HP OpenView
Event Correlation Services
Administrator's Guide

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

Manufacturing Part Number: J1095-90001

September 2003

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 and replacement parts can be obtained from your local Sales and Service Office.

HEWLETT-PACKARD COMPANY

3404 E. Harmony Road

Fort Collins, CO 80528 U.S.A.

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

Trademark Notices

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

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

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

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

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

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

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

Oracle7™ is a trademark of Oracle Corporation, Redwood City, California.

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

UNIX® is a registered trademark of The Open Group.

Perl is a trademark of O'Reilly & Associates, Inc.

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

Contents

Contents

Contact Information

Contacts

Please visit our HP OpenView web site at:

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

There you will find contact information as well as details about the products and services HP OpenView has to offer.

Support

The “hp OpenView support” area of the HP OpenView web site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training Information
- Support program information

1 Introduction

Purpose

The HP OpenView Event Correlation Services Administrator's Guide explains how to manage the ECS Engine. The guide assumes that the ECS Engine is already installed and concentrates on explaining how to do these tasks:

- initializing the ECS Engine
- loading, reloading, and unloading correlation circuits
- loading, updating, saving, and unloading Data and Fact Stores
- starting and stopping the ECS Engine
- saving statistics, error logs, traces, and event logs to monitor the operation of a running ECS Engine
- troubleshooting the ECS Engine

NOTE Ignore all references to the ECS Designer. The ECS Designer is sold as a separate product and must be ordered separately.

Audience

The reader should be experienced in the administration of UNIX-based systems on Network Management.

2 Getting Started

This chapter introduces the HP OV Event Correlation Services (ECS) products (Designer[1] and Engine), explains the principal concepts, and provides an overview of managing the ECS Engine.

The chapter begins with a brief description of how circuits are designed with the ECS Designer and run in an ECS Engine:

- "Designing Correlation Circuits" on page 15.

The next section previews how the ECS Engine fits into the event flow:

- "Event Flow and Circuits" on page 17.

This is followed by a group of chapters describing configuration and setup tasks and issues:

- "Configuring the ECS Engine" on page 23.
- "Setting Up the Environment" on page 28.
- "Integrating User MIBs (DM only)" on page 32.
- "Ensuring Synchronized Timing" on page 41.
- "ECS Engine Files" on page 35.

The last section describes the internal states that the engine and the circuits running on the engine can be in. Later chapters rely on an understanding of the states described in this section:

- "ECS Engine and Correlation Circuit States" on page 37.

NOTE

You may be required to install and configure an **annotation server**. This server is an application that receives requests generated by the ECS Engine, carries out some task, and returns a response. Since an annotation server is implemented by users or third-party developers, obtain installation and configuration procedures from the supplier. For further details, see the *HP OV Event Correlation Services Developer's Guide and Reference*.

1. Ignore all references to the ECS Designer for the rest of this document. The ECS Designer is sold as a separate product and must be ordered separately.

Designing Correlation Circuits

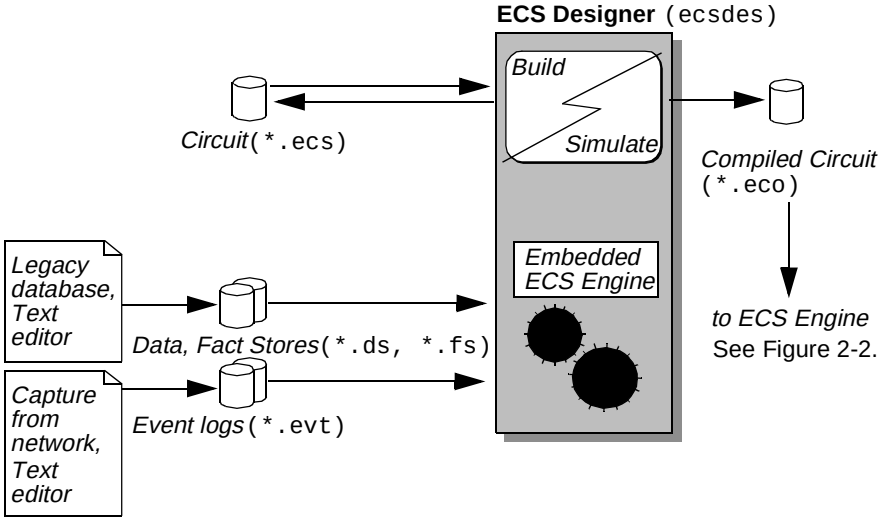
The circuit designer uses the ECS Designer to:

- create a correlation circuit
- test and view any Data and Fact Store files
- compile the circuit
- test the circuit with events generated at the time the problem occurred.

Figure 2-1 summarizes this process. The ECS Designer uses an embedded ECS Engine to simulate the event flow through a circuit. This embedded engine is controlled completely from the Designer (and is not described in this guide).

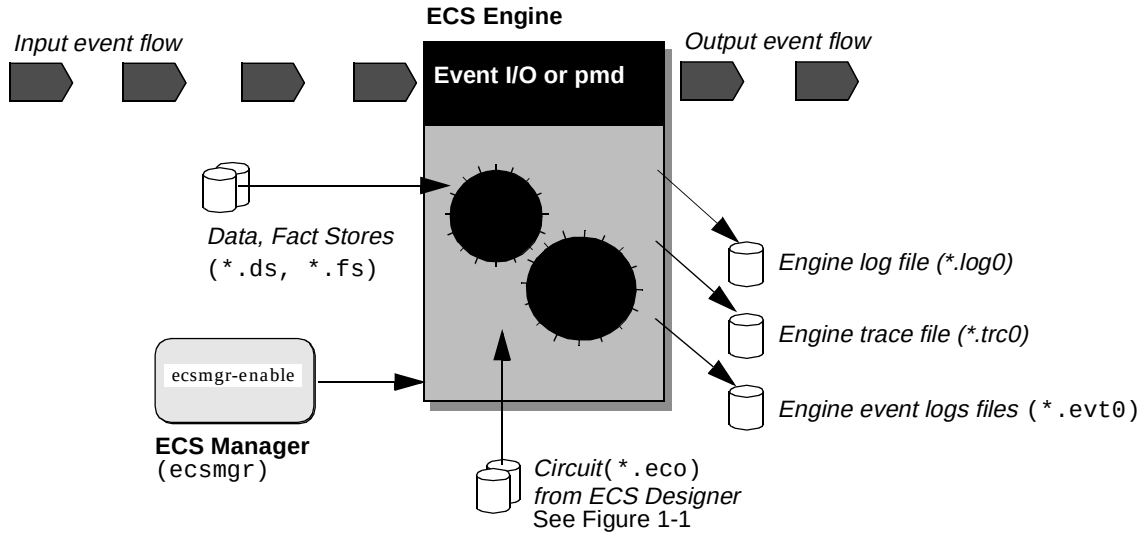
When a correlation circuit is fully tested and debugged, the circuit designer compiles a circuit run-time file and supplies it to you, the administrator. The circuit designer may also supply you with Data and Fact Store files. See Figure 2-2.

Figure 2-1 ECS Designer Main Components



The ECS Engine correlates events according to the rules embedded in one or more compiled circuit files. It may also rely on information from one or more data and fact store files.

Figure 2-2 ECS Engine Main Components



The ECS Engine produces a modified output event flow, together with a number of optional output files including engine log and trace files and event logs.

NOTE

In DM or NNM, CMIP and SNMP events enter and leave the ECS engine through the pmd.

ASCII events can enter and leave the engine through the event I/O in all engines.

Engine log and trace output is directed to the postmaster log and trace files.

Event Flow and Circuits

The flow of events through an ECS engine is determined by the circuits that are loaded and enabled in the configured event streams, and the policy of the circuits and streams.

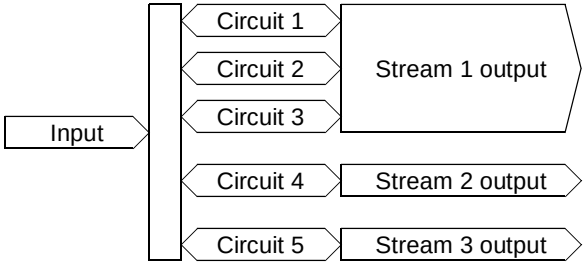
Streams

A stream is a distinct flow of events. Initially there is only one stream called **default** and by default, all circuits that are loaded and enabled are enabled on this stream. Multiple independent event streams are supported by the ECS engine and every event that enters the engine simultaneously flows into every stream.

The concept of a stream only has meaning when talking about the output of a correlation circuit. The circuit itself determines which events it will accept and therefore possibly output to a particular stream. For example, if a circuit is enabled on a stream called 'stream1' then the events output by the circuit are output to 'stream1'.

Each stream is correlated independently, which means that the fate of an event in one stream (whether output or discarded) will not influence the fate of the same event in any other event stream.

Figure 2-3 Event Flow into Circuits and Streams



Stream Policy

Each stream has a configurable policy that determines:

- how circuits affect events in a stream,
- what happens to events that are not accepted by any circuit enabled on the stream,
- what happens to events that are in a circuit when it is disabled.

The stream policy determines how those events are processed:

- A stream with an **output policy** outputs an event unless at least one circuit enabled on the stream discards the event.
- A stream with a **discard policy** outputs an event if any circuit enabled on the stream outputs the event.

These rules are summarized below.

Table 2-1 Summary of Stream Policy Rules

Stream Policy	Default Action When No Circuit is Loaded	Allowed Circuit Policies	An Event is Discarded When...	An Even Output W
Output	Events are output	• Output • Unspecified	At least one circuit in the stream discards the event.	The event is discarded by circuit in the
Discard	Events are discarded	• Discard • Unspecified	The event is not output by any circuit in the stream.	At least one in the stream outputs the e

For example, a circuit designed to create a special warning event when a security violation is detected would probably be given a Discard circuit policy, meaning that it could only be enabled on a stream with a Discard policy. When this stream is initially created it is opaque—that is, it will not output any events until at least one circuit is enabled. When the security violation circuit is enabled, the only events output from the stream are those explicitly output by the circuit, in this case when the circuit detects a security violation. Only then is the special warning event created and emitted.

Compare this with a circuit designed to suppress events during scheduled maintenance. In this case, the circuit would have a policy of Output and could only be enabled on an Output stream. When this stream is enabled it is transparent—that is, it passes all events from its input to its output. When the scheduled maintenance circuit is enabled, the only events suppressed are those that the circuit recognizes as being generated by equipment under scheduled maintenance; all other events are passed through.

In practise, output stream policies are more common than discard stream policies. In NNM, for example, the default stream has an output policy, and this policy should not be changed.

NOTE While the policy of the default stream can be changed through `ecsmgr`, it should not be changed.

Event Duplication

An event is output once, at most, on a given stream; although the same event can be output on more than one stream. This ensures that a stream created for a special purpose, such as for testing, does not interfere with the production network management system.

NOTE This concept of independence does not extend quite so simply to correlation circuits whose output forms these streams. Aspects of this issue are discussed in “Option 2” on page 21.

Correlation Circuits

Specific correlation circuits are loaded into the particular streams after the engine is started. Once loaded, circuits can be enabled or disabled as a whole, or the inputs and outputs of the circuits can be enabled or disabled separately.

A circuit with only input enabled will receive and correlate events, but will not affect the flow of events from the engine. This is useful if a circuit needs time to build up state before it can usefully affect the event flow.

A circuit with only output enabled will no longer receive events from the engine, but events within the circuit will still flow through and out of the circuit and the engine. This is useful when a circuit needs time to finish correlating its events before it is unloaded.

If a circuit (or just its output) becomes disabled while events are still within the circuit, the engine will ignore this circuit when deciding whether to output these events. If the stream policy is set to 'output', and no other circuit is correlating these events, the disabling of the circuit will cause these events to be immediately transmitted from the engine.

Each circuit generally has one or more input ports. Each circuit input port may have conditions associated with it. The conditions are defined using the ECS Designer External Tab and serve to restrict the events that flow into the circuit through the input port.

Enabling Correlation Circuits on More Than One Stream

A correlation circuit can be enabled on more than one stream. There are two ways to do this:

- **Option 1**. Load the correlation circuit once and enable it multiple times, once for each stream.

  For example:

```
ecsmgr -data_load myDatastore myCircuit.ds
ecsmgr -circuit_load myCircuit myCircuit.eco myDatastore
ecsmgr -enable myCircuit
ecsmgr -create_stream test output
```

- **Option 2**. Load the correlation circuit, give it a different symbolic name, and enable it on the desired stream(s).

  For example:

```
ecsmgr -data_load myDatastore myCircuit.ds
ecsmgr -circuit_load myCircuit myCircuit.eco myDatastore
ecsmgr -enable myCircuit
ecsmgr -circuit_load sameCircuit myCircuit.eco -myDatastore
ecsmgr -create_stream test output
```

Option 1 Reduces the processing requirements because the same runtime correlation circuit is used in all streams, This is the preferred option where possible.

NOTE Whenever the configuration of a correlation circuit is modified, there is potential for adversely affecting all streams on which the circuit is enabled. Clearly this might cause undesirable problems if you are testing within a production environment.

Option 2 loads the circuit again and allows you use a different data store, or fact store. Because the second instance of the circuit has a different name to the first, you can change its configuration parameters without affecting the first instance and any stream in which the first instance is enable provided such changes are done though `ecsmgr(1M)`.

All streams on which a correlation circuit is enabled must have a policy compatible with the correlation circuit policy.

NNM

If option 2 is implemented using the NNM Configuration GUI, then you must have a new physical copy of the correlation circuit, the data store and the fact store. A further requirement of managing circuits and streams through the NNM Configuration GUI is that the fact store and data store must have the same name as the correlation circuit (but with different extensions).

Event Input and Output

The ECS Engine has the following event I/O mechanisms:

- In OVO, event input and output is handled internally and cannot be controlled.
- In DM and in NNM, event I/O is normally through the postmaster (*pmd* (1M)), but may be through the ECS Event I/O API.

This is summarized in the following table.

Table 2-2 ECS Event IO

	SNMP	CMIP	ASCII	OpcMsg
OVO	no	no	no	OVO

Table 2-2 ECS Event IO (Continued)

	SNMP	CMIP	ASCII	OpcMsg
DM	pmd	pmd	EIO	no
NNM	pmd	no	EIO	no

Where:

pmd	Events can be input and output via the postmaster
EIO	Events can be input and output via the ECS Event I/O API
OVO	Events are input and output via OVO
no	Events cannot be input or output

In addition, there are restrictions on how streams are handled in some environments:

- For ECS in DM, CMIP and SNMP events output from the default stream only are sent to the pmd. CMIP and SNMP events output from other streams are discarded.
- In ECS for NNM, SNMP events from all streams are sent to the pmd. (CMIP events are not relevant in NNM).
- ASCII events are output through the ECS Event I/O API to any application that has registered to receive the stream carrying the ASCII events.
- ASCII events cannot be output through the pmd.

Configuring the ECS Engine

The ECS Engine must be running before you can load correlation circuits and Data and Fact Stores.

For DM and NNM, the ECS Engine is built into the postmaster (pmd) and is started with the postmaster (through the `ovstart` and `ovstop` commands). In the remainder of this manual, these two engines are known generically as a pmd-linked ECS Engine.

You may also need to run other customer-supplied (or third-party) processes such as:

- An application to feed events to and from the ECS Engine through the ECS Event I/O API. This is a mandatory part of the open engine but can also be present with the ECS Engine for HP OpenView DM if it is necessary to correlate ASCII events.
- An annotation server process that supplies externally defined data (from a database, for example) to a circuit in the ECS Engine.

DM and NNM

The ECS Engine is dynamically linked to the HP OpenView postmaster daemon (pmd) as an intermediate stack. (The postmaster is configured to route events through the ECS Engine with the `$OV_LRF/pmd.ecs.lrf` local registration file).

Uncorrelated events arrive at the ECS Engine through the postmaster. The engine returns the correlated events to the postmaster for distribution. The pmd-linked ECS Engine can also accept ASCII events through the Event I/O API.

See also

- *HP OV Event Correlation Services Developer's Guide and Reference* for details about the Event I/O and annotation server APIs.
- “Starting the ECS Engine” on page 45

Endecoder Configuration

The activation of endecoders in the ECS Engine and the ECS Designer is controlled through the configuration file `$OV_CONF/ecs/ed/ed.conf`. The default configuration (set during installation) may not be appropriate. Installation of DM-linked product configures CMIP and SNMP only.

If you need to support other event types you must change the default configuration. For example, if you require SNMP event support for the open engine, or if you want to support ASCII events in a DM-linked engine then you must edit the configuration file.

The configuration file is a very simple text file containing a line for each supported endecoder module. For example, to support just ASCII events the configuration file should contain just one line:

```
MDL
```

Alternatively, to support both ASCII and SNMP endecoders:

```
MDL
SNMP
```

The keywords you can use in the configuration file are listed in Table 2-3:

Table 2-3 Configuration File Keywords

Keyword	Comments
`MDL`	ASCII events
`SNMP`	ber-encoded SNMP v1 MIB-II Traps.
`CMIP`	ber-encoded CMIP event reports *and* SNMP Traps.

NOTE Changes made to the configuration file take effect when the ECS Engine (or ECS Designer) is next started. For the pmd-linked ECS Engine, changes are read when the pmd is started with the `ovstart` command.

Specifying metadata files

The endecoders rely on the presence of metadata files for detailed information about the structure of specific events. Default metadata files are supplied but, particularly in the case of ASCII events, you will need to recompile them to contain the appropriate event definitions (see “Integrating User MIBs (DM only)” on page 32 and “Integrating ASCII Metadata (ASCII only)” on page 34). The default metadata files used by the endecoders are:

ASCII	`$OV_CONF/ecs/md/mdl/mdl.md`
SNMP and CMIP	`$OV_CONF/ecs/md/ber/ecs.md`

If you need to explicitly specify a metadata file you can do so by setting the appropriate environment variable(s), as follows:

ASCII	`ECS_MDL_MD`
SNMP and CMIP[1]	`ECS_BER_MD`

These environment variables are useful when experimental metadata files are used, and you wish to preserve the originals. This commonly arises when multiple ECS Engines or ECS Designers, with different metadata requirements, are in use on the same machine at the same time.

1. CMIP, ASCII and X733 are no longer supported. Ignore all such references in the document.

ECS Engine Management (ecsmgr)

The ECS Engine is managed with a command line program called `ecsmgr`. You use this program to:

- Operate the ECS Engine
- Monitor the ECS Engine
- Troubleshoot the ECS Engine

NOTE You must be superuser to use `ecsmgr`.

The `ecsmgr -h` command summarizes the usage of `ecsmgr`. You can give only one command at a time. The `ecsmgr` command ignores all commands except the first.

See also *ecsmgr*(1m) manpage.

Engine Instance Number

The instance number uniquely identifies an ECS Engine instance running on a machine. Multiple engine instances can be useful to correlate independent event streams, or when you need to connect the output of one circuit to the input of another. Generally, you can run any number of ECS Engines concurrently on a machine.

DM and NNM	Only one ECS Engine for HP OpenView DM or ECS Engine for HP OpenView NNM can be started on any one machine and the instance number is always 1 (one).
OVO	The OVO Server linked correlation engine is always instance 11. The OVO Agent linked correlation engine is always instance 12.

Operating the ECS Engine

The basic operational tasks control the engine as a whole, individual circuits in the engine and the engine's environment (Fact and Data Stores). These tasks include:

- Managing streams
- Loading circuits, fact files and data files into the engine
- Enabling and disabling circuits
- Updating Fact and Data Stores
- Dumping Fact and Data Stores to file
- Reloading a circuit
- Saving and restoring the Engine configuration

These tasks are described in detail in Chapter 3, "Operating the ECS Engine," on page 43.

Monitoring the ECS Engine

You can check the state of the engine as a whole, as well as the state of individual circuits, and produce event logs, error logs and trace files. This information is useful when you need to verify the correct operation of an engine, provide feedback to a circuit designer, developer, or support staff.

For more detailed information, see Chapter 4, "Monitoring the ECS Engine," on page 69.

Troubleshooting the ECS Engine

If the ECS Engine fails to operate as expected you should check the installation, and you may need to restart the engine.

For more information, see Chapter 5, "Troubleshooting the ECS Engine," on page 87.

Setting Up the Environment

Installation includes an optional procedure to set up environment variables for directory paths common to HP OpenView applications. These environment variables ensure that scripts work across different operating systems and make it easier to maintain them over new releases.

The *HP OV Event Correlation Services Installation Guide* provides instructions for running the `ov.envvars` script to set up environment variables. Use *env*(1) to check if environment variables have been set.

The environment variables required by the ECS Engine and referred to in this book are displayed in Table 2-4.

NOTE

The Windows universal pathnames are relative to the directory into which ECS has been installed. The pathnames shown in Table 2-4 use the environment variable %OV_MAIN% which defaults to `C:\OpenView` but this can be over-ridden during installation.

There is no change in the universal pathnames for all Windows platforms that ECS supports namely Windows NT, Windows 2000 and Windows XP.

Table 2-4 Common Universal Pathnames for the ECS Engine

Universal Name	HP-UX 10.X, 11.X	Solaris 2.X	Windo
$APP_DEFS	/usr/lib/X11/app-defaults	/usr/openwin/lib/app-defaults	%OV_MAIN%\app-defaul
$NCS_BIN	/usr/sbin/ncs	/opt/ncs/install/bin	–
$NCS_CONF	/etc/ncs	/var/ncs	–
$NCS_DB	/var/ncs	/var/ncs	–
$NETFMT	/usr/sbin/netfmt	/opt/OV/bin/netfmt	–

Table 2-4 Common Universal Pathnames for the ECS Engine (Continued)

Universal Name	HP-UX 10.X, 11.X	Solaris 2.X	Windo
$NETFMT_LOG_FILE	/var/adm/nettl.LOG00	/var/opt/OV/log/nettl.LOG00	–
$OV_BACKGROUNDS	/etc/opt/OV/share/backgrounds	/etc/opt/OV/share/backgrounds	%OV_MAIN%\background
$OV_BIN	/opt/OV/bin	/opt/OV/bin	%OV_MAIN%\
$OV_BITMAPS	/etc/opt/OV/share/bitmaps	/etc/opt/OV/share/bitmaps	%OV_MAIN%\bitmaps
$OV_CONF	/etc/opt/OV/share/conf	/etc/opt/OV/share/conf	%OV_MAIN%\
$OV_CONTRIB	/opt/OV/contrib	/opt/OV/contrib	%OV_MAIN%\contrib
$OV_DB	/var/opt/OV/share/databases	/var/opt/OV/share/databases	%OV_MAIN%\databases
$OV_DOC	/opt/OV/doc	/opt/OV/doc	%OV_MAIN%\
$OV_FIELDS	/etc/opt/OV/share/fields	/etc/opt/OV/share/fields	%OV_MAIN%\fields
$OV_GDMO_MIBS	/opt/OV/gdmo_mibs	/opt/OV/gdmo_mibs	–
$OV_HEADER	/opt/OV/include	/opt/OV/include	%OV_MAIN%\include
$OV_HELP	/etc/opt/OV/share/help	/etc/opt/OV/share/help	%OV_MAIN%\help
$OV_HPSMI_MIBS	/opt/OV/hpsmi_mibs	/opt/OV/hpsmi_mibs	%OV_MAIN%\hpsmi_mibs
$OV_INSTALL	/opt/OV/install	/opt/OV/install	–
$OV_LIB	/opt/OV/lib	/opt/OV/lib	%OV_MAIN%\

Table 2-4 Common Universal Pathnames for the ECS Engine (Continued)

Universal Name	HP-UX 10.X, 11.X	Solaris 2.X	Windo
$OV_LOG	/var/opt/OV/share/log	/var/opt/OV/share/log	%OV_MAIN%\
$OV_LRF	/etc/opt/OV/share/lrf	/etc/opt/OV/share/lrf	%OV_MAIN%\
$OV_MAN	/opt/OV/man	/opt/OV/man	%OV_MAIN%\help\C
$OV_MAIN_PATH	/opt/OV	/opt/OV	SRCTARGETD
$OV_NEW_CONF	/opt/OV/newconfig	/opt/OV/newconfig	–
$OV_NLS	/opt/OV/lib/nls	/opt/OV/lib/nls	%OV_MAIN%\lib\nls
$OV_NODELOCK	/var/opt/ifor	/opt/netls/conf	%OV_MAIN%\ifor\ls\co
$OV_PIDS	/var/opt/OV/pids	/var/opt/OV/pids	–
$OV_PRIV_CONF	/etc/opt/OV/conf	/etc/opt/OV/conf	%OV_MAIN%\conf
$OV_PRIV_LOG	/var/opt/OV/share/log	/var/opt/OV/share/log	%OV_MAIN%\
$OV_PRODUCTS	/opt/OV/products	/opt/OV/products	–
$OV_PROG_SAMPLES	/opt/OV/prg_samples	/opt/OV/prg_samples	%OV_MAIN%\prg_sample
$OV_REGISTRATION	/etc/opt/OV/share/registration	/etc/opt/OV/share/registration	%OV_MAIN%\registrati
$OV_RELNOTES	/opt/OV/ReleaseNotes	/opt/OV/ReleaseNotes	%OV_MAIN%\ReleaseNot
$OV_SHARE_LOG	/var/opt/OV/share/log	/var/opt/OV/share/log	%OV_MAIN%\

Table 2-4 Common Universal Pathnames for the ECS Engine (Continued)

Universal Name	HP-UX 10.X, 11.X	Solaris 2.X	Windo
$OV_SNMP_MIBS	/etc/opt/OV/share/snmp_mibs	/etc/opt/OV/share/snmp_mibs	%OV_MAIN%\snmp_mibs
$OV_SOCKETS	/var/opt/OV/sockets	/var/opt/OV/sockets	–
$OV_STACKS	/etc/opt/OV/stacks	/ect/opt/OV/stacks	%OV_MAIN%\stacks
$OV_SYMBOLS	/etc/opt/OV/share/symbols	/etc/opt/OV/share/symbols	%OV_MAIN%\symbols
$OV_TMP	/var/opt/OV/tmp	/var/opt/OV/share/tmp	%OV_MAIN%\
$OV_TOOLS	/opt/OV/tools	/opt/OV/tools	–
$OV_WWW	/opt/OV/www	/opt/OV/www	%OV_MAIN%\

See also

- *ovenvvars*(1)

Integrating User MIBs (DM only)

The Management Information Base (MIB) is relevant to the ECS Engine for HP OpenView DM. This discussion assumes that you are familiar with the DM MIB.

The installation process installs default event metadata for CMIP and SNMP events derived from the following MIB file:

`$OV_GDMO_MIBS/ems.mib`

The following procedure shows you how to augment the event metadata. Augmenting metadata is relevant for both the ECS Designer and the ECS Engine product components.

The metadata is read by the ECS Designer and the ECS Engine once, on start-up. If you change the metadata you must restart these programs for changes to take effect.

DM	You must restart DM itself (using `ovstop`/`opstart`) for changes to the metadata to take effect.

Task

1. Place the MIB files defining the events to be correlated in the `$OV_GDMO_MIBS/` directory.
2. Log in as `superuser`.
3. Type:

ecsber	UNIX
ecsber.bat	Windows NT

   The `ecsber` script (or batch file) takes as input all MIB files in the `$OV_GDMO_MIBS/` directory and creates the metadata files `ecsova.per` or `ecs.def`, `ecs.per`, and `ecs.md` in the `$OV_CONF/ecs/md/ber/` directory.

4. Check that the `ecsber` script executed successfully. The last message the `ecsber` script writes to the terminal should be:

```
Created ecsova.per in $OV_CONF/ecs/md/ber
```

or

```
Created ecs.per, ecs.def and ecs.md in $OV_CONF/ecs/md/ber/
```

5. Examine the output for error messages. If necessary, use the `ovgdmoparse` and `ovmdt` tools to identify other problems.

   The `ecsber` script fails to execute when there are problems with MIB definitions. A common problem is symbols in the default MIB file, `ems.mib`, being duplicated in the user's MIB, especially duplicate definitions of `top`. Comment out unwanted duplicates.

See also

- *ovagen(1m)*, *ovgdmoparse*(1), *ovmdprep*(1m), *ovmdt*(1m), for information about the utilities called by `ecsber`
- `$OV_BIN/ecsber` (*`script`*)
- *HP OpenView DM Agent Platform Administrator's Reference*

Integrating ASCII Metadata (ASCII only)

To encode and decode ASCII[1] events you will require the HP OV Event Correlation Services ASCII Module. This module allows you to compile Message Description Language (MDL) source files into a metadata form that can be loaded by the ECS Designer and the ECS Engine.

The ASCII module is supplied with ECS in NNM.

To compile MDL source:

1. Locate all the MDL source files that you want to encode and decode. The default directory for MDL source files is `$OV_CONF/ecs/ed/mdl/`.
2. Make a backup copy of the existing MDL metadata file. For example, rename it:

   `cd $OV_CONF/ecs/md/mdl/`

   `mv mdl.md mdl.md.old`

   The default metadata file is `$OV_CONF/ecs/md/mdl/mdl.md`, but this can be overridden so check the environment variable `ECS_MDL_MD` for an alternative file.
3. Type the following command to compile all the MDL source files into a single metadata file:

`$OV_BIN/ecsmdt -o $OV_CONF/ecs/md/mdl/mdl.md` ***`file1`*****`.mdl`** ***`file2`*****`.mdl...`**

   where `-o` specifies the location of the metadata output file, and *`file1`*`.mdl`, *`file2`*`.mdl`, etc. is a list of all the MDL source files. All the source files are combined into the specified output file.

All the event descriptions that you want to support must be compiled into *one* metadata file. Usually it is convenient to keep all the MDL event descriptions in a single source file. However, separate source files can be used if desired, and compiled into a single metadata file as described.

See also

- *HP OV Event Correlation Services ASCII Module Guide*

1. CMIP, X733 and ASCII endecoders are not supported any longer. Ignore all references to these endecoders.

ECS Engine Files

The ECS Engine does not require particular filename suffixes. However, because the ECS Designer requires most files to have suffixes indicating the file type, use those recommended in Table 2-5.

Table 2-5 ECS File Types

File Extension	File Type
`*.eco`	Compiled correlation circuits
`*.ds`	Data store files, including updates or dumps from the ECS Engine
`*.fs`	Fact store files, including updates or dumps from the ECS Engine

In addition to the standard file types identified by their extension, there is the default ECS Engine persistence file, `$OV_CONF/ece/<`*instance*`>/config` and the event log files shown in Table 2-6.

Table 2-6 Other Standard ECS Files

File	Description
`ecsin.evt[0|1]`	ECS Engine input event log file
`<`*stream*`>_sout.evt[0|1]`	Stream output default event log
`<`*stream*`>_sdis.evt[0|1]`	Stream discarded event log
`<`*circuit*`>_cout.evt[0|1]`	Circuit output event log
`<`*circuit*`>_sdis.evt[0|1]`	Circuit discarded event log
`<`*stream*`>_pout.evt[0|1]`	Policy output event log
`<`*stream*`>_pdis.evt[0|1]`	Policy discarded event log

There are also files you use for specific optional tasks, such as logging or tracing engine activity. The files are as shown in Table 2-7.

Table 2-7 Files for ECS Engine in DM

File	Description
`pmd.log[0|1]`	Log file for the pmd-linked ECS Engine
`pmd.trc[0|1]`	Trace file for the pmd-linked ECS Engine

The files listed in Table 2-6 are located in the `$OV_LOG/ecs/`*`instance`* universal path and directory where *`instance`* is the ECS Engine instance number. For more information about universal pathnames, see “Setting Up the Environment” on page 28.

See also

- *ecsmgr*(1m)

ECS Engine and Correlation Circuit States

Engine States

A key part of ECS Engine administration is knowing what state the engine and circuit(s) are in, and which command(s) you need to use to effect a change of state.

The ECS Engine has just two states, as shown in Table 2-8. However, there can be many circuits on the same engine and each circuit also has its own state. Issuing a command that changes the circuit state can also change the state of the ECS Engine.

For example, enabling the first correlation circuit moves the ECS Engine into the **Running** state, and disabling the last circuit moves the engine back to the **Idle** state, as shown in Table 2-8.

Table 2-8 States of the ECS Engine

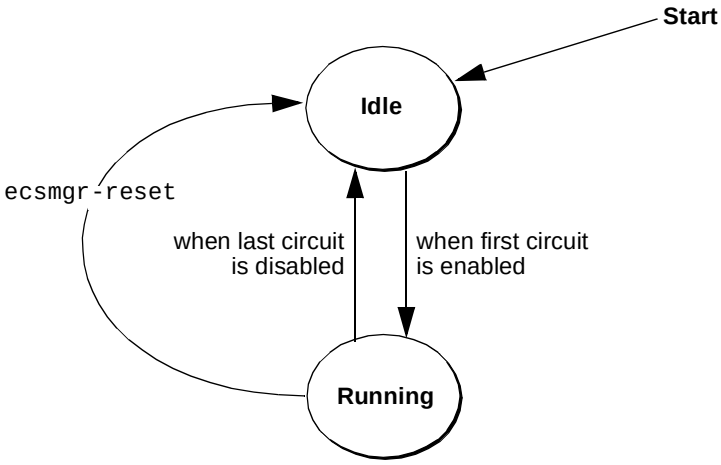
State	Characteristics
Idle	The ECS Engine enters this state when it is started, after a `-reset` is issued, or when the last circuit is disabled.
Running	The ECS Engine enters this state when a circuit is enabled, and leaves it when the last circuit is disabled.

Figure 2-4 shows the `ecsmgr` commands that cause a transition between states of the ECS Engine.

pmd

Issuing an `ecsmgr -reset` command to a pmd-linked ECS Engine may cause the `pmd` memory image to grow. Avoid repeated use of this command. If repeated use is necessary, stop and restart `pmd` occasionally.

Figure 2-4 Management Commands and Engine States



Circuit States

The valid correlation circuit states are shown in Table 2-9 and the transitions between states are illustrated in Figure 2-5.

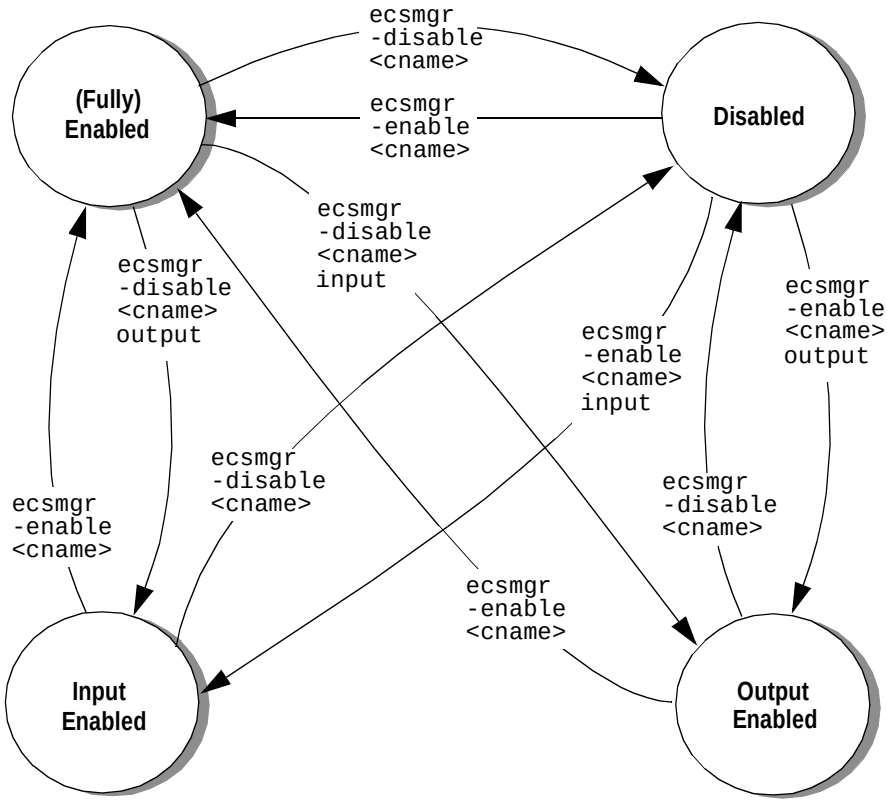
Table 2-9 States of Correlation Circuits

State	Characteristics
Disabled	This is the initial state of a circuit when loaded. No event correlation is occurring within this circuit. If all loaded circuits are Disabled, the ECS Engine state is Idle. If any circuit is enabled, the ECS Engine state is Running. Set a circuit's state to disabled when you want it to have no effect on the event flow.
Input Enabled	The circuit is accepting and correlating events but not contributing to the engine's output. In other words, the ECS engine ignores this circuit when deciding whether or not to output an event. The ECS Engine state is Running. Enable input when you want to prime the circuit with events. For example, Table nodes may need to be exposed to input events for a period before the circuit operation stabilizes.

Table 2-9 States of Correlation Circuits (Continued)

State	Characteristics
Output Enabled	The circuit is correlating events and generating output but not accepting any new events. The ECS Engine state is Running. Enable output only when you want the circuit to run down after normal operation. For example, Unless nodes may need time to pass on stored events.
Enabled	The circuit is accepting and correlating events, and generating output. The ECS Engine state is Running. Enable this state when you want the circuit to correlate events normally.

Figure 2-5 Management Commands and Circuit States



Ensuring Synchronized Timing

Most correlation circuits make assumptions about the time at which events are created. For the circuit to work correctly, the assumptions embedded in its design must be realistic. In some cases this means that all devices that generate events must have their clocks synchronized.

NOTE

When the clocks of event-generating agents or devices are not closely synchronized, the circuit may fail to correlate events as the circuit designer intended.

The ECS Engine operates on UTC (Universal Coordinated Time). The manner in which event creation times are determined is dependent on a number of factors. You should ask the circuit designer and/or the developer that integrated the ECS Engine event I/O system if they have any special requirements for time synchronization and if so, what they are.

Generally, the only requirement is that the real time clock on the machine be accurately set to the correct UTC for where the ECS Engine is located relative to the Greenwich meridian.

3 Operating the ECS Engine

This chapter describes the main tasks associated with operating the ECS Engine. See the next chapter for details on monitoring the engine state.

The chapter begins with:

Then the operation of the ECS Engine is covered in these tasks:

The ECS Engine supports multiple event streams. However, most commands in this chapter are shown without streams. If a particular stream is not specified then the "default" stream is assumed.

If you need to create and deploy multiple streams, see "Managing Streams" on page 64.

The explanations in this chapter assume that you are familiar with the ECS Engine states described in "ECS Engine and Correlation Circuit States" on page 37.

Starting the ECS Engine

Starting a pmd-linked ECS Engine

These ECS Engines are loaded and initialized when the `pmd` is started. All control of the engine is through the `ecsmgr` program.

Starting Event I/O and Annotation

Once the ECS Engine has been started you can start annotation server and event I/O processes. The circuit designer and/or the developer of these processes should provide you with details about how and when to start these programs.

NOTE Do not enable ECS circuits that require specific event I/O or annotation server processes until those processes have been started.

Resetting the ECS Engine

An ECS Engine can be reset at any time. Resetting an engine places it in the same state it was in when the engine was first started. All ECS circuits, Data Stores and Fact Stores are discarded when the engine is reset.

When you initialize the ECS Engine, event logs are rolled over which means `*.evt1` files are deleted and `*.evt0` files are renamed with `*.evt1` suffixes. If you need to keep any `*.evt1` files you should move or rename them before initializing the engine.

If events have been previously logged, the following event log files may be present in `$OV_LOG/ecs/`*`instance/`*:

- `ecsin.evt0` and `ecsin.evt1`

State transition

The ECS Engine can be in any state.

This task changes the engine state to Idle.

Reset the engine

To reset the Open ECS Engine, type:

```
ecsmgr -reset
```

To reset an pmd-linked ECS Engine, delete the file `$OV_CONF/ecs/`*`instance`*`/config` before executing `ovstart`.

Loading an ECS Circuit, Data Store, and Fact Store

The ECS Engine must be loaded with one or more ECS circuits plus, optionally, Data Stores and Fact Stores before correlation can be enabled. You assign a *symbolic name* to ECS circuits, Data Stores and Fact Stores when they are loaded. The symbolic name identifies this circuit in any subsequent commands.

If an ECS circuit uses a Data or Fact Stores, you must load the store(s) before loading the circuit. Then, when the circuit is loaded there are four options:

- correlation circuit alone
- correlation circuit and Data Store
- correlation circuit and Fact Store
- correlation circuit, Data Store, and Fact Store.

You can load any number of correlation circuits and Data and Fact Stores in each ECS Engine. Only one Data Store and one Fact Store can be associated with each circuit. However, stores can be shared by setting several circuits to reference the same Data and Fact Stores.

After a circuit has been loaded it must be enabled, as described in “Enabling an ECS Circuit” on page 50, for event correlation to occur.

Symbolic and file names

You assign symbolic names to correlation circuits, Data Stores, and Fact Stores as they are loaded. These names need not have any relationship to the name of the file from which they were loaded. Since there is no facility in ECS to discover the name of the file from which a circuit or store was loaded, it is recommended that you use names that are easily associated with the original files (the most obvious solution is to base the symbolic name on the file name).

NOTE

When you have loaded a correlation circuit and associated Fact and/or Data store, you cannot delete the stores without first unloading the correlation circuit.

You can update a loaded circuit by executing the `ecsmgr -reload` command. This command simply reloads the same circuit file from the same location, activating any changes made to that file. See "Reloading a Correlation Circuit" on page 61 for details.

State transition

The ECS Engine may be Idle or Running.

The ECS Engine state is not changed. The circuit itself is placed in the Disabled state.

Load a circuit only

To load only a correlation circuit (no associated Data or Fact Stores) type:

ecsmgr -circuit_load *cname cfile*.eco

where *cname* is the name of the loaded circuit and *cfile*.**eco** is the name of the circuit file from which it was loaded.

Circuit and data store

To load a correlation circuit and associated Data Store, you must load the Data Store first, then load the circuit and specify the name of the Data Store:

ecsmgr -data_load *dsname dfile*.ds

ecsmgr -circuit_load *cname cfile*.eco *dsname*

where *dfile*.**ds** is the name of the Data Store file.

Circuit and fact store

To load a correlation circuit and associated Fact Store, you must load the Fact Store first, then load the circuit and specify the name of the Fact Store, type:

ecsmgr -fact_load *fsname ffile*.fs

ecsmgr -circuit_load *cname cfile*.eco 0 *fsname*

where ***fsname*** is a name you assign to the loaded Fact Store and *ffile*.**fs** is the name of the file from which the Fact Store is loaded. The `0` (zero) indicates that no Data Store file has been specified.

Circuit, data and fact stores

To load a correlation circuit and associated Data and Fact Stores, type:

ecsmgr -data_load *dsname dfile*.ds

ecsmgr -fact_load *fsname ffile*.fs

ecsmgr -circuit_load *cname cfile*.eco *dsname fsname*

where *dfile*.ds and *ffile*.fs are the names of the Data and Fact Store files, and *cfile*.eco is the name of the correlation circuit file.

Shared data/fact store

You can associate more than one circuit with the same Data and Fact Store. To do this:

1. Load the Data and Fact stores (using the `-data_load` and `-fact_load` commands.
2. Load the first correlation circuit (using the `-circuit_load` command) and specify the Data and Fact Store names you supplied in the previous step.
3. Load the next correlation circuit and specify the same Data and Fact Store names.

Continue until you have loaded all the required correlation circuits.

Stream

Loading a circuit, data store and fact store is not specific to a stream. You load these components first, then enable the circuit in one or more streams.

Enabling an ECS Circuit

Once an ECS circuit has been loaded it must be enabled before it starts to correlate events. A circuit's input can be enabled or disabled separately from its output. Enabling a circuit places it in one of three states:

- **Input enabled**—the circuit receives events but it does not affect the engine's determination of which events to output. Table nodes and other event storage is primed with events from the input.
- **Output enabled**—the circuit can output events but cannot receive new events. The circuit may output events that were stored in the circuit while the input was enabled, and the output events that are generated by the circuit itself.
- **Enabled**—the circuit can both receive and output events. This is the normal condition of an active ECS circuit.

State transition

If the ECS Engine is in the Idle state it is moved to the Running state when a circuit is successfully enabled. If the ECS Engine is already Running, then there is no change to the state of the engine.

Enable a circuit

To enable both the input and output ports of a loaded correlation circuit, type:

`ecsmgr -enable` ***`cname`***

where *cname* is the name of a loaded correlation circuit.

You can enable a circuit's input and output independently. The correlation circuit starts to affect the flow of events only when both input and output ports are active. This can be useful in a production environment when a circuit needs to be *run up* or *run down*. For example, if a circuit contains a Table node that must be primed with events before the correlation works correctly, you can enable the input, wait until the node is primed, then enable the output.

Enable input only

To enable only the ECS circuit's input ports, type:

`ecsmgr -enable` ***`cname`*** **`input`**

Enable output only

To enable only the ECS circuit's output ports, type:

ecsmgr -enable *cname* output

Once a circuit has been partially enabled using one of these commands, you fully enable the circuit in the usual way:

ecsmgr -enable *cname*

Streams

If a stream is not specified then the correlation circuit is enabled on the *default* stream. To specify a different stream, use the `-stream <stream>` option. For example, to fully enable `myCircuit.eco` on the stream `myStream`:

ecsmgr -stream myStream -enable myCircuit.eco

The stream `myStream` must have been previously created. For details, see “Managing Streams” on page 64.

Enabling Drill Logs

Every stream has associated with it two files

- drill information log
- drill event log

The engine also has it's own correlation info and drill event logs.

The drill logging can be enabled or disabled with the help of the `ecsmgr` commands as below. For information on Drill Down and Custom logging refer to Chapter 5, "Drill Down," on page 95 in the *HP OV ECS Developer's Guide and Reference*.

Enable default drill logging

To enable default drill logs type

```
ecsmgr -log_drill_info on
ecsmgr -log_drill_event on
```

Enable drill logging on new stream

To enable drill logging on a specific stream type

```
ecsmgr -log_drill_info stream stream_name on
ecsmgr -log_drill_event stream stream_name on
```

Register drill log files on new stream

To register the drill log files on a specific stream type

```
ecsmgr -stream stream_name -drill_info_log <path>
ecsmgr -stream stream_name -drill_event log <path>
```

Register drill log files for default logging

To register drill log files for default logging type

```
ecsmgr -log_drill_info stream on
ecsmgr -log_drill_event stream on
```

Loading Perl files

Data external to ECS can be processed using user provided Perl scripts. These Perl files have to be loaded into the ECS engine for further processing. For more information on the format of the Perl file refer to *HP OV ECS Designer's Reference Guide*.

Perl files associated with the circuit must be loaded before the circuit is loaded into the ECS engine. To load the Perl file into the engine

```
ecsmgr -i <instance_number> -load_perl -f <user_supplied
Perl file>
```

Changing the Association Between Stores and Circuits

The Data Store and Fact Store associated with a circuit can be changed while the ECS Engine is running, but the circuits that reference the Data and Fact Stores must be temporarily disabled. If you want to avoid disabling the ECS circuit, you may prefer to update the Data and Fact Stores that have already been loaded. See “Updating the Data and Fact Stores” on page 56 for details.

State transition

The ECS Engine may be Idle or Running. Loaded correlation circuits can be in any state.

If this command unloads the last circuit, the ECS Engine drops back to the Idle state while the circuit is disabled. If the ECS Engine was Idle to start with, this task changes the engine state to Running when the correlation circuit is enabled.

Change data/fact stores

To change the Data and Fact Stores associated with a correlation circuit:

1. Load the new Data and Fact Stores:

ecsmgr -data_load *dsname dfile*.ds

ecsmgr -fact_load *fsname ffile*.fs

where *dsname* is the name of the data store loaded from the data store file called *dfile*.ds, and *fsname* is the name of the fact store loaded from the fact store file called *ffile*.fs. You must make sure that the symbolic names you give to the stores is different from any existing names.

2. Disable each correlation circuit whose association you want to change:

ecsmgr -disable *cname*

where *cname* is the name of the circuit being disabled.

3. Unload each correlation circuit whose association you want to change:

ecsmgr -circuit_unload *cname*

Repeat each of the above two steps for each circuit whose Data and/or Fact Store you want to change. You can then proceed to reload the circuits as described in the next two steps:

4. Load each correlation circuit again and nominate the new Data and Fact Stores:

ecsmgr -circuit_load *cname cfile*.eco *dsname fsname*

where *cfile*.eco is the name of the circuit file to be reloaded, and *dsname* and *fsname* are the names of the new Data and Fact Stores, as specified in the first step.

5. Enable each correlation circuit:

ecsmgr -enable *cname*

6. If necessary, unload the old (no longer used) Data and Fact Stores:

ecsmgr -data_unload *dsname*

ecsmgr -fact_unload *fsname*

where *dsname* is the name of the old data store, and *fsname* is the name of the old fact store. Note that you should only do this if there are no other circuits using these stores.

Streams

Data stores and fact stores are not specific to a stream but correlation circuits are. If a correlation circuit is enabled on a stream other than the default stream then you must specify the stream using the `-stream <stream>` option. See “Enabling an ECS Circuit” on page 50.

Updating the Data and Fact Stores

The Data Store and Fact Store can each be updated while the ECS Engine is in the Running state, and while the circuits that use the stores are fully Enabled. If you update an ECS Engine in the Running state there may be some disruption for one or more of the following reasons:

- Some events are correlated by the contents of the *old* Data and Fact Stores and some by the contents of the *updated* stores.
- Statically evaluated parameters that depend on Data and Fact Store contents are not updated until forced explicitly to be re-evaluated (by issuing the `ecsmgr -circuit_reload` command).
- When there are dependencies between a Data and a Fact Store, correlation may be carried out using the data from a new *Data* Store and relationships from an old *Fact* Store, and vice-versa. You can eliminate this problem if both Data and Fact Store entries are combined in single file and loaded with the `ecsmgr -update` command—see below.

pmd-linked	The postmaster may become stressed if it is heavily loaded and the update contains too many entries.

To minimize the disruption to correlation:

- perform the update when network traffic is low
- combine Data and Fact Store update entries in a single file
- only update a few entries at a time.

The last solution may be useful to avoid stressing the DM postmaster, but has the effect of increasing the chances of using old and new entries together. However, several smaller disruptions may be better than one big one.

See also

See the *HP OpenView Communications Event Correlation Services Designer's Reference* for an explanation of static and dynamic evaluation.

State transition

The ECS Engine may be Idle or Running. Loaded correlation circuits can be in any state.

This task does not cause a change of state.

Before you begin

If necessary, use a text editor to make changes to the data or fact store files.

Before you begin, ask the circuit designer:

- if any statically evaluated parameters need to be re-evaluated
- if there are any dependencies between the Data and Fact Store updates
- whether large Data and Fact Store files can be split into a number of smaller files
- whether Data and Fact Store entries can be combined in a single update file.

Update all stores

To update Data and Fact Stores from a single update file, type:

ecsmgr **-update *cname file***

where *cname* is the name of the circuit associated with the Data and Fact Store to be updated, and *file* is the name of the file containing the store updates. Both data store and Fact store updates can be combined into one file. When the update occurs, the ECS Engine applies the Data and Fact Store updates to the correct store.

Update data store only

To update only the Data Store, type:

ecsmgr -data_update *dsname dfile*

where *dsname* is the name of the Data Store to be updated, and *dfile* is the name of the file containing the store updates.

Update fact store only

To update only the Fact Store, type:

ecsmgr -fact_update *fsname ffile*

where *fsname* is the name of the Fact Store to be updated, and *ffile* is the name of the file containing the store updates.

Reload affected circuits

After the Data and/or Fact Stores have been updated you may want to force the re-evaluation of statically evaluated parameters. Statically evaluated parameters are evaluated only when a circuit is loaded or reloaded. To ensure that references to Data or Fact Store entries in statically evaluated parameters are updated you must complete the following steps for each circuit affected by the update:

1. Disable the circuit:

ecsmgr -disable *cname*

2. Reload the circuit from the circuit file:

ecsmgr -circuit_reload *cname*

   This causes statically evaluated parameters to be re-evaluated and updated Data and Fact Store values to be used.

3. Enable the circuit again to restart correlation:

ecsmgr -enable *cname*

   If necessary you can independently enable the circuit's input and output.

After an update, the Data and Fact Stores may consist of a mixture of entries—some from the files you originally loaded and others from the update files. If you have run many updates you may even have lost track of the files loaded. You can save the current Data and Fact Store to files — see "Dumping Data and Fact Stores to Files" on page 59.

You should also preserve all data and fact store files, so that you can recover the ECS Engine to a known configuration by resetting the engine, loading the original data and fact stores, and then applying all the updates in the same sequence as before.

See the *HP OpenView Communications Event Correlation Services Designer's Reference* for details on how stores are updated.

Streams

Data stores and fact stores are not specific to a stream but correlation circuits are. If a correlation circuit is enabled on a stream other than the default stream then you must specify the stream using the `-stream <stream>` option. See "Enabling an ECS Circuit" on page 50.

Dumping Data and Fact Stores to Files

After you have updated a Data or Fact Store, the contents of the stores in the ECS Engine correspond to the original data and fact store files, plus the net effect of all the updates (deleted, new, and changed entries). You can dump the current state of the store to a file in the same format as the original files. This makes it easy to reconfigure the same or other engine to the same state, and to diagnose problems associated with the state of the Data and Fact Stores.

State transition

The ECS Engine must be in the Idle or Running state. Loaded correlation circuits can be in any state.

This task does not cause a change of state.

Dump the data store

To dump the Data Store to file, type:

ecsmgr -data_dump *dsname file*.ds

where *dsname* is the name of the Data Store to be dumped, and *file*.ds is the name of the file to which the store will be dumped.

Dump the fact store

To dump the Fact Store, type:

ecsmgr -fact_dump *fsname file*.fs

where *fsname* is the name of the Data Store to be dumped, and *file* is the name of the file to which the store will be dumped.

Using stores with the Designer

If you want to load a dumped file into the ECS Designer in Simulate Mode, you must give the filename the correct file extension (.fs for Fact Stores or .ds for Data Stores).

Although the dumped Data and Fact Store files have the same format as the files originally loaded, they do not have any of the comments that may have been in the original files. The headers of the dump files have the creation times set to the time of dumping, and the version number set to zero.

Also NNM data and fact store files have a section marked “Do not edit below this line” which contains current settings made using the NNM Correlation Services GUI.

Also, data and fact store files dumped from an ECS Engine have an offset time for each entry. The oldest version has a relative time of zero. Updates to stores have an offset which is the duration since the oldest version.

Reloading a Correlation Circuit

Reloading a Correlation circuit is done by reference to its *symbolic* name, causing the circuit to be re-read from the (updated) file. A circuit cannot be reloaded if its input or output is enabled in any other stream.

You use this command when the circuit designer changes a circuit file and then supplies a new file (with the same name) for reloading. You also use this command after updating the Data or Fact Store, to force statically evaluated parameters to use the new values.

Data and Fact Stores are unaffected by this procedure; only the named correlation circuit is reloaded.

NOTE

A correlation circuit file must be available with the same pathname and filename as the currently loaded circuit. For example, if the loaded correlation circuit was originally loaded from the file `mycircuit.eco`, then a file called `mycircuit.eco` must exist in the same directory location. If the file cannot be reloaded then the reload fails.

State transition

The ECS Engine may be in the Idle or Running state, and the correlation circuit to be reloaded must be Disabled. If you try to reload a circuit that is Enabled the reload will fail.

This task does not cause a change of engine state. The circuit is in the disabled state before and after being reloaded.

Reload a circuit

To reload a correlation circuit on the default stream, type:

ecsmgr **`-circuit_reload`** ***cname***

where *cname* is the name of the correlation circuit currently loaded.

Streams

The circuit must have been previously disabled from all streams.

Disabling an ECS Circuit

ECS circuits running on an ECS Engine are in one of four states: Enabled, Input Enabled, Output Enabled or Disabled (see "ECS Engine and Correlation Circuit States" on page 37). Only circuits in a disabled state can be unloaded or reloaded. This section describes how to fully disable, or selectively disable just the circuit's input or output.

State transition

The ECS Engine must be Running. The correlation circuit you are disabling must be Enabled, Input Enabled or Output Enabled.

If there is only one circuit enabled, and you completely disable it, the ECS Engine state drops back to Idle. If there are other enabled circuits, the state of these circuits and the ECS Engine remain unchanged.

Disable a circuit

To disable both the input and output of the loaded correlation circuit, type:

`ecsmgr -disable` *`cname`*

where *cname* is the name of the correlation circuit being disabled.

Disable input only

To disable only the circuit input, type:

`ecsmgr -disable` *`cname`* `input`

Disable output only

To disable only the circuit output, type:

`ecsmgr -disable` *`cname`* `output`

Preserving events on Output Streams

To ensure that no events are lost when you disable a correlation circuit in a stream with an output policy, all events that exist within that circuit are flushed out of the Event I/O API (or postmaster in the case of the pmd-linked ECS Engines). You may wish to disable the input to allow the circuit to run down before disabling it.

The flushed events include all detained events but may also include:

- events the correlation circuit has already discarded
- events created but not yet transmitted
- multiple modified copies of original events.

Streams

If the correlation circuit is enabled on a stream other than the default stream then you must use the `-stream` option. For example:

```
ecsmgr -stream test -disable cname
```

Managing Streams

There is a default stream called `default`. This stream has a policy of `output`. The default stream is always present. You cannot remove it.

Examples of when to create additional streams are:

- When you need run correlation circuits that require a `discard` policy.
- When you have an application that subscribes to a stream other than the default stream.
- If you want to create different streams for different applications.
- If you want to create a different stream for test purposes.

Streams are created and removed using the `-create_stream` and `-remove_stream` options for `ecsmgr`. Many `ecsmgr` commands are stream oriented (for example: `-enable_circuit`, `-disable_circuit`, `-policy`). If you want to omit the stream when using these commands, the command is applied to the default stream. If you want the command to apply to any other stream then you must use the `-stream <stream>` option to specify the name of the stream to which the command should be applied.

To create a stream:

`ecsmgr -create_stream <stream> [output|discard]`

If you do not specify `output` or `discard` then the stream policy defaults to `output`. Each stream must have a unique name ('default' is a reserved name).

State transition

The ECS Engine can be in any state. Creating a stream will change the existing state of the engine.

To change the policy of a created stream:

`ecsmgr -stream <stream> output|discard`

The stream policy can be changed only if there are no correlation circuits enabled on it.

To enable a correlation circuit on a stream, first load the correlation circuit and then:

`ecsmgr -stream <stream> -enable`

To log events output by a specified stream:

ecsmgr -log_events stream <*stream*> on

For other stream related event logging and tracing options, see Chapter 4, “Monitoring the ECS Engine,” on page 69.

To remove a stream, first disable all correlation circuits on that stream and then:

ecsmgr -remove_stream <*stream*>

Unloading an ECS Circuit, Data Store, and Fact Store

When an ECS circuit is no longer required to correlate a stream of events it can be unloaded from the ECS Engine, along with its associated Data and Fact Stores (if no other circuit is using them). This frees resources.

If you have made updates to the Data and/or Fact Store, dump the stores to file before you unload them. See “Dumping Data and Fact Stores to Files” on page 59.

State transition

The correlation circuit you want to unload must be Disabled.

This task does not cause a change of state.

Unload a circuit

To unload a correlation circuit from the ECS Engine:

`ecsmgr -circuit_unload` *`cname`*

where *cname* is the name of the circuit to be unloaded.

Unload stores

To unload the associated fact store:

`ecsmgr -fact_unload` *`fsname`*

To unload the associated data store:

ecsmgr **`-data_unload` *`dsname`***

Streams

Unloading circuits or stores is not a stream-oriented command because correlation circuits are associated with a stream only while they are enabled.

Controlling Persistence

An ECS Engine can be configured so that its streams, correlation circuits, data stores and fact stores are restored after a system failure.

Configuration information can be saved to a configuration file and each instance of an ECS Engine can have its own configuration settings.

State transition

The state of the correlation engine(s) will not change from what it was before system failure occurred.

Save the default configuration

To save the current configuration of an ECS Engine to the default configuration file:

```
ecsmgr -save_config
```

Save to a specific configuration file

To save the current configuration to a configuration file called `myConfig`:

```
ecsmgr -save_config myConfig
```

Automatic persistence

The ECS Engine can be configured to automatically saves its current state to the default configuration file whenever any change is made. This configuration file is read whenever the ECS Engine is started. To control automatic saving of configuration information, use:

```
ecsmgr -auto_save on|off
```

Using `-auto_save` causes the state to be saved, thus making this command persistent.

On UNIX, this file is:

`$OV_CONF/ecs/<instance>/config`

and on Windows NT the file is:

`%OV_MAIN%\ecs\<instance>\config`

Restoring a specific configuration

You can restore a previously saved configuration called (for example) `myConfig` with the command:

```
ecsmgr -restore_config myConfig
```

If the configuration file is not specified then the configuration is restored from the default configuration file.

4 Monitoring the ECS Engine

Part of the administrator's responsibilities include monitoring the state of the ECS Engine. This information is needed to determine the detailed state of the engine and each of the circuits, and may also be important to other people, such as circuit designers, who frequently need detailed logs and other information to analyze the effectiveness of circuits. Error logs, trace files and state snapshots can also be requested by support staff when diagnosing operational problems.

This chapter describes the `ecsmgr` options that you use to obtain data about an ECS Engine, correlation circuits, and the Data and Fact Stores.

The first section describes the `-info` option that you will probably use more frequently than any other command `ecsmgr` command. Use it whenever you need to check the state of the ECS Engine and the currently loaded circuits:

- "Displaying ECS Engine Information" on page 71

The ability to log events is central to ECS circuit development and troubleshooting and is described in:

- "Logging Events" on page 78

The other monitoring commands provide more detailed information on the engine state and statistics:

- "Obtaining Engine Statistics" on page 73
- "Logging Errors and Tracing Operations" on page 81.

For a summary of the options described in this chapter, see *ecsmgr*(1m).

Displaying ECS Engine Information

The `ecsmgr -info` command displays the state of the ECS Engine and any loaded correlation circuits, data stores and fact stores. You will frequently use it to verify engine and circuit states before issuing other commands.

State transition

The ECS Engine can be Idle or Running. Loaded correlation circuits can be in any state.

This task does not cause a change of state.

Display engine state

To display the current ECS Engine state, type:

`ecsmgr -info`

The information below will change

Example

This example output from the `ecsmgr -info` command is typical of when the ECS Engine is in the Running state.

```
#ecsmgr -info
engine environment - Full NNM Engine
engine instance - 1
engine version - ECS 3.2 (A.03.20)
time last started - Wed Oct 16 22:23:20 2002
engine trace mask - 0x10000
engine log mask - 0x7
maximum engine log size - 512 KBytes
maximum event log size - 512 KBytes
input event logging - off
default drill info logging - off
default drill event logging - off
automatic configuration saving - on

stream name - default
```

```
stream policy - output (unless discarded by a circuit)
stream event logging - off
stream policy event logging - off
stream drill info logging - off
stream drill event logging - off
circuits enabled in stream - <none>

circuit name - sced1
circuit date - Wed Oct 16 21:49:35 2002
circuit version - 0
circuit unique identifier - 11043977
time circuit load - Wed Oct 16 22:23:49 2002
circuit event logging - off
circuit state - disabled
data store name - sced
data store date - Wed Oct 16 14:08:25 2002
data store version - 1
time data store load - Wed Oct 16 22:23:49 2002
```

Obtaining Engine Statistics

The engine statistics includes details about every node in every correlation circuit in the ECS Engine. You can examine the statistics in conjunction with the event logs and the ECS circuit itself to determine if the ECS Engine is operating as expected.

State transition

The ECS Engine can be Idle or Running. Loaded correlation circuits can be in any state.

This task does not cause a change of state.

NOTE

Because this command returns a large amount of information, you may want to redirect the output to a file.

Display engine stats

To display engine statistics, type:

```
ecsmgr -stats
```

The amount and usefulness of the output varies depending on the state of the ECS Engine and loaded correlation circuits. The most useful statistics are obtained when the engine is Running and at least one circuit is Enabled.

The display verbose statistics, type:

```
ecsmgr -stats verbose
```

The verbose option may be useful if you want to obtain statistics about the internal operation of circuits during the design phase. The command prints statistics for every circuit node.

Example

Following is an example of the statistics returned by the `ecsmgr –stats verbose` command for a very simple correlation circuit containing a Source node and a Sink node, connected by a single Filter node with the fully qualified name `passthru_module.tmpcompound.tmpnode.filter_1_`.

```
# ecsmgr -stats verbose
Engine Statistics -

  input.inputFilters = [((), (), 2.9.3.2.10.4), ((), (), “”),
                                                ((), (), ())]
  engineInstance = 36
  currentTime = 19980407073924.000000Z
  enginelog.errors = 0
  enginelog.warnings = 0
  enginelog.info = 0

Stream Statistics -


  Stream “default” -

    default.in.input = 1
    default.in.new = 0
    default.out.output = 1
    default.out.discarded = 0
    default.out.undecided = 0
    default.out.errors = 0
    default.original.output = 1
    default.policy.num = 1

Circuit Statistics -


  Circuit “circuit1” -

    circuit1.in.input = 0
    circuit1.in.new = 0
    circuit1.out.output = 0
    circuit1.out.discarded = 0
    circuit1.out.undecided = 0
    circuit1.out.errors = 0
    circuit1.original.output = 0

Circuit Node Statistics -


  Circuit “circuit1” -

  extraction_module.tmpcompound.external_port_output_ecmip.
                                              input_numin = 0
  extraction_module.tmpcompound.tmpnode.count_passthru.
```

```
                                    increment_input_numin = 0
  extraction_module.tmpcompound.external_port_output_passthru.
                                  input_maxTD = 596523h14m7s
  extraction_module.tmpcompound.tmpnode.modify_cmip_site.
                               input_minTD = --596523h-14m-8s
  extraction_module.tmpcompound.tmpnode.filter_snmp_site.
                                input_minTD = --596523h-14m-8s
  extraction_module.tmpcompound.tmpnode.modify_snmp_site.
                                input_minTD = --596523h-14m-8s
  extraction_module.tmpcompound.tmpnode.filter_snmp_site.
                           error_output_maxTD = 596523h14m7s
...
```

These statistics refer to the overall inputs and outputs of the ECS Engine as described in Table 4-1 and Table 4-2.

Although every event that enters the engine will enter every stream, the `in.input` value is kept on a per stream basis. This is because a new stream may be created after the engine has already received events and other stream-based statistics are more meaningful when compared with the number of events that have actually entered that stream.

The default stream (which is always present) shows the total number of events that have entered the engine.

Table 4-1 Engine Statistics

Statistic	Description
`currentTime`	The current engine time.
`input.inputFilters`	A description of which events will be received by a circuit within the engine (used by OVO).
`enginelog.errors`	The number of errors created for the engine log (regardless of the current engine log mask).
`enginelog.warnings`	The number or warnings created for the engine log (regardless of the current engine log mask).
`enginelog.info`	The number of information messages created for the engine log (regardless of the current engine log mask).

Table 4-2 Stream and Circuit Statistics

Num	Statistic	Formerly	Description
1	`in.input`	input.numEvents	The total number of events that have entered each stream.
2	`in.new`	-	The number of new events in each stream (for example, created or modified events).
3	`out.output`	-	The total number of events output on each stream.
4	`out.discarded`	-	The number of events discarded in each stream either by circuits or by the stream policy (that is, ref.count = 0 and not output).
5	`out.undecided`	-	The number of events currently undecided within each stream (that is, ref. count > 0 and not output.
6	`out.errors`	output.discards	The number of events discarded because of error (due to either endecode errors on input, or errors on output).
7	`policy.num`	-	The number of events that did not enter any circuit within the stream.
8	`original.output`	-	The number of original events output on each stream (that is, total - created or modified)

NOTE These statistics *never* count temporary or composite events, and they do not count created events until the events are output from the engine.

The number of events seen by a stream is the number of events input into the stream plus the number of events created within the stream. Therefore the total number of events = (1) + (2).

Every event in a stream will either be output on the stream (3), discarded in the stream (4), discarded by error (6), or currently undecided (5). Therefore, at any point in time (1) + (2) = (3) + (4) + (5) + (6).

The overall effectiveness of a stream on reducing the event flow is calculated with (3) / (1).

The proportion of the original event flow preserved by a stream is calculated with (8) / (1).

The proportion of events that are accepted by a circuit is calculated with [(1) - (7)] / (1). This is like “coverage” for events.

Table 4-3 Node Statistics

Statistic	Type	Description
*portname*_numin	integer	The number of events that have arrived at this input port since node creation. (*portname* is input, reset_input, etc.)
*portname*_minTD	duration	Minimum transit delay of events arriving at this input port.
*portname*_maxTD	duration	Maximum transit delay of events arriving on this input port.
*portname*_numout	integer	The number of events that have exited through this output port since node creation.
*portname*_minTD	duration	Minimum transit delay of events exiting from this output port.
*portname*_maxTD	duration	Maximum transit delay of events exiting from this output port.

Logging Events

The ECS Engine can log events as they arrive at the engine, as they leave in a specific stream, or as they leave a specific correlation circuit. It can also log events that fail to enter any circuit within a stream and which are therefore handled according to the stream's policy.

Each event log file contains ASCII representations of the logged events in a form can be displayed and edited using conventional text editing tools. See the *HP OpenView Event Correlation Services Designer's Reference* for details about the event log file format.

Log file roll over

The event log files with `*.evt0` suffixes grow until they reach the maximum log size, then they are renamed with `*.evt1` suffixes, and logging continues with new `*.evt0` files. The previous `*.evt1` log files are overwritten. You may want to move or rename existing `*.evt1` log files.

The maximum log size is set to 512K bytes by default. To set the maximum event log size to some other figure, type:

`ecsmgr -max_log_size event kbytes`

where *kbytes* is the new maximum log size in Kilobytes.

CAUTION

Event logging can affect performance and should only be used for testing during correlation circuit design and commissioning, or troubleshooting during operation

State transition

The ECS Engine must be in the Idle or Running state. However, circuit log commands do not take effect until the circuit is enabled.

This task does not cause a change of state.

Input event logging

Input events enter every configured stream, so input event logging is not stream specific. To enable | disable the logging of all events that arrive at the engine, type:

`ecsmgr -log_events input on|off`

The input events log is written to `ecsin.evt0`.

This file contains events received into correlation circuits and also events discarded by the filters at the external input ports of circuits.

If the correlation circuit contains Annotate nodes to which an annotation server responds, then these responses are logged to `ecsin.evt0` also.

Stream event logging

Output events can be logged on a per-stream basis.When event logging is enabled, events output by a stream are written to *`streamname`*`_sout.evt0` and events discarded by the stream are written to *`streamname`*`_sdis.evt0`.

To enable|disable the logging of events leaving via a particular stream, type:

`ecsmgr -log_events stream[`*`streamname`*`] on|off`

If the stream name is omitted then the default stream is assumed, and events are logged to `default_sout.evt0` and `default_sdis.evt0`.

Circuit event logging

Output event logging can be on a per-circuit basis. When circuit event logging is enabled, events output by the circuit are logged to *`circuitname`*`_cout.evt0` and events discarded by the circuit are logged to *`circuitname`*`_cdis.evt0` and events discarded by the circuit are logged to *`circuitname`*`_cdis.evt0`.

To enable|disable logging events leaving a particular circuit:

`ecsmgr -log_events circuit `*`circuitname`*` on|off`

NOTE

Unlike the engine input and output log files, *circuit* log files do not roll over from *`circuitname`*`.evt0` to *`circuitname`*`.evt1` when the `ecsmgr -reset` command is issued (although they do roll over when the maximum log size is reached).

Annotate requests are logged to the circuit output logs. Annotate responses are logged to the engine input logs.

Policy event logging

Events that do not enter any circuit within a stream are handled according to the policy of the stream. When policy event logging is enabled, those events output by the stream's policy are written to *`streamname`*`_pout.evt0` and those events discarded by the stream's policy are written to *`streamname`*`_pdis.evt0`.

To enable|disable policy event logging, type:

`ecsmgr -log_events policy[`*`streamname`*`] on|off`

If the correlation circuit contains Annotate nodes that send requests to an annotation server, then these annotation requests are also logged to `ecsout.evt0`. Annotation responses are logged to the input events log which is written to `ecsin.evt0`.

You can begin logging both input and output events simultaneously by executing `ecsmgr -log_events on`. You can end *all* logging by executing `ecsmgr -log_events off` no matter which command you entered to start event logging.

Logging Errors and Tracing Operations

You can log error messages from the ECS Engine to the log file, and trace the internal operations of the ECS Engine to the trace file. In each case the level of detail written to the file is controlled by a bitmask.

CAUTION

Tracing should only be used during testing or troubleshooting. Tracing records every action of the ECS Engine, both normal and abnormal, and can affect performance. Avoid activating tracing during normal operations. On the other hand, error logging is only activated when a problem occurs and does not impact performance during normal operations. Error logging can and should be enabled during normal operation.

DM and NNM

Logging and tracing require you to align Logging and Tracing Masks for the DM and NNM postmaster with those for the ECS Engine, as explained in the following section.

Setting the Postmaster Log and Trace Mask (DM and NNM)

The ECS Engine and postmaster log settings must be aligned. The ECS Engine log and trace settings are controlled by two separate bit masks that control the level of detail in the log and trace. When the ECS Engine is linked to the postmaster, log and trace information is output to the `pmd.log0` and `pmd.trc0` files. The postmaster has its own mask to control log and trace information. Only log and trace information that conforms with *both* settings is output to the `pmd.log0` and `pmd.trc0` files.

Postmaster log and trace settings are configured with the postmaster's `pmdmgr` command. ECS Engine log and trace settings are configured with the `ecsmgr` command. This task shows how to set the postmaster's logging and tracing mask. Once the postmaster has been set you can control logging and tracing in the ECS Engine.

The postmaster's mask is set to the default value of `0x00400007` in the `$OV_LRF/pmd.ecs.lrf` file. In this default mask:

- the value `0x00000007` enables the logging of disaster, error, and warning messages
- the value `0x00400000` enables tracing.

You can change the postmaster's mask value while the postmaster and the ECS Engine are running.

Determine what level of logging you require from Table 4-4, then sum (OR) the individual hex values to obtain the value of *mask*. The value of *mask* can be expressed in decimal or hexadecimal format.

Table 4-4 Postmaster ECS Stack Mask Values

Postmaster Message Type	Mask Value	ECS Message Type
DISASTERS	0x0000000**1**	ETL_LOGDISASTER
ERRORS	0x0000000**2**	ETL_LOGERROR
WARNINGS	0x0000000**4**	ETL_LOGWARNING
INFORM	0x0000000**8**	ETL_LOGINFORM
–	0x00**4**00000	ETL_TRACE

For example, to enable logging of disasters, errors and informative messages only (and *not* warnings) you sum `0x00000001` + `0x00000002` + `0x00000008` = `0x0000000b`. The ECS Engine should have the same the log mask, or a subset of it. To continue the example, if the engine's log mask is set to `0x00000004`, allowing only warning messages, then *no* messages are logged because none of the bits in the two masks match up.

CAUTION The postmaster's mask affects both logging *and* tracing, so when changing the logging level be careful not to change the tracing level inadvertently, and vice-versa. If all bits of the postmaster's mask for the ECS Engine's stack are set (`0xffffffff`), a `pm_xxx_timer` message appends to the `pmd.trc0` file once per second, degrading the performance of the postmaster.

Set postmaster log/trace mask

To set the postmaster log and trace mask, type:

pmdmgr -SECSS\;T *mask*

where T*mask* is the postmaster log and trace mask.

Continue with "Enabling the ECS Engine Log" on page 83 or "Enabling the ECS Engine Trace" on page 84, if required.

See also

- *pmdmgr*(1m)

Enabling the ECS Engine Log

Consider activating the engine log at all times. The engine log does *not* affect the performance of a running ECS Engine since logging only occurs when errors are generated, and this is exactly when you need the additional information.

State transition

The ECS Engine may be Idle or Running. Loaded correlation circuits can be in any state.

This task does not cause a change of state.

DM and NNM	The postmaster's logging and tracing mask for the ECS Engine's stack must have been set to allow the required level of logging for the engine log — see "Setting the Postmaster Log and Trace Mask (DM and NNM)" on page 81.

Log engine errors

To set the level of engine logging, type:

ecsmgr -log *mask*

Any `ecsmgr` command errors and ECS Engine error messages are now append to the log file.

NOTE

By default the mask is set to `ecsmgr -log 0x00000007` to log `ETL_LOGDISASTER`, `ETL_LOGERROR`, and `ETL_LOGWARNING` messages.

See also

- *ecsmgr*(1m)

Enabling the ECS Engine Trace

When engine tracing is enabled, the ECS Engine appends messages to the engine trace file `$OV_LOG/pmd.trc0` for the postmaster. These messages document both normal operations *and* errors, and can include messages sent to the engine log, plus trace messages for internal operations.

CAUTION Tracing can affect performance. Avoid activating tracing during normal operations. The main value of the engine trace is during design, commissioning, and troubleshooting of correlation circuits.

Determine what level of tracing you require, from Table 4-5, then sum (OR) the individual values to obtain the value of *mask*. The *mask* can be expressed in decimal or hexadecimal format.

Table 4-5 ECS Engine Trace Mask Values

Type of Message	Mask Value (hex)
ETL_EVENTRANSFER	0x0000000**1**
ETL_EVENTDISCARD	0x0000000**2**
ETL_EVENTCREATE	0x0000000**4**
ETL_DEBUGGER	0x0000000**8**
ETL_CREATEDELETE	0x0000001**0**
ETL_INITIALISE	0x0000002**0**
ETL_ENTRYEXIT	0x0000004**0**
ETL_LOOKUP	0x0000008**0**
ETL_MEMORY	0x0000010**0**
ETL_PROCESSINGINFO	0x0000020**0**
ETL_CIRCUITLOAD	0x0000040**0**
ETL_INVARIANTFAIL	0x0000080**0**
ETL_INTERP	0x0000100**0**

Table 4-5 ECS Engine Trace Mask Values (Continued)

Type of Message	Mask Value (hex)
ETL_INTERP_DETAIL	0x0000**2**000
ETL_EVENTDELETE	0x0000**4**000
ETL_MANAGEMENT	0x0000**8**000
ETL_SYSTEM_TRACE	0x000**1**0000

State transition

The ECS Engine may be Idle or Running. Loaded correlation circuits can be in any state.

This task does not cause a change of state.

DM and NNM

The postmaster's logging and tracing mask for the ECS Engine's stack must have been set to enable tracing for the correlation — see "Setting the Postmaster Log and Trace Mask (DM and NNM)" on page 81.

Saving a Snapshot of the Correlation Engine

Complete information about the correlation engine and the installed ECS circuit can help in determining if the correlation engine is correlating as intended. This task shows you how to save a snapshot of a running engine to a file.

State transition

The correlation engine should be in the Running state.

CAUTION

This command saves a very large quantity of information and can take some time to complete. Events may be lost or the correlation may be adversely affected, so only use this command when diagnosing problems.

Procedure

In the following procedure, the snapshot file is assumed to be saved in the present working directory with the name of *myCircuit*.ss.

- Type:

ecsmgr -snapshot *myCircuit*.ss

The resulting snapshot file myCircuit.ss contains information about the state of the correlation engine at the time of the snapshot, including events at various nodes in the ECS circuit. Some of this information can also be obtained by ecsmgr –info and ecsmgr –stats.

5 Troubleshooting the ECS Engine

This chapter begins with common problems you should check before attempting to diagnose a problem with the ECS Engine:

- “Eliminating Common Faults” on page 89.

It continues with a description of the `ecsmgr` options used to diagnose failures and restore operations:

- “Recovering from a Failure” on page 93
- “Verifying an Installation” on page 95.

See also

- *ecsmgr*(1m)

Eliminating Common Faults

Useful information

The following sources of information may describe late changes in the product and assist in troubleshooting:

- release notes in `$OV_RELNOTES`
- *ecsmgr*(1m) manpage

NOTE

When errors occur, the ECS Engine attempts to send an error message to the engine log file. Always run the ECS Engine with error logging enabled—see “Logging Errors and Tracing Operations” on page 81.

Checklist of faults

Before attempting to diagnose a problem, you might consider whether one of the following situations may be the cause.

The ECS Engine does not run.

❑ Is a license available (check the `pmd.log0` files)?

— Is the ECS Engine able to access the license server?

You might not be able to start an ECS Engine if the license server is inaccessible owing to network faults.

— Are there more ECS Engines attempting to run than there are licenses?

You must purchase more licenses or reduce the number of running engines.

— Has a demonstration or evaluation license expired?

You must purchase a license to continue to use ECS.

See the *HP OpenView Event Correlation Services Installation Guide* for additional information about licenses.

❑ Are manpages unavailable?

— Make sure `$OV_MAN/` is included in your `$MANPATH` variable.

In some operating systems, the `man` command does not search the default directories when the `$MANPATH` variable is defined explicitly—see *man*(1). In these cases you should also include the default `man` paths in your `$MANPATH` variable.

❑ Is on-line help unavailable?

— Make sure `%OV_MAIN%/` is included in your `PATH`.

For example, `PATH =%OV_MAIN%;%PATH%`

❑ Is the ECS Engine (and/or DM (or NNM)) running?

— Check the ECS Engine with the `ecsmgr -info` command.

See also "Recovering from a Failure" on page 93 for a detailed diagnostic procedure. The procedures in Chapter 4, "Monitoring the ECS Engine," on page 69, describe how to capture other data useful in diagnosis.

A correlation circuit cannot be loaded or enabled.

❑ Is the ECS Engine continually saying `Failed to reload circuit` or `Failed to start engine` when you execute an `ecsmgr` command?

The `ecsmgr -reload` command attempts to reload a correlation circuit from the original filename and path. If the file is moved or renamed, the ECS Engine unloads the circuit but is unable to reload it. You must reset the ECS Engine and load all circuits, data stores, and fact stores again.

Also check that you are using the right instance number. If you do not specify an instance then it defaults to 1 (one).

❑ Do the files have the correct read and write permissions?

Change file read, write and ownership permissions as required.

The correlation circuit is overloaded, or correlating inefficiently or incorrectly.

❑ Is there sufficient memory in the host machine?

Although you may have the recommended amount of memory, some correlation circuits may require more. Discuss with the circuit designer whether:

— the correlation circuit contains errors in logic

- — the correlation circuit is too large or complex
- — too many events are being stored in the correlation circuit
- — too many events are being combined or created
- — the Data and Fact Stores are too large.

❑ Are events failing to reach other applications in time?

- — Are you logging events to event log files?
- — Is the engine trace enabled?

Both event logging and engine trace consume considerable system resources, so that the ECS Engine may be unable to keep up with the event stream. Turn off logging and tracing unless diagnosing correlation circuits and the correlation process.

It is recommended that the engine error log be enabled at all times as no performance penalty is incurred until an error occurs.

❑ Are the file permissions set to allow ECS circuit, Data Store, and Fact Store files to be loaded?

Change file read, write and ownership permissions as required.

Considering the problem

If the preceding factors are not the cause of the problem, consider the list of aspects below. Consider how diagnostic information can help isolate the problem—see Chapter 4, “Monitoring the ECS Engine,” on page 69, for possible troubleshooting options.

- What is the scope of the problem?

  What functionality is affected? What functionality is *not* affected?

- Has the context changed?

  Has any software been updated or reconfigured: operating system, HP OpenView Distributed Management products, ECS Engine, DM or NNM, correlation circuits, and Data and Fact Store files?

  Has the file system been changed: any reorganization, renaming, altered permissions, changed mounts, and so on?

- What is the time of occurrence, the duration, and the frequency of the problem?

  Does the problem occur at the same time of day, or on a particular day of the week?

Does it always last about the same length of time?

Does it recur often or infrequently?

What other activities occur with the same pattern of incidence that could be causing the problem?

- Can the problem be reproduced?

  While reproducing the problem, collect as much data as you can to pinpoint the problem.

Recovering from a Failure

You may need to determine whether the ECS Engine or the postmaster is the possible cause of a problem, and how to restore operations.

1. Check the status of the ECS Engine by typing:

   `ecsmgr -info`

   The `ecsmgr -info` command returns the current state of the ECS Engine, or times out. Did this command return the current state?

   - **Yes:** Go to the procedure in Chapter 3, "Operating the ECS Engine," on page 43 that is appropriate to the current state of the ECS Engine.
   - **No:** The ECS Engine may be heavily loaded, otherwise there is no engine running with the specified instance number. If this command times out or returns a message such as `Failed to obtain engine status`. Go to Step 2.

NOTE

If `Connection to engine failed` appears, this means that there is no engine running with a matching instance number.

2. To check the status of the postmaster, type:

   `ovstatus pmd`

   This command returns `object manager name: pmd` plus other information about the postmaster. Is the postmaster running?

   - **Yes:** The ECS Engine may have failed or be in an indeterminate state. Go to Step 1
   - **No:** Examine the postmaster log and trace files `$OV_LOG/pmd.log0` and `$OV_LOG/pmd.trc0` for clues to the failure, then take corrective action.

   Go to "Verifying an Installation" on page 95 if you suspect that the ECS Engine files may be corrupted.

Preservation of events

When you execute the `ecsmgr -reset` command while the ECS Engine is running, the current correlation circuits are automatically unloaded. Events that are currently in a circuit may be output or not, depending on the policy of individual streams. See “Managing Streams” on page 64.

See also

- *ecsmgr*(1m), *ovaddobj*(1m), *ovdelobj*(1m), *ovstart*(1m), *ovstatus*(1m), *ovstop*(1m)
- *HP OpenView Distributed Management Agent Platform Administrator's Reference*

Verifying an Installation

To verify the installation process, you can run a confidence test at any time after the installation. The confidence test runs for approximately two minutes and reports on the status of the ECS installation.To run the test, execute the following:

```
$OV_BIN/ecsconftest
```

NOTE You must be logged in as root (or have superuser access) to run the `ecsconftest` command.

For information about the options you can use with this command, see the *ecsconftest*(1m) reference page.

The confidence test attempts to start the postmaster (`ovstart`) if it is not already running. DM or NNM is left running after the confidence test has completed.

All modules must be correctly installed, configured and licensed for the confidence test to run.

Glossary

Abstract Syntax Notation 1 (ASN.1) An OSI standard related to the Presentation Layer where the abstract representation of the data is independent of its physical encoding. It is specified in ISO/IEC 8824, X.208.

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

annotation API A set of application program interface functions and data structures that supports the transfer of data between an external annotation server and one or more Annotate nodes in an ECS circuit.

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, and the information returned is typically obtained external to the ECS Engine and the annotation server.

arrival time The time an event arrives at the ECS engine in Universal Coordinated Time (UTC).

ASCII American Standard Code for Information Interchange. A standard used by computers for interpreting binary numbers as characters.

ASN.1 Abstract Syntax Notation 1.

attribute An object characteristic or property that describes the current state of the object and which has a unique identifier by which it is accessed. In ECS, for example, the “eventTime” attribute of a CMIP event, or the “Rate” attribute of a Rate node. See event attribute; identifier; correlation node attribute.

attribute-value pair The combination of an attribute identifier and the value of that attribute for a specific object. In ECS, attribute-value pairs are represented as key-value pairs in an ECDL dictionary. See also key-value pair; dictionary.

Basic Encoding Rules (BER) Defines how ASN.1 data types are encoded for transport on the network.

breakpoint A point in a program at which execution is halted so that the program’s status, contents of variables and other factors can be examined. In the ECS Designer, in simulation mode, breakpoints are locations in a correlation circuit where event processing is halted to allow for manual intervention.

canvas The working area of the ECS Designer screen. This is where you place, connect, and configure correlation nodes to create your correlation circuit.

CCITT The International Telegraph and Telephone Consultative Committee, an international organization concerned with proposing recommendations for international communications. Replaced by the International Telecommunications

Union, Telecommunications (ITU-T) in 1992. See International Telecommunications Union, Telecommunications (ITU-T).

circuit *See See correlation circuit.*

CMIP *See See Common Management Information Protocol (CMIP).*

Common Management Information Protocol (CMIP) A protocol for exchanging network management information in an OSI environment (ISO/ITU-T X.710). CMIP communicates management information between a manager and an agent. CMIP allows a manager to retrieve (get) management information from, or to alter (set) management information on an agent. CMIP also allows the manager to create and delete instances of an object managed by the agent, or perform an action on an object. An agent can also emit unsolicited messages, called notifications, to alert managers of noteworthy local conditions.

component event An event that is combined with other events to create a new event. In ECS, a composite event is composed of two or more component events. See composite event.

composite event In ECS, a composite event consists of a structured aggregation of addressible 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. See also component event; primitive event; temporary event.

compound node A graphical element that represents a container of lower level components. The lower level components will be displayed when the user opens the compound node. In ECS, a correlation circuit fragment may be encapsulated in a compound node, hence creating a new user-defined correlation node. Compound nodes may be added to libraries and re-used by reference or by copy. Compare with primitive node.

condition (parameter) In ECS, a condition is an ECDL expression specified for a correlation node parameter, usually involving attribute from an event, that returns a value used to modify the behavior of the correlation node.

correlation A procedure for evaluating the relationship between sets of data or objects to determine the degree to which changes in one are accompanied by changes in the other. In ECS, correlation is a process of analyzing a stream of events by filtering and detecting patterns and replacing groups of events with single events that have (possibly) higher information content.

correlation circuit In ECS, a collection of interconnected primitive nodes 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, to be accepted from the external event stream. A correlation circuit can be loaded into an ECS Engine.

correlation circuit port The logical connections between a correlation circuit and the containing infrastructure where events enter and leave the circuit. These ports may be configured to select a subset of events in the input event stream, based upon event encoding type and event syntax. A single port may be connected to multiple Source/Sink nodes, and a single Source/Sink node may be connected to multiple circuit ports.

correlation engine The ECS runtime 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 the one or more correlation circuits loaded into the correlation engine.

correlation node A processing element in a correlation circuit. See also compound node; primitive node.

correlation node attribute A property of a correlation node that can be read from another correlation node. The Count, Rate, and Table nodes have attributes (which may be exported by a containing compound node as attributes of the compound node). Attributes are addressed using a dot notation: “node_name.attribute_name”.

correlation node parameter In the ECS Designer, a correlation node parameter is an ECDL expression used to configure a correlation node.

correlation node port One of possibly many connection points of a correlation node used to interconnect correlation nodes. Events enter a correlation node through a port and leave a correlation node through a port. Port types include input, output, control, reset, and error ports. In the ECS Designer, ports visually indicate the sense of the associated event flow. Optional ports are not displayed by default.

creation time The time an event was created. Inside the ECS Engine creation time is represented in Universal Coordinated Time (UTC).

daemon A process that “serves” clients. Sometimes referred to as a server.

data store In ECS, a component of the ECS Engine which holds user-specified named data items of an ECDL data type. The entries in the data store may be referenced from the ECDL expressions configured into the correlation nodes. A correlation circuit may be associated with one of the possibly many data stores loaded into the correlation engine.

data type A particular kind of data; for example integer, alphanumeric, boolean, date. In ECS, data types are ECDL data types which define the type and range of values to which an identifier may be assigned. Every value in ECDL has a data type, but the type need not be explicitly stated. The types range from simple types such as integers, to compound types such as dictionaries and lists, and special types such as functions and events.

dictionary (data type) In ECS, a dictionary is an ECDL data type comprised of an unordered list of key-value pairs. Any value is accessed via reference to the key.

Within ECS, an event is treated as a dictionary with attribute names being the dictionary keys which provide access to the attribute values.

Distributed Management Platform (DM) HP OpenView Communications Distributed Management Platform, the platform which provides the infrastructure for implementing OSI-based management solutions.

DM *See See Distributed Management Platform (DM)*

duration data type In ECS, a duration is an ECDL data type used to represent relative or elapsed time values. Compare with time data type.

dynamic parameter A parameter whose value is determined during program execution. In ECS, an ECDL expression configured for a correlation node parameter which is evaluated each time an event enters the correlation node. Typically, the value returned by a dynamic parameter changes for each event processed.

ECDL *See See Event Correlation Description Language (ECDL).*

ECS *See See Event Correlation Services (ECS).*

ECS circuit *See See correlation circuit.*

ECS Designer The ECS Designer is the ECS component which you use to create and test correlation circuits. The ECS Designer works in two modes: build mode where you create correlation circuits, and simulate mode where you test the circuits.

ECS Engine *See See correlation engine.*

ecsmgr The command line program used to administer a running ECS Engine.

endecode In ECS, a term used to refer to a combined encoding or decoding function or capability. An endecode module is an architectural entity which provides encoding and decoding for a specific type of event.

evaluation license A license granted for a specific period of time for the purpose of evaluating ECS.

event An event is an unsolicited notification such as an SNMP trap, a CMIP notification, or a TL1 event, generated by an agent process in a managed object or by a user action. Events usually indicate a change in the state of a managed object or cause an action to occur. In ECS, an event is encoded as a primitive, compound, or temporary event. ECS events contain header attributes added to the input events to assist the processing of the events while they are in the ECS correlation circuit. The header attributes are stripped before the events are transmitted from the ECS circuit.

event attribute A characteristic property of an event. In ECS, event attributes are either part of the internally created event header common to all event types, or part of the event body that contains the input event.

Event Correlation Description Language (ECDL) The language used to specify correlation circuits (node relationships, parameter expressions, data and fact store values) for the ECS Engine.

Event Correlation Services (ECS) The HP OpenView Communications Event Correlation Services product.

event encoding type The first and highest level in the three-tiered ECS event classification system. An event's encoding type determines the endecode module that will be used to translate the event to and from its native format. For example, CMIP notifications and SNMP traps both use the BER encoding type. ASCII events use the MDL encoding type, and OVO messages use the OVO encoding type. See also event syntax; event type

event flow An ECS circuit represented graphically as a circuit schematic consisting of correlation nodes interconnected by lines (connections). See also correlation circuit.

event body The body of an event depends on the event class. The body of a primitive event is the original message, trap or event; the body of a temporary event may be empty; and the body of a composite event consists of other events.

event header Inside ECS and event is augmented with additional information such as the event encoding type, event syntax, event type, and event class. This information is carried in a header that is attached to the event body. See also event body.

event I/O API A set of application program interface functions and data structures that supports the input and output of events to and from the ECS Engine.

event syntax The rules governing the structure and content of an event. In ECS, the event syntax is the second level in the three-tiered ECS event classification system. An event's syntax determines how the event's attributes are read and written. For example, SNMP traps have an event syntax of Trap-PDU and CMIP notifications have an event syntax that evaluates to an OID identifying the GDMO notification. ASCII events have a syntax determined by the MDL definition used to read and write them. See also event encoding type; event type.

event type A classification of an event into a particular category that further defines the nature of the event. In ECS, the event type is the third and lowest level in the three-tiered event classification system. The event type is represented by the ECS header attribute "event_type". For SNMP traps the event type is the generic trap number (1-6). The CMIP event type is the OID of the notification. ASCII events have an event type determined by the MDL definition used to read and write them. See also event encoding type; event syntax.

expiry time Annotation requests are valid for a limited time, determined by the Annotate node's Time Limit parameter. The expiry time is the time at which the annotation request was generated plus the Time Limit. In other words, it is the time at which the request expires.

expression In general, a set of reserved words, symbols, variables, and functions that is evaluated to provide a result. In ECS, an expression is any collection of valid ECDL statements. Note that ECDL is a functional language that has no concept of variables.

fact store A component of the ECS Engine which stores relationships between objects. Any two objects which may be any ECDL data type, 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.

FLEXlm A Licensing technology used in stand-alone and DM-integrated ECS products.

floating license A license where there is a single license server for all licensing clients on the network. Any licensing client on the network can access the license server to check out a license.

function A general term for a portion of a program that performs a specific task. In ECS, an ECDL function is one of the built-in functions or operators, or a user defined function. ECDL functions can be named or anonymous, but must return an ECDL value.

GDMO See Guidelines for the Definition of Managed Objects (GDMO).

Greenwich Mean Time Standard time used throughout the world based on the mean solar time of the meridian of Greenwich. See Universal Coordinated Time (UTC).

Guidelines for the Definition of Managed Objects (GDMO) Describes a formal method for describing the important characteristics and operations of an object class. Specified in ISO 10165-4, X.722.

HP OpenView A family of network and system management products, and an architecture for those products. HP OpenView includes development environments and a wide variety of management applications.

identifier A name that within a given scope uniquely identifies the object with which it is associated.

IEC International Electrotechnical Commission.

IEEE Institute of Electronic and Electrical Engineers.

International Telecommunications Union, Telecommunications (ITU-T) The ITU is a world-wide organization within which governments and industry coordinate the establishment and operation of telecommunications networks and services. It is responsible for the regulation, standardization, coordination and development of international telecommunications as well as the harmonization of national policies. The ITU is an agency of the United Nations. In 1992 it took over the functions of the CCITT.

ISO International Standards Organization.

OVO *See See IT/Operations (OVO).*

OpenView Operations (OVO) HP OpenView Operations, a distributed client/server software solution that helps system administrators detect, solve, and prevent problems occurring in networks, systems, and applications.

ITU-T International Telecommunications Union, Telecommunications.

key-value pair A data storage item consisting of a search key paired with a value. In ECDL, a key-value pair is written as “key => value”. See also dictionary.

library In ECS, a repository for compound nodes. Compound nodes in the library may be referenced from a circuit, or copied from the library and modified.

license The legal right to use a feature in a software program.

license server The server processes that manage access to ECS features by licensed users.

list data type a variable-length ordered set of values all of the same data type. In ECDL, a list data type may contain a set of values of any other ECDL data type including complex types such as lists and tuples.

Management Information Base (MIB) A logical collection of configuration and status values that can be accessed via a network management protocol.

MDL *See See Message Description Language.*

message description Detailed information about an event or message. In ECS, a description of the attributes and formatting of a text-based event (message), that allows the MDL endecode module to decode and encode events consistent with that syntax. Message descriptions which are written in Message Description Language (MDL) are translated into metadata before being used by the ECS engine endecode module. See metadata.

Message Description Language A language used to describe a text event’s attributes and formatting. Each text event syntax has its own message definition written in MDL. See also message definition; event syntax.

metadata Data about data. In ECS, message descriptions are translated into metadata which is a form which maximizes access performance by the MDL endecode module. See message description. CMIP and SNMP metadata is derived from MIBs.

MIB Management Information Base (MIB).

Network Node Manager (NNM)

Definition to come from OVSD.

NNM *See See Network Node Manager (NNM).*

node 1. A computer system or device (e.g., a printer, router, bridge) in a network. 2. A graphical element in a drawing that acts as a junction or connection point for other graphical elements. 3. In ECS, see correlation node.

nodelock license A license where the license server and license clients must be on the same machine, meaning that the licensed application is "locked" to running on the node that is the license server.

object identifier (OID) A unique sequence of numbers or string of characters used for specifying the identity of an object, that is obtained from an authorized registration authority or an algorithm designed to generate universally unique values.

OID *See See object identifier (IOD).*

oid data type In ECS, an oid is an ECDL data type which contains an Object Identifier in dot-separated notation (e.g., 1.2.3.4.5). Where the data item is dynamically interpreted, at least three elements (2 dots) are required to avoid interpretation as a real data type.

Open Systems Interconnection (OSI) A standardization model in which a manager process is responsible for executing specific management functions requested by the user through interactions with an agent process. The agent process represents the management services offered by the managed objects.

OSI *See See Open Systems Interconnection (OSI).*

parameter *See In ECS, see correlation node parameter.*

pmd HP OpenView postmaster daemon.

port 1. A location for passing information into and out of a network device. 2. In ECS, a location for passing events into and out of a correlation node or a correlation circuit. See correlation node port; correlation circuit port.

primitive event An ECS internal event which encapsulates an input event. Several header attributes are added as a header for correlation and control purposes, which are stripped before the primitive event leaves the ECS engine. See also event; temporary event; composite event.

reserved word Words that have special meaning in ECS and cannot be used for any other identifier.

Simple Network Management Protocol (SNMP) The ARPA network management protocol running above TCP/IP used to communicate network management information between a manager and an agent. SNMPv2 has extended functionality over the original protocol.

simulate *See See simulation.*

simulation In general, the imitation by a program of a process or set of conditions affecting one or more objects such that the results of the program reflect the impact of the process or changes in conditions. In ECS, a simulation is the process of feeding events from an event log file through the correlation circuit to observe the behavior of the correlation circuit using aids such as breakpoints, tracing, and stepping.

SNMP *See See Simple Network Management Protocol (SNMP).*

SNMP trap An unconfirmed event, generated by an SNMP agent in response to some internal state change or fault condition, which conforms to the protocol specified in RFC-1155. See event.

socket stack An interface that supports interprocess communication based on the use of file handles. In ECS a socket stack is used to communicate with the ECS Engine for command, i/o and annotation purposes.

Software Distributor (SD) HP OpenView multi-platform software installation product.

static parameters In general, parameters whose values are determined prior to program execution. In ECS, a statically evaluated parameter is a correlation node parameter where the value is defined when the correlation circuit is loaded. The value does not change when an event enters the associated node/port. See dynamic parameters.

syntax In general, the rules governing the structure and content of a language or the description of an object. In ECS, see event syntax.

Telecommunications Management Network (TMN) The term used to identify a homogeneous approach to the management of heterogeneous networks. It is defined in the international standards referred to as ITU-TSS M3100. TMN recommendations incorporate OSI NM concepts, principles, protocols and application services.

temporary event In ECS, an event that is created transparently by particular correlation nodes, and which may exist only within a correlation circuit. Temporary events may consist only of header attributes created by the correlation engine, or they may additionally contain user data. Temporary events cannot be transmitted outside the correlation engine. See also event; primitive event; composite event.

time data type An ECDL data type that includes time and date.

TL1 Transaction Language One was developed by Bellcore and is a management system protocol that uses structured text messages to pass information about networks and network element states.

TMN See Telecommunications Management Network (TMN).

transit delay The difference between an event's arrival time and its creation time. Transit delays can be caused by external network delays or by deliberately introduced delays in an ECS circuit.

trap *See See SNMP trap; event.*

tuple data type An ECDL data type. A data structure consisting of a fixed collection of elements, where each element is a simple ECDL type or a complex ECDL data type.

Universal Coordinated Time (UTC)

Standard time used throughout the world based on the mean solar time of the meridian of Greenwich. Formerly known as Greenwich Mean Time (GMT).

universal pathname A set of environment variables that describe standard pathnames. Universal pathnames hide variations between pathnames on different versions of Unix.

UTC *See See Universal Coordinated Time (UTC).*

X/Open Management Protocol (XMP) An API specified by the X/Open standards body that provides a common access mechanism to both CMIS and SNMP management protocol services.

XMP *See See X/Open Management Protocol (XMP).*

Zulu *See See Universal Coordinated Time (UTC).*

Index