Launch the map presenter.

Step 2. Check the box for `Open New Window.`

Step 3. Create an empty filter map:

1. Click `File:New`.
2. In the New Map dialog box, enter the name for the filter map to be created (for example, **`BSCFilterMap`**).
3. For Map Class, select `FilterMap` from the list.
4. Click `[Apply]`.

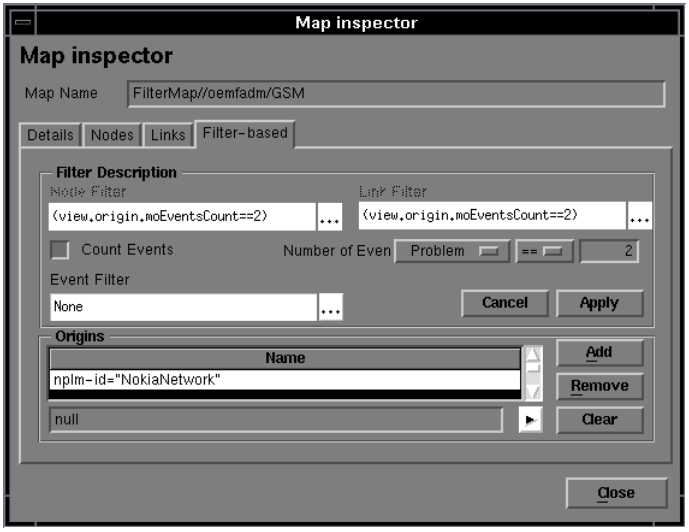
An empty Filter map appears.

Step 4. Click `MapGadget:Map Inspector`. For detailed information about the Map Inspector, see the online help.

Step 5. Click the `Filter-based` tab in the `Map Inspector` window.

Step 6. Specify the filter and origin for the Filter map.

Figure 18-1 Filter Based Map Attributes



For example, to filter map objects with the object class name BSC:

1. Click [...] in the Node Filter area to display the Filter Panel.
2. In the Filter Panel click or, then click Operators:Binary->==.
3. Double-click the upper _undef_, then select MOCName from the list.
4. Double-click the second _undef_, then enter **"BSC"** (include the double quotes), then press **Enter**.
5. Click [Parse].
6. Click [Apply] and [Close].
7. Click [Apply] in the Filter Description frame.
8. Enter the origin of the map. (The origin is the FDN of the root of the Filter map.)
9. Click [Add].

   Objects appear in the newly created filter map.
10. Click File:Save to save the map and the filter description.

The default label for objects displayed in the Filter map is the RDN.

Federation Entitlement

One of the salient features of OV Topology Server is its ability to handle more than one million distributed objects over a distributed object management system. OV Topology Server is inherently scaleable without compromising the existing installation. OV Topology Server manages objects transparently by distributing them over multiple topology servers.

The use of multiple topology servers necessitates planning the object distribution before installing and configuring the topology server.

Network Distribution Model

Installation is a term used within OVSACN to descibe an entire configured managed network. An installation can manage up to a million object instances and may include up a maximum of twenty FM Servers. To support these large numbers of managed objects, OV Topology Server requires a logical grouping of managed objects within an installation. Managed objects are grouped into logical partitions. Machines in the installation are grouped into locations. These terms are explained in detail before describing the object model to be used.

Location: This is a cluster of one or more machines offering OV Topology Server services. Thus, each location consists of:

- One FM Server
- One GUI Server
- One or more topology GUIs
- Applicable third-party software

OV Topology Server and third party application software execute in a location. An installation can have one or more such locations. The OV Topology Server and third-party application software is replicated in all locations of the installation.

Each location is autonomously manageable and is capable of operating independent of the other locations. However, as complete service is provided by all locations cooperating together, degradation of services occurs when a location is taken out of service.

In each installation, one location is designated as the **Primary Location**. The primary location is the administrative host of the entire installation. All information is configured and distributed from the primary location. The primary location is not functionally different from any other location.

The name of each location is the UNIX hostname of the machine on which the FM Server executes and is determined and assigned when installing the FM Server software.

Each location can comfortably handle up to ten topology GUIs linked to one GUI Server. Each GUI Server is linked to only one FM Server, though they do communicate transparently to the installation.

Each location can have one or more telecom subagents linked to one FM Server.

All managed objects linked to a telecom subagent serviced by a location belong to that location, and are serviced by its FM Server. All managed objects are logically categorized into partitions. Each location provides and complements the service operations pertaining to the partitions (the managed objects) that it supports.

Partition: This is a subset of the managed objects in an installation belonging to a single location. Partitions can be as large as the location. Hence, when defining the managed objects that belong to a partition, the limit to the number of objects in the location should be observed. The smallest partitionable object type is a network equipment.

An object can belong to only one partition. However, many managed objects can belong to a single partition. Each partition is supported by only one FM Server, and, therefore, belongs to only one location.

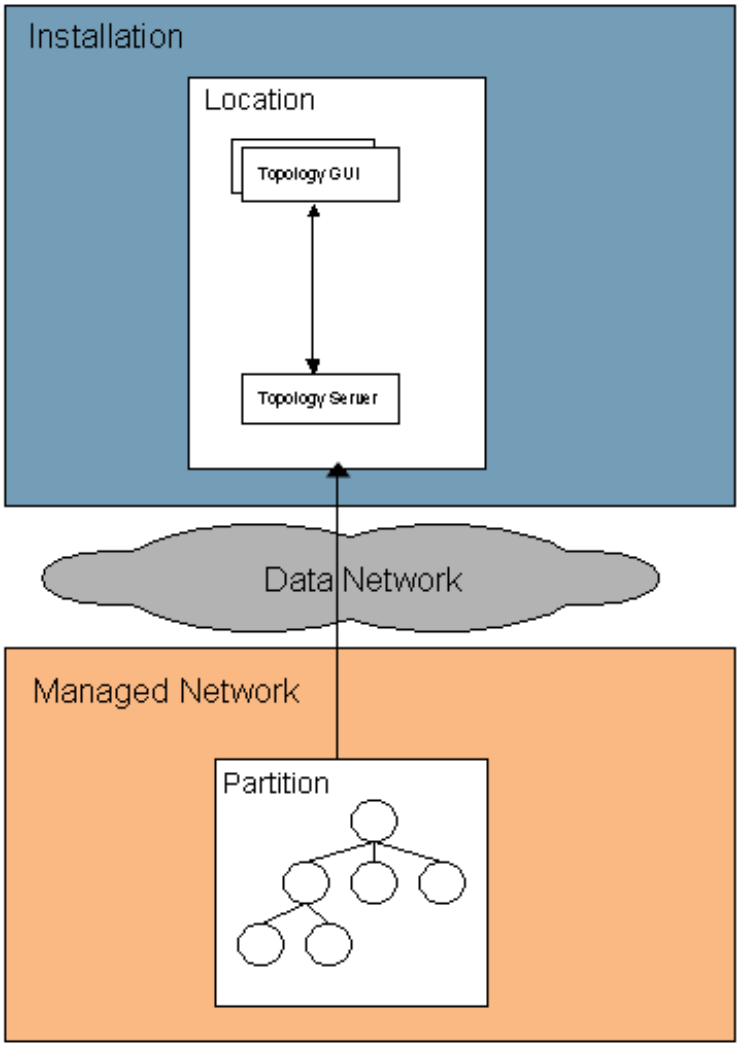
A partition is defined by identifying a managed object as the `root` of the partition. All managed objects contained, either directly or indirectly under the partition `root` belong implicitly to that partition. The implicit membership is terminated when a managed object contained under the `root` is explicitly defined as the `root` of another partition.

For example, suppose an object A contains an object B that contains the objects C and D. Suppose A and D are defined explicitly as the roots of two partitions. Then B is directly contained under A and is a member of partition A. C, being contained under B, is indirectly contained under partition A and is an implicit member of that partition. Now D, being contained under B, would have been an implicit member of partition A. However, as D is explicitly defined as the root of a partition, it is no

longer part of partition A.

Figure 18-2 shows a sample partitioning of an installation.

Figure 18-2 Sample Installation Showing Partitions



Partitions are an internal concept for grouping managed objects and are totally transparent to operators. This concept is used only for the configuration and administration of an installation.

19 Troubleshooting

This chapter details the utilities and programs to be run and files to be checked should there be any problem while running OVSACN. 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.
- Tracking issues when messages are not forwarded to the iNOC Console.
- Checking alarm message configuration.

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 *HP OpenView Service Assurance for Communication Networks Release Notes* and the available HP OpenView software patches that solve some of the known product defects.

Prerequisites

This section assumes that OVSACN is installed according to your configuration type using the instructions outlined in *HP OpenView Service Assurance for Communication Networks Installation Guide*, with the OVO admin GUI, OVO operator GUI, and topology GUI available to you.

Viewing Log Files

Table 19-1 provides the location and description of the some of the log files associated with HP OpenView Service Assurance for Communication Networks.

Table 19-1 Useful Log Files

Log File	Description
`/var/adm/syslog/syslog.log`	System log file.
`/var/opt/OV/log/Telco/ovtopoinjector.log`	Injector log file.
`/var/opt/OV/log/Telco/ovtopodivertor.log`	Diverter log file.
`/var/opt/OV/log/Telco/ovtopodivertor.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/log/OpC/opcmsglg`	OpC message log.
`/var/opt/OV/log/Telco.ovtopoadaptor.log`	Telecom adaptor log file.
`/var/opt/OV/log/Telco/ovtopoadaptor.trc`	Telecom adaptor trace file.
`/var/opt/OV/log/Telco/ovtopoadaptor.pid`	Process ID of the running telecom adaptor.
`/var/opt/OV/share/log/Telco/svc2map.log`	Service to element map navigation log file.
`/var/opt/OV/share/log/Telco/opc2sa.log`	Message to problem navigation log file.

Table 19-1 Useful Log Files

Log File	Description
`/var/opt/OV/share/log/Telco/sysName/OpC2SAIntegApp/`	Contains additional message to problem navigation log files.

Troubleshooting Message Delivery

This section captures some helpful troubleshooting information related to issues with the delivery of messages within OVSACN. It attempts to divide the problem and then list the most likely causes of particular areas known to cause problems. The focus here is specifically on message delivery issues that are likely caused by configuration issues.

Initial Troubleshooting of Entry Configurations

When the OV Topology Server is not installed and configured, messages collected and processed at the agent level flow directly to the OVO message browser. If the messages you are attempting to deliver do not appear in the OVO message browser, see "Troubleshooting the Data Collector" on page 352. If messages appear the OVO message browser, but table lookup processing (`$SELECT( )`) is not parsing messages as expected, see "Troubleshooting the Telecom Diverter and Injector" on page 350.

Initial Troubleshooting of Standard Configurations

When the optional OV Topology Server is installed and configured, messages that are topology-related are expected to be diverted and injected into the topology server for further processing. If messages are not delivered to the topology GUI or the OVO message browser, there are several initial areas to check to help narrow the problem:

- If the message displays in the topology problem presenter, but does not display in the OVO message browser, see "Troubleshooting the Adaptor" on page 347.
- If the message displays in the OVO message browser, but not the topology GUI, see "Troubleshooting the Topology GUI" on page 350.
- If the messages do not display in either GUI, disable the MSI interface for the agent node (meaning, the node where the messages are being inserted), and regenerate the messages.

  To disable the MSI interface in the OVO admin GUI:

  — Locate the agent system in the node bank.

  — Right-click on the node.

— Click [**Modify**]. The Modify Node window displays.

— Click [**Advanced options**]. The Node Advanced Options window displays.

— Uncheck Enable Output to disable the agent MSI interface.

— Click [**Close**] to close the Node Advanced Options window.

— Click [**OK**] in the Modify Node window to accept changes.

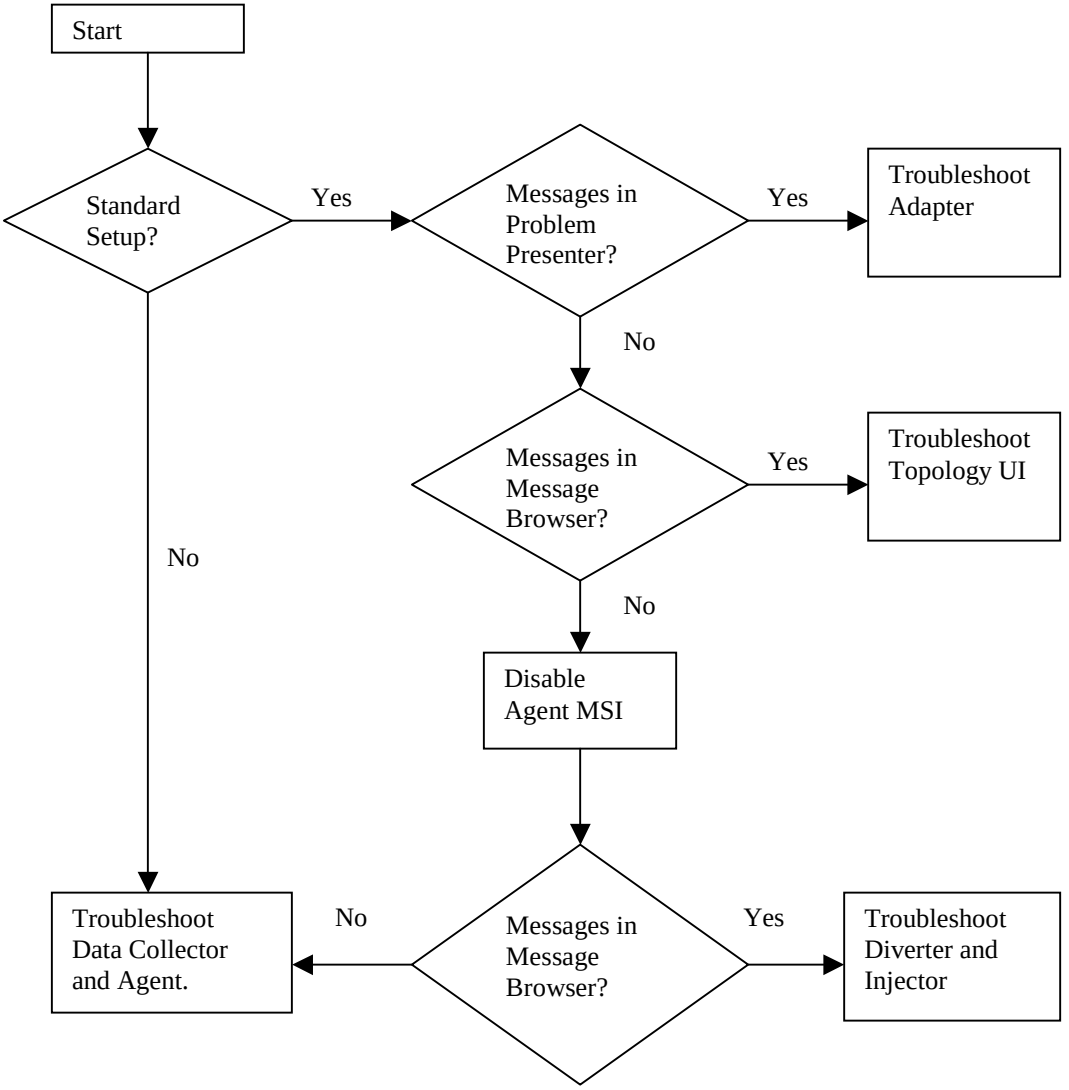
If the messages are delivered successfully, see "Troubleshooting the Telecom Diverter and Injector" on page 350.

If the messages are not delivered successfully, see "Troubleshooting the Data Collector" on page 352.

IMPORTANT Re-enable the agent MSI interface with the Divert Messages mode selected after troubleshooting.

See Figure 19-1 for a pictorial representation of the steps to follow when initially troubleshooting your configuration.

Figure 19-1 Initial Troubleshooting



Troubleshooting the Adaptor

This section lists areas to check if the message displays in the topology GUI, but does not display in the OVO operator GUI.

The most common failures are related to template deployment and user responsibilities configuration.

Adaptor Checklist

Check to see that:

- `ovtopoadaptor` is executing. Check by executing `ovstatus ovtopoadaptor`.
- `TelcoTopology` templates are assigned and installed on the management server by executing `/opt/OV/bin/OpC/opctemplate` on the management server and looking for the `TelcoTopology` templates (`TelcoProblems` and `TelcoServiceStatus`).
- User `opc_adm` has the `Telco_Op` profile assigned.
- The user logging onto the iNOC Console has the correct profiles and permissions assigned.
- `Telco_Op` profile has the correct message groups and node groups assigned.
- The target node exists in the appropriate node group.
- The user session is restarted after responsibilities are changed.
- A correct mapping exists for the node in the FDN-to-shortname mapping, and the `Mappings.xml` has been deployed and applied on the OVO management server.
- The target node is a `Message Allowed` managed node type. Click `Actions:Node->Modify` in the node bank. Click `Message Allowed` in the `Type of Managed Node` pane.
- The topology server processes are executing. Check by executing `ovsa_admin –status` on the topology server.
- Management server agent processes are executing. Check by executing `opcagt –status`.
- For service messages only, the service is assigned to the appropriate user responsibilities.

Other tasks to perform on the adaptor in order to get messages to display

in the iNOC Console include:

- Stopping and restarting the adaptor by executing: `ovstop ovtopoadaptor`; `ovstart ovtopoadaptor`
- Checking `/var/opt/OV/log/Telco/ovtopoadaptor.log` for errors. See `ovtopoadaptor(1m)` reference page for responding to particular error messages.
- If the topology database is changed while the problem manager is stopped, stop and restart the adaptor.

Adaptor Templates

For the adapter to function correctly, the `TelcoTopology` template group must be assigned and installed on the management server's agent. To verify this:

- Select the node from the node bank.
- Click **`Agent:Assign Templates`**. You should see the `TelcoTopology` template group listed in the `Define Configuration` window. If `TelcoTopology` template group is not listed, then click [**`Add`**] and locate the `TelcoTopology` template group.
- Click **`Agent:Install/Update SW & Config`** to install the assigned templates on a managed node. You must re-install assigned templates after any modifications to the templates are made.
- To view a list of installed templates on a managed node, execute `/opt/OV/bin/OpC/opctemplate` from the agent system. Be aware that this command does not indicate the current version of the templates.

Additionally, you can modify the `TelcoTopology` templates to enable agent message logging. Open the `Message Source Template` window. Click `Telco` from the `Template Groups` pane, and double-click the `TelcoTopology` template group. Select a template and click [**`Options`**] to open the `Options` window. Check all desired logging options.

Agent incoming message handling is logged in the `/var/opt/OV/log/OpC/opcmsglg` file. Check to see if the messages are being received in the agent on the management server. Look for messages where the message text starts with `MSG TEXT` or `Event Type`. If the messages are being injected to the agent, then it is either an OVO user responsibilities problem (see "OVO User Responsibilities" on page

349) or a node assignment problem (see “Node Assignment” on page 349).

OVO User Responsibilities

There are two sets of OVO users that must have the required permissions to view topology-related problems. The first is the default administrator user `opc_adm`. The adaptor connects as the user `opc_adm` to perform various operations. Second is the OVO user name connecting to the OVO server. If this user's display responsibilities are not correct and the `opc_adm` responsibilities are correct, then messages are correctly synchronized by the adaptor, but do not display for the target user.

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.
- The node group of interest is assigned to the user and the user profile. See “Node Assignment” on page 349.

Node Assignment

The preferred method for assigning the node for an incoming message is via the FDN-to-Node mapping configured in the `Mappings.xml` file. Make sure that a mapping exists for the network element associated with the problems in question, and that a node exists in the node bank for that mapping. The node is created during the deploy/apply processing, so the lack of a node can indicate that the apply process is incomplete. If no mapping exists in the `Mappings.xml` file, the network element shortname is used.

If the problem exists in the topology GUI problems presenter, select the problem and click [`Details`]. The managed object details section provides the managed object instance and network element shortname for the target element.

Note that if the target node does not exist in the OVO node bank, the message is silently dropped by OVO. In this case, a warning message appears in the `ovtopoadaptor.log` file that indicates the message was not activated in OVO within a five minute period.

Troubleshooting the Topology GUI

If messages display in the OVO message browser, but not the topology GUI problems presenter, verify the following:

- Are the templates processing the messages as Topo-Smart messages? See "Modifying Topo-Smart Templates" on page 155 for information on how to define Topo-Smart templates.
- Is the MSI interface enabled on the agent node? For instructions, see "Enable Message Stream Interface" on page 330
- Are the templates assigned to the agent node Topo-Smart templates?
- Is the `Message Type` field in the message source templates set to `Telco_<something>`?
- Do the templates have the agent MSI interface enabled and set to `Divert Messages`. For instructions, see page 158.
- Is the problems presenter filtering enabled and filtering out the target problems?
- Is the `Show all` option checked in the problems presenter?

Troubleshooting the Telecom Diverter and Injector

The telecom diverter, with process `ovtopodiverter`, is responsible for listening on the agent message stream interface (MSI) for topology related messages, processing them for table lookups and topology related fields, and then forwarding topology-related messages to the telecom injector. The telecom injector delivers the messages as alarms to the topology server.

Telecom Diverter and Injector Checklist

Check the following if you believe that the diverter and injector should process messages, but messages are not being properly handled:

- Verify that both the telecom diverter and injector are running by executing `opcragt -status -id 11 <agent hostname>`.

- Verify that all topology services are running on the topology server by executing `ovsa_admin -status`.
- Check that the agent MSI is enabled for the agent node.
- Check that the agent MSI is enabled with `Divert Messages` set in the appropriate template conditions. For instructions, see page 158.
- Check that the `Message Type` field is set to `Telco_<`*`something`*`>` in the appropriate templates.
- Check that the templates have correctly tagged and formatted topology fields. See "Topo-Smart Templates" on page 351 for troubleshooting information and "X.733 Fields to XML Elements" on page 156 for instructions on adding the appropriate XML tags.
- Assign and reinstall the templates.
- Verify that the topology server hostname variable, `TOPOSRV_HOST`, is set correctly in `/etc/opt/OV/Telco/telco.env`. If not, you need to execute the `ovtelco.setup` on the management server and re-apply your agent installation.
- 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.

Topo-Smart Templates

Topo-Smart templates require particular care when creating. In addition to the various conditions required so the telecom diverter sees the messages, the message text produced by the Topo-Smart templates must be in a particular format for the telecom injector. Verify the following:

- The output `Message Text` field has `!:::!` followed by a correctly encoded XML document. For instructions on creating Topo-Smart templates, see "Modifying Topo-Smart Templates" on page 155.
- Check the `ovtopodiverter.log` for errors indicating issues with parsing the XML tagged topology fields.
- For the perceived severity (`PSEV`), event type (`ETY`), and probable cause (`PC`) tags, the values must be chosen from a specific set of

enumerated values. See “Understanding X.733 Message Format” on page 141 for the list of valid values. See also Appendix A, “List of Probable Causes,” on page 369 for a list of acceptable probable causes.

$SELECT() Processing Checklist

If messages are being delivered, but a $SELECT does not appear to be being processed correctly, check the following:

- The target column, table name, and index columns match the values in the `agent.xml` configuration file.
- The agent configuration is deployed and applied successfully.
- The agent node MSI is enabled.
- The agent MSI is enabled in the appropriate template.
- 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 also attempt to match the quotes exactly. For example, if the message contains `“something”`, including the quotes, then for the table lookup to match then`“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

These are some things to check with your data collector configuration:

- 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 elements 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 system:

- 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.

Starting and Stopping OVSACN

The script `ovsa_admin` assists in the stopping and starting of all of the processes associated with HP OpenView Service Assurance for Communication Networks. The script is located in `/opt/OV/Telco/bin` on the OVO server and in `/opt/OEMF/V5.0/TopoSrv/bin` on the topology server. The output of `ovsa_admin` is written to a log file in `/tmp` called `ovsa_admin.log`.

To stop all associated processes of OV Telecom Extensions for OV Operations and OV Topology Server, run the following command:

```
ovsa_admin stop
```

To start all associated processes of OV Telecom Extensions for OV Operations and OV Topology Server, run the following command:

```
ovsa_admin start
```

To check the status all associated processes of OV Telecom Extensions for OV Operations and OV Topology Server, run the following command:

```
ovsa_admin status
```

The `ovsa_admin` commands stop, start, and check the status of the following components:

- OVO
- OVO agent
- Telecom agent
- Telecom adaptor
- Corba
- Topology server, including the FM server and the GUI server

Troubleshooting OV Topology Server

Incorrect configuration of the system may result in messages not reaching the problems presenter. OV Topology Server maintains independent logs that receive messages at each stage of event processing. Thus, any problem due to incorrect or improper configuration can be tracked quickly.

Is there a problem in logging messages into alarm database?

If OV Topology Server is unable to log the messages in the log files, it writes an error message in the syslog file and then logs a more detailed message in the `ovfmpeld.log` file. In this file, it writes the problem encountered and lists the messages that created this problem. Check the messages listed in the file `ovfmpeld.log` and re-configure the message configuration details.

Are the messages being displayed in the problems presenter but not reflecting the status update on the map?

For this problem check the following conditions:

a. Check the alarm severity. If the alarm severity is `Indeterminate` no status change is reflected on the map presenter. The network status manager ignores alarms with this severity.

b. Check the managed object instance of the alarm. It must match that of the object on the map.

c. Check the length of the specific problem field. If it is a string, it must not exceed 64 characters.

Checking ECS Operations

This section describes problems that could occur while using ECS and the corrective actions to be taken.

Is the ECS Engine running?

1. Log on to the server for which you wish to check that the ECS Engine is running. Lookup the contents of the Correlation Gateway configuration file (for the FM server, it is `$FMSETC/share/newconf/ovfmpcgw.conf`

   Check the setting of the `ECS_INSTANCE`.

2. Change user to `root` and use the following command to check whether the engine instance for ECS is running:

   ecsmgr -i *ECS_INSTANCE* -info

   Where *ECS_INSTANCE* is the instance number of the ECS Engine.

   If the ECS Engine is not running the following message is displayed:

```
connection to engine failed
Failed to obtain engine status
```

   If the ECS Engine is running, then the engine details, the circuit details, and the datastore and factstore are displayed. The following is a sample of the information displayed for ECS Engine instance 1 running the sample circuit SCEN1:

```
Engine Status :
engine environment - SA
engine instance - 1
engine version - ECS 3.1 (A.03.10)
engine state - running
engine policy - output (unless discarded by a circuit)
time last started - Thu Oct  2 14:05:10 1997
engine trace mask - 0x0
engine log mask - 0x7
maximum engine log size - 512 Kbytes
maximum event log size - 512 Kbytes
input event logging - off
output event logging - off circuit name - SCEN1
circuit date - Mon Sep 15 17:52:52 1997
circuit version - 0
circuit unique identifier - 6862879
```

```
time circuit load - Thu Oct  2 14:05:27 1997
circuit state - enabledfact store name - SCEN1_FACT
fact store date - Thu Aug 17 13:27:30 1995
fact store version - 0
time fact store load - Thu Oct  2 14:05:27 1997
data store name - SCEN1_DATA
data store date - Tue Nov 21 18:40:23 1995
data store version - 0
time data store load - Thu Oct  2 14:05:27 1997
```

Where can I find the ECS Engine error log information?

1. Log on to the server for which you wish to check the ECS error log. Lookup the contents of the Correlation Gateway configuration file (for the FM server, it is `$FMSETC/share/newconf/ovfmpcgw.conf`, and for the MD server it is `$MDETC/share/conf/fmpmdcgw.conf`).

   Check the setting of the `ECS_INSTANCE`.

2. Change directory to `$OV_LOG/ecs/`*`ECS_INSTANCE`*.

   Where *`ECS_INSTANCE`* is the instance number of the ECS Engine.

Why are alarms not being correlated?

If all alarms seem to be appearing in the problems presenter, first check whether the ECS Engine is running as explained above. If it is, then check the transit delay time for the circuit. The transit delay time must be set judiciously. It must cover the maximum possible time that an alarm would take to travel from the source to the destination, In this case the source is the device emitting the alarm and the destination is the ECS Engine.

The Transit Delay Time is set using the "External" tab in the ECS Designer window. Refer to the relevant ECS documentation for details.

Are the alarms being correlated? Are they passing through the ECS Engine?

To check whether alarms are being correlated, first check that the ECS Engine is running as explained above.

To check if the alarms are being received by the ECS Engine, execute the following command:

`ecsmgr -I `*`ECS_INSTANCE`*` -stats`

Where `ECS_INSTANCE` is the instance number of the ECS Engine.

The command displays the statistics of the ECS Engine and the circuits

loaded.

The ECS Engine statistics consist of the number of engine and filter details and the alarms processed. If the engine is able to receive alarms it displays the following statistics:

```
Engine Statistics -

input.inputFilters = [("OVC/Assurance", "PFFX733Alarm", ())]
input.bypassed = 0
output.discards = 0
engineInstance = 1
input.initialDiscards = 0
currentTime = 19971114025629.000000Z
input.numEvents = 8
output.numEvents = 8
```

Check the reading against the following statements:

```
input.bypassed = 0
output.discards = 0
```

If the engine is configured correctly, then the reading against the above statements must be zero (0). If it is not zero, then there is an error in the ECS Engine configuration in the Encoder/Decoder module.

1. Ensure that the Encoder/Decoder shared library is properly loaded.
2. Check the file `$OV_CONF/ecs/ed/ed.conf`. "OVC/Assurance" must appear only once in this file.

Now check the reading against the following statements:

```
input.numEvents = 8
output.numEvents = 8
```

These refer to the number of alarms received (input) and the number of alarms sent out (output) by the ECS Engine. If the number of alarms received by the ECS Engine has a reading greater than zero (0) , then the engine is receiving alarms. If both the statements listed above have the same figure against them, then all the alarms that have been received by the engine have been processed and sent to the Correlation Gateway. If the number of alarms output by the ECS Engine is less than the input, it indicates that the circuit has dropped some alarms. Ensure that the circuit loaded is designed to drop alarms.

20 Commands Quick Reference

Commands Quick Reference

The following table lists all of the OV Telecom Extensions for OV Operations and OV Topology Server commands and files. The commands and files are listed in alphabetical order.

Table 20-1

Command	User	Description
`Agent.dtd`	read-only	Provides structure for defining agent configuration data stored in agent XML files.
`corba.answers`	N/A	Configuration file that contains settings needed to customize CORBA.
`corba.setup`	`root`	Configures the CORBA environment on the topology server.
`corba.unsetup`	`root`	Unconfigures the CORBA environment on the topology server.
`corba.validate`	`root`	Reports status of CORBA servers and database instances.
`fms.answers`	`N/A`	Configuration file that contains settings needed to customize FM server on the topology server.
`fms.setup`	`root`	Configures the FM server environment on the topology server.
`fms.unsetup`	`root`	Unconfigures the FM server environment on the topology server.
`fms.validate`	`root`	Reports status of FM servers and related components.
`fmsalmbackup`	`root` or `oemfadm`	Backs up the problems database.
`fmsalmrestore`	`root` or `oemfadm`	Restores the problems database that was backed up using the `fmsalmbackup` command.

Table 20-1

Command	User	Description
fmsalmrecover	root or oemfadm	Recovers the problems database.
fmscfgsync	root	Replicates and synchronizes new configuration data among all topology servers.
fmsdeinit	root	Deinitializes the FM server and prepares for system configuration.
fmseccfgupd	root or oemfadm	Updates the event correlation rules at the FM Server without restarting the server.
fmsinfo	root or oemfadm	Provides product information and status for all FM servers.
fmsinit	root	Initializes the FM server and topology server databases.
fmslogconadmin	root or oemfadm	Establishes logical connectivity between managed objects that are not physically connected but whose alarms must be correlated.
fmsmhexport	root or oemfadm	Exports information in the OVC/Assurance topology and presentation databases into an ASCII file.
fmsmhimport	root or oemfadm	Copies object information stored in the intermediate file into the topology and presentation databases.
fmsmhdomainimport	root or oemfadm	Updates the databases with the object records whose domain information was not updated in the presentation database. This information was stored in the failed operation files.
fmsmhimporttopo	root or oemfadm	Copies object information stored in the intermediate file into the topology database without updating the presentation database.

Table 20-1

Command	User	Description
fmsmhpdbimport	root or oemfadm	Updates the presentation database with objects whose presentation details were not updated in the presentation database. This information was stored in the failed operation files.
fmsmhtoposync	root or oemfadm	Compares the files generated by the current run of fmsmhexport and its previous run. Provides a list of the updates made to the topology and presentation database.
fmsomeventbackup	root or oemfadm	Backs up the OM Event audit trail.
fmsomeventrestore	root or oemfadm	Restores the OM Event records that were backed up using fmsomeventbackup
fmsomgen	root or oemfadm	Reads OM event records from a specified file and generates the events.
fmsopfix	oemfadm	Corrects inconsistencies after an interruption that causes the Operation Profile Configurator to fail.
fmsopcfg	oemfadm	Opens the Operation Profile Configuration utility.
fmsoutagebackup	root or oemfadm	Backs up the outage plan audit trail
fmsoutagerestore	root or oemfadm	Restores the outage plan information backed up using the fmsoutagebackup command
fmssdrexport	root or oemfadm	Copies system distribution rules data to a file on the topology server.
fmssdrimport	root or oemfadm	Pushes system distribution rules data to target machines.
fmsstart	root	Starts the FM Server.
fmsstate	root or oemfadm	Queries and sets the state of the FM server on the topology server. It is recommended not to run this command.

Table 20-1

Command	User	Description
fmsstatus	root or oemfadm	Displays status of all processes of the FM Server from which it is executed.
fmsstop	root	Shuts down the FM Server.
fmssysconfig	root or oemfadm	Interactively configures the FM server.
fmssysunconfig	root or oemfadm	Unconfigures the FM server.
fmstopodbexport	root or oemfadm	Pushes the topology database data to a file.
fmstopodbimport	root or oemfadm	Copies the topology database to target systems.
fmstopofix	oemfadm	Fixes the inconsistencies logged by the HA Managed Object Manager.
guicstart	root	Starts the GUI Client.
guidb.answers	N/A	Configuration file that contains settings needed to customize GUI database.
guidb.setup	root	Configures the GUI database environment on the topology server.
guidb.unsetup	root	Unconfigures the GUI database environment on the topology server.
guidb.validate	root	Reports status of GUI database instance.
guidbbkup	root or oemfadm	Backs up the presentation information stored in the GUIDB.
guidbdeinit	root or oemfadm	Removes all data from the GUI database.
guidbenv	N/A	File that contains environment variables for GUIDB commands.

Table 20-1

Command	User	Description
guidbinit	root or oemfadm	Loads initialization data into presentation database.
guidbmocsym	root or oemfadm	Imports the necessary information to create user-defined map object classes and define the mapping between managed object classes and map object classes from a data file.
guidbrestore	root or oemfadm	Restores the GUI database to the state it was when the backup was taken.
guidbsysconfig	root or oemfadm	Interactively creates and initializes the presentation database.
guis.answers	N/A	Configuration file that contains settings needed to customize GUI server.
guis.setup	root	Configures the GUI server environment on the topology server.
guis.unsetup	root	Unconfigures the GUI server environment on the topology server.
guis.validate	root	Reports status of GUI server system.
guisadmin	oemfadm	Administrators and customizes the user interface.
guisenv	N/A	Contains the environment variables for the GUI server.
guisinfo	root or oemfadm	Provides post-configuration and operation status of GUI server.
guisstart	root	Starts the GUI Server.
guisstatus	root	Displays the status of the GUI server processes.
guisstop	root	Stops the GUI Server.
guissysconfig	root	Configures the GUI server system.
guissysunconfig	root	Unconfigures the GUI server system.

Table 20-1

Command	User	Description
Mappings.dtd	read-only	Provides structure for defining mapping definitions in a mapping XML file.
logdisplaymsg	any user	Launches a dialog box to display errors from third party products to the topology GUI.
oemferr	any user	Invokes the Error Checking utility that displays the details of the topology server error messages logged in the syslog file.
oemfinfo	root or oemfadm	Provides product information and status for all components on the topology server.
oemflinkmapenvvars	N/A	Contains the environment variables that affect the oemflinkmap function. This file exits on the machine hosting the presentation database.
oemflinkmapimport	oemfadm	Creates the link object client submap.
oemfstart	root	Starts the topology server and the related servers or applications.
oemfstatus	root or oemfadm	Displays the status of all topology server processes.
oemfstop	root	Stops the topology server and the related servers or applications.
operator.setup	root	Configures the OV Topology Server default operator telco_op.
operator.unsetup	root	Removes the OV Topology Server default operator telco_op.
ovcfg_set_location	root	Defines the location name used by the topology configuration tools.
ovcfgagent	root	Configures the telecom OVO agent in XML format.
ovcfgcapture	root	Imports existing OVC/Assurance topology data into OV Topology Server configuration.

Table 20-1

Command	User	Description
ovcfgdeploy	root	Generates target configuration files from XML configuration files.
ovcfgmappings	root	Configures the mapping definitions needed for OVO and OV Topology Server in XML format.
ovcfgnewproject	root	Creates a new set of project ocnfiguration files.
ovcfgproject	root	Manipulate project definitions in XML format.
ovcfgsa	root	Starts the Telecom Configurator GUI.
ovcfgtopodata	root	Configures the upper topology instances in XML format.
ovcfgtopomodel	root	Configures the topology object model in XML format.
ovcfgvalidate	root	Checks the validity of all of the project XML files against their respective DTDs.
ovoconf.apply	root	Pushes configuration files from the topology server to the proper location on the OVO server.
ovprobinfo	root	Queries and prints out topology problem information generated by the telecom adaptor.
ovsa_admin	root	Starts, stops, and checks status of the HP OpenView Service Assurance for Communication Networks components.
ovsadc	root	Collects alarms from network elements and forward them to OVO opcmsg interceptor.
ovtelco.setup	root	Sets up the OV Telecom Extensions for OV Operations environment.
ovtopoadaptor	root	Synchronizes the state between the topology server and OVO message server. Typically, this command should not be by an administrator except to turn tracing on.

Table 20-1

Command	User	Description
`ovtopoadapter.log`	N/A	Logs errors and current state of the telecom adaptor.
`ovtopoconf.apply`	`root`	Pushes topology configuration files to the proper location on the topology server.
`ovtopodiverter`	`root`	Routes Topo-Smart messages from the OVO agent to the telecom injector. Typically, this command should not be by an administrator except to turn tracing on.
`ovtopoinjector`	`root`	Routes the Topo-Smart messages to OV Topology Server. Typically, this command should not be by an administrator except to turn tracing on.
`ovtoposrv.setup`	`root`	Configures the OV Topology Server, including FMS, GUIS, GUIDB, CORBA, and default operator.
`Project.dtd`	read-only	Provides structure for defining project definitions in a project XML file.
`TopoData.dtd`	read-only	Provides structure for defining topology instances in XML format.
`TopoModel.dtd`	read-only	Provides structure for defining topology object model in XML format.

A List of Probable Causes

This chapter provides the list of probable causes as listed in the X.733 document. The X.721 list is used for global variables and M.3100 variables are used for local values.

Probable Causes

The list of probable causes are classified into five categories:

- Communication (for the list under communicationAlarm)
- Quality of service (for the list under qualityofServiceAlarm)
- Processing error (for the list under processingErrorAlarm)
- Equipment (for the list under equipmentAlarm)
- Environmental (for the list under environmentalAlarm)

The X.721 probable causes are listed in alphabetic order each event type in Table A:

Table A-1 List of Probable Causes

Event Type	Probable Cause
Communication	• callEstablishmentError • communicationsProtocolError • communicationsSubsystemFailure • degradedSignal • dTE-DCEInterfaceError • framingError • lANError • localNodeTransmissionError • lossOfFrame • lossOfSignal • remoteNodeTransmissionError

Table A-1 List of Probable Causes

Event Type	Probable Cause
Quality of service	• bandwidthReduced • congestion • performanceDegraded • queueSizeExceeded • resourceAtOrNearingCapacity • responseTimeExcessive • retransmissionRateExcessive • thresholdCrossed
Processing error	• applicationSubsystemFailure • configurationOrCustomizationError • corruptData • cpuCyclesLimitExceeded • fileError • outOfMemory • softwareError • softwareProgramAbnormallyTerminated • softwareProgramError • storageCapacityProblem • underlyingResourceUnavailable • versionMismatch

Table A-1 List of Probable Causes

Event Type	Probable Cause
Equipment	• adapterError • dataSetOrModemError • equipmentMalfunction • inputDeviceError • inputOutputDeviceError • multiplexerProblem • outputDeviceError • powerProblem • processorProblem • receiveFailure • receiverFailure • timingProblem
Environmental	• enclosureDoorOpen • excessiveVibration • fireDetected • floodDetected • heatingOrVentilationOrCoolingSystemProblem • humidityUnacceptable • leakDetected • materialSupplyExhausted • pumpFailure • pressureUnacceptable • temperatureUnacceptable • toxicLeakDetected

Probable Causes

The M.3100 probable causes are listed in alphabetic order under Alarm Types in the following table:

Alarm Type	Probable Cause
Communication	• aIS • callSetUpFailure • degradedSignal • farEndReceiverFailure • framingError • lossOfFrame • lossOfPointer • lossOfSignal • payloadTypeMismatch • transmissionError • remoteAlarmInterface • excessiveBER
Equipment	• backplaneFailure • dataSetProblem • equipmentIdentifierDuplication • externalIFDeviceProblem • lineCardProblem • multiplexerProblem • nEIdentifierDuplication • powerProblem • processorProblem • protectionPathFailure • receiverFailure • replaceableUnitMissing

Alarm Type	Probable Cause
Environmental	• airCompressorFailure • airConditioningFailure • airDryerFailure • batteryDischarging • batteryFailure • commercialPawerFailure • coolingFanFailure • engineFailure • fireDetectorFailure • fuseFailure • generatorFailure • lowBatteryThreshold
Processing error	• storageCapacityProblem • memoryMismatch • corruptData • outOfCPUCycles • sfwrEnvironmentProblem • sfwrDownloadFailure

Probable Causes

B Applications and Tasks

This appendix lists the applications and their associated task names that are provided with OVC/Assurance.

List of Applications and Tasks

The following are the default applications and their related tasks that are provided with OVC/Assurance. These application and task names will be displayed in the Operation Profile Configurator when the Application Maintenance function is invoked.

There are four applications:

- managed_object_application

  This is the application name associated with the actions taken with respect to managed objects via the managed object manager. This application is referred to as the Map Presenter in the GUI Client.

- problem_application

  This is the application name associated with actions taken with respect to problems for managed objects via the problem manager. This application is referred to as the Problems Presenter in the GUI Client.

- outage_plan_application

  This is the application name associated with actions taken with respect to outage plans for managed objects via the outage plan manager. This application is referred to as the Outage Plan Presenter in the GUI Client.

- om_event_application

  This is the application name associated with actions taken with respect to OM events for managed objects via the OM Event manager. This application is referred to as the OM Event Presenter in the GUI Client.

The following four tables list the tasks associated with each of the applications mentioned above.

Table B-1 Tasks Associated with managed_object_application

Task Name	Description
get	This task must to assigned to all users being provided access to this application. It enables the user to view the Map Presenter. Enables the user to receive notifications from the FM Server. It also enables the user to query managed object information from the FM Server.
create	Enables the user to create managed objects.
delete	Enables the user to remove or delete managed object.
add_to_domains	Having created a managed object, this task enables the user to dynamically add this newly created object instance to a domain. Though this task can be selected and assigned to any operator, it will be made available only to the administrator in the Map Presenter.
remove_from_domains	When a managed object is removed from the FMS, it needs to removed from the domain, too. This task enables the user to remove the object instance from the domain.
clear_propagated_alarm_status[a]	Enables the user to clear the alarm status tagged to one or more managed object instances. This task can be performed using the menu option or a toolbar icon.
set_administrative_state[a]	Enables the user to place a managed object in the administrative state from the Map Presenter.
load_on_demand	This task must to assigned to all users being provided access to this application. It enables the application to pre-load appropriate information on the Map Presenter. Instructs the problem handler to pre-load the problems into the client controller when the application is started, based on the managed object selected by user from the Map Presenter.

a. The function related to this task is not available to the operator in the current release.

Table B-2 Tasks Associated with problem_application

Task Name	Description
get	This task must to assigned to all users being provided access to this application. 1. Enables the user to receive notifications from the FM Server. 2. Enables the user to query problem details from the FM Server.
set_trouble_ticket	Enables the user to submit one or more trouble -tickets to a problem. Enables the user to launch the trouble ticket application. The tt-id received through the above actions will be entered in the FM Server.
annotate	Enables the user to submit annotation notes to a problem.
own	Enables the user to own a problem.
disown	Enables the user to disown a problem that the user had earlier owned. To effectively use this task, the user must also be assigned the task `own`.
disown_any	Enables the user to disown problem conditions owned by other users.
discharge	Enables the user to discharge problem conditions that the user had earlier owned. To effectively use this task, the user must also be assigned the task `own`.
get_history	Enables the user to query all alarms associated with a selected problem. This is the Problem History function.
get_ne_history[a]	Enables the user to query both active and historical alarms associated with any selected network element.
get_related_alarms	Enables the user to query alarms related to a problem.

Table B-2 Tasks Associated with problem_application

Task Name	Description
load_on_demand	This task must to assigned to all users being provided access to this application. It enables the application to pre-load appropriate information on the Problem Presenter. Instructs the problem handler to pre-load the problems into the client controller when the application starts.

a. The function related to this task is not available to the operator in the current release.

Table B-3 Tasks Associated with outage_plan_application

Task Name	Description
get	This task must to assigned to all users being provided access to this application. 1. Enables the user to receive notifications from the FM Server. 2. Enables the user to query outage plan details on the FM Server.
manage	Enables the user to submit outage plans for one or more managed object instances.
restore	Enables the user to “manually” restore a managed object instance that is currently placed in outage. it is assumed that the same user had submitted the outage plan. Hence to use this task the user must also be assigned the task `manage`.
update	Enables the user to modify an existing outage plan that the user had created. Hence to effectively use this task the user must also be assigned the task `manage`.
restore_any	Enables the user to “manually” restore managed object that has been set on outage by another user.
update_any	Enables the user to modify outage plans submitted by other users.
load_on_demand	This task must to assigned to all users being provided access to this application. It enables the application to pre-load appropriate information on the Outage Plan Presenter. Instructs the application handler to pre-load the outage plans into the client controller when the application starts.

Table B-4 Tasks Associated with om_event_application

Task Name	Description
get	This task must to assigned to any user being provided access to this application. 1. Enables the user to receive notifications from the FM Server. 2. Enables the user to query OM events on the FM Server.
own	Enables the user to own an OM event.
disown	Enables the user to disown an OM event that the user had earlier owned. To effectively use this task, the user must also be assigned the task own.
disown_any	Enables the user to disown the OM event owned by any other user.
discharge	Enables the user to discharge an OM event that the user had owned earlier. To effectively use this task, the user must also be assigned the task own.
load_on_demand	This task must to assigned to all users being provided access to this application. It enables the application to pre-load appropriate information on the OM Event Presenter. Instructs the problem handler to pre-load the records into the client controller when the application starts.

Glossary

acknowledge Active messages in the iNOC console's Active Messages browser and Problem Presenter can be acknowledged by an administrator or operator or automatically after an action has been completed successfully.

action A response to a message triggered by an event.

admin group A system-defined user group. Users belonging to this group have supervisory rights over other users. Administrators can discharge and disown problem conditions owned by other users.

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

agent A management component deployed on a system for the purpose of collecting events and injecting them as alarms into the management system. The agent performs basic event data collection and initial processing into a normalized alarm format.

alarm A message about an event that is 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 describe actions and tasks that have been applied to solve a given problem.

annotation server A user-supplied server that receives a request from an annotation node within a correlation circuit, performs some action, and returns a response to the annotate node. The action performed by the annotation server may involve information extracted from events in the circuit. The information returned is typically obtained external to the annotation server.

application handlers Applications that connect to the graphical user interface and are responsible for the accuracy and completeness of the data in the Client Controller.

application server Consists of a set of CORBA client applications, referred to as the application handlers, which interface between the topology server and the GUI Client. The application handlers supported are problem handler, map handler, outage plan handler, and OM event handler.

ARS Remedy Corporation's Action Request System, a network-based trouble ticketing and tracking system.

attributes Properties associated with a managed object class (See managed object class) and are registered under a registration identifier (See Registration ID).

attribute value assertion (AVA) The association of an attribute with a value, written in the form `attribute registration id = value`.

button panel A row of buttons either at the bottom or the right side of a window. Each button has a specific function. The function of a button is activated only when the button is highlighted. Menu greying indicates that the function is either not available or not applicable.

circuit See Correlation circuit.

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

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

Client Controller The GUI Server process containing the object model and data and the views that manage the GUI Client presenters.

column based parser The parser type used for message classes that follow a single fixed column format.

component Physical objects contained within a network element. Components may or may not emit alarms.

component class A logical class type in an object model that can be contained under a network element class, termination point class, and other component classes. Objects in this class may or may not emit alarms.

composite event In ECS, a composite event consists of a structured aggregation of addressed component events, each of which may be a primitive event, a temporary event, or a composite event. A composite event may only exist within a correlation circuit.

connection See Link.

connection class A logical class type of an object model that can be contained under a network class. Objects in this class are used to link two managed objects via their termination point objects.

connection symbol A symbol on a network map that connects two map symbols.

containment hierarchy The rooted tree that is constructed by applying the relationship "is contained within" to the actual object instances (See Managed object instance). Lower level object instances are contained within object instances one level higher in the containment tree to which they are attached. The containment hierarchy follows the containment rules specified by the containment tree.

containment tree The rooted tree that is constructed by applying the relationship “can be contained into” object classes (See Managed object class). Lower level object classes can be contained within an object class one level higher in the containment tree to which they are attached. The containment tree specifies the containment rules by listing the classes of object instances that a particular object instance (of a particular object class) may contain.

CORBA (Common Object Request Broker Architecture)

A specification for objects to locate and activate one another in a distributed computing infrastructure.

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

event correlation is as specified by one or more correlation circuits loaded into the correlation engine.

correlation node A set of customizable processing elements that facilitates the correlation of event storms in real time.

data collector A data collector receives messages emitted from network elements and forwards them to the agent.

data store In ECS, a component of the ECS engine that holds user-specified values for named data items. A correlation circuit may be associated with one of many data stores loaded into the correlation engine.

details An operation on the Problem Presenter that displays additional information about a network element emitting an alarm. The Problem Details panel displays the problem condition, owner, and FDN for a managed object.

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

discharge An operation that removes a problem or event from the table area of table presenters. When you discharge a problem or event, it is removed for all users. Only operators who own a problem or event can discharge the problem or event.

disown An operation that releases a problem or event from its owner. When an event or problem is disowned, it appears in the table area of a Problem or OM Event Presenter for all users as unacknowledged. It is recommended that operators disown problems and events when they are not monitoring the network.

ECS See Event Correlation Services (ECS).

ECS circuit See Correlation circuit.

ECS Designer An ECS component that is used to create and test correlation circuits. It works in two modes: build and simulate. Must be purchased separately.

ECS Engine See Correlation engine.

Element Management System (EMS) Vendor- or device-specific components that provide device-specific interfaces for receiving events or monitoring the end network elements.

event correlation A process of filtering superfluous messages based on user-configured criteria.

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

explodable Map symbols that result in a submap upon double-clicking.

fact store A component of the ECS Engine that stores relationships among objects. Any two objects may be related using any user-defined relationship. The facts may be accessed at runtime by the ECDL expressions configured into the correlation node parameters.

Fault Management Server (FM Server) A topology server component that contains a representation of the underlying network. The FM Servers receive, store, and manage messages from the agents to which they are connected.

FDN (Fully distinguished name) The FDN uniquely identifies an object instance. The FDN is formed by concatenating all of the relative distinguished names (RDNs) for each object instance in the containment path from the root of the containment hierarchy to the base of the object instance that is being identified. The FDN is written as /RDN/RDN/.../RDN where each RDN is the RDN of the object instances along the containment path.

filter A condition that changes, suppresses, or redirects information to the topology GUI Clients.

FM Server See Fault Management Server.

GUI Client A set of GUIs that enable users to view topology-specific information. It connects to the topology server.

GUIDB See GUI database.

GUI database Used to store persistent graphical information, including user preferences and graphic layouts. Also known as the presentation database.

GUI Server A topology server component that is responsible for managing the display processes. It runs the application handlers that form the bridge between the FM Server process and the GUI Client.

high availability An optional package that enables the topology server to continue operations in spite of a single point of hardware or software failure.

history An operation in a Problem Presenter that displays all alarms associated with an active problem. When an operator owns or discharges a problem, all alarms associated with that problem become historical data and are not available with this operation.

host A server or workstation.

hostname The name of the server in the network.

information icon An icon that replaces secondary state icons when more than two secondary state icons are present for a single node.

installation A term used to describe an entire managed network, installed and configured with the topology server. Also referred to as a site.

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 Stands for Internet Protocol. This is a datagram-oriented network layer protocol used by TCP and UDP protocols. Its main function is to route datagrams among nodes in different networks.

IPC Stands for InterProcess Communication. In this document, IPC specifically refers to the HP-UX IPC facility.

link Refers to the object that is represented by the connection symbol.

locate An operation in the table presenters that locates a problem, outage, or event in a managed network.

location A cluster of one of more machines offering telecom topology services. Each location contains only one topology server.

log file Files that store received messages emitted from network elements in a managed network, including a raw log, a report log, an unknown log, and a message class log.

managed object A logical or physical resource that can be managed. Every managed object is a member of a specific object class, whose members share the same set of attributes, operations that can be performed on them, notifications they can emit, and behaviors. With the topology server, managed objects are represented graphically by map symbols on Map Presenters.

managed object class (MOC) A group of managed object instances with the same or similar properties. An object class is registered under a unique registration ID, and is defined by a list of attributes, a list of naming attributes, a set of operations that it supports, and notifications or events that it can emit.

managed object instance (MOI) A representation of a specific occurrence of an object class to be managed. The MOI is addressed by a FDN.

management domain A logical grouping of network elements that is not dependent upon the physical boundaries of the network.

management server The central system from which all messages emitted from managed nodes are forwarded. The OV Operations software and relational database reside on the management server.

map presenter A type of GUI Client presenter that displays network topology information and status. It displays details of elements being monitored in a network. It also receives updates regarding the state and status of managed objects and network topology, and displays this information on a map.

map symbol A symbol on a network map that represents a managed object instance. An icon is assigned to each managed object

class, so managed object instances can be displayed in a Map Presenter. This assignment is performed by an administrator using the GUI Server configurator, the Admin Panel.

MC/ServiceGuard Allows the creation of a high availability cluster of HP 9000 Series 800 computers. A high availability computer system allows applications to continue in spite of a hardware or software failure.

message A structured, readable piece of information about a status, event, or problem related to a managed node.

message classes A network element can have multiple message classes. Basic details, such as message logging, must be specified for each message class. Configuration information includes message headers and trailers, message format, and message mapping to CMISE format.

naming attribute The attribute of an object class which distinguishes the object instances belonging to that class.

network class A logical class type of an object model that is the highest containment class of managed objects. Network classes, network element classes, and connection classes can be contained under a network class.

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

network element class A logical class type of an object model that consists of network elements. This is the highest class of objects that can emit alarms.

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

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

node A connection point for data transmissions.

object A representation of a logical or physical entity or resource, or a group of such physical entities that exist in the network. Examples of objects are a network, a computer, an interface, and a process. Objects are represented graphically by the symbols that appear on submaps.

object instance See Managed object instance.

object model Enables the classification of devices in a monitored network into object classes, including network class, network element class, connection class, termination point class, and component class.

OM Event Presenter A type of GUI Client presenter that manages and displays non-alarm events generated by management applications. It contains the problem management functions: own, disown, discharge, and locate.

operation profile See profile.

Operation Profile Configurator A topology server GUI with which administrators configure operator IDs, roles, operation profiles, filters, and application domains.

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

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

outage plan Outage schedules to track when network elements are to be taken out of service from a monitored network.

outage plan presenter A type of GUI Client presenter that displays outage plans and schedules in tabular form. It contains the problem management functions: submit, modify, locate, and restore.

outstanding alarms Alarms, both new and acknowledged that have not yet been discharged.

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

OV DM TMN HP OpenView TMN Distributed Management Platform.

ovstart The program that starts up the OV DM TMN processes. This program is (normally) run automatically on system startup and can only be run by the superuser (i.e. UNIX System Administrator).

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

OVw HP OpenView Windows, an advanced graphical user interface designed to integrate network management and system management applications.

own An operation that assigns a problem or event to an operator. Owned problems and events appear in the table area of Problem and OM Event Presenters to all users as owned. After a problem or event is owned, a trouble ticket can be created to track the problem, or an operator can discharge it.

parser type The parser that is used for identifying the messages.

partition A set of managed objects grouped together based on physical characteristics or network technology. Each partition is associated with one location.

PDU (Protocol Data Unit) Used for the specification of association requests and responses.

presentation database See GUI database.

primary FM Server The FM Server from which system distribution rules are set. It is the server used to configure details for an installation or site.

problem The result of correlating multiple alarms with the same target object, probable cause, and specific problem and presenting them as a single instance to the user. Problems are higher level abstractions of groups of underlying alarms.

problem presenter A type of GUI Client presenter that displays problems in a tabular form. It contains the problem management functions: own, disown, discharge, locate, history, and details.

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

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.

RAV (Raw alarm viewer) A type of alarm viewer that displays real-time or query-based raw alarm messages for any network element in a managed network.

RDN (Relative Distinguished Name) List of Attribute Value Assertions (AVAs) of the naming attributes of the object class to which the object belongs. The RDN is written AVA, AVA, . . . , AVA.

registration ID A sequence of numbers used to uniquely identify such things as attributes and object classes.

regular expression based parser A parser type for message classes that can transmit messages in more than one format. These messages may or may not contain all the fields defined for the network element object class.

restore An operation in an Outage Plan Presenter that renews the status of network elements after an outage expires.

server A computer or network that provides service to other computers on the network (clients).

SNMP (Simple Network Management Protocol) The ARPA network management protocol used primarily for managing TCP/IP networks.

status propagation rules A list of statements that indicate to the system the parameters by which the status of the child class object in a network map hierarchy changes that of the parent class object. These rules are defined in the `sprules.conf` file in the `$FMSETC/share/newconf` directory.

submap A term used in the Map Presenter to refer to a view of a network map. For example, one submap may show all the nodes on a particular network, while another submap may show all the software subsystems of a particular node. The application or user that creates a submap determines the content of the submap.

submit An operation in an Outage Plan Presenter that enables users to create or modify an outage plan for a network element.

symbol A graphical representation of an object in a Map Presenter. An object can be represented by multiple symbols. A symbol has the characteristics symbol type, status, and label.

table area The rows and columns of table presenters where problems, outages, and events are displayed. It can be customized by users.

table presenters A classification for a type of GUI Client presenters that display problems, outages, and events in tabular form. Displayed attributes and operations differ for the different presenters. Table presenters include the Problem Presenter, Outage Plan Presenter, and the OM Event Presenter.

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.

termination point An object that connects to a link object.

termination point class A logical class type of an object model that can be contained under network element classes and other termination point classes. Objects in this class are used to form connections between managed objects via termination point objects.

topology server An OpenView software product that enables customers to manage telecom-specific networks. It integrates with OV Operations to provide a complete solution network management system.

TTS Trouble ticketing system.

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.

user handler A process responsible for handling topology server GUI logon requests and starting appropriate GUI and server processes and clients.

WAV (Web-based alarm viewer) A type of alarm viewer that displays problem information about a managed network in read-only format in a web browser. The WAV is useful for operators at remote sites.

X.733 A standard alarm format for OpenView Service Assurance for Convergent Services. X.733 specifies a well-defined set of alarm fields and values.

Index

P