HP OpenView
Event Correlation Services
Designer's Guide

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

Manufacturing Part Number: J1095-90313

January 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.

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

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

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

Contents

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 ECS Designer's Guide contains the information you require to efficiently use the ECS Designer. This guide contains information on:

- Understanding the components of the HP OV ECS Designer.
- Developing event correlation circuits to load into the HP OV ECS Engine.
- Debugging the correlation circuits you develop, using the ECS Designer in Simulate mode.
- Interfacing your circuits to your network.

Audience

This manual is written for network personnel who design and build event correlation circuits. Readers of this document are assumed to have the following background:

- A general business understanding of the network. In particular, an understanding of the event types generated within the network.
- Substantial knowledge of network management and administration. This includes an understanding of the problems which typically occur within their network and the events generated when these problems occur.
- Scripting language knowledge, or some knowledge of flowcharting and programing concepts.

OVO

Where a paragraph relates to a specific topic, the topic is identified in the left margin. For example, information specific to ECS for HP OpenView Operations is identified with OVO in the left margin

Audience

2 HP OpenView ECS Overview

In this chapter

This section provides an overview of HP OpenView Event Correlation Services (ECS). It describes each ECS component, and explains how ECS can help you diagnose network faults. It contains:

What is an Event?

An event is a special kind of message generated by a device in your network. The event indicates a change of state of the network device. An event has the following attributes:

- It is transitory
- It is either transformed or is lost
- It is stateless
- It has no name and therefore it can not be referenced.

In a managed network, incidents such as component failures, congestion, or a network element reverting to a backup system may cause a change of state that generates an event.

Event Protocols

ECS handles four types of event protocols, SNMP, CMIP, ASCII and OpC events.

NOTE

SNMP events are called traps, CMIP events are called notifications, and OVO events are called messages. Within ECS they are encapsulated in a common format, and called events.

Events may contain information relating to the time they were generated. For example, a CMIP event may have a creation timestamp generated by the system clock at its origin. An event may also contain information about its origin, type and change of state. The type of information an event contains depends on the standards with which it complies, i.e. SNMP, CMIP, ASCII or OpC protocols, its encoding type (`ber` or `mdl`), its syntax, and the event type.

About Event Storms

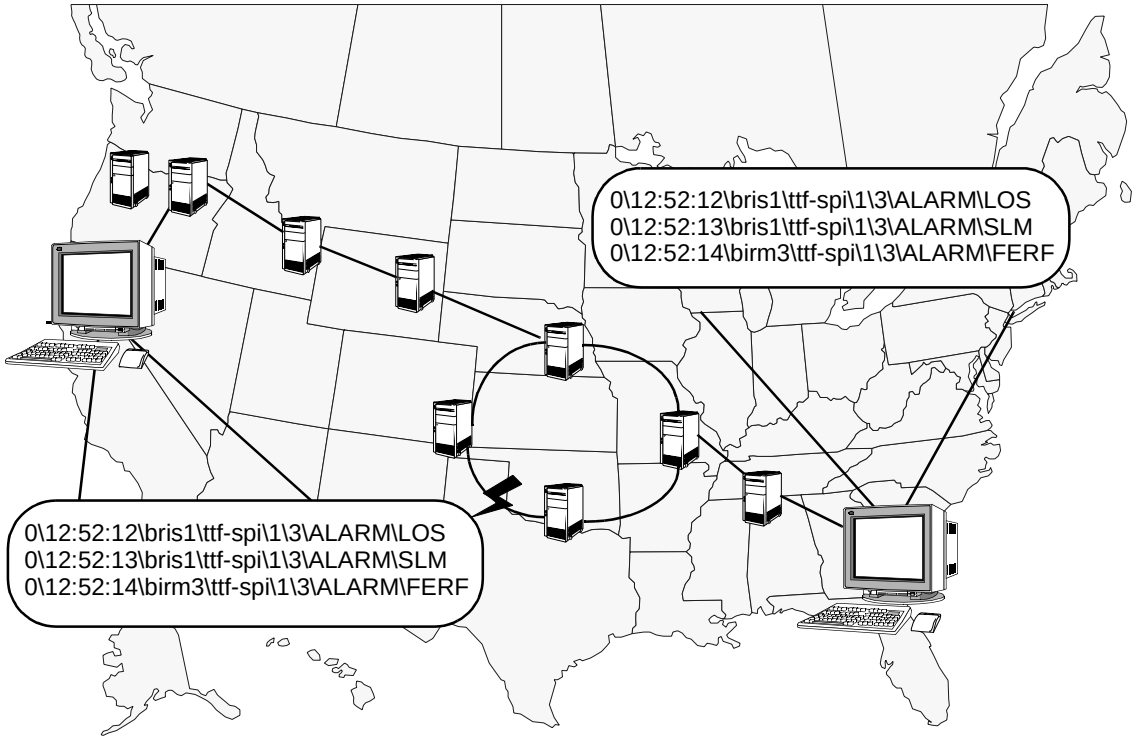
One of the main problems network operators and administrators face today is the large number of events (called event storms) that a network can generate following a change of state. These event storms make it

difficult to quickly and accurately identify the cause of the problem and take appropriate action.

Take the scenario where a major optical fiber cable is damaged by construction work. The *ripple-through* effect on each network element, dependent on the carrier, causes each network element to generate an event indicating its change of state. This can cause an event storm of several thousand events, milliseconds after the cable is damaged. Examples of events generated in an optical fiber cut include:

- Events from the optical fiber cut itself. This can result in hundreds of thousands of events per second for an OC-192 fiber
- LOS (Loss of Signal) events generated by ADM's (Add-Drop Multiplexors) directly affected
- LOF (Loss of Frame) events
- FERF (Far End Receive Failure) events
- TIM (Trace Identifier Mismatch) events
- EE (Excessive Errors) events.

Figure 2-1 Scenario of a Major Optical Fiber Cut



Network operators are not usually interested in the actual events. They want to know what the underlying fault was that caused the event and where the fault occurred.

How can ECS Help?

ECS can identify and highlight changes in the state of a network (by suppressing and correlating event storms), and then pass on or generate events which have more significance or a higher value.

ECS allows you to develop applications to effectively manage event storms in diverse, high speed telecommunication or IT environments.

ECS is designed to handle the high rate of events generated from the latest network technologies. It acts as a toolkit to simplify the

integration of event correlation into management applications such as:

- Faults/Alarms
- Performance
- Configuration
- Security
- Accounting/Billing.

NOTE Although ECS can correlate events at very high rates, the network itself determines the rate at which ECS receives events for processing.

Correlating Events in ECS

Event correlation is the process by which you can identify relationships between events. Once identified, this process can produce a smaller number of new events with the same or higher information content. This assists in the task of diagnosing network faults.

ECS implements a new paradigm for correlation systems:

- Each correlation rule is implemented by a discrete decision-making element called a node.
- The behavior of a node depends on the node type, the values you assigned to the node parameters, and the logical rule or control expression you configured into the node.
- Each node has one or more input ports and one or more output ports.
- A node receives events through the input ports, and if appropriate, sends events through the output ports.
- To create a correlation circuit, you link the nodes together by connecting node output ports to the input ports of nodes further along the path event flow. Linking nodes creates complex combinations of correlation rules. The circuit provides a visual representation of correlation rules of any complexity.
- The connection policy is very flexible. You can connect a node output port to many input ports of other nodes. Similarly, you can connect the output ports of many nodes to the input ports of a single node.

- Events flow through the correlation circuit, from Source nodes to Sink nodes.
- Events can follow multiple paths through a circuit, depending on how you connected and defined the nodes within the circuit.
- At most, only one instance of an event is transmitted from a circuit.
- Events that are not suppressed by a circuit are either output from the circuit or discarded by the circuit, depending on the policy for which the circuit is designed.
- Groups of interconnected nodes may be combined to produce reusable Compound nodes with creator-defined event processing characteristics. This allows complex correlation rules to be encapsulated as a new node type — a named Compound node.

About ECS

ECS is based upon the **EventFlow** technology invented by HP's Bristol Laboratories (UK). The fundamental design principle is to correlate events in real-time, even when events arrive out of creation time order.

The need for a high-speed correlation technology has become apparent with the advent of new high-speed telecommunications and IT technologies. The implementation of these new technologies is expected to accelerate as bandwidth requirements accelerate.

ECS Components

ECS comprises two main components: the ECS Designer and the ECS Engine. The ECS Designer is used to build correlation circuits. The ECS Engine is used to run one or more correlation circuits.

The pages to follow provide an overview of these components.

The ECS Designer

You use the ECS Designer to visually develop correlation circuits. This gives a greater level of productivity and efficiency over *rule-based* systems as you can visualize the flow of events through the circuit.

The ECS Designer works in two modes:

- Build Mode — where you build and modify a correlation circuit
- Simulate Mode — where you simulate and test a correlation circuit.

NOTE The ECS Designer can be used to build and test only a single correlation circuit at a time.

The current mode affects how the ECS Designer works. Each mode enables different menu items, tool bars, and actions, as appropriate for that mode.

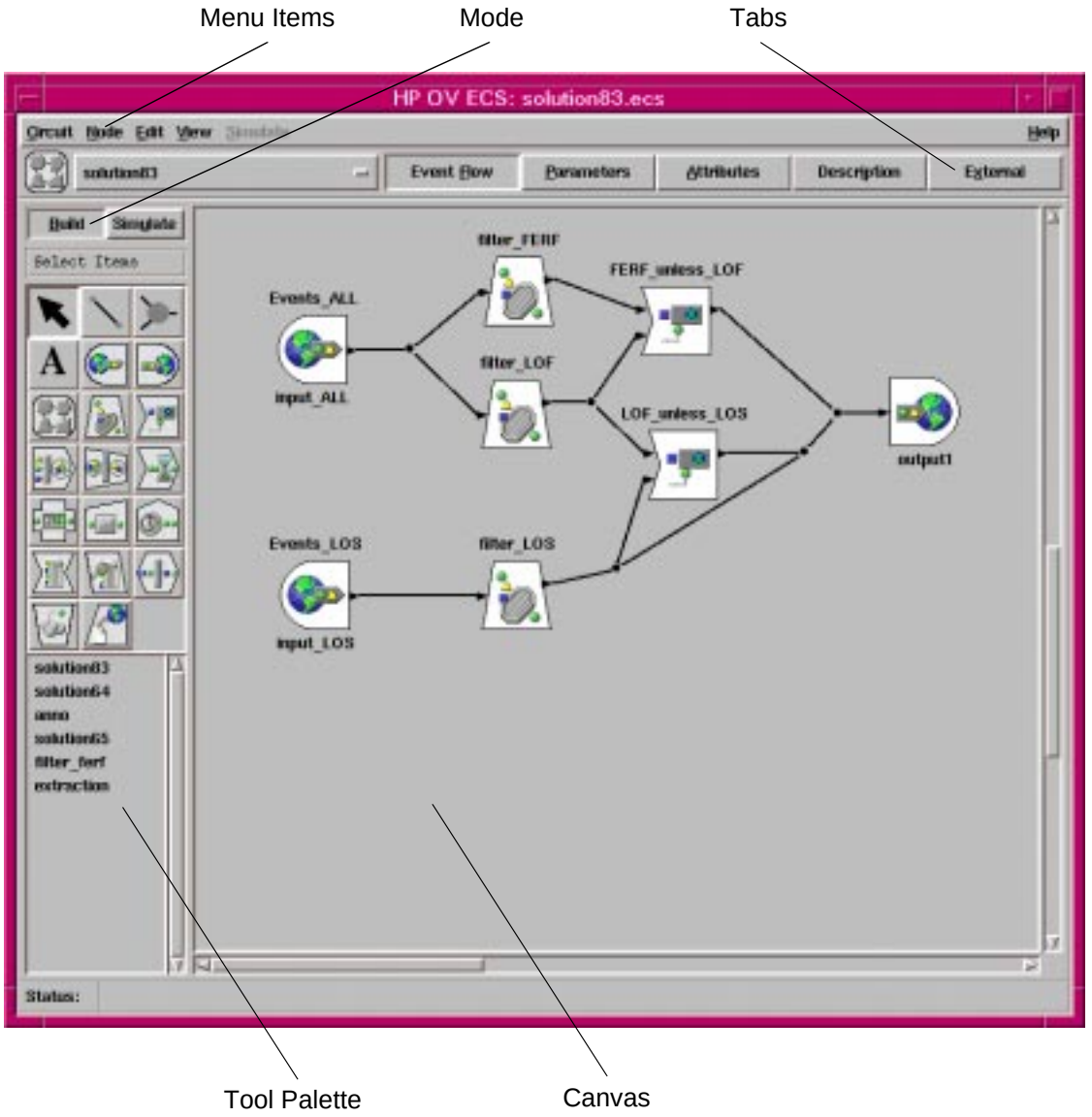
The ECS Designer always starts in Build mode. You can create and verify a new circuit (or load a non-verified circuit), then switch to Simulate mode to test the circuit.

To switch between Build and Simulate modes, select the [Build] or [Simulate] button, as appropriate (see Figure 2-2 and Figure 2-3).

NOTE ECS verifies the circuit when you select the [Simulate] button. If the circuit fails to verify, you cannot switch to Simulate mode.

The ECS Designer in Build Mode

Figure 2-2 The ECS Designer (Build Mode)



The ECS Designer uses a circuit design paradigm, similar to an electronic circuit, to simplify the task of developing event correlation. In Build Mode, the ECS Designer provides a selection of *nodes* that you can *click and place with the mouse* to build a circuit for processing incoming events. These nodes perform specific functions, such as filtering or modifying events within the circuit. See Chapter 4, “Building a Circuit,” on page 63 for more information about each node.

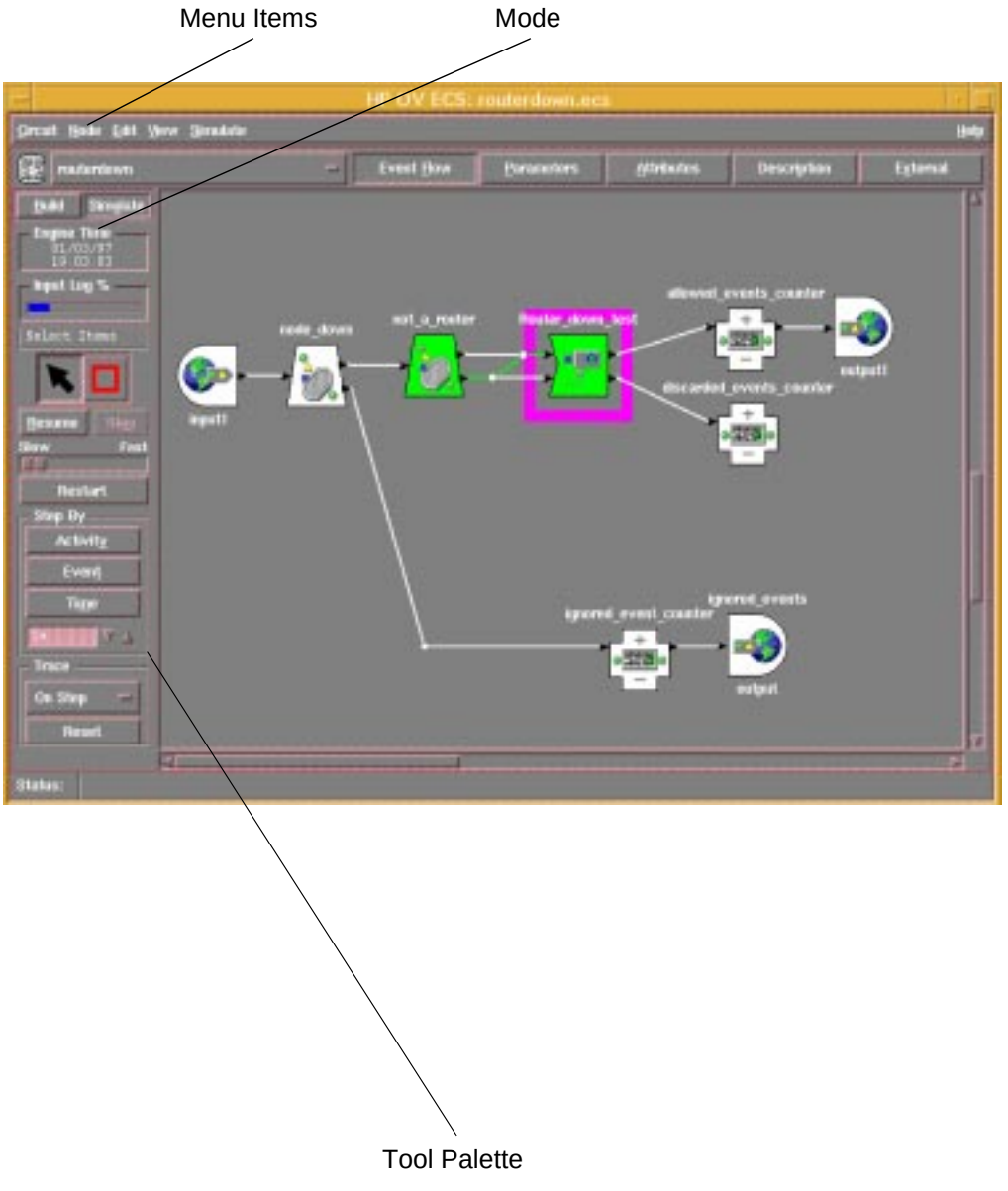
You can group nodes together to create Compound nodes. You can then use these Compound nodes and primitive nodes as building blocks to create sophisticated correlation circuits. These circuits can perform complex and high speed event correlation functions.

You can rapidly build up a library of reusable Compound nodes — reducing the time and cost of building new correlation circuits.

When you have built a correlation circuit, you must compile it. The ECS Designer generates a compiled circuit file that can then be loaded into the ECS Engine.

The ECS Designer in Simulate Mode

Figure 2-3 The ECS Designer (Simulate Mode)



It is essential that you verify and test a circuit using captured real events before you deploy it to the ECS Engine.

In Simulate Mode, you test a circuit by feeding it a set of events that are representative of the events the circuit is designed to correlate. Usually these events are captured (in an event log) from the network where the circuit is to be used.

The entire set of events can be allowed to run freely through the circuit or you can monitor the progress of particular (marked) events. In the latter case, the set of events may be allowed to free-run until a marked event is reached. The progress of the marked event can be visually monitored by single-stepping it from node to node through the circuit and then the set can be allowed to free-run again until another marked event is reached.

Simulation is performed by an ECS Engine built into the ECS Designer. This allows the ECS Designer to maintain control over the engine's notion of time.

You can dynamically adjust the engine's clock speed during a simulation. Slowing down the clock speed allows you to observe the progress of events from node to node.

You can pause the simulation at any point during event input to:

- Examine the state of any node.
- View the list of events processed by a node.

You can also:

- Inspect the contents of Data and Fact Stores.
- Check the values of circuit and event attributes.
- Examine events.

Different colors and visual cues indicate the progress of an event through the circuit and the different types of responses from a node. This allows you to see whether a node parameter evaluated true or false, whether an error occurred, and which path was taken (where alternative output ports are available).

The ECS Engine

The ECS Engine performs specific correlation actions on an event stream, resulting in a stream of correlated events. The engine can

reference information externally in the form of Fact and Data Stores. The ECS Engine is available as a standalone component, linked to HP OpenView DM, as part of HP OpenView Operations, and integrated with HP OpenView NNM.

NNM

For HP OpenView Network Node Manager, the ECS Engine is linked to the standard NNM stack. It does not appear as a standalone component. The engine is capable of processing SNMP traps and ASCII events, is not capable of processing OVO or CMIP events.

OVO

For OVO, the ECS Engine is provided as a library that is linked to the OVO runtime processes. It does not appear as a standalone component. This engine is capable of processing OVO messages only, and is not capable of processing ASCII, SNMP or CMIP events.

About Fact and Data Stores

The Fact and Data Stores are local caches of network information and tuning parameters that a correlation circuit can use to define and control its behavior. Once loaded these stores become memory resident.

You can update both the Fact Store and the Data Store while the engine is running.

The Fact Store

The Fact Store is one of two stores that enable a circuit to be more flexible and data driven.

The Fact Store *stores* relationships between objects. These relationships enable the events to be related back to the physical world.

For example, if card `x` is contained in equipment `y`, then for the purposes of event correlation, events from source `x` may be considered to originate from equipment `y`.

You can design the circuit to define a relationship such as `is_contained_in`, and then insert this into the Fact Store. For example:

```
card x is_contained_in equipment y
```

The Data Store

The ECS Engine uses the Data Store, like the Fact Store, to evaluate circuit parameters that refer to Data Store entries. You can use the Data Store to parameterize the circuit.

The Data Store contains a table of global values. Unique keys identify each table entry and you use these keys to select a value from the table.

The administrator can modify the Data Store and load it into an ECS Engine. The Data Store takes effect when you reset the circuit.

For example, a duration parameter on a table node specifies how long (in seconds) events remain in the current region of the table. You can specify the value of the parameter as:

```
datastore (my_duration)
```

This expression uses the value of the Data Store entry `my_duration`.

Statically evaluated parameters will pick up values from the Data Store at circuit load time and dynamically evaluated parameters will pick up current values as required.

Relationship to HP OpenView

ECS provides network event correlation over heterogeneous networks.

The ECS Engine can receive events in the following ways:

- Through the DM postmaster daemon (`pmd`) to the ECS Engine. Only SNMP and CMIP events are received this way. HP OpenView ECS integrates as a service stack within the `pmd`. The HP OpenView DM `pmd` can be configured to route the events into ECS using the `pmd` local registration file (LRF). ECS provides a configuration and management command line utility, ECS Manager (`ecsmgr`), for the administration of the ECS Engine.
- Through a socket interface to a standalone ECS Engine. ASCII and SNMP events can be received this way. The customer is responsible for interfacing with the socket interface through the supplied ECS EIO API.
- From the NNM postmaster daemon to the ECS Engine. Only SNMP events can be received this way. However, ASCII events can also be received from the socket interface, if implemented.
- Through the OVO stack. Only OVO messages can be received through this interface.

Once inside the ECS Engine, events are assigned to a specific stream. Each stream is a logically separate flow of events which is created by the ECS Engine. Correlation circuits are enabled on one or more of these streams. If a particular stream is not specified by the input source then events are applied to the default stream.

3 Approach to Circuit Design

In this chapter

This chapter provides a structured approach to designing event correlation circuits and contains:

This chapter describes the following terms and concepts in detail:

- Correlation Circuits
- Compound nodes
- Primitive, Composite, and Temporary events
- Timing considerations.

Take time to read and understand these terms and concepts as they appear throughout the procedures described later in this guide.

Overview of Correlation Circuits

A correlation circuit is a collection of nodes that is configured to perform event filtering and correlation. In addition to the collection of nodes, a circuit contains configuration parameters that define the event types and transit delays the circuit accepts from the external event source.

Nodes

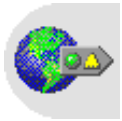
The nodes you use to develop correlation circuits appear on the ECS Designer tool palette. These nodes can be grouped under five types:

Type	Description
External Connection Nodes	Source and Sink Nodes These nodes connect the circuit to the outside word.
Event Traffic Cop Nodes	Filter, Unless, Combine and Rearrange Nodes These nodes control the flow and structure of primitive events.
Time Control Nodes	Clock, Count, Rate and Delay Nodes These nodes provide time control facilities.
Event Manipulation Nodes	Table, Extract, Modify, Create and Annotate Nodes These nodes manipulate the information contained in each event.
Compound Nodes	A singular node that represents an entire circuit comprised of other nodes. A circuit can be designed so that it can be saved (encapsulated) as a compound node and used as a single node in another circuit.

The different node types are described below. Refer to the *HP OpenView Event Correlation Services Designer's Reference* for a full description of each node.

External Connection Nodes

Source Node

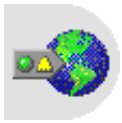


The Source node is the point of entry into a circuit or Compound node for events from the external event stream.

You can have more than one Source node in a circuit or Compound node.

Each Source node in a circuit corresponds to an external input port for that circuit.

Sink Node



The Sink node simply forwards an event from the circuit or Compound node to the appropriate external event stream. The events must be primitive.

You can have more than one Sink node in a circuit.

Each Sink node in a circuit corresponds to an external output port for that circuit.

Event Traffic Cop Nodes

Filter Node

The Filter node passes events that satisfy a user defined condition. When the Filter node receives an event, it tests the event against the condition. The event is passed on only if it meets the requirements specified in the condition.

Unless Node

The Unless node passes an event unless an inhibiting event occurs within a specified time window.

This node stores events arriving at the input port, then outputs the events when there has been no inhibiting event within the specified duration.

Combine Node

The Combine node lets you group events and deal with them as a single entity. It is typically used to group events that have been created almost simultaneously, into a single composite event.

Rearrange Node

The Rearrange node receives a composite event and transmits a new event constructed from the events contained in the original composite event.

Time Control Nodes

Clock Node

After it is started, the Clock node generates temporary events at specified intervals counted from the start time. The temporary event is empty and does not contain a body. These events can be used to trigger other activities in the correlation circuit, and cannot be transmitted from the circuit.

Count Node

The Count node maintains a count of events passing through the node and makes this count available to other nodes as an attribute.

Counts can be maintained for the total number of events flowing through the node, as well as for sub-categories of events.

Rate Node

The Rate node measures the number of events per second passing through the node over a defined window of time and makes this value available to other nodes as an attribute.

Rates can be maintained for the total number of events flowing through the node, as well as for sub-categories of events.

Delay Node

The Delay node detains each incoming event until the difference between its creation time and the current time exceeds the specified amount.

This is an effective way of sorting events into creation time order, when they have arrived out of order due to network delays.

Event Manipulation Nodes

Table Node

The Table node stores a history of extracted attributes of events or incoming events and provides an interface that enables other nodes to query it. Incoming events are immediately copied to the output port.

The Table node stores extracted atributes of events or whole events in event creation time order (not arrival time order). This sequence of events can then be queried by other nodes to determine the recent history of event order. For example, when a certain number of consecutive events indicate an unacceptable error rate.

Extract Node

The Extract node creates and outputs composite events containing the received event and one or more other events copied from one or more Table nodes. A parameter specifies which events the node copies.

Modify Node

The Modify node allows event attributes to be added, deleted, or changed.

Create Node

The Create node creates a new primitive event for each event that arrives at its input port.

Annotate Node

The Annotate node delays events while it queries an external annotate process. If a response is received within the specified time, a composite event is transmitted containing the original event together with the data retrieved from the external annotate process.

NOTE Annotation requires the presence of an external annotation process. The construction of an annotation process requires interfacing with the ECS ANNO API, described in *HP OpenView ECS Developer's Guide and Reference*.

Compound Nodes

A Compound node is a correlation circuit encapsulated into a reusable object. Once encapsulated, you use this Compound node like any other node in a correlation circuit. Encapsulation provides the following benefits:

- Facilitates modular **programming** of event correlation circuits. This approach makes debugging and maintenance easier.
- Permits **information containment**, so you can work at a higher level of abstraction using a Compound node. You can deal with the services offered by the node rather than the detail of the correlation circuit the node contains.
- Allows you to generate a library of Compound nodes to resolve common problems.

A circuit can include any number of Compound nodes.

The major difference between a circuit and a Compound node is that a circuit is completely defined and you can load it into an ECS Engine. In other words, a circuit is a Compound node with a defined external interface.

When you retrieve an existing Compound node from the library, ECS reads it as a referenced object. The Compound node may have parameters and attributes, that you can define or reference. However, you cannot edit the contents of a referenced Compound node. See Chapter 6, “Designing Large Circuits,” on page 155 for details about editing Compound nodes retrieved from the library.

Primitive, Composite, and Temporary Events

Chapter 2, “HP OpenView ECS Overview,” on page 17 discusses the type of events ECS receives from the external environment. ECS decodes these events for internal use and can also generate events internally, or combine events for internal purposes. These events fall into three categories:

- Primitive events
- Temporary events
- Composite events.

Primitive Events

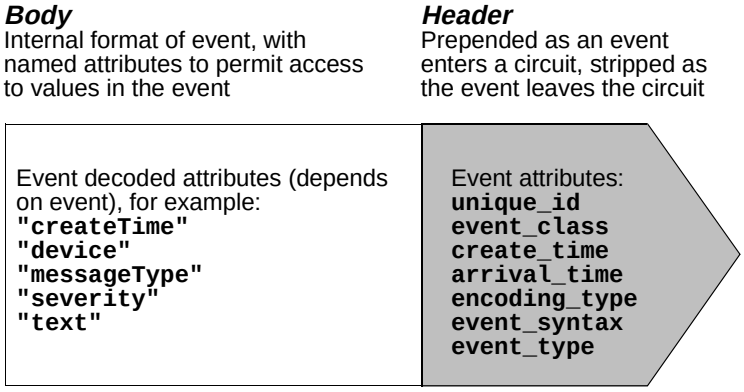
Primitive events are the ECS internal representations of events while they are inside a circuit.

When the ECS Engine receives an event into a circuit, it decodes the events from their native format to an internal format. When the ECS Engine transmits events out of a circuit, it encodes the internal format back into the native format.

The Create and Modify nodes create primitive events in internal format that are encoded into the native format as they are transmitted from a circuit.

A primitive event consists of a header and a body—see Figure 3-1. The header contains attributes required for internal housekeeping as the event travels through a circuit. The body consists of the actual event in its internal format.

Figure 3-1 Structure of a Primitive Event



You access the header attributes of a primitive event by token name (all header names are token data types, so must not be enclosed in quotation marks in expressions). For more information about data types, see Chapter 7, "Data Types" in the *HP OpenView Event Correlation Services Designer's Reference.*

You can test and read header attributes but you cannot modify them.

You access the body attributes of a primitive event by string name. The names of body attributes are determined by the event encoding type:

ber The names of the body attributes are derived from the MIBs of the agents that originated the events. Only SNMP and CMIP events are ber encoded.

mdl The names are determined by the mdl definition. See *HP OpenView Event Correlation Services ASCII Module* for details.

OpC_Msg The names of attributes are fixed. See *OVO Messages in ECS* in the *HP OpenView Event Correlation Services Designer's Reference* for details.

Temporary Events

HP OpenView ECS generates temporary events for use within the ECS Engine. These events exist only within the engine and cannot be transmitted out of the engine. Temporary events may carry data obtained from external processes or simply trigger other nodes into action. For example, the Clock node emits a temporary event that contains no body attributes.

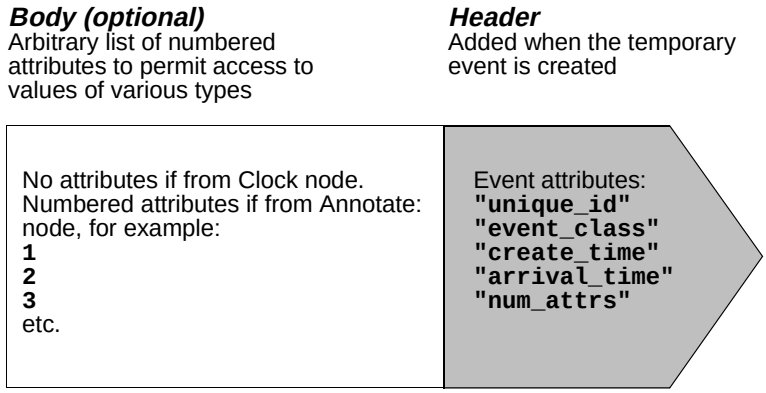
Temporary events are created by:

- An Annotate node when it receives a response event from a manager external to a circuit. The temporary event is enclosed in the composite event generated by a successful annotation.
- A Clock node at regular intervals.

A temporary event consists of a header and an optional body—see Figure 3-2. The header contains attributes required for internal housekeeping as the event travels through a circuit.

The body of a temporary event created by an Annotate node contains numbered attributes that return information obtained from an external source. The body of a temporary event created by a Clock node is empty (i.e. contains no attributes).

Figure 3-2 Structure of a Temporary Event



You access the header attributes of a temporary event by token name. You can test and read header attributes but you cannot modify them.

You access the body attributes of temporary events created by an Annotate Node by a number. Events generated by a Clock node do not have a body.

Refer to the *HP OpenView Event Correlation Services Designer's Reference* for a more detailed description of temporary events.

Composite Events

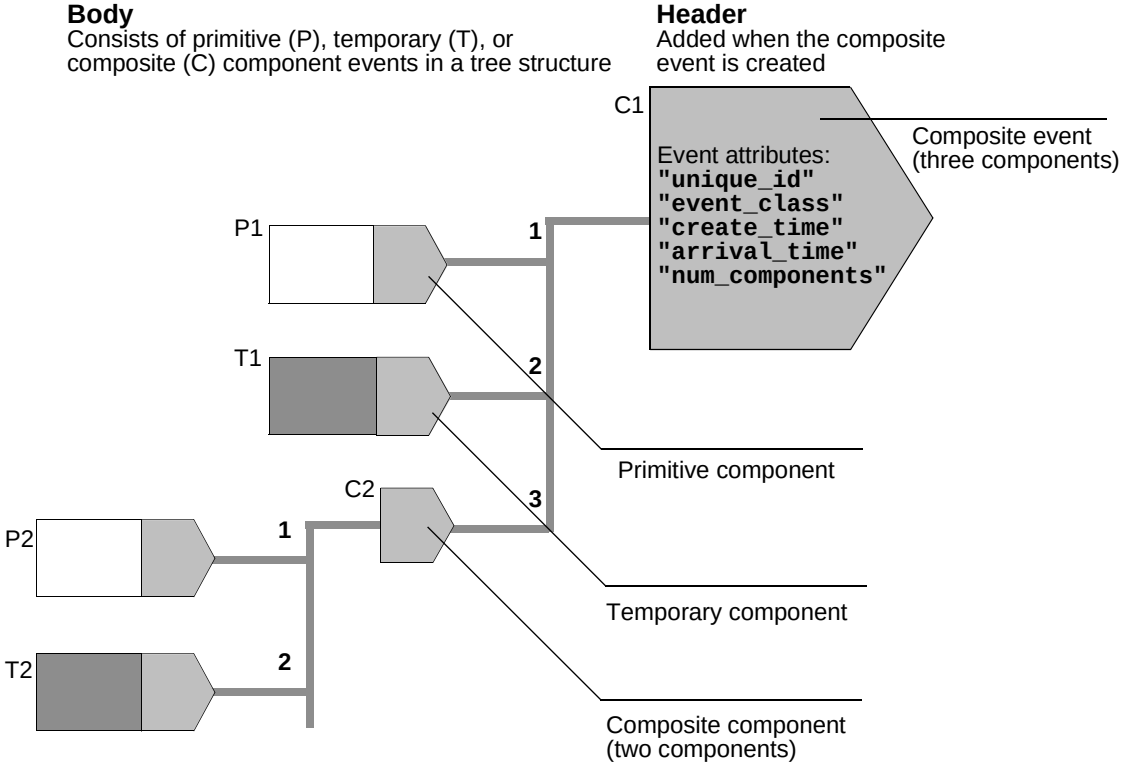
HP OpenView ECS generates composite events for use only within a circuit. They are a combination of primitive, temporary, or composite events packaged together so that you can treat them as a single event.

You use composite events to avoid the overhead of dealing with related events separately. For example, you might package several events together so a Create node can extract information from different attributes of several events to include in a single new event.

The Annotate, Combine, and Extract nodes create composite events as their normal output. The Unless node creates composite events only as an error output. Normally, you use the Rearrange node to extract a primitive or temporary event out of a composite event, but it is also capable of creating a new composite event out of components of a received composite event.

A composite event contains a header to which the component events (in a tree structure that makes up the *body*) are attached—see Figure 3-3. The header contains attributes required for internal housekeeping as the composite event travels through a circuit.

Figure 3-3 Structure of a Composite Event



You access the header attributes of a composite event by token name. You can test and read header attributes, but you cannot alter them.

Refer to the *HP OpenView Event Correlation Services Designer's Reference* for a more detailed description of composite events.

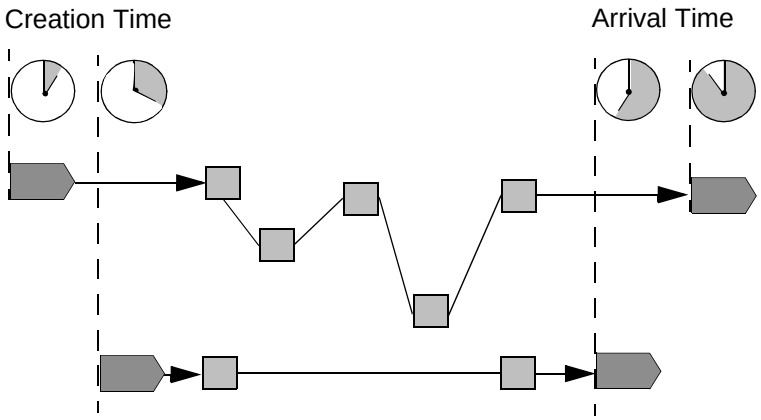
Components in a composite event are addressed by the numbered path (bold numerals in Figure 3-3). For example, to access the unique id of event T2, enter the node parameter as **`input_event 3 2 unique_id`**.

Timing Considerations

Transit delays make it difficult to use the individual times at which events arrive at the ECS Engine to determine which of two events occurred first.

Therefore, where possible, ECS uses the event creation time for event ordering.

Figure 3-4 Transit Delays



Events may arrive at the ECS Engine in an order different to their creation order. This is due to network transit delays and the fact that events can come from different parts of the network.

Use a Delay node when nodes further down the path require the event arrival order to be the same as the event creation order. A Delay node restrains an event from continuing until the current engine time is a specified number of seconds after the time the event was created. This action effectively sorts events into the order of their creation times. Events that are already older than the specified delay period are allowed to continue without further delay.

In some correlation problems, an event may only be of interest for a particular length of time. With ECS you can specify the maximum and minimum transit delays for a specified input circuit port.

You must specify the transit delay for any circuit containing nodes with memory on the node ports (`Unless`, `Table`, `Delay` or `Combine` node). ECS

takes the transit delay into account when evaluating the time windows for these nodes.

CAUTION

If you do not define minimum and maximum transit delays for circuits containing `Unless`, `Table`, `Delay` or `Combine` nodes, you cannot simulate and test the circuit.

Designing Correlation Circuits

Design the correlation circuit in modules containing a small number of primitive nodes. This makes it easier to solve problems and debug, and ensures that the circuit is reusable further on.

To design a circuit:

1. Determine the position of ECS Engines within the network
2. Identify and understand the problem you are trying to resolve
3. Capture and analyze the event logs
4. Define the relationship between events
5. Identify the inputs and outputs of the circuit
6. Consider the stream policy
7. Partition the problem
    a. Define the structure of initial filtering
    b. Define the structure of the correlation
    c. Integrate filtering and correlation to produce output
8. Optimize the circuit.

Generally, it is best to design circuits that emulate the action of a competent network operator in recognizing event patterns, rather than trying to systematically identify all possible outcomes.

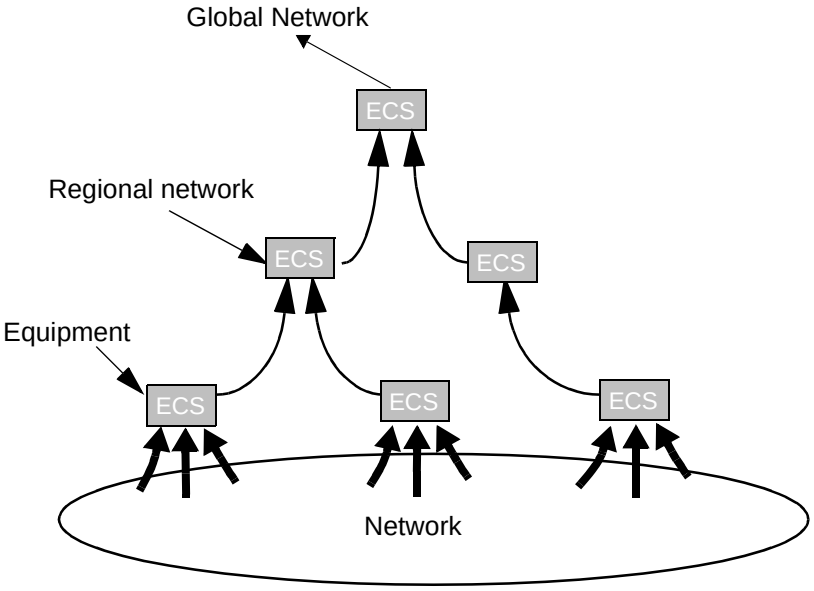
Determining the Position of ECS Engines

Fundamental to the design of HP OpenView ECS is the ability to deploy the ECS Engine as near to the source of events as possible. In an extreme environment, an ECS Engine is embedded into each piece of equipment that generates events.

Correlated events from each piece of equipment are forwarded to another ECS Engine that performs the correlation across multiple systems. This reduces the number of events as early as possible, avoiding the need (and associated cost) of management networks handling event storms. Many events may be reduced to a few events at each level of correlation.

The first step in defining event correlation for your environment is to determine where the ECS Engines need to be deployed within the network. The positioning of the ECS Engine determines the **view** the engine has of the network, the type of events to be processed, and the type of information to be derived from the output.

Figure 3-5 Levels of ECS Deployment



The circuit design for each ECS Engine depends on its position within the network.

In the scenario illustrated in Figure 3-5, ECS Engines are deployed at three levels in the network to optimize the filtering and correlation of events.

1. At the lowest level, the ECS Engines are deployed close to the equipment. At this level the engines can only *see* events from the equipment. The main purpose of the correlation circuit at this level is to filter the event storms that each network element (or group of elements) may generate. The circuit passes on uncleared and non-duplicated events indicating the alarm status, and suppresses duplicate/irrelevant and spurious events.

2. At the second level, the circuit passes the filtered events through to a regional network level. This may relate to a geographic region of a telecommunications network. Here the events are correlated to provide better information as to the state of equipment and their relationship within the network.
3. The third level is the highest level. The ECS Engine correlates events relating to the status of the complete network for that environment. These events can be fed through to an application such as a fault management platform.

Once you have determined the position of the ECS Engine within the network, design an appropriate correlation circuit for that engine.

Identifying and Understanding the Problem

The next step in designing a correlation circuit is to define the network problem you are trying to resolve. What are the problems you are experiencing and how will those problems manifest themselves in terms of:

- The events generated by equipment within the network.
- The window of time relating to the problem.
- Recurrent patterns of events that indicate a problem *signature*.
- The relationship between the events.
- The events that are generated as a consequence of the problem or are unrelated to the problem.

Define in detail the purpose and goal of the circuit you are designing. Consider the incoming events and what you expect to be output.

For example, a wave-guide compressor on a microwave repeater tower emits an event each time the compressor is powered up or powered down. No other event is emitted from the tower to indicate status.

The wave-guide always leaks to some degree, so a cycle of power-up and power-down events is normal as long as they both occur within an appropriate time frame.

The problem occurs when the compressor fails or the wave-guide leaks excessively and cannot be kept at the appropriate pressure. To develop a circuit to identify these problems, the circuit must:

- Identify if there is no power-down event within 30 minutes of a power-up event (compressor having difficulty pressurizing the wave-guide).
- Identify if the time span between power-down and power-up is less than 5 minutes (indicates wave-guide not holding the pressure).
- Identify if there is no power up event within a 12 hour period (compressor failed).

In summary, the circuit must identify wave-guide leakage or compressor failure.

Analyzing Event Logs

To identify the events you need to filter and correlate, capture a log of events generated at the time the problem occurred.

This event log can be an historical log captured by the network administrator when the problem has occurred in the past, or you can generate the log in a test environment.

You determine the events of interest from the event log. As part of this analysis, consider:

- What pattern of events is consistently associated with the problem signature?
- What are the event types?
- What are the transit delays?
- Are the timing relationships between the events important?
- Is the pattern of events always the same?

In the wave-guide example described earlier, the event log contains a pattern of events from the same device (the compressor). The log also contains a pattern of **matched couples** (power-on, power-off) indicating a time span between the couples of between 30 and 45 minutes. This

means that either the compressor is having difficulty in pressurizing the wave-guide or the compressor is operating outside of its parameters.

If the event log indicates a time span of less than five minutes between power-down and power-up events, then the wave-guide is leaking excessively.

NOTE

For some problems, not all events associated with the problem are necessary for correlation. The problem signature may contain redundant information.

For information about generating a log of events, see “Capturing and Analyzing an Event Log” on page 90.

OVO

The procedure for logging OVO events differs from other event types. See “Logging OVO Events” on page 91

Defining the Relationship between Events

Consider the relationships between the events of interest before determining the filtering or correlation requirements.

This allows you to determine the events you need to input to the nodes within the circuit, and to set up appropriate event paths between the nodes.

Identifying Circuit Inputs and Outputs

Define all inputs you expect into the circuit and the source of the input. Define all outputs you expect from the circuit.

For example, compressors in our wave-guide scenario emit events of a registered event type. This event type would be the initial input. The events contain other attributes that you can use to determine the device to which they relate and when they were generated.

In determining the events you want to output from the circuit, consider the following:

- What output event, or events, do you require?
- What would the information content of the output event be?

- How is the output event information content obtained?
- Which events should be suppressed?

Considering the Event Flow

Before designing a circuit, it is important to understand the effect you intend your circuit to have on the event flow.

In the wave-guide scenario, there are four categories of events that are of concern for the circuit:

- Power-up events (input)
- Power-down events (input)
- A warning event (created when excessive wave-guide leakage or compressor failure is detected) (output)
- All other events, unrelated to our correlation scenario

The ECS Engine, with assistance from your circuit, decides whether to output or to discard each of these events.

For each of these event categories, you first have to decide whether the engine should output the event when a correlation circuit is introduced. This decision should be based on the desired engine behavior before any circuit is loaded.

One way of handling events would be to always output the warning event when it occurs, as well as to output all *other* events which enter the circuit. The circuit would not affect the engine's decision to output these two categories of events.

A harder decision would be what to do with the Power-up and Power-down events. With the new warning event, the network operator may no longer care to see the individual Power-up and Power-down events. In this case, the circuit should discard the Power-up and Power-down events.

Alternatively, the circuit may want to preserve the event flow of the Power-up and Power-down events so that the ECS administrator can see the warning event in addition to the Power-up and Power-down events.

The section, “Considering the Stream Policy” on page 53 describes default engine behavior in detail.

Considering the Stream Policy

Each event stream in the ECS Engine has a configurable *policy* for handling events. The stream policy determines the default behavior in the following cases:

- Stream policy controls event handling *before* any circuits are loaded in this stream. Events can either be *discarded*, or allowed to pass through and be *output*.
- Stream policy also determines what happens to events that are not accepted by any of the circuits currently loaded in a stream. Again, the engine can `discard` or `output` these events.
- Stream policy determines which circuits can be loaded into that stream. Only circuits that are designed for that stream policy can be loaded. You cannot load a circuit designed for one policy into a stream with a different policy.

The two stream policies are:

- **output** (output an event, unless it is discarded by a circuit in this stream)

  This would be the expected behavior in a network and system management environment, where only some fault signatures are extracted from the event stream and replaced with fewer or possibly new events.

- **discard** (discard an event, unless it is output by a circuit in this stream)

  This behavior is less commonly required. However, it is appropriate for example, when analyzing security violations. In this case you want to discard all events, except when a violation is detected and the alarm system should be triggered.

The purpose of a circuit is to change the default stream behavior. In an `output` stream, circuits change the event flow by discarding specific events and by creating new events. In a `discard` stream, circuits change the event flow by outputting events and by creating new events.

Circuits must be designed for a specific policy. You cannot load a circuit designed for one policy into a stream with a different policy. In other words, all circuits in a given stream must have the same policy, and that policy is determined when the stream is created.

Because a circuit must be loaded in a stream of the appropriate policy, the circuit designer should work with the administrator to ensure that this happens.

Partitioning the Problem

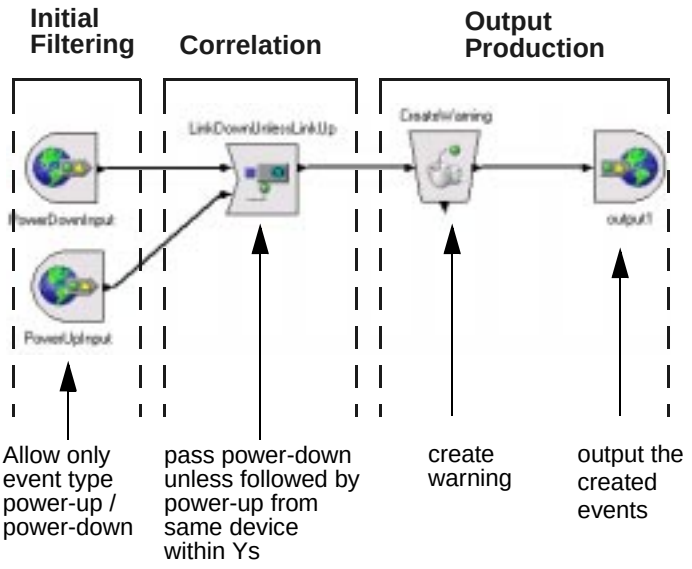
There are four distinct areas of an event correlation circuit that can be used to partition the problem:

- policy (`output` or `discard`)
- initial filtering
- correlation
- output production.

To understand examples used for each area of problem partitioning, it is important to first understand the example scenario described in the section, “Example of the Wave-guide Circuit”.

Example of the Wave-guide Circuit

Figure 3-6 Partitioning the Problem



The following steps outline the correlation circuit we will use to describe the problem partitioning approach in further sections of this chapter. The circuit accepts Power-down and Power-up events and discards transient power-loss notifications (Power-down event followed by a Power-up in a given time interval, originating from the same source). Other Power-down events cause the circuit to create a warning event.

1. **Policy**

   An *output* policy is selected to ensure that events that are not accepted into the circuit are output. In other words the circuit will affect this event stream by discarding selected events only.

2. **External Filtering**

   The Source nodes shown in the diagram select Power-up and Power-down events from the external event stream and pass them to the correlation stage.

3. **The Unless Node**

   The Unless node is fed Power-down events through the Input port, and Power-up events through the Inhibitor port.

   The Unless node is configured to detect Power-down events that are followed by corresponding Power-up events, from the same device within a given time period. These events are discarded by the Unless node. Power-up events never leave the Unless node, and are also discarded.

   The remaining Power-down events (those not followed by a Power-up) are then sent to the Create node.

4. **The Create Node**

   The Create node accepts the remaining Power-down events and creates a warning event for each one it receives. The warning event is sent to the Output node of the circuit. The original Power-down event is discarded.

5. **The Output Node**

   The warning event is transmitted from the circuit to the external event stream through the Output node.

Conclusion The only events that enter the correlation circuit are Power-up and Power-down events. The circuit discards every event that enters it. The only event that can ever be output is a warning event, created when a Power-down event is not followed by a Power-up event.

The following sections expand on this outline, and show examples of different scenarios depending on the selected *stream policy*.

Filtering

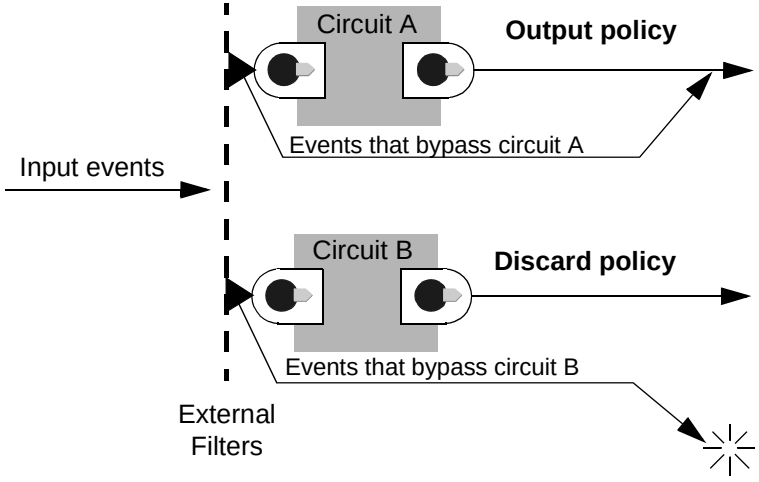
Filtering is the process of dividing events into two categories: those that pass the filter condition, and those that do not. There are two main types of filters:

- **External Filtering** is applied to the stream of events before it enters a circuit. Those events that pass the filter enter the circuit. The fate of events that do not pass is determined by the stream policy and the effect of other circuits.
- **Filter nodes** are used to divide events into two categories *within* a circuit. By default, a Filter node has just one output port, the True output port, that emits events that pass the filter condition. But Filter nodes also have a False output port which can be enabled if required. When a port is not connected, events emitted from that port are discarded.

Filtering is used to eliminate unwanted events from the event flow, and to detect specific event types. Although Filters are very flexible, in that the Condition expression can be arbitrarily complex, they have one important limitation: they have no ability to correlate events over time. Filters have no memory and process each event in exactly the same way, regardless of the events that come before or those that follow.

Filter events as early as possible in the circuit. Removing unnecessary events early in the correlation process reduces the work on the rest of the circuit and simplifies circuit design.

Figure 3-7 External Filtering and Stream Policy



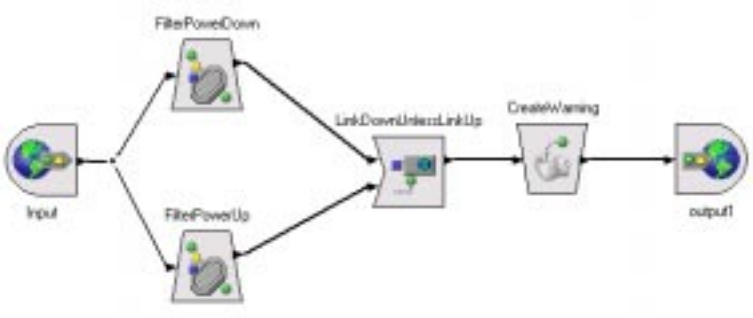
You can perform external filtering at a circuit input port (on the external side of the Source node) to select only appropriate events to enter the circuit. Events that are not selected do not enter the circuit and their fate is determined by the stream policy, or other circuit(s) that do accept these events.

It is important to consider what happens to the events that are rejected by the external filters, otherwise the circuit may have unintended effects on these events. For example, consider what happens to events that are neither Power-up nor Power-down events:

- An output policy would output these events.
- A discard policy would discard them.

If, in addition to Power-up and Power-down events, there are other events such as Power-fail events, then you need to decide what to do with them. If you want to pass Power-fail events on then you must implement the circuit for an output policy stream. If you want to suppress Power-fail events then you could implement the circuit for a discard policy. However, a discard policy will discard *all* events that do not enter a circuit. To selectively discard Power-fail events you need to redesign the circuit so that Power-fail events are selected by the external filter and then discarded inside the circuit. The next example shows one way to accomplish this.

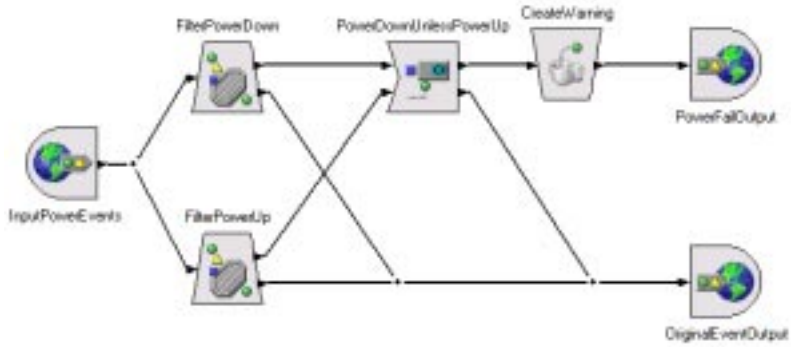
Figure 3-8 Alternative Wave-guide Circuit



The wave-guide circuit shown in Figure 3-7 on page 57 can be designed using Filter nodes instead of external filtering, as shown in Figure 3-8. Here, the external filtering selects all Power events. Inside the circuit, we use Filter nodes to divide the event flow into Power-up and Power-down events. Any other event (e.g. Power-fail) is discarded.

Both the circuits we have looked at so far discard all events that enter them. If, instead, we wanted to preserve the original events then we need to provide a connection to the output for them. Figure 3-9 shows one way to do this.

Figure 3-9 Power-up/Power-down in Output ECS Stream Policy



Most nodes provide a secondary output for events that are not output from the main output port. These output ports are usually disabled when a node is first placed on the canvas, and must be configured before you can connect to them.

Figure 3-10 shows an alternative way to accomplish the same thing, which is quicker and simpler because it uses only the default output ports of nodes.

Figure 3-10 Power-up/Power-down All Events Preserved

Notice that the two upper Filter nodes can discard events because all events are output anyway via the third Filter node. The third Filter node has a Condition parameter of `true` to pass all events. This Filter node is required because a Source node cannot be connected directly to a Sink node.

Correlation

Correlation is the processing of an event stream to improve its value, perhaps by making it smaller and perhaps by improving its information content. This processing is performed on the basis of relationships between events.

In the wave-guide scenario, to determine when the compressor cycles excessively, you want to look for Power-up events that occur soon after a Power-down. In other words, you want to output Power-down events *unless* a Power-up event arrives from the same device within a certain time. The Unless node has the required logic.

The Unless node has two parameters:

- The **Time Window** defines the minimum and maximum time between the creation time of the input event and the inhibitor event.

  All events that arrive at the Unless node Input port are detained until it is impossible for an inhibitor event to arrive to inhibit the event. This does not always equate to the *maximum time*. The time of detainment depends upon the maximum transit delays of the inhibitor and input events, and also the actual transit delays of the events.

- The **Condition** is a boolean expression that is evaluated once for each pair of input and inhibiting events, whenever an event arrives at the Unless node. If the expression evaluates to true then the input event is sent out the Inhibited Output port.

Like any other node, if an output port (such as the Inhibited Output) is not connected then the event is discarded.

Correlation for the wave-guide scenario is simply a matter of configuring the Unless node so that Power-down events flow into the Input port and Power-up events flow into the Inhibitor Input port. The Time Window is set to `(0s, 30s)` to allow Power-up events to inhibit Power-down events at any time from 0 to 30 seconds after the Power-down event was created. The Condition parameter expression is `input_event "deviceId" = inhibitor_event "deviceId"` which evaluates true if both the Power-up and the Power-down events come from the same device. Finally, the Output port is connected to a Create node. The Inhibitor Output port is not connected to anything because we want to discard inhibited events.

Output Production

The output from a correlation circuit can be:

- A subset of the events that entered the circuit.
- Modified versions of the events that entered the circuit.
- Completely new events.

In the example scenario, you connect the output of the filtering and correlation processes to a Create node. The Create node creates a warning event to output from the circuit. This event contains information indicating unacceptable wave-guide leakage.

Optimizing the Circuit

The combination of nodes you use determines the performance of the ECS Engine. You can achieve the same results using different combinations of nodes. When you have created a circuit, review it to determine whether there is a more efficient way of achieving the same result.

Consider the following guidelines when optimizing the performance of a circuit:

- Choose the appropriate stream policy (output or discard).
- Use external filtering to limit the events that enter a circuit.
- Use Filter nodes to relieve some of the computational work done by Unless nodes.
- Use more nodes rather than long, extensive conditions on fewer nodes. This makes it easier to follow the circuit logic during simulation.
- Place filters as close as possible to the input of the circuit. This reduces the number of events to be processed by nodes further down the path.
- Minimize the number of tests in a condition; they are expensive on performance.
- Keep duration windows to a minimum. This reduces delays and the amount of memory used within a node.
- Place Filter nodes before a Combine node to reduce the processing required.

4 Building a Circuit

In this chapter

This chapter provides an overview and guide to using the HP OpenView ECS Designer in Build mode. It contains:

For information about using the ECS Designer in Simulate mode, see Chapter 5, "Simulating and Testing a Circuit," on page 121.

Using the ECS Designer in Build Mode

The ECS Designer works in two modes:

- Build Mode — where you create and modify circuits
- Simulate Mode — where you debug and verify circuits.

The ECS Designer always starts in Build mode which you use to design and build your correlation circuit or Compound node.

Starting the ECS Designer

The method of starting the ECS Designer depends upon the environment in which you are using ECS. The following tables describe the various methods of invoking the ECS Designer.

Table 4-1 Starting the ECS Designer under HP-UX or Solaris

Environment	Method
HP OpenView Windows	Select `ECS Designer` from the `Tools` menu. Also see Command line below.
HP OpenView NNM	Select `ECS Designer` from the `Tools` menu. Also see Command line below.
HP OpenView Operations	1. Select `Windows-> Message Source Templates` from the `OVO` menu. 2. In the `Message Source Templates` window, double click on `Default` to expand the tree in the left hand area of the window. 3. Select an appropriate template (for example, an ECS Agent). 4. Select the template in the right hand area. 5. Select the [`Circuit...`] button to start ECS Designer.
Command line	**`ecsdes [-file` *`file`*`] [-resource` *`value`*`]`** **`ecsdes -nnm [-file` *`file`*`] [-resource` *`value`*`]`**

Table 4-2 Starting the ECS Designer under Windows NT

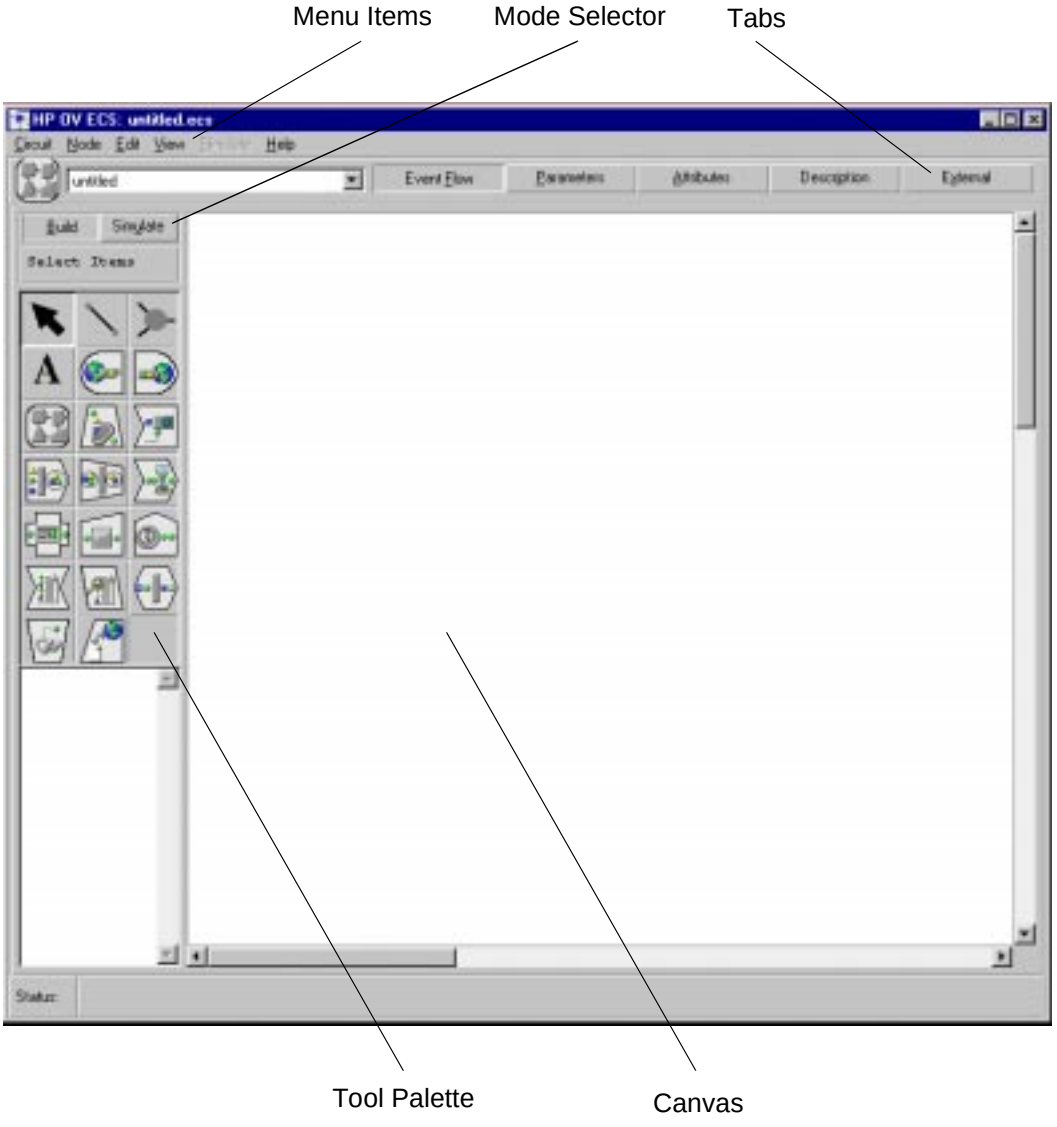
Environment	Method
HP OpenView Windows	1. Click on the [Start] button and point to Programs. 2. Select <program_group>-> ECS Designer
HP OpenView NNM	1. Click on the [Start] button and point to Programs. 2. Select <program_group>-> ECS Designer for NNM

The ecsdes command has a number of options that are described in the *ecsdes*(1m) manpage.

NOTE Before you start the ECS Designer, you must install the appropriate licenses. Refer to the *HP OpenView ECS Installation Guide* for instructions.

Unless you specified a file on the command line, a blank ECS Designer window is displayed as shown in Figure 4-1. The main display area is called the **canvas** and this is where you design and build your circuit.

Figure 4-1 The ECS Designer (Build Mode)



Help System

To display context sensitive help, press **F1** or select the `[Help]` button in a dialog box. To display the Help Table of Contents, select `Help:Table of Contents` from the menu.

Mouse

To select an item, position the mouse pointer over the item and click the left mouse button.

Click the right mouse button to pop up a menu of frequently used options. This menu varies depending on what you select with the mouse pointer.

Keyboard Accelerators

Frequently performed menu selections have keyboard accelerators.

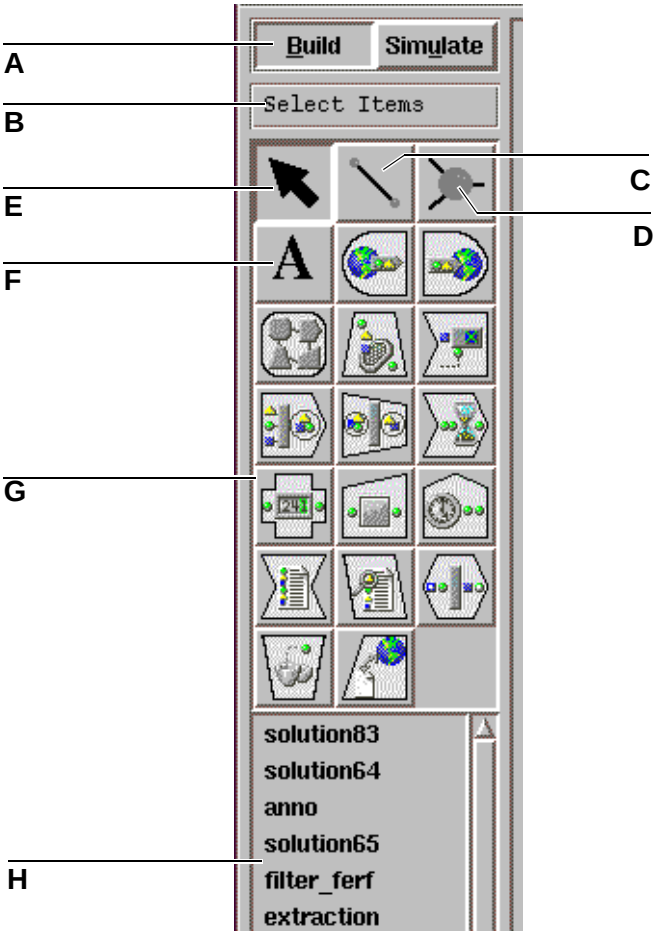
The key combinations appear next to items in the menu. For example, in the Edit menu pull-down, the Cut item is represented as `Cut Ctrl+X`.

To use these accelerators, hold down the first key and press the second key. For example, to cut a selection, hold down `Ctrl` and press `X`.

Using the Build Tool Palette

The Build tool palette at the left of the main window contains the circuit nodes and tools you use to draw or modify correlation circuits. Figure 4-2 shows the tool palette. The letters refer to important parts of the palette that are described in the following paragraphs:

Figure 4-2 Build Tool Palette



The Build tool palette consists of three areas as labelled in Figure 4-2:

- Build/Simulate Toggle (A)
- Tool and Node Palette (B through G)
- Custom Palette (H)

These are described in detail in the sections that follow.

Build/Simulate Toggle

This area contains the [Build] and [Simulate] buttons:

A The ECS Designer always starts in Build mode. Select Simulate when you have designed your circuit and you want to simulate it. See Chapter 5, “Simulating and Testing a Circuit,” on page 121.

Tool and Node Palette

This area contains the tools and nodes you use to build and modify correlation circuits.

To select a tool, click on the tool within the palette. When selected, ECS displays the tool as a depressed button and describes your selection in the information box above the palette. The way you use the tool varies depending on which tool you select.

You select nodes the same way that you select tools. When you have selected a node, click on the canvas to drop that node on the canvas. Double click on a node button to lock it on so you can drop several nodes of the same type.

When you place a node on the canvas, it has a cyan color. This indicates that you have not yet configured or connected the node. When you configure and connect the node, the background becomes gray.

B This information box indicates the tool you have selected (see “Tool and Node Palette” on page 70). The description changes when you select a tool. You cannot enter information directly into this box.

C Use the Connect tool to connect nodes together to create a correlation circuit. When you have selected the tool, click on the first point, then drag the line to the second point and release the mouse button.

To connect a series of nodes, double-click on the Connect tool.

The cursor changes shape to indicate node selection/placement mode. The Connection cursor is initially a *cross* and changes to a *bulls eye* when you drag the Connect tool over a valid connection.

D Use the Junction tool to place a junction on a connection between two nodes. To place a junction, select the Junction tool and click anywhere on the connection between two nodes.

E Use the Selection tool to move nodes around and select palette items. To select palette items, choose the Selection tool and click on the palette item. To move a node, choose the Selection tool, click on the node and 'drag and drop' it in place.

F Use the Text tool to place text annotations on the canvas. To enter text, select the Text tool and double-click on the location where you want to enter text. ECS displays the `Text` window. Type over the word `Text` with the text you want to place on the canvas, then select `[OK]`.

G The main node selection panel enables you to select specific types of primitive nodes to place on your canvas. The name of the primitive node appears in the information box (B) when you select the node. For information about each node, see "Nodes" on page 35.

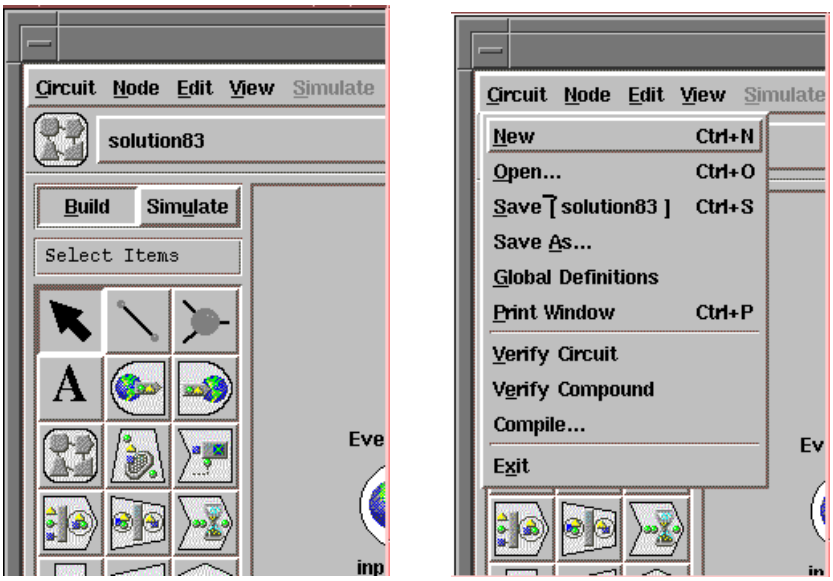
Custom Palette

H This area displays a library of pre-defined Compound nodes. You can select a Compound node from the palette and place it on the canvas just as you would any other node. For information about building a library of compound nodes, see Chapter 6, "Designing Large Circuits," on page 155.

Using the Build Menus

The following sections describe the active menu selections available in Build mode.

Figure 4-3 **Build Mode and Circuit Menu**



Circuit Menu

New	Creates a new, empty circuit. If you have an existing circuit displayed, ECS closes this circuit. If you have not yet saved the existing circuit, ECS prompts you to save it. This command is not available in ECS Designer for OVO. You must create a new template instead.
Open...	Displays the file browser to enable you to find and open a previously saved circuit. This command is not available in ECS Designer for OVO. You must select and edit a template instead

`Save`	Saves your current circuit design session. The name in braces [ ] indicates the file name of the circuit. This item is disabled if the circuit has not been modified since the last save.
`Save As...`	Saves the current circuit to a different file name. Selecting this displays the file browser in which you enter the new file name. This command is not available in ECS Designer for OVO.
`Global Definitions`	Displays the `Global Definitions` window. This window enables you to enter global definitions for the current circuit. For more information, see “Defining Global Definitions” on page 83.
`Print Window`	Prints the visible part of the circuit.
`Verify Circuit`	Verifies the entire circuit, including any Compound nodes. If errors are found, the `Verify Circuit` window stays on the screen. For more information, see “Verifying the Correlation Circuit” on page 117.
`Verify Compound`	Verifies the current level of the circuit (including any Compound nodes within the current level). This feature is useful for debugging complex circuits with embedded Compound nodes. If errors are found, the `Verify Circuit` window stays on the screen.
`Compile...`	Generates a file that can be loaded into the ECS Engine. ECS displays the file browser in which you enter the file name and directory. The file suffix `.eco` is automatically appended. For more information about what happens during a circuit compile, see “Compiling the Circuit File” on page 197.

OVO

ECS for HP OpenView Operations does not have a `Compile...` item on the `Circuit` menu. These circuits are automatically complied when you save them.

`Exit`	Closes the current circuit design session and exits the ECS Designer. If the current circuit has unsaved changes, ECS prompts you to save before exiting.

Node Menu

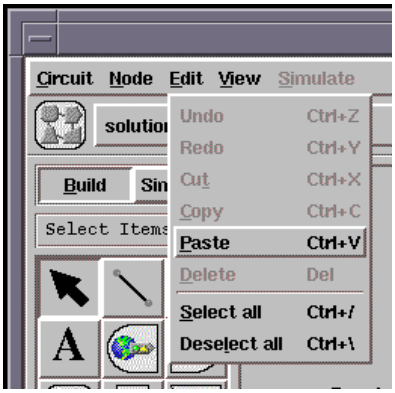
Figure 4-4 **Node Menu**

`Open`	Opens the highlighted Compound node. This is the same as double clicking on the node. If you have imported the node from the Library, you cannot modify the node. Use `Make Local` if you want to modify the node.
`Configure`	Displays the `Configure – (Node name)` window. Use this window to define the Parameters and Ports for a selected node and document these definitions for future reference. You can also display this window by placing the cursor on the node and selecting the right mouse button. For more information, see “Defining Circuit and Node Properties” on page 78.
`Make Compound from Selection`	Creates a Compound node from the highlighted group of nodes.
`Make Local`	Copies a Compound node (from the library) into the current circuit. The Compound node then becomes specific to your circuit, and you can edit it.

`Add Input Port`	Adds an input port to the selected Compound node or Combine node.
`Add Output Port`	Adds an output port to the selected Compound node or Combine node.
`Remove Input Port`	Removes an input port from the selected Compound node or Combine node.
`Remove Output Port`	Removes an output port from the selected Compound node or Combine node.
`Save to Library...`	Saves a selected Compound node to the library for reuse. ECS displays the file browser for you to select a library. If you have not verified the Compound node, ECS prompts you to do so. ECS does not save any defined external properties for the Compound node to the library as they are only appropriate at the circuit level.
`Import from Library...`	Imports a saved Compound node from the library. You cannot modify this node unless you use `Node:Make Local` to copy the node definition into your circuit, or you open the library circuit and edit it directly. Select this item to display the file browser from which you can select a Compound node.

Edit Menu

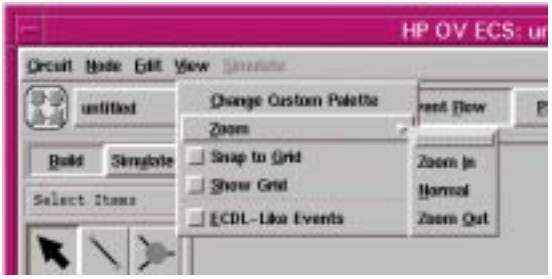
Figure 4-5 **Edit Menu**



`Undo`	Undoes the last action you performed. ECS remembers the last five actions you performed and displays the most recent action after `Undo`. If this item is unavailable and grayed out, you cannot undo the last action.
`Redo`	Redoes the last action undone. ECS displays the undone action you performed after `Redo`. If this item is unavailable and grayed out, you cannot redo the last undone action.
`Cut`	Removes the selected objects to the clipboard. The previous clipboard contents are lost.
`Copy`	Copies the selected objects to the clipboard. The previous clipboard contents are lost.
`Paste`	Pastes the clipboard contents to the canvas.
`Delete`	Deletes the selected objects.
`Select all`	Selects all objects on the canvas.
`Deselect all`	Deselects all selected objects on the canvas.

View Menu

Figure 4-6 **View Menu**



`Change Custom Palette`	Displays the file browser from where you can select a directory of Compound nodes and display the directory contents in the Custom Palette. The Custom Palette is the area below the main tool palette (Figure 6-1 on page 159).
`Zoom`	Displays a sub-menu of items to `Zoom in`, `Zoom out` or return to `Normal`. On Unix only, click on the dotted lines at the top of this sub-menu to “tear off” the menu and place it in a convenient position on your Desktop.
`Snap to Grid`	Snaps objects you place on the canvas to a grid of horizontal and vertical lines. This item toggles on and off. The `Snap to Grid` selection only positions objects you select and move after you have selected the menu item. Existing nodes are not aligned with the grid.
`Show Grid`	Show or hide grid lines. This item toggles on and off.

Simulate Menu

The Simulate Menu is only available when you switch to Simulate mode in the ECS Designer. See “Using the Simulate Menu” on page 129.

Help Menu

`Overview`	Provides an overview of the ECS Designer.
`Table of Contents`	Displays the ECS Designer Online Help Table of Contents. You can also display the Help Index from this window.
`On Window`	Displays Help for the current window.
`Using Help`	Provides an overview of the Help system and how to use it.
`About HP OV ECS Designer`	Displays current release and copyright information for the ECS Designer and associated software.

Defining Circuit and Node Properties

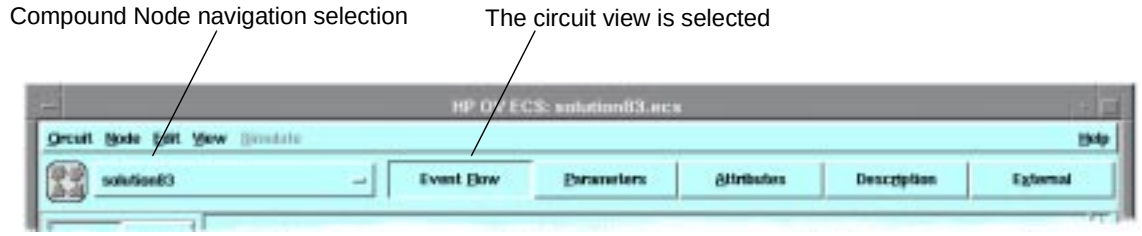
A circuit can be considered as a ‘big’ node. Using Compound Nodes, you can encapsulate a circuit in a node. It is not surprising then, that circuits and nodes share a similar set of properties.

You can set properties for the circuit, and the nodes within the circuit. To define the circuit properties, use the tabs that appear at the top of the ECS Designer canvas. To define the properties of nodes within the circuit, use the tabs in the Configure window. The following section describes circuit and node properties in more detail.

Circuit Properties

The ECS Designer presents a number of tab options that you use to define the properties of a circuit, return to the circuit view, or document your circuit.

Figure 4-7 The Circuit Properties Tab Bar



The tab bar also presents a navigation button displaying the name of the Compound node you are viewing, and enables you to view any enclosing circuit or Compound node.

Clicking the navigation button displays the levels within the Compound node. If you have drilled down into a Compound node, select the level to which you want to return.

NOTE To drill down into a Compound node, double click on the node or highlight the node and select `Open` from the `Node` menu.

The tab buttons display the following details:

Event Flow Displays the circuit design canvas where you draw and modify circuits. This is the default screen that appears when you start the ECS Designer.

Parameters Displays the circuit parameter details where you add or delete parameters to the encapsulating Compound node. Use these parameters to configure nodes within the Compound node. If you reuse this node from the library, the parameters contain default values. You can overwrite these defaults whenever you use this Compound node.

For more information, see “Creating a Parameter” on page 164.

Attributes

Displays the attribute details where you add attributes to or delete attributes from the encapsulating Compound node. These must be attributes of specific nodes within the Compound node.

Description

Displays a text box where you can view and enter comments about the encapsulating Compound node. These comments are saved to a file that the Help system calls when you request help on the Compound node.

It is important to describe how you configured the external parameters and attributes, as this is the only place where these details can be viewed outside the circuit/node.

External

Displays the external event filtering details where you define the properties of events entering the circuit Input Ports. You must define the circuit ports (by placing Source nodes) before you can select them in this window. The External tab is also where you assign a stream policy to the circuit.

Node Properties

Use the `Configure` window to define the parameters, ports and naming of nodes. This is one of the most frequently used windows in ECS.

To open the `Configure` window:

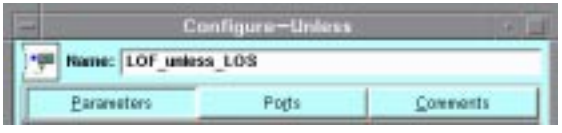
1. Click on the node you want to configure. The node appears highlighted on a gray background.
2. Select `Node:Configure` from the menu

   or

   Click the right mouse button to display a pop-up menu, select `Configure` and release.

   ECS displays the `Configure` *`Node name`* window containing the tabs you use to define the node properties.

Figure 4-8 **Node Property Tab Menu Bar**



The node is given a default name, such as *node*_1_. To change the node name, highlight the default name in the Name panel at the top of the window and type over it with the new name.

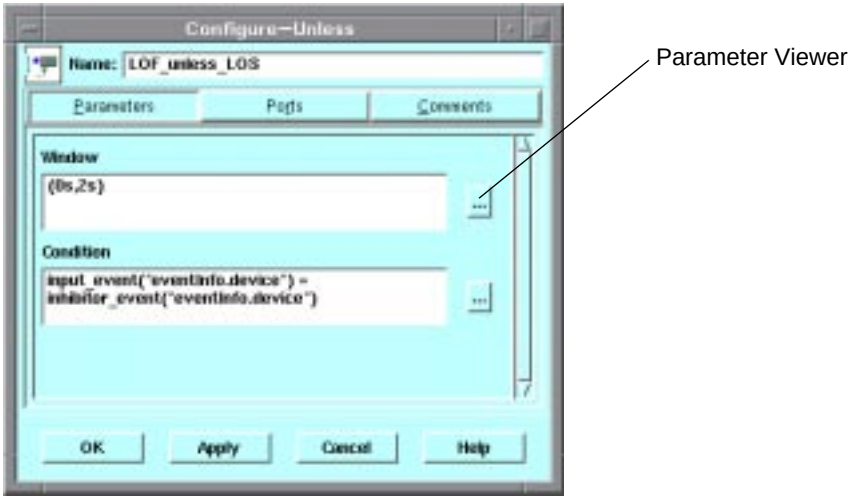
NOTE

Any node that is given a name that ends with the underscore character (_) will not appear on the ECS Designer canvas.

Parameters Tab

Select the Parameters tab to display the node parameters. Each node type has its own specific parameters. Figure 4-9 shows the Window and Condition parameters of the Unless Node. For more information about the parameters for each type of node, see the *HP OpenView Event Correlation Services Designer's Reference*.

Figure 4-9 The Configure Parameters Window

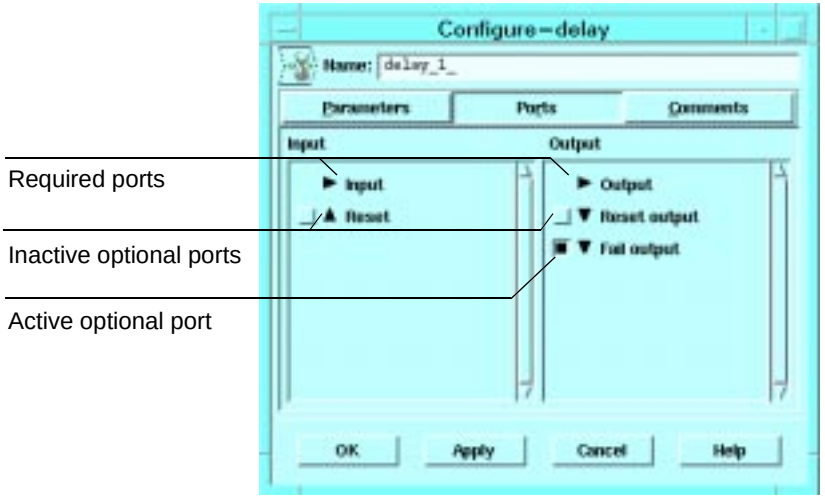


If you cannot see the entire Window or Condition, select [...] to display the Parameter Editor window.

Ports Tab

Select the Ports tab to define the input and output ports on a node. You cannot select default ports. When you select a port, the radio button changes to black. See Figure 4-10.

Figure 4-10 The Configure Dialog — Ports Screen



Ports are positioned in the same order as they are listed in the dialog.

In this example there are three active ports: Input, Output and Fail Output

Input ports

Nodes may have one or more of the following input ports:

Input	This is the main input port for most nodes. Generally, at least *one* main input port must be connected.
Reset Input	When any event arrives at this port, it causes the node to reset to a known condition. Typically, any events waiting at a main input port are flushed out of a Fail Output port. Attributes and parameters may be reset to initial or default values. This port need not be connected.

Output ports

Nodes may have one or more of the following output ports:

Output	This is the main output port from which most nodes transmit processed events.
Error Output	Some parameters have expressions that can include events as arguments. If the expression cannot evaluate the event, the event is transmitted from this port. The Error Output port need not be connected but, if not connected and an error occurs, an error message is appended to the ECS error-log file if logging is enabled.

Fail Output	When an event arrives at the Reset Input port, any events waiting at a main Input port are transmitted from the Fail Output port. This port need not be connected.
Reset Output	Any event received by the Reset Input port of a node is immediately transmitted from the node's corresponding Reset Output port. This lets you chain Reset Output ports to Reset Input ports to selectively reset only certain nodes in a chain, without affecting other nodes. This port need not be connected.

Comments Tab Select the `Comments` tab to document the node.

Defining Global Definitions

Define common functions, object identifiers or event attribute names in the `Global Definitions...` window as named values. You can then use these named values in node parameters throughout the circuit.

The parameters you enter to define a node's behavior can refer to global definitions. For example, a Filter node parameter may reference a named value called **`DEVICE`** that you defined in the `Global Definitions` window as an event attribute "`eventInfo.notificationIdentifier`". See Figure 4-11.

Figure 4-11 The Global Definitions Window

To open the `Global Definitions` window, select `Circuit:Global Definitions...` from the menu bar. See Figure 4-11.

Enter the named values and definitions directly in to this window.

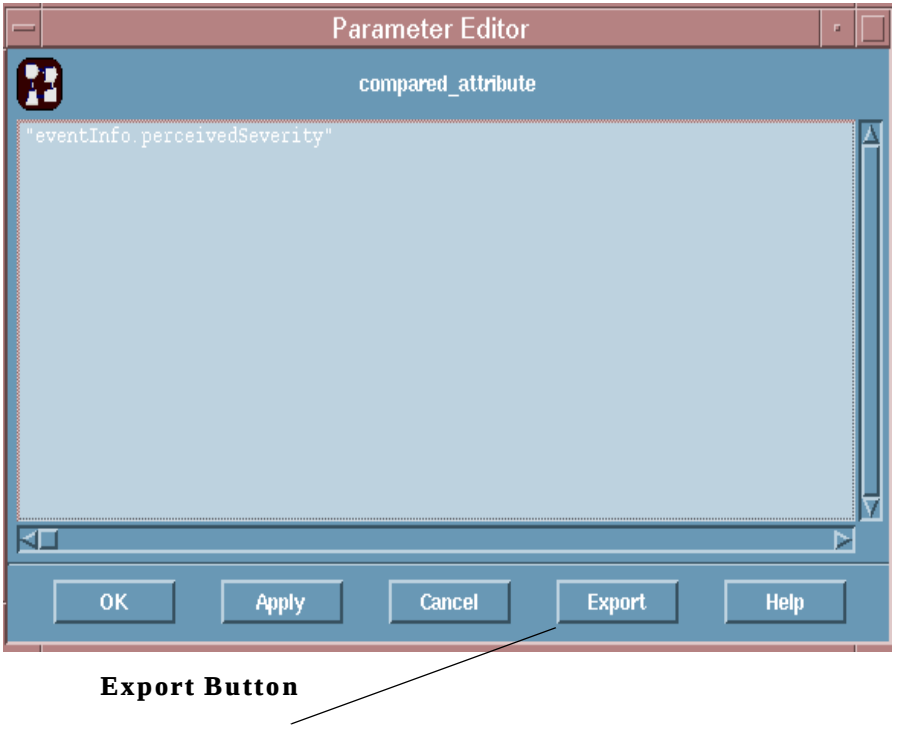
When you are defining node parameters, keep the `Global Definitions...` window open for easy reference.

Import - Export of ECDL statements

The nodes in the ECS circuit require correlation conditions to be given. These conditions will be evaluated by the ECS engine for correlation. The conditions are specified using ECDL. These ECDL statements can be either imported or exported into the node configuration using the ECS designer. The configuration window of the nodes has a new button provided for “Export”.

It is possible for the user to export the contents of the node parameters. The parameters, global definitions can be exported into a text file, by selecting the “export” button in the window. The user is prompted to enter the filename into which the contents have to be exported. The exported files are stored in an ASCII format.

Figure 4-12 Import-Export of ECDL Statements



It is also possible for the user to import ECDL statements in text file into node configuration parameter and global definitions. ECDL statements from a particular file can be stored into one text file. This file can be

specified in the configuration parameter window with the usage of "$include" directive before that.

The addition of this directive does not require any change in the designer.

```
let
        val evtype = [ ("ber", "Trap-PDU"),
                       ("mdl", "SimpleEvent"),
                       ("ber","EventReportArgument") ]
        val events = create_events evtype


        val _ = (nth 1 events) alter (
$include alterspec/TrapPdu.ecdl
                                      )


        val _ = (nth 2 events) alter (
$include alterspec/SimpleEvt.ecdl
                                      )


        val _ = (nth 3 events) alter (
$include alterspec/EventRepArg.ecdl
                                      )
in
        events
end
```

NOTE

The filename is relative to the directory where designer is started.

If an alternative directory is needed, then the directory has to be exported into an environment variable before the designer is launched.

For Example, $include $ECDL_HOME/*filename*.ecdl

where

$ECDL_HOME = *path to which it is exported*

The contents of `alterspec/TrapPdu.ecdl`, `alterspec/SimpleEvt.ecdl` and `alterspec/EventRepArg.ecdl` gets replaced in place. The relative path of the file has to specified in order that the contents can be imported.

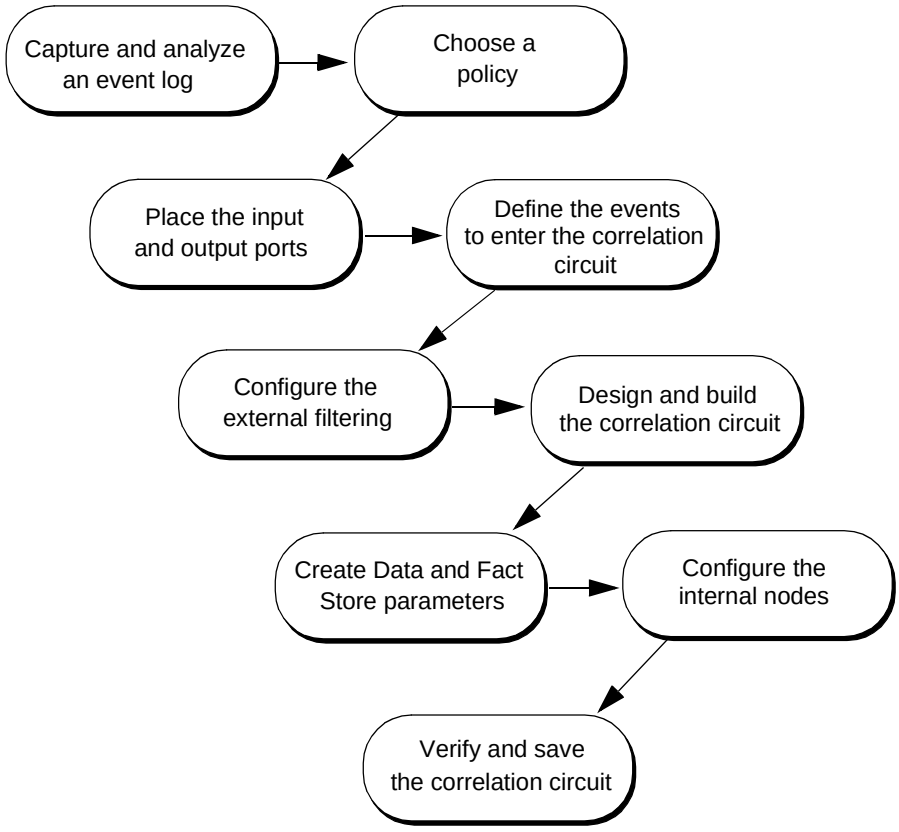
The Circuit Design Process

A circuit should be designed to do one task. If multiple correlation tasks exist then separate circuits should be designed for each task.

The ECS Designer can display and simulate only a single circuit at a time. However the ECS Engine supports multiple streams and multiple circuits in each stream. The effect of all the circuits on a stream is the combined effect of all the circuits running in parallel.

The flowchart shown in Figure 4-13 summarizes the main tasks in designing a small correlation circuit. For information about designing and building large circuits, see Chapter 5, “Simulating and Testing a Circuit,” on page 121.

Figure 4-13 Task Flow for Circuit Design



Capturing and Analyzing an Event Log

To determine your event filtering and correlation needs, analyze an event log generated at the time the problem occurred. The event log helps you to determine the events you want to enter the circuit, those you want to filter, and those you want to correlate.

You also need to determine the events you expect to exit from the circuit.

Capturing an Event Log

You can use the ECS Engine to log network events to a file. First ensure that the ECS Engine is running then turn on input event logging by executing the following command:

```
ecsmgr -log_events input on
```

When you have captured a log of events, disable the event logging:

```
ecsmgr -log_events input off
```

OVO

This procedure does not apply to HP OpenView Operations. For information about logging OVO events, see “Logging OVO Events” on page 91.

Analyzing the Event Log

You can view the event log file (a flat ASCII file) with any text editor. To analyze this file:

- Identify the window of time relating to the problem.
- Identify all events relating to the problem within this window.
- Determine the events to be removed because they do not relate to the problem.
- Identify any patterns within the remaining events that you can use as the problem's signature.
- Determine what events you can filter because they are either duplicated or the relevant information is contained within other events.
- Determine what events you expect to be output from the circuit.

Once you have determined this information, you can build a circuit.

Logging OVO Events

You can configure OVO's event correlation processes to log messages on both the management server and the managed node.

Messages going into and out of the ECS Engine are logged in the files `ecevilg` and `ecevolg` respectively. These log files are used for debugging and testing circuits in the ECS Designer, and reside in the following directories:

Management Server	HP-UX 10.X	`/var/opt/OV/log/OpC/mgmt_sv`
	Windows NT	
Managed Node	HP-UX 10.X	`/var/opt/OV/log/OpC`
	Windows NT	

For example, to simulate a correlation circuit in the ECS Designer, you need the log file `evevilg`. You must transfer the file manually from the managed node to the server.

In order to rebuild an original message from the content of the log file, the log file must contain as much information as possible relating to the messages that passed through the correlation circuit. Consequently, the structure of all OVO message log files is designed to ensure that they contain the required message attributes in the appropriate format.

You control Input and Output event logging from the `Options` window, which you access from the `Message Source Templates` window. Enabling (setting the appropriate check mark) logging for any one of the templates you distribute to a managed node means that logging is switched on for all event correlation templates on that managed node.

Optical Fibre Cut Scenario

To understand the value of an event log file in the process of designing and building a circuit, let's look at a simple scenario.

The Introduction to this Guide describes and illustrates the scenario of a major optical fibre cut (Figure 2-1 on page 21). In this scenario a large number of events are generated. If an event log was captured at the time the problem occurred and you analyzed its content, you would see a

pattern of hierarchical dependencies. You could use the information in the event log to determine:

1. The events to input to the circuit, and events to be filtered.
2. The conditions under which specific events are to be processed.
3. The events to be output from the circuit.

Example 4-1 Example Entry in the Event Log File

```
EventReportArgument {
    managedObjectClass {1 3 6 1 4 1 11 2 2 2 2},
    man agedObjectInstance distinguishedName : {
        {
            {
               attributeType {2 9 3 2 7 4},
               attributeValue SystemId name : "rvtf.hp.com.au"
            }
        },
        {
            {
               attributeType {2 9 3 2 7 1},
               attributeValue SimpleNameType number : 819354328
            }
        }
    },
    eventTime "19951219063926Z",
    eventType {1 2 3 4 5 115 1},
    eventInfo ASN1Module.Alarm {
       severity 0,
       source "Melbourne",
       device "ADM390",
       description "LOS"
    }
}
% ber:EventReportArgument:
# Alarm
```

In this scenario, the events are in a CMIP protocol, created as two event types. The events of interest have an `eventInfo.description` field which can have a value of LOS, LOF, or FERF.

To continue the example, the information in the event log file could tell you that the new circuit will need to:

- Accept only those events directly related to the problem

- Pass all LOS events out of the circuit
- Pass a LOF event if a LOS event has not been created by the same device within two seconds
- Pass a FERF event if no LOF events have been created by the same device within two seconds.

After you have built a circuit to perform the analysis described above, you can test it for correct operation by using the content of the captured Event Log file as input to the circuit while it is still resident within the ECS Designer.

Using Compound Nodes

Network management problems can sometimes be simplified by encapsulating each part of a circuit in a Compound node. This makes the problem more manageable, easier to test, and makes parts of the circuit reusable.

Compound nodes are described in Chapter 3, “Approach to Circuit Design,” on page 33. The use of Compound nodes is further described in Chapter 6, “Designing Large Circuits,” on page 155.

The scenario in this chapter is simple and does not use Compound nodes.

Choose a Policy

Each stream of events in an ECS Engine has a policy that defines what happens to events that do not enter any of the circuits enabled on that stream, and events that are not explicitly discarded or output. The stream policy can affect the way you design a circuit, and you can set the policy so that a circuit can only be loaded into a stream with that policy.

Table 4-3 Circuit Policies

Policy	Description
Output	Outputs events that are not discarded by the circuit.
Discard	Discards events that are not output by the circuit. This policy is not available for NNM correlations.
Unspecified	A circuit that has an unspecified policy will run in a stream with either policy.

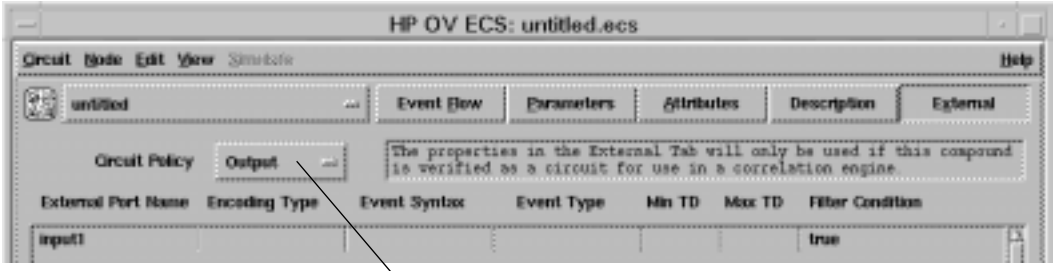
Most circuits are designed for an output policy. The cable cut scenario requires an output policy to ensure that events not related to the problem (that is, events not selected for input to the circuit) are passed on unaffected.

Use a discard policy for circuits that are in special-purpose streams, such as an event stream that drives a security breach alarm. Here, you only want the alarm to be raised if a specific pattern is detected and you do not want any other event to raise the alarm.

If the circuit can be enabled in a stream of either policy then the circuit policy can be set to `Unspecified`. For example, a circuit that modifies events by translating event status codes from one set of standards to another, may be required to run in streams with either policy.

The policy is set on the External tab of the circuit and can be changed at any time while the ECS Designer is in Build mode. You cannot change the policy of a circuit in Simulate mode, or when the circuit is being run in the ECS Engine. See Figure 4-14 on page 95.

Figure 4-14 Circuit Policy



Placing the Input and Output Ports

To define input and output ports for the circuit, place Source and Sink nodes on the ECS Designer canvas.

Events enter and exit the circuit through the Source and Sink nodes. At the highest level (circuit), events enter the circuit from the external environment. At embedded levels (Compound node), events enter the circuit from the enclosing circuit.

To place Source and Sink nodes:

1. Select a `Source` node from the tool palette and place it on the left of the canvas.

   The node is connected to an input port with the default name `input1`.

2. Select a `Sink` node from the tool palette and place it on the right of the canvas.

   The node is connected to an output port with the default name `output1`.

You can have more than one Source or Sink node.

Changing the Node Names

Rename each node to remind you of the node's intended purpose. To change the default name on a node:

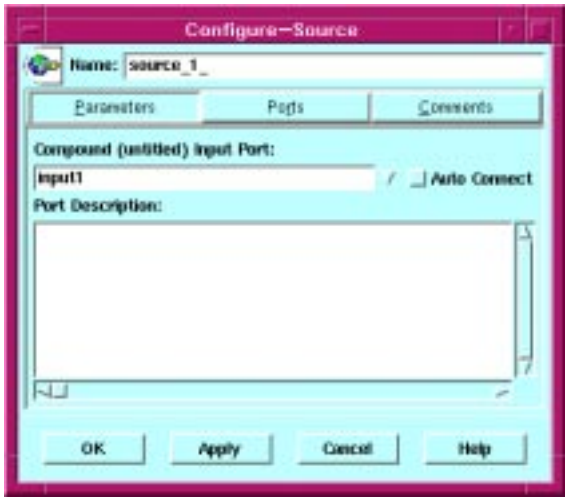
1. Click on the node you want to rename.
2. Select `Node:Configure` from the menu. ECS displays the `Configure node` window.
3. Select the `Parameter` tab to display the parameter window (see Figure 4-15).
4. Type over the default node name with the new node name.

   You can also change the input port name.

5. Select `[OK]` to close the `Configure — node` window.

NOTE Any node that is given a name that ends with the underscore character (_) will not appear on the ECS Designer canvas.

Figure 4-15 The Configure Window



NOTE If you are configuring ports for a Compound node, you can select one input and one output port to be autoconnected when you drop the node on an existing connection. ECS splits the existing connection and forms new connections; one from the starting point of the original connection to the autoconnect input port of the compound node, and another from the autoconnect output port of the compound node to the original connection's endpoint.

Defining the Events to Enter the Correlation Circuit

Based on the analysis of the log file you collected, you know the type of events you want to enter the circuit, and the minimum and maximum transit delays for the incoming events. You have also placed one or more Source nodes on the canvas. You now use the `External` tab to specify which events can enter the circuit through each Source node.

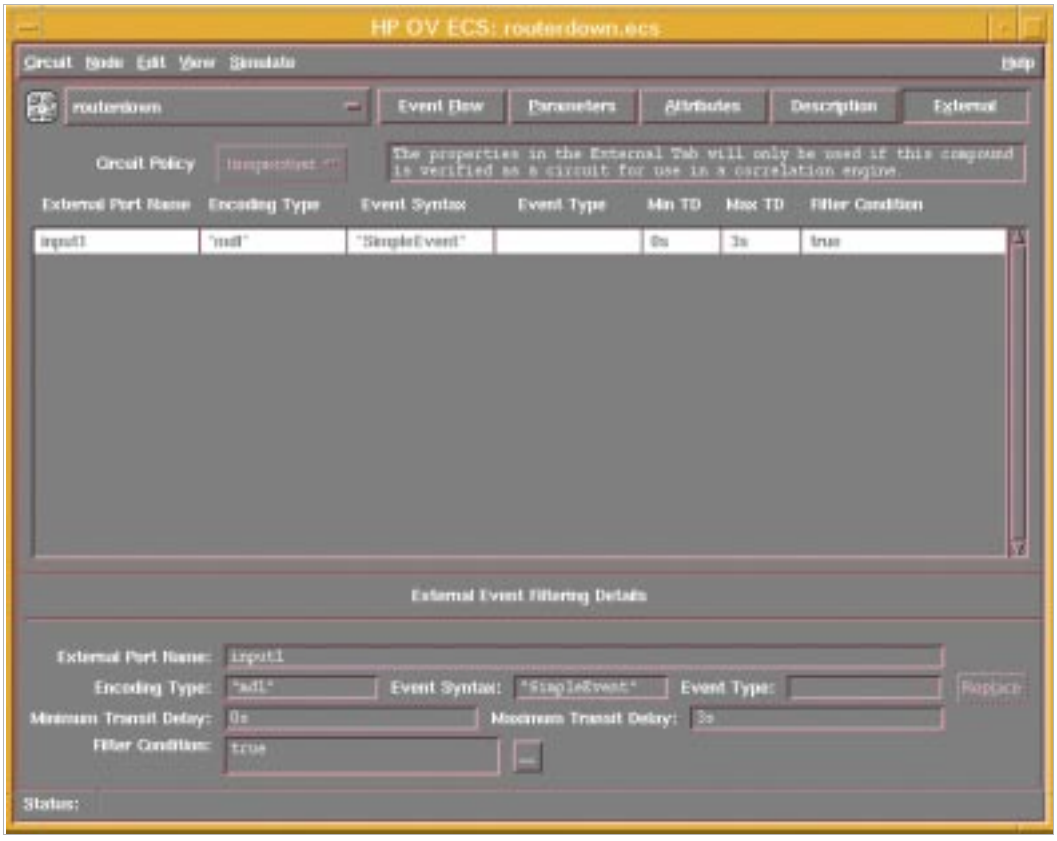
When you place a Source node on the canvas, ECS associates an external port with the node. The external port is used only when the circuit is connected directly to the external event stream. If the circuit is used as a Compound node then the external ports are not used and the associated External tab properties are ignored.

The External tab properties control the policy for which this circuit is designed, and the events selected to enter each of the Source nodes. There are three major criteria by which events can be selected:

- **The kind of event**. The Encoding Type, Event Syntax and Event Type are efficient filter mechanisms that are used to select events. Some endecoders are able to perform this selection without the need to decode the event.
- **Transit Delay (TD)**. Events that are too old (or too young) can be excluded from entering certain Source nodes. Top-level circuits (those not destined to be used as a Compound node) with nodes that store events at their input ports (the `Unless`, `Table`, `Delay` or `Combine` node), *must* specify an acceptable Transit Delay range.

  If you do not specify a transit delay for these circuits, they will fail to load in the ECS Engine, and you will not be able to Simulate the circuit in the ECS Designer.
- **Filter Condition**. This allows you to specify a selection test based on the internal event attributes. Filter Conditions may be less efficient than other selection criteria as the event may need to be decoded, and the Filter Condition expression must be evaluated for each event entering the filter node.

Figure 4-16 External Tab Option



The Type of Event

You can specify the type of events you want to enter the circuit through each port. The type of event is specified by the Encoding Type, Event Syntax and Event Type parameters on the External tab. The more parameters you specify, the more efficient the selection process is. For example, just specifying Encoding Type will ensure that only events of the specified encoding type (`BER`, `MDL`, or `OpC_Msg`) can enter the circuit. However if you specify all three parameters: `Encoding Type`, `Event Syntax` and `Event Type`, then the test is actually performed more efficiently. The events selected to enter the circuit are even more tightly selected, meaning that the circuit has to process fewer events.

The three parameters that specify the kind of event are explained in greater detail in the following sections.

Event Encoding Type

An event's encoding type determines the encode/decode module used to translate an event to and from its native format. The value corresponds to the event's `encoding_type` header attribute. The main protocols and encoding types are shown in Table 4-4.

Table 4-4 Event Type Parameter Values for Common Protocols

Protocol	Value	Event Encoding Type
CMIP	`"ber"`	The string value `"ber"` selects all events that use Basic Encoding Rules (SNMP and CMIP)
SNMP		
ASCII	`"mdl"`	The string value `"mdl"` selects all events that use Message Description Language.
OVO	`"OpC_Msg"`	The string value "OpC_Msg"

Event Syntax

An event's syntax determines how the event's attributes are read and written. The event syntax is stored in the event's `event_syntax` header attribute. The main protocols and event syntaxes are shown in Table 4-5.

Table 4-5 Event Syntax Parameter Values for Common Protocols

Protocol	Event Syntax
CMIP	The OID (identifying the GDMO notification)
SNMP	The string value `"Trap-PDU"`
ASCII	The syntax name determined by the MDL message definition used to read and write these events. For example `"SimpleMDL"`
OVO	The string value `"OpC_Msg"`

Event Type

The event type is represented by the ECS header attribute `event_type`. The main protocols and event types are shown in Table 4-6.

Table 4-6 Event Type Parameter Values for Common Protocols

Protocol	Event Type
CMIP	The OID of the notification
SNMP	The generic trap number (1-6)
ASCII	The type determined by the MDL definition used to read and write these events or the syntax if the `event_type` is not specified
OVO	Message attribute `"msgtype"`

Transit Delays

ECS uses the transit delay value to determine the waiting period for the Unless, Table, Combine, and Delay nodes.

Use the Minimum and Maximum Transit Delay fields to specify a window of time for events to arrive at the input port of the node. The delay values are the difference between the current clock time of the ECS Engine and the creation time of the event.

You can perform different types of correlation depending on how long it took an event to arrive. By configuring the transit delay on more than one port, you can direct events into an appropriate event path within the circuit.

NOTE Compensating for transit delays depends on knowing the event's creation time. Creation time is only known for some events. See "Time Synchronization" in the *HP OpenView Event Correlation Services Designer's Reference* for details.

Filter Condition

ECS uses the filter condition to determine whether the events can enter the circuit or not.

The filter condition is an ECDL expression which will evaluate to either TRUE or FALSE. The circuit accepts only those events that satisfy the filter condition i.e the expression evaluates to TRUE.

NOTE By default the filter condition is set to TRUE.

Specifying Event Type, Syntax, Encoding Type and Transit Delays

To define the event type, event syntax, event encoding type, and transit delays of incoming events:

1. Select the External tab to display the external filtering details. See Figure 4-16 on page 99.
2. Select the input port you want to define from the External Port Name list.

   The port appears in the External Event Filtering Details panel.
3. Enter the Encoding Type. This must be a string value.
4. Enter the Event Syntax.
5. Enter the Event Type.
6. Enter the Minimum Transit Delay.

   This value must be an integer with s m or h for seconds, minutes or hours. For example 0s.
7. Enter the Maximum Transit Delay.

   This value must be an integer with s m or h for seconds, minutes or hours. For example 10s.
8. Enter the ECDL expression in the Filter Condition.

   This is set to TRUE by default.

   For example input_device("deviceID")="VIKING".

9. Select [Replace] to write the values to the selected port.

Designing and Building the Correlation Circuit

Setting up Input and Output Ports

Filtering is the process of reducing the number of duplicate events or removing events that are no longer relevant or are not relevant to the goal of the circuit. To increase the efficiency of the ECS Engine, perform the filtering as early as possible within the circuit.

You can perform the initial filtering on the Input ports. In this scenario, you want to input all LOS events and these are event type `1.2.3.4.5.115.1`.

To input all LOS events into the circuit:

1. Select a Source node from the tool palette and place it on the canvas.
2. Click on the `Source` node, then select `Node:Configure` from the menu.
3. Change the `node Name` to:

   `Events_LOS`

   and the `Input port:` to:

   `input_LOS`.

   Naming the node and ports helps you identify the intended event path later.
4. Select `[OK]` to save the configuration.
5. Select the `External` tab to display the external filtering details.
6. Select an input port and define the `Event Type:` as:

   `1.2.3.4.5.115.1`

   Only events of this type can enter the `input_LOS` port.
7. Set the `Minimum Transit Delay:` to:

   `0s`.
8. Set the `Maximum Transit Delay:` to:

10s.

ECS ignores events with transit delays outside this time window.

9. Select [Replace] to save the event type and transit delay values.
10. Select the Event Flow tab to return to the circuit.
11. Select another Source node from the tool palette and place it on the canvas.
12. Click on the Source node, then select Node:Configure from the menu.
13. Change the node Name to:

    Events_ALL

    and the Input port: to:

    input_ALL
14. Select [OK] to save the configuration.
15. Select the External tab to display the external filtering details.
16. Select the input_ALL port and define the Event Type: as blank.

    This port accepts all other events.
17. Set the Minimum Transit Delay: to:

    0s.
18. Set the Maximum Transit Delay: to:

    10s.

    ECS ignores events with transit delays outside this time window.
19. Select [Replace] to save the event type and transit delay values.
20. Select the Event Flow tab to return to the circuit.
21. Select a Sink node from the tool palette and place it on the canvas.
22. Click on the Sink node, then select Node:Configure from the menu.
23. Change the node Name to:

    Events_OUT
24. Select [OK] to save the configuration.

The source nodes are now the entry points for event with distinctly

different attributes.

The first source node only allows events of type `1.2.3.4.5.115.1` with an `eventInfo.description` field of LOS to enter the circuit. Filter the events from this source to pass only those events with the required attributes.

The second source node allows all remaining events to enter the circuit. The next step is to filter events from this source into two paths, one passing LOF events and the other passing FERF events.

See “Configuring Internal Nodes” on page 107 for details.

Configuring Internal Nodes

Event correlation involves processing an event stream to improve its value, perhaps by reducing the number of events in the stream and/or perhaps by improving the information content of the stream. This processing is based on the relationship between the events.

In the scenario, there are two flows of events derived from the two input ports (Source nodes) and two correlation activities to perform:

- To pass a LOF event if a LOS event has not occurred within two seconds
- To pass a FERF event if a LOS event has not occurred within two seconds.

To perform this correlation, you must configure a group of nodes along the paths the events will follow. These node must be provided with parameters that cause them to perform the appropriate filtering.

Working with Nodes

When you are laying out correlation circuits and Compound nodes, there are certain filtering and correlation actions you use frequently. Table 4-7 lists these common event correlation and filtering tasks and the node appropriate to each task:

Table 4-7 Node Behavior

If you want to do this task …	Then use this node …
Remove events not matching a condition.	Filter Node (`True Output` port)
Remove events matching a condition.	Filter Node (`False Output` port)
Remove an event A if another event B occurs between 1 and 2 seconds *after* event A.	Unless Node (`Window` parameter: `(1s,2s)`)

Table 4-7 Node Behavior

If you want to do this task …	Then use this node …
Remove an event A if another event B occurs between 1 to 2 seconds *before* event A.	Unless Node (`Window` parameter: `(-2s, -1s)`)
Group events together into a composite event.	Combine Node
Extract a component event from a composite event.	Rearrange Node
Rearrange component events in a composite event.	Rearrange Node
Calculate the rate at which events flow along a connection (events/sec).	Rate Node
Generate an event periodically to trigger other nodes.	Clock Node
Count the number of events passing a point within the Event Flow	Count Node
Create a new event when triggered by a specific incoming event.	Create Node
Store copies of whole event or extracted event attributes in creation time order to be referenced by other nodes	Table Node
Extract copies of events from a Table node.	Extract Node
Reorder incoming events into creation time order.	Delay Node
Modify the values of an incoming event's attributes.	Modify Node

Node Parameters

You can modify each node's specific behavior by changing the node's parameters.

Use the Parameter tab of the Configure window to enter and modify node parameters.

Node parameters are written in ECDL (Event Correlation Description Language). See the *HP OpenView Event Correlation Services Designer's Reference* for details on node parameters and ECDL.

Node Attributes

The Count, Rate, and Table nodes have attributes. These are data values that can be read by parameters in other nodes. For example, the Count node has an attribute value called Count that returns the number of events that have passed through the node.

You cannot enter or modify primitive node attributes.

Node Ports

Each node has a set of ports that you can connect to ports of other nodes to create the circuit. Some ports are required and these ports are always displayed when you select the node. Others are optional.

Use the Connect tool to connect node ports.

To add ports, use the Configure Node window. See "Node Properties" on page 80.

For More Information

Each of the primitive nodes are explained in the *HP OpenView Event Correlation Services Designer's Reference*. Refer to this book to determine the appropriate node to use within the circuit.

Placing and Configuring the Filter Nodes

Before you place the Filter nodes, make sure you have placed the Source and Sink nodes on the canvas and configured these nodes for the correct event types and transit delays. See "Setting up Input and Output Ports" on page 104.

On the first event pathway, the filtering passes only LOS events from the input port configured for Event Type: 1.2.3.4.5.115.1 (Source Node Events_LOS).

To add a Filter node:

1. Select a Filter node from the tool palette and place it on the canvas.

2. Select the Connect tool from the tool palette and connect the Events_LOS Source node to the Filter node.

   The Status line displays the name of the port and node you are connecting. See Figure 4-17.

Figure 4-17 Status Line Message

3. Click on the Filter node, then select Node:Configure from the menu.
4. Change the node Name to:

   filter_LOS

   Naming the node helps you identify the intended event path later.

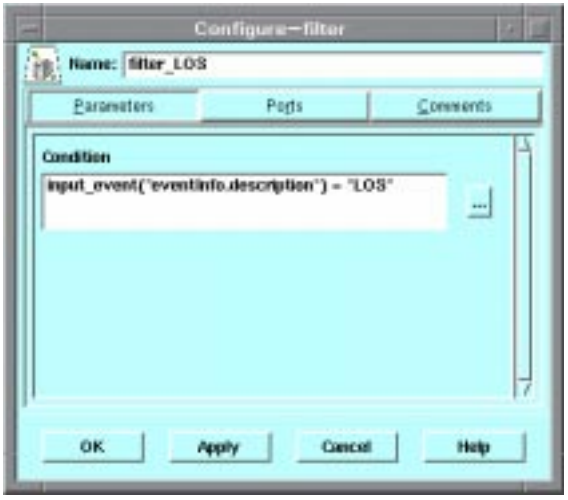
5. Enter the Condition as:

   input_event "eventInfo.description" = "LOS"

   in the Parameters page of the Configure Filter window. See Figure 4-18. This node passes events that evaluate true to the condition. If the expression evaluates to true, the event is passed out the True output, otherwise it is passed out the False output.

6. Select [OK] to complete the node configuration.
7. Select the Connect tool from the tool palette and connect the filter_LOS Filter node to the Events_Out Sink node.

Figure 4-18 Configuring the Filter Node



In the second event pathway, all remaining events entering the input port are defined by the Source node `Events_ALL`. The filtering must pass only FERF and LOF events. To optimize the circuit design, create two separate paths, one of FERF events and one of LOF events.

To create the two paths, place two Filter nodes on the canvas, then name the nodes `filter_FERF` and `filter_LOF`.

To configure and connect the two Filter nodes:

1. Click on the first Filter node and select `Node:Configure` from the menu.
2. Change the node `Name` to:

   `filter_FERF`

   Naming the node helps you identify the intended event path later.
3. Enter the `Condition` as:

   `input_event "eventInfo.description" = "FERF"`

   in the `Parameters` page of the `Configure Filter` window.
4. Select `[OK]` to complete the node configuration.
5. Click on the second Filter node and select `Node:Configure` from the

menu.

6. Change the node Name to:

   filter_LOF

7. Enter the Condition as:

   input_event "eventInfo.description" = "LOF"

   in the Parameters page of the Configure filter window.

8. Select [OK] to complete the node configuration.

9. Select the Connect tool from the tool palette and connect the Events_ALL Source node to the filter_FERF Filter node.

10. Select the Connect tool again, click on the connection between the Events_ALL Source node and the filter_FERF Filter node, and connect it to the filter_LOF Filter node.

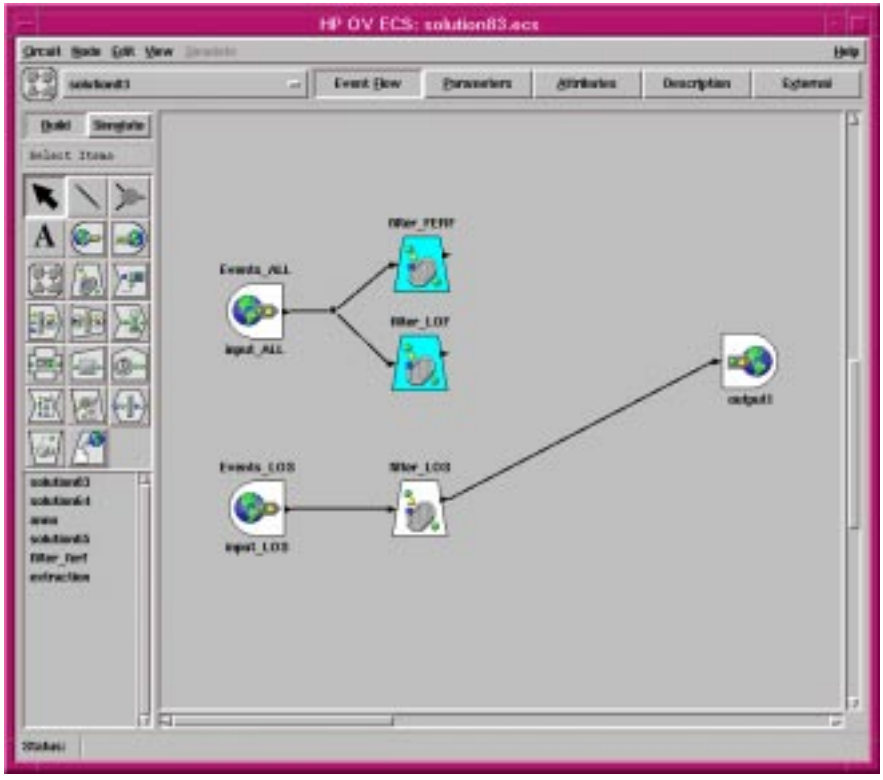
    ECS creates a junction node automatically.

The filtering part of the circuit is now complete. There are three paths for events flowing within the circuit:

- A path for FERF events
- A path for LOF events
- A path for LOS events.

Your circuit should look similar to Figure 4-19.

Figure 4-19 The Filtering Circuit



Placing and Configuring the Unless Nodes

In the scenario, there are now three event paths derived from the filtering. The two correlation activities to build into the circuit are:

- To pass a LOF event if a LOS event has not occurred within two seconds
- To pass a FERF event if a LOF event has not occurred within two seconds.

This type of correlation can be performed by an Unless node.

To place and configure the Unless nodes:

1. Select an Unless node from the tool palette and place it on the canvas

opposite the filter_FERF Filter node.

2. Click on the Unless node and select Node:Configure from the menu.
3. Change the node Name to:

   FERF_unless_LOF

   Naming the node helps you identify the intended event path later.

4. Enter the Window as:

   (0s,2s)

   This defines the window of time in which an event can arrive at the Unless node's input port, from current time until 2 seconds later.

5. Enter the Condition as:

   ```
   input_event "eventInfo.device"
    = inhibitor_event "eventInfo.device"
   ```

   This Condition requires the input event to come from the same device as the inhibitor event.

6. Select [OK] to complete the node configuration.
7. Select the Connection tool from the tool palette and connect the filter_FERF Filter node to the input port of the FERF_unless_LOF Unless node.

   The *HP OpenView Event Correlation Services Designer's Reference* describes the position of the ports for each node. In the case of the Unless node the input port is the upper left port, the inhibitor port is the lower left port.

8. Select the Connection tool and connect the filter_LOF Filter node to the inhibitor port (lower left) of the FERF_unless_LOF Unless node.
9. Select the Connection tool, click on the connection between the filter_LOS Filter node and the Events_Out Sink node, and connect this to the output port of the FERF_unless_LOF Unless node.

   ECS creates a junction node automatically.

10. Select an Unless node from the tool palette and place it on the canvas opposite the filter_LOF Filter node.
11. Click on the Unless node and select Node:Configure from the menu.
12. Change the node Name to:

LOF_unless_LOS.

13. Enter the Window as:

    (0s,2s)

14. Enter the Condition as:

    ```
    input_event "eventInfo.device"
     = inhibitor_event "eventInfo.device"
    ```

15. Select [OK] to complete the node configuration.

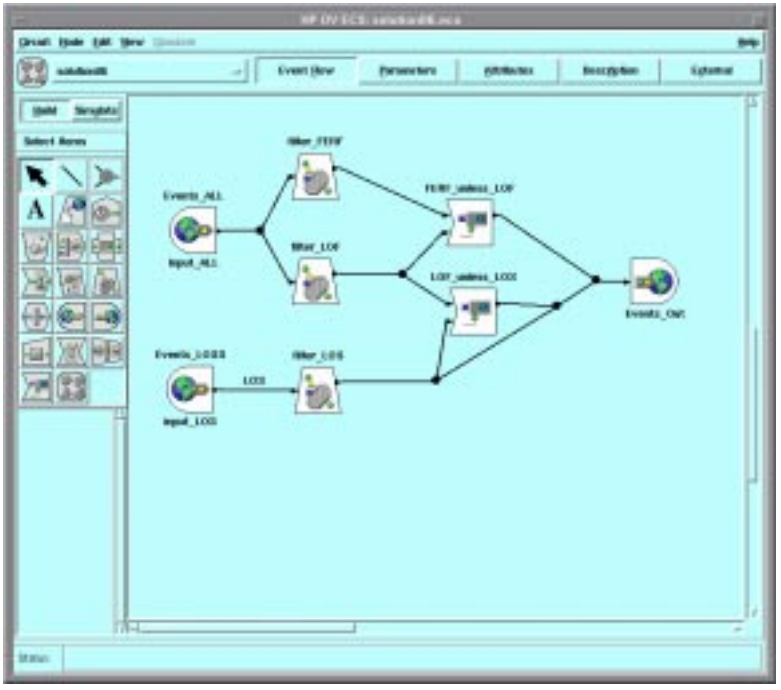
16. Select the Connection tool from the tool palette, click on the connection between the filter_LOF Filter node and the FERF_unless_LOF Unless node, and connect this to the input port of the LOF_unless_LOS Unless node.

17. Select the Connection tool, click on the connection between the filter_LOS Filter node and the Events_Out Sink node, and connect this to the inhibitor port of the LOF_unless_LOS Unless node.

18. Select the Connection tool, click on the connection between the filter_LOS Filter node and the Events_Out Sink node, and connect this to the output port of the LOF_unless_LOS Unless node.

The Event Correlation circuit is now complete and should appear similar to Figure 4-20.

Figure 4-20 The Scenario Circuit



Verifying and Saving the Correlation Circuit

You must verify the circuit before you can simulate and test it. Although it would be wise to do so, you need not save the circuit before simulation and testing.

Verifying the Correlation Circuit

Verify the circuit to check the configuration of the nodes and to ensure that the parameters are specified correctly.

If you have used Compound nodes within the circuit, `Verify Circuit` checks all levels of the circuit and the input ports.

To verify a circuit, select `Circuit:Verify Circuit` from the menu. ECS displays the `Verify Circuit` window. See Figure 4-21.

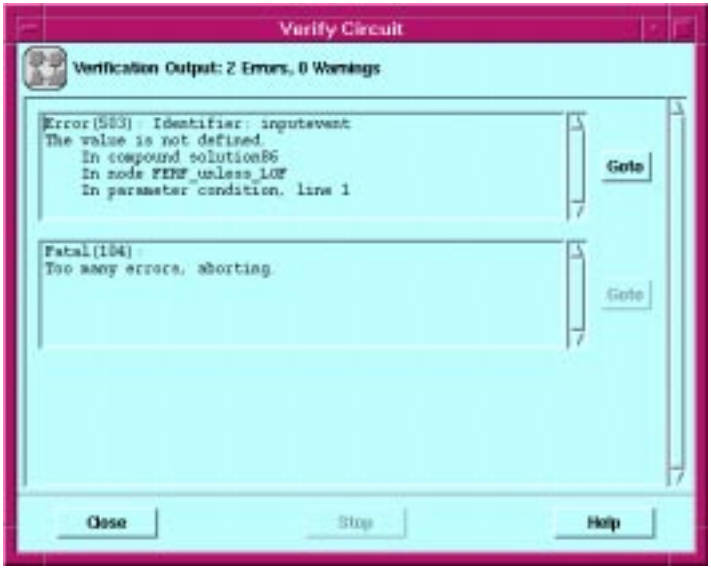
If the verification is successful, the `Verify Circuit` window closes and the `status` window displays `Verification Success`.

If an error occurs, the `Verify Circuit` window remains open and indicates the name of the node in error together with the error details. To go to the node in error, select `[Goto]`.

The node in error appears on the canvas window highlighted in red. Select `Configure` and correct the condition.

To close the `Verify Circuit` window, select `[OK]`.

Figure 4-21 Verify Circuit Window



Saving the Correlation Circuit

When you are designing and building a new circuit, ECS displays the circuit name as `untitled`. The `Save` item on the `Circuit` menu appears grayed out (unavailable) until you name the circuit. To do this, use `Save As`.

OVO

ECS for HP OpenView Operations does not have a `Save As` item on the `Circuit` menu. See “Saving OVO Circuits” on page 119 for more information.

To save a circuit:

1. Select `Circuit:Save As` from the menu to display the file browser.
2. Enter the circuit name (including path) in the `File` panel.
3. Select `[OK]` to save the circuit.

File Naming Restrictions

Use file names that start with a letter and contain only letters, digits and the underscore character (_). File names must not contain any of the

reserved words listed in Chapter 6 of the *HP OpenView Event Correlation Services Designer's Reference*. For example: `my_file21.ecs` is a valid file name. The extension `.ecs` is supplied automatically.

Location of Circuit Files

You can save circuit files to any directory. However, if you are creating a Compound node, you must save the node to the Compound node library. See "The Compound Node Library" on page 168 in Chapter 6, "Designing Large Circuits," on page 155

Saving OVO Circuits

You can save a correlation circuit in the ECS Designer for IT/Operations at any time during the editing process. HP OpenView IT/Operations allocates a circuit name that associates the circuit file with the message source template you selected. If the circuit is saved without being verified, ECS will prompt you to verify the circuit. If verification is refused or fails, the circuit will be saved in an unverified state.

NOTE The `Circuit:Save As` menu item does not appear in the ECS Designer for HP OpenView IT/Operations.

Opening an Existing Circuit

To open an existing circuit:

1. Select `Circuit:Open` from the menu to display the file browser.
2. Enter the circuit name in the `File:` panel or select a circuit name from the directory listing.
3. Select `[OK]` to load the circuit.

   If you have an existing circuit open, ECS closes it. If the existing circuit has unsaved changes, ECS prompts you to save the circuit.

OVO

To open a circuit in the ECS Designer for HP OpenView IT/Operations, select the template from the `Message Source Templates` window, then select `Circuit` to display the associated correlation circuit.

Where to Next

When you have designed and built your event correlation circuit, the next step is to simulate and test the circuit. Testing the circuit helps you to determine whether the nodes you selected, and the conditions you specified for the nodes, provide the behavior and logic you anticipated.

To simulate and test a circuit, switch to Simulate mode in the ECS Designer. Refer to Chapter 5, “Simulating and Testing a Circuit,” on page 121.

5 Simulating and Testing a Circuit

In this chapter

This chapter provides an overview and guide to using the ECS Designer in Simulate mode. It contains:

For information about using the ECS Designer in Build mode, see Chapter 4, "Building a Circuit," on page 63.

Using the ECS Designer in Simulate Mode

The ECS Designer always starts in Build mode. When you have designed and built a correlation circuit, switch to Simulate mode to test the circuit and identify any problems or unexpected behavior.

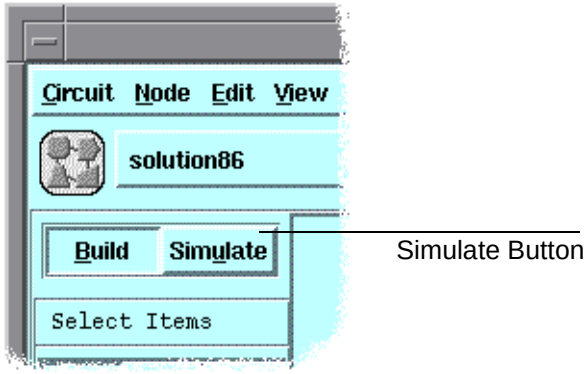
In Simulate mode, the ECS Designer uses a built-in correlation engine to perform the simulation and testing. The ECS Designer controls the engine's notion of time, allowing you to step events through a circuit and examine the state of the circuit in detail at any point.

The ECS Designer in Simulate mode allows you to load and examine events logs to feed into the circuit you are testing. You can also examine the output log.

Switching to Simulate mode

To switch to Simulate mode, select `[Simulate]` from the Build Tool Palette.

Figure 5-1 The Simulate Button



When you switch from Build mode to Simulate mode, the circuit is verified automatically and loaded into the correlation engine. If the verification is successful, the circuit appears in the ECS Designer window shown in Figure 5-2.

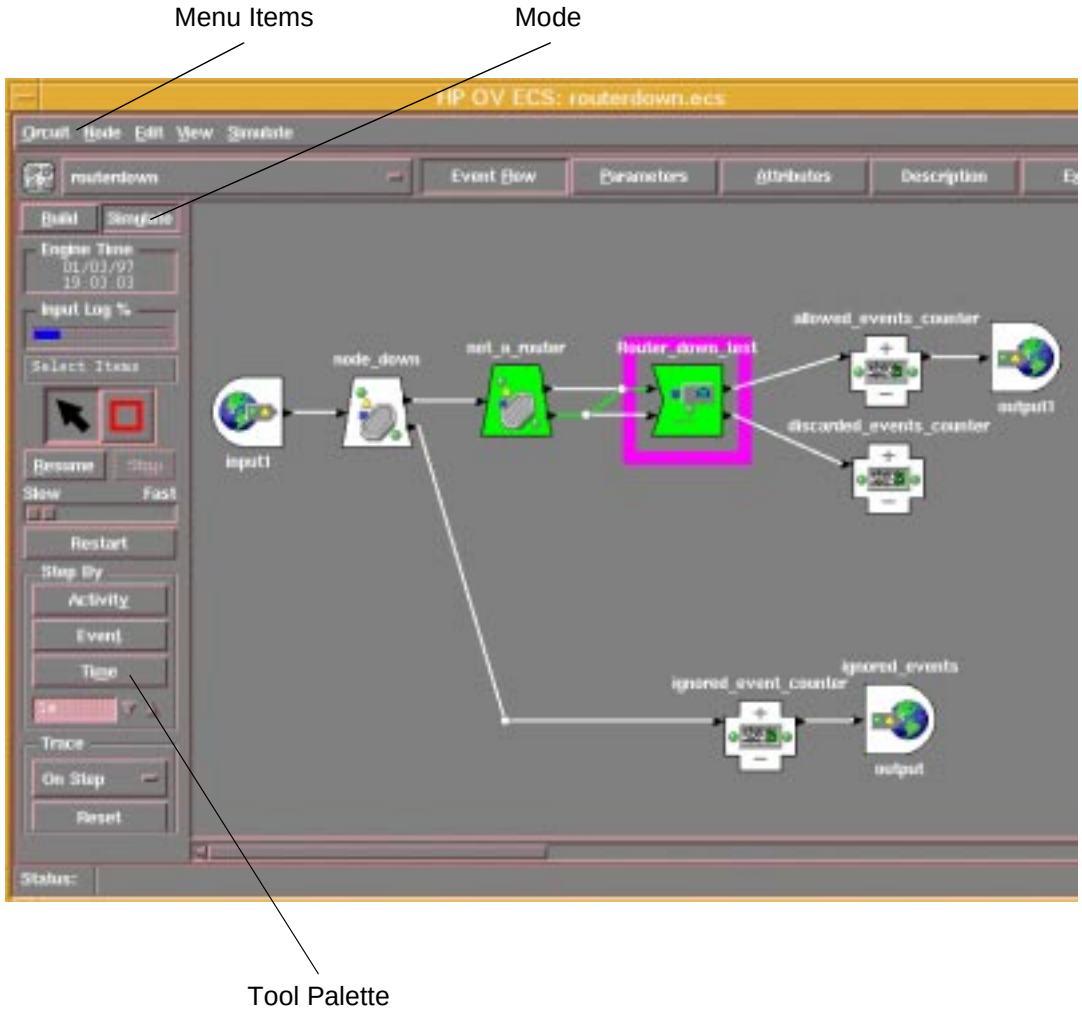
NOTE

If there is a Fact or Data Store associated with the circuit, the circuit fails to load because the Fact and Data store have not been loaded. To continue, load the Fact or Data Store. See “Loading a Fact Store or Data Store” on page 137.

If the circuit fails to load and there is no associated Fact or Data Store, select `Engine Log` from the `Simulate` menu. The `Engine Log` window contains an error message indicating the reason the circuit failed to load. See “Viewing the Engine Log” on page 152.

Despite the fact that you receive an error message, you are still able to simulate your circuit once you have loaded the Fact and/or Data Store.

Figure 5-2 The ECS Designer (Simulate Mode)



Using the Simulate Tool Palette

The Simulate tool palette contains the tools and settings you need to run and trace events through the circuit. You must set these options before you run the event log through the circuit.

The Simulate tool palette consists of two areas, described below.

Build/Simulate Toggle

This area contains the Build and Simulate buttons. The ECS Designer always starts in `Build` mode. Select `Simulate` when you have designed your circuit and you want to simulate its operation.

Tool Palette

This area contains the tools you use to run events through the circuit and manage these events:

`[Engine Time]`	Displays the correlation engine time derived from information entered in the event log file. The format of the display is *`mm/dd/yy hh:mm:ss`*. The date is in the default format defined by the operating system locale and language.
`[Input Log%]`	Provides a visual indication of the percentage of the input event log file that has been processed through the circuit.
`[Select Items]`	Selects nodes around and palette items. The `Select Items` tool and `Set Breakpoints` tool are toggle buttons. The window panel above the tool indicates which is selected.
`Set Breakpoints`	Changes the cursor to a red square. When you click on a node with this cursor, the breakpoint toggles on or off for the node.

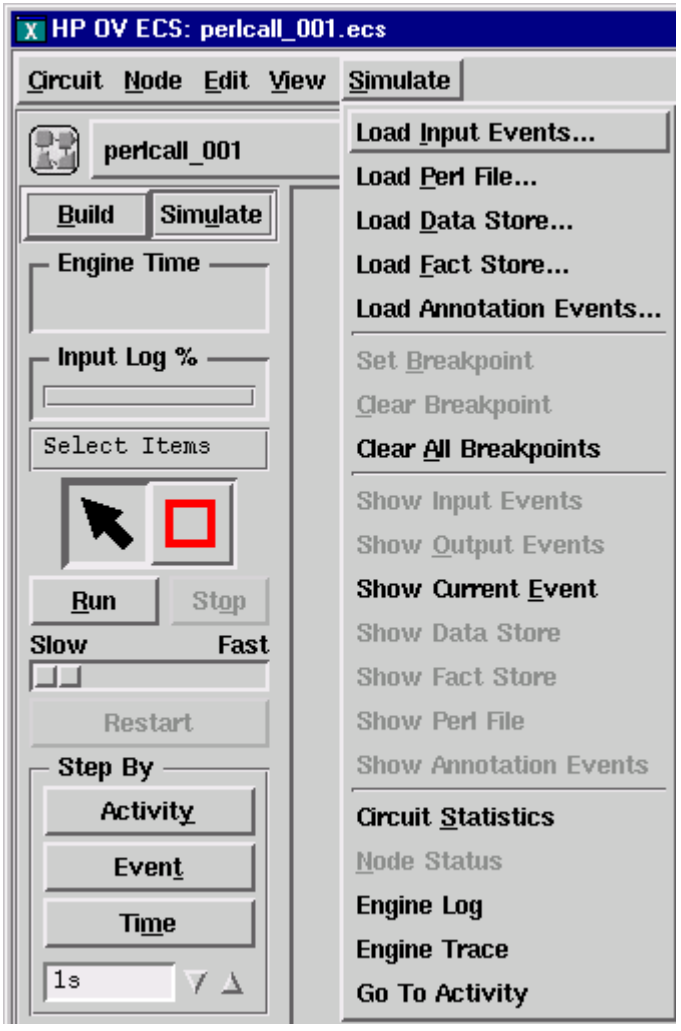
`[Run]|[Stop]`	Start or stop the processing of the event log through the circuit. When you select `[Run]`, events are processed until you select `[Stop]`, or until all events have been processed from the event log. If a breakpoint is set, processing will only continue until the breakpoint is reached (regardless whether the breakpoint is on an event or a node). When all events have been processed, ECS displays the message: `No more events left to process`. The `[Run]` button changes to `[Resume]` when selected.
`[Slow-Fast]`	Provides a slider that controls the rate the events pass through the circuit. You can adjust this slider during the simulation.
`[Restart]`	Purges all existing events stored in memory within nodes, and purges all events from the output log.
`Step by - [Activity]`	Steps an event until it reaches the input port of the next node. If there is no activity for the current event at the current time, the simulation time is advanced by single seconds until either an activity occurs or the next event is due. `Step by – [Activity]` continues advancing time and inputting events until: • An activity occurs. • The input log has been emptied. • A breakpoint is reached.
`Step by - [Event]`	Steps a single event through the circuit.

`Step by - [Time]`	Advances the simulation time by the duration defined in the time window. ECS displays all activity in the current view that occurs within the defined period. You can increment the simulation time in steps of one second or more. ECS displays any activity that occurs before the time advances to the next period. Any events within the input log that have a time stamp within the defined period are fed in to the circuit at the appropriate time. You can use `Step by – [Time]` to advance the simulation time beyond the entry time of the last event in the input log. This is useful if events are delayed within the circuit and you want to see how they are processed.
`Step by – time window`	Increases or decreases the time step. Click on the arrow buttons at the side of the time window to increase or decrease the time step. Alternatively, you can enter a duration directly in to the time window.
`Trace`	Displays the tracing options (see Figure 5-7 on page 143). Tracing provides a visual indication of the path the event takes through the circuit. As the event moves through the circuit and enters nodes, the path changes color and the nodes appear highlighted.
`Trace - [No Trace]`	Turns the tracing off.
`Trace - [On Step]`	Displays a trace of the passage of an event, resetting the trace for each step through the circuit.
`Trace - [On Event]`	Displays a trace of the passage of an event through your circuit, resetting the trace for each event.
`Trace - [Manual]`	Displays a trace of the passage of events through your circuit until you manually reset the trace.
`[Reset]`	Resets the trace and all trace color highlighting back to normal.

Using the Simulate Menu

Use the `Simulate` menu items to load and view input events, view related engine log files, and view a node's status. The following sections describe the active menu selections available in `Simulate` mode.

Figure 5-3 Simulate Menu



`Load Input Events...`	Loads a file of events to test your circuit. ECS displays the file browser for you to select the required event file.
`Load Perl File`	Loads the user defined Perl file(if any). ECS displays the file browser for you to select the required Perl file. Refer to *HP OV ECS Designer's Reference Guide* for Perl file formats.
`Load Data Store...`	Loads the associated Data Store (if any). ECS displays the file browser for you to select a Data Store file.
`Load Fact Store...`	Loads the associated Fact Store (if any). ECS displays the file browser for you to select a Fact Store file.
`Load Annotation Events...`	Loads an input log of annotation response events to test your circuit. ECS displays the file browser for you to select an annotation log file.

`Set Breakpoint`	Sets a breakpoint on the selected node. If you select a node with a breakpoint, this item is disabled.
`Clear Breakpoint`	Clears the breakpoint on a selected node. If you select a node with no breakpoint, this item is disabled.
`Clear All Breakpoints`	Clears all node breakpoints defined within a circuit.
`Show Input Events`	Displays the Input Log window. The Input Log displays the attributes as well as the arrival time, create time, identifier, encoding type and event type of the events within the log.
`Show Output Events`	Displays the Output Log window. The Output Log displays the attributes as well as the output time, create time, identifier, encoding type and event type of the events being output from the circuit.
`Show Current Event`	Displays the details of the event being processed through the circuit.
`Show Data Store`	Displays the contents of the Data Store file loaded with this circuit. If a Data Store file is not loaded, this item is disabled.
`Show Fact Store`	Displays the contents of the Fact Store file loaded with this circuit. If a Fact Store file is not loaded, this item is disabled.
`Show Perl File`	Displays the contents of the Perl file loaded with this circuit. If a Perl file is not loaded, this item is disabled. Refer to *HP OV ECS Designer's Reference Guide* for Perl file formats.
`Show Annotation Events`	Displays the annotation response events that have been loaded to test this circuit. If an Annotation Events file is not loaded, this item is disabled.

`Circuit Statistics`	Displays status information about the circuit.
`Node Status`	Displays the number of events that have passed through each port on the selected node, together with the transit delays and the attributes associated with the node.
`Engine Log`	Displays the Engine Log window. This window displays explicit warning and error messages, together with clues to fault resolution where possible. For example, when a Filter node refers to an attribute of an incoming event, but the incoming event does not have that attribute defined, an error message is generated by the event passing out the error output port of the Filter node (if the port is not connected).
`Engine Trace`	Displays information about the correlation engine operation including event movement, creation and deletion.
`[Go To Activity]`	Takes you to where the last activity occurred on a primitive node. This item is useful if the event has passed into a Compound node within a circuit.

Using Event Logs

To test a circuit in Simulate mode, you need to load a log of input events. This log should contain the type of events you are filtering or correlating, and the events must have the appropriate time stamps for your circuit.

See Chapter 4, “Building a Circuit,” on page 63 for information about creating an event log file.

NOTE

Event log file names must contain only ASCII characters. The file name must end with the suffix `.evt`, `.evt0`, or `.evt1`. For example, `solution_84.evt` is a valid file name.

Loading the Event Log

To load an event input log:

1. Select `Simulate:Load Input Events...` from the menu.

   The file browser displays the available input logs.

2. Select the `input` log you want to load.

To run the events through the circuit, see “Using Run” on page 145.

Viewing the Event Log

To view the contents of a loaded event input log, select `Simulate:Show Input Events` from the menu. See Figure 5-4 on page 134.

To display the events that are output from the circuit, select `Simulate:Show Output Events` from the menu. See Figure 5-9 on

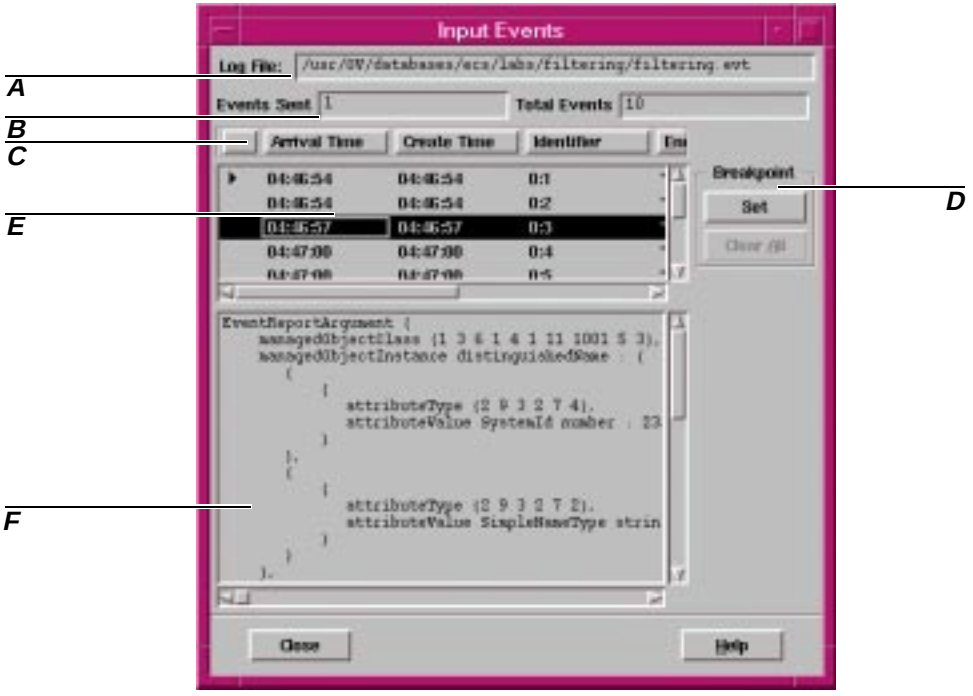
page 149.

OVO

Messages going into and out of the ECS Engine are logged in the files `ecevilg` and `ecevolg` respectively. These log files are used for debugging and testing circuits in the ECS Designer for Operations, and reside in the following directories:

- Management Server (HP-UX 10.X)
  `/var/opt/OV/log/OpC/mgmt_sv`
- Managed Node:
  — HP-UX 10.X: `/var/opt/OV/log/OpC`

Figure 5-4 Input Log Window



The Input Log window displays the contents of the event log file you have loaded. The letters refer to important parts of the window:

1. This panel displays the path and filename of the input event log you have loaded.

2. The two panels, `Events Sent` and `Total Events` display a numerical representation of the processing of the event log file. `Total Events` displays the total number of events in the input log, and `Events Sent` displays the number of these events that have been sent through the circuit.

3. This area displays the column titles for the columns of information in the main window described in item E.

   Select a title with the mouse and 'drag and drop' to rearrange the columns in a different order.

   Select the right or left edge of a title box to adjust the column width.

   The new order is maintained for the current session. When you restart the ECS Designer, the columns revert to the default settings.

4. The `Breakpoint` panel contains two buttons. Click `[Set]` to set a breakpoint on the selected event within the log. The event breakpoint displays a red square in the first column (default setting) of the event list window (see item E). If you select an event that has a breakpoint set, the `[Set]` button changes to `[Clear]`, allowing you to turn off the breakpoint for that event.

   The `[Clear All]` button clears all event breakpoints within the event log file. See "Setting Breakpoints" on page 140.

5. The main event list window displays:

   - arrival time
   - create time
   - identifier
   - encoding type
   - event syntax
   - event type.

   A red square in the first column (default setting) of this panel shows an event with a breakpoint. The arrow head indicates the next event to be sent into the circuit.

   Select an event within this panel to display the event's attributes in the window below (see item F).

   Use the horizontal and vertical scroll bars to view the information within the panel.

6. This panel displays the attributes of the event you selected in the main event list window.

Loading a Fact Store or Data Store

You must load Fact or Data Stores associated with your circuit into the correlation engine before you run an event log through the circuit.

NOTE If you load a Fact or Data Store, and then load another circuit into the correlation engine, ECS evaluates the new circuit against the Fact or Data Store currently in memory.

To load a Fact Store or Data Store:

1. Select:

   `Simulate:Load Data Store...`

   or

   `Simulate:Load Fact Store...`

   from the menu. ECS displays the file browser.

2. Select the Data Store or Fact Store file you have associated with your circuit to load the Store into the correlation engine.

ECS evaluates the circuit parameters against the parameter values within the Store.

Viewing the Fact Store or Data Store

You can only view the contents of the Fact or Data Store when the correlation engine is running an event log through the circuit.

To view the values in a loaded Fact Store or Data Store, select:

`Simulate:Show Fact Store`

or

`Simulate:Show Data Store`

from the menu. ECS displays the `Fact Store` or `Data Store` window. See Figure 5-5 on page 138.

Figure 5-5 Data Store Window

Loading and Viewing Perl files

You must load the Perl files associated with your circuit before you load the circuit into the ECS engine.

NOTE There can be only one Perl file running for each instance of the ECS engine. Repeated use results in complete reintialisation of the Perl interpreter.

To load the Perl files using the ECS Designer

a. Select `Simulate:Load Perl` from the menu. ECS displays the file browser.

b. Select the Perl file you have associated with your circuit to load the file into the correlation engine.

To view the Perl files using the ECS Designer, select

`Simulate:Show Perl` from the menu. ECS displays the `Show Perl` window displaying the contents of the Perl file.

Setting up the Simulation

Before you run an event log through the circuit, consider how you want the simulation to run:

- Do you want to set breakpoints on nodes or events?
- Do you want to run the entire log through the circuit or step events through individually?
- What trace options do you want to set?

Setting Breakpoints

A breakpoint is a location in a circuit or log file where the flow of events is interrupted.

You can set breakpoints on nodes or events. For example, you can use breakpoints to stop an event at a particular node or stop an event log after processing a certain number of events. You can then check the status of the node or log. To continue the event flow, select `[Resume]`.

Setting Breakpoints on Nodes

To set a breakpoint on a node using the Set Breakpoints tool:

1. Select the `Set Break Points` tool from the tool palette. The cursor changes to a red square.
2. Click on the node where you want to set the breakpoint.

   The node displays a red border to indicate the breakpoint.
3. Select `[Run]` to run the event log through the circuit or Compound node. The simulation stops when an event reaches the breakpoint. The node is highlighted with a purple square and the `Status` line displays `Breakpoint on node`.
4. Select `Simulate:Node Status` from the menu to display information about how the selected node processed the incoming events.

To delete the breakpoint, reselect the node using the `Set Break Points` tool. The red border is removed from the node.

You can also set and clear breakpoints using the `Simulate:Set Breakpoint` and `Simulate:Clear Breakpoint` menu items.

To clear all breakpoints on all nodes in the circuit, select `Simulate:Clear All Breakpoints`.

Setting Breakpoints on Events

To set a breakpoint on an event in the input log:

1. Select `Show Input Events...` from the `Simulate` menu to display the `Input Log` window. See Figure 5-4 on page 134.
2. Click on the event where you want to set the breakpoint. ECS highlights the event.
3. Select `[Breakpoint]` in the `Input Log` window.

   The `[Breakpoint]` button toggles to `[Reset]` so you can cancel the breakpoint. A red square is displayed in front of the event with the breakpoint set.
4. Select `[Run]` to run the event log through the circuit or Compound node. The event stops at the breakpoint. The node is highlighted with a purple square and the `Status:` line displays `Breakpoint on event`.
5. Select `Node Status` from the `Simulate` menu to display information about how the selected node processed the incoming events.

To delete the breakpoint, highlight the event in the `Input Log` window and select `[Reset]`.

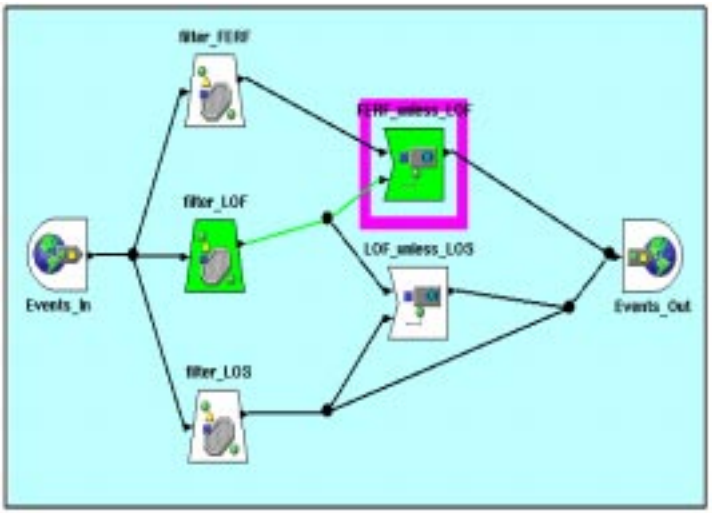
To clear all breakpoints on all events in the event log, select `[Clear All]` from the `Input Log` window.

To speed up simulation, turn tracing off. That way, events flow through the circuit at maximum speed until the breakpoint is reached.

Setting Trace Options

The trace option highlights the path of an event through the circuit. As the events pass through the circuit, the links and nodes change color to green. When the events stop at a node, the node is highlighted by a purple square.

Figure 5-6 Display of the Trace Option

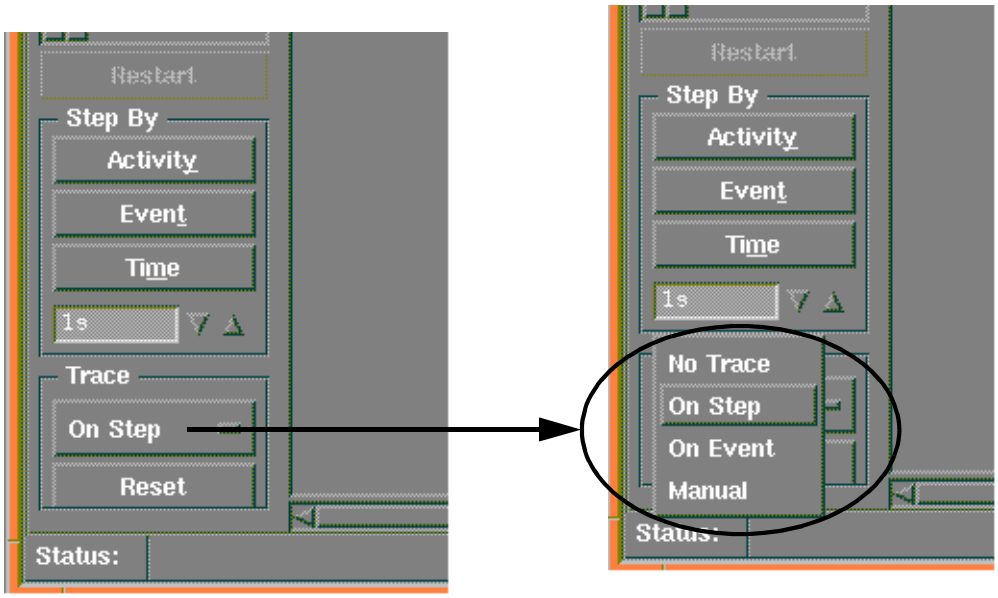


You can use any one of three trace settings:

`Trace — [On Step]`	Displays a trace of the passage of an event, resetting the trace for each step through the circuit.
`Trace — [On Event]`	Displays a trace of the passage of an event through your circuit, resetting the trace for each event.
`Trace — [Manual]`	Displays a trace of the passage of events through your circuit until you manually reset the trace.

You can also turn the trace function off. All trace options appear in the `Trace` box at the bottom of the Simulate Tool Palette.

Figure 5-7 The Expansion Button within the Trace Frame



Turning Trace Off

To turn the trace off:

1. Click on the expansion button in the Trace box. The button expands to display the options: `No Trace`, `On Step`, `On Event` and `Manual`.
2. Select `No Trace` to disable the trace.

Setting Trace on Step

Use `On Step` to trace an event and clear the highlighting before each step within the circuit.

To turn trace `On Step` on:

1. Click on the expansion button in the Trace box. The button expands to display the options: `No Trace`, `On Step`, `On Event` and `Manual`.
2. Select `On Step` to enable trace `On Step`.

Setting Trace on Event

Use `On Event` to trace an event and clear the highlighting before each new event enters the circuit.

To turn trace `On Event` on:

1. Click on the expansion button in the Trace box. The button expands to present the options: `No Trace`, `On Step`, `On Event` and `Manual`.
2. Select `On Event` to enable trace `On Event`.

Setting Trace on Manual

Use `Manual` trace to trace an event through the circuit and disable the automatic removal of highlighting.

To turn the trace on to manual:

1. Click on the expansion button in the Trace frame. The button expands to present the options: `No Trace`, `On Step`, `On Event` and `Manual`.
2. Select `Manual` to enable manual trace.

To reset the trace, select `[Reset]`.

Running the Simulation

Once you have loaded the event log, defined breakpoints, and configured trace options, you are ready to run a simulation. You can step the events through the circuit individually or you can let the correlation engine run the entire event log through the circuit. Use the Run/Resume buttons in the Simulate Tool Palette for testing your circuit.

Using Run

To run the log file through the circuit, select [Run] on the Simulate tool palette.

The correlation engine processes the log file until it meets a breakpoint, a problem, or there are no more events left. The [Run] button changes to [Resume].

To stop the event flow, select [Stop].

To continue the event flow, select [Resume] or a Step option.

To control the speed of the event flow, adjust the [Slow – Fast] slider with the mouse. You can adjust this slider at any time during the simulation.

To rerun the log file, select [Restart], then [Run] or a Step option. The correlation engine reloads the event log and runs it through the circuit.

The Input Log% progress indicator shows the percentage of the log file remaining at any time. The Engine Time box displays the time relative to the processing of events.

Figure 5-8 Display When Event Log is Empty

When the correlation engine has run the entire event log through the circuit, a dialog box displays the message! No more events left to process. Select [OK] to close the dialog box and continue.

Use Step By Time to clear any events that may be held within nodes in the circuit (see “Stepping by Time” on page 147).

Stepping Events

Use the Step By option to step events through the circuit individually by activity, by event, or by time.

Stepping by Activity

Use Step by activity to run the events through the circuit one activity (step) at a time. To step by activity:

1. Ensure you have an event log loaded. If the correlation engine has run the event log through the circuit, [Reset] the trace then [Restart] the log.
2. Select [Activity] from the Step by box in the Simulate tool palette. The correlation engine steps the event through the circuit until it reaches the input of the next primitive node.
3. Select [Activity] again to run the event through the node.
4. Select Simulate:Node Status to display information about how the selected node processed the event.

Stepping by Event

Use Step by event to run events through the circuit one event at a time.

To step by event:

1. Ensure you have an event log loaded. If the correlation engine has run the event log through the circuit, [Reset] the trace, then [Restart] the log.
2. Select [Event] from the Step by box in the Simulate tool palette. The correlation engine runs events through the circuit one at a time.
3. Select Simulate:Node Status to display information about how the selected node processed the event.
4. Select [Event] to run a new event through the circuit.

Stepping by Time

Use Step by time to identify timing problems within the circuit. Nodes such as the Table and Delay node delay events for a period of time. Use Step by time to run events through these nodes.

1. Ensure you have an event log loaded. If the correlation engine has run the event log through the circuit, [Reset] the trace, then [Restart] the log.
2. Set the time period in the window below the Time button in the Step by box on the Simulate tool palette.

   You can modify the time increment by clicking the arrow buttons or by typing over the time currently displayed in the Step By box.
3. Select [Time] from the Simulate tool palette to run all events created within that time period through the circuit.

   The correlation engine updates all nodes for the specified period, releases events as appropriate, then stops the simulation.
4. Select Simulate:Node Status to display information about how the selected node processed the event.
5. Select [Time] again to move events through the circuit for the specified time period.

Analyzing the Results of the Simulation

Use the following windows to view details about how the correlation engine processed events through the circuit and what events exited the circuit:

- Output Log window
- Node Status window
- Engine Log window
- Engine Trace window.

These windows are described below.

Viewing the Output Events

The Output Log window displays the events output from the circuit. You can save the information in this window to a file.

To display the Output Log, select `Show Output Events` from the `Simulate` menu.

NOTE

When you simulate a circuit, the output event log always contains events output by the circuit. In addition, the event log may contain events that *do not enter the circuit*. (An Output circuit policy causes events that do not enter the circuit to be transmitted directly to the output.) This can make it difficult to see how the circuit is operating. To prevent these events from cluttering the log, it may be necessary to set the circuit policy to Discard while running it in the simulator. If you do this, don't forget to set the circuit policy back to Output when you have finished.

Figure 5-9 Output Log Window

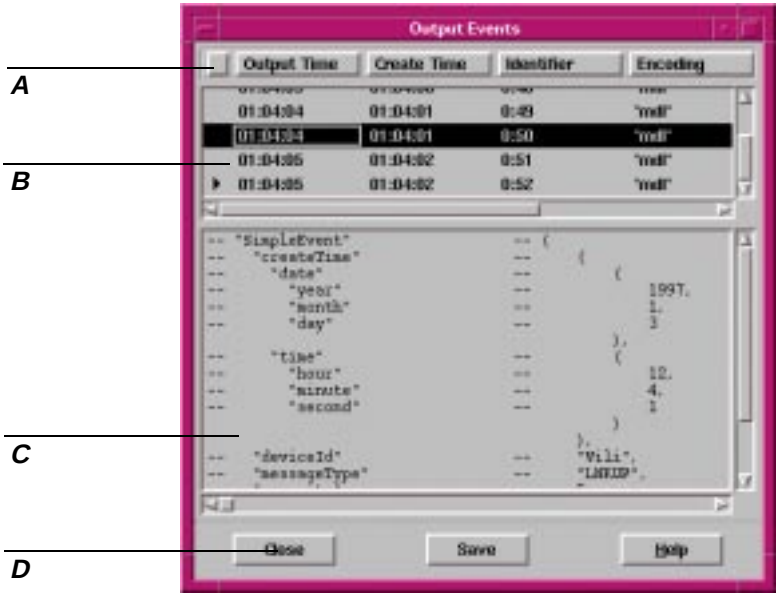


Figure 5-9 shows the Output Log. The letters refer to the following areas of the log:

1. This area displays the titles of each column in the main window described in item B.

   Select a title with the mouse and 'drag and drop' to rearrange the columns in a different order.

   Select the right or left edge of a title box to adjust the column width.

   The new order is maintained for the current session. When you restart the ECS Designer, the columns revert to the default settings.

2. The main event list window displays the following event details:

   - output time
   - create time
   - identifier
   - encoding type
   - event syntax

- event type.

The arrow head shows the last event to leave the circuit.

Select an event within this panel to display the event's attributes in the window below, see item C.

Use the horizontal and vertical scroll bars to view the information within the panel.

3. This panel displays the attributes of the event you selected in the main event list window.
4. Select [Close] to close this window.

   Use [Save] to save the content of the Output Log to a file. This is useful if you need to refer to the information later. For example, you might need it for support or to compare the results of the simulation against an earlier run. Select [Save] to display the file browser, then enter the filename to which you want to save the Output Log.

   Select [Help] to display online help specific to the window.

Viewing Node Status

The Node Status window displays information on the status of the selected node.

To display the Node Status:

1. Click on the node of interest.

   The node displays a grey border.

2. Select Simulate:Node Status from the menu.

   or

   Click the right mouse button and select Status from the popup menu.

   ECS displays the Node Status window.

Figure 5-10 Node Status Window

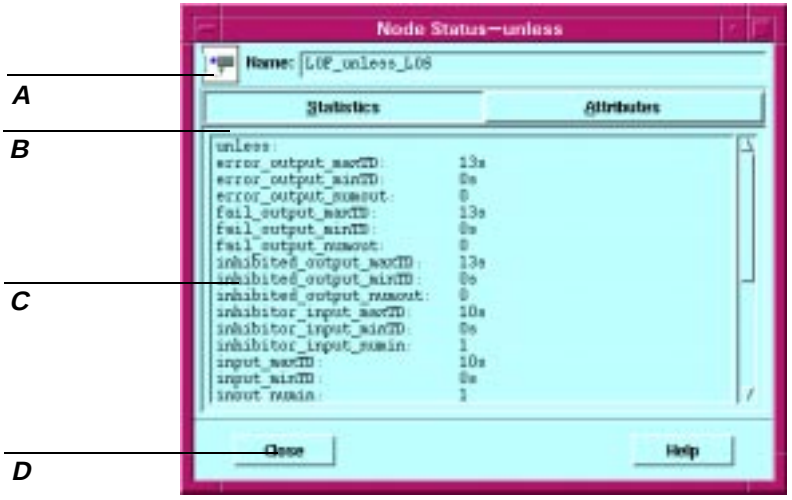


Figure 5-10 shows the Node Status window. The letters refer to important areas of the window that are described below:

1. This area displays the name of the node you have selected.
2. Use the two selection buttons, `[Statistics]` and `[Attributes]` to move between the two windows.
3. The `Statistics` window displays information about events passing through the node and the state of the node. This information includes:
   - Minimum transit delays for each port.
   - Maximum transit delays for each port.
   - Number of events entering each port.
   - Number of events exiting each port.

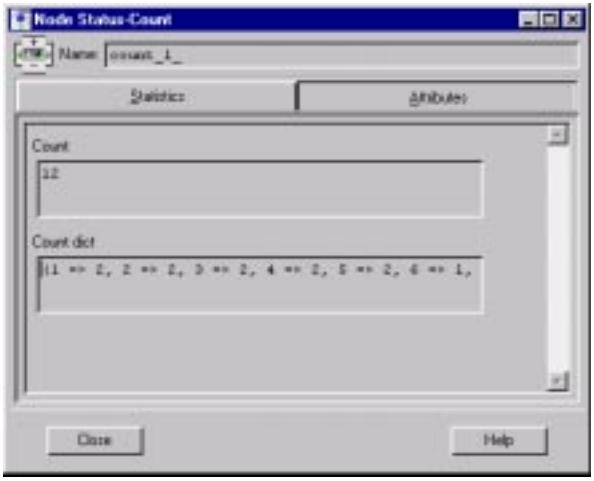
   Use the vertical scroll bars to view the information within the panel.

   The `Attributes` window displays the names and the current value of the attributes. See Figure 5-11.

   If there are no attributes associated with this node, the `Attributes` window is empty.
4. Select `[Close]` to close this window.

   Select `[Help]` to display online help specific to this window.

Figure 5-11 **Node Status Attributes Window**



Viewing the Engine Log

The Engine Log displays operational information about the correlation engine, including errors in processing within the circuit. There are four types of messages available from this log:

Log disaster These messages are usually sent to the log prior to a fatal error. They contain information about the cause of the error.

Log error These messages are sent to the log to indicate an error in processing. This can be an expression within a node evaluating incorrectly or failing to be evaluated. Processing continues, but the results may be suspect.

Log warning These messages indicate some anomaly in processing. Processing continues and is unlikely to cause a more serious error later.

Log inform These messages display information about processing within the engine. They may be useful when debugging a circuit but can usually be safely ignored.

To display the Engine Log, select `Simulate:Engine Log` from the menu.

The Engine Log window only contains information when a problem has occurred or a significant occurrence has taken place within the engine. See Figure 5-12.

Select [Update] to update the current display as events pass through the circuit.

Use [Save] to save the contents of the Engine Log to a file. This is useful if you need to refer to the information later. Select [Save] to display the file browser, then enter the filename to which you want to save the Engine Log.

Figure 5-12 Engine Log Window

All log messages are in the format:

LOG [<source_file>@<line_number>]:<message>

The ecslogmsg(5) manpage contains an explanation of the log messages.

Viewing the Engine Trace

The Engine Trace window displays the internal process of the engine. Tracing includes:

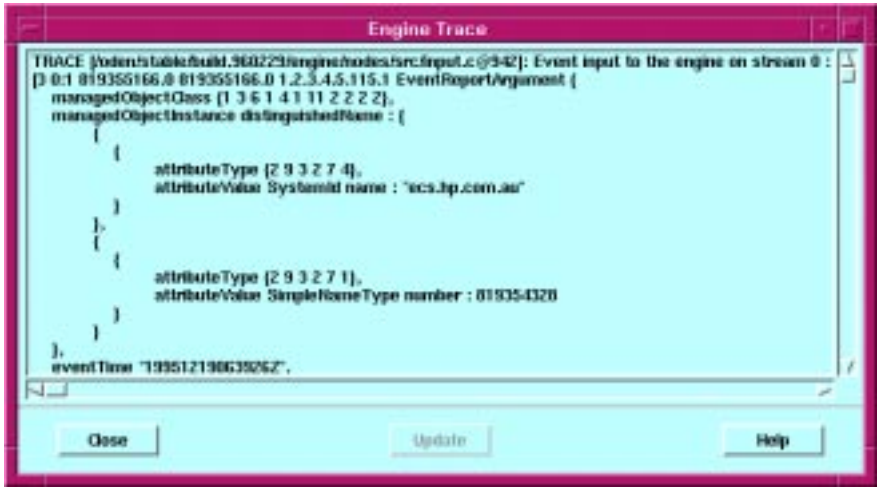
- Event movement
- Event creation and deletion
- Node processing activities.

To display the Engine Trace window, select Simulate:Engine Trace from the menu. See Figure 5-13.

Select [Update] to update the current display as events pass through the circuit.

Use [Save] to save the contents of the Engine Trace to a file. This is useful if you need to refer to the information later. Select [Save] to display the file browser, then enter the filename to which you want to save the Engine Trace.

Figure 5-13 Engine Trace Window



All trace messages are in the format:

```
TRACE [<source_file>@<line_number>]:<message>
```

6 Designing Large Circuits

In this chapter

This chapter provides an overview of how to design large circuits (using Compound nodes) that are easy to maintain and deploy. It contains:

Overview of Designing Large Circuits

The size of a correlation circuit varies depending on the problem it is designed to resolve. The ECS Designer allows you to develop and manage large circuits and then encapsulate parts of these circuits into reusable objects, called Compound nodes. This approach provides the following benefits:

- Facilitates modular *programming* of event correlation circuits. This approach makes debugging and maintenance easier.
- Permits *information containment*, so you can work at a higher level of abstraction using a Compound node. You can deal with the services offered by the node rather than the detail of the correlation circuit the node contains.
- Increases productivity by allowing you to re-use Compound nodes. You can generate a library of Compound nodes to resolve common problems.

The concept of Compound nodes is defined in Chapter 3, “Approach to Circuit Design,” on page 33.

Referenced and Local Compound Nodes

You use Compound nodes within circuits as either referenced or local nodes.

NOTE A referenced Compound node is denoted by a grey “drop shadow”. A local Compound node does not display a shadow.

Referenced Compound Nodes

Referenced Compound nodes are Compound nodes you have selected from the library and imported into a circuit. You cannot change the Compound node without breaking that reference (making the Compound node a Local Compound node).

Local Compound Nodes

A Local Compound node is specific to the circuit in which it is contained. Any changes you make to a local Compound node remain within that circuit.

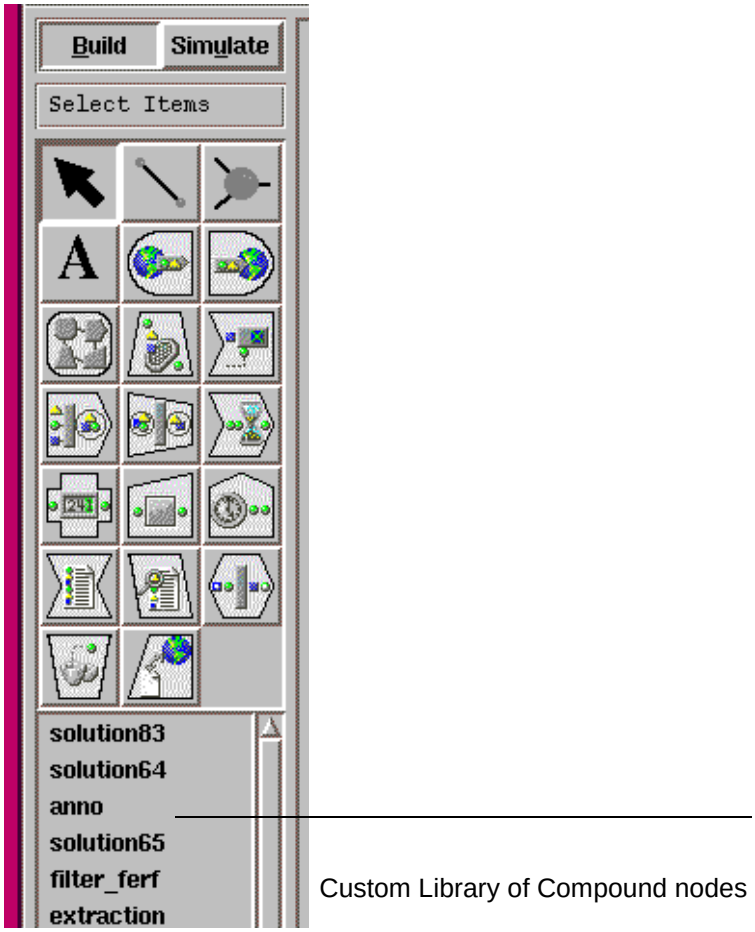
The Custom Library Palette

The ECS Designer displays a library of Compound nodes in the Custom Palette on the ECS Designer Build tool palette. The Compound node names that appear in the Custom Library palette are actually the filenames assigned to the compound nodes under UNIX or Microsoft Windows NT (see Figure 6-1). To place these nodes on the canvas, select its name from the Custom Palette and place it on the canvas.

To display a different library of Compound nodes:

1. Select `View:Change Custom Palette` from the menu to display the file browser.
2. Select the directory containing the Compound nodes you want to use. The contents of the directory are displayed in the Custom Palette.

Figure 6-1 The Custom Library Palette



Creating Compound Nodes

There are two ways to create a Compound node:

- Top-down — place a Compound node, open it, and create the correlation circuit.
- Bottom-up — design the correlation circuit you want to encapsulate, select all the nodes in the circuit, and make them a Compound node.

NOTE

If you are configuring ports for a Compound node, you can select one input and one output port to be autoconnected when you drop the node on an existing connection. ECS splits the existing connection and forms new connections; one from the starting point of the original connection to the autoconnect input port of the compound node, and another from the autoconnect output port of the compound node to the original connection's endpoint.

Creating Compound Nodes — Top-down

To create a Compound node using the top-down approach:

1. Select a Compound node from the Custom Palette and place it on the canvas.
2. Use `Node:Add Input Port` and `Node:Add Output Port` to add ports to the Compound node. A Compound node may have up to 100 ports.
3. Double-click on the `Compound` node to drill down and display the circuit (if any) on the ECS Designer canvas.
4. Build or update the correlation circuit.
5. Select `Circuit:Verify Compound` from the menu to verify the encapsulated circuit.
6. Re-display the enclosing circuit. To do this, click the right mouse button, drag to `Parent` and release.

To save the Compound node to the library:

1. Select the `Compound` node.

2. Select `Node:Save to Library` from the menu to display the file browser.
3. Enter a file name and select `[OK]`.

Figure 6-2 **Node Menu**

NOTE `Add Input Port`, `Add Output Port` and `Save to Library` can be accessed through the Short-cut Menu. To access this menu, place the mouse pointer over the compound node and depress the right mouse button.

Figure 6-3 **Short-cut Menu**

Creating Compound Nodes — Bottom-up

To create a Compound node using the bottom-up approach:

1. Select `Circuit:New` to display a blank ECS Designer canvas.
2. Place the Source and Sink nodes, and build the correlation circuit.
3. Use the mouse to select the part of the circuit that you want to encapsulate into a Compound node.
4. Select `Node:Make Compound from Selection` from the menu.

The selected circuit is replaced with a Compound node. The ECS Designer adds and connects the input and output ports automatically.

To save the Compound node to the library:

1. Select `Node:Save to Library` from the menu to display the file browser.
2. Enter a file name and select `[OK]`.

Editing Compound Nodes

To edit a Compound node:

1. Select `Circuit:Open` from the menu to display the file browser.
2. Select the `Compound` node and click on `[OK]`.

When you have changed the Compound node, select `Circuit:Save` to save it. If it is a referenced Compound node, the change is reflected in all circuits that refer to that node.

CAUTION

Changing a referenced Compound node may have a different effect on each circuit that references the node. Make sure you know the impact of the change before you make it.

Creating Parameters for Compound Nodes

Compound nodes, like primitive nodes, can have parameters and attributes.

Use the `Parameters` tab to add or delete parameters to a Compound node. You can use these parameters to configure specific nodes within the Compound node.

Entering a default parameter value is optional. When the compound node is reused in another circuit, it contains the values you set with the `default` parameter. These defaults can be overwritten. If there is no default value, you must configure parameters for the Compound node. To do this, use the `Configure` *node* window.

Figure 6-4 The Parameter Details Window

Creating a Parameter

To create a parameter for a Compound node:

1. Display the Compound node circuit on the screen.
2. Select the `Parameters` tab from within the Compound node to display the `Parameter Details` window (see Figure 6-4).
3. Enter the parameter name in the `Name` field. This parameter name appears in the `Parameter` box of the `Configure` window for the Compound node.
4. Enter the default value in the `Default` field. When this node is reused, it contains any values you set with this parameter. You can

overwrite these defaults when you configure the node.

5. Enter the data type in the Type field. You can select from the list of types or click in the box and enter a data type.
6. Enter comments in the Description field. Identify the node within the Compound node to which this parameter relates. This field is optional.
7. Select [Add] to add the details to the parameter list.

Parameter Detail Buttons

Button	Description
[Clear]	Clears the details in the Parameter Details window.
[Add]	Adds this entry to the parameter list.
[Replace]	Replaces the selected entry within the list.
[Delete]	Deletes the selected entry within the list.

Creating Attributes for Compound Nodes

Use the `Attributes` tab to create an attribute for an encapsulating Compound node.

The Count, Rate and Table nodes have attribute values that can be referenced by other nodes. If these nodes are inside a Compound node, their attributes can not be read directly. To make them available, you must configure the Compound node.

Figure 6-5 The Attribute Details Window

Creating an Attribute

To create an attribute for a Compound node:

1. Select `Attributes` from within the Compound node to display the `Attribute Details` window (see Figure 6-5).
2. Enter the name of the exported attribute in the `Exported Attribute Name` field. This name is used by other nodes to reference this attribute.
3. Enter the name of the internal node in the `Internal Node Name` field. This is the name of the internal node supplying the attribute.
4. Enter the attribute name in the `Node Attribute Name` field. This is the name of the attribute to be exported from the internal node.
5. Enter comments in the `Description` field. This field is optional.
6. Select `[Add]` to add the details to the attributes list.

Attribute Detail Buttons

Button	Description
[Clear]	Clears the details in the Attribute Details window.
[Add]	Adds this entry to the attribute list.
[Replace]	Replaces the selected entry within the list.
[Delete]	Deletes the selected entry within the list.

The Compound Node Library

The Compound node library contains a list of Compound nodes that can be referenced from within circuits.

This library is simply a directory containing ECS circuits that can be used as Compound nodes.

Importing Compound Nodes from the Library

To import a Compound node from the library into a circuit:

1. Select `Node:Import from Library` to display the file browser.
2. Select the node you want to import, then click `[OK]` to place the node on the canvas. See the node menu in Figure 6-2.

You can also select the node from the Custom Library palette. See "The Custom Library Palette" on page 158 for more information.

When you select a Compound node from a Library it is imported into your circuit as a Referenced node. When you "drill down" into the node (to display the encapsulated circuit) the status bar displays the message:

`Viewing read-only copy of library compound <compound_name>`

To change the node within the circuit, highlight the node and select `Node:Make Local` from the menu. Once you make the node local, it is specific to the circuit within which it is contained. The library node remains unchanged, but a copy is included in the circuit.

NOTE A referenced Compound node is denoted by a grey, *drop shadow*. When the node is made local, this shadow disappears.

Saving Compound Nodes to the Library

To save a Compound node to the library, select the node, then select `Node:Save to Library` from the menu. ECS displays the file browser for you to select the directory and enter the Compound node name.

If you overwrite the library node with the local node you have changed, all circuits that reference that library node will not reflect that change, until the circuit is compiled (or re-compiled).

NOTE If an error occurs in a Compound node, the circuits that reference that node will not compile.

Library Paths

The ECS Designer only loads Compound nodes from the `$OV_DB/ecs/` directory. When you create a Compound node, save it to this directory or a sub-directory.

To change the location of this directory, use the environment variable `ECSLIBPATH`

Creating Help for Compound Nodes

You use the Help system to determine the purpose of a Compound node when you retrieve it from the library, or to check on the details of how a Compound node is configured.

When you select a Compound node and press **F1**, the Help system checks to see if there is information in one or more of the following areas:

- The `Description` tab.
- The `Description` field of the Attributes tab.
- The `Description` field of the Parameters tab.
- The `Description` field of the input or output port for the Compound node's Source and Sink nodes.

If there is no information in these areas, the generic help topic for a Compound node is displayed. If information is available in one or more of the above areas, a help topic is displayed showing:

- The name of the Compound node and whether it is local or referenced.
- Any information entered in the `Description` tab.
- Any information entered in the `Description` field of the Parameters tab.
- The name of the output port.
- Any information entered in the `Description` field of the output port field for the Compound node's Sink node.
- The name of the input port.
- Any information entered in the `Description` field of the input port field for the Compound node's Source node.
- Any information entered in the `Description` field of the Attributes tab.

An example of a Compound node help topic is shown in Figure 6-6.

Figure 6-6 Compound Help Topic

To create the help topic, enter the appropriate text in one or more of the description fields. The Help system displays this text when you request context sensitive help on a Compound node.

To display the help topic you created, select the node and press [F1].

NOTE You cannot display this help topic through the main Help system index or table of contents.

Global Definitions and Compound Nodes

You can use global definitions to parameterize a correlation circuit. For example, the parameter for a Filter node within a Compound node may reference a named value that is defined in the `Global Definitions` window.

Global definitions can exist in all circuits, but are only visible in the outermost circuit. If you reuse a Compound node in another circuit, ECS ignores the global definition parameters in the Compound node and refers instead to the encapsulating circuit parameters.

Select `Circuit:Global Definitions` from the menu to display the global definitions for the circuit containing the imported Compound node.

Refer to “Defining Global Definitions” on page 83 for more information.

7 Interfacing with the Environment

In this chapter

This chapter describes the features provided by HP OpenView Communications ECS that allow you to introduce data into your event correlation circuits without having to edit and re-compile the circuit. It contains:

Overview

Data available in the environment external to the ECS Engine is contained in:

- the Fact Store
- the Data Store
- and the Annotate node.

The Data Store and Fact Store permit business, topology, and operating factors specific to a device or set of circumstances in a managed network to be separated from the general correlation rules defined by the correlation circuit. The behavior of the correlation circuit can be changed by updating the Data Store and Fact Store as local conditions change.

The Data and Fact Stores support event contexts. An event's context is the set of Data and Fact Store values in existence at the time that the event was created. Each event can have its own Data and Fact Store values, based on the event's creation time. Event contexts overcome the problems that would otherwise arise when the Data and Fact Stores are updated while events are in transit. The use of event contexts is optional. You can choose whether or not to examine the context of an event by using different data and fact store access functions.

An Annotate node may be used to obtain data from outside the ECS Engine for use within a correlation circuit. This data may be used to make correlation decisions, or it may be added to created or modified events to increase the useful information in the events.

Fact and Data Stores

Fact and Data Stores contain information you can use to parameterize circuits either during simulation, or when they are loaded into the ECS Engine. A circuit can reference Data and Fact Stores anywhere a value is specified. That is, in global values or node parameters.

NOTE

A circuit can only reference one Data and Fact Store at any one time, and one or both stores may be empty. In case a Data or Fact Store do not contain any data, they still must have the header (`# line`) in order to be referenced.

Fact and Data Stores files are loaded into memory when you load the correlation circuit. Changes to these files (including additions and deletions) can be reloaded to update the memory resident stores.

Multiple Data Stores and Fact Stores can be loaded into an ECS Engine at the same time. Any correlation circuit can be associated with any desired Data or Fact Store. This allows these Stores to be either private to a circuit or shared between circuits.

The ECS Administrator is responsible for loading the Data and Fact Stores into the ECS Engine at runtime. You can load these stores into the correlation engine when you simulate the circuit in the ECS Designer (Simulate Mode).

For statically evaluated parameters, the circuit references the Data Store when it is loaded in the ECS Engine or simulated in the ECS Designer. For dynamically evaluated parameters, the circuit references the Data Store every time the expression is evaluated.

For example, in an Unless node, a reference to a Data or Fact Store value in a `Window` parameter is statically evaluated. ECS evaluates the reference when the circuit is loaded in the ECS Engine or simulated in the ECS Designer. However, a Data or Fact Store reference in a `Condition` parameter, is evaluated each time an event arrives at the node, because the `Condition` parameter is dynamically evaluated.

When you update the Data Store while the correlation engine is running, the Unless node's dynamic parameter is affected by the change, but the static parameter is not affected.

NNM

In the ECS Engine for NNM if the value of a statically evaluated parameter is changed then the circuit is automatically reloaded so that the change takes effect.

OVO

If you use the ECS Designer for OVO, note that the Fact and Data Stores are not distributed with the message source templates (and circuits). They must be manually copied to the correct location before the templates are activated on the Management Server or Managed Node. See Chapter 8, “Packaging Circuits,” on page 195 for more information.

Using a Data Store within a Circuit

You can use the Data Store to store information such as duration, size, thresholds etc. This information is stored in name-value pairs:

```
"TableSize" = 20
"MaxRate" = 256
"Route4" = 12.3.4.15.2
```

Use the name to identify the associated data value. Names are usually (but do not have to be) String data types.

For example, if a parameter specifies:

```
input_event "fail_rate" < datastore "MaxRate"
```

then the value associated with "MaxRate" is retrieved from the Data Store when the expression is evaluated.

Creating and Updating Data Stores

The Data Store is loaded into memory from a file. You create this file as an ASCII file to which you can add or delete Store entries. An example of a Data Store file is shown in Figure 7-1.

Figure 7-1 Example of a Data Store file

The first line of the Data Store file must contain:

#path#date#version#future

where:

- # a hash sign precedes each component.
- *path* is the full path and file name of the Data Store file.
- *date* is the date the file was created.
- *version* is the Data Store file version number. Use `ecsmgr -info` to display the version number (See *HP OVC Event Correlation Services Administrator's Guide*).
- *future* is reserved for future use and must be 0.

For example, Version 1 of the Data Store file `/usr/telecom/telcodata.ds` created on August 17 1996 would have the following first line:

```
#/usr/telecom/telcodata.ds#Thu Aug 17 13:27:30 1996#1#0
```

When you create a Data Store file, use a file name suffix of `.ds` The ECS Designer only loads Data Store files with this suffix.

Precede each entry in the Data Store file with:

```
ADD DATA
```

or

```
DELETE DATA
```

These prefixes must be in uppercase.

The representation of each value or name must be valid for the data type. For example, `"MaxRate"` is a string data type that must be surrounded by quotation marks. For example, to associate the value 42 with MaxRate:

```
ADD DATA ("MaxRate" , 42)
```

Precede any comments in the file with two hyphens (--). ECS ignores all text from the start of the comment to the end of the current line.

Loading and Viewing a Data Store

The ECS Administrator is generally responsible for loading the Data Store into the ECS Engine at runtime.

Refer to the *HP OVC Event Correlation Services Administrator's Guide* for information on loading the Data Store file.

To load a Data Store in the ECS Designer in Simulate mode:

1. Select `Load Data Store...` from the `Simulate` menu to display the file browser.
2. Select the Data Store file associated with the loaded circuit.

NOTE Circuits and Data Stores are completely independent. If you load another circuit after loading the Data Store, the ECS Designer evaluates the new circuit against the current Data Store in memory.

To view the current Data Store values using the ECS Designer, select `Simulate:Show Data Store` (see Figure 7-1). The simulation must be `running` when viewing a Data Store in the ECS Designer.

Using a Fact Store within a Circuit

The Fact Store contains relationship values. These values are stored in the form: *`item1`*,*`relationship`*,*`item2`*. The relationship can be any user-defined concept, such as:

- is equal to
- is the parent of
- is contained in

The relationship is represented only as an integer value. For example, if the value 3 means *is contained in*:

`("ADM16",3,"EXCHNG2")` means "ADM16" is contained in "EXCHNG2"

The related items can be anything you need in the circuit, such as `switch5, rack17, cabinet10, circuitABC`, etc. For example: equipment 10, is contained in, rack27.

Parameters in circuit nodes can test these relationships. For example, a function within a parameter can check whether a relationship is present and return a Boolean value of `true` or `false`. The function is:

`fact_exists (leftValue, relation, rightValue)`

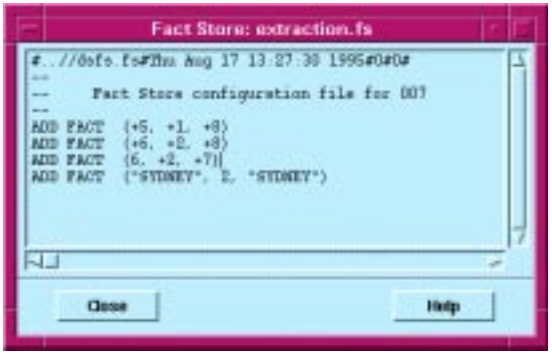
The Fact Store entries can reflect network topology information so you can determine event relationships dynamically without building network model information directly into the circuit.

It may be helpful to assign meaningful names to the integer values representing each relationship. That way, you can use named values (set as Global Definitions) in circuit parameters. (However the Fact Store itself must use the integer value.)

Creating and Updating Fact Stores

Like the Data Store, the Fact Store is loaded into memory from an ASCII file. An example of a Fact Store file is shown in Figure 7-2.

Figure 7-2 **Example of a Fact Store file**



The first line of the Fact Store file must contain the information:

#path#date#version#future

where:

- # a hash sign precedes each component
- *path* is the full path and file name of the Fact Store file
- *date* is the date the file was created
- *version* is the Fact Store file version number. ECS displays this value when you enter `ecsmgr -info`. (See *HP OVC Event Correlation Services Administrator's Guide*)
- *future* is reserved for future use and must be 0.

When creating a Fact Store file, use a file name suffix of `.fs` The ECS Designer only loads Fact Store files with this suffix.

Precede each entry in the Fact Store file with:

```
ADD FACT
```

or

```
DELETE FACT
```

These prefixes must be in uppercase.

The representation of each value or name must be valid for the data type.

Precede comments in the file with two hyphens (--). ECS ignores all text from the start of the comment to the end of the current line.

Loading and Viewing a Fact Store

The ECS Administrator is generally responsible for loading the Fact Store into the ECS Engine at runtime.

Refer to the *HP OVC Event Correlation Services Administrator's Guide* for information on loading the Fact Store file.

The simulation must be in a `reset` state (not running) when loading a Fact Store into the ECS Designer.

To load a Fact Store in the ECS Designer in Simulate mode:

1. Select `Load Fact Store...` from the `Simulate` menu to display the file browser.

2. Select the Fact Store file associated with the circuit.

NOTE Circuits and Fact Stores are completely independent. If you load another circuit after loading the Fact Store, the ECS Designer evaluates the new circuit against the current Fact Store in memory.

To view the current Fact Store values, select `Simulate:Show Fact Store` (see Figure 7-2). The simulation must be `running` when viewing a Fact Store in the ECS Designer.

Event Contexts

Sometimes, the simple view of Data and Fact stores that we have presented so far in this chapter is not sufficient. Consider a correlation circuit that creates a warning alarm when a security violation occurs, but suppresses all other events. The only output from the circuit should be security alarms.

Now let's say that the user ID of authorized users (those for whom an alarm is NOT raised) is stored in a Data Store. Authorizations change constantly and the Data Store is therefore updated regularly by an automated process.

In a busy network it would be possible for events to be delayed until after the Data Store has been updated, resulting in incorrect authorizations or unnecessary alarms being generated.

The solution to this problem is to use event contexts and Data and Fact Store versioning to ensure that events are processed in the context in which they are created.

An event is created at a particular time as a result of a state change in some managed object. The real-world entities represented by data store and fact store entries have particular values at that time. In an ideal situation, the store entries are changed to reflect the new real-world conditions whenever these conditions change. If event-emitting objects and associated or related systems and services are automatically reconfigured as a result of the state change, the data and fact entries may be changed automatically.

In order to interpret the event, it may be necessary to know the context that existed at the time the event was created. This requires that the Data and Fact Store entries which existed at that time be available, even

if the entries have been subsequently updated. In other words, there must be multiple versions of the facts and data available, and it must be possible to access the appropriate context.

A data or fact store entry is accessed as either the most recently updated version of an entry (using the `current context`) or the version of the entry that existed at the creation time of the event (using the `event context`). ECDL operations are provided to access the current or the event context.

To use the `current context` you access the Data and Fact Stores as previously described (using `datastore`, `fact_exists`, `fact_find_left`, and `fact_find_right`).

To use `event context` you must satisfy two requirements:

- You must enhance your circuit to specify the event context to be used.
- You must add versioning information to the Data and Fact Store files.

These requirements are discussed in the next two sections.

Using Event Contexts in a Circuit To use event contexts when accessing a Data or Fact Store in a circuit you must specify the event whose context is to be used, at the point where the value is retrieved from the datastore. For example, to write a Filter node expression that uses the event context of the input event to retrieve the “critical” value from the datastore, in context, you would write:

```
input_event "severity"=input_event data_store "critical"
```

The term `input_event data_store` returns the version of the data store entry associated with the input event’s context. Then the value for the key `"critical"` is obtained from this context. So the final result is the value keyed to the `"critical"` Data Store entry, at the time `input_event` was created.

Fact Store contexts are accessed a little differently. Instead of just providing the event whose context is to be used, a different set of functions is provided for fact store contexts:

```
fact_exists_ctxt (input_event fact_store)
 (left_fact, relationship, right_fact)

fact_find_left_ctxt (input_event fact_store)
 (right_fact, relationship)

fact_find_right_ctxt (input_event fact_store)
```

```
(left_fact, relationship)
```

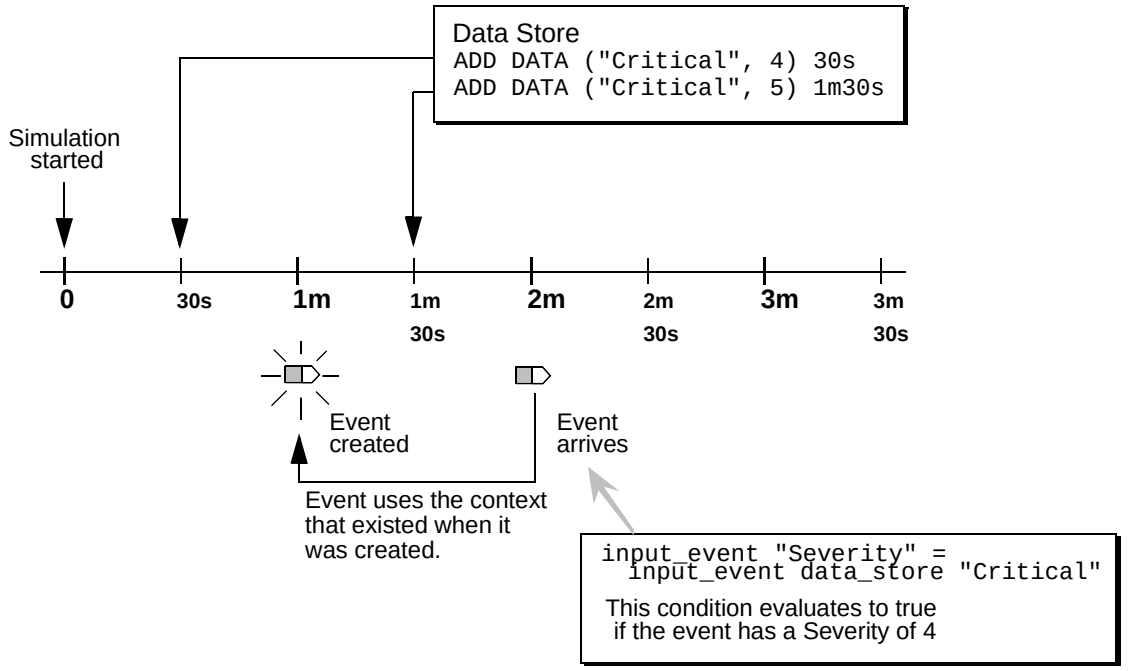
Data and Fact Store Versioning The ECS Runtime Engine requires no special action to support Data and Fact Store versioning: each time a change is made to a value it is time-stamped, and the old value is retained until no longer needed (until no event in the engine has that context and no future event could require that context, based on the configured Transit Delays for the circuit).

However, in the ECS Designer in Simulate mode, to simulate event contexts you must ensure that Data and Fact Store entries are marked with version information. Versioning information is provided after the Data or Fact Store entry, as a Duration since the starting time of the simulation. The starting time of the simulation is the creation time of the first event in the input event log. For example, a Data Store might contain the following entries for the "Critical" value:

```
ADD DATA ("Critical", 4) 30s
ADD DATA ("Critical", 5) 1m30s
DELETE DATA("Critical") 2m
```

This sequence of entries creates a Data Store entry with the key "Critical" 30s after the beginning of the simulation, the value is updated 1m 30s after the beginning of the simulation, and is deleted 2m after the beginning of the simulation. If the Transit Delay set on the circuit's External tab is 3m, and an event 1m old arrives 2m after the start of the simulation and uses the event's context to access the "Critical" Data store value, then the value returned is 4.

Figure 7-3 Data and Fact Store Versioning Timeline



The Annotate Node

The Annotate node is a primitive node used to obtain data from outside the ECS Engine for use within the correlation circuit. This data may be used to make correlation decisions, or it may be added to events to increase their useful information content.

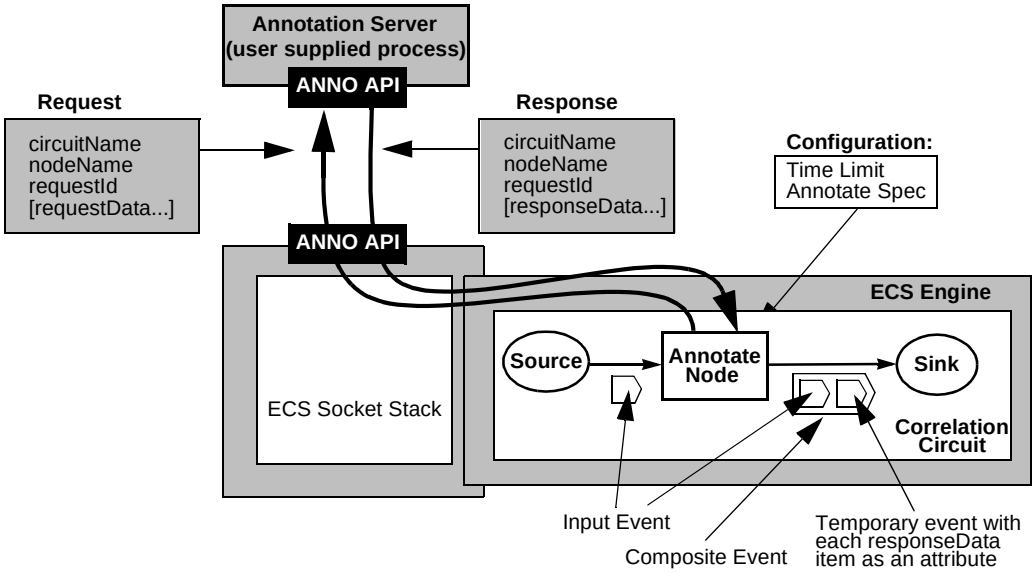
You must create an external annotation server to service the annotation requests generated by the Annotate node. The server is not part of the HP OpenView Communications ECS product.

When an event enters the Annotate node Input port, it causes the ECS Engine to generate an `annotation request` which is transmitted to an external annotation server which is registered to receive it. The request is then examined and an `annotation response` is formed. Once the response is sent, the ECS Engine decodes the response to determine the request which corresponds to it, and forwards its contents to the appropriate node.

Any required data may be returned (subject to the limits of the protocol). Most ECS data types are supported (string, integer, real, time, duration, Boolean, list, tuple, etc.), including any combination of these types. The data in both the request and the response is specified as a List data type.

The response event is returned directly to the requesting Annotate node. The response does not enter through any of the circuit's Source nodes. A composite event is created containing the input event and a temporary event that contains the data returned by the Annotation Request.

Figure 7-4 The Annotation Mechanism



Example source code for an annotation server is provided. This example opens a connection using the `ANNO I/O` and receives annotation requests. Using the information they contain, the example annotation server generates an annotation response which is sent back to the ECS Engine.

The example code is intended as a starting point for developers to create their own Annotation Servers, especially if the events are going to contain complex data structures that must be decoded/encoded.

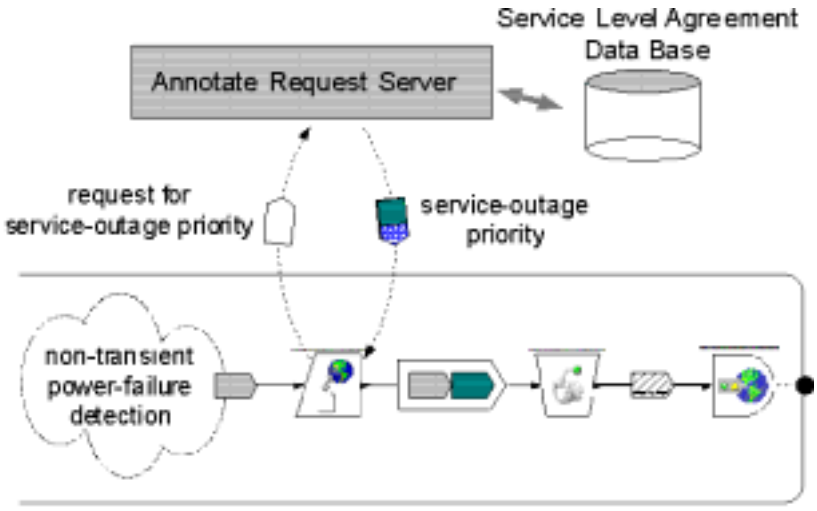
For a full description of the Annotate node, including information on where to find the example source code, refer to the *HP OVC Event Correlation Services Designer's Reference*.

Using the Annotate Node

The following example illustrates how an Annotate node may be used within a circuit. The actual parameters specified depend on how you define the Annotation Server.

In a scenario where a circuit is designed to detect a non-transient power failure, the power-failure event is sent to the input port of the Annotate node. This event contains an event attribute that indicates the device where the power failure has occurred.

Figure 7-5 Example Use of an Annotate Node



In this example, the correlation circuit needs to determine the severity of the power failure, by referring to a Service Level Agreement Database outside of ECS. A warning event of the correct priority can then be created. The Annotate Request Server is given the name of device that has failed, it then consults the database, and determines the criticality of the power failure.

The power-failure event arriving at the Annotate node's input port causes the ECS Engine to generate an `annotation request` event. The value of the Annotate node's `Annotate spec` is automatically encoded into the `annotation request` event.

To provide the Annotate Request Server the name of the device that has failed, the Annotate node's `Annotate spec` would refer to the device name attribute of the input event:

```
[input_event "device"]
```

The `annotation request` event is transmitted to the Annotate Request Server.

The Annotate node waits for the duration specified in the `Time limit` parameter for a response. In this example, an annotation response event is received from the Annotate Request Server within the defined duration.

The ECS Engine uses the attribute values from the annotation response event to determine the Annotate node and input event to which the annotation response event relates. (There can be a number of Annotate nodes in a circuit, and each Annotate node may have a number of annotation requests outstanding.)

The relevant Annotate node creates a composite event containing the relevant input event and a temporary event holding the request response data.

The composite event is sent to the Annotate node's output port, connected to a Create node. See Figure 7-5.

The information in the composite event is used by the Create node to create a new event, containing the correct priority information.

Testing the Annotate Node within a Correlation Circuit

A circuit using an Annotate node can be tested using the ECS Designer in Simulate mode. As you cannot communicate with the annotation server using the ECS Designer (the annotation server only communicates with the ECS Engine), you have to simulate annotation requests using event logs.

To simulate the annotation and test your Annotate node, you can use one of the following ways:

- Run the circuit in the ECS Engine connected to the real annotation server and collect the input log. This log is then used as both the input and annotation log. When used as an input log only the events are read (the annotation responses are ignored), and when used as the annotation log only the annotation responses are read (the events are ignored).
- Run the circuit in the ECS Designer's simulation mode using an input log designed for testing the circuit (this can be either hand crafted or an event log from a running ECS Engine). Capture the annotation requests and manually alter them to be annotation responses. You can now simulate the annotation circuit in the ECS Designer.

  The first approach is more appropriate when you are trying to diagnose a circuit design problem with an existing annotation server. The second approach is generally preferred because it is quicker and simpler, and does not require the ECS Engine or an annotation

server. Both approaches are described in detail in the following sections.

Generating an Annotation Log using the ECS Engine

1. Using the ECS Designer, design and build a circuit that performs annotation. In this example, the circuit is called `anno.ecs`.
2. Select `Circuit:Compile` from the ECS Designer menu to convert the circuit to a form that can be loaded by the engine. The compiled circuit is called `anno.eco`.
3. Start an ECS Engine and load the circuit, naming it "simulate". You *must* give it the name "simulate" if you want the resulting logs to be usable by the ECS Designer. For example:

   ```
   ecsd
   ecsmgr -circuit_load simulate anno.eco
   ```

4. Enable input logging on the circuit and enable the circuit. For example:

   ```
   ecsmgr -log_events_in on
   ecsmgr -enable simulate
   ```

5. Start your annotation server(s).
6. Fire events at the ECS Engine – either live events or an event log sent using the `ecsevgen` utility.
7. When your input events have finished, and you have received all of your annotation responses, shut down the ECS Engine and your input and annotation event log is in the `$OV_LOG/ecs/1/ecsin.evt0` log file just generated.
8. Rename the event log to `ecsin.evt` so you can read it in the ECS Designer.
9. Start the ECS Designer and load the circuit.
10. Switch to Simulate mode by selecting the `[Simulate]` button.
11. Load the `ecsin.evt` log as the input log by selecting `Simulate:Load Input Events`. Inspection should show that it contains only normal events, not the annotation responses.
12. Load the `ecsin.evt` log as your annotation log by selecting `Simulate:Load Annotation Events`. Inspection should show that it contains only annotation responses.

13. Start the simulation by selecting [Run]. The input events should enter the circuit and generate annotation requests. These requests will match the appropriate responses out of the annotation responses log, but they will not enter the circuit until the transit delay has expired.
14. If annotation responses arrive after the last event in the input log, select the Step-by [Time] button to see the responses enter the circuit. You may find it convenient to set a breakpoint on the Annotate node and set a large value for the time.

Generating an Annotation Response Log using the ECS Designer

It is frequently quicker and easier to use the ECS Designer to generate annotation requests, and to hand craft these into annotation responses.

The great advantage of this approach is that you can develop and simulate circuit containing Annotate nodes before building the annotation server. Once you know how the circuit and the annotation server *should* operate, you can construct the annotation server to provide the responses required.

1. Using the ECS Designer, design and build a circuit.
2. Switch to Simulate mode by selecting the [Simulate] button.
3. Load input events by selecting `Simulate:Load Input Events` from the menu bar.
4. Run a simulation using these input events by selecting the [Run] button. The circuit generates annotation requests, though there are no annotation responses to match them.
5. Save the output log by selecting `Simulate:Show Output Events-> Save`. The output log contains the annotation requests.
6. Using a text editor, change all occurrences of `% anno:request:` to `% anno:response:`. This will change the requests to matching responses. Now you can hand craft the annotation responses to match what you expect your annotation server to reply. If you want to simulate transit delays between the request and response then add the number of seconds delay to the end of the line. For example, to simulate a 12 second transit delay before the response:

   ```
   % anno:response:12
   ```

   Save the response log.

7. Reset the simulation by selecting the [Reset] button, and load the response log by selecting Simulate:Load Annotation Events.
8. Start the simulation by selecting the [Run] button. The input events should enter the circuit and generate annotation requests. These requests will match the appropriate responses out of the annotation responses log, but the responses will not enter the circuit until the transit delay has expired.
9. If annotation responses arrive after the last event in the input log, select the Step-by [Time] button to see the responses enter the circuit. You may find it convenient to set a breakpoint on the Annotate node and set a large value for the time.

Drill Down

The process of extracting correlating events from a corrleted event is called drill down. The drill information for an event flowing through a circuit will be specified by the circuit designer within the node's condition parameter. At the output of an event from the circuit, the engine will log this information in either a specified Correlation log or the default Correlation log. The events refered from within the drill information will also be logged seperately on a DrillEventLog. DrillEvent log can be either an application specified log or a default log.The Correlation log will have the complete drill information of an event logged as a tree for example

```
e1:e2,relation_string:.......::e5,relation_string
```

where `e1,e2 ...` are the unique event-ids generated by the ECS engine during the event flow. e1 represents the event whose drill information is followed by the first ':'. A correlating event of e1 and its relationship with e1 form a correlation tuple seperated by ','.The corrleation tuples are seperated by ':'. The drill-up and drill-down information are seperated by '::'. In the specified drill-info format the drill information will follow the drill-down information.

In the above example, e1 is the correlated output event whose drill information follows after the first ':'. There may be no drill information for e1 which will mean that e1 is output without being correlated or the circuit designer did not choose to put any drill information for this event . The tuples after '::' specify drill up information . For example , e1 is a correlating event of e5 and any other events that follow.

8 Packaging Circuits

In this chapter

This chapter describes the process of packaging circuits and preparing them for use within the ECS Engine. It contains:

NOTE

Correlation circuits developed for use with HP OpenView Network Node Manager (NNM) must be specially packaged before they are deployed. Please refer to the separate document *HP OpenView ECS Designing Correlations for NNM*.

Preparing the Circuit

You must compile the circuit and package the associated files before you send them to the ECS Administrator to load into the ECS Engine. The files required by the ECS Administrator are:

- circuit file (`.eco`)
- Data Store file (`.ds`)
- Fact Store file (`.fs`).

OVO

Some change here In ECS for OVO, correlation circuits are distributed as part of the Event Correlation (EC) template distribution. The Data and Fact Stores are not distributed with the EC template (and correlation circuit). You must copy these files to the following locations:

HP-UX 10.X	`/var/opt/OV/conf/OpC/mgmt_sv/dstore.ds`
	`/var/opt/OV/conf/OpC/mgmt_sv/fstore.fs`

You must copy the Data and/or Fact Stores before the EC templates are activated on the Management Server or Managed Node.

Compiling the Circuit File

Before you compile the circuit file, ensure the circuit has been simulated in the ECS Designer (Simulate mode) with the appropriate log files. See Chapter 5, “Simulating and Testing a Circuit,” on page 121.

OVO

`Compile` is not applicable to ECS for OVO. These circuits are automatically compiled when you save them.

When you have tested the circuit, compile the circuit file:

1. Select `Circuit:Compile...` from the menu to display the file browser.
2. Enter the circuit name (including path) in the `File` panel.

   Do not add a suffix as `.eco` is automatically appended. The circuit file is written to wherever you chose in the file browser.

NOTE Circuit file names must start with a letter; can only contain letters, digits, and the underscore character (_); and cannot be an ECDL reserved word. For example, `test_84.eco` is a valid file name; `84-test.eco` is invalid because it starts with a digit; and `case.eco` is invalid because it uses an ECDL reserved word.

3. Select `[OK]` to display the `Compile Circuit` window.

   If the verification is successful, the `Compile Circuit` window closes automatically. The `status` bar in the main ECS Designer window displays `Verification Success`.

   If an error occurred, the `Compile Circuit` window remains open and indicates the name of the node in error together with the error that has occurred. To go to the node in error, select `[Goto]`.

   The node in error appears on the canvas window highlighted in red. Select `Configure` and correct the condition.

To close the `Compile Circuit` window, select `[OK]`.

Pass the circuit file to the ECS Administrator to load into the ECS Engine.

Preparing the Data Store and Fact Store Files

The Data Store and Fact Store files must be loaded into the ECS Engine. Pass these files to the ECS Administrator, together with the circuit file, for deployment.

If you designed and created a circuit that references non-existent Data Store or Fact Store entries, the circuit verifies and compiles correctly. However, when the circuit is run within the ECS Engine, the engine writes an exception code 6, NotFound error, to the Engine Log.

The ECS Administrator may ask you some questions relating to the Data Store and Fact Store files. Be prepared to answer the following questions, especially if you are updating existing files:

- Are there any statically evaluated parameters that must be re-evaluated?
- Are there any dependencies between the Data Store and Fact Store updates?

- Can large Data Store and Fact Store files be split into a number of smaller files?
- Can Data Store and Fact Store files be combined into one file?

If a circuit references a Data Store or Fact Store that has not been loaded, the ECS Engine writes the following message to the engine log file:

```
LOG [/engine/nodes/src/loader.c@2582]: Error processing Circuit
File /tmp/FAAa11829 for circuit simulate: exception during
initialization: NotFound
```

The engine also writes the following message to the engine trace file (if enabled):

```
TRACE [/engine/interpreter/src/interpreter.c@3675]: Exception
NotFound in built in function dataStore

TRACE [interpreter]: ECDL location is
                              “untitled_module.table_1_ max_events"
```

You can handle these exceptions in ECDL. Refer to the *HP OpenView Event Correlation Services Designer's Reference* for more information on Handling Exceptions.

9 Troubleshooting

In this chapter

This chapter provides directions for troubleshooting the ECS Designer and contains:

NOTE

For troubleshooting information related specifically to ECS correlation circuits in the HP OpenView NNM environment, refer to the troubleshooting guide in the NNM online help, and the separate publication *HP OpenView ECS Designing Correlations for NNM*.

Eliminating Common Faults

Useful information

The following sources of information describe late changes in the product and can help you resolve common faults:

- release notes in `$OV_MAIN_PATH/`
- *ecsdes*(1m) manpage for information about the `ecsdes` command.
- *ecslogmsg*(5) manpage for an explanation of the messages sent to the Engine Log window.

Checklist of faults

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

The ECS Designer does not run.

❏ Is a license available?

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

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

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

— Did a demonstration or evaluation license time out?

— Did you attempt to restart the ECS Designer too quickly?

The license server needs time to *clean up* if an ECS Designer session stops abnormally.

For more information about licensing, see Chapter 4, “Setting Up Network Licensing” in the *HP OpenView Event Correlation Services Installation Guide*.

❏ Do executables fail to run and are manpages unavailable?

— Make sure `$OV_BIN/` is included in the `PATH` variable.

— Make sure `$OV_MAN/` is included in the `MANPATH` variable (UNIX only).

In some operating systems, the UNIX `man` command does not search the default directories when the `MANPATH` variable is

defined explicitly—see *man*(1). In these cases, include the default `man` paths in your `MANPATH` variable.

Common Problems in the ECS Designer

Use the following list of suggestions to resolve common problems in the ECS Designer:

- Check the Engine Log and Engine Trace windows if you are unsure why a circuit verification failed when you switch to Simulate mode. A message is often written to the Engine Log file.
- Refer to the `ecslogmsg(5)` manpage if you are unsure about the meaning of a message in the Engine Log window.
- Check the properties of files and directories if you have difficulty loading, or finding, a log file or circuit.

Analyzing the Problem

Any change relating to CORBA If the suggestions above are not the cause of the problem, consider the following:

❑ 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, associated software, ECS Engine, correlation circuits, or Data and Fact Store files?

— Have there been any hardware changes to the host system or to the topology of the managing network or managed network?

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

❑ What is the time of occurrence, duration, and 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 occurrence 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.

Troubleshooting OVO Problems

There are a number of options you can choose when investigating event correlation-related problems in the context of IT/Operations. The problems you encounter can concern the malfunction of either the correlation circuits and templates you design and distribute or the ECS Engine itself. First of all you need to establish whether the following conditions are true:

- MSI output must be enabled on the management server and the managed node. Enabling MSI output makes only sense on nodes where correlation should work. You enable MSI output by opening the following sequence of windows:

Management Server	`Actions->Server->Configure->Enable MSI Output`
Managed Node	`Node Bank->Modify Node->Node Advanced Options:MSI – Enable Output`

- Output to the MSI must be enabled and the option `Divert/Copy Messages` set for each of the message conditions that output messages to the event-correlation engine. You do this by opening the following sequence of windows: `Node Bank->Message Source Templates->message and Suppress Conditions->Condition Number # ->Advanced Options:MSI – Enable Output`
- The `message-type` attribute must be set correctly for each message condition. The ITO administrator defines message types: a message type is usually the name of the sub-group to which the message belongs. You set the `message-type` attribute by opening the following sequence of windows: `Node Bank->Message Source Templates->Message and Suppress Conditions->Condition Number # :Message Type`
- The `event_type` field in OVO must be filled with a *string*, and *thus quotation marks must be used*.
- Minimum and maximum transit delays (`Min/Max TD`) can cause a message not to be processed, although everything else is set correctly.
- The message attribute you specify in the `message-type` attribute field for a given message condition must match the message attribute

defined in the `event_type` field of the appropriate event correlation circuit input port. Where there are several input ports for a circuit, set the input port for the part of the circuit that you want to process the `message-type`. All other messages will then flow through the other circuit input ports.

If all these conditions are true, then the message should pass through the correlation process as expected. However, if you want the message to be discarded during the correlation process, then verifying whether or not the event-correlation process is running correctly is more difficult.

The easiest way to find out whether the message is being generated by the template in the first place (since if it were indeed being generated by the template but then correctly discarded by the event-correlation process, it would not appear in the `Message Browser` window) is either to disable the output to the MSI so that the message passes directly to the Message Browser, or set the `Divert/Copy` message to MSI switch to `Copy`.

The result of this is that either the original message or a copy of it will appear in the `Message Browser` window. If the message does appear in the window when the output to the MSI is switched off, you can concentrate further investigation on the event-correlation process itself.

NOTE Make sure that you redistribute the event correlation template after disabling (or re-enabling) output to the MSI and after toggling the `Divert/Copy message to MSI` switch.

Reporting Problems

Inquire about HP support from the HP sales office that supplied HP OpenView Event Correlation Services. The form below is a guideline only. Send the completed form by facsimile, or edit the `$OV_RELNOTES/ecs_prob_report` file to send by e-mail. See the `$OV_RELNOTES` release notes or your support contract for contact details.

Any ORB related changes in the product part of table

Originator details

Problem Summary:			
Support Contract:			
Your Reference:		Date	
HP Reference:		Date	
Report Type	[ ] Problem Report		
	[ ] Enhancement Request		
	[ ] User Comment		
Product	[ ] ECS Engine		
	[ ] ECS Designer		
	[ ] HP OpenView DM installed?		
	[ ] HP OpenView OVO installed?		
Submitter:			
Name:			
Job Title:			
Telephone:		Time zone:	
Facsimile:			
Email:			
Web:			

Organization:				
Company:				
Address	Street			
	City			
	State		Postcode/Zip	
	Country			
Mail	P.O. box			
	City			
	State		Postcode/Zip	
	Country			

Problem Description

Sequence	*Describe how the product was being used before the appearance of the problem (attach pages as required).*

Symptoms	*Describe how the problem first appears and its continuing behavior.*

Severity	1-5: _____	*(1 = low, 5 = high)*
Priority	1-5: _____	*(1 = low, 5 = high)*
Show Stopper:	Y/N: _____	

Describe how the problem affects your use of ECS:

Reproducible: Y/N: _____

Procedure to follow to reproduce the problem:

Supporting Data

Version	[ ] ECS Engine/DM/OVO:			
		(*Type*: "**`ecsmgr -info`**")		
	Patch number:		Date	
			Date	
			Date	
	[ ] ECS Designer			
		(*Select*: Help->About HP OV ECS Designer)		
	Patch number:		Date	
			Date	
			Date	
Platform	Manufacturer:		Model:	
	Clock speed		Memory:	
	Operating system		Version:	
	HP OpenView Products:		Version:	

Files captured

Please collect as much of the following data as possible and be ready to supply it on request. All information supplied is treated in the strictest confidence, and is not used for any purpose other than diagnosis without your written permission.

Core dump	[ ]	`core`*`file`*
Engine status	[ ]	*Type*: **`ecsmgr -info > filename`**
Engine snapshot	[ ]	*Type*: **`ecsmgr-snapshot> filename`**
Engine statistics	[ ]	*Type*: **`ecsmgr -stats > filename`**

ECS circuit	[ ]	`*.eco` *compiled file*
	[ ]	`*.ecs` *uncompiled file*
Data store file	[ ]	`*.ds` *file*
Fact store file	[ ]	`*.fs` *file*
MIBs for events	[ ]	`*.mib` *files*
MDL descriptions	[ ]	`*.mdl` *files*
Event logs	[ ]	`*.evt0` *and* `*.evt1` *files*
Engine log	[ ]	`*.log0` *and* `*.log1`
Engine traces	[ ]	`*.trc0` *and* `*.trc1`
Installation verify	[ ]	`ecsconftest.log` *file*
	[ ]	*Type:* **`ecsconftest > filename`**
Other (specify)	[ ]	_____
		_____

Feedback

History *If you believe that this problem is related to other problems which you have previously reported, please provide references to those reports if possible. Explain how you believe the previous problems to be related to this problem (attach pages as required).*

Your Ref:	_____	Date	_____	HP Ref:	_____
Your Ref:	_____	Date	_____	HP Ref:	_____
Your Ref:	_____	Date	_____	HP Ref:	_____

_____

_____

_____

Recommendation: If you believe that you know what needs to be done to prevent this problem from recurring, please supply your recommendation. HP also welcomes suggestions for improving the product.

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 annotaton 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 correlation circuit.*

CMIP *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 Distributed Management Platform, the platform which provides the infrastructure for implementing OSI-based management solutions.

DM *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 Event Correlation Description Language (ECDL).*

ECS *See Event Correlation Services (ECS).*

ECS circuit *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 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 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 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 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 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 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 Open Systems Interconnection (OSI).*

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.

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

Zulu *See Universal Coordinated Time (UTC).*

Index

Index

Index